Reprints in Theory and Applications of Categories, No. 22, 2012.

CATEGORY THEORY FOR COMPUTING SCIENCE

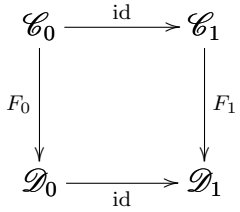
MICHAEL BARR AND CHARLES WELLS

Transmitted by Richard Blute, Robert Rosebrugh and Alex Simpson. Reprint published on 2012-09-19, revised 2013-09-22.
2010 Mathematics Subject Classification: 18-01,68-01.
Key words and phrases: Category theory, computing science.

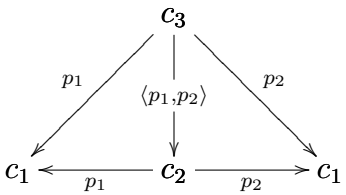
Preface to the TAC reprint (Revised 2013-09-22)

This is a reprint of the final version, published by the Centre de Recherche Mathématique at the Université de Montréal. We are now aware of the following errors. If any others are reported, we will post corrections at `ftp.math.mcgill.ca/barr/pdffiles/ctcserr.pdf` and at `http://www.abstractmath.org/CTCS/ctcserr.pdf`.

There is a diagram missing in the solution to problem 4.1.4. The problem is on page 98 and the solution on page 438. The diagram is



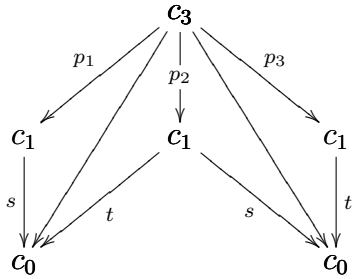
On page 302, the second of the first line of diagrams near the middle of the page should be



The left bottom p_1 goes in the wrong direction and there is a spurious 0.

Dan Synek has pointed out that the definition of cone is not really coherent because it depends on an ambient category and there is none. Accordingly, we define a cone as a rooted graph in which there is exactly one arrow from the root to each other node (along with, in general, other arrows). A cone in a category is a limit cone if the root is the limit in the usual sense if the arrows out of the root are the transition arrows of a limit of the remaining nodes and arrows of the cone. A model of the sketch is required to take each cone to a limit cone.

He also pointed out that in the description of the sketch for categories, three arrows have to be added to the underlying graph of the sketch, all appearing in display in the middle of page 303. The first one, on the left hand diagram of that display, is the unlabeled arrow $c_2 \longrightarrow c_0$ ("the object in the middle of a composite"). The other two are arrows $c_3 \longrightarrow c_0$ so that the right hand diagram of the same display becomes:



Category Theory for Computing Science

Michael Barr
Charles Wells

Category Theory for Computing Science

Michael Barr
Department of Mathematics and Statistics
McGill University

Charles Wells
Department of Mathematics
Case Western Reserve University

For
Becky, Adam and Joe
and
Matt and Peter

Contents

Preface **xi**

1 Preliminaries **1**
1.1 Sets 1
1.2 Functions 3
1.3 Graphs 8
1.4 Homomorphisms of graphs 11

2 Categories **15**
2.1 Basic definitions 15
2.2 Functional programming languages as categories 20
2.3 Mathematical structures as categories 23
2.4 Categories of sets with structure 27
2.5 Categories of algebraic structures 32
2.6 Constructions on categories 35
2.7 Properties of objects and arrows in a category 40
2.8 Monomorphisms and subobjects 47
2.9 Other types of arrow 53
2.10 Factorization systems 58

3 Functors **65**
3.1 Functors 65
3.2 Actions 74
3.3 Types of functors 80
3.4 Equivalences 84
3.5 Quotient categories 88

4 Diagrams, naturality and sketches **93**
4.1 Diagrams 93
4.2 Natural transformations 101
4.3 Natural transformations between functors 109
4.4 The Godement calculus of natural transformations 117
4.5 The Yoneda Lemma and universal elements 121
4.6 Linear sketches (graphs with diagrams) 127
4.7 Linear sketches with constants: initial term models 133
4.8 2-categories 140

5 Products and sums **153**
5.1 The product of two objects in a category 153
5.2 Notation for and properties of products 157
5.3 Finite products 168
5.4 Sums 178
5.5 Natural numbers objects 182
5.6 Deduction systems as categories 186
5.7 Distributive categories 188

6 Cartesian closed categories **195**
6.1 Cartesian closed categories 195
6.2 Properties of cartesian closed categories 202
6.3 Typed λ-calculus 208
6.4 λ-calculus to category and back 210
6.5 Arrows vs. terms 212
6.6 Fixed points in cartesian closed categories 215

7 Finite product sketches **219**
7.1 Finite product sketches 220
7.2 The sketch for semigroups 225
7.3 Notation for FP sketches 231
7.4 Arrows between models of FP sketches 234
7.5 The theory of an FP sketch 237
7.6 Initial term models for FP sketches 239
7.7 Signatures and FP sketches 245

8 Finite discrete sketches **251**
8.1 Sketches with sums 251
8.2 The sketch for fields 254
8.3 Term algebras for FD sketches 257

9 Limits and colimits **265**
9.1 Equalizers 265
9.2 The general concept of limit 268
9.3 Pullbacks 273
9.4 Coequalizers 277
9.5 Cocones 280
9.6 More about sums 285
9.7 Unification as coequalizer 289
9.8 Properties of factorization systems 294

10 More about sketches **299**
10.1 Finite limit sketches 299
10.2 Initial term models of FL sketches 304
10.3 The theory of an FL sketch 307
10.4 General definition of sketch 309

11 The category of sketches **313**
11.1 Homomorphisms of sketches 313
11.2 Parametrized data types as pushouts 315
11.3 The model category functor 320

12 Fibrations **327**
12.1 Fibrations 327
12.2 The Grothendieck construction 332
12.3 An equivalence of categories 338
12.4 Wreath products 341

13 Adjoints **347**
13.1 Free monoids 347
13.2 Adjoints 350
13.3 Further topics on adjoints 356
13.4 Locally cartesian closed categories 360

14 Algebras for endofunctors **363**
14.1 Fixed points for a functor 363
14.2 Recursive categories 368
14.3 Triples 372
14.4 Factorizations of a triple 374
14.5 Scott domains 376

15 Toposes **383**
15.1 Definition of topos 384
15.2 Properties of toposes 387
15.3 Is a two-element poset complete? 391
15.4 Presheaves 393
15.5 Sheaves 395
15.6 Fuzzy sets 400
15.7 External functors 403
15.8 The realizability topos 408

16 Categories with monoidal structure **413**
16.1 Closed monoidal categories 413
16.2 Properties of $A \multimap C$ 417
16.3 $*$-autonomous categories 422
16.4 The Chu construction 424

Solutions to the exercises **431**
Solutions for Chapter 1 431
Solutions for Chapter 2 433
Solutions for Chapter 3 442
Solutions for Chapter 4 449
Solutions for Chapter 5 460
Solutions for Chapter 6 467
Solutions for Chapter 7 471
Solutions for Chapter 8 475
Solutions for Chapter 9 477
Solutions for Chapter 10 488
Solutions for Chapter 11 489
Solutions for Chapter 12 491
Solutions for Chapter 13 494
Solutions for Chapter 14 498
Solutions for Chapter 15 506
Solutions for Chapter 16 510

Bibliography **517**

Index **531**

Preface

This book is a textbook in basic category theory, written specifically to be read by researchers and students in computing science. We expound the constructions we feel are basic to category theory in the context of examples and applications to computing science. Some categorical ideas and constructions are already used heavily in computing science and we describe many of these uses. Other ideas, in particular the concept of adjoint, have not appeared as widely in the computing science literature. We give here an elementary exposition of those ideas we believe to be basic categorical tools, with pointers to possible applications when we are aware of them.

In addition, this text advocates a specific idea: the use of sketches as a systematic way to turn finite descriptions into mathematical objects. This aspect of the book gives it a particular point of view. We have, however, taken pains to keep most of the material on sketches in separate sections. It is not necessary to read to learn most of the topics covered by the book.

As a way of showing how you can use categorical constructions in the context of computing science, we describe several examples of modeling linguistic or computational phenomena categorically. These are not intended as the final word on how categories should be used in computing science; indeed, they hardly constitute the initial word on how to do that! We are mathematicians, and it is for those in computing science, not us, to determine which is the best model for a given application.

The emphasis in this book is on understanding the concepts we have introduced, rather than on giving formal proofs of the theorems. We include proofs of theorems only if they are enlightening in their own right. We have attempted to point the reader to the literature for proofs and further development for each topic.

In line with our emphasis on understanding, we frequently recommend one or another way of thinking about a concept. It is typical of most of the useful concepts in mathematics that there is more than one way of perceiving or understanding them. It is simply not true that everything about a mathematical concept is contained in its definition. Of course it is true that in some sense all the *theorems* are inherent in its definition, but not what makes it useful to mathematicians or to scientists who use mathematics. We believe that the more ways you have of perceiving an idea, the more likely you are to recognize situations in your own work where the idea is useful.

We have acted on the belief just outlined with many sentences beginning with phrases such as 'This concept may be thought of as ...'. We have been warned that doing this may present difficulties for a nonmathematician who has only just mastered *one* way of thinking about something, but we feel it is part of learning about a mathematical topic to understand the contextual associations it has for those who use it.

About categories

Categories originally arose in mathematics out of the need of a formalism to describe the passage from one type of mathematical structure to another. A category in this way represents a kind of mathematics, and may be described as **category as mathematical workspace**.

A category is also a mathematical structure. As such, it is a common generalization of both ordered sets and monoids (the latter are a simple type of algebraic structure that include transition systems as examples), and questions motivated by those topics often have interesting answers for categories. This is **category as mathematical structure**.

A third point of view is emphasized in this book. A category can be seen as a structure that formalizes a mathematician's description of a type of structure. This is the role of **category as theory**. Formal descriptions in mathematical logic are traditionally given as formal languages with rules for forming terms, axioms and equations. Algebraists long ago invented a formalism based on tuples, the method of signatures and equations, to describe algebraic structures. In this book, we advocate categories in their role as formal theories as being in many ways superior to the others just mentioned. Continuing along the same path, we advocate sketches as finite specifications for the theories.

Changes in the second and third editions

The second edition contained new examples and exercises, many new items in the bibliography, and new sections or chapters on 2-categories, distributive categories, monoidal categories and $*$-autonomous categories. The third edition contains some new examples and exercises as well as new material on factorization, final algebras and Chu objects. In contrast to the second edition, this third edition contains in the printed text all the chapters as well as answers to all the exercises.

Lists of errors and corrections to each of the three editions is available by anonymous FTP from `ftp.math.mcgill.ca/pub/barr`, as well as by web browser at `http://www.cwru.edu/artsci/math/wells/pub/papers.html`.

Topics

Chapter 1 contains preliminary material on graphs, sets and functions. The reader who has taken a discrete mathematics course may wish to skip this chapter. However, some fine points concerning sets and functions are discussed that may be worth looking at.

Chapter 2 introduces categories and gives many examples. We also give certain simple constructions on categories and describe elementary properties of objects and arrows.

Chapter 3 introduces functors, which are the mappings that preserve the structure of categories. We make certain constructions here that will be needed in later chapters.

Chapter 4 deals with three related topics: diagrams, natural transformations and sketches. Probably the first thing noncategorists notice about category theory is the proliferation of diagrams: here we begin the heavy use of diagrams in this book. We discuss representable functors, universal objects and the Yoneda embedding, which are fundamental tools for the categorist. We also introduce 2-categories in this chapter, as well as a very weak version of sketch called a linear sketch.

Chapter 5 introduces products and sums. This allows one to use categories, in their role as theories, to specify functions of several variables and to specify alternatives. In programming languages these appear as record structures and variant records.

Chapter 6 is an introduction to cartesian closed categories, which have been a major source of interest to computer scientists because they are equivalent in theoretical power to typed lambda calculus. In this chapter, we outline briefly the process of translating between typed lambda calculus and cartesian closed categories. Normally, in learning a new language, one should plunge right in speaking it instead of translating. However, it may be helpful for the suspicious reader to see that translation is possible. We outline two translation processes in this book: the one mentioned here and another in Chapter 7. Except for those two places, category theory is everywhere presented in its own terms.

Chapter 7 introduces finite product sketches, which have the expressive power of multisorted universal algebra. These sketches provide a formalism for universal algebra that provide a natural definition of models in categories other than the category of sets, and being based on graphs, they also incorporated multisortedness into the definition in an intrinsic way. In this chapter, a method is given for translating between finite product sketches and the formalism of signatures and equations used in traditional universal algebra. More expressive types of sketches are described in Chapters 8, 10 and 11.

Chapter 8 introduces finite discrete sketches, which have more expressive power than universal algebra, in that they allow one to express alternatives.

Chapter 9 introduces the general concepts of limit and colimit of which the constructions in Chapter 5 were a special case. These are basic constructions in category theory that allow the formation of equationally defined subtypes and quotients. We describe unification in terms of coequalizers of free models of a certain kind of theory (free theory).

Chapter 10 describes finite limite sketches, which are more powerful than universal algebra in a different way from the finite discrete sketches of Chapter 8, allowing partial operations, but only those whose domain can be described equationally. These sketches have somewhat more expressive power than universal Horn theories. We also describe briefly the most general types of sketches.

Chapter 11 introduces mappings between sketches, which are applied to the description of parametrized data types. In this chapter, sketches are also shown to be institutions in the sense of Goguen and Burstall.

Chapter 12 describes fibrations and the Grothendieck construction, which have applications to programming language semantics. We also consider wreath products, a type of fibration which has been used in the study of automata.

Chapter 13 discusses the concept of adjointness, which is one of the grand unifying ideas of category theory. It is closely related to two other ideas: representable functors and the Yoneda Lemma. Many of the constructions in preceding chapters are examples of adjoints.

We discuss representable functors, universal objects and the Yoneda Lemma in Chapter 4, but we have deliberately postponed adjoints until we have several examples of them in applications. The concept of adjoint appears difficult and unmotivated if introduced too early. Nevertheless, with some exceptions, Chapter 13 can be read after having finished Section 4.5. (This is discussed in more detail in the introduction to Chapter 13.)

Chapter 14 contains a miscellany of topics centered around the idea of the algebra for a functor. We use this to define fixed points for a functor, to introduce the notion of a triple (monad), and to develop the Smyth-Plotkin technique for constructing Scott domains.

Chapter 15 introduces toposes. A topos is a kind of generalized set theory in which the logic is intuitionistic instead of classical. Toposes (or computable subcategories thereof) have often been thought the correct arena for programming language semantics. Categories of fuzzy sets are recognized as almost toposes, and modest sets, which are thought by many to be the best semantic model of polymorphic lambda calculus, live in a specific topos.

Chapter 16 introduces monoidal categories, $*$-autonomous categories and the Chu construction. The latter form a model of linear logic.

Most sections have exercises which provide additional examples of the concepts and pursue certain topics further. Many exercises can be solved by carefully keeping track of the definitions of the terms involved. A few exercises are harder and are marked with a dagger. Some of those so marked require a certain amount of ingenuity (although we do not expect the reader to agree in every case with our judgment on this!). Others require familiarity with some particular type of mathematical structure. For example, although we define monoids in the text, a problem asking for an example of a monoid with certain behavior can be difficult for someone who has never thought about them before reading this book.

This third edition contains solutions to all the exercises. The solutions to the easy exercises, especially in the early chapters, go into considerable detail. The solutions to the harder exercises often omit routine verifications.

Other categorical literature

Nearly all of the topics in category theory in this book are developed further in the authors' monograph [Barr and Wells, 1985]. Indeed, the present text could be used as an introduction to that monograph. Most of the topics, except sketches, are also developed further in [Mac Lane, 1971], [McLarty, 1992], [Freyd and Scedrov, 1990] and [Borceux, 1994]. Other texts specifically concerning applications to computing science include [Asperti and Longo, 1991], [Crole, 1994], [Gunter, 1992], [Manes and Arbib, 1986], [Pierce, 1991], [Rydeheard and Burstall, 1988] and [Walters, 1991]. Various aspects of the close relationship between logic and categories (in their role as theories) are treated in [Makkai and Reyes, 1977], [Lambek and Scott, 1986], [Bell, 1988] and [Adámek and Rosický, 1994]. Recent collections of papers in computer science which have many applications of category theory are [Pitt *et al.*, 1986], [Pitt, Poigné and Rydeheard, 1987], [Ehrig *et al.*, 1988], [Main *et al.*, 1988], [Gray and Scedrov, 1989], [Pitt *et al.*, 1989], [Pitt *et al.*, 1991], [Fourman, Johnstone and Pitts, 1992], [Seely, 1992], [Pitt, Rydeheard and Johnstone, 1995] and [Moggi and Rosolini, 1997]. The reader may find useful the discussions of the uses of category theory in computing science in [Goguen, 1991], [Fokkinga, 1992] and in the tutorials in [Pitt *et al.*, 1986].

Since this is an expository text, we make no effort to describe the history of the concepts we introduce or to discover the earliest references to theorems we state. In no case does our statement of a theorem constitute a claim that the theorem is original with us. In the few cases where it is original, we have announced the theorem in a separate research article.

We do give an extensive bibliography; however, the main criteria for inclusion of a work in the bibliography are its utility and availability, not the creation of a historical record.

Prerequisites

This text assumes some familiarity with abstract mathematical thinking, and some specific knowledge of the basic language of mathematics and computing science of the sort taught in an introductory discrete mathematics course.

Terminology

In most scientific disciplines, notation and terminology are standardized, often by an international nomenclature committee. (Would you recognize Einstein's equation if it said $p = HU^2$?) We must warn the nonmathematician reader that such is not the case in mathematics. There is no standardization body and terminology and notation are individual and often idiosyncratic. We will introduce and stick to a fixed notation in this book, but any reader who looks in another source must expect to find different notation and even different names for the same concept – or what is worse, the same name used for a different concept. We have tried to give warnings when this happens, with the terminology at least.

Acknowledgments for the first edition

In the preparation of this book, the first author was assisted by a grant from the NSERC of Canada and the second by NSF grant CCR-8702425. The authors would also like to thank McGill University and Case Western Reserve University, respectively, for sabbatical leaves, and the University of Pennsylvania for a very congenial setting in which to spend those leaves.

We learned much from discussions with Adam Barr, Robin Cockett, George Ernst, Tony Hoare, Colin McLarty, Pribhakar Mateti, William F. Ogden, Robert Paré and John Power. Bob Harper, Tony Hoare and an anonymous referee made many helpful corrections and suggestions. We would especially like to thank Benjamin Pierce, who has read the book from beginning to end and found scores of errors, typographical and otherwise.

Acknowledgments for the second edition

We are grateful to the many readers who reported errors and obscurities in the first edition. They include Nils Andersen, David Benson, Anthony Bucci, Anders Gammelgaard, Stephen J. Bevan, Andrew Malton, Jean-Pierre Marquis, Frank Piessens, Richard Rarick, Paul Taylor, Todd Turnidge, Nico Verwer, Al Vilcius, Jodelle Wuertzer, and Han Yan. We are also grateful to Barbara Beeton for help with fonts.

Michael Barr
Department of Mathematics
and Statistics
McGill University
805 Sherbrooke St. W.
Montréal, Québec
Canada H3P 1S4
barr@barrs.org

Charles Wells
Department of Mathematics
Case Western Reserve University
10900 Euclid Ave
Cleveland, OH 44106-7058
USA
charles@freude.com

1
Preliminaries

This chapter is concerned with some preliminary ideas needed for the introduction to categories that begins in Chapter 2. The main topics are (i) an introduction to our notation and terminology for sets and functions and a discussion of some fine points that can cause trouble if not addressed early, and (ii) a discussion of graphs, by which we mean a specific type of directed graph. Graphs are a basis for the definition of category and an essential part of the definition of both commutative diagrams and sketches.

1.1 Sets

The concept of **set** is usually taken as known in mathematics. Instead of attempting a definition, we will give a *specification* for sets and another one for functions that is adequate for our purposes.

1.1.1 Specification A **set** is a mathematical entity that is distinct from, but completely determined by, its elements (if any). For every entity x and set S, the statement $x \in S$, read 'x is an element of S', is a sentence that is either true or false.

A finite set can be defined by listing its elements within braces; for example, $\{1, 3, 5\}$ is a set completely determined by the fact that its only elements are the numbers 1, 3 and 5. In particular, the set $\{\ \}$ with no elements is called the **empty set** and is denoted $\emptyset$.

The **setbuilder notation** defines a set as the collection of entities that satisfy a predicate; if x is a variable ranging over a specific type of data and $P(x)$ is a predicate about that type of data, then the notation $\{x \mid P(x)\}$ denotes the set of all things that have the same type as x about which $P(x)$ is true. Thus, if x is a variable of type real, $\{x \mid x > 7\}$ is the set of real numbers greater than 7. The set $\{x \mid P(x)\}$ is called the **extension** of the predicate P.

1.1.2 Notation We denote the set of natural numbers (nonnegative integers) by $\mathbf{N}$, the set of all integers by $\mathbf{Z}$, the set of all rational numbers by $\mathbf{Q}$, and the set of all real numbers by $\mathbf{R}$.

1.1.3 Russell's paradox The setbuilder notation (which implicitly supposes that every predicate determines a set) has a bug occasioned by **Russell's paradox**, which uses setbuilder notation to define something that cannot be a set:

$$\{S \mid S \text{ is a set and } S \notin S\}$$

This purports to be the set of all those sets that are not elements of themselves. If this were indeed a set T, then $T \in T$ implies by definition that $T \notin T$, whereas $T \notin T$ implies by definition that $T \in T$. This contradiction shows that there is no such set T.

A simple way to avoid this paradox is to restrict x to range over a particular type of data (such as one of the various number systems – real, integers, etc.) that already forms a set. This prophylaxis guarantees safe sets.

> If you find it difficult to comprehend how a set could be an element of itself, rest assured that most approaches to set theory rule that out. (But see [Devlin, 1993].) Following those approaches, the (impossible) set T consists of *all* sets, so that we now know that there is no set whose elements are all the sets that exist – there is no 'set of all sets'.

1.1.4 Definition If S and T are sets, the **cartesian product** of S and T is the set $S \times T$ of ordered pairs (s, t) with $s \in S$ and $t \in T$. Thus in setbuilder notation, $S \times T = \{(s, t) \mid s \in S \text{ and } t \in T\}$. Observe that $T \times S = \{(t, s) \mid s \in S \text{ and } t \in T\}$, and so is not the same set as $S \times T$ unless $S = T$.

The ordered pair (s, t) is determined uniquely by the fact that its first coordinate is s and its second coordinate is t. That is essentially a specification of ordered pairs. The formal categorical definition of product in Section 5.1 is based on this.

More generally, an **ordered n-tuple** is a sequence $(a_1, \ldots, a_n)$ determined uniquely by the fact that for $i = 1, \ldots, n$, the ith coordinate of $(a_1, \ldots, a_n)$ is a_i. Then the cartesian product $S_1 \times S_2 \times \cdots \times S_n$ is the set of all n-tuples $(a_1, \ldots, a_n)$ with $a_i \in S_i$ for $i = 1, \ldots, n$.

1.1.5 Definition A **relation** α from a set S to a set T is a subset of $S \times T$. Any such subset is a relation from S to T.

Extreme examples of relations from S to T are the empty set and the set $S \times T$. Another useful example is the **diagonal relation** Δ_S (often written simply Δ) from S to S for any set S; by definition,

$$\Delta_S = \{(x, x) \mid x \in S\}$$

The usual order relations such as '$<$' and '$\leq$' on $\mathbf{R}$ (and other sets of numbers) are examples of relations; thus '$<$' in this sense is the set

$$\{(r, s) \mid \text{ There is a positive number } t \text{ such that } r + t = s\}$$

As this last example illustrates, if α is a relation from S to T, then we write $s\ \alpha\ t$ to mean that $(s,t) \in \alpha$. However, the usual symbol for Δ is of course '='.

Observe that if $S \neq T$, then a relation from S to T is not the same thing as a relation from T to S (because then $S \times T \neq T \times S$).

1.2 Functions

1.2.1 Specification A **function** f is a mathematical entity with the following properties:

F–1 f has a **domain** and a **codomain**, each of which must be a set.

F–2 For every element x of the domain, f has a **value** at x, which is an element of the codomain and is denoted $f(x)$.

F–3 The domain, the codomain, and the value $f(x)$ for each x in the domain are all determined completely by the function.

F–4 Conversely, the data consisting of the domain, the codomain, and the value $f(x)$ for each element x of the domain completely determine the function f.

The domain and codomain are often called the **source** and **target** of f, respectively.

The notation $f : S \longrightarrow T$ is a succinct way of saying that the function f has domain S and codomain T. It is used both as a verb, as in 'Let $f : S \longrightarrow T$', which would be read as 'Let f be a function from S to T', and as a noun, as in 'Any function $f : S \longrightarrow T$ satisfies ...', in which the expression would be read as 'f from S to T'. One also says that f is of type 'S arrow T'.

We will use the **barred arrow notation** to provide an anonymous notation for functions. For example, the function from $\mathbf{R}$ to $\mathbf{R}$ that squares its input can be denoted $x \mapsto x^2 : \mathbf{R} \longrightarrow \mathbf{R}$. The barred arrow goes from *input datum to output datum* and the straight arrow goes from *domain to codomain*. The barred arrow notation serves the same purpose as the logicians' lambda notation, $\lambda x.x^2$, which we do not use except in the discussion of λ-calculus in Chapter 6. The barred arrow notation, like the lambda notation, is used mainly for functions defined by a formula.

1.2.2 The significance of F–3 is that a function is not merely a rule, but a rule together with its domain and codomain. This is the point of view taken by category theorists concerning functions, but is not shared by all mathematicians. Thus category theorists insist on a very strong form of typing. For example, these functions

(i) $x \mapsto x^2 : \mathbf{R} \to \mathbf{R}^+$
(ii) $x \mapsto x^2 : \mathbf{R} \to \mathbf{R}$
(iii) $x \mapsto x^2 : \mathbf{R}^+ \to \mathbf{R}^+$
(iv) $x \mapsto x^2 : \mathbf{R}^+ \to \mathbf{R}$

(where $\mathbf{R}^+$ is the set of nonnegative reals) are four different functions. This distinction is not normally made in college mathematics courses, and indeed there is no reason to make the distinction there, but it turns out to be necessary to make it in category theory and some other branches of abstract mathematics.

It can be useful to make the distinction even at an elementary level. For example, every set S has an **identity function** $\mathrm{id}_S : S \to S$ for which $\mathrm{id}_S(x) = x$ for all $x \in S$. If S is a subset of a set T, then there is an **inclusion function** $i : S \to T$ for which $i(x) = x$ for all $x \in S$. The functions id_S and i are different functions because they have different codomains, even though their value at each element of their (common) domain is the same.

1.2.3 Definition The **graph of a function** $f : S \to T$ is the set of ordered pairs: $\{(x, f(x)) \mid x \in S\}$.

Thus the graph of a function from S to T is a relation from S to T as defined in Definition 1.1.5. However, not any relation will do; it must have the **functional property** that for all $s \in S$, there is one and only one $t \in T$ such that (s, t) is in the graph.

Many texts, but not this one, define a function to be a relation with the functional property. That definition is not equivalent to Definition 1.2.3; a relation from S to T with the functional property determines S (it is the set of all first coordinates of ordered pairs of the relation) but not T, so that a function defined in that way does not determine its codomain. Some writers who define a function to be a relation with the functional property use the word **mapping** for what we call a function (where the domain and codomain are part of the definition). In this book (as in most books) we use the word "mapping" or "map" synonymously with "function".

1.2.4 Definition The **image** of a function (also called its range) is its set of values; that is, the image of $f : S \to T$ is $\{t \in T \mid \exists s \in S$ for which $f(s) = t\}$. The image of each of the squaring functions mentioned in 1.2.2 is of course the set of nonnegative reals.

The metaphor behind the names "map" and "image" reveals a way of thinking that is basic in modern mathematics: A function $f : S \to T$ is thought of as a *map in the space* T (in the sense of cartography) of its domain S. Thus a map of New York City ($= S$) is a collection of symbols on a piece T of paper together with the (partially implicit) information about which points on the

paper T correspond to actual locations in the city S; this is the function that takes a point in the city to a point on the sheet of paper. The *image* is the set of symbols and their arrangement on the page, forgetting what in the city they correspond to.

The fact that the locations on the paper correspond to the locations in the city comes from the fact that the map is (approximately) *shape-preserving*. Most functions actually used in mathematics preserve some kind of structure.

1.2.5 Definition A function $f : S \longrightarrow T$ is **injective** if whenever $s \neq s'$ in S, then $f(s) \neq f(s')$ in T.

Another name for injective is **one to one**. The identity and inclusion functions described previously are injective. The function $x \mapsto x^2 : \mathbf{R} \longrightarrow \mathbf{R}$ is not injective since it takes 2 and -2 to the same value, namely 4. On the other hand, $x \mapsto x^3$ is injective.

There is a unique function $e : \emptyset \longrightarrow T$ for any set T. It has no values. It is vacuously injective.

Do not confuse the definition of injective with the property that all functions have that if $s = s'$ then $f(s) = f(s')$. Another way of saying that a function is injective is via the contrapositive of the definition of injective: if $f(s) = f(s')$, then $s = s'$.

1.2.6 Definition A function $f : S \longrightarrow T$ is **surjective** if its image is T.

The identity function on a set is surjective, but no other inclusion function is surjective.

Observe that the definition of surjective depends on the specified codomain; for example, of the four squaring functions listed in 1.2.2, only (i) and (iii) are surjective. A surjective function is often said to be onto.

A function is **bijective** if it is injective and surjective. A bijective function is also called a **one to one correspondence**.

1.2.7 Functions and cartesian products If S and T are sets, the cartesian product $S \times T$ is equipped with two **coordinate** or **projection** functions $\mathrm{proj}_1 : S \times T \longrightarrow S$ and $\mathrm{proj}_2 : S \times T \longrightarrow T$. The coordinate functions are surjective if S and T are both nonempty. Coordinate functions for products of more than two sets are defined analogously.

There are two additional notational devices connected with the cartesian product.

1.2.8 Definition If X, S and T are sets and $f : X \longrightarrow S$ and $g : X \longrightarrow T$ are functions, then the function $\langle f, g\rangle : X \longrightarrow S \times T$ is defined by $\langle f, g\rangle(x) = (f(x), g(x))$ for all $x \in X$.

1.2.9 Definition If X, Y, S and T are sets and $f : X \to S$, $g : Y \to T$ are functions, then $f \times g : X \times Y \to S \times T$ is the function defined by $(f \times g)(x, y) = (f(x), g(y))$. It is called the **cartesian product** of the functions f and g.

These functions are discussed further in Chapter 5.

1.2.10 Definition If $f : S \to T$ and $g : T \to U$, then the **composite function** $g \circ f : S \to U$ is defined to be the unique function with domain S and codomain U for which $(g \circ f)(x) = g(f(x))$ for all $x \in S$. In the computing science literature, $f; g$ is often used for $g \circ f$.

Category theory is based on composition as a fundamental operation in much the same way that classical set theory is based on the 'element of' or membership relation.

In categorical treatments, it is necessary to insist, as we have here, that the *codomain of f be the domain of g* for the composite $g \circ f$ to be defined. Many texts in some other branches of mathematics require only that the image of f be included in the domain of g.

1.2.11 Definition If $f : S \to T$ and $A \subseteq S$, then the **restriction** of f to A is the composite $f \circ i$, where $i : A \to S$ is the inclusion function. Thus the squaring function in 1.2.2(iii) is the restriction to $\mathbf{R}^+$ of the squaring function in 1.2.2(i).

Similarly, if $T \subseteq B$, f is called the **corestriction** of the function $j \circ f : S \to B$ to T, where j is the inclusion of T in B. Thus, in 1.2.2, the function in (i) is the corestriction to $\mathbf{R}^+$ of the function in (ii).

1.2.12 Functions in theory and practice The concept of function can be explicitly defined in terms of its domain, codomain and graph. Precisely, a function $f : S \to T$ could be defined as an ordered triple (S, Γ, T) with the property that Γ is a subset of the cartesian product $S \times T$ with the functional property (Γ is the graph of f). Then for $x \in S$, $f(x)$ is the unique element $y \in T$ for which $(x, y) \in \Gamma$. Such a definition clearly satisfies specification 1.2.1.

The description of functions in 1.2.1 is closer to the way a mathematician thinks of a function than the definition in 1.2.12. For a mathematician, a function has a domain and a codomain, and if x is in the domain, then there is a well defined value $f(x)$ in the codomain. It is wrong to think that a function is *actually* an ordered triple as described in the preceding paragraph in the same sense that it is wrong for a programmer writing in a high level language to think of the numbers he deals with as being expressed in binary notation. The possible definition of function in the preceding paragraph is an *implementation* of the specification for function, and just as with program

specifications the expectation is that one normally works with the specification, not the implementation, in mind. We make a similar point in 5.1.1 when we discuss ordered pairs in the context of categorical products.

> In understanding the difference between a specification of something and an implementation of it, it may be instructive to read the discussion of this point in [Halmos, 1960], Section 6, who gives the definition of an ordered pair. The usual definition is rather unnatural and serves only to demonstrate that a construction with the required property exists. Specifications are also discussed in [Wells, 1995].

1.2.13 Definition Let S and T be sets, and let $\mathrm{Hom}(S,T)$ denote the set of all functions with domain S and codomain T. (This fits with the standard notation we introduce in Chapter 2.) Let $f : T \longrightarrow V$ be a function. The function

$$\mathrm{Hom}(S,f) : \mathrm{Hom}(S,T) \longrightarrow \mathrm{Hom}(S,V)$$

is defined by

$$\mathrm{Hom}(S,f)(g) = f \circ g$$

$\mathrm{Hom}(S,f)$ is an example of a **hom function**. Note that $\mathrm{Hom}(S,x)$ is overloaded notation: when x is a set, $\mathrm{Hom}(S,x)$ is a set of functions, but when x is a function, so is $\mathrm{Hom}(S,x)$.

As an example of how one works with Hom functions, we show that if S is not the empty set, then f is injective if and only if $\mathrm{Hom}(S,f)$ is injective.

Suppose f is injective and that $\mathrm{Hom}(S,f)(g) = \mathrm{Hom}(S,f)(h)$. Then $f \circ g = f \circ h$. Let x be any element of S. Then $f(g(x)) = f(h(x))$ and f is injective, so $g(x) = h(x)$. Since x is arbitrary, $g = h$. Conversely, suppose f is not injective. Then for some $t, u \in T$ with $t \neq u$, $f(t) = f(u)$. Define functions $g : S \longrightarrow T$ and $h : S \longrightarrow T$ to be the constant functions with values t and u respectively. Because S is nonempty, $g \neq h$. For any $x \in S$, $f(g(x)) = f(t) = f(u) = f(h(x))$, so

$$\mathrm{Hom}(S,f)(g) = f \circ g = f \circ h = \mathrm{Hom}(S,f)(h)$$

Hence $\mathrm{Hom}(S,f)$ is not injective.

1.2.14 Exercises

> Most introductory texts in discrete mathematics provide dozens of exercises concerning sets, functions and their properties, and operations such as union, intersection, and so on. We regard our discussion as establishing notation, not as providing a detailed introduction to these concepts, and so do not give such exercises here. The exercises we do provide here allow a preliminary look at some categorical constructions that will appear in detail later in the book.

1. In the notation of 1.2.13, let $h : W \longrightarrow S$ be a function and define a function $\mathrm{Hom}(h,T) : \mathrm{Hom}(S,T) \longrightarrow \mathrm{Hom}(W,T)$ by $\mathrm{Hom}(h,T)(g) = g \circ h$. Show that if T has at least two elements, then h is *surjective* if and only if $\mathrm{Hom}(h,T)$ is *injective*.

2. a. Using the notation of 1.2.13, show that the mapping that takes a pair $(f : X \longrightarrow S, g : X \longrightarrow T)$ of functions to the function $\langle f,g\rangle : X \longrightarrow S \times T$ defined in Definition 1.2.8 is a bijection from $\mathrm{Hom}(X,S) \times \mathrm{Hom}(X,T)$ to $\mathrm{Hom}(X, S \times T)$.

b. If you set $X = S \times T$ in (a), what does $\mathrm{id}_{S\times T}$ correspond to under the bijection?

3. Let S and T be two disjoint sets, and let V be a set.

a. Let $\phi : \mathrm{Hom}(S,V) \times \mathrm{Hom}(T,V) \longrightarrow \mathrm{Hom}(S \cup T, V)$ be the mapping that takes a pair $(f : S \longrightarrow V,\ g : T \longrightarrow V)$ to the function $\langle f|g\rangle : S \cup T \longrightarrow V$ defined by

$$\langle f|g\rangle(x) = \begin{cases} f(x) & \text{if } x \in S \\ g(x) & \text{if } x \in T \end{cases}$$

Show that ϕ is a bijection.

b. If you set $V = S \cup T$ in (a), what is $\phi(\mathrm{id}_{S\cup T})$?

4. If $\mathscr{P}(C)$ denotes the power set (set of all subsets) of C, then $\mathrm{Rel}(A,B) = \mathscr{P}(A \times B)$ denotes the set of relations from A to B. Let $\phi : \mathrm{Rel}(A,B) \longrightarrow \mathrm{Hom}(A, \mathscr{P}B)$ be defined by

$$\phi(\alpha)(a) = \{b \in B \mid (a,b) \in \alpha\}$$

a. Show that ϕ is a bijection.

b. Let $A = B$. What corresponds to Δ_A under this bijection?

c. If we let $A = \mathscr{P}B$ then $\phi^{-1} : \mathrm{Hom}(\mathscr{P}B, \mathscr{P}B) \longrightarrow \mathrm{Rel}(\mathscr{P}B, B)$. What is $\phi^{-1}(\mathrm{id}_{\mathscr{P}B})$?

1.3 Graphs

The type of graph that we discuss in this section is a specific version of directed graph, one that is well adapted to category theory, namely what is often called a directed multigraph with loops. A graph is a constituent of a sketch, which we introduce in Chapter 4, and is an essential ingredient in the definition of commutative diagram, which is the categorist's way of expressing equations. The concept of graph is also a precursor to the concept of category itself: a category is, roughly speaking, a graph in which paths can be composed.

1.3.1 Definition and notation Formally, to specify a **graph**, you must specify its **nodes** (or **objects**) and its **arrows**. Each arrow must have a specific **source** (or **domain**) node and **target** (or **codomain**) node. The notation '$f : a \longrightarrow b$' means that f is an arrow and a and b are its source and target, respectively. If the graph is small enough, it may be drawn with its nodes indicated by dots or labels and each arrow by an actual arrow drawn from its source to its target.

There may be one or more arrows – or none at all – with given nodes as source and target. Moreover, the source and target of a given arrow need not be distinct. An arrow with the same source and target node will be called an **endoarrow** or **endomorphism** of that node.

We will systematically denote the collection of nodes of a graph $\mathscr{G}$ by G_0 and the collection of arrows by G_1, and similarly with other letters ($\mathscr{H}$ has nodes H_0, $\mathscr{C}$ has nodes C_0, and so on). The nodes form the zero-dimensional part of the graph and the arrows the one-dimensional part.

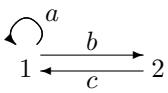
1.3.2 Example Let $G_0 = \{1, 2\}$, $G_1 = \{a, b, c\}$,

$$\text{source}(a) = \text{target}(a) = \text{source}(b) = \text{target}(c) = 1$$

and

$$\text{target}(b) = \text{source}(c) = 2$$

Then we can represent $\mathscr{G}$ as



(1.1)

1.3.3 Example The **graph of sets and functions** has all sets as nodes and all functions between sets as arrows. The source of a function is its domain, and its target is its codomain.

In this example, unlike the previous ones, the nodes do not form a set. (See 1.1.3.) This fact will not cause trouble in reading this book, and will not usually cause trouble in applications. We use some standard terminology for this distinction.

1.3.4 Definition A graph that has a *set* of nodes and arrows is a **small graph**; otherwise, it is a **large** graph.

Thus the graph of sets and functions is a large graph. More generally we refer to any kind of mathematical structure as 'small' if the collection(s) it is built on form sets, and 'large' otherwise.

Note that if $\mathscr{G}$ is a small graph, source : $G_1 \longrightarrow G_0$ and target : $G_1 \longrightarrow G_0$ are functions.

1.3.5 Definition A graph is called **discrete** if it has no arrows.

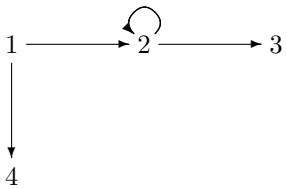
In particular, the empty graph, with no nodes and no arrows, is discrete. A small discrete graph is essentially a set; small discrete graphs and sets are usefully regarded as the same thing for most purposes.

1.3.6 Definition A graph is **finite** if the number of nodes *and* arrows is finite.

1.3.7 Example It is often convenient to picture a relation on a set as a graph. For example, let $A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$ and

$$\alpha = \{(1, 2), (2, 2), (2, 3), (1, 4)\}$$

Then α can be pictured as

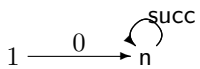


(1.2)

Of course, graphs that arise this way never have more than one arrow with the same source and target. Such graphs are called **simple graphs**.

Note that the graph of a function, as defined in 1.2.1, is a relation (see 1.1.5), and so corresponds to a graph in the sense just described. The resulting picture has an arrow from each element x of the domain to $f(x)$ so it is not the graph of the function in the sense used in calculus.

1.3.8 Example Sometimes one can represent a data structure by a graph. This graph represents the set $\mathbf{N}$ of natural numbers in terms of zero and the successor function (adding 1):



(1.3)

The name '1' for the left node is the conventional notation to require that the node denote a **singleton set**, that is, a set with exactly one element. In 4.7.6 we provide a formal mathematical meaning to the idea that this graph generates the natural numbers. Right now, this is just a graph with nodes named '1' and 'n'.

This informal idea of a graph representing a data type will become the basis of the formal theory of sketches in Chapter 4.

1.3.9 Example In the same spirit as Example 1.3.8, let us see what data type is represented by the graph

$$\mathsf{a} \underset{\mathsf{t}}{\overset{\mathsf{s}}{\rightrightarrows}} \mathsf{n} \tag{1.4}$$

The data type has a signature consisting of two objects, call them a and n, and two arrows, let us call them (temporarily) $\mathsf{s}, \mathsf{t} : \mathsf{a} \longrightarrow \mathsf{n}$. But if we interpret a as arrows, n as nodes and s and t as source and target, this is exactly what we have defined a small graph to be: two sets and two functions from one of the sets to the other. For this reason, this graph is called the graph of graphs. (See [Lawvere, 1989].)

1.3.10 Exercises

1. The graphs in this section have labeled nodes; for example, the two nodes in (1.3) are labeled '1' and 'n'. Produce a graph analogous to (1.4) that expresses the concept of 'graph with nodes labeled from a set L'.

2. Let $\mathscr{G}$ be a graph with set of nodes N and set of arrows A. Show that $\mathscr{G}$ is simple if and only if the function $\langle \text{source}, \text{target} \rangle : A \longrightarrow N \times N$ is injective. (This uses the pair notation for functions to products as described in 1.2.8.)

1.4 Homomorphisms of graphs

A homomorphism of graphs should preserve the abstract shape of the graph. A reasonable translation of this vague requirement into a precise mathematical definition is as follows.

1.4.1 Definition A **homomorphism** ϕ from a graph $\mathscr{G}$ to a graph $\mathscr{H}$, denoted $\phi : \mathscr{G} \longrightarrow \mathscr{H}$, is a pair of functions $\phi_0 : G_0 \longrightarrow H_0$ and $\phi_1 : G_1 \longrightarrow H_1$ with the property that if $u : m \longrightarrow n$ is an arrow of $\mathscr{G}$, then $\phi_1(u) : \phi_0(m) \longrightarrow \phi_0(n)$ in $\mathscr{H}$.

It is instructive to restate this definition using the source and target mappings from 1.3.9: let $\text{source}_{\mathscr{G}} : G_1 \longrightarrow G_0$ be the source map that takes an arrow (element of G_1) to its source, and similarly define $\text{target}_{\mathscr{G}}$, $\text{source}_{\mathscr{H}}$ and $\text{target}_{\mathscr{H}}$. Then the pair of maps $\phi_0 : G_0 \longrightarrow H_0$ and $\phi_1 : G_1 \longrightarrow H_1$ is a graph homomorphism if and only if

$$\text{source}_{\mathscr{H}} \circ \phi_1 = \phi_0 \circ \text{source}_{\mathscr{G}}$$

and

$$\text{target}_{\mathscr{H}} \circ \phi_1 = \phi_0 \circ \text{target}_{\mathscr{G}}$$

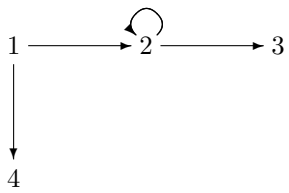
1.4.2 Notation of the form $a : B \longrightarrow C$ is overloaded in several ways. It can denote a set-theoretic function, a graph homomorphism or an arrow in a graph. In fact, all three are instances of the third since there is a large graph whose nodes are sets and arrows are functions (see 1.3.3) and another whose nodes are (small) graphs and arrows are graph homomorphisms.

Another form of overloading is that if $\phi : \mathscr{G} \longrightarrow \mathscr{H}$ is a graph homomorphism, ϕ actually stands for a pair of functions we here call $\phi_0 : G_0 \longrightarrow H_0$ and $\phi_1 : G_1 \longrightarrow H_1$. In fact, it is customary to omit the subscripts and use ϕ for all three (the graph homomorphism as well as its components ϕ_0 and ϕ_1).

This does not lead to ambiguity in practice; in reading about graphs you are nearly always aware of whether the author is talking about nodes or arrows. We will keep the subscripts in this section and drop them thereafter.

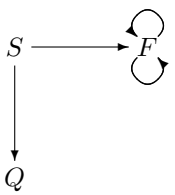
1.4.3 Example If $\mathscr{G}$ is any graph, the **identity homomorphism** $\mathrm{id}_{\mathscr{G}} : \mathscr{G} \longrightarrow \mathscr{G}$ is defined by $(\mathrm{id}_{\mathscr{G}})_0 = \mathrm{id}_{G_0}$ (the identity function on the set of nodes of $\mathscr{G}$) and $(\mathrm{id}_{\mathscr{G}})_1 = \mathrm{id}_{G_1}$.

1.4.4 Example If $\mathscr{G}$ is the graph



(1.5)

and $\mathscr{H}$ is this graph,



(1.6)

then there is a homomorphism $\phi : \mathscr{G} \longrightarrow \mathscr{H}$ for which $\phi_0(1) = S$, $\phi_0(2) = \phi_0(3) = F$ and $\phi_0(4) = Q$, and ϕ_1 takes the loop on 2 and the arrow from 2 to 3 both to the upper loop on F; what ϕ_1 does to the other two arrows is forced by the definition of homomorphism. Because there are two loops on F there are actually four possibilities for ϕ_1 on arrows (while keeping ϕ_0 fixed).

1.4.5 Example If $\mathscr{H}$ is any graph with a node n and a loop $u : n \longrightarrow n$, then there is a homomorphism from *any* graph $\mathscr{G}$ to $\mathscr{H}$ that takes every node of $\mathscr{G}$ to n and every arrow to u. This construction gives two other homomorphisms from $\mathscr{G}$ to $\mathscr{H}$ in Example 1.4.4 besides the four mentioned there. (There are still others.)

1.4.6 Example There is a homomorphism σ from Example 1.3.8 to the graph of sets that takes the node called 1 to a one-element set, which in contexts like this we will denote $\{*\}$, and that takes the node n to the set $\mathbf{N}$ of **natural numbers**. (Following the practice in computing science rather than mathematics, we start our natural numbers at 0.) The homomorphism σ takes the arrow $1 \longrightarrow \mathsf{n}$ to the function $* \mapsto 0$ that picks out the natural number 0, and $\sigma(\mathsf{succ})$, naturally, is the function that adds 1. This is an example of a model of a sketch, which we discuss in 4.7.7. This homomorphism gives a semantics for the sketch constituted by the abstract graph of 1.3.8.

1.4.7 Example The homomorphism in Example 1.4.6 is not the only homomorphism from Example 1.3.8 to sets. One can let n go to the set of integers $(\mathrm{mod}\, k)$ for a fixed k and let succ be the function that adds one $(\mathrm{mod}\, k)$ (it wraps around). You can also get other homomorphisms by taking this example (or 1.4.6) and adjoining some extra elements to the set corresponding to n which are their own successors.

1.4.8 Example Example 1.3.9 can be given a semantics in the same way as Example 1.3.8. If $\mathscr{G}$ is *any* small graph, there is a graph homomorphism ϕ from the diagram in (1.4) to the graph of sets for which $\phi_0(\mathsf{n})$ is the set of nodes of $\mathscr{G}$, $\phi_0(\mathsf{a})$ is the set of arrows, and ϕ_1 takes the arrows labeled source and target to the corresponding functions from the set of arrows of $\mathscr{G}$ to the set of nodes of $\mathscr{G}$.

Moreover, the converse is true: any homomorphism from (1.4) to the graph of sets gives a graph. The nodes of the graph are the elements of $\phi_0(\mathsf{n})$ and the arrows are the elements of $\phi_0(\mathsf{a})$. If $f \in \phi_0(\mathsf{a})$, then the source of f is $\phi_1(\mathrm{source})(f)$ and the target is $\phi_1(\mathrm{target})(f)$. Thus the graph of Example 1.3.2 corresponds to the homomorphism ϕ where $\phi_0(\mathsf{n}) = \{1, 2\}$, $\phi_0(\mathsf{a}) = \{a, b, c\}$, $\phi_1(\mathrm{source})$ is the function $a \mapsto 1, b \mapsto 1, c \mapsto 2$ and $\phi_1(\mathrm{target})$ is the function $a \mapsto 1, b \mapsto 2, c \mapsto 1$.

In short, graph homomorphisms from (1.4) to the graph of sets correspond to what we normally call graphs.

1.4.9 Notation In an expression like '$\phi_1(\mathrm{source})(f)$', ϕ_1 is a function whose value at 'source' is a function that applies to an arrow f. As this illustrates, the application operation associates to the left.

1.4.10 Exercises

1. Show that if the codomain $\mathscr{H}$ of a graph homomorphism ϕ is a simple graph, then ϕ_1 is determined uniquely by ϕ_0.

2. Show that the composite of graph homomorphisms is a graph homomorphism. Precisely: if $\phi : \mathscr{G} \to \mathscr{H}$ and $\psi : \mathscr{H} \to \mathscr{K}$ are graph homomorphisms, then define the composite $\psi \circ \phi$ by requiring that $(\psi \circ \phi)_0 = \psi_0 \circ \phi_0$ and $(\psi \circ \phi)_1 = \psi_1 \circ \phi_1$. Then prove that $\psi \circ \phi$ is a graph homomorphism.

3. Let ϕ be a homomorphism $\mathscr{G} \to \mathscr{H}$ for which both ϕ_0 and ϕ_1 are bijective. Define $\psi : \mathscr{H} \to \mathscr{G}$ by $\psi_0 = (\phi_0)^{-1}$ and $\psi_1 = (\phi_1)^{-1}$.

a. Show that ψ is a graph homomorphism from $\mathscr{H}$ to $\mathscr{G}$.

b. Using the definition of composite in the preceding exercise, show that both $\psi \circ \phi = \mathrm{id}_{\mathscr{G}}$ and $\phi \circ \psi = \mathrm{id}_{\mathscr{H}}$.

2
Categories

A category is a graph with a rule for composing arrows head to tail to give another arrow. This rule is subject to certain conditions, which we will give precisely in Section 2.1. The connection between functional programming languages and categories is described in Section 2.2. Some special types of categories are given in Section 2.3. Sections 2.4 and 2.5 are devoted to a class of examples of the kind that originally motivated category theory. The reader may wish to read through these examples rapidly rather than trying to understand every detail.

Constructions that can be made with categories are described in Section 2.6. Sections 2.7, 2.8 and 2.9 describe certain properties that an arrow of a category may have. Section 2.10 describes factorization systems, which are useful structures in many categories. This section is used only in Chapters 9 and 16.

2.1 Basic definitions

Before we define categories, we need a preliminary definition.

2.1.1 Definition Let $k > 0$. In a graph $\mathscr{G}$, a **path** from a node x to a node y of length k is a sequence $(f_1, f_2, \ldots, f_k)$ of (not necessarily distinct) arrows for which

(i) $\text{source}(f_k) = x$,
(ii) $\text{target}(f_i) = \text{source}(f_{i-1})$ for $i = 2, \ldots, k$, and
(iii) $\text{target}(f_1) = y$.

By convention, for each node x there is a unique path of length 0 from x to x that is denoted $()$. It is called the **empty path** at x.

Observe that if you draw a path as follows:

$$\cdot \xrightarrow{f_k} \cdot \xrightarrow{f_{k-1}} \cdots \xrightarrow{f_2} \cdot \xrightarrow{f_1} \cdot$$

with the arrows going from left to right, f_k will be on the left and the subscripts will go down from left to right. We do it this way for consistency with composition (compare 2.1.3, C–1).

For any arrow f, (f) is a path of length 1. As an example, in Diagram (1.3) on page 10, there is just one path of each length k from n to n, namely $()$, (succ), $(\mathsf{succ}, \mathsf{succ})$, and so on.

2.1.2 Definition The set of paths of length k in a graph $\mathscr{G}$ is denoted G_k.

In particular, G_2, which will be used in the definition of category, is the set of pairs of arrows (g, f) for which the target of f is the source of g. These are called **composable pairs** of arrows.

We have now assigned two meanings to G_0 and G_1. This will cause no conflict as G_1 refers indifferently either to the collection of arrows of $\mathscr{G}$ or to the collection of paths of length 1, which is essentially the same thing. Similarly, we use G_0 to represent either the collection of nodes of $\mathscr{G}$ or the collection of empty paths, of which there is one for each node. In each case we are using the same name for two collections that are not the same but are in a natural one to one correspondence. Compare the use of '2' to denote either the integer or the real number. As this last remark suggests, one might want to keep the two meanings of G_1 separate for purposes of implementing a graph as a data structure.

The one to one correspondences mentioned in the preceding paragraph were called 'natural'. The word is used informally here, but in fact these correspondences are natural in the technical sense (Exercise 10 of Section 4.3).

2.1.3 Categories A **category** is a graph $\mathscr{C}$ together with two functions $c : C_2 \longrightarrow C_1$ and $u : C_0 \longrightarrow C_1$ with properties C–1 through C–4 below. (Recall that C_2 is the set of paths of length 2.) The elements of C_0 are called **objects** and those of C_1 are called **arrows**. The function c is called **composition**, and if (g, f) is a composable pair, $c(g, f)$ is written $g \circ f$ and is called the **composite** of g and f. If A is an object of $\mathscr{C}$, $u(A)$ is denoted id_A, which is called the **identity** of the object A.

C–1 The source of $g \circ f$ is the source of f and the target of $g \circ f$ is the target of g.

C–2 $(h \circ g) \circ f = h \circ (g \circ f)$ whenever either side is defined.

C–3 The source and target of id_A are both A.

C–4 If $f : A \longrightarrow B$, then $f \circ \mathrm{id}_A = \mathrm{id}_B \circ f = f$.

The significance of the fact that the composite c is defined on G_2 is that $g \circ f$ is defined if and only if the source of g is the target of f. This means that composition is a function whose domain is an equationally defined subset of $G_1 \times G_1$: the equation requires that the source of g equal the target of f. It follows from this and C–1 that in C–2, one side of the equation is defined if and only if the other side is defined.

In the category theory literature, id_A is often written just A.

2.1.4 Terminology In much of the categorical literature, 'morphism', 'domain' and 'codomain' are more common than 'arrow', 'source' and 'target'. In this book we usually use the language we have just introduced of 'arrow', 'source' and 'target'. We will normally denote objects of categories by capital letters but nodes of graphs (except when we think of a category as a graph) by lower case letters. Arrows are always lower case.

In the computing science literature, the composite $g \circ f$ is sometimes written $f; g$, a notation suggested by the perception of a typed functional programming language as a category (see 2.2.1).

We have presented the concept of category as a two-sorted data structure; the sorts are the objects and the arrows. Categories are sometimes presented as one-sorted – arrows only. The objects can be recovered from the fact that C–3 and C–4 together characterize id_A (Exercise 4), so that there is a one to one correspondence between the objects and the identity arrows id_A.

2.1.5 Definition A category is **small** if its objects and arrows constitute sets; otherwise it is **large** (see the discussion of Russell's paradox, 1.1.3).

The category of sets and functions defined in 2.1.11 below is an example of a large category. Although one must in principle be wary in dealing with large classes, it is not in practice a problem; category theorists have rarely, if ever, run into set-theoretic difficulties.

2.1.6 Definition If A and B are two objects of a category $\mathscr{C}$, then the set of all arrows of $\mathscr{C}$ that have source A and target B is denoted $\mathrm{Hom}_{\mathscr{C}}(A, B)$, or just $\mathrm{Hom}(A, B)$ if the category is clear from context. This generalizes the notation of Definition 1.2.13.

Thus for each triple A, B, C of objects, composition induces a function

$$\mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) \longrightarrow \mathrm{Hom}(A, C)$$

A set of the form $\mathrm{Hom}(A, B)$ is called a **hom set**. Other common notations for $\mathrm{Hom}(A, B)$ are $\mathscr{C}(A, B)$ and $\mathscr{C}(AB)$.

2.1.7 The reference to the *set* of all arrows from A to B constitutes an assumption that they do indeed form a set. A category with the property that $\mathrm{Hom}(A, B)$ is a set for all objects A and B is called **locally small**. All categories in this book are locally small.

2.1.8 Definition For any path $(f_1, f_2, \ldots, f_n)$ in a category $\mathscr{C}$, define $f_1 \circ f_2 \circ \cdots \circ f_n$ recursively by

$$f_1 \circ f_2 \circ \cdots \circ f_n = (f_1 \circ f_2 \circ \cdots \circ f_{n-1}) \circ f_n, \quad n > 2$$

2.1.9 Proposition The general associative law. *For any path*

$$(f_1, f_2, \ldots, f_n)$$

in a category $\mathscr{C}$ *and any integer* k *with* $1 < k < n$,

$$(f_1 \circ \cdots \circ f_k) \circ (f_{k+1} \circ \cdots \circ f_n) = f_1 \circ \cdots \circ f_n$$

In other words, you can unambiguously drop the parentheses.

In this proposition, the notation $f_{k+1} \circ \cdots \circ f_n$ when $k = n - 1$ means simply f_n.

This is a standard fact for associative binary operations (see [Jacobson, 1974], Section 1.4) and can be proved in exactly the same way for categories.

2.1.10 Little categories The smallest category has no objects and (of course) no arrows. The next smallest category has one object and one arrow, which must be the identity arrow. This category may be denoted **1**. Other categories that will be occasionally referred to are the categories $\mathbf{1} + \mathbf{1}$ and **2** illustrated below (the loops are identities). In both cases the choice of the composites is forced.

$\mathbf{1} + \mathbf{1}$ **2** (2.1)

2.1.11 Categories of sets The **category of sets** is the category whose objects are sets and whose arrows are functions (see 1.2.1) with composition of functions for c and the identity function from S to S for id_S. The statement that this is a category amounts to the statements that composition of functions is associative and that each identity function $\mathrm{id}_S : S \longrightarrow S$ satisfies $f \circ \mathrm{id}_S = f$ and $\mathrm{id}_S \circ g = g$ for all f with source S and all g with target S. The fact that composition of functions is associative follows by using Definition 1.2.10 repeatedly:

$$((h \circ g) \circ f)(x) = (h \circ g)(f(x)) = h(g(f(x))) = h((g \circ f)(x)) = (h \circ (g \circ f))(x)$$

The properties of the identity function follow from the definition of the identity function (1.2.2).

In this text, the category of sets is denoted **Set**. There are other categories whose objects are sets, as follows.

2.1.12 Definition The **category of finite sets**, denoted **Fin**, is the category whose objects are finite sets and whose arrows are all the functions between finite sets.

2.1.13 Definition A **partial function** from a set S to a set T is a function with domain S_0 and codomain T, where S_0 is some subset of S. The **category Pfn of sets and partial functions** has all sets as objects and all partial functions as arrows. If $f : S \to T$ and $g : T \to V$ are partial functions with f defined on $S_0 \subseteq S$ and g defined on $T_0 \subseteq T$, the composite $g \circ f : S \to V$ is the partial function from S to V defined on the subset $\{x \in S_0 \mid f(x) \in T_0\}$ of S by the requirement $(g \circ f)(x) = g(f(x))$.

It is worth checking that composition so defined is associative. Let $f : S \to T$, $g : T \to V$ and $h : V \to W$ be partial functions with domains of definition $S_0 \subseteq S$, $T_0 \subseteq T$ and $V_0 \subseteq V$ respectively. We must show

(i) $(h \circ g) \circ f$ has the same domain of definition as $h \circ (g \circ f)$, and
(ii) For x in that common domain of definition,

$$((h \circ g) \circ f)(x) = (h \circ (g \circ f))(x)$$

For (i), the domain of definition of $(h \circ g) \circ f$ is the set of $x \in S$ such that $f(x)$ is in the domain of definition of $h \circ g$. The latter is the set of $t \in T$ such that $g(t)$ is in V_0. Thus, the domain of definition of $(h \circ g) \circ f$ is $\{x \in S_0 \mid g(f(x)) \in V_0\}$. Since $g(f(x)) = (g \circ f)(x)$, that is precisely the domain of definition of $h \circ (g \circ f)$. As for (ii), the proof is the same as for ordinary functions (Section 2.1.11).

2.1.14 Definition Let α be a relation from a set S to a set T and β a relation from T to U (see 1.1.5). The **composite** $\beta \circ \alpha$ is the relation from S to U defined as follows: If $x \in S$ and $z \in U$, $(x, z) \in \beta \circ \alpha$ if and only if there is an element $y \in T$ for which $(x, y) \in \alpha$ and $(y, z) \in \beta$. With this definition of composition, the **category Rel of sets and relations** has sets as objects and relations as arrows. The identity for a set S is the diagonal relation $\Delta_S = \{(x, x) \mid x \in A\}$.

Other examples of categories whose objects are sets are the category of sets and injective functions and the category of sets and surjective functions (Exercises 1 and 2).

Categories also arise in computing science in an intrinsic way. Three examples of this concern functional programming languages (2.2.1), automata with typed states (3.2.6) and deductive systems (Section 5.6). In Sections 2.3, 2.4 and 2.5, we discuss some of the ways in which categories arise in mathematics.

2.1.15 Exercises

1. Prove that sets (as objects) and injective functions (as arrows) form a category with functional composition as the composition operation c.

2. Do the same as Exercise 1 for sets and surjective functions.

3. Show that composition of relations (2.1.14) is associative.

4. Prove the following for any arrow $u : A \longrightarrow A$ of a category $\mathscr{C}$. It follows from these facts that C–3 and C–4 of 2.1.3 characterize the identity arrows of a category.

a. If $g \circ u = g$ for every object B of $\mathscr{C}$ and arrow $g : A \longrightarrow B$, then $u = \mathrm{id}_A$.

b. If $u \circ h = h$ for every object C of $\mathscr{C}$ and arrow $h : C \longrightarrow A$, then $u = \mathrm{id}_A$.

2.2 Functional programming languages as categories

The intense interest in category theory among researchers in computing science in recent years is due in part to the recognition that the constructions in functional programming languages make a functional programming language look very much like a category. The fact that deduction systems are essentially categories has also been useful in computing science.

In this section we describe the similarities between functional programming languages and categories informally, and discuss some of the technical issues involved in making them precise. Deduction systems are discussed in Section 5.6.

2.2.1 Functional programming languages A functional programming language may be described roughly as one that gives the user some primitive types and operations and some constructors from which one can produce more complicated types and operations.

What a pure functional programming language in this sense does *not* have is variables or assignment statements. One writes a program by applying constructors to the types, constants and functions. 'Running' a program consists of applying such an operator to constants of the input type to obtain a value.

This is what was called 'function-level programming' in Backus [1981a] and [1981b]. (See also [Williams, 1982].) Another widely held point of view is that functional programming means no assignment statements: variables may appear but are not assigned to. Most languages called functional programming languages (for example Haskell and Miranda) are functional in this sense.

We will discuss Backus-style functional programming languages here. The lambda calculus, with variables, is discussed in Chapter 6; see particularly 6.5.3.

2.2.2 The category corresponding to a functional programming language A functional programming language has:

FPL–1 Primitive data types, given in the language.

FPL–2 Constants of each type.

FPL–3 Operations, which are functions between the types.

FPL–4 Constructors, which can be applied to data types and operations to produce derived data types and operations of the language.

The language consists of the set of all operations and types derivable from the primitive data types and primitive operations. The word 'primitive' means given in the definition of the language rather than constructed by a constructor. Some authors use the word 'constructor' for the primitive operations.

2.2.3 If we make two assumptions about a functional programming language and one innocuous change, we can see directly that a functional programming language L corresponds in a canonical way to a category $C(L)$.

A–1 We must assume that there is a do-nothing operation id_A for each type A (primitive and constructed). When applied, it does nothing to the data.

A–2 We add to the language an additional type called 1, which has the property that from every type A there is a unique operation to 1. We interpret each constant c of type A as an arrow $c : 1 \longrightarrow A$. This incorporates the constants into the set of operations; they no longer appear as separate data.

A–3 We assume the language has a composition constructor: take an operation f that takes something of type A as input and produces something of type B, and another operation g that has input of type B and output of type C; then doing one after the other is a derived operation (or program) typically denoted $f; g$, which has input of type A and output of type C.

Functional programming languages generally have do-nothing operations and composition constructors, so A–1 and A–3 fit the concept as it appears in the literature. The language resulting from the change in A–2 is operationally equivalent to the original language.

Composition must be associative in the sense that, if either of $(f; g); h$ or $f; (g; h)$ is defined, then so is the other and they are the same operation. We

must also require, for $f : A \longrightarrow B$, that $f; \mathrm{id}_B$ and $\mathrm{id}_A; f$ are defined and are the same operation as f. That is, we impose the **equations** $f; \mathrm{id}_B = f$ and $\mathrm{id}_A; f = f$ on the language. Both these requirements are reasonable in that in any implementation, the two operations required to be the same would surely do the same thing.

2.2.4 Under those conditions, a functional programming language L has a category structure $C(L)$ for which:

FPC–1 The types of L are the objects of $C(L)$.

FPC–2 The operations (primitive and derived) of L are the arrows of $C(L)$.

FPC–3 The source and target of an arrow are the input and output types of the corresponding operation.

FPC–4 Composition is given by the composition constructor, written in the reverse order.

FPC–5 The identity arrows are the do-nothing operations.

The reader may wish to compare the discussion in [Pitt, 1986].

Observe that $C(L)$ is a *model* of the language, not the language itself. For example, in the category $f; \mathrm{id}_B = f$, but in the language f and $f; \mathrm{id}_B$ are different source programs. This is in contrast to the treatment of languages using context free grammars: a context free grammar generates the actual language.

2.2.5 Example As a concrete example, we will suppose we have a simple such language with three data types, `NAT` (natural numbers), `BOOLEAN` (true or false) and `CHAR` (characters). We give a description of its operations in categorical style.

(i) `NAT` should have a constant $0 : 1 \longrightarrow$ `NAT` and an operation `succ` : `NAT` $\longrightarrow$ `NAT`.

(ii) There should be two constants `true`, `false` : $1 \longrightarrow$ `BOOLEAN` and an operation $\neg$ subject to the equations $\neg \circ$ `true` = `false` and $\neg \circ$ `false` = `true`.

(iii) `CHAR` should have one constant $c : 1 \longrightarrow$ `CHAR` for each desired character c.

(iv) There should be two type conversion operations `ord` : `CHAR` $\longrightarrow$ `NAT` and `chr` : `NAT` $\longrightarrow$ `CHAR`. These are subject to the equation `chr` $\circ$ `ord` = $\mathrm{id}_{\texttt{CHAR}}$. (You can think of `chr` as operating modulo the number of characters, so that it is defined on all natural numbers.)

An example program is the arrow 'next' defined to be the composite `chr` $\circ$ `succ` $\circ$ `ord` : `CHAR` $\longrightarrow$ `CHAR`. It calculates the next character in order.

This arrow 'next' is an arrow in the category representing the language, and so is any other composite of a sequence of operations.

2.2.6 The objects of the category $C(L)$ of this language are the types `NAT`, `BOOLEAN`, `CHAR` and 1. Observe that typing is a natural part of the syntax in this approach.

The arrows of $C(L)$ consist of all programs, with two programs being identified if they must be the same because of the equations. For example, the arrow

$$\texttt{chr} \circ \texttt{succ} \circ \texttt{ord} : \texttt{CHAR} \longrightarrow \texttt{CHAR}$$

just mentioned and the arrow

$$\texttt{chr} \circ \texttt{succ} \circ \texttt{ord} \circ \texttt{chr} \circ \texttt{ord} : \texttt{CHAR} \longrightarrow \texttt{CHAR}$$

must be the same because of the equation in (iv).

Observe that `NAT` has constants $\texttt{succ} \circ \texttt{succ} \circ \ldots \circ \texttt{succ} \circ 0$ where `succ` occurs zero or more times. In the exercises, n is the constant defined by induction by $1 = \texttt{succ} \circ 0$ and $n + 1 = \texttt{succ} \circ n$.

Composition in the category is composition of programs. Note that for composition to be well defined, if two composites of primitive operations are equal, then their composites with any other program must be equal. For example, we must have

$$\texttt{ord} \circ (\texttt{chr} \circ \texttt{succ} \circ \texttt{ord}) = \texttt{ord} \circ (\texttt{chr} \circ \texttt{succ} \circ \texttt{ord} \circ \texttt{chr} \circ \texttt{ord})$$

as arrows from `CHAR` to `NAT`. This is handled systematically in 3.5.8 using the quotient construction.

This discussion is incomplete, since at this point we have no way to introduce n-ary operations for $n > 1$, nor do we have a way of specifying the flow of control. The first will be remedied in Section 5.3.14. Approaches to the second question are given in Section 5.7.6 and Section 14.2. See also [Wagner, 1986a]. Other aspects of functional programming languages are considered in 5.3.14 and 5.4.8.

2.2.7 Exercise

1. Describe how to add a predicate '`nonzero`' to the language of this section. When applied to a constant of `NAT` it should give `true` if and only if the constant is not zero.

2.3 Mathematical structures as categories

Certain common mathematical structures can be perceived as special types of categories.

2.3.1 Preordered and ordered sets If S is a set, a subset $\alpha \subseteq S \times S$ is called a **binary relation** on S. It is often convenient to write $x\alpha y$ as shorthand for $(x, y) \in \alpha$. We say that α is **reflexive** if $x\alpha x$ for all $x \in S$ and **transitive** if $x\alpha y$ and $y\alpha z$ implies $x\alpha z$ for all x, y, $z \in S$.

A set S with a reflexive, transitive relation α on it is a structure (S, α) called a **preordered set**. This structure determines a category $C(S, \alpha)$ defined as follows.

CO–1 The objects of $C(S, \alpha)$ are the elements of S.

CO–2 If $x, y \in S$ and $x\alpha y$, then $C(S, \alpha)$ has exactly one arrow from x to y, denoted (y, x). (The reader might have expected (x, y) here. This choice of notation fits better with the right-to-left composition that we use. Note that the domain of (y, x) is x and the codomain is y.)

CO–3 If x is not related by α to y there is no arrow from x to y.

The identity arrows of $C(S, \alpha)$ are those of the form (x, x); they belong to α because it is reflexive. The transitive property of α is needed to ensure the existence of the composite described in 2.1.3, so that $(z, y) \circ (y, x) = (z, x)$.

2.3.2 Example The category $C(S, \alpha)$ for $S = \{C, D\}$ and

$$\alpha = \{\langle C, C\rangle, \langle C, D\rangle, \langle D, D\rangle\}$$

is the category **2** exhibited in (2.1), page 18.

2.3.3 Ordered sets A preordered set (S, α) for which α is antisymmetric (that is $x\alpha y$ and $y\alpha x$ imply $x = y$) is called an **ordered set** or **poset** (for 'partially ordered set'). Two examples of posets are $(\mathbf{R}, \leq)$, the real numbers with the usual ordering, and for any set S, the poset $(\mathscr{P}(S), \subseteq)$, the set of subsets of S with inclusion as ordering.

It is often quite useful and suggestive to think of a category as a generalized ordered set, and we will refer to this perception to illuminate constructions we make later.

2.3.4 Semigroups A **semigroup** is a set S together with an associative binary operation $m : S \times S \longrightarrow S$. The set S is called the **underlying set** of the semigroup.

Normally for s and t in S, $m(s, t)$ is written 'st' and called 'multiplication', but note that it does not have to satisfy the commutative law; that is, we may have $st \neq ts$. A **commutative semigroup** is a semigroup whose multiplication is commutative.

It is standard practice to talk about 'the semigroup S', naming the semigroup by naming its underlying set. This will be done for other mathematical structures such as posets as well. Mathematicians call this practice 'abuse

of notation'. It is occasionally necessary to be more precise; that happens in this text in Section 13.1.

2.3.5 Powers We set $s^1 = s$ and, for any positive integer k, $s^k = ss^{k-1}$. Such powers of an element obey the laws $s^k s^n = s^{k+n}$ and $(s^k)^n = s^{kn}$ (for *positive* k and n). On the other hand, the law $(st)^k = s^k t^k$ requires commutativity.

2.3.6 Empty semigroup We specifically allow the **empty semigroup**, which consists of the empty set and the empty function from the empty set to itself. (Note that the cartesian product of the empty set with itself *is* the empty set.) This is not done in most of the non-category theory literature; it will become evident later (Section 9.5.2) why we should include the empty semigroup.

2.3.7 Definition An **identity element** e for a semigroup S is an element of S that satisfies the equation $se = es = s$ for all $s \in S$. There can be at most one identity element in a semigroup (Exercise 3).

2.3.8 Definition A **monoid** is a semigroup with an identity element. It is **commutative** if its binary operation is commutative.

It follows from the definition that a monoid is *not* allowed to be empty: it must contain an identity element. It also follows that we can extend the notation in 2.3.5 to 0 by defining x^0 to be the identity element of the monoid. The laws $s^k s^n = s^{k+n}$ and $(s^k)^n = s^{kn}$ then hold for all nonnegative k and n.

2.3.9 Examples One example of a semigroup is the set of positive integers with addition as the operation; this is a semigroup but not a monoid. If you include 0 you get a monoid.

The **Kleene closure** A^* of a set A is the set of strings (or lists) of finite length of elements of A. We write the lists in parentheses; for example (a, b, d, a) is an element of $\{a, b, c, d\}^*$. Some parts of the computer science literature call these strings instead of lists and write them this way: '*abda*'. A^* includes the empty list () and for each element $a \in A$ the list (a) of length one.

The operation of concatenation makes the Kleene closure a monoid $F(A)$, called the **free monoid** determined by A. The empty list is the identity element. We write concatenation as juxtaposition; thus

$$(a, b, d, a)(c, a, b) = (a, b, d, a, c, a, b)$$

Note that the underlying set of the free monoid is A^*, not A. In the literature, A is usually assumed finite, but the Kleene closure is defined for any set A. The elements of A^* are lists of *finite* length in any case. When A is nonempty, A^* is an infinite set.

The concept of freeness is a general concept applied to many kinds of structures. It is treated systematically in Chapter 13.

2.3.10 Definition A **submonoid** of a monoid M is a subset S of M with the properties:

SM–1 The identity element of M is in S.

SM–2 If $m, n \in S$ then $mn \in S$. (One says that S is **closed** under the operation.)

2.3.11 Examples The natural numbers with addition form a submonoid of the integers with addition. For another example, consider the integers with multiplication as the operation, so that 1 is the identity element. Again the natural numbers form a submonoid, and so does the set of *positive* natural numbers, since the product of two positive numbers is another one. Finally, the singleton set $\{0\}$ is a subset of the integers that is closed under multiplication, and it is a monoid, but it is not a submonoid of the integers on multiplication because it does not contain the identity element 1.

2.3.12 Monoid as category A monoid M determines a category $C(M)$.

CM–1 $C(M)$ has one object, which we will denote $*$; $*$ can be chosen arbitrarily. A simple uniform choice is to take $* = M$.

CM–2 The arrows of $C(M)$ are the elements of M with $*$ as source and target.

CM–3 Composition is the binary operation on M.

(This construction is revisited in Section 3.4.)

Thus a category can be regarded as a generalized monoid, or a 'monoid with many objects'. This point of view has not been as fruitful in mathematics as the perception of a category as a generalized poset. However, for computing science, we believe that the monoid metaphor is worth considering. It is explored in this book primarily in Chapter 12.

2.3.13 Remark Many categorists *define* a monoid to be a category with one object (compare 2.3.12) and a preordered set to be a category in which every hom set is either empty or a singleton (compare 2.3.1). This can be justified by the fact that the category of monoids and the category of one-object categories are 'equivalent' as defined in Section 3.4.

2.3.14 Exercises

1. For which sets A is $F(A)$ a commutative monoid?

2. Prove that for each object A in a category $\mathscr{C}$, $\mathrm{Hom}(A, A)$ is a monoid with composition of arrows as the operation.

3. Prove that a semigroup has at most one identity element. (Compare Exercise 4 of Section 2.1.)

2.4 Categories of sets with structure

The typical use of categories has been to consider categories whose objects are sets with mathematical structure and whose arrows are functions that preserve that structure. The definition of category is an abstraction of basic properties of such systems. Typical examples have included categories whose objects are spaces of some type and whose arrows are continuous (or differentiable) functions between the spaces, and categories whose objects are algebraic structures of some specific type and whose arrows are homomorphisms between them.

In this section we describe various categories of sets with structure. The following section considers categories of semigroups and monoids.

Note the contrast with Section 2.3, where we discussed certain mathematical structures as categories. Here, we discuss categories whose *objects* are mathematical structures.

2.4.1 Definition The **category of graphs** has graphs as objects and homomorphisms of graphs (see 1.4.1) as arrows. It is denoted **GRF**. The category of small graphs (see 1.3.4) and homomorphisms between them is denoted **Grf**.

Let us check that the composite of graph homomorphisms is a graph homomorphism (identities are easy). Suppose $\phi : \mathscr{G} \longrightarrow \mathscr{H}$ and $\psi : \mathscr{H} \longrightarrow \mathscr{K}$ are graph homomorphisms, and suppose that $u : m \longrightarrow n$ in $\mathscr{G}$. Then by definition $\phi_1(u) : \phi_0(m) \longrightarrow \phi_0(n)$ in $\mathscr{H}$, and so by definition

$$\psi_1(\phi_1(u)) : \psi_0(\phi_0(m)) \longrightarrow \psi_0(\phi_0(n)) \text{ in } \mathscr{K}$$

Hence $\psi \circ \phi$ is a graph homomorphism.

The identity homomorphism $\mathrm{id}_{\mathscr{G}}$ is the identity function for both nodes and arrows.

2.4.2 The category of posets If (S, α) and (T, β) are posets, a function $f : S \to T$ is **monotone** if whenever $x \alpha y$ in S, $f(x) \beta f(y)$ in T.

The identity function on a poset is clearly monotone, and the composite of two monotone functions is easily seen to be monotone, so that posets with monotone functions form a category. A variation on this is to consider only **strictly monotone** functions, which are functions f with the property that if $x \alpha y$ and $x \neq y$ then $f(x) \beta f(y)$ and $f(x) \neq f(y)$.

In 2.3.1, we saw how a single poset is a category. Now we are considering the *category* of *posets*.

We must give a few words of warning on terminology. The usual word in mathematical texts for what we have called 'monotone' is 'increasing' or 'monotonically increasing'. The word 'monotone' is used for a function that either preserves *or reverses* the order relation. That is, in mathematical texts a function $f : (X, \alpha) \to (T, \beta)$ is also called monotone if whenever $x \alpha y$ in S, $f(y) \beta f(x)$ in T.

2.4.3 ω-complete partial orders We now describe a special type of poset that has been a candidate for programming language semantics. Actually, these days more interest has been shown in various special cases of this kind of poset, but the discussion here shows the approach taken.

Let $(S, \leq)$ be a poset. An **ω-chain** in S is an infinite sequence $s_0, s_1, s_2, \ldots$ of elements of S for which $s_i \leq s_{i+1}$ for all natural numbers i. Note that repetitions are allowed. In particular, for any two elements s and t, if $s \leq t$, there is a chain $s \leq t \leq t \leq t \leq \ldots$.

A **supremum** or **least upper bound** of a subset T of a poset $(S, \leq)$ is an element $v \in S$ with the following two properties:

SUP–1 For every $t \in T$, $t \leq v$.

SUP–2 If $w \in S$ has the property that $t \leq w$ for every $t \in T$, then $v \leq w$.

The supremum of a subset T is unique: if v and v' are suprema of T, then $v \leq v'$ because v' satisfies SUP–1 and v satisfies SUP–2, whereas $v' \leq v$ because v satisfies SUP–1 and v' satisfies SUP–2. Then $v = v'$ by antisymmetry.

If v satisfies SUP–1, it is called an **upper bound** of T.

2.4.4 Definition A poset $(S, \leq)$ is an **ω-complete partial order**, or **ω-CPO**, if every chain has a supremum. If the poset also has a minimum element, it is called a **strict ω-CPO**. In this context, the minimum element is usually denoted $\perp$ and called 'bottom'.

Note: This usage of the word 'complete' follows the customary usage in computing science. However, you should be warned that this use of 'complete' conflicts with standard usage in category theory, where 'complete' refers to

limits, not colimits, so that ω-complete means closed under infimums of all *descending* chains. This concept is discussed in Definitions 9.2.9 and 9.5.2.

For example, every powerset of a set is a strict ω-CPO with respect to inclusion (Exercise 4).

2.4.5 Example A more interesting example from the point of view of computing science is the set $\mathscr{P}$ of partial functions from some set S to itself as defined in 2.1.13. A partial function on S can be described as a set f of ordered pairs of elements of S with the property that if $(s,t) \in f$ and $(s,t') \in f$, then $t = t'$. (Compare 1.2.3.)

We give the set $\mathscr{P}$ a poset structure by defining $f \leq g$ to mean $f \subseteq g$ as a set of ordered pairs. It follows that if f and g are partial functions on S, then $f \leq g$ if and only if the domain of f is included in the domain of g and for every x in the domain of f, $f(x) = g(x)$.

2.4.6 Proposition *$\mathscr{P}$ is a strict ω-CPO.*

Proof. Let $\mathscr{T}$ be a chain in $\mathscr{P}$. The supremum t of $\mathscr{T}$ is simply the union of all the sets of ordered pairs in $\mathscr{T}$. This set t is clearly the supremum if indeed it defines a partial function. So suppose (x,y) and (x,z) are elements of t. Then there are partial functions u and v in $\mathscr{P}$ with $(x,y) \in u$ and $(x,z) \in v$. Since $\mathscr{T}$ is a chain and the ordering of $\mathscr{P}$ is inclusion, there is a partial function w in $\mathscr{T}$ containing both (x,y) and (x,z), for example whichever of u and v is higher in the chain. Since w is a partial function, $y = z$ as required.

The bottom element of $\mathscr{P}$ is the empty function. □

2.4.7 Definition A function $f : S \to T$ between ω-CPOs is **continuous** if whenever s is the supremum of a chain $\mathscr{C} = (c_0, c_1, c_2, \ldots)$ in S, then $f(s)$ is the supremum of the image $f(\mathscr{C}) = \{f(c_i) \mid i \in \mathbf{N}\}$ in T. A continuous function between strict ω-CPOs is **strict** if it preserves the bottom element.

A continuous function is monotone, as can be seen by applying it to a chain $(s,t,t,t,\ldots)$ with $s \leq t$. Thus it follows that the image $f(\mathscr{C})$ of the chain $\mathscr{C}$ in the definition is itself a chain. See [Barendregt, 1984], pp.10ff, for a more detailed discussion of continuous functions.

The ω-complete partial orders and the continuous maps between them form a category. Strict ω-complete partial orders and strict continuous functions also form a category. In fact the latter is a subcategory in the sense to be defined in 2.6.1.

2.4.8 Functions as fixed points Let $f : S \to S$ be a set function. An element $x \in S$ is a **fixed point** of f if $f(x) = x$. Fixed points of functions are of interest in computing science because they provide a way of solving recursion equations. Complete partial orders provide a natural setting for expressing this idea.

2.4.9 Example Consider the ω-CPO $\mathscr{P}$ of partial functions from the set $\mathbf{N}$ of natural numbers to itself. There is a function $\phi : \mathscr{P} \to \mathscr{P}$ that takes a partial function $h : \mathbf{N} \to \mathbf{N}$ in $\mathscr{P}$ to a partial function k defined this way:

(i) $k(0) = 1$;

(ii) for $n > 0$, $k(n)$ is defined if and only if $h(n-1)$ is defined, and $k(n) = nh(n-1)$.

For example, if $h(n) = n^2$ for $n \in \mathbf{N}$, then

$$\phi(h)(n) = \begin{cases} 1 & \text{if } n = 0 \\ n(n-1)^2 & \text{if } n > 0 \end{cases}$$

ϕ is a continuous function from $\mathscr{P}$ to itself. To see this, suppose $\mathscr{H} = (h_0, h_1, \ldots)$ is a chain with supremum h. Then h is the union of all the partial functions h_i as sets of ordered pairs. To show that $\phi(h)$ is the supremum of $\phi(\mathscr{H}) = (\phi(h_0), \phi(h_1), \ldots)$, we must show that for any n, $\phi(h)(n)$ is defined if and only if $\phi(h_i)(n)$ is defined for some i and then $\phi(h)(n) = \phi(h_i)(n)$. That says that $\phi(h)$ is the union of all the partial functions $\phi(h_i)$ and so is the supremum of $\phi(\mathscr{H})$.

Suppose $\phi(h_i)(n)$ is defined. If $n = 0$ then $\phi(h_i)(0) = \phi(h)(0) = 1$ by definition of ϕ. Otherwise, $h_i(n-1)$ is defined and $\phi(h_i)(n) = nh_i(n-1)$. Then $h(n-1) = h_i(n-1)$ since h is the union of the h_i, so $\phi(h)(n) = nh(n-1) = nh_i(n-1) = \phi(h_i)(n)$.

Suppose $\phi(h)(n)$ is defined. Then $h(n-1)$ is defined and $\phi(h)(n) = nh(n-1)$. Since h is the union of the h_i there must be some i for which $h_i(n-1)$ is defined and $h(n-1) = h_i(n-1)$. But then by definition of ϕ, $\phi(h_i)(n) = nh_i(n-1) = nh(n-1) = \phi(h)(n)$ as required.

The unique fixed point of ϕ is the factorial function $f(n) = n!$. In the first place, $f(0) = 1 = \phi(f)(0)$. Also, f is a total function and $f(n) = nf(n-1)$, so that $\phi(f)$ is defined and by definition of ϕ, $\phi(f)(n) = nf(n-1) = f(n)$, so $\phi(f) = f$. If also $\phi(g) = g$, then $g(0) = 1$ and $g(n) = \phi(g)(n) = ng(n-1)$ so by induction g is the factorial function.

The following proposition, applied to the poset $\mathscr{P}$ of partial functions, warrants the general recursive construction of functions.

2.4.10 Proposition *Let $(S, \leq)$ be a strict ω-CPO and $f : S \longrightarrow S$ a continuous function. Then f has a least fixed point, that is an element $p \in S$ with the property that $f(p) = p$ and for any $q \in S$, if $f(q) = q$ then $p \leq q$.*

Proof. Form the chain

$$\mathscr{C} = (\bot, f(\bot), f \circ f(\bot), \ldots, f^k(\bot), \ldots)$$

Note that indeed $f^k(\bot) \leq f^{k+1}(\bot)$ for $k = 0, 1, \ldots$ by induction: $\bot \leq f(\bot)$, and if $f^{k-1}(\bot) \leq f^k(\bot)$, then $f^k(\bot) \leq f^{k+1}(\bot)$ because f is continuous.

Let p be the supremum of the chain $\mathscr{C}$. Since f is continuous, $f(p)$ is the supremum of the chain $f(\mathscr{C}) = (f(\bot), f^2(\bot), \ldots)$. But p is an upper bound of $f(\mathscr{C})$ so $f(p) \leq p$. On the other hand, the only element in $\mathscr{C}$ not in $f(\mathscr{C})$ is $\bot$, which is less than anything, so $f(p)$ is an upper bound of $\mathscr{C}$. Thus $p \leq f(p)$. Hence $p = f(p)$.

If $f(q) = q$, then $\bot \leq q$, $f(\bot) \leq f(q) = q$, ..., $f^n(\bot) \leq f^n(q) = q$, and so on, so that q is an upper bound of $\mathscr{C}$. Hence $p \leq q$. □

This construction will be made in a wider context in Sections 6.6 and 14.1.

2.4.11 Exercises

1. Let (S, α) and (T, β) be sets with relations on them. A **homomorphism** from (S, α) to (T, β) is a function $f : S \longrightarrow T$ with the property that if $x \alpha y$ in S then $f(x) \beta f(y)$ in T.

a. Show that sets with relations and homomorphisms between them form a category.

b. Show that if (S, α) and (T, β) are both posets, then $f : S \longrightarrow T$ is a homomorphism of relations if and only if it is a monotone map.

2. Show that (strict) ω-complete partial orders and (strict) continuous functions form a category.

3. Let $\mathbf{R}^+$ be the set of nonnegative real numbers. Show that the poset $(\mathbf{R}^+, \leq)$ is not an ω-CPO.

4. Show that for every set S, the poset $(\mathscr{P}(S), \subseteq)$ is a strict ω-CPO.

5.† Give an example of ω-CPOs with a monotone map between them that is not continuous. (Hint: Adjoin two elements to $(\mathbf{Z}, \leq)$ that are greater than any integer.)

6. Let $g : \mathbf{N} \longrightarrow \mathbf{N}$ be the function such that $f(n) = 2^n$. Exhibit g as the least fixed point of a continuous function $\psi : \mathscr{P} \longrightarrow \mathscr{P}$ (analogous to the function ϕ of Example 2.4.9).

7. The Fibonacci function F is defined by $F(0) = 0$, $F(1) = 1$ and $F(n) = F(n-1) + F(n-2)$ for $n > 1$. Exhibit the Fibonacci function as the least fixed point of a continuous function from an ω-CPO to itself.

2.5 Categories of algebraic structures

In this section, we discuss the categories whose objects are semigroups or monoids. These are typical of categories of algebraic structures; we have concentrated on semigroups and monoids because transition systems naturally form monoids. The material in this section will come up primarily in examples later, and need not be thoroughly understood in order to read the rest of the book.

2.5.1 Homomorphisms of semigroups and monoids If S and T are semigroups, a function $h : S \longrightarrow T$ is a **homomorphism** if for all $s, s' \in S$, $h(ss') = h(s)h(s')$.

A **homomorphism of monoids** is a semigroup homomorphism between monoids that preserves the identity elements: if e is the identity element of the domain, $h(e)$ must be the identity element of the codomain.

2.5.2 Examples The identity function on any monoid is a monoid homomorphism. If M is a monoid and S is a submonoid (see 2.3.10), the inclusion function from S to M is a monoid homomorphism. Another example is the function that takes an even integer to 0 and an odd integer to 1. This is a monoid homomorphism from the monoid of integers on multiplication to the set $\{0, 1\}$ on multiplication.

Since identity functions are homomorphisms and homomorphisms compose to give homomorphisms (see Exercise 1), we have two categories: **Sem** is the category of semigroups and semigroup homomorphisms, and **Mon** is the category of monoids and monoid homomorphisms.

2.5.3 Example Let S be a semigroup with element s. Let $\mathbf{N}^+$ denote the semigroup of positive integers with addition as operation. There is a semigroup homomorphism $p : \mathbf{N}^+ \longrightarrow S$ for which $p(k) = s^k$. That this is a homomorphism is just the statement $s^{k+n} = s^k s^n$ (see 2.3.5).

2.5.4 A *semigroup* homomorphism between monoids need not preserve the identities. An example of this involves the trivial monoid E with only one element e (which is perforce the identity element) and the monoid of all integers with multiplication as the operation, which is a monoid with identity 1. The function that takes the one element of E to 0 is a semigroup homomorphism that is not a monoid homomorphism. And, by the way, even though $\{0\}$ is a subsemigroup of the integers with multiplication and even though it is actually a monoid, it is *not* a submonoid.

2.5.5 Inverses of homomorphisms As an example of how to use the definition of homomorphism, we show that the inverse of a bijective semigroup homomorphism is also a semigroup homomorphism. Let $f : S \longrightarrow T$ be a bijective semigroup homomorphism with inverse g. Let $t, t' \in T$. We have to show that $g(t)g(t') = g(tt')$. Since f is injective, it is sufficient to show that

$$f(g(t)g(t')) = f(g(tt'))$$

The right hand side is tt' because g is the inverse of f, and the left hand side is

$$f(g(t)g(t')) = f(g(t))f(g(t'))$$

because f is a homomorphism, but that is also tt' since g is the inverse of f.

This sort of theorem is true of other algebraic structures, such as monoids. It is not true for posets (see 2.7.12 below).

2.5.6 Isomorphisms of semigroups If a homomorphism of semigroups has an inverse that is a homomorphism (equivalently, as we just saw, if it is bijective), we say that the homomorphism is an **isomorphism**. In this case, the two semigroups in question have the same abstract structure and are said to be **isomorphic**. As we will see later, the property of possessing an inverse is taken to define the categorical notion of isomorphism (2.7.4).

It is important to understand that there may in general be many different isomorphisms between isomorphic semigroups (Exercise 4).

We now discuss two important types of examples of monoid homomorphisms that will reappear later in the book. The first example is a basic property of free monoids.

2.5.7 Kleene closure induces homomorphisms Let A and B denote sets, thought of as alphabets. Let $f : A \longrightarrow B$ be any set function. We define $f^* : A^* \longrightarrow B^*$ by $f^*((a_1, a_2, \ldots, a_k)) = (f(a_1), f(a_2), \ldots, f(a_k))$. In particular, $f^*() = ()$ and for any $a \in A$, $f^*(a) = f(a)$.

Then f^* is a homomorphism of monoids, a requirement that, in this case, means it preserves identity elements (by definition) and concatenation, which can be seen from the following calculation: Let $a = (a_1, a_2, \ldots, a_m)$ and $a' = (a'_1, a'_2, \ldots, a'_n)$ be lists in A^*. Concatenating them gives the list

$$aa' = (a_1, a_2, \ldots, a_m, a'_1, a'_2, \ldots, a'_n)$$

Then

$$\begin{aligned} f^*(a)f^*(a') &= f^*(a_1, a_2, \ldots, a_m)f^*(a'_1, a'_2, \ldots, a'_n) \\ &= (f(a_1), f(a_2), \ldots, f(a_m))(f(a'_1), f(a'_2), \ldots, f(a'_n)) \end{aligned}$$

$$\begin{aligned} &= (f(a_1), f(a_2), \ldots, f(a_m), f(a'_1), f(a'_2), \ldots, f(a'_n)) \\ &= f^*(a_1, a_2, \ldots, a_m, a'_1, a'_2, \ldots, a'_n) \\ &= f^*(aa') \end{aligned}$$

Thus any set function between sets induces a monoid homomorphism between the corresponding free monoids.

The function f^* is called αf in [Backus, 1981a] and in modern functional languages is usually called map f or maplist f.

The other important example is a basic construction of number theory.

2.5.8 The remainder function The set $\mathbf{Z}$ of all integers forms a monoid with respect to either addition or multiplication. If k is any positive integer, the set $\mathbf{Z}_k = \{0, 1, \ldots, k-1\}$ of **remainders** of k is also a monoid with respect to addition or multiplication $(\mathrm{mod}\, k)$. Here are more precise definitions.

2.5.9 Definition Let k be a positive integer and n any integer. Then $n \bmod k$ is the unique integer $r \in \mathbf{Z}_k$ for which there is an integer q such that $n = qk + r$ and $0 \leq r < k$.

It is not difficult to see that there is indeed a unique integer r with these properties.

Define an operation '$+_k$' of addition $(\mathrm{mod}\, k)$ by requiring that

$$r +_k s = (r + s) \bmod k$$

The operation of addition of the contents of two registers in a microprocessor may be addition $(\mathrm{mod}\, k)$ for k some power of 2 (often complicated by the presence of sign bits).

2.5.10 Proposition *$(\mathbf{Z}_k, +_k)$ is a monoid with identity 0.*

We also have the following.

2.5.11 Proposition *The function $n \mapsto (n \bmod k)$ is a monoid homomorphism from $(\mathbf{Z}, +)$ to $(\mathbf{Z}_k, +_k)$.*

A similar definition and proposition can be given for multiplication.

2.5.12 Exercises

1. Show that the composite of semigroup (respectively monoid) homomorphisms is a semigroup (respectively monoid) homomorphism.

2. Prove Proposition 2.5.10.

3. Prove Proposition 2.5.11.

4.† Exhibit two distinct isomorphisms between the monoid with underlying set $\{0, 1, 2, 3\}$ and addition (mod 4) as operation and the monoid with underlying set $\{1, 2, 3, 4\}$ and multiplication (mod 5) as operation.

5. Using the terminology of 2.5.7, show that if f is an isomorphism then so is f^*.

2.6 Constructions on categories

If you are familiar with some branch of abstract algebra (for example the theory of semigroups, groups or rings) then you know that given two structures of a given type (e.g., two semigroups), you can construct a 'direct product' structure, defining the operations coordinatewise. Also, a structure may have substructures, which are subsets closed under the operations, and quotient structures, formed from equivalence classes modulo a congruence relation. Another construction that is possible in many cases is the formation of a 'free' structure of the given type for a given set.

All these constructions can be performed for categories. We will outline the constructions here, except for quotients, which will be described in Section 3.5. We will also describe the construction of the slice category, which does not quite correspond to anything in abstract algebra (although it is akin to the adjunction of a constant to a logical theory). You do not need to be familiar with the constructions in other branches of abstract algebra, since they are all defined from scratch here.

2.6.1 Definition A **subcategory** $\mathscr{D}$ of a category $\mathscr{C}$ is a category for which:

S–1 All the objects of $\mathscr{D}$ are objects of $\mathscr{C}$ and all the arrows of $\mathscr{D}$ are arrows of $\mathscr{C}$ (in other words, $\mathscr{D}_0 \subseteq \mathscr{C}_0$ and $\mathscr{D}_1 \subseteq \mathscr{C}_1$).

S–2 The source and target of an arrow of $\mathscr{D}$ are the same as its source and target in $\mathscr{C}$ (in other words, the source and target maps for $\mathscr{D}$ are the restrictions of those for $\mathscr{C}$). It follows that for any objects A and B of $\mathscr{D}$, $\mathrm{Hom}_{\mathscr{D}}(A, B) \subseteq \mathrm{Hom}_{\mathscr{C}}(A, B)$.

S–3 If A is an object of $\mathscr{D}$ then its identity arrow id_A in $\mathscr{C}$ is in $\mathscr{D}$.

S–4 If $f : A \to B$ and $g : B \to C$ in $\mathscr{D}$, then the composite (in $\mathscr{C}$) $g \circ f$ is in $\mathscr{D}$ and is the composite in $\mathscr{D}$.

2.6.2 Examples As an example, the category **Fin** of finite sets and all functions between them is a subcategory of **Set**, and in turn **Set** is a subcategory of the category of sets and *partial* functions between sets (see 2.1.12 and 2.1.13). These examples illustrate two phenomena:

(i) If A and B are finite sets, then $\mathrm{Hom}_{\mathbf{Fin}}(A, B) = \mathrm{Hom}_{\mathbf{Set}}(A, B)$. In other words, every arrow of **Set** between objects of **Fin** is an arrow of **Fin**.

(ii) The category of sets and the category of sets and partial functions, on the other hand, have exactly the same *objects*. The phenomenon of (i) does not occur here: there are generally *many* more partial functions between two sets than there are full functions.

Example (i) motivates the following definition.

2.6.3 Definition If $\mathscr{D}$ is a subcategory of $\mathscr{C}$ and for every pair of objects A, B of $\mathscr{D}$, $\mathrm{Hom}_{\mathscr{D}}(A, B) = \mathrm{Hom}_{\mathscr{C}}(A, B)$, then $\mathscr{D}$ is a **full subcategory** of $\mathscr{C}$.

Thus **Fin** is a full subcategory of **Set** but **Set** is not a full subcategory of the category of sets and partial functions.

Example 2.6.2(ii) also motivates a (less useful) definition, as follows.

2.6.4 Definition If $\mathscr{D}$ is a subcategory of $\mathscr{C}$ with the same objects, then $\mathscr{D}$ is a **wide** subcategory of $\mathscr{C}$.

Thus in the case of a wide subcategory, only the arrows are different from those of the larger category. In 2.7.13 we provide an improvement on this concept.

As an example, **Set** is a wide subcategory of the category **Pfn** of sets and partial functions.

2.6.5 Example Among all the objects of the category of semigroups are the monoids, and among all the semigroup homomorphisms between two monoids are those that preserve the identity. Thus the category of monoids is a subcategory of the category of semigroups that is neither wide nor full (for the latter, see 2.5.4).

As it stands, being a subcategory requires the objects and arrows of the subcategory to be identical with some of the objects and arrows of the category containing it. This requires an uncategorical emphasis on what something *is* instead of on the specification it satisfies. We will return to this example in 2.8.13 and again in 3.1.7.

2.6.6 The product of categories If $\mathscr{C}$ and $\mathscr{D}$ are categories, the **product** $\mathscr{C} \times \mathscr{D}$ is the category whose objects are all ordered pairs (C, D) with C an object of $\mathscr{C}$ and D an object of $\mathscr{D}$, and in which an arrow $(f, g) : (C, D) \longrightarrow (C', D')$ is a pair of arrows $f : C \longrightarrow C'$ in $\mathscr{C}$ and $g : D \longrightarrow D'$ in $\mathscr{D}$. The identity of (C, D) is $(\mathrm{id}_C, \mathrm{id}_D)$. If $(f', g') : (C', D') \longrightarrow (C'', D'')$ is another arrow, then the composite is defined by

$$(f', g') \circ (f, g) = (f' \circ f, g' \circ g) : (C, D) \longrightarrow (C'', D'')$$

2.6.7 The dual of a category Given any category $\mathscr{C}$, you can construct another category denoted $\mathscr{C}^{\mathrm{op}}$ by reversing all the arrows. The **dual** or **opposite** $\mathscr{C}^{\mathrm{op}}$ of a category $\mathscr{C}$ is defined by:

D–1 The objects and arrows of $\mathscr{C}^{\mathrm{op}}$ are the objects and arrows of $\mathscr{C}$.

D–2 If $f : A \longrightarrow B$ in $\mathscr{C}$, then $f : B \longrightarrow A$ in $\mathscr{C}^{\mathrm{op}}$.

D–3 If $h = g \circ f$ in $\mathscr{C}$, then $h = f \circ g$ in $\mathscr{C}^{\mathrm{op}}$.

The meaning of D–2 is that source and target have been reversed. It is easy to see that the identity arrows have to be the same in the two categories $\mathscr{C}$ and $\mathscr{C}^{\mathrm{op}}$ and that C–1 through C–4 of Section 2.1 hold, so that $\mathscr{C}^{\mathrm{op}}$ is a category.

2.6.8 Example If M is a monoid, then the opposite of the category $C(M)$ is the category determined by a monoid M^{op}; if $xy = z$ in M, then $yx = z$ in M^{op}. (Hence if M is commutative then $C(M)$ is its own dual. Compare Exercise 6 of 3.3.) Similar remarks may be made about the opposite of the category $C(P)$ determined by a poset P. The opposite of the poset $(\mathbf{Z}, \leq)$, for example, is $(\mathbf{Z}, \geq)$.

2.6.9 Both the construction of the product of two categories and the construction of the dual of a category are purely formal constructions. Even though the original categories may have, for example, structure-preserving functions of some kind as arrows, the arrows in the product category are simply pairs of arrows of the original categories.

Consider **Set**, for example. Let A be the set of letters of the English alphabet. The function $v : A \longrightarrow \{0, 1\}$ that takes consonants to 0 and vowels to 1 is an arrow of **Set**. Then the arrow $(\mathrm{id}_A, v) : (A, A) \longrightarrow (A, \{0, 1\})$ of $\mathbf{Set} \times \mathbf{Set}$ is not a function, not even a function of two variables; it is merely the arrow of a product category and as such is an ordered pair of functions.

A similar remark applies to duals. In $\mathbf{Set}^{\mathrm{op}}$, v is an arrow from $\{0, 1\}$ to A. And that is all it is. It is in particular *not* a function from $\{0, 1\}$ to A.

Nevertheless, it is possible in some cases to prove that the dual of a familiar category is essentially the same as some other familiar category. One such category is **Fin**, see 3.4.6.

The product of categories is a formal way to make constructions dependent on more than one variable. The major use we make of the concept of dual is that many of the definitions we make have another meaning when applied to the dual of a category that is often of independent interest. The phrase **dual concept** or **dual notion** is often used to refer to a concept or notion applied in the dual category.

2.6.10 Slice categories If $\mathscr{C}$ is a category and A any object of $\mathscr{C}$, the **slice category** $\mathscr{C}/A$ is described this way:

SC–1 An object of $\mathscr{C}/A$ is an arrow $f : C \longrightarrow A$ of $\mathscr{C}$ for some object C.

SC–2 An arrow of $\mathscr{C}/A$ from $f : C \longrightarrow A$ to $f' : C' \longrightarrow A$ is an arrow $h : C \longrightarrow C'$ with the property that $f = f' \circ h$.

SC–3 The composite of $h : f \longrightarrow f'$ and $h' : f' \longrightarrow f''$ is $h' \circ h$.

It is necessary to show that $h' \circ h$, as defined in SC–2, satisfies the requirements of being an arrow from f to f''. Let $h : f \longrightarrow f'$ and $h' : f' \longrightarrow f''$. This means $f' \circ h = f$ and $f'' \circ h' = f'$. To show that $h' \circ h : f \longrightarrow f''$ is an arrow of $\mathscr{C}/A$, we must show that $f'' \circ (h' \circ h) = f$. That follows from this calculation:

$$f'' \circ (h' \circ h) = (f'' \circ h') \circ h = f' \circ h = f$$

The usual notation for arrows in $\mathscr{C}/A$ is deficient: the same arrow h can satisfy $f = f' \circ h$ and $g = g' \circ h$ with $f \neq g$ or $f' \neq g'$ (or both). Then $h : f \longrightarrow f'$ and $h : g \longrightarrow g'$ are *different* arrows of $\mathscr{C}/A$.

2.6.11 The importance of slice categories comes in part with their connection with indexing. An **S-indexed set** is a set X together with a function $\tau : X \longrightarrow S$. If $x \in X$ and $\tau(x) = s$ then we say x is of **type** s, and we also refer to X as a **typed set**.

The terminology 'S-indexed set' is that used by category theorists. Many mathematicians would cast the discussion in terms of the collection $\{\tau^{-1}(s) \mid s \in S\}$ of subsets of X, which would be called a **family of sets indexed by** S.

2.6.12 Example The set $G = G_0 \cup G_1$ of objects and arrows of a graph $\mathscr{G}$ is an example of a typed set, typed by the function $\tau : G \longrightarrow \{0, 1\}$ that takes a node to 0 and an arrow to 1. Note that this depends on the fact that a node is not an arrow: G_0 and G_1 are disjoint.

2.6.13 Indexed functions A function from a set X typed by S to a set X' typed by the same set S that preserves the typing (takes an element of type s to an element of type s) is exactly an arrow of the slice category $\mathbf{Set}/S$. Such a function is called an **indexed function** or **typed function**.

It has been fruitful for category theorists to pursue this analogy by thinking of objects of any slice category $\mathscr{C}/A$ as objects of $\mathscr{C}$ indexed by A.

2.6.14 Example A graph homomorphism $f : \mathscr{G} \to \mathscr{H}$ corresponds to a typed function according to the construction in Example 2.6.12. However, there are typed functions between graphs that are not graph homomorphisms, for example the function from the graph (1.1), page 9, to the graph (1.3), page 10, defined by

$$1 \mapsto 1, 2 \mapsto \mathsf{n}, a \mapsto 0, b \mapsto 0, c \mapsto \mathsf{succ}$$

This is not a graph homomorphism because it does not preserve source and target.

2.6.15 Example Let (P, α) be a poset and let $C(P)$ be the corresponding category as in 2.3.1. For an element $x \in P$, the slice category $C(P)/x$ is the category corresponding to the set of elements greater than or equal to x. The dual notion of coslice gives the set of elements less than or equal to a given element.

2.6.16 The free category generated by a graph For any given graph $\mathscr{G}$ there is a category $F(\mathscr{G})$ whose objects are the nodes of $\mathscr{G}$ and whose arrows are the paths in $\mathscr{G}$. Composition is defined by the formula

$$(f_1, f_2, \ldots, f_k) \circ (f_{k+1}, \ldots, f_n) = (f_1, f_2, \ldots, f_n)$$

This composition is associative, and for each object A, id_A is the empty path from A to A. The category $F(\mathscr{G})$ is called the **free category generated by the graph** $\mathscr{G}$. It is also called the **path category** of $\mathscr{G}$.

2.6.17 Examples The free category generated by the graph with one node and no arrows is the category with one object and only the identity arrow, which is the empty path. The free category generated by the graph with one node and one loop on the node is the free monoid with one generator (Kleene closure of a one-letter alphabet); this is isomorphic with the nonnegative integers with + as operation.

The free category generated by the graph in Example 1.3.8 has the following arrows

(a) An arrow $\mathrm{id}_1 : 1 \to 1$.

(b) For each nonnegative integer k, the arrow $\mathsf{succ}^k : \mathsf{n} \to \mathsf{n}$. This is the path $(\mathsf{succ}, \mathsf{succ}, \ldots, \mathsf{succ})$ (k occurrences of succ). This includes $k = 0$ which gives id_{n}.

(c) For each nonnegative integer k, the arrow $\mathsf{succ}^k \circ 0 : 1 \longrightarrow \mathsf{n}$. Here $k = 0$ gives $0 : 1 \longrightarrow \mathsf{n}$.

Composition obeys the rule $\mathsf{succ}^k \circ \mathsf{succ}^m = \mathsf{succ}^{k+m}$.

2.6.18 It is useful to regard the free category generated by *any* graph as analogous to Kleene closure (free monoid) generated by a set (as in 2.3.9). The paths in the free category correspond to the strings in the Kleene closure. The difference is that you can concatenate any symbols together to get a string, but arrows can be strung together only head to tail, thus taking into account the typing.

In 13.2.3 we give a precise technical meaning to the word 'free'.

2.6.19 Exercises

1. Let M be a monoid. Show that the opposite of the category $C(M)$ determined by M is also the category determined by a monoid, called M^{op}.

2. Do the same as the preceding exercise for posets.

3. Give examples of posets P, Q and R for which $C(P)$ (the category determined by the poset) is a wide but not full subcategory of $C(Q)$ and $C(P)$ is a full but not wide subcategory of $C(R)$.

4.† Show by example that the requirement S–3 in 2.6.1 does not follow from the other requirements.

2.7 Properties of objects and arrows in a category

The data in the definition of category can be used to define properties that the objects and arrows of the category may have. A property that is defined strictly in terms of the role the object or arrow has in the category, rather than in terms of what it really is in any sense, is called a **categorical definition**. Such definitions are *abstract* in the sense that a property a thing can have is defined entirely in terms of the external interactions of that thing with other entities.

The examples of categorical definitions in this section and the next are of several simple concepts that can be expressed directly in terms of the data used in the definition of category. Other concepts, such as limit, naturality and adjunction, require deeper ideas that will be the subject of succeeding chapters.

2.7.1 Isomorphisms In general, the word 'isomorphic' is used in a mathematical context to mean indistinguishable in form. We have already used it in this way in 2.5.6. It turns out that it is possible to translate this into categorical language in a completely satisfactory way. To do this, we first need the concept of inverse.

2.7.2 Definition Suppose $f : A \to B$ and $g : B \to A$ are arrows in a category for which $f \circ g$ is the identity arrow of B and $g \circ f$ is the identity arrow of A. Then g is an **inverse** to f, and, of course, f is an inverse to g.

2.7.3 As an example of how to use the definition of inverse, we show that if $f : A \to B$ has an inverse, it has only one. Suppose $g : B \to A$ and $h : B \to A$ have the properties that $g \circ f = h \circ f = \mathrm{id}_A$ and $f \circ g = f \circ h = \mathrm{id}_B$. Then

$$g = g \circ \mathrm{id}_B = g \circ (f \circ h) = (g \circ f) \circ h = \mathrm{id}_A \circ h = h$$

Note that this does not use the full power of the hypothesis.

From this uniqueness, we can conclude that if $f : A \to A$ has an inverse, then $(f^{-1})^{-1} = f$. Proof: All four of the following equations are true by definition of inverse:

(i) $f^{-1} \circ f = \mathrm{id}_A$.
(ii) $f \circ f^{-1} = \mathrm{id}_A$.
(iii) $f^{-1} \circ (f^{-1})^{-1} = \mathrm{id}_A$.
(iv) $(f^{-1})^{-1} \circ f^{-1} = \mathrm{id}_A$.

It follows that both f and $(f^{-1})^{-1}$ are arrows g such that $f^{-1} \circ g = \mathrm{id}_A$ and $g \circ f^{-1} = \mathrm{id}_A$. Thus $f = (f^{-1})^{-1}$ by uniqueness of the inverse of f^{-1}.

2.7.4 Definition Suppose that $\mathscr{C}$ is a category and that A and B are two objects of $\mathscr{C}$. An arrow $f : A \to B$ is said to be an **isomorphism** if it has an inverse. In that case, we say that A is **isomorphic to** B, written $A \cong B$.

In a monoid, an element which is an isomorphism in the corresponding category is usually called **invertible**.

2.7.5 Definition An arrow $f : A \to A$ in a category (with the source and target the same) is called an **endomorphism**. If it is invertible, it is called an **automorphism**.

2.7.6 Examples Any identity arrow in any category is an isomorphism (hence an automorphism). In the category determined by a partially ordered set, the *only* isomorphisms are the identity arrows. If, in the category determined by a monoid, every arrow is an isomorphism the monoid is called

a **group**. Because of this, a category in which every arrow is an isomorphism is called a **groupoid**.

2.7.7 Definition A property that objects of a category may have is **preserved by isomorphisms** if for any object A with the property, any object isomorphic to A must also have the property.

2.7.8 To show that two objects are isomorphic, it is sufficient to exhibit an isomorphism between them. To show they are not isomorphic is often more difficult, since the definition requires checking every arrow between the two objects. In practice it is almost always done by finding some property that is preserved by isomorphisms and is possessed by one of the objects and not possessed by the other. See 2.7.11.

2.7.9 Proposition *A function in* **Set** *is an isomorphism if and only if it is a bijection.*

Proof. Suppose first that $f : S \longrightarrow T$ is an isomorphism; it therefore has an inverse $g : T \longrightarrow S$. Then (i) f is injective, for if $f(x) = f(y)$, then $x = g(f(x)) = g(f(y)) = y$. Also (ii) f is surjective, for if $t \in T$, then $f(g(t)) = t$.

Conversely, suppose that $f : S \longrightarrow T$ is bijective. Define $g : T \longrightarrow S$ by saying that $g(t)$ is the unique element $x \in S$ for which $f(x) = t$. There is such an element x because f is surjective, and x is unique because f is injective. The definition itself says that $f(g(t)) = t$ for *any* $t \in T$, so in particular $f(g(f(x))) = f(x)$; since f is known to be injective, it follows that $g(f(x)) = x$, as required. □

It follows that two finite sets are isomorphic (in the category of sets) if and only if they have the same number of elements.

2.7.10 Example A graph homomorphism $\phi : \mathscr{G} \longrightarrow \mathscr{H}$ is an isomorphism if and only if both ϕ_0 and ϕ_1 are bijections. This follows immediately from Exercise 3 of Section 1.4.

2.7.11 Example For ordered sets, the situation is different. For example, the following partially ordered sets A and B containing three elements each are not isomorphic in the category of partially ordered sets and monotone functions: A consists of three elements $a < b < c$ and B consists of three elements $x < y,\ x < z$, but no relation holds between y and z.

It seems clear that A and B are not isomorphic, but it might seem hard to say why. One way is simply to observe that A is totally ordered (for any two elements u and v, either $u < v$ or $v < u$) and B is not totally ordered.

Since being totally ordered is preserved by isomorphisms, the two posets cannot be isomorphic.

An alternative proof is possible in the case of the preceding example: exhaustively consider all 6 bijections between the two posets and show that none of those that are monotone have inverses that are monotone. This approach clearly is unacceptably time-consuming for any interesting problem.

It is often quite hard to show that two structures are not isomorphic. One often approaches this problem by trying to find numerical invariants. In the case at hand, a simple invariant is the **depth**, that is the length of the longest totally ordered subset: the depth of A is 3 and the depth of B is 2. A set of invariants is called **complete** if it is sufficient to decide the isomorphism class. Complete sets of invariants rarely exist. Depth is sufficient in the case of 2.7.11 but is certainly not in general.

2.7.12 It is often the case that in a category of sets with structure and structure-preserving functions, a structure-preserving function that is a bijection is automatically an isomorphism. This is not always the case, as is illustrated by the poset example in 2.7.11. In fact, the bijection from A to B that takes x to a, y to b and z to c preserves the order relation. Of course, it also takes a pair of incomparable elements to comparable ones, but the point is it preserves the order, insofar as it exists.

2.7.13 From the categorist's point of view there is no reason to distinguish between two isomorphic objects in a category, since the interesting fact about a mathematical object is the way it relates to other mathematical objects and two isomorphic objects relate to other objects in the same way. For this reason, the concept of wide category (Definition 2.6.4) is not in the spirit of category theory. What really should matter is whether the subcategory contains an isomorphic copy of every object in the big category. This motivates the following definition.

2.7.14 Definition A subcategory $\mathscr{D}$ of a category $\mathscr{C}$ is said to be a **representative subcategory** if every object of $\mathscr{C}$ is isomorphic to some object of $\mathscr{D}$.

2.7.15 Example Let $\mathscr{D}$ be the category whose objects are the empty set and all sets of the form $\{1, 2, \ldots, n\}$ for some positive integer n and whose arrows are all the functions between them. Then $\mathscr{D}$ is a representative subcategory of **Fin** (Definition 2.1.12), since there is a bijection from any nonempty finite set to some set of the form $\{1, 2, \ldots, n\}$. Note that $\mathscr{D}$ is also full in **Fin**.

2.7.16 Terminal and initial objects An object T of a category $\mathscr{C}$ is called **terminal** if there is *exactly* one arrow $A \to T$ for each object A of $\mathscr{C}$. We usually denote the terminal object by 1 and the unique arrow $A \to 1$ by $\langle\rangle$.

The dual notion (see 2.6.7), an object of a category that has a unique arrow *to* each object (including itself), is called an **initial object** and is often denoted 0.

2.7.17 Examples In the category of sets, the empty set is initial and any one-element set is terminal. Thus the category of sets has a *unique* initial object but many terminal objects. The one-element monoid is both initial and terminal in the category of monoids. In the category determined by a poset, an object is initial if and only if it is an absolute minimum for the poset, and it is terminal if and only if it is an absolute maximum. Since there is no largest or smallest whole number, the category determined by the set of integers with its natural order (there is an arrow from m to n if and only if $m \leq n$) gives an example of a category without initial or terminal object.

In the category of semigroups, the empty semigroup (see 2.3.6) is the initial object and any one-element semigroup is a terminal object. On the other hand, the category of *nonempty* semigroups does not have an initial object. Warning: To prove this, it is not enough to say that the initial object in the category of semigroups is the empty semigroup and that semigroup is missing here! You have to show that no other object can be the initial object in the smaller category. One way to do this is to let U be the semigroup with two elements 1 and e with $1 \cdot e = e \cdot 1 = e$, $1 \cdot 1 = 1$ and $e \cdot e = e$. Then any nonempty semigroup S has *two* homomorphisms to U: the constant function taking everything to 1 and the one taking everything to e. Thus no nonempty semigroup S can be the initial object.

2.7.18 Proposition *Any two terminal (respectively initial) objects in a category are isomorphic.*

Proof. Suppose T and T' are terminal objects. Since T is terminal, there is an arrow $f : T' \to T$. Similarly, there is an arrow $g : T \to T'$. The arrow $f \circ g : T \to T$ is an arrow with target T. Since T is a terminal object of the category, there can be only one arrow from T to T. Thus it must be that $f \circ g$ is the identity of T. An analogous proof shows that $g \circ f$ is the identity of T'. □

2.7.19 Constants In **Set**, an element x of a set A is the image of a function from a singleton set to A that takes the unique element of the singleton to x. Thus if we pick a specific singleton $\{*\}$ and call it 1, the elements of the set A are in one to one correspondence with $\mathrm{Hom}(1, A)$, which is the set

of functions from the terminal object to A. Moreover, if $f : A \longrightarrow B$ is a set function and x is an element of A determining the function $x : 1 \longrightarrow A$, then the element $f(x)$ of B is essentially the same thing as the composite $f \circ x : 1 \longrightarrow B$. Because of this, the categorist typically thinks of an element $x \in A$ as *being* the constant $x : 1 \longrightarrow A$.

An arrow $1 \longrightarrow A$ in a category, where 1 is the terminal object, is called a **constant of type** A. Thus each element of a set is a constant in **Set**. On the other hand, each monoid M has just one constant $1 \longrightarrow M$ in the category of monoids, since monoid homomorphisms must preserve the identity. The name 'constant' is explained by Exercise 7.

The more common name in the categorical literature for a constant is **global element** of A, a name that comes from sheaf theory (see Section 15.5).

> A terminal object is an object with exactly one arrow $\langle\rangle : A \longrightarrow 1$ to it from each object A. So the arrows *to* 1 are not interesting. Global elements are arrows *from* the terminal object. There may be none or many, so *they* are interesting.

2.7.20 If 1 and $1'$ are two terminal objects in a category and $x : 1 \longrightarrow A$ and $x' : 1' \longrightarrow A$ are two constants with the property that $x' \circ \langle\rangle = x$ (where $\langle\rangle$ is the unique isomorphism from 1 to $1'$), then we regard x and x' as the *same* constant. Think about this comment as it applies to elements in the category of sets, with two different choices of terminal object, and you will see why.

2.7.21 Exercises

1. Show that if $f : A \longrightarrow B$ and $g : B \longrightarrow C$ are isomorphisms in a category with inverses $f^{-1} : B \longrightarrow A$ and $g^{-1} : C \longrightarrow B$, then $g \circ f$ is an isomorphism with inverse $f^{-1} \circ g^{-1}$. (This is sometimes called the 'Shoe–Sock Theorem': to undo the act of putting on your socks, then your shoes, you have to take off your *shoes*, then your *socks*.)

2. Give examples of posets P_1, P_2 and P_3 with the following properties:

(a) $P_1 = P_1{}^{\mathrm{op}}$.

(b) $P_2 \neq P_2{}^{\mathrm{op}}$ but P_2 is isomorphic to $P_2{}^{\mathrm{op}}$.

(c) P_3 is not isomorphic to $P_3{}^{\mathrm{op}}$.

(See Exercise 2 of Section 2.6.)

3.† Give examples of a monoid M for which $M \neq M^{\mathrm{op}}$ (meaning the binary operations are different) but M and M^{op} are isomorphic (compare 2.6.8) and one for which M and M^{op} are not isomorphic. (See Exercise 1 of Section 2.6.)

4. Show that a poset isomorphic to a totally ordered set must be totally ordered.

5. Let $(P, \leq)$ be a poset. Show that in the corresponding category $C(P, \leq)$ (see 2.3.1), no two distinct objects are isomorphic.

6. Show that in the category of semigroups (respectively monoids), the isomorphisms are exactly the bijective homomorphisms. (This is not true for ordered sets, as we saw in 2.7.12. It is true in any variety of universal algebras, and it is not true in most interesting categories of topological spaces.)

7. Show that the following two statements about a set function $f : A \longrightarrow B$ are equivalent.

(a) $f = k \circ \langle\rangle$, where $k : 1 \longrightarrow B$ (hence is a constant in the sense of 2.7.19) and $\langle\rangle : A \longrightarrow 1$ is the unique function given by definition of terminal object.

(b) For all x and y in A, $f(x) = f(y)$.

8. Show that in the category of graphs and graph homomorphisms, the graph with one node and one arrow is the terminal object.

9. An arrow $f : A \longrightarrow A$ in a category is **idempotent** if $f \circ f = f$. Show that in **Set** a function is idempotent if and only if its image is the same as its set of fixed points. (For example, applying a specific sorting method to a set of files is idempotent, since if you sort an already sorted file you leave it the same.)

10. An idempotent (see the preceding problem) $f : A \longrightarrow A$ in a category is **split** if there is an object B and functions $g : A \longrightarrow B$ and $h : B \longrightarrow A$ for which $h \circ g = f$ and $g \circ h = \mathrm{id}_B$.

a. Show that every idempotent in **Set** is split.

b.† Give an example of a category with a non-split idempotent.

11. A category in which every arrow is an isomorphism is a **groupoid**. A category in which every arrow is an identity arrow is called **discrete**. Prove or disprove:

(i) Any two objects in a groupoid are isomorphic.

(ii) A groupoid in which no two distinct objects are isomorphic is discrete.

(iii) A poset P for which $C(P)$ is a groupoid is discrete.

2.8 Monomorphisms and subobjects

2.8.1 Monomorphisms A function $f : A \to B$ in **Set** is injective if for any $x, y \in A$, if $x \neq y$, then $f(x) \neq f(y)$. A monomorphism is a particular type of arrow in a category which generalizes the concept of injective function; in particular, a monomorphism in the category of sets is exactly an injective function. If f is an arrow in an arbitrary category, we use the same definition, except for one change required because the concept of 'element' no longer makes sense.

2.8.2 Definition $f : A \to B$ is a **monomorphism** if for any object T of the category and any arrows $x, y : T \to A$, if $x \neq y$, then $f \circ x \neq f \circ y$.

We often write $f : A \rightarrowtail B$ to indicate that f is a monomorphism and say that f is **monic** or that f is **mono**.

In Definition 2.8.2 and many like it, what replaces the concept of element of A is an arbitrary arrow into A. In this context, an arbitrary arrow $a : T \to A$ is called a **variable element** of A, parametrized by T. When a is treated as a variable element and f has source A, one may write $f(a)$ for $f \circ a$. Using this notation, f is a monomorphism if for any variable elements $x, y : T \to A$, if $x \neq y$, then $f(x) \neq f(y)$.

The following theorem validates the claim that 'monomorphism' is the categorical version of 'injective'.

2.8.3 Theorem *In the category of sets, a function is injective if and only if it is a monomorphism.*

Proof. Suppose $f : A \to B$ is injective, and let $a, a' : T \to A$ be variable elements of A. If $a \neq a'$ then there is an (ordinary) element $t \in T$ for which $a(t) \neq a'(t)$. Then $f(a(t)) \neq f(a'(t))$, so $f \circ a \neq f \circ a'$. Hence f is monic.

Conversely, suppose f is monic. Since global elements (see 2.7.19) are elements, this says that for any global elements $x, y : 1 \to A$ with $x \neq y$, $f \circ x \neq f \circ y$, i.e., $f(x) \neq f(y)$, which means that f is injective. □

2.8.4 Examples In most familiar categories of sets with structure and functions that preserve structure, the monomorphisms are exactly the injective functions. In particular, the monomorphisms in **Mon** are the injective homomorphisms (proved in 2.8.5 below). Some other examples are in the exercises. This is evidence that Definition 2.8.2 is the correct categorical definition generalizing the set-theoretic concept of injectivity.

In the category determined by a poset, every arrow is monic. A monic element of the category determined by a monoid is generally called **left cancellable**.

An isomorphism in any category is a monomorphism. For suppose f is an isomorphism and $f \circ x = f \circ y$. This calculation shows that $x = y$:

$$x = f^{-1} \circ f \circ x = f^{-1} \circ f \circ y = y$$

2.8.5 We now show that a monomorphism in the category of monoids is an injective homomorphism, and conversely.

Let $f : M \longrightarrow M'$ be a monoid homomorphism. Suppose it is injective. Let $g,\ h : V \longrightarrow M$ be homomorphisms for which $f \circ g = f \circ h$. For any $v \in V$, $f(g(v)) = f(h(v))$, so $g(v) = h(v)$ since f is injective. Hence $g = h$. It follows that f is a monomorphism. Essentially the same proof works in other categories of structures and structure-preserving maps – if the map is injective it is a monomorphism for the same reason as in **Set**.

However, the converse definitely does not work that way. The proof for **Set** in Theorem 2.8.3 above uses distinct global elements x and y, but a monoid need not have distinct global elements. For example, let **N** denote the monoid of nonnegative integers with addition as operation. Then the only global element of **N** on addition is 0. So we have to work harder to get a proof.

Suppose f is a monomorphism. Let $x,\ y \in M$ be distinct elements. Let $p_x : \mathbf{N} \longrightarrow M$ take k to x^k and similarly define p_y; p_x and p_y are homomorphisms since for all x, $x^{k+n} = x^k x^n$ (see 2.3.5 and the discussion after Definition 2.8.2). Since $x \neq y$, p_x and p_y are *distinct* homomorphisms. If $f(x) = f(y)$ then for all positive integers k,

$$f(p_x(k)) = f(x^k) = f(x)^k = f(y)^k = f(y^k) = f(p_y(k))$$

so that $f \circ p_x = f \circ p_y$ which would mean that f is not a monomorphism. Thus we must have $f(x) \neq f(y)$ so that f is injective.

The trick in the preceding paragraph was to find an object (**N** here) that allows one to distinguish elements of the arbitrary monoid M. In **Set**, the corresponding object was the terminal object, but that does not work for **Mon**: each monoid has exactly one global element because a map from the one-element monoid must have the identity element as value. An object that allows one to (uniquely) distinguish elements in a category of sets with structure is said to 'represent the underlying functor'. This concept is presented in 4.3.10 and 4.5.1.

We now state two propositions that give some elementary properties of monomorphisms.

2.8.6 Proposition *Suppose $f : A \longrightarrow B$ and $g : B \longrightarrow C$ in a category $\mathscr{C}$. Then*

(a) *If f and g are monomorphisms, so is $g \circ f$.*

(b) *If $g \circ f$ is a monomorphism, so is f.*

Proof. We prove the second statement and leave the first to you (Exercise 2). Suppose $g \circ f$ is a monomorphism. To show that f is a monomorphism, assume $f \circ x = f \circ y$ for some arrows $x, y : C \longrightarrow A$. Then

$$(g \circ f) \circ x = g \circ (f \circ x) = g \circ (f \circ y) = (g \circ f) \circ y$$

so, since $g \circ f$ is a monomorphism, $x = y$. □

2.8.7 Proposition *Let $m : C \longrightarrow 0$ be a monomorphism into an initial object. Then m is an isomorphism.*

Proof. Let $i : 0 \longrightarrow C$ be the unique arrow given by definition of initial object. Then $m \circ i$ and id_0 are both arrows from 0 to 0 and so must be the same. It remains to show that $i \circ m = \mathrm{id}_C$. This follows from the fact that $m \circ i \circ m = m \circ \mathrm{id}_C$ and that m is a monomorphism. □

2.8.8 Subobjects The concept of subobject is intended to generalize the concept of subset of a set, submonoid of a monoid, subcategory of a category, and so on. This idea cannot be translated exactly into categorical terms, since the usual concept of subset violates the strict typing rules of category theory: to go from a subset to a set requires a change of type, so there is no feasible way to say that the same element x is in both a set and a subset of the set.

Because of this, any categorical definition of subobject will not give exactly the concept of subset when applied to the category of sets. However, the usual definition of subobject (which we give here in Definition 2.8.11) produces, in **Set**, a concept that is naturally equivalent to the concept of subset in a strong sense that we will describe in 2.8.12. The definition, when applied to sets, defines subset in terms of the inclusion function.

2.8.9 We need a preliminary idea. If $f : A \longrightarrow B$ is an arrow in a category, and for some arrow $g : C \longrightarrow B$ there is an arrow $h : A \longrightarrow C$ for which $f = g \circ h$, we say f **factors through** g. This is because the equation $g \circ h = f$ can be solved for h.

> The use of the word 'factor' shows the explicit intention of categorists to work with functions in an algebraic manner: a category is an algebra of functions.

Suppose $f_0 : C_0 \longrightarrow C$ and $f_1 : C_1 \longrightarrow C$ are monomorphisms in a category. Let us say that $f_0 \sim f_1$ if each factors through the other.

2.8.10 Proposition *Let $f_0 \sim f_1$. Then the factors implied by the definition of $\sim$ are unique and are inverse isomorphisms. Moreover, the relation $\sim$ is an equivalence relation on the collection of arrows with target C.*

Proof. The definition implies the existence of arrows $g : C_0 \longrightarrow C_1$ and $h : C_1 \longrightarrow C_0$ such that $f_1 \circ g = f_0$ and $f_0 \circ h = f_1$. The arrows g and h are unique because f_0 and f_1 are monomorphisms. Moreover, $f_1 \circ g \circ h = f_0 \circ h = f_1 = f_1 \circ \mathrm{id}$; since f_1 is a monomorphism, we conclude that $g \circ h = \mathrm{id}$. Similarly, $h \circ g = \mathrm{id}$.

That $\sim$ is reflexive follows by taking the factor to be the identity arrow, and it is symmetric by definition. For transitivity, you get the required factor by composing the given factors; we leave the details to you. □

2.8.11 Definition In a category $\mathscr{C}$, a **subobject** of an object C is an equivalence class of monomorphisms under $\sim$. The subobject is a **proper subobject** if it does not contain id_C.

Observe that it follows immediately from Proposition 2.8.7 that an initial object in a category has no proper subobjects.

2.8.12 Subobjects in the category of sets In **Set**, a monomorphism is an injection, so a subobject is an equivalence class of injections. The following sequence of statements are each easy to prove and together form a precise description of the connection between subobjects and subsets in the category of sets. Similar remarks can be made about other categories of sets with structure, such as semigroups, monoids or posets.

In these statements, S is a set.

(a) Let $\mathscr{O}$ be a subobject of S.

  (i) Any two injections $m : A \longrightarrow S$ and $n : B \longrightarrow S$ in $\mathscr{O}$ have the same image; call the image I.

  (ii) The inclusion $i : I \longrightarrow S$ is equivalent to any injection in $\mathscr{O}$, hence is an element of $\mathscr{O}$.

  (iii) If $j : J \longrightarrow S$ is an inclusion of a subset J into S that is in $\mathscr{O}$, then $I = J$ and $i = j$.

  (iv) Hence every subobject of S contains exactly one inclusion of a subset of S into S, and that subset is the image of any element of $\mathscr{O}$.

(b) Let $i : T \longrightarrow S$ be the inclusion of a subset T of S into S.

  (i) Since i is injective, it is an element of a subobject of S.

  (ii) Since the subobjects are equivalence classes of an equivalence relation, they are disjoint, so i is not in two subobjects.

(iii) Hence the subsets of S with their inclusion maps form a complete set of class representatives for the subobjects of S.

Thus subobjects, given by a categorical definition, are not the same as subsets, but each subset determines and is determined by a unique subobject.

Because of this close relationship, one frequently says, of objects A and B in a category, 'Let A be a subobject of B', meaning that one has in mind a certain equivalence class of monomorphisms that in particular contains a monomorphism $A \rightarrowtail B$. You should be aware that there may be many other monomorphisms from A to B that are not in the equivalence class, just as from any subset A of a set B there are generally many injective functions from A to B other than the inclusion.

2.8.13 As a consequence of the properties of the subobject construction, categorists take a different attitude toward substructures such as subsets and submonoids, as compared to many other mathematicians. For them, A is a subobject or substructure of B if there is a monomorphism from A to B, and the subobject is the equivalence class determined by that monomorphism. For example, let $\mathbf{Z}$ denote the set of integers and $\mathbf{R}$ the set of real numbers. In calculus classes, $\mathbf{Z}$ is a subset of $\mathbf{R}$; an integer actually is a real number. For the categorist, it suffices that there be a monic (injective) map from $\mathbf{Z}$ to $\mathbf{R}$.

That monic map is a kind of *type conversion*. (See [Reynolds, 1980] for a more general view.) An integer need not actually be thought of as a real number, but there is a standard or canonical way (translate this statement as 'a monic map') to regard an integer as a real number. This mapping is regarded by the categorist as an inclusion, even though in fact it may change what the integer really is.

In a computer language, converting an integer to a real may increase the storage allotted to it and change its representation. Something similar happens in many approaches to the foundations of mathematics, where real numbers are constructed from integers by a complicated process (Dedekind cuts or Cauchy sequences), which results in an embedding of the integers in the real numbers. Just as for computer languages, this embedding changes the form of an integer: instead of whatever it was before, it is now a Dedekind cut (or Cauchy sequence).

In traditional texts on foundations, this construction had to be modified to replace the image of each integer by the actual integer, so that the integers were actually inside the real numbers. From the categorical point of view, this is an unnecessary complication. This change results in replacing a monomorphism $\mathbf{Z} \longrightarrow \mathbf{R}$ by an equivalent monomorphism (one that determines the same subobject). From an *operational* point of view, the integers behave the same way whether this change is made or not.

2.8.14 Categories and typing In category theory, the inclusion map is usually made explicit. From the computing science point of view, category theory is a very strongly typed language, more strongly typed than any computer language. For example, the strict categorist will refer explicitly to the inclusion map from the nonzero real numbers to the set of all real numbers when talking of division. In a computer language this would correspond to having two different types, `REAL` and `NONZERO_REAL`, set up in such a way that you can divide a `REAL` only by a `NONZERO_REAL`. To multiply a `REAL` by a `NONZERO_REAL`, the strong typing would require you to convert the latter to a `REAL` first.

To be sure, categorists themselves are not always so strict; but when they are not strict they are aware of it. (Compare the comments in 1.2.2. Nor is this discussion meant to imply that computer languages should have such strict typing: rather, the intention is to illustrate the way category theory handles types.)

2.8.15 Exercises

1. a. Show that if an arrow is a monomorphism in a category, it is a monomorphism in any subcategory it happens to be in.

b. Give an example showing that a monomorphism in a subcategory need not be a monomorphism in the category containing the subcategory. (Hint: Look at small finite categories.)

2. Show that if $f : A \longrightarrow B$ and $g : B \longrightarrow C$ are monomorphisms, so is $g \circ f$.

3. Show that there are categories in which for some arrows f and g, $g \circ f$ is a monomorphism but g is not. (Compare Proposition 2.8.6.)

4. Prove the statements in 2.8.12.

5. Show that if $h : C \longrightarrow D$ is an isomorphism, then $h \sim \mathrm{id}_D$ (as defined in 2.8.9).

6. Show that an initial object has no proper subobjects.

7. Show that if A is a subobject of a terminal object and B is any object then there is at most one arrow from B to A. Conclude that any arrow from A is monic.

8. Find all the subobjects of the terminal object in each category:
a. Set.
b. The category of graphs and graph homomorphisms.
c. The category of monoids and monoid homomorphisms.

9.† Give an explicit description of the monomorphisms in **Rel**.

2.9 Other types of arrow

2.9.1 Epimorphisms Epimorphisms in a category are the same as monomorphisms in the dual category. So $f : S \to T$ is an epimorphism if for any arrows $g, h : T \to X$, $g \circ f = h \circ f$ implies $g = h$. An epimorphism is said to be **epic** or an **epi**, and may be denoted with a double-headed arrow, as in $f : S \twoheadrightarrow T$.

2.9.2 Proposition *A set function is an epimorphism in* **Set** *if and only if it is surjective.*

Proof. Suppose $f : S \to T$ is surjective, and $g, h : T \to X$ are two functions. If $g \neq h$, then there is some particular element $t \in T$ for which $g(t) \neq h(t)$. Since f is surjective, there is an element $s \in S$ for which $f(s) = t$. Then $g(f(s)) \neq h(f(s))$, so that $g \circ f \neq h \circ f$.

Conversely, suppose f is *not* surjective. Then there is some $t \in T$ for which there is no $s \in S$ such that $f(s) = t$. Now define two functions $g : T \to \{0, 1\}$ and $h : T \to \{0, 1\}$ as follows:

(i) $g(x) = h(x) = 0$ for all x in T except t.

(ii) $g(t) = 0$.

(iii) $h(t) = 1$.

Then $g \neq h$ but $g \circ f = h \circ f$, so f is not an epimorphism. □

2.9.3 In contrast to the situation with monomorphisms, epimorphisms in categories of sets with structure are commonly not surjective. For example the nonnegative integers and the integers are both monoids under addition, and the inclusion function i is a homomorphism which is certainly not surjective. However, it is an epimorphism.

Here is the proof: any homomorphism h whose domain is the integers is determined completely by its value $h(1)$. For positive m, $m = 1 + 1 + \cdots + 1$, so

$$h(m) = h(1 + 1 + \cdots + 1) = h(1)h(1) \cdots h(1)$$

where we write the operation in the codomain as juxtaposition. Also, $h(-1)$ is the inverse of $h(1)$, since

$$h(1)h(-1) = h(-1)h(1) = h(-1 + 1) = h(0)$$

which must be the identity of the codomain. Since an element of a monoid can have only one inverse, this means $h(-1)$ is uniquely determined by $h(1)$. Then since every negative integer is a sum of -1's, the value of h at every negative integer is also determined by its value at 1.

Now suppose that g and h are two homomorphisms from the monoid of integers into the same codomain. Then g and h are both determined by their value at 1. Since 1 is a positive integer, this means that if $g \circ i = h \circ i$, then $g = h$. Thus i is an epimorphism.

2.9.4 Proposition *Let $f : A \longrightarrow B$ and $g : B \longrightarrow C$. Then*

(a) *If f and g are epimorphisms, so is $g \circ f$.*

(b) *If $g \circ f$ is an epimorphism, so is g.*

Proof. This is the dual of Proposition 2.8.6.

2.9.5 In **Set** an arrow that is both monic and epic is bijective (Theorems 2.8.3 and 2.9.2), and hence an isomorphism. In general, this need not happen. One example is the inclusion of $\mathbf{N}$ in $\mathbf{Z}$ in **Mon** described in 2.9.3 (an inverse would also have to be an inverse in **Set**, but there isn't one since the inclusion is not bijective). An easier example is the arrow from C to D in the category $\mathbf{2}$ in (2.1). It is both monic and epic (vacuously) but there is no arrow from D to C so it is not an isomorphism because there is no arrow in the category that could be its inverse.

2.9.6 An arrow $f : A \longrightarrow B$ in a category is an isomorphism if it has an inverse $g : B \longrightarrow A$ which must satisfy both the equations $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. If it only satisfies the second equation, $f \circ g = \mathrm{id}_B$, then f is a **left inverse** of g and (as you might expect) g is a **right inverse** of f.

2.9.7 Definition Suppose f has a right inverse g. Then f is called a **split epimorphism** (f is "split by g") and g is called a **split monomorphism**.

A split epimorphism is indeed an epimorphism: if $h \circ f = k \circ f$ and f has a right inverse g, then $h = h \circ f \circ g = k \circ f \circ g = k$, which is what is required for f to be an epimorphism. A dual proof shows that a split monomorphism is a monomorphism.

Using the usual axioms of set theory, every surjection in **Set** is a split epimorphism. For if $f : A \longrightarrow B$, then choose, for each $b \in B$, some element $a \in A$ such that $f(a) = b$. The existence of such an a is guaranteed by surjectivity. Define $g(b)$ to be a. Then $f(g(b)) = f(a) = b$ for any $b \in B$, so $f \circ g = \mathrm{id}_B$.

> The so-called axiom of choice is exactly what is required to make all those generally infinitely many choices. And in fact, one possible formulation of the axiom of choice is that every epimorphism split.

Epimorphisms in other categories may not be split. The function that includes the monoid of nonnegative integers on addition in the monoid of all the integers on addition, which we mentioned in 2.9.3, certainly does not have a right inverse in the category of monoids, since it does not have a right inverse in the category of sets. There are plenty of examples of epimorphisms of monoids which *are* surjective which have no right inverse in the category of monoids, although of course they do in the category of sets (Exercise 2).

Unlike epis, which always split in the category of sets, monics in **Set** do not always split. Every arrow out of the empty set is monic and, save for the identity of $\emptyset$ to itself, is not split. On the other hand, every monic with nonempty source does split. We leave the details to you.

2.9.8 Hom sets The elementary categorical definitions given in the last section and this one can all be phrased in terms of hom set. In any category, $\mathrm{Hom}(A, B)$ is the set of arrows with source A and target B.

Thus a terminal object 1 satisfies the requirement that $\mathrm{Hom}(A, 1)$ is a singleton set for every object A, and an initial object 0 satisfies the dual requirement that $\mathrm{Hom}(0, A)$ is always a singleton. And $\mathrm{Hom}(1, A)$ is the set of constants (global elements) of A.

2.9.9 If $f : B \longrightarrow C$, f induces a set function

$$\mathrm{Hom}(A, f) : \mathrm{Hom}(A, B) \longrightarrow \mathrm{Hom}(A, C)$$

defined by composing by f on the left: for any $g \in \mathrm{Hom}(A, B)$, that is, for any $g : A \longrightarrow B$, $\mathrm{Hom}(A, f)(g) = f \circ g$, which does indeed go from A to C. (Compare Exercise 1 of Section 1.2.)

Similarly, for any object D, $f : B \longrightarrow C$ induces a set function

$$\mathrm{Hom}(f, D) : \mathrm{Hom}(C, D) \longrightarrow \mathrm{Hom}(B, D)$$

(note the reversal) by defining $\mathrm{Hom}(f, D)(h) = h \circ f$ for $h \in \mathrm{Hom}(C, D)$.

In terms of these functions, we can state this proposition, which we leave to you to prove.

2.9.10 Proposition *An arrow $f : B \longrightarrow C$ in a category*

(i) *is a monomorphism if and only if* $\mathrm{Hom}(A, f)$ *is injective for every object A;*

(ii) *is an epimorphism if and only if* $\mathrm{Hom}(f, D)$ *is injective (!) for every object D;*

(iii) *is a split monomorphism if and only if* $\mathrm{Hom}(f, D)$ *is surjective for every object D;*

(iv) *is a split epimorphism if and only if* $\mathrm{Hom}(A, f)$ *is surjective for every object* A*;*

(v) *is an isomorphism if and only if any one of the following equivalent conditions holds:*

(a) *it is both a split epi and a mono;*

(b) *it is both an epi and a split mono;*

(c) $\mathrm{Hom}(A, f)$ *is bijective for every object* A*;*

(d) $\mathrm{Hom}(f, A)$ *is bijective for every object* A*.*

Although many categorical definitions can be given in terms of hom sets, no categorical definition *must* be; in fact, some mathematicians consider category theory to be a serious alternative to set theory as a foundation for mathematics (see many works of Lawvere, including [1963] and [1966] as well as [McLarty, 1989]) and for that purpose (which is not our purpose, of course), definition in terms of hom sets or any other sets must be avoided.

2.9.11 Discussion Categorical definitions, as illustrated in the simple ideas of Sections 2.7, 2.8 and 2.9, provide a method of abstract specification which has proved very useful in mathematics. They have, in particular, clarified concepts in many disparate branches of mathematics and provided as well a powerful unification of concepts across these branches.

The method of categorical definition is close in spirit to the modern attitude of computing science that programs and data types should be specified abstractly before being implemented and that the specification should be kept conceptually distinct from the implementation. We believe that the method of categorical definition is a type of abstract specification which is suitable for use in many areas of theoretical computing science. This is one of the major themes of this book.

When a category $\mathscr{C}$ is a category of sets with structure, with the arrows being functions which preserve the structure, a categorical definition of a particular property does not involve the elements (in the standard sense of set theory) of the structure. Such definitions are said to be **element-free**, and that has been regarded as a great advantage of category theory.

Nevertheless, as we have seen, some definitions can be phrased in terms of *variable elements*. This allows us the option of using familiar modes of thinking about things in terms of elements even in general categories. In the case of the definition of monomorphism 2.8.2, the definition phrased in terms of variable elements is identical with the definition in **Set**. On the other hand, an epimorphism f (see 2.9.1) is a variable element with the property that any two different arrows out of its target must have different values at f. In some sense it is a variable element with a lot of variation.

This is an example of a situation where the variable element point of view is not very familiar.

The idea of variable element has much in common with the way mathematicians and physicists once thought of variable quantities. Perhaps thirty years from now the variable element idea will be much more pervasive and the idea that an epimorphism is an element with a lot of variation will be the natural way to describe it.

2.9.12 Exercises

1. Show that a surjective monoid homomorphism is an epimorphism.

2. Let $\mathbf{Z}_n$ denote the monoid of integers $(\mathrm{mod}\, n)$ with addition $(\mathrm{mod}\, n)$ as operation. Show that the map $\phi : \mathbf{Z}_4 \to \mathbf{Z}_2$ that takes 0 and 2 to 0 and 1 and 3 to 1 is a surjective monoid epimorphism and is not split.

3. Let M be a monoid.

a. Show that if M is finite then an element is a monomorphism in $C(M)$ if and only if it is an epimorphism in $C(M)$ if and only if it is an isomorphism in $C(M)$.

b. Give an example showing that the assumption of finiteness in (a) cannot be relaxed.

4. Show that in the category $C(P)$ determined by a poset P, the only split epis or split monos are the identity arrows.

5. Show that if h has a left inverse g, then $h \circ g$ is a split idempotent (see Exercise 10 of Section 2.7).

6. Prove Proposition 2.9.10. (For (iii), set $D = B$.)

7.† Show that in the category of graphs and graph homomorphism, a homomorphism $f : \mathscr{G} \to \mathscr{H}$ has any of the following properties if and only if both $f_0 : G_0 \to H_0$ and $f_1 : G_1 \to H_1$ (which are set functions) have the property in **Set**:

(i) epic;

(ii) monic;

(iii) an isomorphism.

8. An epimorphism $f : A \to B$ in a category is **extremal** if whenever $f = m \circ g$ where m is monic implies m is an isomorphism.

a. Show that a split epi is extremal.

b. Let $\mathscr{C}$ be a category in which every arrow that is both monic and epic is an isomorphism. Show that every epimorphism in $\mathscr{C}$ is extremal.

2.10 Factorization systems

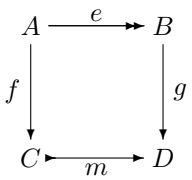
It is a familiar fact that every function in the category of sets can be factored as an epimorphism (surjection) followed by a monomorphism (injection). Similarly, every homomorphism in the category of abelian groups can be factored as an epimorphism followed by a monomorphism. The properties of these factorizations were abstracted early in the days of category theory, where they were known as bicategory structures. Under the name factorization system, they have a number of uses in category theory.

2.10.1 Definition A **factorization system** in a category $\mathscr{C}$ consists of two subclasses $\mathscr{E}$ and $\mathscr{M}$ of the arrows of $\mathscr{C}$ subject to the conditions

FS—-1 If $\mathscr{I}$ is the class of isomorphisms, then $\mathscr{M} \circ \mathscr{I} \subseteq \mathscr{M}$ and $\mathscr{I} \circ \mathscr{E} \subseteq \mathscr{E}$.

FS—-2 Every arrow f in $\mathscr{C}$ factors as $f = m \circ e$ with $m \in \mathscr{M}$ and $e \in \mathscr{E}$.

FS—-3 In any commutative square



with $e \in \mathscr{E}$ and $m \in \mathscr{M}$, there is a unique $h : B \to C$ such that $h \circ e = f$ and $m \circ h = g$.

The last condition is referred to as the "diagonal fill-in". If (as is the case in many examples) either every arrow in $\mathscr{M}$ is monic or every arrow in $\mathscr{E}$ is epic, then the uniqueness requirement in this condition may be omitted (Exercise 2).

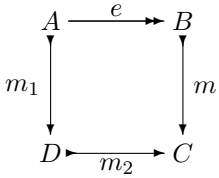
In discussing categories with factorization systems the usual convention is to denote an element of $\mathscr{M}$ with a tailed arrow and an element of $\mathscr{E}$ with a double-headed arrow. This may conflict with the conventions described after Definitions 2.8.2 and 2.9.1.

2.10.2 Example In **Set**, the class $\mathscr{E}$ of epimorphisms and the class $\mathscr{M}$ of monomorphisms constitute a factorization system. In many categories of algebraic structures (including monoids), the class $\mathscr{E}$ of regular epimorphisms (defined in Section 9.4.3) and the class $\mathscr{M}$ of monomorphisms constitute a factorization system. In the category of monoids and monoid homomorphisms, the class of all epis and all monos is not a factorization system.

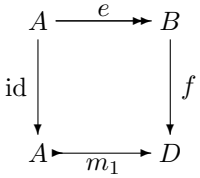
In the rest of this section, we assume $\mathscr{E}$ and $\mathscr{M}$ constitute a factorization system.

2.10.3 Proposition *The classes $\mathscr{E}$ and $\mathscr{M}$ are each closed under composition.*

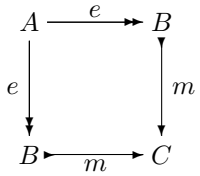
Proof. Suppose $m_1 : A \rightarrowtail D$ and $m_2 : D \rightarrowtail C$ with m_1 and m_2 in $\mathscr{M}$. Factor $m_2 \circ m_1 = m \circ e$ with $m \in \mathscr{M}$ and $e \in \mathscr{E}$. The diagonal fill-in in the square



is an arrow $f : B \to D$ such that $f \circ e = m_1$ and $m_2 \circ f = m$. The diagonal fill-in in the square



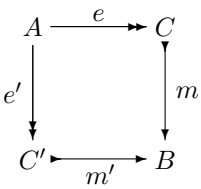
is a map $g : B \to A$ such that $g \circ e = \mathrm{id}$ and $m_1 \circ g = f$. Since $m \circ e \circ g = m_2 \circ m_1 \circ g = m_2 \circ f = m = m \circ \mathrm{id}$ and $e \circ g \circ e = e = \mathrm{id} \circ e$ both the identity and $e \circ g$ supply a diagonal fill-in in the square



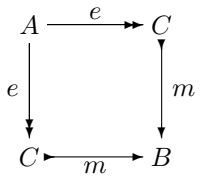
and hence, by uniqueness, are equal. This shows that e is an isomorphism and hence that $m_2 \circ m_1 = m \circ e \in \mathscr{M}$. The argument for $\mathscr{E}$ is dual. □

2.10.4 Proposition *If $f : A \to B$ factors as $A \xrightarrow{e} C \xrightarrow{m} B$ and also as $A \xrightarrow{e'} C' \xrightarrow{m'} B$ then there is a unique arrow $g : C \to C'$ such that $g \circ e = e'$ and $m' \circ g = m$; moreover g is an isomorphism.*

Proof. The arrow g is the diagonal fill-in in the square

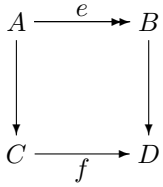


To see that g is an isomorphism, we transpose the square to get a map $g' : C' \to C$ such that $g' \circ e' = e$ and $m \circ g' = m'$. Then we note that these equations imply that both the identity and $g' \circ g$ fill in the square

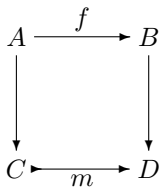


and the uniqeness of the diagonal fill-in forces $g' \circ g = \mathrm{id}$ and similarly $g \circ g' = \mathrm{id}$. □

2.10.5 Proposition *Suppose $f : C \to D$ satisfies the condition that for all $e : A \twoheadrightarrow B$ in $\mathscr{E}$, any commutative square*

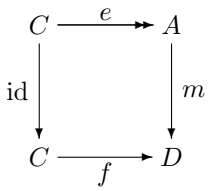


has a unique diagonal fill-in. Then $f \in \mathscr{M}$. Dually, if $g : A \to B$ satisfies the condition that for all $m : C \rightarrowtail D$ in $\mathscr{M}$, any commutative square



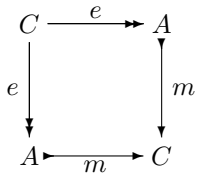
has a unique diagonal fill-in. Then $g \in \mathscr{E}$.

Proof. Factor f as $C \overset{e}{\twoheadrightarrow} A \overset{m}{\rightarrowtail} D$. From the diagonal fill-in in the square



we get a map $g : A \to C$ such that $g \circ e = \mathrm{id}$ and $f \circ g = m$. Then from $e \circ g \circ e = e$ and $m \circ e \circ g = f \circ g = m$ we see that both the identity and

$e \circ g$ fill in the diagonal of

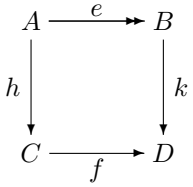


The uniqueness of the diagonal fill-in then implies that e is an isomorphism, whence $f = m \circ e \in \mathscr{M}$ by FS–1. □

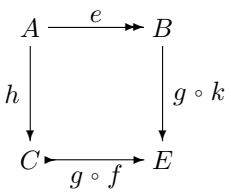
2.10.6 Corollary *Every isomorphism is in $\mathscr{E} \cap \mathscr{M}$.* □

2.10.7 Proposition *Suppose every arrow in $\mathscr{E}$ is an epimorphism. Then $g \circ f \in \mathscr{M}$ implies that $f \in \mathscr{M}$.*

Proof. Suppose the composite $C \xrightarrow{f} D \xrightarrow{g} E$ is in $\mathscr{M}$. Suppose we have a commutative square



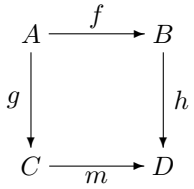
The diagonal fill-in in the square



provides a map $l : B \longrightarrow C$ such that $l \circ e = h$. Then e can be cancelled on the right of $f \circ l \circ e = f \circ h = k \circ e$ to conclude that $f \circ l = k$. If l' were another choice, e can be cancelled from $l \circ e = l' \circ e$.

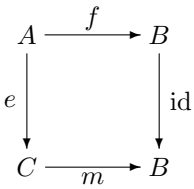
2.10.8 Proposition *Suppose $\mathscr{E}/\mathscr{M}$ is a factorization system on the category $\mathscr{C}$. Suppose $f : A \longrightarrow B$ is an arrow of $\mathscr{C}$ such that for all $m : C \longrightarrow D$*

in $\mathscr{M}$, and any commutative square

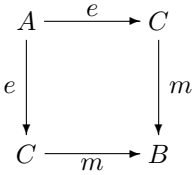


there is a diagonal fill-in (not even assumed unique) $l : B \longrightarrow C$ making both triangles commute. Then $f \in \mathscr{E}$.

Proof. Factor f as $A \xrightarrow{e} C \xrightarrow{m} B$ with $m \in \mathscr{M}$ and $e \in \mathscr{E}$. In the diagram



the diagonal fill-in property of f implies the existence of an arrow $l : B \longrightarrow C$ such that $l \circ f = e$ and $m \circ l = \mathrm{id}$. The diagram



has the identity as diagonal fill-in. But also $l \circ m \circ e = l \circ f = e$ and $m \circ l \circ m = m$. Thus the uniqeness of the diagonal fill-in (which is assumed to hold the top arrow is in $\mathscr{E}$ and the bottom arrow is in $\mathscr{M}$) forces $l \circ m = \mathrm{id}$ which implies that m is an isomorphism and then $f = m \circ e \in \mathscr{E}$. □

Of course, the dual property characterizes the arrows of $\mathscr{M}$.

More properties of factorization systems will be explored in Chapter 9.

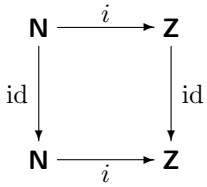
2.10.9 Exercises

1. Show that every category has a factorization system in which $\mathscr{E}$ consists of all arrows and $\mathscr{M}$ of all isomorphisms. (Switching the roles of $\mathscr{E}$ and $\mathscr{M}$ gives another one. Thus every category in which not every arrow is an isomorphism has at least two distinct factorization systems.)

2. Let $(\mathscr{E}, \mathscr{M})$ be a factorization system. Show that if either every arrow in $\mathscr{M}$ is monic or every arrow in $\mathscr{E}$ is epic then the requirement that h be unique in FS–3 of Definition 2.10.1 may be omitted.

3. Show that the class $\mathscr{E}$ of epimorphisms and $\mathscr{M}$ of monomorphisms constitute an epi-mono factorization system in **Set**.

4. Let $\mathbf{Z}$ be the monoid of all integers on addition, and $\mathbf{N}$ the monoid of all nonnegative integers on addition. Let $i : \mathbf{N} \to \mathbf{Z}$ be inclusion (which is both monic and epic, see Section 2.9.3). Show that the diagram



commutes but has no diagonal fill-in. Hence the set $\mathscr{E}$ of all epimorphisms and the set $\mathscr{M}$ of all monomorphisms do not constitute a factorization system in the category of monoids.

3
Functors

A functor F from a category $\mathscr{C}$ to a category $\mathscr{D}$ is a graph homomorphism which preserves identities and composition. It plays the same role as monoid homomorphisms for monoids and monotone maps for posets: it preserves the structure that a category has. Functors have another significance, however: since one sort of thing a category can be is a mathematical workspace (see Preface), many of the most useful functors used by mathematicians are *transformations from one type of mathematics to another.*

Less obvious, but perhaps more important, is the fact that many categories that are mathematically interesting appear as categories whose objects are a natural class of functors into the category of sets. This point of view will be explored in the chapters on sketches.

The first three sections define functors, give examples and describe some properties functors may have. Section 3.4 defines the concept of equivalence of categories, which captures the idea that two categories are the same from the categorical point of view. The last section concerns quotients of categories, which have quotients of monoids as special cases. This concept is used only in the chapters on sketches (and in 4.1.13, which itself is used only for sketches).

3.1 Functors

A functor is a structure-preserving map between categories, in the same way that a homomorphism is a structure-preserving map between graphs or monoids. Here is the formal definition.

3.1.1 Definition A **functor** $F : \mathscr{C} \to \mathscr{D}$ is a pair of functions $F_0 : \mathscr{C}_0 \to \mathscr{D}_0$ and $F_1 : \mathscr{C}_1 \to \mathscr{D}_1$ for which

F–1 If $f : A \to B$ in $\mathscr{C}$, then $F_1(f) : F_0(A) \to F_0(B)$ in $\mathscr{D}$.

F–2 For any object A of $\mathscr{C}$, $F_1(\mathrm{id}_A) = \mathrm{id}_{F_0(A)}$.

F–3 If $g \circ f$ is defined in $\mathscr{C}$, then $F_1(g) \circ F_1(f)$ is defined in $\mathscr{D}$ and $F_1(g \circ f) = F_1(g) \circ F_1(f)$.

By F–1, a functor is in particular a homomorphism of graphs. Following the practice for graph homomorphisms, the notation is customarily overloaded (see 1.4.2): if A is an object, $F(A) = F_0(A)$ is an object, and if f is an arrow, $F(f) = F_1(f)$ is an arrow. The notation for the constituents $F_0 : \mathscr{C}_0 \to \mathscr{D}_0$ and $F_1 : \mathscr{C}_1 \to \mathscr{D}_1$ is not standard, and we will use it only for emphasis.

3.1.2 Example It is easy to see that a monoid homomorphism $f : M \to N$ determines a functor from $C(M)$ to $C(N)$ as defined in 2.3.12. On objects, a homomorphism f must take the single object of $C(M)$ to the single object of $C(N)$, and F–1 is trivially verified since all arrows in $C(M)$ have the same domain and codomain and similarly for $C(N)$. Then F–2 and F–3 say precisely that f is a monoid homomorphism. Conversely, every functor is determined in this way by a monoid homomorphism.

3.1.3 Example Let us see what a functor from $C(S, \alpha)$ to $C(T, \beta)$ must be when (S, α) and (T, β) are posets as in 2.3.1. It is suggestive to write both relations α and β as '$\leq$' and the posets simply as S and T. Then there is exactly one arrow from x to y in S (or in T) if and only if $x \leq y$; otherwise there are no arrows from x to y.

Let $f : S \to T$ be the functor. F–1 says if there is an arrow from x to y, then there is an arrow from $f(x)$ to $f(y)$; in other words,

$$\text{if } x \leq y \text{ then } f(x) \leq f(y)$$

Thus f is a monotone map (see 2.4.2). F–2 and F–3 impose no additional conditions on f because they each assert the equality of two specified arrows between two specified objects and in a poset as category all arrows between two objects are equal.

3.1.4 Example If $\mathscr{C}$ is a category, the functor

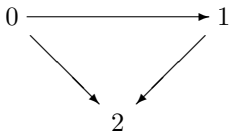
$$P_1 : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$$

(see 2.6.6) which takes an object (C, D) to C and an arrow $(f, g) : (C, D) \to (C', D')$ to f is called the **first projection**. There is an analogous second projection functor P_2 taking an object or arrow to its second coordinate.

3.1.5 Example Let $\mathbf{2} + \mathbf{2}$ be the category that can be pictured as

$$0 \to 1 \qquad 1' \to 2$$

with no other nonidentity arrows, and the category **3** the one that looks like



(3.1)

Define the functor $F : \mathbf{2} + \mathbf{2} \to \mathbf{3}$ to take 0 to 0, 1 and $1'$ to 1, and 2 to 2. Then what it does on arrows is forced.

Note that the image of F includes all of $\mathscr{D}$ except the composite arrow from $0 \to 2$. This example shows that the image of a functor need not be a subcategory of the codomain category.

3.1.6 The category of categories The category **Cat** has all small categories as objects and all functors between such categories as arrows. The composite of functors is their composite as graph homomorphisms: if $F : \mathscr{C} \to \mathscr{D}$ and $G : \mathscr{D} \to \mathscr{E}$, then $G \circ F : \mathscr{C} \to \mathscr{E}$ satisfies $G \circ F(C) = G(F(C))$ for any object C of $\mathscr{C}$, and $G \circ F(f) = G(F(f))$ for any arrow f of $\mathscr{C}$. Thus $(G \circ F)_i = G_i \circ F_i$ for $i = 0, 1$.

We note that the composition circle is usually omitted when composing functors so that we write $GF(C) = G(F(C))$.

It is sometimes convenient to refer to a category **CAT** which has all small categories and ordinary large categories as objects, and functors between them. Since trying to have **CAT** be an object of itself would raise delicate foundational questions, we do not attempt here a formal definition of **CAT**.

3.1.7 Example The inclusion map of a subcategory is a functor. As we pointed out in 2.8.13, the categorical point of view does not require that the object and arrows of a subcategory actually be objects and arrows of the bigger category, only that there be a monomorphism from the subcategory to the category. For example, **Set** is a subcategory of **Rel**: the monomorphic functor takes every set to itself and each function $f : S \to T$ to its graph $\{(s, t) \mid t = f(s)\}$, which is indeed a relation from S to T.

This approach has the strange result that two different categories can each be regarded as subcategories of the other one (Exercises 8 and 9).

3.1.8 Underlying functors Forgetting some of the structure in a category of structures and structure-preserving functions gives a functor called an **underlying functor** or **forgetful functor**. The functor $U : \mathbf{Mon} \to \mathbf{Sem}$ which embeds the category of monoids into the category of semigroups by forgetting that a monoid has an identity is an example of an underlying functor.

Another example is the functor which forgets *all* the structure of a semigroup. This is a functor $U : \mathbf{Sem} \to \mathbf{Set}$. There are lots of semigroups with the same set of elements; for example, the set $\{0, 1, 2\}$ is a semigroup on addition (mod 3) and also a different semigroup on multiplication (mod 3). The functor U applied to these two different semigroups gives the same set, so U is not injective on objects, in contrast to the forgetful functor from monoids to semigroups.

We will not give a formal definition of underlying functor. It is reasonable to expect any underlying functor U to be faithful (see 3.3.2 below) and that if f is an isomorphism and $U(f)$ is an identity arrow then f is an identity arrow.

3.1.9 Example A small graph has *two* underlying sets: its set of nodes and its set of arrows. Thus there is an underlying functor $U : \mathbf{Grf} \to \mathbf{Set} \times \mathbf{Set}$ for which for a graph $\mathscr{G}$, $U(\mathscr{G}) = (G_0, G_1)$; an arrowset functor $A : \mathbf{Grf} \to \mathbf{Set}$ which takes a graph to its set of arrows and a graph homomorphism to the corresponding function from arrows to arrows; and a similarly defined nodeset functor $N : \mathbf{Grf} \to \mathbf{Set}$ which takes a graph to its set of nodes.

3.1.10 Example If you forget you can compose arrows in a category and you forget which arrows are the identities, then you have remembered only that the category is a graph. This gives an underlying functor $U : \mathbf{Cat} \to \mathbf{Grf}$, since every functor is a graph homomorphism although not vice versa.

As for graphs, there are also set-of-objects and set-of-arrows functors $O : \mathbf{Cat} \to \mathbf{Set}$ and $A : \mathbf{Cat} \to \mathbf{Set}$ which take a category to its set of objects and set of arrows respectively, and a functor to the appropriate set map.

3.1.11 Example In 2.6.10, we described the notion of a slice category $\mathscr{C}/A$ based on a category $\mathscr{C}$ and an object A. An object is an arrow $B \to A$ and an arrow from $f : B \to A$ to $g : C \to A$ is an arrow $h : B \to C$ for which

$$g \circ h = f$$

There is a functor $U : \mathscr{C}/A \to \mathscr{C}$ that takes the object $f : B \to A$ to B and the arrow h from $B \to A$ to $C \to A$ to $h : B \to C$. This is called the underlying functor of the slice. In the case that $\mathscr{C} = \mathbf{Set}$, an object $T \to S$ of $\mathbf{Set}/S$ for some set S is an S-indexed object, and the effect of the underlying functor is to forget the indexing.

3.1.12 Free functors The **free monoid functor** from **Set** to the category of monoids takes a set A to the free monoid $F(A)$, which is the Kleene

closure A^* with concatenation as operation (see 2.3.9), and a function $f : A \to B$ to the function $F(f) = f^* : F(A) \to F(B)$ defined in 2.5.7.

To see that the free monoid functor is indeed a functor it is necessary to show that if $f : A \to B$ and $g : B \to C$, then $F(g \circ f) : F(A) \to F(C)$ is the same as $F(g) \circ F(f)$, which is immediate from the definition, and that it preserves identity arrows, which is also immediate.

The Kleene closure is itself a functor from **Set** to **Set**, taking A to A^* and f to f^*. It is the composite $U \circ F$ of the underlying functor $U : \mathbf{Mon} \to \mathbf{Set}$ and the free functor $F : \mathbf{Set} \to \mathbf{Mon}$, but of course it can be defined independently of U and F.

3.1.13 Example The free category on a graph is also the object part of a functor $F : \mathbf{Grf} \to \mathbf{Cat}$. What it does to a graph is described in 2.6.16. Suppose $\phi : \mathscr{G} \to \mathscr{H}$ is a graph homomorphism. The objects of the free category on a graph are the nodes of the graph, so it is reasonable to define $F(\phi)_0 = \phi_0$. Now suppose $(f_n, f_{n-1}, \ldots, f_1)$ is a path, that is, an arrow, in $F(\mathscr{G})$. Since functors preserve domain and codomain, we can define $F(\phi)_1(f_n, f_{n-1}, \ldots, f_1)$ to be $(\phi_1(f_n), \phi_1(f_{n-1}), \ldots, \phi_1(f_1))$ and know we get a path in $F(\mathscr{H})$. That F preserves composition of paths is also clear.

3.1.14 The map-lifting property The free category functor $F : \mathbf{Grf} \to \mathbf{Cat}$ and also other free functors, such as the free monoid functor (3.1.12), have a map lifting property called its **universal mapping property** which will be seen in Section 13.2 as the defining property of freeness. We will describe the property for free categories since we use it later. The free monoid case is done in detail in Proposition 13.1.2.

Let $\mathscr{G}$ be a graph and $F(\mathscr{G})$ the free category generated by $\mathscr{G}$. There is a graph homomorphism with the special name $\eta\mathscr{G} : \mathscr{G} \to U(F(\mathscr{G}))$ which includes a graph $\mathscr{G}$ into $U(F(\mathscr{G}))$, the underlying graph of the free category $F(\mathscr{G})$. The map $(\eta\mathscr{G})_0$ is the identity, since the objects of $F(\mathscr{G})$ are the nodes of $\mathscr{G}$. For an arrow f of $\mathscr{G}$, $(\eta\mathscr{G})_1(f)$ is the path (f) of length one. This is an inclusion arrow in the generalized categorical sense of 2.8.13, since f and (f) are really two distinct entities.

3.1.15 Proposition *Let $\mathscr{G}$ be a graph and $\mathscr{C}$ a category. Then for every graph homomorphism $h : \mathscr{G} \to U(\mathscr{C})$, there is a unique functor $\widehat{h} : F(\mathscr{G}) \to \mathscr{C}$ with the property that $U(\widehat{h}) \circ \eta\mathscr{G} = h$.*

Proof. If $()$ is the empty path at an object a, we set $\widehat{h}() = \mathrm{id}_a$. For an object a of $F(\mathscr{G})$ (that is, node of $\mathscr{G}$), define $\widehat{h}(a) = h(a)$. And for a path $(a_n, a_{n-1}, \ldots, a_1)$, $\widehat{h}$ is 'map h':

$$\widehat{h}(a_n, a_{n-1}, \ldots, a_1) = (h(a_n), h(a_{n-1}), \ldots, h(a_1))$$

As noted in 2.1.1, there is a unique empty path for each node a of $\mathscr{G}$. Composing the empty path at a with any path p from a to b gives p again, and similarly on the other side. That is why the program returns id_a for the empty path at a.

3.1.16 Powerset functors Any set S has a powerset $\mathscr{P}S$, the set of all subsets of S. There are three different functors F for which F_0 takes a set to its powerset; they differ on what they do to arrows. One of them is fundamental in topos theory; that one we single out to be called the powerset functor.

If $f : A \longrightarrow B$ is any set function and C is a subset of B, then the **inverse image** of C, denoted $f^{-1}(C)$, is the set of elements of A which f takes into C: $f^{-1}(C) = \{a \in A \mid f(a) \in C\}$. Thus f^{-1} is a function from $\mathscr{P}B$ to $\mathscr{P}A$.

Note that for a bijection f, the symbol f^{-1} is also used to denote the inverse function. Context makes it clear which is meant, since the input to the inverse image function must be a subset of the codomain of f, whereas the input to the actual inverse of a bijection must be an *element* of the codomain.

3.1.17 Definition The **powerset functor** $\mathscr{P} : \mathbf{Set}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ takes a set S to the powerset $\mathscr{P}S$, and a set function $f : A \longrightarrow B$ (that is, an arrow from B to A in $\mathbf{Set}^{\mathrm{op}}$) to the inverse image function $f^{-1} : \mathscr{P}B \longrightarrow \mathscr{P}A$.

Although we will continue to use the notation f^{-1}, it is denoted f^* in much of the categorical literature.

To check that $\mathscr{P}$ is a functor requires showing that $\mathrm{id}_A^{-1} = \mathrm{id}_{\mathscr{P}A}$ and that if $g : B \longrightarrow C$, then $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$, where both compositions take place in **Set**.

3.1.18 A functor $F : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathscr{D}$ is also called a **contravariant functor** from $\mathscr{C}$ to $\mathscr{D}$. As illustrated in the preceding definition, the functor is often defined in terms of arrows of $\mathscr{C}$ rather than of arrows of $\mathscr{C}^{\mathrm{op}}$. Opposite categories are most commonly used to provide a way of talking about contravariant functors as ordinary (**covariant**) functors: the opposite category in this situation is a purely formal construction of no independent interest (see 2.6.9).

3.1.19 The other two functors which take a set to its powerset are both covariant. The **direct** or **existential image** functor takes $f : A \longrightarrow B$ to the function $f_* : \mathscr{P}A \longrightarrow \mathscr{P}B$, where $f_*(A_0) = \{f(x) \mid x \in A_0\}$, the set of values of f on A_0. The **universal image** functor takes A_0 to those values of f which come *only* from A_0: formally, it takes $f : A \longrightarrow B$ to $f_! : \mathscr{P}A \longrightarrow \mathscr{P}B$, with

$$f_!(A_0) = \{y \in B \mid f(x) = y \text{ implies } x \in A_0\} = \{y \in B \mid f^{-1}(\{y\}) \subseteq A_0\}$$

3.1.20 Hom functors Let $\mathscr{C}$ be a category with an object C and an arrow $f : A \longrightarrow B$. In 2.9.9, we defined the function $\mathrm{Hom}(C, f) : \mathrm{Hom}(C, A) \longrightarrow \mathrm{Hom}(C, B)$ by setting $\mathrm{Hom}(C, f)(g) = f \circ g$ for every $g \in \mathrm{Hom}(C, A)$, that is for $g : C \longrightarrow A$. We use this function to define the **covariant hom functor** $\mathrm{Hom}(C, -) : \mathscr{C} \longrightarrow \mathbf{Set}$ as follows:

HF–1 $\mathrm{Hom}(C, -)(A) = \mathrm{Hom}(C, A)$ for each object A of $\mathscr{C}$;

HF–2 $\mathrm{Hom}(C, -)(f) = \mathrm{Hom}(C, f) : \mathrm{Hom}(C, A) \longrightarrow \mathrm{Hom}(C, B)$ for $f : A \longrightarrow B$.

The following calculations show that $\mathrm{Hom}(C, -)$ is a functor. For an object A, $\mathrm{Hom}(C, \mathrm{id}_A) : \mathrm{Hom}(C, A) \longrightarrow \mathrm{Hom}(C, A)$ takes an arrow $f : C \longrightarrow A$ to $\mathrm{id}_A \circ f = f$; hence $\mathrm{Hom}(C, \mathrm{id}_A) = \mathrm{id}_{\mathrm{Hom}(C,A)}$. Now suppose $f : A \longrightarrow B$ and $g : B \longrightarrow D$. Then for any arrow $k : C \longrightarrow A$,

$$\begin{aligned}
\Big(\mathrm{Hom}(C, g) \circ \mathrm{Hom}(C, f)\Big)(k) &= \mathrm{Hom}(C, g)\Big(\mathrm{Hom}(C, f)(k)\Big) \\
&= \mathrm{Hom}(C, g)(f \circ k) \\
&= g \circ (f \circ k) \\
&= (g \circ f) \circ k \\
&= \mathrm{Hom}(C, g \circ f)(k)
\end{aligned}$$

In terms of variable elements, $\mathrm{Hom}(C, f)$ takes the variable elements of A with parameter set C to the variable elements of B with parameter set C.

There is a distinct covariant hom functor $\mathrm{Hom}(C, -)$ for each object C. In this expression, C is a parameter for a family of functors. The argument of each of these functors is indicated by the dash. An analogous definition in calculus would be to define the function which raises a real number to the nth power as $f(-) = (-)^n$ (here n is the parameter). One difference in the hom functor case is that the hom functor is overloaded and so has to be defined on two different kinds of things: objects and arrows.

3.1.21 Definition For a given object D, the **contravariant hom functor**

$$\mathrm{Hom}(-, D) : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$$

is defined for each object A by

$$\mathrm{Hom}(-, D)(A) = \mathrm{Hom}(A, D)$$

and for each arrow $f : A \longrightarrow B$,

$$\mathrm{Hom}(-, D)(f) = \mathrm{Hom}(f, D) : \mathrm{Hom}(B, D) \longrightarrow \mathrm{Hom}(A, D)$$

Thus if $g : B \longrightarrow D$, $\mathrm{Hom}(f, D)(g) = g \circ f$.

3.1.22 Definition The **two-variable hom functor**

$$\mathrm{Hom}(-,-) : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \longrightarrow \mathbf{Set}$$

takes a pair (C, D) of objects of C to $\mathrm{Hom}(C, D)$, and a pair (f, g) of arrows with $f : C \longrightarrow A$ and $g : B \longrightarrow D$ to

$$\mathrm{Hom}(f, g) : \mathrm{Hom}(A, B) \longrightarrow \mathrm{Hom}(C, D)$$

where for $h : A \longrightarrow B$,

$$\mathrm{Hom}(f, g)(h) = g \circ h \circ f$$

which is indeed an arrow from C to D.

In this case we also use the product of categories as a formal construction to express functors of more than one argument. From the categorical point of view, a functor always has one argument, which as in the present case might well be an object in a product category (an ordered pair).

3.1.23 Exercises

1. Show that in the definition of functor, the clause '$F_1(g) \circ F_1(f)$ is defined in $\mathscr{D}$' can be omitted.

2. Describe the initial and terminal objects in the category of categories and functors.

3. Prove that the existential and universal image functors of 3.1.19 are functors.

4. a. Prove that a functor is a monomorphism in the category of categories if and only if it is injective on both objects and arrows. (Compare Exercise 7(ii) of Section 2.9. The corresponding statement for epimorphisms is not true (Section 3.3).)

b. Prove that the functor $U : \mathbf{Mon} \longrightarrow \mathbf{Sem}$ described in 3.1.8 is a monomorphism.

5. Given a semigroup S, construct a monoid $M = S \cup \{e\}$, using a new element e not in S and different for each semigroup S. For example, you could take $e = \{S\}$. The multiplication in M is defined this way:

(i) xy is the product in S if both x and y are in S.

(ii) $xe = ex = x$ for all $x \in M$.

M is denoted S^1 in the semigroup literature. Show that

a. S^1 is a monoid (note that if S is already a monoid, S^1 is too but with a new identity element);

b. there is a functor $F : \mathbf{Sem} \longrightarrow \mathbf{Mon}$ which takes each semigroup S to S^1 and each semigroup homomorphism $f : S \longrightarrow T$ to a monoid homomorphism $f^1 : S^1 \longrightarrow T^1$ which is the same as f on S and which takes the added element to the added element;

c. F is a monomorphism in **Cat**.

6. Let $U : \mathbf{Mon} \longrightarrow \mathbf{Set}$ be the underlying set functor, and $F : \mathbf{Set} \longrightarrow \mathbf{Mon}$ the free monoid functor. For every set A and monoid M, construct a function

$$\beta : \mathrm{Hom}_{\mathbf{Set}}(A, U(M)) \longrightarrow \mathrm{Hom}_{\mathbf{Mon}}(F(A), M)$$

by defining

$$\beta(f)(a_1 a_2 \cdots a_n) = f(a_1) f(a_2) \cdots f(a_n)$$

(the right side is the product in M) for $f : A \longrightarrow U(M)$. Show that β is a bijection.

7. Let $U : \mathbf{Mon} \longrightarrow \mathbf{Sem}$ be the functor of 3.1.7 and $F : \mathbf{Sem} \longrightarrow \mathbf{Mon}$ the functor of Exercise 5. Define a function

$$\gamma : \mathrm{Hom}_{\mathbf{Sem}}(S, U(M)) \longrightarrow \mathrm{Hom}_{\mathbf{Mon}}(F(S), M)$$

by $\gamma(h)(s) = h(s)$ if $s \in S$ and $\gamma(h)(e_S) = 1$ if e_S is the new element added to S to construct $F(S)$, where 1 is the identity element of M. Show that γ has the claimed codomain and is a bijection. (Compare Exercise 6.)

8. Show that each of **Mon** and **Sem** is a subcategory of the other.

9. Show that **Set** and **Pfn** are each subcategories of the other. (Hint: To construct a monic functor from **Pfn** to **Set**, take each set A to the set $A \cup \{A\}$, and a partial function $f : A \longrightarrow B$ to the full function f' for which $f'(x) = f(x)$ if $f(x)$ is defined, and $f'(x) = B$ otherwise.)

10.† Let $\mathscr{A}$ be a category. Show that $\mathscr{A}$ is discrete (see Exercise 11 of Section 2.7) if and only if every set function $F : \mathscr{A}_0 \longrightarrow \mathscr{B}_0$, where $\mathscr{B}$ is any *category*, is the object part of a unique functor from $\mathscr{A}$ to $\mathscr{B}$.

11.† A category $\mathscr{B}$ is **indiscrete** if every set function $F : \mathscr{A}_0 \longrightarrow \mathscr{B}_0$, where $\mathscr{A}$ is any category, is the object part of a unique functor from $\mathscr{A}$ to $\mathscr{B}$. Give a definition of 'indiscrete' in terms of the objects and arrows of $\mathscr{B}$.

3.2 Actions

In this section, we discuss set-valued functors as a natural generalization of finite state machines. This section is referred to only in Chapter 12 and in a few scattered examples. Set-valued functors also have theoretical importance in category theory because of the Yoneda Lemma (Section 4.5).

3.2.1 Monoid actions Let M be a monoid with identity 1 and let S be a set. An **action** of M on S is a function $\alpha : M \times S \longrightarrow S$ for which

A–1 $\alpha(1, s) = s$ for all $s \in S$.

A–2 $\alpha(mn, s) = \alpha(m, \alpha(n, s))$ for all $m, n \in M$ and $s \in S$.

It is customary in mathematics to write ms for $\alpha(m, s)$; then the preceding requirements become

A′–1 $1s = s$ for all $s \in S$.

A′–2 $(mn)s = m(ns)$ for all $m, n \in M$ and $s \in S$.

When actions are written this way, S is also called an M**-set**. The same syntax ms for $m \in M$ and $s \in S$ is used even when different actions are involved. This notation is analogous to (and presumably suggested by) the notation $c\mathbf{v}$ for scalar multiplication, where c is a scalar and $\mathbf{v}$ is a vector.

It is useful to think of the set S as a **state space** and the elements of M as acting to induce **transitions** from one state to another.

3.2.2 Definition Let M be a monoid with actions on sets S and T. An **equivariant map** from S to T is a function $\phi : S \longrightarrow T$ with the property that $m\phi(s) = \phi(ms)$ for all $m \in M$ and $s \in S$. The identity function is an equivariant map and the composite of two equivariant maps is equivariant. This means that for each monoid M, monoid actions and equivariant maps form a category M–**Act**.

3.2.3 Actions as functors Let α be an action of a monoid M on a set S. Let $C(M)$ denote the category determined by M as in 2.3.12. The action α determines a functor $F_\alpha : C(M) \longrightarrow$ **Set** defined by:

AF–1 $F_\alpha(*) = S$.

AF–2 $F_\alpha(m) = s \mapsto \alpha(m, s)$ for $m \in M$ and $s \in S$.

This observation will allow us to generalize actions to categories in 3.2.6.

3.2.4 Example One major type of action by a monoid is the case when the state space is a vector space and M is a collection of linear transformations closed under multiplication. However, in that case the linear structure (the fact that states can be added and multiplied by scalars) is extra structure which the definition above does not require. Our definition also does not require that there be any concept of continuity of transitions. Thus, the definition is very general and can be regarded as a *nonlinear, discrete* approach to state transition systems.

Less structure means, as always, that fewer theorems are true and fewer useful tools are available. On the other hand, less structure means that more situations fit the axioms, so that the theorems that are true and the tools that do exist work for more applications.

3.2.5 Example A particularly important example of a monoid action occurs in the study of finite state machines. Let A be a finite set, the **alphabet** of the machine, whose elements may be thought of as characters or tokens, and let S be another finite set whose elements are to be thought of as states. We assume there is a distinguished state $s_0 \in S$ called the **start state**, and a function $\phi : A \times S \longrightarrow S$ defining a transition to a state for each token in A and each state in S. Such a system $\mathscr{M} = (A, S, s_0, \phi)$ is a **finite state machine**. Note that there is no question of imposing axioms such as A–1 and A–2 because A is not a monoid.

Any string w in A^* induces a sequence of transitions in the machine $\mathscr{M}$ starting at the state s_0 and ending in some state s. Precisely, we define a function $\phi^* : A^* \times S \longrightarrow S$ by:

FA–1 $\phi^*((), s) = s$ for $s \in S$.

FA–2 $\phi^*((a)w, s) = \phi(a, \phi^*(w, s))$ for any $s \in S$, $w \in A^*$ and $a \in A$.

Recall that the free monoid $F(A)$ is the set A^* with concatenation as multiplication. The function ϕ^* as just defined is thus an action of $F(A)$ on S. The identity of A^* is the empty word () and by FA–1, $\phi^*((), a) = a$ for all $a \in A$, so A–1 follows. As for A–2, if we assume that

$$\phi^*(wv, m) = \phi^*(w, \phi^*(v, m))$$

for words w of length k, then

$$\begin{aligned}&\phi^*((a)wv, m) = \phi(a, \phi^*(wv, m))\\ &= \phi(a, \phi^*(w, \phi^*(v, m))) = \phi^*((a)w, \phi^*(v, m))\end{aligned}$$

The first and third equality are from the definition of ϕ, while the second is from the inductive hypothesis.

Finite state machines in the literature often have added structure. The state space may have a subset F of **acceptor states** (or **final states**). The subset L of A^* of strings which drive the machine from the start state to an acceptor state is then the set of strings, or language, which is **recognized** by the machine $\mathscr{M}$. This is the machine as **recognizer**. A compiler typically uses a finite state machine to recognize identifiers in the input file.

Another approach is to assume that the machine outputs a string of symbols (not necessarily in the same alphabet) for each state it enters or each transition it undergoes. This is the machine as **transducer**.

An elementary introduction to finite state machines may be found in [Lewis and Papadimitriou, 1981]. Two more advanced texts which use algebraic methods to study finite state machines (primarily as recognizers) are those by Eilenberg [1976] and Lallement [1979]. The latter book has many other applications of semigroup theory as well.

3.2.6 Set-valued functors as actions Suppose we wanted to extend the idea of an action by introducing typing. What would the result be?

To begin with, we would suppose that in addition to the state space S, there was a type set T and a function $\text{type} : S \to T$ that assigned to each element $s \in S$ an element $\text{type}(s) \in T$.

In describing the elements of M, one must say, for an $m \in M$ and $s \in S$, what is $\text{type}(ms)$. Moreover, it seems that one might well want to restrict the types of the inputs on which a given m acts. In fact, although it might not be strictly necessary in every case, it seems clear that we can, without loss of generality, suppose that each $m \in M$ acts on only one kind of input and produces only one kind of output. For if m acted on two types of output, we could replace m by two elements, one for each type. Thus we can imagine that there are two functions we will call input and output from M to T for which $\text{input}(m)$ is the type of element that m acts on and $\text{output}(m)$ is the type of $m(s)$ for an element s of type $\text{input}(m)$.

In the untyped case, we had that M was a monoid, but here it is clearly appropriate to suppose that $m_1 * m_2$ is defined only when $\text{output}(m_2) = \text{input}(m_1)$. It is reasonable to suppose that for each type t, there is an operation $1_t \in M$ whose input and output types are t and such that for any $m \in M$ of input type t, we have $m * 1_t = m$ and for any $m \in M$ of output type t, we have $1_t * m = m$.

As for the action, we will evidently wish to suppose that when $s \in S$ has type t and $m, m' \in M$ have input types t, t', respectively, and output types t', t'', respectively, then $m'(m(s)) = (m' * m)(s)$ and $1_t(s) = s$.

Now it will not have escaped the reader at this point that M and T together constitute a category $\mathscr{C}$ whose objects are the elements of T and arrows are the elements of M. The input and output functions are just the source and target arrows and the 1_t are the identities.

M and S make up exactly the data of a set-valued functor on $\mathscr{C}$. Define a functor $F : \mathscr{C} \to \mathbf{Set}$ by letting $F(t) = \{s \in S \mid \text{type}(s) = t\}$. If m is an arrow of $\mathscr{C}$, that is an element of M, let its input and output types be t and t', respectively. Then for F to be a functor, we require a function $F(m) : F(t) \to F(t')$. Naturally, we define $F(m)(s) = ms$, which indeed has type t'. The facts that F preserves composition and identities are an easy consequence of the properties listed above.

This construction can be reversed. Let $\mathscr{C}$ be a small category and suppose we have a functor $F : \mathscr{C} \to \mathbf{Set}$ for which $F(C)$ and $F(D)$ are disjoint whenever C and D are distinct objects of $\mathscr{C}$ (this disjointness requirement is necessary to have a category, but can be forced by a simple modification of F – see Exercise 6 of Section 4.5). Then we can let T be the set of objects of $\mathscr{C}$, M the set of arrows and $S = \bigcup_{t \in T} F(t)$. The rest of the definitions are evident and we leave them to the reader.

Thus if $\mathscr{C}$ is a small category, a functor $F : \mathscr{C} \to \mathbf{Set}$ is an action which generalizes the concept of monoid acting on a set.

3.2.7 Example For any given object C of a category $\mathscr{C}$, the hom functor $\text{Hom}(C, -)$ (see 3.1.20) is a particular example of a set-valued functor. When the category $\mathscr{C}$ is a monoid, it is the action by left multiplication familiar in semigroup theory. A theorem generalizing the Cayley theorem for groups is true, too (see 4.5.2).

3.2.8 Variable sets It may be useful to think of a set-valued functor $F : \mathscr{C} \to \mathbf{Set}$ as an action, not on a typed set, but on a single *variable* set. The objects of $\mathscr{C}$ form a parameter space for the variation of the set being acted upon. Another way of saying this is that each object of $\mathscr{C}$ is a *point of view*, that the set being acted upon looks different from different points of view, and the arrows of $\mathscr{C}$ are changes in point of view (as well as inducing transitions). See [Barr, McLarty and Wells, 1985].

3.2.9 Machines with typed actions The concept generalizing finite state machines is based on the perception that words in a typed alphabet are paths in a graph whose nodes are the types.

Formally, a **typed finite state machine** consists of a graph $\mathscr{G}$ and a graph homomorphism ϕ from $\mathscr{G}$ to the category of finite sets. Thus for each node n of $\mathscr{G}$ there is a set $\phi(n)$, and for each arrow $f : m \to n$ of $\mathscr{G}$ there is a function $\phi(f) : \phi(m) \to \phi(n)$.

What corresponds to the action of the free monoid on the states in the case of ordinary finite state machines is the action of the free category $F(\mathscr{G})$ generated by $\mathscr{G}$. The words in the free monoid are now paths in the free category. The action ϕ generates an action $\phi^* : F(\mathscr{G}) \to \mathbf{Set}$ according to this recursive definition, which is a precise generalization of Section 3.2.5.

FA′–1 $\phi^*(()_C)(x) = x$ for $x \in F(C)$, where $()_C$ denotes the empty path from C to C.

FA′–2 For $x \in \mathrm{dom}(f_1)$,

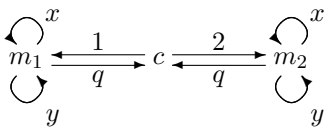
$$\phi^*(f_n, f_{n-1}, \ldots, f_1)(x) = \phi^*(f_n, \ldots, f_2)[\phi(f_1)(x)]$$

This in other notation is a special case of the functor $F(\phi)$ defined in 3.1.13.

3.2.10 Remark If you wanted to model nondeterminism, $\phi(f)$ for an arrow $f : m \longrightarrow n$ could not be a set function because the result of applying $\phi(f)$ to $\phi(m)$ might take a state of $\phi(m)$ to several states, or no state, of $\phi(n)$. One way to solve this would be to take the meaning of the action in **Rel** instead of in **Set**. Specifically, $\phi(m)$ would consist of the ordered pairs (x, y) with the property that the action of m *could* take x to y. If the machine stalls on x, then there would be no pair in $\phi(m)$ beginning with x.

This example is the tip of an iceberg. Probably the commonest type of research article that applies category theory to computing science is one that proposes some specific category other than **Set** to be the semantics of a certain kind of program or programming language.

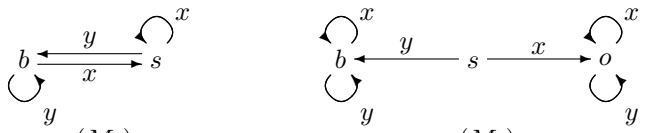
3.2.11 Example We can illustrate typed states by modeling an application program with modes. Let's suppose we have a program that can do two different things. It has three modes, one in which you choose what to do and the other two which perform the two tasks. For simplicity, we suppose that the two tasks are to recognize two languages L_1 and L_2 using finite state machines m_1 and m_2 respectively. This configuration can be exhibited as a graph $\mathscr{G}$:



(3.2)

This assumes the two languages are subsets of $\{x, y\}^*$. The two arrows labeled x are two different arrows with the same label. The same remark applies to y and q (which can be interpreted as 'quit').

To make an actual typed finite state machine using this schema, let us suppose that L_1 is the language of all those strings ending with x and L_2 is the language of all those strings that start with x. These are recognized by the following machines, in which s is the start state, s accepts in M_1 and o accepts in M_2. The states labeled b can be thought of as 'bad' and o as 'OK'.



(3.3)

Let $\mathscr{C}$ be the free category generated by $\mathscr{G}$. We define a functor $F : \mathscr{C} \longrightarrow \mathbf{Set}$ that implements the machine described in the previous paragraph. To avoid confusion, we put subscripts 1 or 2 on the start states and the states labeled b of the two machines to distinguish them. Then F is defined on objects by

(i) $F(c) = \{s, a, r\}$. s is 'start', a is 'accept' and r is 'reject'.
(ii) $F(m_1) = \{s_1, b_1\}$.
(iii) $F(m_2) = \{s_2, o, b_2\}$.

The values of F on arrows are exhibited in this table. The occurrence of r in the row marked b_1 and the column marked q means that $F(q)(b_1) = r$. A blank means the function is not defined at that symbol. This table is more compact than it might be because it takes advantage of the fact that two arrows each are labeled x, y and q.

	1	2	q	x	y
s	s_1	s_2			
a	s_1	s_2			
r	s_1	s_2			
s_1			a	s_1	b_1
b_1			r	s_1	b_1
s_2			r	o	b_2
o			a	o	o
b_2			r	b_2	b_2

You can check, for example, that feeding this machine the string $1yxxq$ causes it to wind up in state a, and feeding it $1yyq2xyyq$ causes it to wind up in state a after passing through state r at the first q. One could attach an output function to node c causing it to report acceptance or rejection. Note that many strings (for example any string not starting with 1 or 2) are not allowed as input because they violate the typing rules. One could add a node e (for error) to $\mathscr{G}$ with appropriate transitions to account for this. $F(e)$ might then have several states corresponding to different kinds of errors.

We have made some choices in saying that $F(1)$ and $F(2)$ can be applied to a and r. For example, we could have reconfigured the graph $\mathscr{G}$ to split the nodes c into two nodes, one for starting and one for reporting a result.

3.2.12 Exercise

1. Let S be a set. The **full transformation monoid** on S, denoted $FT(S)$, is the set of all functions from S to S with composition as the operation. Show that the following is equivalent to the definition in 3.2.1 of monoid action: an action by a monoid M on S is a monoid homomorphism from M to $FT(S)$.

3.3 Types of functors

Since **Cat** is a category, we already know about some types of functors. Thus a functor $F : \mathscr{C} \to \mathscr{D}$ is an isomorphism if there is a functor $G : \mathscr{D} \to \mathscr{C}$ which is inverse to F. This implies that F is bijective on objects and arrows and conversely a functor which is bijective on objects and arrows is an isomorphism.

We have already pointed out (Exercise 4 of Section 3.1) that a functor is a monomorphism in **Cat** if and only if it is injective on both objects and arrows. Epimorphisms in **Cat** need not be surjective, since the example in 2.9.3 is actually an epimorphism in **Cat** between the categories determined by the monoids (Exercise 8).

3.3.1 Full and faithful We will now consider properties of functors which are more intrinsic to **Cat** than the examples just given.

Any functor $F : \mathscr{C} \to \mathscr{D}$ induces a set mapping

$$\mathrm{Hom}_{\mathscr{C}}(A, B) \to \mathrm{Hom}_{\mathscr{D}}(F(A), F(B))$$

for each pair of objects A and B of $\mathscr{C}$. This mapping takes an arrow $f : A \to B$ to $F(f) : F(A) \to F(B)$.

3.3.2 Definition A functor $F : \mathscr{C} \to \mathscr{D}$ is **faithful** if the induced mapping is injective on every hom set.

Thus if $f : A \to B$ and $g : A \to B$ are different arrows, then $F(f) \neq F(g)$. However, it is allowed that $f : A \to B$ and $g : C \to D$ may be different arrows, with $F(A) = F(C)$, $F(B) = F(D)$ and $F(f) = F(g)$, provided that either $A \neq C$ or $B \neq D$.

3.3.3 Example Underlying functors are typically faithful. Two different monoid homomorphisms between the same two monoids must be different as set functions.

On the other hand, consider the set $\{0, 1, 2\}$. It has two different monoid structures via addition and multiplication (mod 3) (and many other monoid structures, too), but the two corresponding identity homomorphisms are the

same as set functions (have the same underlying function). Thus underlying functors need not be injective.

3.3.4 Definition A functor $F : \mathscr{C} \to \mathscr{D}$ is **full** if the induced mapping is surjective for every hom set.

A full functor need not be surjective on either objects or arrows. A full subcategory (2.6.3) is exactly one whose embedding is a full and faithful functor.

That the underlying functor from the category of semigroups to the category of sets is not full says exactly that not every set function between semigroups is a semigroup homomorphism. Note that this functor is surjective on objects, since every set can be made into a semigroup by letting $xy = x$ for every pair x and y of elements.

3.3.5 Example The functor $F : \mathscr{C} \to \mathscr{D}$ which takes A and B to C and X and Y to Z (and so is forced on arrows) in the picture below (which omits identity arrows) is *not* full. That is because $\mathrm{Hom}(A, B)$ is empty, but $\mathrm{Hom}(F(A), F(B)) = \mathrm{Hom}(C, C)$ has an arrow in it – the identity arrow. This functor is faithful even though not injective, since two arrows between the same two objects do not get identified.

(3.4)

3.3.6 Preservation of properties A functor $F : \mathscr{C} \to \mathscr{D}$ **preserves** a property P of arrows if whenever f has property P, so does $F(f)$.

3.3.7 Examples The fact that a monomorphism in the category of monoids must be injective can be worded as saying that the underlying functor preserves monomorphisms (since an injective function in **Set** is a monomorphism). The statement that an epimorphism in **Mon** need not be surjective is the same as saying that the underlying functor does not preserve epimorphisms.

As another example, consider the functor $F : \mathbf{2} \to \mathbf{Set}$ ($\mathbf{2}$ is shown in (2.1), page 18) defined by $C \mapsto \{1, 2\}$, $D \mapsto \{3, 4\}$ and the arrow from C to D going to the constant function $1 \mapsto 3$, $2 \mapsto 3$ from $F(C)$ to $F(D)$. The arrow from C to D is monic and epic (vacuously) but its value in **Set**

takes 1 and 2 both to 3, so is not injective and hence not a monomorphism. It is also not an epimorphism. Thus F preserves neither monomorphisms nor epimorphisms.

The story is different for isomorphisms. (Note that the arrow from C to D in **2** is not an isomorphism!)

3.3.8 Proposition *Every functor preserves isomorphisms.*

Proof. This is because the concept of isomorphism is defined in terms of equations involving composition and identity. If $f : A \longrightarrow B$ is an isomorphism with inverse g, then $F(g)$ is the inverse of $F(f)$. One of the two calculations necessary to prove this is that $F(g) \circ F(f) = F(g \circ f) = F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$; the other calculation is analogous. □

3.3.9 Definition A functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ **reflects** a property P of arrows if whenever $F(f)$ has property P then so does f (for *any* arrow that F takes to $F(f)$).

It follows from 2.5.5 and the definition of isomorphism (2.7.4) that a bijective semigroup homomorphism must be an isomorphism. That is the same as saying that the underlying functor from **Sem** to **Set** reflects isomorphisms. The same remark applies to **Mon**. The underlying functor from the category of posets and monotone maps does not reflect isomorphisms (see 2.7.11).

A full and faithful functor reflects isomorphisms, but in fact it does a bit more than that, as described by the following proposition.

3.3.10 Proposition *Let $F : \mathscr{C} \longrightarrow \mathscr{D}$ be full and faithful, and suppose A and B are objects of $\mathscr{C}$ and $u : F(A) \longrightarrow F(B)$ is an isomorphism in $\mathscr{D}$. Then there is a unique isomorphism $f : A \longrightarrow B$ for which $F(f) = u$.*

Proof. By fullness, there are arrows $f : A \longrightarrow B$ and $g : B \longrightarrow A$ for which $F(f) = u$ and $F(g) = u^{-1}$. Then

$$F(g \circ f) = F(g) \circ F(f) = u^{-1} \circ u = \mathrm{id}_{F(A)} = F(\mathrm{id}_A)$$

But F is faithful, so $g \circ f = \mathrm{id}_A$. A similar argument shows that $f \circ g = \mathrm{id}_B$, so that g is the inverse of f. □

3.3.11 Corollary *A full and faithful functor reflects isomorphisms.*

3.3.12 Corollary *Let $F : \mathscr{C} \longrightarrow \mathscr{D}$ be a full and faithful functor. If $F(A) = F(B)$ for objects A and B of $\mathscr{C}$, then A and B are isomorphic.*

Proof. Apply Proposition 3.3.10 to the identity arrow from $F(A)$ to $F(A) = F(B)$. □

3.3.13 You can also talk about a functor preserving or reflecting a property of objects. For example, since a terminal object in **Mon** is a one-element monoid and a one-element set is a terminal object, the underlying functor from **Mon** to **Set** preserves terminal objects. It also reflects terminal objects. It does not preserve initial objects, but it does reflect initial objects although vacuously: the empty set is the only initial object in **Set** and the underlying set of a monoid cannot be empty since it must have an identity element. We leave the details to you.

3.3.14 Exercises

1. What does it mean for a functor to be faithful if
a. it is between the categories determined by monoids?
b. it is between the categories determined by posets?

2. Same question as 1 for 'full'.

3. Is the forgetful functor from **Mon** to **Sem** full?

4. Is the free monoid functor faithful? Full?

5. Show that the powerset functor is faithful but not full.

6. Show that **Rel** is isomorphic to its own dual.

7. Show that a groupoid (see Exercise 11) is isomorphic to its own dual.

8. Show that the example in 2.9.3 is an epimorphism in **Cat** when the monoids involved are regarded as categories; hence epimorphisms in **Cat** need not be surjective.

9. Prove that every functor preserves split monos and split epis.

10. a. Does the underlying functor from **Sem** to **Set** preserve or reflect initial objects? What about terminal objects?
b. Same questions for the underlying functor from **Cat** to **Grf** (3.1.10).

11. Show that for any category $\mathscr{C}$ and object A of $\mathscr{C}$, the hom functor $\mathrm{Hom}(A, -)$ (see 3.1.20) preserves terminal objects.

12. Let $\mathscr{C}$ be a category and $f : B \longrightarrow A$ an arrow of $\mathscr{C}$. Show that the slice category $(\mathscr{C}/A)/f$ is isomorphic to the slice category $\mathscr{C}/B$. ('A slice of a slice of $\mathscr{C}$ is a slice of $\mathscr{C}$.') Hint: The functor from $(\mathscr{C}/A)/f$ to $\mathscr{C}/B$ takes an object $w : (g : C \longrightarrow B) \longrightarrow f$ to $g : C \longrightarrow B$, and its inverse takes an object $u : C \longrightarrow B$ to $u : f \circ u \longrightarrow f$.

3.4 Equivalences

In this section we define what it means for two categories to be equivalent. The correct concept turns out to be weaker than requiring that they be isomorphic – that is, that there is a functor from one to the other which has an inverse in **Cat**. In order to understand the issues involved, we first take a close look at the construction of the category corresponding to a monoid in Section 2.3.12. It turns out to be a functor.

3.4.1 Monoids and one-object categories For each monoid M we constructed a small category $C(M)$ in 2.3.12. We make the choice mentioned there that the one object of $C(M)$ is M. Note that although an element of $C(M)$ is now an arrow from M to M, it is *not* a set function.

For each monoid homomorphism $h : M \longrightarrow N$, construct a functor $C(h) : C(M) \longrightarrow C(N)$ as follows:

CF–1 On objects, $C(h)(M) = N$.

CF–2 $C(h)$ must be exactly the same as h on arrows (elements of M).

It is straightforward to see that $C(h)$ is a functor and that this construction makes C a functor from **Mon** to the full subcategory of **Cat** of categories with exactly one object. We will denote this full subcategory as **Ooc**.

There is also a functor $U : \mathbf{Ooc} \longrightarrow \mathbf{Mon}$ going the other way.

UO–1 For a category $\mathscr{C}$ with one object, $U(\mathscr{C})$ is the monoid whose elements are the arrows of $\mathscr{C}$ and whose binary operation is the composition of $\mathscr{C}$.

UO–2 If $F : \mathscr{C} \longrightarrow \mathscr{D}$ is a functor between one-object categories, $U(F) = F_1$, that is, the functor F on arrows.

The functors U and C are not inverse to each other, and it is worthwhile to see in detail why.

The construction of C is in part arbitrary. We needed to regard each monoid as a category with one object. The choice of the elements of M to be the arrows of the category is obvious, but what should be the one object? We chose M itself, but we could have chosen some other thing, such as the set $\{e\}$, where e is the identity of M. The only real requirement is that it not be an element of M (such as its identity) in order to avoid set-theoretic problems caused by the category being an element of itself. The consequence is that we have given a functor $C : \mathbf{Mon} \longrightarrow \mathbf{Ooc}$ in a way which required arbitrary choices.

The arbitrary choice of one object for $C(M)$ means that if we begin with a one-object category $\mathscr{C}$, construct $M = U(\mathscr{C})$, and then construct $C(M)$,

the result will not be the same as C unless it happens that the one object of $\mathscr{C}$ is M. Thus $C \circ U \neq \mathrm{id}_{\mathbf{Ooc}}$, so that U is not the inverse of C. (In this case $U \circ C$ is indeed $\mathrm{id}_{\mathbf{Mon}}$.)

C is not surjective on objects, since not every small category with one object is in the image of C; in fact a category $\mathscr{D}$ is $C(M)$ for some monoid M only if the single object of $\mathscr{D}$ is actually a monoid and the arrows of $\mathscr{D}$ are actually the arrows of that monoid. This is entirely contrary to the spirit of category theory: we are talking about specific elements rather than specifying behavior. Indeed, in terms of specifying behavior, the category of monoids and the category of small categories with one object ought to be essentially the same thing.

The fact that C is not an isomorphism of categories is a signal that isomorphism is the wrong idea for capturing the concept that two categories are essentially the same. However, every small category with one object is *isomorphic* to one of those constructed as $C(M)$ for some monoid M. This is the starting point for the definition of equivalence.

3.4.2 Definition A functor $F : \mathscr{C} \to \mathscr{D}$ is an **equivalence of categories** if there are:

E–1 A functor $G : \mathscr{D} \to \mathscr{C}$.

E–2 A family $u_C : C \to G(F(C))$ of isomorphisms of $\mathscr{C}$ indexed by the objects of $\mathscr{C}$ with the property that for every arrow $f : C \to C'$ of $\mathscr{C}$, $G(F(f)) = u_{C'} \circ f \circ u_C^{-1}$.

E–3 A family $v_D : D \to F(G(D))$ of isomorphisms of $\mathscr{D}$ indexed by the objects of $\mathscr{D}$, with the property that for every arrow $g : D \to D'$ of $\mathscr{D}$, $F(G(g)) = v_{D'} \circ g \circ v_D^{-1}$.

If F is an equivalence of categories, the functor G of E–1 is called a **pseudo-inverse** of F. That the functor C of 3.4.1 is an equivalence (with pseudo-inverse U) is left as an exercise. The families u and v are **natural isomorphisms**, and the arrows u_D and v_D are **components** of the natural isomorphism. These concepts are defined in general in 4.2.18.

The idea behind the definition is that not only is every object of $\mathscr{D}$ isomorphic to an object in the image of F, but the isomorphisms are compatible with the arrows of $\mathscr{D}$; and similarly for $\mathscr{C}$. (See Exercise 6 of Section 4.2.)

3.4.3 Example Let $\mathscr{C}$ be the category with two objects A and B, their identities, and two other arrows $i : A \to B$ and $j : B \to A$ that are inverse isomorphisms between the objects:

$$A \underset{j}{\overset{i}{\rightleftarrows}} B \tag{3.5}$$

Let $\mathscr{D} = \mathbf{1}$ be the category with one object E and its identity arrow. Then $\mathscr{C}$ and $\mathscr{D}$ are equivalent. The unique functor from $\mathscr{C}$ to $\mathscr{D}$ has two pseudo-inverses, each taking the unique object of $\mathscr{D}$ to one of the two isomorphic objects of $\mathscr{C}$.

We give the details for one of these. Let $F : \mathscr{C} \longrightarrow \mathscr{D}$ be the functor that takes A and B to E and $G : \mathscr{D} \longrightarrow \mathscr{C}$ the functor that takes E to A. The family required by E–2 consists of $u_A = \mathrm{id}_A$ and $u_B = j$. That required by E–3 consists of id_E. We have for example

$$G(F(i)) = G(\mathrm{id}_E) = \mathrm{id}_A = j \circ i \circ \mathrm{id}_A = u_B \circ i \circ {u_A}^{-1}$$

The other equations required by E–2 and E–3 are similar or easier.

3.4.4 Theorem *Let $F : \mathscr{C} \longrightarrow \mathscr{D}$ be an equivalence of categories and $G : \mathscr{D} \longrightarrow \mathscr{C}$ a pseudo-inverse to F. Then F and G are full and faithful.*

Proof. Actually, something more is true: if F and G are functors for which E–2 is true, then F is faithful. For suppose $f, f' : C \longrightarrow C'$ in $\mathscr{C}$ and $F(f) = F(f')$ in $\mathscr{D}$. Then $G(F(f)) = G(F(f'))$ in $\mathscr{C}$, so that

$$f = u_{C'}^{-1} \circ G(F(f)) \circ u_C = u_{C'}^{-1} \circ G(F(f')) \circ u_C = f'$$

Thus F is faithful. A symmetric argument shows that if E–3 is true then G is faithful.

Now suppose $F : \mathscr{C} \longrightarrow \mathscr{D}$ is an equivalence of categories and $G : \mathscr{D} \longrightarrow \mathscr{C}$ is a pseudo-inverse to F. We now know that F and G are faithful. To show that F is full, suppose that $g : F(C) \longrightarrow F(C')$ in $\mathscr{D}$. We must find $f : C \longrightarrow C'$ in $\mathscr{C}$ for which $F(f) = g$. Let $f = u_{C'}^{-1} \circ G(g) \circ u_C$. Then a calculation using E–2 shows that $G(F(f)) = G(g)$. Since G is faithful, $F(f) = g$. □

Proposition 3.3.10 implies that an equivalence of categories does not take nonisomorphic objects to isomorphic ones.

An alternative definition of equivalence sometimes given in the literature uses the concept of representative functor. A functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ is **representative** if every object of $\mathscr{D}$ is isomorphic to an object in the image of F. (Thus a subcategory is representative in the sense of Definition 2.7.14 if the inclusion functor is representative.) Then a functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ is an equivalence if it is full, faithful, and representative. This definition can be proved equivalent to ours. The proof requires the axiom of choice.

3.4.5 Inequivalence For any property P of a category that can be defined in terms of composition and identities, if $\mathscr{C}$ and $\mathscr{D}$ are equivalent categories, then either they both have property P or neither of them does. This is an imprecise statement; in particular, a property preserved by equivalence can require that two arrows be the same but it cannot require that

two objects be the same. A formal language that expresses the properties preserved by equivalence is given by Freyd and Scedrov [1990], sections 1.39–1.3(10). See also [Bergman and Berman, 1998].

This observation provides a way to show that two categories are not equivalent. For example, **Set** and **Mon** are not equivalent because there is no arrow in **Set** from the terminal object to the initial object, but in **Mon** the initial and terminal objects are isomorphic. Similarly **Set** and the category of posets and monotone functions are not equivalent because there are only two nonisomorphic sets that have only one automorphism (the empty set and a singleton set), but there are many nonisomorphic posets that have only one automorphism, for example any two totally ordered finite posets of different cardinality.

3.4.6 Example The category **Fin**, of finite sets and functions between them, is equivalent to the opposite of the category of finite Boolean algebras and homomorphisms between them. We sketch the construction here, omitting the many necessary verifications. (Boolean algebras are defined in 5.7.5.) A homomorphism $h : B \to B'$ is a monotone function which preserves meets, joins, $\top$, $\bot$ and complements (these requirements are redundant).

Let **FBool** denote the category of finite Boolean algebras and homomorphisms. Let $F : \mathbf{Fin}^{\mathrm{op}} \to \mathbf{FBool}$ take a finite set to its powerset, which is a Boolean algebra with inclusion for the ordering. If $f : S \to T$ is a function, $F(f) : \mathscr{P}T \to \mathscr{P}S$ takes a subset of T to its inverse image under f; this function $F(f)$ is a homomorphism of Boolean algebras.

To construct the pseudo-inverse, we need a definition. An **atom** in a finite Boolean algebra B is an element a for which there are no elements $b \in B$ such that $\bot < b < a$. It is a fact that any element $b \in B$ is the join of the set of atoms beneath it (this may be *false* in infinite Boolean algebras). It is also true that if $h : B \to B'$ is a homomorphism of Boolean algebras and A is the set of atoms of B, then the join of all the elements $h(a)$ for $a \in A$ is $\top$ in B', and for any two atoms a_1, a_2 of B, $h(a_1) \wedge h(a_2) = \bot$. It follows from this that if b' is an atom of B', then there is a unique atom a of B for which $b' \leq h(a)$.

Now define a functor $G : \mathbf{FBool}^{\mathrm{op}} \to \mathbf{Fin}$ as follows. Let B be a finite Boolean algebra. Then $G(B)$ is the set of atoms of B. If $h : B \to B'$ is a homomorphism, then $G(h) : G(B') \to G(B)$ takes an atom a' of B' to the unique atom a of B for which $a' \leq h(a)$. This makes G a functor.

3.4.7 Proposition *The functor F is an equivalence with pseudo-inverse G.*

The component of the required natural isomorphism from a finite set S to $G(F(S))$ takes an element $x \in S$ to the singleton $\{x\}$. The component of

the natural isomorphism from a finite Boolean algebra B to $F(G(B))$ (the latter is the set of all subsets of all the atoms of B) takes an element $b \in B$ to the set of atoms under b. We omit the details.

3.4.8 Exercises

1. Prove that a functor which is an isomorphism in **Cat** is an equivalence.

2.† Let **Pfn** denote the category of sets and partial functions defined in 2.1.13. Let **Pts** denote the category whose objects are sets with a distinguished element (called **pointed sets**) and whose arrows are functions which preserve the distinguished element. In other words, if S is a set with distinguished element s and T is a set with distinguished element t, then an arrow of **Pts** is a function $f : S \longrightarrow T$ for which $f(s) = t$. Show that **Pfn** and **Pts** are equivalent categories. (Hint: The functor from **Pfn** adds a new element to each set and completes each partial function to a total function by assigning the new element to each element where it was formerly undefined. The functor in the other direction takes a pointed set to the set without the point and if a function has the distinguished point as a value for some input, it becomes undefined at that input.)

3. Show that the category of preordered sets and increasing maps is equivalent to the full category of those small categories with the property that $\mathrm{Hom}(A, B)$ never has more than one element.

4.† (For the reader conversant with vector spaces and linear mappings.) Let $\mathscr{L}$ denote the category of finite dimensional vector spaces and linear maps. Let $\mathscr{M}$ be the category whose objects are the natural numbers, and for which an arrow $M : m \longrightarrow n$ is an $n \times m$ matrix. When $n = 0$ or $m = 0$ or both, there is just one arrow called 0. Composition is matrix multiplication. Any composite involving 0 gives the 0 arrow. Show that $\mathscr{L}$ and $\mathscr{M}$ are equivalent categories.

5. a. Prove that the functor $C : \mathbf{Mon} \longrightarrow \mathbf{Ooc}$ constructed in 3.4.1 is an equivalence of categories.

b. Prove directly, without using Theorem 3.4.4, that it is full and faithful.

3.5 Quotient categories

A quotient of a category by a congruence relation on the arrows is very similar to the concept of the quotient of a monoid by a congruence relation. We will describe the construction from scratch; you need not know about congruence relations to understand it. The construction we describe is not the most general possible: it merges arrows, but not objects.

The constructions of this section are used in 4.1.13 and in the chapters on sketches.

3.5.1 Definition An equivalence relation $\sim$ on the arrows of a category $\mathscr{C}$ is a **congruence relation** if:

CR–1 Whenever $f \sim g$, then f and g have the same domain and the same codomain.

CR–2 In this setting,

$$A \xrightarrow{h} B \underset{g}{\overset{f}{\rightrightarrows}} C \xrightarrow{k} D$$

if $f \sim g$, then $f \circ h \sim g \circ h$ and $k \circ f \sim k \circ g$.

We denote the congruence class containing the arrow f by $[f]$.

3.5.2 Definition Let $\sim$ be a congruence relation on the arrows of $\mathscr{C}$. Define the **quotient category** $\mathscr{C}/\sim$ as follows.

QC–1 The objects of $\mathscr{C}/\sim$ are the objects of $\mathscr{C}$.

QC–2 The arrows of $\mathscr{C}/\sim$ are the congruence classes of arrows of $\mathscr{C}$.

QC–3 If $f : A \longrightarrow B$ in $\mathscr{C}$, then $[f] : A \longrightarrow B$ in $\mathscr{C}/\sim$.

QC–4 If $f : A \longrightarrow B$ and $g : B \longrightarrow C$ in $\mathscr{C}$, then $[g] \circ [f] = [g \circ f] : A \longrightarrow C$ in $\mathscr{C}/\sim$.

It follows from Exercise 1 that QC–4 is well defined and that the result is indeed a category.

3.5.3 Definition Let $\mathscr{C}/\sim$ be the quotient of a category $\mathscr{C}$ by a congruence relation $\sim$. Define $Q : \mathscr{C} \longrightarrow \mathscr{C}/\sim$ by $QA = A$ for an object A and $Qf = [f]$ for an arrow f of $\mathscr{C}$.

It is immediate from QC–4 that Q is a functor.

3.5.4 Proposition *Let $\sim$ be a congruence relation on a category $\mathscr{C}$. Let $F : \mathscr{C} \longrightarrow \mathscr{D}$ be any functor with the property that if $f \sim g$ then $F(f) = F(g)$. Then there is a unique functor $F_0 : \mathscr{C}/\sim \longrightarrow \mathscr{D}$ for which $F_0 \circ Q = F$.*

The proposition says that every way of passing from $\mathscr{C}$ to some other category which merges congruent arrows factors through Q uniquely. This can be perceived in another way: $\mathscr{C}/\sim$ is the category constructed from $\mathscr{C}$ by making the fewest identifications consistent with forcing two congruent arrows in $\mathscr{C}$ to be the same arrow.

3.5.5 Factorization of functors Every functor factors through a faithful one in the following precise sense. Let $F : \mathscr{C} \to \mathscr{D}$ be a functor. The **relation $\sim$ induced by** F on arrows of $\mathscr{C}$ is defined by requiring that $f \sim g$ if and only if f and g have the same domain and codomain and $F(f) = F(g)$.

3.5.6 Proposition *The relation $\sim$ induced by F is a congruence relation on $\mathscr{C}$, and the functor $F_0 : \mathscr{C}/\sim \to \mathscr{D}$ induced by Proposition 3.5.4 is faithful.*

The proof is contained in Exercise 4 below.

A faithful set-valued functor (see 3.3.2) is one for which two different arrows act differently on at least one state. In the special case of monoid actions this is precisely the definition of 'faithful' used in the literature (not a coincidence), and the preceding proposition is a well-known fact about monoid actions.

3.5.7 The intersection of any set of congruence relations on a category is also a congruence relation (Exercise 2). This means that if α is any relation on $\mathscr{C}$ with the property that if $f \alpha g$ then f and g have the same domain and the same codomain, then there is a unique smallest congruence relation generated by α.

Thus in particular given two arrows $f : A \to B$ and $g : A \to B$ in a category $\mathscr{C}$, there is a quotient category by the congruence relation generated by requiring that $f = g$. This is called **imposing the relation** $f = g$. Two arrows in $\mathscr{C}$ are merged in the quotient category if requiring that $f = g$ forces them to be merged.

3.5.8 The category of a programming language We described in 2.2.6 the category $C(L)$ corresponding to a simple functional programming language L defined there. We can now say precisely what $C(L)$ is.

The definition of L in 2.2.5 gives the primitive types and operations of the language. The types are the nodes and the operations are the arrows of a graph. This graph generates a free category $F(L)$, and the equations imposed in 2.2.5(ii) and (iv) (each of which says that two arrows of $C(L)$ must be equal) generate a congruence relation as just described. The resulting quotient category is precisely $C(L)$.

When one adds constructors such as record types to the language, the quotient construction is no longer enough. Then it must be done using sketches. The construction just given is in fact a special case of a model of a sketch (see Section 4.6).

3.5.9 Functorial semantics Functors provide a way to give a meaning to the constructs of the language L just mentioned. This is done by giving a functor from $C(L)$ to some category suitable for programming language semantics, such as those discussed in 2.4.3.

We illustrate this idea here using a functor to **Set** as the semantics for the language described in 2.2.5. **Set** is for many reasons unsuitable for programming language semantics, but it is the natural category for expressing our intuitive understanding of what programming language constructs mean.

Following the discussion in 2.2.5, we define a semantics functor $\Sigma : C(L) \to \mathbf{Set}$. To do this, first we define a function F on the primitive types and operations of the language.

(i) $F(\texttt{NAT})$ is the set of natural numbers. The constant 0 is the number 0 and $F(\texttt{succ})$ is the function which adds 1.

(ii) $F(\texttt{BOOLEAN})$ is the set $\{\texttt{true}, \texttt{false}\}$. The constants `true` and `false` are the elements of the same name, and $F(\neg)$ is the function which switches true and false.

(iii) $F(\texttt{CHAR})$ is the set of 128 ASCII symbols, and each symbol is a constant.

(iv) $F(\texttt{ord})$ takes a character to its ASCII value, and $F(\texttt{chr})$ takes a number n to the character with ASCII code n modulo 128.

Let $F(L)$ be the free category generated by the graph of types and operations, as in 2.6.16. By Proposition 3.1.15, there is a functor $\widehat{F} : F(L) \to \mathbf{Set}$ which has the effect of F on the primitive types and operations.

This functor $\widehat{F}$ has the property required by Proposition 3.5.4 that if $\sim$ is the congruence relation on $F(L)$ generated by the equations of 2.2.5(ii) and (iv), then $f \sim g$ implies that $\widehat{F}(f) = \widehat{F}(g)$ (Exercise 5). This means that there is a functor $\Sigma : C(L) \to \mathbf{Set}$ (called F_0 in Proposition 3.5.4) with the property that if x is any primitive type or operation, then $\Sigma(x) = F(x)$.

The fact that Σ is a functor means that it preserves the meaning of programs; for example the program (path of arrows) $\texttt{chr} \circ \texttt{succ} \circ \texttt{ord}$ ought to produce the next character in order, and in fact

$$\Sigma(\texttt{chr} \circ \texttt{succ} \circ \texttt{ord})$$

does just that, as you can check. Thus it is reasonable to refer to Σ as a possible semantics of the language L.

We will return to this example in Section 4.3.12. The construction of $C(L)$ and Σ are instances of the construction of the theory of a sketch in Section 7.5.

3.5.10 Exercises

1. Show that an equivalence relation $\sim$ satisfying CR–1 is a congruence relation if and only if, for all arrows f_1, f_2, g_1, g_2 as in this diagram,

$$A \underset{f_2}{\overset{f_1}{\rightrightarrows}} B \underset{g_2}{\overset{g_1}{\rightrightarrows}} C$$

if $f_1 \sim f_2$ and $g_1 \sim g_2$, then $g_1 \circ f_1 \sim g_2 \circ f_2$.

2. Show that the intersection of congruence relations is a congruence relation.

3. Show that the quotient functor in 3.5.3 is full. (Warning: This exercise would be incorrect if we allowed the more general definition of quotient, which allows merging objects as well as arrows.)

4. Let $F : \mathscr{C} \to \mathscr{D}$ be a functor.

a. Show that the relation $\sim$ induced by F (defined in 3.5.5) is a congruence relation.

b. Show that the induced functor $F_0 : \mathscr{C}/\sim \to \mathscr{D}$ is faithful.

c. Conclude from this and the preceding exercise that every functor $F : \mathscr{C} \to \mathscr{D}$ factors as a full functor followed by a faithful functor.

5. Let $\widehat{F}$ and $\sim$ be defined as in 3.5.9. Prove that $f \sim g$ implies that $\widehat{F}(f) = \widehat{F}(g)$.

6. Let M be a monoid. A **congruence** on M is an equivalence relation $\sim$ with the property that it is a congruence relation for the category $C(M)$ determined by M.

a. Show that an equivalence relation $\sim$ on M is a congruence relation if and only if for all elements m, n, n' of M, if $n \sim n'$ then $mn \sim mn'$ and $nm \sim n'm$.

b. Let K be the subset $\{(m, n) \mid m \sim n\}$ of the monoid $M \times M$. Show that K is a submonoid of $M \times M$ if and only if $\sim$ is a congruence relation. ($M \times M$ is the monoid whose elements are all ordered pairs of elements of M with multiplication $(m, n)(m', n') = (mm', nn')$.)

4

Diagrams, naturality and sketches

Commutative diagrams are the categorist's way of expressing equations. Natural transformations are maps between functors; one way to think of them is as a deformation of one construction (construed as a functor) into another. A sketch is a graph with imposed commutativity and other conditions; it is a way of expressing structure. Models of the structure are given by functors, and homomorphisms between them by natural transformations.

All this will become clearer as the chapter is read. It turns out that the concepts just mentioned are all very closely related to each other. Indeed, there is a sense in which diagrams, functors and models of sketches are all different aspects of the same idea: they are all types of graph homomorphisms in which some or all of the graphs are categories.

The first three sections introduce diagrams, commutative diagrams and natural transformations, three basic ideas in category theory. These concepts are used heavily in the rest of the book. Section 4.4 gives the Godement rules. These form the basis of the algebra of functors and natural transformations, which is studied in more abstract form in Section 4.8.

Section 4.5 introduces the concepts of representable functor, the Yoneda embedding and universal elements. Working through the details of this presentation is an excellent way of learning to work with natural transformations.

We also recommend studying the introduction to linear sketches and linear sketches with constants in Sections 4.6 and 4.7 as an excellent way to familiarize yourself with both commutative diagrams and natural transformations. However, these two sections may be skipped unless you are going to read Chapters 7, 8, 10 or 11.

Section 4.8 introduces 2-categories, a notion of category that allows mappings between arrows (called 2-cells) that has been found useful to model program refinement, among other things. This section is not used in the rest of the book.

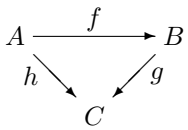
4.1 Diagrams

We begin with diagrams in a graph and discuss commutativity later.

4.1.1 Definition Let $\mathscr{I}$ and $\mathscr{G}$ be graphs. A **diagram** in $\mathscr{G}$ of **shape** $\mathscr{I}$ is a homomorphism $D : \mathscr{I} \to \mathscr{G}$ of graphs. $\mathscr{I}$ is called the **shape graph** of the diagram D.

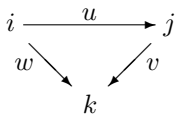
We have thus given a new name to a concept which was already defined (not uncommon in mathematics). A diagram is a graph homomorphism from a different point of view.

4.1.2 Example At first glance, Definition 4.1.1 may seem to have little to do with what are informally called diagrams, for example



(4.1)

The connection is this: a diagram in the sense of Definition 4.1.1 is pictured on the page with a drawing of nodes and arrows as for example in Diagram (4.1), which could be the picture of a diagram D with shape graph



(4.2)

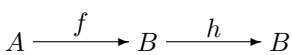
defined by $D(i) = A$, $D(j) = B$, $D(k) = C$, $D(u) = f$, $D(v) = g$ and $D(w) = h$.

4.1.3 Example Here is an example illustrating some subtleties involving the concept of diagram. Let $\mathscr{G}$ be a graph with objects A, B and C (and maybe others) and arrows $f : A \to B$, $g : B \to C$ and $h : B \to B$. Consider these two diagrams, where here we use the word 'diagram' informally:

(a) (b)

(4.3)

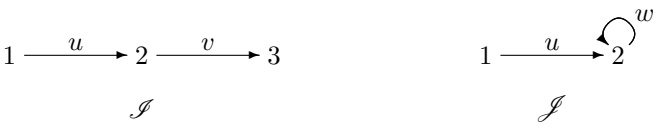
These are clearly of different shapes (again using the word 'shape' informally). But the diagram



(4.4)

is the same shape as (4.3)(a) even though as a graph it is the same as (4.3)(b).

To capture the difference thus illustrated between a graph and a diagram, we introduce two shape graphs



(4.5)

(where, as will be customary, we use numbers for the nodes of shape graphs). Now diagram (4.3)(a) is seen to be the diagram $D : \mathscr{I} \to \mathscr{G}$ with $D(1) = A$, $D(2) = B$, $D(3) = C$, $D(u) = f$ and $D(v) = g$; whereas diagram (4.3)(b) is $E : \mathscr{J} \to \mathscr{G}$ with $E(1) = A$, $E(2) = B$, $E(u) = f$ and $E(w) = h$. Moreover, Diagram (4.4) is just like D (has the same shape), except that v goes to h and 3 goes to B.

4.1.4 Our definition in 4.1.1 of a diagram as a graph homomorphism, with the domain graph being the shape, captures both the following ideas:

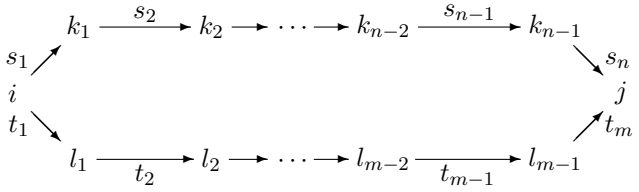
(i) A diagram can have repeated labels on its nodes and (although the examples did not show it) on its arrows, and

(ii) Two diagrams can have the same labels on their nodes and arrows but be of different shapes: Diagrams (4.3)(b) and (4.4) are *different diagrams* because they have different shapes.

4.1.5 Commutative diagrams When the target graph of a diagram is the underlying graph of a category some new possibilities arise, in particular the concept of commutative diagram, which is the categorist's way of expressing equations.

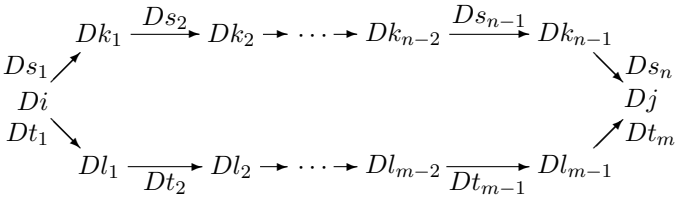
In this situation, we will not distinguish in notation between the category and its underlying graph: if $\mathscr{I}$ is a graph and $\mathscr{C}$ is a category we will refer to a diagram $D : \mathscr{I} \to \mathscr{C}$.

We say that D is **commutative** (or **commutes**) provided for any nodes i and j of $\mathscr{I}$ and any two paths



(4.6)

from i to j in $\mathscr{I}$, the two paths



(4.7)

compose to the same arrow in $\mathscr{C}$. This means that

$$Ds_n \circ Ds_{n-1} \circ \ldots \circ Ds_1 = Dt_m \circ Dt_{m-1} \circ \ldots \circ Dt_1$$

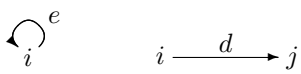
4.1.6 Much ado about nothing There is one subtlety to the definition of commutative diagram: what happens if one of the numbers m or n in Diagram (4.7) should happen to be 0? If, say, $m = 0$, then we interpret the above equation to be meaningful only if the nodes i and j are the same (you go nowhere on an empty path) and the meaning in this case is that

$$Ds_n \circ Ds_{n-1} \circ \ldots \circ Ds_1 = \mathrm{id}_{D_i}$$

(you do nothing on an empty path). In particular, a diagram D based on the graph

commutes if and only if $D(e)$ is the identity arrow from $D(i)$ to $D(i)$.

Note, and note well, that both shape graphs



(a) (b)

have models that one might think to represent by the diagram

but the diagram based on (a) commutes if and only if $f = \mathrm{id}_A$, while the diagram based on (b) commutes automatically (no two nodes have more than one path between them so the commutativity condition is vacuous).

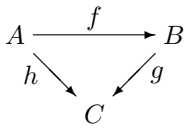
We will always picture diagrams so that distinct nodes of the shape graph are represented by distinct (but possibly identically labeled) nodes in the

picture. Thus a diagram based on (b) in which d goes to f and i and j both go to A will be pictured as

$$A \xrightarrow{f} A$$

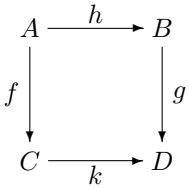
In consequence, one can always deduce the shape graph of a diagram from the way it is pictured, except of course for the actual names of the nodes and arrows of the shape graph.

4.1.7 Examples of commutative diagrams – and others The prototypical commutative diagram is the triangle



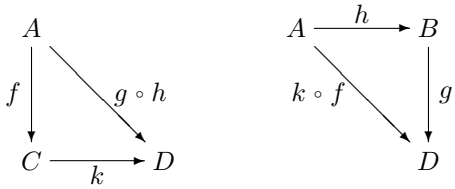
(4.8)

that commutes if and only if h is the composite $g \circ f$. The reason this is prototypical is that any commutative diagram – unless it involves an empty path – can be replaced by a set of commutative triangles. This fact is easy to show and not particularly enlightening, so we are content to give an example. The diagram



(4.9)

commutes if and only if the two diagrams



(4.10)

commute (in fact if and only if either one does).

4.1.8 Example An arrow $f : A \longrightarrow B$ is an isomorphism with inverse $g : B \longrightarrow A$ if and only if

$$A \underset{g}{\overset{f}{\rightleftarrows}} B \tag{4.11}$$

commutes. The reason for this is that for this diagram to commute, the two paths () and (g, f) from A to A must compose to the same value in the diagram, which means that $g \circ f = \mathrm{id}_A$. A similar observation shows that $f \circ g$ must be id_B.

4.1.9 Graph homomorphisms by commutative diagrams The definition of graph homomorphism in 1.4.1 can be expressed by a commutative diagram. Let $\phi = (\phi_0, \phi_1)$ be a graph homomorphism from $\mathscr{G}$ to $\mathscr{H}$. For any arrow $u : m \to n$ in $\mathscr{G}$, 1.4.1 requires that $\phi_1(u) : \phi_0(m) \to \phi_0(n)$ in $\mathscr{H}$. This says that $\phi_0(\mathrm{source}(u)) = \mathrm{source}(\phi_1(u))$, and a similar statement about targets. In other words, these diagrams must commute:

$$\begin{array}{ccc} G_1 & \xrightarrow{\phi_1} & H_1 \\ {\scriptstyle \mathrm{source}}\downarrow & & \downarrow{\scriptstyle \mathrm{source}} \\ G_0 & \xrightarrow[\phi_0]{} & H_0 \end{array} \qquad \begin{array}{ccc} G_1 & \xrightarrow{\phi_1} & H_1 \\ {\scriptstyle \mathrm{target}}\downarrow & & \downarrow{\scriptstyle \mathrm{target}} \\ G_0 & \xrightarrow[\phi_0]{} & H_0 \end{array} \tag{4.12}$$

In these two diagrams the two arrows labeled 'source' are of course different functions; one is the source function for $\mathscr{G}$ and the other for $\mathscr{H}$. A similar remark is true of 'target'.

4.1.10 This point of view provides a pictorial proof that the composite of two graph homomorphisms is a graph homomorphism (see 2.4.1). If $\phi : \mathscr{G} \to \mathscr{H}$ and $\psi : \mathscr{H} \to \mathscr{K}$ are graph homomorphisms, then to see that $\psi \circ \phi$ is a graph homomorphism requires checking that the outside rectangle below commutes, and similarly with target in place of source:

$$\begin{array}{ccccc} G_1 & \xrightarrow{\phi_1} & H_1 & \xrightarrow{\psi_1} & K_1 \\ {\scriptstyle \mathrm{source}}\downarrow & & \downarrow{\scriptstyle \mathrm{source}} & & \downarrow{\scriptstyle \mathrm{source}} \\ G_0 & \xrightarrow[\phi_0]{} & H_0 & \xrightarrow[\psi_0]{} & K_0 \end{array} \tag{4.13}$$

The outside rectangle commutes because the two squares commute. This can be checked by tracing (mentally or with a finger or pointer) the paths from G_1 to K_0 to verify that

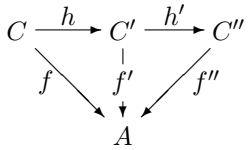
$$\mathrm{source} \circ \psi_1 \circ \phi_1 = \psi_0 \circ \mathrm{source} \circ \phi_1 \tag{4.14}$$

because the right square commutes, and that

$$\psi_0 \circ \mathrm{source} \circ \phi_1 = \psi_0 \circ \phi_0 \circ \mathrm{source} \tag{4.15}$$

because the left square commutes. The verification process just described is called 'chasing the diagram'. Of course, one can verify the required fact by writing the equations (4.14) and (4.15) down, but those equations hide the source and target information given in Diagram (4.13) and thus provide a possibility of writing an impossible composite down. For many people, Diagram (4.13) is much easier to remember than equations (4.14) and (4.15). However, diagrams are more than informal aids; they are formally-defined mathematical objects just like automata and categories.

The proof in 2.6.10 that the composition of arrows in a slice gives another arrow in the category can be represented by a similar diagram:



These examples are instances of **pasting** commutative diagrams together to get bigger ones. (See 4.8.16.)

4.1.11 Associativity by commutative diagrams The fact that the multiplication in a monoid or semigroup is associative can be expressed as the assertion that a certain diagram in **Set** commutes.

Let S be a semigroup. Define the following functions, using the cartesian product notation for functions of 1.2.9:

(i) $\text{mult} : S \times S \longrightarrow S$ satisfies $\text{mult}(x, y) = xy$.

(ii) $S \times \text{mult} : S \times S \times S \longrightarrow S \times S$ satisfies

$$(S \times \text{mult})(x, y, z) = (x, yz)$$

(iii) $\text{mult} \times S : S \times S \times S \longrightarrow S \times S$ satisfies

$$(\text{mult} \times S)(x, y, z) = (xy, z)$$

That the following diagram commutes is exactly the associative law.

$$\begin{array}{ccc} S \times S \times S & \xrightarrow{S \times \text{mult}} & S \times S \\ {\scriptstyle \text{mult} \times S}\downarrow & & \downarrow{\scriptstyle \text{mult}} \\ S \times S & \xrightarrow[\text{mult}]{} & S \end{array} \tag{4.16}$$

4.1.12 Normally, associativity is expressed by the equation $x(yz) = (xy)z$ for all x, y, z in the semigroup. The commutative diagram expresses this same fact *without the use of variables*. Of course, we did use variables in defining the functions involved, but we remedy that deficiency in Chapter 5 when we give a categorical definition of products.

Another advantage of using diagrams to express equations is that diagrams show the source and target of the functions involved. This is not particularly compelling here but in other situations the two-dimensional picture of the compositions involved makes it much easier to follow the discussion.

4.1.13 In 3.5.7, we described how to force two arrows in a category $\mathscr{C}$ to be the same by going to a quotient category. More generally, you can make any set $\mathscr{D}$ of diagrams in $\mathscr{C}$ commute, by imposing all the relations of the form

$$Ds_n \circ Ds_{n-1} \circ \ldots \circ Ds_1 \sim Dt_m \circ Dt_{m-1} \circ \ldots \circ Dt_1$$

where

$$\begin{array}{ccccccccc} & & Dk_1 & \xrightarrow{Ds_2} & Dk_2 & \rightarrow \cdots \rightarrow & Dk_{n-2} & \xrightarrow{Ds_{n-1}} & Dk_{n-1} \\ & \nearrow^{Ds_1} & & & & & & & \searrow^{Ds_n} \\ Di & & & & & & & & Dj \\ & \searrow_{Dt_1} & & & & & & & \nearrow_{Dt_m} \\ & & Dl_1 & \xrightarrow[Dt_2]{} & Dl_2 & \rightarrow \cdots \rightarrow & Dl_{m-2} & \xrightarrow[Dt_{m-1}]{} & Dl_{m-1} \end{array} \tag{4.17}$$

are two paths in any diagram $D \in \mathscr{D}$. As before, if one of these paths is the empty path the other must be an identity arrow in order for the diagram to commute.

4.1.14 Diagrams as functors In much of the categorical literature, a diagram in a category $\mathscr{C}$ is a functor $D : \mathscr{E} \rightarrow \mathscr{C}$ where $\mathscr{E}$ is a category. Because of Proposition 3.1.15, a graph homomorphism into a category extends uniquely to a functor based on the free category generated by the graph, so that diagrams in our sense generate diagrams in the functorial sense. On the other hand, any functor is a graph homomorphism on the underlying graph of its domain (although not conversely!), so that every diagram in the sense of functor is a diagram in the sense of graph homomorphism.

4.1.15 Exercises

1. Draw a commutative diagram expressing the fact that an arrow $f : A \longrightarrow B$ factors through an arrow $g : C \longrightarrow B$. (See 2.8.9.)

2. Draw a commutative diagram to express the fact that addition of real numbers is commutative.

3. Draw commutative diagrams expressing the equations occurring in the definition of the sample functional programming language in 2.2.5.

4. Express the definition of functor using commutative diagrams.

4.2 Natural transformations

4.2.1 Unary operations In Section 4.1 we saw that diagrams in a category are graph homomorphisms to the category from a different point of view. Now we introduce a third way to look at graph homomorphisms to a category, namely as models. To give an example, we need a definition.

4.2.2 Definition A **unary operation** on a set S is a function $u : S \longrightarrow S$.

This definition is by analogy with the concept of binary operation on a set. A set with a unary operation is a (very simple) algebraic structure, which we call a **u-structure**. If the set is S and the operation is $f : S \longrightarrow S$, we say that (S, f) is a u-structure, meaning (S, f) denotes a u-structure whose underlying set is S, and whose unary operation is f. This uses positional notation in much the same way as procedures in many computer languages: the first entry in the expression '(S, f)' is the name of the underlying set of the u-structure and the second entry is the name of its operation.

4.2.3 A homomorphism of u-structures should be a function which preserves the structure. There is really only one definition that is reasonable for this idea: if (S, u) and (T, v) are u-structures, $f : S \longrightarrow T$ is a **homomorphism of u-structures** if $f(u(s)) = v(f(s))$ for all $s \in S$. Thus this diagram must commute:

$$\begin{array}{ccc} S & \xrightarrow{f} & T \\ {\scriptstyle u}\downarrow & & \downarrow{\scriptstyle v} \\ S & \xrightarrow[f]{} & T \end{array} \tag{4.18}$$

It is not difficult to show that the composite of two homomorphisms of u-structures is another one, and that the identity map is a homomorphism, so that u-structures and homomorphisms form a category.

4.2.4 Models of graphs We now use the concept of u-structure to motivate the third way of looking at graph homomorphisms to a category.

4.2.5 Let $\mathscr{U}$ be the graph with one node u_0 and one arrow e:

Let us define a graph homomorphism $D : \mathscr{U} \longrightarrow \mathbf{Set}$ as follows: $D(u_0) = \mathbf{R}$ and $D(e) = x \mapsto x^2$. Now $(\mathbf{R}, x \mapsto x^2)$ is a u-structure, and the notation we have introduced in 4.2.1 tells us that we have chosen $\mathbf{R}$ to be its underlying set and $x \mapsto x^2$ to be its unary operation. Except for the arbitrary names 'u_0' and 'e', the graph homomorphism D communicates the same information: '$\mathbf{R}$ is the particular set chosen to be the value of u_0, and $x \mapsto x^2$ is the particular function chosen to be the value of e.'

In this sense, a u-structure is essentially the same thing as a diagram in **Set** of shape $\mathscr{U}$: a u-structure 'models' the graph $\mathscr{U}$. This suggests the following definition.

4.2.6 Definition A **model** M of a graph $\mathscr{G}$ is a graph homomorphism $M : \mathscr{G} \longrightarrow \mathbf{Set}$.

We will see how to define a monoid as a model involving a graph homomorphism (and other ingredients) in Chapter 7. We had to introduce u-structures here to have an example for which we had the requisite techniques. The technique we are missing is the concept of product in a category, which allows the definition of operations of arity (see Definition 7.7.1) greater than one.

> Both category theory and mathematical logic have concepts of 'model'. Both are formalisms attempting to make precise the relationship between the (formal) description of a mathematical structure and the structure itself. In logic, the precise description (the syntax) is given by a logical theory; in category theory by sketches or by categories regarded as theories. Good introductions to various aspects of categorical logic and model theory are given by [Makkai and Reyes, 1977], [Lambek and Scott, 1986], [Makkai and Paré, 1990] and [Adámek and Rosický, 1994].

4.2.7 Example As another example, consider this graph (see 1.3.9):

$$\mathsf{a} \underset{\mathsf{target}}{\overset{\mathsf{source}}{\rightrightarrows}} \mathsf{n} \tag{4.19}$$

A model M of this graph consists of sets $G_0 = M(\mathsf{n})$ and $G_1 = M(\mathsf{a})$ together with functions source $= M(\mathsf{source}) : G_1 \longrightarrow G_0$ and target $= M(\mathsf{target}) : G_1$

$\longrightarrow G_0$. To understand what this structure is, imagine a picture in which there is a dot corresponding to each element of G_0 and an arrow corresponding to each element $a \in G_1$ which goes from the dot corresponding to source(a) to the one corresponding to target(a). It should be clear that the picture so described is a graph and thus the graph (4.19) is a graph whose models are graphs!

This definition makes the semantics of 1.3.9 into a mathematical construction.

4.2.8 Models in arbitrary categories The concept of model can be generalized to arbitrary categories: if $\mathscr{C}$ is any category, a **model of $\mathscr{G}$ in $\mathscr{C}$** is a graph homomorphism from $\mathscr{G}$ to $\mathscr{C}$. For example, a model of the graph for u-structures in the category of posets and monotone maps is a poset and a monotone map from the poset to itself.

In this book, the bare word 'model' always means a model in **Set**.

4.2.9 Natural transformations between models of a graph In a category, there is a natural notion of an arrow from one model of a graph to another. This usually turns out to coincide with the standard definition of homomorphism for that kind of structure.

4.2.10 Definition Let $D, E : \mathscr{G} \longrightarrow \mathscr{C}$ be two models of the same graph in a category. A **natural transformation** $\alpha : D \longrightarrow E$ is given by a family of arrows αa of $\mathscr{C}$ indexed by the nodes of $\mathscr{G}$ such that:

NT–1 $\alpha a : Da \longrightarrow Ea$ for each node a of $\mathscr{G}$.

NT–2 For any arrow $s : a \longrightarrow b$ in $\mathscr{G}$, the diagram

$$\begin{array}{ccc} Da & \xrightarrow{\alpha a} & Ea \\ {\scriptstyle Ds}\downarrow & & \downarrow{\scriptstyle Es} \\ Db & \xrightarrow[\alpha b]{} & Eb \end{array} \tag{4.20}$$

commutes.

The commutativity of the diagram in NT–2 is referred to as the **naturality condition** on α. The arrow αa for an object a is the **component** of the natural transformation α at a.

Note that you talk about a natural transformation from D to E only if D and E have the same domain (here $\mathscr{G}$) as well as the same codomain (here $\mathscr{C}$) and if, moreover, the codomain is a category. In this situation, it is often convenient to write $\alpha : D \longrightarrow E : \mathscr{G} \longrightarrow \mathscr{C}$.

4.2.11 Definition Let D, E and F be models of $\mathscr{G}$ in $\mathscr{C}$, and $\alpha : D \longrightarrow E$ and $\beta : E \longrightarrow F$ natural transformations. The **composite** $\beta \circ \alpha : D \longrightarrow F$ is defined componentwise: $(\beta \circ \alpha)a = \beta a \circ \alpha a$.

4.2.12 Proposition *The composite of two natural transformations is also a natural transformation.*

Proof. The diagram that has to be shown commutative is the outer rectangle of

$$\begin{array}{ccccc} Da & \xrightarrow{\alpha a} & Ea & \xrightarrow{\beta a} & Fa \\ {\scriptstyle Ds}\downarrow & & \downarrow{\scriptstyle Es} & & \downarrow{\scriptstyle Fs} \\ Db & \xrightarrow[\alpha b]{} & Eb & \xrightarrow[\beta b]{} & Fb \end{array} \tag{4.21}$$

for each arrow $s : a \longrightarrow b$ in $\mathscr{G}$. The rectangle commutes because the two squares do; the squares commute as a consequence of the naturality of α and β. □

> It is interesting that categorists began using modes of reasoning like that in the preceding proof because objects of categories generally lacked elements; now one appreciates them for their own sake *because* they allow element-free (and thus variable-free) arguments.

4.2.13 It is even easier to show that there is an identity natural transformation between any model D and itself, defined by $(\mathrm{id}_D)a = \mathrm{id}_{Da}$. We then have the following proposition, whose proof is straightforward.

4.2.14 Proposition *The models of a given graph $\mathscr{G}$ in a given category $\mathscr{C}$, and the natural transformations between them, form a category. We denote this category by* $\mathbf{Mod}(\mathscr{G}, \mathscr{C})$.

4.2.15 Example The natural transformations between models in **Set** of the u-structure graph $\mathscr{U}$ defined in 4.2.5 are exactly the homomorphisms of u-structures defined in 4.2.3. The graph described in 4.2.5 has one object u_0 and one arrow e, so that a natural transformation from a model D to a model E has only one component which is a function from $D(u_0)$ to $E(u_0)$. If we set $S = D(u_0)$, $u = D(e)$, $T = E(u_0)$, $v = E(e)$, and we define $\alpha u_0 = f$, this is the single component of a natural transformation from D to E. Condition NT–2 in 4.2.10 coincides in this case with the diagram in 4.2.3: the naturality condition is the same as the definition of homomorphism of u-structures. It follows that the category of u-structures and homomorphisms is essentially $\mathbf{Mod}(\mathscr{U}, \mathbf{Set})$.

4.2.16 Example A homomorphism of graphs is a natural transformation between models of the graph

$$\mathsf{a} \underset{\text{target}}{\overset{\text{source}}{\rightrightarrows}} \mathsf{n}$$

The two graphs in Diagram (4.12) are the two necessary instances (one for the source and the other for the target) of Diagram (4.20). In a similar way, Diagram (4.21), used to show that the composite of two natural transformations is a natural transformation, reduces in this case to the commutativity of Diagram (4.13): specifically, the only possibilities (other than those in which s is an identity arrow) for a and b in Diagram (4.21) are $a = \mathsf{a}$ and $b = \mathsf{n}$, giving two diagrams shaped like Diagram (4.21), one for $s = \text{source}$ (that is Diagram (4.13)) and the other for $s = \text{target}$.

4.2.17 Example A model of the graph

$$0 \xrightarrow{\ u\ } 1 \tag{4.22}$$

in an arbitrary category $\mathscr{C}$ is essentially the same as an arrow in $\mathscr{C}$ (see 4.2.22 below). A natural transformation from the model represented by the arrow $f : A \to B$ to the one represented by $g : C \to D$ is a pair of arrows $h : A \to C$ and $k : B \to D$ making a commutative diagram:

$$\begin{array}{ccc} A & \xrightarrow{\ h\ } & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[\ k\ ]{} & D \end{array} \tag{4.23}$$

The component at 0 is h and the component at 1 is k. The category of models in $\mathscr{C}$ is called the **arrow category** of $\mathscr{C}$; it is often denoted $\mathscr{C}^{\to}$.

4.2.18 Natural isomorphisms A natural transformation $\alpha : F \to G : \mathscr{G} \to \mathscr{D}$ is called a **natural isomorphism** if there is a natural transformation $\beta : G \to F$ which is an inverse to α in the category $\mathbf{Mod}(\mathscr{G}, \mathscr{D})$. Natural isomorphisms are often called **natural equivalences**.

4.2.19 Example The arrow $(h, k) : f \to g$ in the arrow category of a category $\mathscr{C}$, as shown in (4.23), is a natural isomorphism if and only if h and k are both isomorphisms in $\mathscr{C}$. This is a special case of an important fact about natural isomorphisms, which we now state.

4.2.20 Theorem *Suppose $F : \mathscr{G} \to \mathscr{D}$ and $G : \mathscr{G} \to \mathscr{D}$ are models of $\mathscr{G}$ in $\mathscr{D}$ and $\alpha : F \to G$ is a natural transformation of models. Then α is a natural isomorphism if and only if for each node a of $\mathscr{G}$, $\alpha a : F(a) \to G(a)$ is an isomorphism of $\mathscr{D}$.*

Proof. Suppose α has an inverse $\beta : G \to F$ in $\mathbf{Mod}(\mathscr{G}, \mathscr{D})$. Then for any node a, by Definition 4.2.11, Definition 4.2.13, and the definition of inverse,

$$\alpha a \circ \beta a = (\alpha \circ \beta)a = \mathrm{id}_G\, a = \mathrm{id}_{G(a)}$$

and

$$\beta a \circ \alpha a = (\beta \circ \alpha)a = \mathrm{id}_F\, a = \mathrm{id}_{F(a)}$$

which means that the arrow βa is the inverse of the arrow αa, so that αa is an isomorphism in $\mathscr{D}$ as required.

Conversely, suppose that for each node a of $\mathscr{G}$, $\alpha a : F(a) \to G(a)$ is an isomorphism of $\mathscr{D}$. The component of the inverse β at a node a is defined by letting $\beta a = (\alpha a)^{-1}$. This is the only possible definition, but it must be shown to be natural. Let $f : a \to b$ be an arrow of the domain of F and G. Then we have

$$\begin{aligned} Ff \circ (\alpha a)^{-1} &= (\alpha b)^{-1} \circ (\alpha b) \circ Ff \circ (\alpha a)^{-1} \\ &= (\alpha b)^{-1} \circ Gf \circ (\alpha a) \circ (\alpha a)^{-1} \\ &= (\alpha b)^{-1} \circ Gf \end{aligned}$$

which says that β is natural. The second equality uses the naturality of α. □

4.2.21 Monic natural transformations Let $\alpha : F \to G$ be a natural transformation between models of $\mathscr{G}$ in $\mathscr{D}$. Suppose each component of α is a monomorphism in $\mathscr{D}$. Then it is easy to prove that α is a monomorphism in $\mathbf{Mod}(\mathscr{G}, \mathscr{D})$.

However, in contrast to Theorem 4.2.20, the converse need not be true. As an example (thanks to Andrew Richardson), let $\mathscr{D}$ be the subcategory of **Set** consisting of

(a) Objects: $\{1\}$, $\{1,2\}$, $\{1,2,3\}$ and $\{1,2,4\}$.

(b) Arrows:

  (i) The identity functions.

  (ii) $g : \{1,2\} \to \{1\}$, $h : \{1,2,3\} \to \{1\}$ and $k : \{1,2,4\} \to \{1\}$ the only possible functions.

  (iii) $f : \{1,2,3\} \to \{1,2\}$ and $u : \{1,2,4\} \to \{1,2\}$ the constant functions with value 1.

(iv) $v : \{1,2,4\} \to \{1,2\}$ the constant function with value 2.

(c) Composition of set functions as composition.

The category can be pictured like this:

$$\begin{array}{ccccc} \{1,2,3\} & & \xrightarrow{h} & & \{1\} \\ & \Big\downarrow{\scriptstyle f} & & & \Big\downarrow{\scriptstyle \mathrm{id}_{\{1\}}} \\ \{1,2,4\} & \underset{v}{\overset{u}{\rightrightarrows}} & \{1,2\} & \xrightarrow[g]{} & \{1\} \end{array} \tag{4.24}$$

The square commutes, so $(h,g) : f \to \mathrm{id}_{\{1\}}$ is an arrow in $\mathscr{D}^{\to}$. Moreover, (h,g) is monic in $\mathscr{D}^{\to}$ (because the only natural transformation with codomain f is $(\mathrm{id}_{\{1,2,3\}}, \mathrm{id}_{\{1,2\}})$) but its component g is not monic in $\mathscr{D}$ because $g \circ u = g \circ v$.

4.2.22 'Essentially the same' In 4.2.17, we said that a model in an arbitrary category $\mathscr{C}$ of the graph (4.22) is 'essentially the same' as an arrow in $\mathscr{C}$. This is common terminology and usually refers implicitly to an equivalence of categories. We spell it out in this case.

Let us say that for a category $\mathscr{C}$, $\mathscr{C}'$ is the category whose objects are the arrows of $\mathscr{C}$ and for which an arrow from f to g is a pair (h,k) making Diagram (4.23) commute.

A model M of the graph (4.22) in a category $\mathscr{C}$ specifies the objects $M(0)$ and $M(1)$ and the arrow $M(u)$. $M(u)$ has domain $M(0)$ and codomain $M(1)$. But the domain and codomain of an arrow in a category are uniquely determined by the arrow. So that the only necessary information is which arrow $M(u)$ is.

Now we can define a functor $F : \mathscr{C}^{\to} \to \mathscr{C}'$. On objects it take M to $M(u)$. The remarks in the preceding paragraph show that this map on objects is bijective. If $M(u) = f$ and $N(u) = g$, an arrow from M to N in $\mathscr{C}^{\to}$ and an arrow from f to g in $\mathscr{C}'$ are the same thing – a pair (h,k) making Diagram (4.23) commute. So we say F is the identity on arrows. It is straightforward to see that F is actually an isomorphism of categories. Exercise 4 below gives another example of this phenomenon.

In most texts, the arrow category $\mathscr{C}^{\to}$ is defined the way we defined $\mathscr{C}'$.

When being very careful, one would say as above that a model in $\mathscr{C}$ of the graph (4.22) is essentially the same as an arrow in $\mathscr{C}$, and that a u-structure is essentially the same as a model of $\mathscr{U}$ (as in 4.2.15). Frequently, one says more bluntly that a model of (4.22) in $\mathscr{C}$ *is* an arrow in $\mathscr{C}$ and that a u-structure *is* a model of $\mathscr{U}$ (and 'is an **N**-set' – see Exercise 2). This usage is perhaps based on the conception that the description 'model of (4.22) in $\mathscr{C}$' and 'arrow

in $\mathscr{C}$' are two ways of describing the same mathematical object, which exists independently of any particular description. Not all mathematicians share this conception of mathematical objects.

4.2.23 Exercises

1. What is the model of (4.2.7) that is the graph (4.2.7)?

2. Let **N** denote the monoid of nonnegative integers with addition as operation. Give explicit isomorphisms between these categories:

(i) The category **u-Struc** of u-structures and homomorphisms, as defined in 4.2.3.

(ii) $\mathbf{Mod}(\mathscr{U}, \mathbf{Set})$, defined in 4.2.5 and 4.2.14.

(iii) **N-Act**, as defined in 3.2.1.

3. Let $\mathscr{C}$ be a category with object B. Exhibit the slice category $\mathscr{C}/B$ as a subcategory of the arrow category of $\mathscr{C}$ defined in 4.2.17 (see 2.6.10). Is it full?

4. Let $\mathscr{G}$ be the graph with two nodes and no arrows, and $\mathscr{C}$ any category. Show that $\mathbf{Mod}(\mathscr{G}, \mathscr{C})$ is isomorphic to $\mathscr{C} \times \mathscr{C}$.

5. Prove directly, without using Theorem 4.2.20, that in Diagram (4.23), the arrow (h, k) is an isomorphism in the arrow category of $\mathscr{C}$ if and only if h and k are both isomorphisms in $\mathscr{C}$.

6. Let $\mathscr{C}$ and $\mathscr{D}$ be categories and $F : \mathscr{C} \to \mathscr{D}$ an equivalence. Show that every arrow of $\mathscr{D}$ is isomorphic in the arrow category of $\mathscr{D}$ to an arrow in the image of F. (In this sense, an equivalence of categories is 'surjective up to isomorphism' on both objects and arrows. See 3.4.2.)

7. Let $\mathscr{G}$ be a graph and $\mathscr{C}$ a category and let $\mathscr{C}^{\to}$ be the arrow category as in 4.2.17. Show that there is a one-one correspondence between models of $\mathscr{G}$ in $\mathscr{C}^{\to}$ and triples (E, α, F) where E and F are models of $\mathscr{G}$ in $\mathscr{C}$ and $\alpha : E \to F$ is a natural transformation between them. (More about this in Exercise 8 of Section 4.3.)

8. Suppose $D, E : \mathscr{G} \to \mathscr{C}$ are two models of a graph in a category and $\alpha : D \to E$ is a natural isomorphism. Suppose we let, for a a node of $\mathscr{G}$, $\beta a = (\alpha a)^{-1}$. Show that the collection of βa forms a natural transformation (a natural isomorphism in fact) from E to D.

4.3 Natural transformations between functors

A functor is among other things a graph homomorphism, so a natural transformation between two functors is a natural transformation of the corresponding graph homomorphisms. The following proposition is an immediate consequence of 4.2.12.

4.3.1 Proposition *If $\mathscr{C}$ and $\mathscr{D}$ are categories, the functors from $\mathscr{C}$ to $\mathscr{D}$ form a category with natural transformations as arrows.*

We denote this category by $\mathbf{Func}(\mathscr{C}, \mathscr{D})$. Other common notations for it are $\mathscr{D}^{\mathscr{C}}$ and $[\mathscr{C}, \mathscr{D}]$. Tennent [1986] provides an exposition of the use of functor categories for programming language semantics.

Of course, the *graph* homomorphisms from $\mathscr{C}$ to $\mathscr{D}$, which do not necessarily preserve the composition of arrows in $\mathscr{C}$, also form a category $\mathbf{Mod}(\mathscr{C}, \mathscr{D})$ (see 4.2.14), of which $\mathbf{Func}(\mathscr{C}, \mathscr{D})$ is a full subcategory.

A natural transformation from one functor to another is a special case of a natural transformation from one graph homomorphism to another, so the ideas we have presented concerning natural transformations between graph homomorphisms apply to natural transformations between functors as well. In particular, Theorems 4.2.12 and 4.2.20 are true of natural transformations of functors.

If $\mathscr{C}$ is not a small category (see 2.1.5), then $\mathbf{Func}(\mathscr{C}, \mathscr{D})$ may not be locally small (see 2.1.7). This is a rather esoteric question that will not concern us in this book since we will have no occasion to form functor categories of that sort.

We motivated the concept of natural transformation by considering models of graphs, and most of the discussion in the rest of this section concerns that point of view. Historically, the concept first arose for functors and not from the point of view of models.

4.3.2 Examples We have already described some examples of natural transformations, as summed up in the following propositions.

In 3.1.14, we defined the graph homomorphism $\eta\mathscr{G} : \mathscr{G} \to U(F(\mathscr{G}))$ which includes a graph $\mathscr{G}$ into $U(F(\mathscr{G}))$, the underlying graph of the free category $F(\mathscr{G})$.

4.3.3 Proposition *The family of arrows $\eta\mathscr{G}$ form a natural transformation from the identity functor on* **Grf** *to $U \circ F$, where U is the underlying graph functor from* **Cat** *to* **Grf**.

The proof is left as an exercise.

In 3.4.2, we defined the concept of equivalence of categories.

4.3.4 Proposition *A functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ is an equivalence of categories with pseudo-inverse $G : \mathscr{D} \longrightarrow \mathscr{C}$ if and only if $G \circ F$ is naturally isomorphic to $\mathrm{id}_{\mathscr{C}}$ and $F \circ G$ is naturally isomorphic to $\mathrm{id}_{\mathscr{D}}$.*

Proof. Conditions E–2 and E–3 of 3.4.2 can be recast as the statement that $G(F(f)) \circ u_C = u_{C'} \circ f$ and that $F(G(g)) \circ v_D = v_{D'} \circ g$, in other words that the following diagrams commute:

$$\begin{array}{ccc} C & \xrightarrow{u_C} & G(F(C)) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle G(F(f))} \\ C' & \xrightarrow[u_{C'}]{} & G(F(C')) \end{array} \qquad \begin{array}{ccc} D & \xrightarrow{v_D} & F(G(D)) \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle F(G(g))} \\ D' & \xrightarrow[v_{D'}]{} & F(G(D')) \end{array}$$

In this form, they are the statements that u is a natural transformation from $\mathrm{id}_{\mathscr{C}}$ to $G \circ F$ and that v is a natural transformation from $\mathrm{id}_{\mathscr{D}} \longrightarrow F \circ G$. Since each component of u and each component of v is an isomorphism, u and v are natural equivalences. □

4.3.5 Example Let $\alpha : M \times S \longrightarrow S$ and $\beta : M \times T \longrightarrow T$ be two actions by a monoid M (see 3.2.1). Let $\phi : S \longrightarrow T$ be an equivariant map. If F and G are the functors corresponding to α and β, as defined in 3.2.3, then ϕ is the (only) component of a natural transformation from F to G. Conversely, the only component of any natural transformation from F to G is an equivariant map between the corresponding actions.

4.3.6 Example Let $U : \mathbf{Mon} \longrightarrow \mathbf{Set}$ be the underlying functor from the category of monoids. Define $U \times U : \mathbf{Mon} \longrightarrow \mathbf{Set}$ as follows:

(i) For a monoid M, $(U \times U)(M) = U(M) \times U(M)$.

(ii) For a monoid homomorphism $h : M \longrightarrow N$,

$$(U \times U)(h)(m, n) = (h(m), h(n))$$

Then monoid multiplication is a natural transformation from $U \times U$ to U. Formally: Let $\mu : U \times U \longrightarrow U$ be the family of maps whose value at a monoid M is the function $\mu M : U(M) \times U(M) \longrightarrow U(M)$ defined by $\mu M(m, m') = mm'$, the product of m and m' in M. Then μ is a natural transformation. (The function μM is *not* in general a monoid homomorphism, unless M is commutative.)

It is instructive to see why μ is a natural transformation. Let $h : M \longrightarrow N$ be a monoid homomorphism. We must show that the following diagram

commutes:

$$\begin{array}{ccc} (U\times U)(M) & \xrightarrow{\mu M} & U(M) \\ {\scriptstyle (U\times U)(h)}\downarrow & & \downarrow{\scriptstyle h} \\ (U\times U)(N) & \xrightarrow[\mu N]{} & U(N) \end{array} \tag{4.25}$$

The top route takes an element $(m, m') \in (U \times U)(M)$ to $h(mm')$. The lower route takes it to $h(m)h(m')$. The commutativity of the diagram then follows from the fact that h is a homomorphism.

4.3.7 Example The bijection of Exercise 4 of Section 1.2 can be seen to be a natural isomorphism, this time between contravariant functors.

Let B be a fixed set. We define a functor $R : \mathbf{Set}^{\,\mathrm{op}} \to \mathbf{Set}$ such that for a set A, $R(A) = \mathrm{Rel}(A, B)$ as defined in the exercise. For a set function $F : A' \to A$ and relation $\alpha \in \mathrm{Rel}(A, B)$, define $R(F)(\alpha)$ to be the relation $\alpha' \in \mathrm{Rel}(A', B)$ defined by $a'\alpha' b$ if and only if $F(a')\alpha b$. It is easy to see that this makes $R : \mathbf{Set}^{\,\mathrm{op}} \to \mathbf{Set}$ a functor. (Note that $R(A) = \mathbf{Rel}(A, B)$, but R is *not* $\mathrm{Hom}_{\mathbf{Rel}}(-, B)$.)

For each A, let $\phi_A : \mathrm{Rel}(A, B) \to \mathrm{Hom}(A, \mathscr{P}B)$ be the bijection of the exercise. If we check that the functions ϕ_A are the components of a natural transformation from R to $\mathrm{Hom}(A, \mathscr{P}B)$, the transformation will automatically be a natural isomorphism by Theorem 4.2.20. To show that it is natural, let $\alpha \in \mathbf{Rel}(A, B)$ and $a' \in A'$. Then

$$\begin{aligned} \mathrm{Hom}(F, \mathscr{P}B)\left(\phi_A(\alpha)(a')\right) &= \phi_A(\alpha)\left(F(a')\right) = \{b \mid F(a')\alpha b\} \\ &= \{b \mid a'\alpha' b\} = \phi_{A'}\left(R(F)(a')\right) \end{aligned}$$

as required.

This natural isomorphism can be taken to be the defining property of a topos (Section 15.2.2).

4.3.8 Natural transformations involving lists Many of the operations on lists available in functional programming languages can be seen as natural transformations involving the Kleene closure or list functor (3.1.12). For example, one can apply the Kleene closure twice to get the list of lists functor that takes a set A to A^{**}. An element of A^{**} is a list of lists. For example, if $A = \{a, b\}$, one of the elements of A^{**} is $w = ((a, b), (b, b, a), (), (a))$. If $f : A \to B$ is a function, f^{**} takes w to $((f(a), f(b)), (f(b), f(b), f(a)), (), (f(a)))$.

The operation of **flattening** a list simply concatenates the lists in the list; for example, $\mathrm{flatten}(w) = (a, b, b, b, a, a)$. Of course, flatten is a distinct

function for each set A; if we write $\text{flatten}_A : A^{**} \longrightarrow A^*$ for each set A, then flatten is a natural transformation from ** to *, as you can see by checking that this diagram commutes for each function $f : A \longrightarrow B$:

$$\begin{array}{ccc} A^{**} & \xrightarrow{\text{flatten}_A} & A^* \\ {\scriptstyle f^{**}}\downarrow & & \downarrow{\scriptstyle f^*} \\ B^{**} & \xrightarrow[\text{flatten}_B]{} & B^* \end{array} \tag{4.26}$$

Another operation in functional programming languages consists of applying a binary operation to a list. This is called **reduce**, **apply** or **fold**. We shall consider only the case when the binary operation is associative. (When it is not associative, some choice is made about how to associate the list.) This gives a natural transformation from $F \circ U$ to $\text{id}_{\mathbf{Mon}}$, where $F : \mathbf{Set} \longrightarrow \mathbf{Mon}$ is the free monoid functor and $U : \mathbf{Mon} \longrightarrow \mathbf{Set}$ is the underlying set functor. For example, if M is a monoid with elements k, m and n, then $\text{reduce}(k, k, n, m) = k^2nm$, the product of the list using the operation of M. In this case, naturality means that for any monoid homomorphism $h : M \longrightarrow N$, $h \circ \text{reduce}_M = \text{reduce}_N \circ F(U(h))$, which is easily checked.

Note that although both reduce and flatten take lists as arguments, reduce_M is a monoid homomorphism with domain $F(U(M))$ (whose elements are lists), whereas flatten_A is a set function with domain A^{**}. This is reflected by the fact that, when implemented in a programming language, reduce takes an operation as well as a list as argument, but flatten takes only a list.

These and other list operations can often be generalized to sets of expressions instead of sets of lists. In particular, flatten in this more general sense is one of the fundamental constituents of triples (Section 14.3), namely μ.

More about these ideas may be found in [Bird, 1986] and [Spivey, 1989].

4.3.9 Natural transformations of graphs We now consider some natural transformations involving the category **Grf** of graphs and homomorphisms of graphs.

4.3.10 Example In 3.1.9, we defined the functor $N : \mathbf{Grf} \longrightarrow \mathbf{Set}$. It takes a graph $\mathscr{G}$ to its set G_0 of nodes and a homomorphism ϕ to ϕ_0. Now pick a graph with one node $*$ and no arrows and call it $\mathscr{E}$. Let $V = \text{Hom}_{\mathbf{Grf}}(\mathscr{E}, -)$.

A graph homomorphism from the graph $\mathscr{E}$ to an arbitrary graph $\mathscr{G}$ is evidently determined by the image of $\mathscr{E}$ and that can be any node of $\mathscr{G}$. In other words, nodes of $\mathscr{G}$ are 'essentially the same thing' as graph homomorphisms from $\mathscr{E}$ to $\mathscr{G}$, that is, as the elements of the set $V(\mathscr{G})$.

We can define a natural transformation $\alpha : V \longrightarrow N$ by defining

$$\alpha\mathscr{G}(f) = f_0(*)$$

where $\mathscr{G}$ is a graph and $f : \mathscr{E} \longrightarrow \mathscr{G}$ is a graph homomorphism (arrow of **Grf**). There must be a naturality diagram (4.20) for each arrow of the source category, which in this case is **Grf**. Thus to see that α is natural, we require that for each graph homomorphism $g : \mathscr{G}_1 \longrightarrow \mathscr{G}_2$, the diagram

$$\begin{array}{ccc} V\mathscr{G}_1 & \xrightarrow{Vg} & V\mathscr{G}_2 \\ {\scriptstyle \alpha\mathscr{G}_1}\downarrow & & \downarrow{\scriptstyle \alpha\mathscr{G}_2} \\ N\mathscr{G}_1 & \xrightarrow[Ng]{} & N\mathscr{G}_2 \end{array}$$

commutes. Now Ng is g_0 (the node map of g) by definition, and the value of V (which is a hom functor) at a homomorphism g composes g with a graph homomorphism from the graph $\mathscr{E}$. Then we have, for a homomorphism $f : \mathscr{E} \longrightarrow \mathscr{G}_1$ (i.e., an element of the upper left corner of the diagram),

$$(\alpha\mathscr{G}_2 \circ Vg)(f) = \alpha\mathscr{G}_2(g \circ f) = (g \circ f)_0(*)$$

while

$$(Ng \circ \alpha\mathscr{G}_1)(f) = Ng(f_0(*)) = g_0(f_0(*))$$

and these are equal from the definition of composition of graph homomorphisms.

The natural transformation α is in fact a natural isomorphism (Exercise 9). This shows that N is naturally isomorphic to a hom functor. Such functors are called 'representable', and are considered in greater detail in 4.5.1.

4.3.11 Connected components A node a can be **connected** to the node b of a graph $\mathscr{G}$ if it is possible to get from a to b following a sequence of arrows of $\mathscr{G}$ in either direction. In order to state this more precisely, let us say that an arrow a 'has' a node n if n is the domain or the codomain (or both) of a. Then a is connected to b means that there is a sequence $(c_0, c_1, \ldots, c_n)$ of arrows of $\mathscr{G}$ with the property that a is a node (either the source or the target) of c_0, b is a node of c_n, and for $i = 1, \ldots, n$, c_{i-1} and c_i have a node in common. We call such a sequence an **undirected path** between a and b.

It is a good exercise to see that 'being connected to' is an equivalence relation. (For reflexivity: a node is connected to itself by the empty sequence.)

An equivalence class of nodes with respect to this relation is called a **connected component** of the graph $\mathscr{G}$, and the set of connected components is called $W\mathscr{G}$.

Connected components can be defined for categories in the same way as for graphs. In that case, each connected component is a full subcategory.

If $f : \mathscr{G} \to \mathscr{H}$ is a graph homomorphism and if two nodes a and b are in the same component of $\mathscr{G}$, then $f(a)$ and $f(b)$ are in the same component of $\mathscr{H}$; this is because f takes an undirected path between a and b to an undirected path between $f(a)$ and $f(b)$. Thus the arrow f induces a function $Wf : W\mathscr{G} \to W\mathscr{H}$, namely the one which takes the component of a to the component of $f(a)$; and this makes W a functor from **Grf** to **Set**.

For a graph $\mathscr{G}$, let $\beta\mathscr{G} : N\mathscr{G} \to W\mathscr{G}$ be the set function which takes a node of $\mathscr{G}$ to the component of $\mathscr{G}$ that contains that node. (The component is the *value* of $\beta\mathscr{G}$ at the node, not the codomain.) Then $\beta : N \to W$ is a natural transformation. It is instructive to check the commutativity of the requisite diagram.

4.3.12 Example In 3.5.9, we described a functor Σ which provided a meaning in **Set** for each program in the programming language L of 2.2.5. A person more oriented to machine language might have preferred to give the meaning of all the data in terms of numbers, in particular the integers between 0 and 2^K for some fixed number $K \geq 7$ (the constraint is to accommodate the ASCII codes).

Thus one could define a functor Σ' for which

(i) $\Sigma'(\texttt{NAT})$ is the set of integers between 0 and $2^K - 1$. Then the constant 0 would be the number 0, but $\Sigma'(\text{succ})$ would have to calculate the successor modulo 2^K.

(ii) $\Sigma'(\texttt{BOOLEAN})$ is the set $\{0, 1\}$, with $\texttt{true} = 1$ and $\texttt{false} = 0$.

(iii) For each character c, $\Sigma'(c)$ is the ASCII code for c. Then we would have to take $\Sigma'(\texttt{CHAR})$ to be the set $A = \{n \in \mathbf{N} \mid 0 \leq n \leq 127\}$.

For each of the three types T in our language, we have a way of rewriting each datum in $\Sigma(T)$ to become the corresponding datum in $\Sigma'(T)$. This rewriting becomes a function $\beta T : \Sigma(T) \to \Sigma'(T)$:

(i) $\beta(\texttt{NAT})(n)$ is n modulo 2^K.

(ii) $\beta(\texttt{BOOLEAN})(\texttt{true}) = 1$ and $\beta(\texttt{BOOLEAN})(\texttt{false}) = 0$.

(iii) $\beta(\texttt{CHAR})(c)$ is the ASCII code of c for each character c.

In order to preserve the intended meaning, $\Sigma'(\text{succ})$ would have to be the successor function modulo 2^K, $\Sigma'(\texttt{ord})$ would have to be the inclusion of A into $\Sigma'(\texttt{NAT})$ and $\Sigma'(\texttt{chr})$ would have to be the function from $\Sigma'(\texttt{NAT})$ to A which takes the remainder modulo 128.

Preserving the meaning of `ord` (an informal idea) means formally that this diagram must commute, as it does with the definitions given of $\Sigma(\texttt{ord})$ and $\Sigma'(\texttt{ord})$:

$$\begin{array}{ccc} \Sigma(\texttt{CHAR}) & \xrightarrow{\beta(\texttt{CHAR})} & \Sigma'(\texttt{CHAR}) \\ \Sigma(\texttt{ord})\downarrow & & \downarrow\Sigma'(\texttt{ord}) \\ \Sigma(\texttt{NAT}) & \xrightarrow[\beta(\texttt{NAT})]{} & \Sigma'(\texttt{NAT}) \end{array}$$

Similar remarks apply to the preservation of the other operations. This is a special case of a general principle that, given two functors G and G' which are semantics in some sense, a natural transformation $\beta : G \longrightarrow G'$ can be said to preserve the meaning of the operations.

The natural transformation β was constructed for the given data types. The only constructor in L, namely composition, does not destroy the natural transformation property: if the given β gives the naturality property for primitive operations, it does so for all their composites, as well. This is an instance of the following proposition.

4.3.13 Proposition *Let $F, G : \mathscr{C} \longrightarrow \mathscr{D}$ be functors. Let S be a (possibly empty) set of arrows of $\mathscr{C}$ with the property that every arrow of $\mathscr{C}$ is a composite of arrows of S and let $\beta C : F(C) \longrightarrow G(C)$ be an arrow of $\mathscr{D}$ for each object C of $\mathscr{C}$. Suppose for every arrow $f : A \longrightarrow B$ of S this diagram commutes:*

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ \beta A\downarrow & & \downarrow\beta B \\ G(A) & \xrightarrow[G(f)]{} & G(B) \end{array} \tag{4.27}$$

Then β is a natural transformation.

Proof. if $f : A \longrightarrow B$ and $g : B \longrightarrow C$ are arrows of S, then the outer rectangle below commutes because the two squares do:

$$\begin{array}{ccccc} F(A) & \xrightarrow{F(f)} & F(B) & \xrightarrow{F(g)} & F(C) \\ \beta A\downarrow & & \downarrow\beta B & & \downarrow\beta C \\ G(A) & \xrightarrow[G(f)]{} & G(B) & \xrightarrow[G(g)]{} & G(C) \end{array}$$

The naturality diagram for the case $f = \mathrm{id}$ (that is, the empty composite) is automatic. The proof follows from these facts by induction. □

4.3.14 Exercises

1. Prove Proposition 4.3.3.

2. Show that the family $\beta\mathscr{G}$ of arrows taking a node to its component defined in 4.3.11 is indeed a natural transformation.

3. Let $\mathscr{C}$ be a category. A **subfunctor** of a functor $F : \mathscr{C} \to \mathbf{Set}$ is a functor $G : \mathscr{C} \to \mathbf{Set}$ with the property that for each object C of $\mathscr{C}$, $G(C) \subseteq F(C)$ and such that for each arrow $f : C \to C'$ and each element $x \in GC$, we have that $Gf(x) = Ff(x)$. Show that the inclusion function $i_C : G(C) \to F(C)$ is a natural transformation.

4. Show that the map which takes an arrow of a graph to its source is a natural transformation from A to N. (See 3.1.9.) Do the same for targets. (Actually, every operation in any multisorted algebraic structure gives a natural transformation. Example 4.3.6 was another example of this. See [Linton, 1969b], [Linton, 1969a].)

5.† Show that if $\mathscr{C}$ is a discrete category with set of objects $\mathscr{C}_0$ (hence essentially a set), then $\mathbf{Func}(\mathscr{C}, \mathbf{Set})$ is equivalent to the slice category $\mathbf{Set}/\mathscr{C}_0$. (See 2.6.10.)

6. For each set S, let $\{\}S : S \to \mathscr{P}S$ be the function which takes an element x of S to the singleton subset $\{x\}$.

a. Show that $\{\}$ is a natural transformation from the identity functor on **Set** to the direct image powerset functor $\mathscr{P}$. (See 3.1.16.)

b. Show that $\{\}$ is not a natural transformation from the identity functor on **Set** to the universal image powerset functor. (See 3.1.19.)

c. Explain why it does not even make sense to ask whether it is a natural transformation to the inverse image powerset functor.

7. Verify the claims in 4.3.5.

8. Show that the results of Exercise 7 of Section 4.2 are still true if we replace the graph $\mathscr{G}$ by a category $\mathscr{D}$.

9. Show that the natural transformation of 4.3.10 is a natural isomorphism.

10. a. Show that for any integer k, the set of paths of length k of a graph is the object part of a functor $P_k : \mathbf{Grf} \to \mathbf{Set}$. (See 2.1.2.)

b. Show that P_0 is naturally isomorphic to the node functor N defined in 3.1.9.

c. Show that P_1 is naturally isomorphic to the arrow functor A of 3.1.9.

4.4 The Godement calculus of natural transformations

We collect here, mostly without proof, some of the basic combinatorial properties of functors and natural transformations. These rules were first codified by Godement [Godement, 1958]. The notation given in Definitions 4.4.2 and 4.4.3 is used throughout the book, but the remainder of this section is used only in Section 4.8. Verifying (some of) the Godement properties is an excellent way to familiarize yourself with natural transformations.

4.4.1 Let $F : \mathscr{A} \to \mathscr{B}$ and $G : \mathscr{B} \to \mathscr{C}$ be functors. There is a composite functor $G \circ F : \mathscr{A} \to \mathscr{C}$ defined in the usual way by $G \circ F(A) = G(F(A))$. Similarly, let H, K and L be functors from $\mathscr{A} \to \mathscr{B}$ and $\alpha : H \to K$ and $\beta : K \to L$ be natural transformations. Recall that this means that for each object A of $\mathscr{A}$, $\alpha A : HA \to KA$ and $\beta A : KA \to LA$. Then as in 4.2.11, we define $\beta \circ \alpha : H \to L$ by

$$(\beta \circ \alpha)A = \beta A \circ \alpha A$$

Things get more interesting when we mix functors and natural transformations. For example, suppose we have three categories $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$, four functors, two of them, $F, G : \mathscr{A} \to \mathscr{B}$ and the other two $H, K : \mathscr{B} \to \mathscr{C}$, and two natural transformations $\alpha : F \to G$ and $\beta : H \to K$. We picture this situation as follows:

$$\mathscr{A} \;\underset{G}{\overset{F}{\Downarrow \alpha}}\; \mathscr{B} \;\underset{K}{\overset{H}{\Downarrow \beta}}\; \mathscr{C} \tag{4.28}$$

4.4.2 Definition The natural transformation $\beta F : H \circ F \to K \circ F$ is defined by the formula $(\beta F)A = \beta(FA)$ for an object A of $\mathscr{A}$.

The notation $\beta(FA)$ means the component of the natural transformation β at the object FA. This is indeed an arrow from $H(F(A)) \to K(F(A))$ as required. To show that βF is natural requires showing that for an arrow $f : A \to A'$ of $\mathscr{A}$, the diagram

$$\begin{array}{ccc} H(F(A)) & \xrightarrow{\beta FA} & K(F(A)) \\ {\scriptstyle H(F(f))}\downarrow & & \downarrow{\scriptstyle K(F(f))} \\ H(F(A')) & \xrightarrow[\beta FA']{} & K(F(A')) \end{array} \tag{4.29}$$

commutes, but this is just the naturality diagram of β applied to the arrow $F(f) : F(A) \longrightarrow F(A')$.

4.4.3 Definition The natural transformation $H\alpha : H \circ F \longrightarrow H \circ G$ is defined by letting $(H\alpha)A = H(\alpha A)$ for an object A of $\mathscr{A}$, that is the value of H applied to the arrow αA.

To see that $H\alpha$ thus defined is natural requires showing that

$$\begin{array}{ccc} H(F(A)) & \xrightarrow{H(\alpha A)} & H(G(A)) \\ {\scriptstyle H(F(f))}\downarrow & & \downarrow{\scriptstyle H(G(f))} \\ H(F(A')) & \xrightarrow[H(\alpha A')]{} & H(G(A')) \end{array}$$

commutes. This diagram is obtained by applying the functor H to the naturality diagram of α. Since functors preserve commutative diagrams, the result follows.

Note that the proofs of naturality for βF and for $H\alpha$ are quite different. For example, the second requires that H be a functor, while the first works if F is merely an object function.

The definitions of βF and $H\alpha$ are quite different, in fact. The first is the natural transformation whose value at an object A of $\mathscr{A}$ is the component of β on the object FA while the value of the second is the result of applying the functor H to the component αA (which is an arrow of $\mathscr{B}$). Nevertheless, we use similar notations. The reason for this is that their formal properties are indistinguishable. In fact, even categorists quite commonly (though not universally) distinguish them by writing β_F but $H\alpha$. That notation emphasizes the fact that they are semantically different. The notation used here is chosen to emphasize the fact that they are syntactically indistinguishable. More precisely, the left/right mirror image of each of Godement's rules given below is again a Godement rule.

In a great deal of mathematical reasoning, one forgets the semantics of the situation except at the beginning and the end of the process, relying on the syntactic rules in the intermediate stages. This is especially true in the kind of 'diagram chasing' arguments so common in category theory. For that reason, the notation we have adopted emphasizes the syntactic similarity of the two constructions, rather than the semantic difference.

In Exercise 2, we give another, more sophisticated definition of βF and $H\alpha$ which shows that they can be thought of as *semantically* parallel, as well.

4.4.4 There is a second way of composing natural transformations. The naturality of β in Diagram (4.29) implies that for any object A of $\mathscr{A}$, the diagram

$$\begin{array}{ccc} (H \circ F)A & \xrightarrow{(H\alpha)A} & (H \circ G)A \\ {\scriptstyle (\beta F)A}\downarrow & & \downarrow{\scriptstyle (\beta G)A} \\ (K \circ F)A & \xrightarrow[(K\alpha)A]{} & (K \circ G)A \end{array} \tag{4.30}$$

commutes. We define $\beta * \alpha : H \circ F \longrightarrow K \circ G$ by requiring that its component at A be $(K\alpha)A \circ (\beta F)A$, which of course is the same as $(\beta G)A \circ (H\alpha)A$.

4.4.5 Proposition *$\beta * \alpha$ is a natural transformation.*

Proof. We have that $\beta * \alpha = K\alpha \circ \beta F$ by definition of $\beta * \alpha$ and Definition 4.2.11. It is therefore a natural transformation by Proposition 4.2.12. □

We usually call $\beta \circ \alpha$ the **vertical composite** and $\beta * \alpha$ the **horizontal composite**. (Warning: Some authors use $\circ$ for the horizontal composite.) One must keep careful track of the difference between them. Fortunately, the notations do not often clash, since usually only one makes sense.

There is one case in which the two notations can clash. If $\mathscr{A} = \mathscr{B} = \mathscr{C}$ and $G = H$, then $\beta \circ \alpha : F \longrightarrow K$, while $\beta * \alpha : G \circ F \longrightarrow K \circ G$. This clash is exacerbated by the habit among many categorists of omitting the composition circle and $*$, except for emphasis. We will often omit the $*$, but not the circle. On the other hand, no confusion can possibly arise from the overloading of the circle notation to include composition of arrows, functors and natural transformations since their domains uniquely define what kind of composition is involved.

4.4.6 Proposition *Horizontal composition of natural transformations is associative.*

Proof. In the situation,

$$\mathscr{A} \underset{G}{\overset{F}{\Downarrow \alpha}} \mathscr{B} \underset{K}{\overset{H}{\Downarrow \beta}} \mathscr{C} \underset{M}{\overset{L}{\Downarrow \gamma}} \mathscr{D} \tag{4.31}$$

we have that

$$\gamma * (\beta * \alpha) = \gamma KG \circ L(\beta G \circ H\alpha) = \gamma KG \circ L\beta G \circ LH\alpha$$

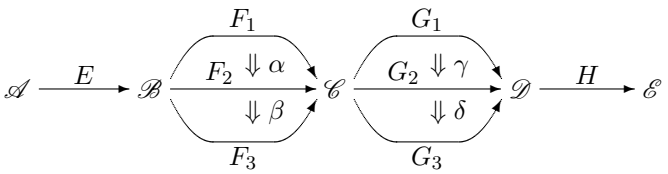
because L is a functor, while

$$(\gamma * \beta) * \alpha = (\gamma K \circ L\beta)G \circ LH\alpha = \gamma KG \circ L\beta G \circ LH\alpha$$

by Definition 4.2.11. □

4.4.7 Godement's five rules There are thus several kinds of composites. There is a composite of functors, vertical and horizontal composite of natural transformations and the composite of a functor and a natural transformation in either order (although the latter is in fact the horizontal composite of a natural transformation and the identity natural transformation of a functor, a fact we leave to an exercise). The possibilities are sufficiently numerous that it is worth the effort to codify the rules.

Let $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$, $\mathscr{D}$ and $\mathscr{E}$ be categories; $E : \mathscr{A} \to \mathscr{B}$, F_1, F_2 and $F_3 : \mathscr{B} \to \mathscr{C}$, G_1, G_2 and $G_3 : \mathscr{C} \to \mathscr{D}$, and $H : \mathscr{D} \to \mathscr{E}$ be functors; and $\alpha : F_1 \to F_2$, $\beta : F_2 \to F_3$, $\gamma : G_1 \to G_2$, and $\delta : G_2 \to G_3$ be natural transformations. This situation is summarized by the following diagram:



(4.32)

Then

G–1 $(\delta \circ \gamma)(\beta \circ \alpha) = (\delta\beta) \circ (\gamma\alpha)$.
G–2 $(H \circ G_1)\alpha = H(G_1\alpha)$.
G–3 $\gamma(F_1 \circ E) = (\gamma F_1)E$.
G–4 $G_1(\beta \circ \alpha)E = (G_1\beta E) \circ (G_1\alpha E)$.
G–5 $\gamma\alpha = (\gamma F_2) \circ (G_1\alpha) = (G_2\alpha) \circ (\gamma F_1)$.

The expression $G_1(\beta \circ \alpha)E$ in G–4 is not ambiguous because of Exercise 1. G–1 is called the **Interchange Law**. It is the basis for the definition of 2-category in Section 4.8.

4.4.8 Exercises

1. Show that, using the notation of the Godement rules, $(G_1\alpha)E = G_1(\alpha E)$.

2. Show that in the Diagram (4.28), the composites βF and $H\alpha$ are the horizontal composites $\beta * \mathrm{id}_F$ and $\mathrm{id}_H * \alpha$ respectively.

3. a. Show (using Exercise 2) that Godement's fifth rule is an instance of the first.

b. Show that Godement's fourth rule follows from the first and the associativity of horizontal composition.

4.5 The Yoneda Lemma and universal elements

For an arbitrary category $\mathscr{C}$, the functors from $\mathscr{C}$ to **Set** are special because the hom functors $\mathrm{Hom}(C, -)$ for each object C of $\mathscr{C}$ are set-valued functors. In this section, we introduce the concept of representable functor, the Yoneda Lemma, and universal elements, all of which are based on these hom functors. These ideas have turned out to be fundamental tools for categorists. They are also closely connected with the concept of adjunction, to be discussed later (note Theorem 13.3.2 and Proposition 13.3.6).

If you are familiar with group theory, it may be illuminating to realize that representable functors are a generalization of the regular representation, and the Yoneda embedding is a generalization of Cayley's Theorem.

We have already considered set-valued functors as actions in Section 3.2.

4.5.1 Representable functors A functor from a category $\mathscr{C}$ to the category of sets (a **set-valued functor**) is said to be **representable** if it is naturally isomorphic to a hom functor; see 3.1.20. A covariant functor is representable if it is naturally isomorphic to $\mathrm{Hom}(C, -)$ for some object C of $\mathscr{C}$; in this case one says that C **represents** the functor. A contravariant functor is representable if it is naturally isomorphic to $\mathrm{Hom}(-, C)$ for some object C (and then C represents the contravariant functor).

We have already looked at one example of representable functor in some detail in 4.3.10, where we showed that the set-of-nodes functor for graphs is represented by the graph with one node and no arrows. The set-of-arrows functor is represented by the graph with two nodes and one arrow between them (Exercise 3).

4.5.2 The Yoneda embedding Let $\mathscr{C}$ be a category. There is a functor $Y : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Func}(\mathscr{C}, \mathbf{Set})$, the **Yoneda functor**, defined as follows. Note that Y must take an object of $\mathscr{C}$ to a functor and an arrow of $\mathscr{C}$ to a natural transformation.

Y–1 For an object C of $\mathscr{C}$, $Y(C) = \mathrm{Hom}(C, -)$.

Y–2 If $f : D \longrightarrow C$ in $\mathscr{C}$ and A is an object of $\mathscr{C}$, then the component $Y(f)A$ of $Y(f) : \mathrm{Hom}(C, -) \longrightarrow \mathrm{Hom}(D, -)$ is $\mathrm{Hom}(f, A) : \mathrm{Hom}(C, A) \longrightarrow \mathrm{Hom}(D, A)$ (see 3.1.21).

Note that $Y(C)$ is a *covariant* hom functor and that $Y(f)A$ is a component of a *contravariant* hom functor.

To see that $Y(f)$ is a natural transformation requires checking that this diagram commutes for every arrow $k : A \longrightarrow B$ of $\mathscr{C}$:

$$\begin{array}{ccc} \mathrm{Hom}(D,A) & \xrightarrow{\mathrm{Hom}(D,k)} & \mathrm{Hom}(D,B) \\ {\scriptstyle Y(f)A}\uparrow & & \uparrow{\scriptstyle Y(f)B} \\ \mathrm{Hom}(C,A) & \xrightarrow[\mathrm{Hom}(C,k)]{} & \mathrm{Hom}(C,B) \end{array}$$

To see that it commutes, start with $h : C \longrightarrow A$, an arbitrary element of the lower left corner. The lower route takes this to $k \circ h$, then to $(k \circ h) \circ f$. The upper route takes it to $k \circ (h \circ f)$, so the fact that the diagram commutes is simply a statement of the associative law. In a monoid, that this diagram commutes is the statement that the function defined by left multiplying by a given element commutes with the function defined by right multiplying by another given element.

$Y(f) : \mathrm{Hom}(C, -) \longrightarrow \mathrm{Hom}(D, -)$ is the **induced natural transformation** corresponding to f.

The main theorem concerning Y is the following.

4.5.3 Theorem *$Y : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Func}(\mathscr{C}, \mathbf{Set})$ is a full and faithful functor.*

The fact that Y is full and faithful is encapsulated in the following remarkable corollary.

4.5.4 Corollary *Every natural transformation*

$$\mathrm{Hom}(C, -) \longrightarrow \mathrm{Hom}(D, -)$$

is given by composition with a unique arrow $D \longrightarrow C$. The natural transformation is an isomorphism if and only if the corresponding arrow $D \longrightarrow C$ is an isomorphism. In particular, if $F : \mathscr{C} \longrightarrow \mathbf{Set}$ is represented by both C and D, then $C \cong D$.

This means that you can construct an arrow in a category by constructing a natural transformation between hom functors. This is one of the most widely used techniques in category theory.

Proof. Theorem 4.5.3 is an immediate consequence of the Yoneda Lemma (4.5.8). We give a direct proof here. This proof is an excellent exercise in manipulating natural transformations and hom sets.

Let $f, g : D \longrightarrow C$ in $\mathscr{C}$. The component

$$Y(f)C : \mathrm{Hom}(C, C) \longrightarrow \mathrm{Hom}(D, C)$$

of the natural transformation $Y(f)$ at C takes id_C to f, and similarly $Y(g)C$ takes id_C to g. Thus if $f \neq g$, then $Y(f)C \neq Y(g)C$, so that $Y(f) \neq Y(g)$. Thus Y is faithful.

We must show that Y is full. Given $\phi : \mathrm{Hom}(C, -) \longrightarrow \mathrm{Hom}(D, -)$, we get the required $f : D \longrightarrow C$ by one of the basic tricks of category theory: we define $f = \phi C(\mathrm{id}_C)$. The component of ϕ at C is a function $\phi C : \mathrm{Hom}(C, C) \longrightarrow \mathrm{Hom}(D, C)$, so this definition makes sense.

To complete the proof, we must prove that if $k : C \longrightarrow A$ is any arrow of $\mathscr{C}$, then $\phi A(k) = k \circ f : D \longrightarrow A$. This follows from the fact that the following diagram commutes by naturality of ϕ:

$$\begin{array}{ccc} \mathrm{Hom}(C, C) & \xrightarrow{\mathrm{Hom}(C,k)} & \mathrm{Hom}(C, A) \\ \Big\downarrow{\scriptstyle \phi C} & & \Big\downarrow{\scriptstyle \phi A} \\ \mathrm{Hom}(D, C) & \xrightarrow[\mathrm{Hom}(D,k)]{} & \mathrm{Hom}(D, A) \end{array}$$

If you start in the northwest corner with id_C, the upper route takes you to $\phi A(k)$ in the southeast corner, whereas the lower route takes you to $k \circ f$, as required. □

4.5.5 By replacing $\mathscr{C}$ by $\mathscr{C}^{\mathrm{op}}$ in Theorem 4.5.3, we derive a second Yoneda functor $J : \mathscr{C} \longrightarrow \mathbf{Func}(\mathscr{C}^{\mathrm{op}}, \mathbf{Set})$ which is also full and faithful. For an object C of $\mathscr{C}$, $J(C) = \mathrm{Hom}(-, C)$, the contravariant hom functor. If $f : C \longrightarrow D$ in $\mathscr{C}$ and A is an object of $\mathscr{C}$, then the component

$$J(f)A : \mathrm{Hom}(A, C) \longrightarrow \mathrm{Hom}(A, D)$$

of the natural transformation $J(f) : \mathrm{Hom}(-, C) \longrightarrow \mathrm{Hom}(-, D)$ is

$$\mathrm{Hom}(A, f) : \mathrm{Hom}(A, C) \longrightarrow \mathrm{Hom}(A, D)$$

The fact that J is full and faithful means that an arrow from A to B of $\mathscr{C}$ can be uniquely defined by giving a natural transformation from $\mathrm{Hom}(-, A)$ to $\mathrm{Hom}(-, B)$. This statement is the dual of Corollary 4.5.4. Such a natural transformation $\alpha : \mathrm{Hom}(-, A) \longrightarrow \mathrm{Hom}(-, B)$ has a component $\alpha T : \mathrm{Hom}(T, A) \longrightarrow \mathrm{Hom}(T, B)$ for each object T of $\mathscr{C}$. The effect of this is that you can define an arrow from A to B by giving a function $\alpha T : \mathrm{Hom}(T, A) \longrightarrow \mathrm{Hom}(T, B)$ for each object T which prescribes a variable element of B for each variable element of A (as described in 2.8.2), in such

a way that for each $f : T' \longrightarrow T$, the diagram

$$\begin{array}{ccc} \mathrm{Hom}(T, A) & \xrightarrow{\alpha T} & \mathrm{Hom}(T, B) \\ {\scriptstyle \mathrm{Hom}(f,A)}\downarrow & & \downarrow{\scriptstyle \mathrm{Hom}(f,B)} \\ \mathrm{Hom}(T', A) & \xrightarrow[\alpha T']{} & \mathrm{Hom}(T', B) \end{array}$$

commutes. This can be summed up by saying, 'An arrow is induced by defining its value on each variable element of its domain, provided that the definition is natural with respect to change of parameters.'

4.5.6 Elements of a set-valued functor Corollary 4.5.4 says that any natural transformation from $\mathrm{Hom}(C, -)$ to $\mathrm{Hom}(D, -)$ is given by a unique arrow from D to C, that is, by an element of $\mathrm{Hom}(D, C) = \mathrm{Hom}(D, -)(C)$. Remarkably, the result remains true when $\mathrm{Hom}(D, -)$ is replaced by an arbitrary set-valued functor.

Suppose $F : \mathscr{C} \longrightarrow \mathbf{Set}$ is a functor and C is an object of $\mathscr{C}$. An element $c \in F(C)$ induces a natural transformation from the representable functor $\mathrm{Hom}(C, -)$ to F by the formula

$$f \mapsto F(f)(c) \tag{4.33}$$

That is, if $f : C \longrightarrow C'$ is an element of $\mathrm{Hom}(C, C')$, the definition of functor requires an induced function $F(f) : F(C) \longrightarrow F(C')$ and this function can be evaluated at $c \in F(C)$.

4.5.7 Proposition *Formula (4.33) defines a natural transformation*

$$\mathrm{Hom}(C, -) \longrightarrow F$$

Proof. Let $\alpha C' : \mathrm{Hom}(C, C') \longrightarrow F(C')$ take f to $F(f)(c)$ for $c \in F(C)$. We must show that for any $g : C' \longrightarrow B$,

$$\begin{array}{ccc} \mathrm{Hom}(C, C') & \xrightarrow{\alpha C'} & F(C') \\ {\scriptstyle \mathrm{Hom}(C,g)}\downarrow & & \downarrow{\scriptstyle F(g)} \\ \mathrm{Hom}(C, B) & \xrightarrow[\alpha B]{} & F(B) \end{array} \tag{4.34}$$

commutes. We have, for $f \in \mathrm{Hom}(C, C')$,

$$\begin{aligned} \alpha B(\mathrm{Hom}(C, g)(f)) &= \alpha B(g \circ f) = F(g \circ f)(c) \\ &= F(g)(F(f)(c)) = F(g)(\alpha C'(f)) \end{aligned}$$

as required.

4.5.8 Theorem (Yoneda Lemma) *Formula (4.33) defines a one to one correspondence between elements of $F(C)$ and natural transformations*

$$\mathrm{Hom}(C,-) \longrightarrow F$$

Proof. Suppose that c and c' are different elements of $F(C)$. Then the natural transformation corresponding to c takes id_C to c whereas the one corresponding to c' takes id_C to c'. Thus the mapping of the Yoneda Lemma is injective.

Suppose $\beta : \mathrm{Hom}(C,-) \longrightarrow F$ is a natural transformation. Then we have $\beta C : \mathrm{Hom}(C,C) \longrightarrow F(C)$. Let $c = \beta C(\mathrm{id}_C) \in F(C)$. For any $f : C \longrightarrow C'$, the naturality of β gives that

$$\beta C'(\mathrm{Hom}(C,f)(\mathrm{id}_C)) = F(f)(\beta C)(\mathrm{id}_C)$$

The left hand side is $\beta C'(f)$ and the right hand side is $F(f)(c)$. Thus β is the natural transformation given by Formula (4.33), so that the mapping of the Yoneda Lemma is surjective. □

4.5.9 Definition Let $F : \mathscr{C} \longrightarrow \mathbf{Set}$ be a functor and let c be an element of $F(C)$ for some object C of $\mathscr{C}$. If the natural transformation from $\mathrm{Hom}(C,-)$ to F induced by c is an isomorphism, then c is a **universal element** of F.

The existence of a universal element means that F is representable (see 4.5.1). The converse is also true because a natural isomorphism $\alpha : \mathrm{Hom}(C,-) \longrightarrow F$ is, from the Yoneda lemma, induced by a unique element c of $F(C)$ and by definition α is an isomorphism if and only if c is universal.

The unique element c can be calculated using the following fact.

4.5.10 Proposition *Let $\alpha : \mathrm{Hom}(C,-) \longrightarrow F$ be a natural isomorphism. The unique element $c \in F(C)$ inducing α is $\alpha C(\mathrm{id}_C)$.*

Proof. For an arbitrary $f : C \longrightarrow C'$, $\alpha C'(f) = F(f)(\alpha C(\mathrm{id}_C))$ because this diagram must commute (chase id_C around the square):

$$\begin{array}{ccc} \mathrm{Hom}(C,C) & \xrightarrow{\alpha C} & F(C) \\ {\scriptstyle \mathrm{Hom}(C,f)}\downarrow & & \downarrow{\scriptstyle F(f)} \\ \mathrm{Hom}(C,C') & \xrightarrow[\alpha C']{} & F(C') \end{array}$$

Then, by Formula (4.33), $\alpha C(\mathrm{id}_C)$ must be the required unique element c. □

A detailed example of the use of this construction is in the proof of Proposition 5.2.14.

4.5.11 The definition of universal element can be reworded in elementary terms using Formula (4.33), as follows.

4.5.12 Proposition *Let $F : \mathscr{C} \to \mathbf{Set}$ be a functor, C an object of $\mathscr{C}$ and c an element of $F(C)$. Then c is a universal element of F if and only if for any object C' of $\mathscr{C}$ and any element $x \in F(C')$ there is a unique arrow $f : C \to C'$ of $\mathscr{C}$ for which $x = F(f)(c)$.*

Proof. If c is a universal element then the mapping (4.33) must be an isomorphism, hence every component must be bijective by Theorem 4.2.20. This immediately ensures the existence and uniqueness of the required arrow f. Conversely, the existence and uniqueness of f for each C' and $x \in F(C')$ means that there is a bijection $\alpha C' : \mathrm{Hom}(C, C') \to F(C')$ for every C' which takes $f : C \to C'$ to $F(f)(c)$. By Proposition 4.5.7, these are the components of a natural transformation, which is therefore a natural isomorphism by Theorem 4.2.20. □

In the case of a functor $F : \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$, c in $F(C)$ is a universal element if for any object C' of $\mathscr{C}$ and any element $x \in F(C')$ there is a unique arrow $f : C' \to C$ for which $x = F(f)(c)$.

4.5.13 Corollary *If $c \in F(C)$ and $c' \in F(C')$ are universal elements, then there is a unique isomorphism $f : C \to C'$ such that $F(f)(c) = c'$.*

The proof is left as Exercise 10.

Universal elements are considered again in Proposition 13.3.6. The exposition in [Mac Lane, 1971] uses the concept of universal element (defined in the manner of the preceding proposition) as the central idea in discussing representable functors and adjunction.

4.5.14 Exercises

1. Show that a representable functor preserves terminal objects but not necessarily initial objects.

2. Show that $\mathrm{Hom}_{\mathbf{Set}}(1, -)$ is naturally isomorphic to the identity functor on **Set**. ('A set is its set of global elements.' In terms of 4.5.1 this says that a singleton set represents the identity functor on **Set**.)

3. Show that the arrow functor $A : \mathbf{Grf} \to \mathbf{Set}$ of 3.1.9 is represented by the graph **2** which is pictured as

$$1 \xrightarrow{\ e\ } 2$$

(Compare Exercise 10 of Section 4.3.)

4. Show that the set of objects of a small category is 'essentially the same thing' as the set of global elements of the category (as an object of **Cat**), and translate this into a natural isomorphism following the pattern of 4.3.10.

5. Is the set of arrows of a small category the object part of a functor? If it is, is it representable?

6. Prove that any set-valued functor $F : \mathscr{C} \longrightarrow \mathbf{Set}$ is naturally isomorphic to a functor for which if C and D are distinct objects of $\mathscr{C}$, then $F(C)$ and $F(D)$ are disjoint sets.

7. Let $D : \mathbf{Set} \longrightarrow \mathbf{Set}$ be the functor for which for a set A, $D(A) = A \times A$, and for a function $f : A \longrightarrow B$, $D(f) : A \times A \longrightarrow B \times B$ is the function defined by $D(f)(a_1, a_2) = (f(a_1), f(a_2))$. Show that D is representable and find a universal element for D.

8. Show that the second Yoneda embedding J defined in 4.5.5 is full and faithful.

9.† Formulate carefully and prove that the equivalence in the Yoneda Lemma is natural in both C and F.

10. Verify the claim made in 4.5.13 that if $c \in F(C)$ and $c' \in F(C')$ are both universal elements of the functor $F : \mathscr{C} \longrightarrow \mathbf{Set}$, then there is a unique isomorphism $f : C \longrightarrow C'$ such that $F(f)(c) = c'$.

4.6 Linear sketches (graphs with diagrams)

Specifications in mathematics and computer science are most commonly expressed using a formal language with rules spelling out the semantics. However, there are other objects in mathematics intended as specifications that are not based on a formal language. Many of these are tuple-based; for example the signature of an algebraic structure or the tuple specifying a finite state machine.

A sketch is another kind of formal abstract specification of a mathematical structure; it is based on a graph rather than on a formal language or tuple. The semantics is often a functor; in other contexts it is a structure generalizing Goguen's initial algebra semantics.

Each sketch generates a (categorical) theory; this theory is a category that in a strong sense contains all the syntax implied by the sketch.

4.6.1 Linear sketches We construct a hierarchy of types of sketches here and in Chapters 7, 8 and 10. Each new type uses additional categorical constructions to provide more expressive power than the preceding types.

What we can do now is describe a very simple type of structure using 'linear' sketches. It can describe multisorted algebraic structures with only

unary operations. If you are not familiar with multisorted algebraic structures, it will not matter.

4.6.2 Definition A **linear sketch** $\mathscr{S}$ is a pair $(\mathscr{G}, \mathscr{D})$ where $\mathscr{G}$ is a graph and $\mathscr{D}$ is a collection of diagrams in $\mathscr{G}$. Because of the motivating example of algebraic structures, the arrows of the graph of a sketch (not just a linear sketch) are often called **operations** of the sketch.

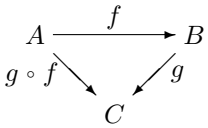
4.6.3 Definition A **model of a linear sketch** $\mathscr{S}$ in a category $\mathscr{C}$ is a model (graph homomorphism) $M : \mathscr{G} \to \mathscr{C}$ such that whenever $D : \mathscr{I} \to \mathscr{G}$ is a diagram in $\mathscr{D}$, then $M \circ D$ is a *commutative* diagram in $\mathscr{C}$. The diagrams represent the equations which have to be true in all models.

We write $M : \mathscr{S} \to \mathscr{C}$ for such a model. This use of the same symbol to denote both the sketch homomorphism and the graph homomorphism is a bit of notational overloading that in practice is always disambiguated by context. The collection of all models of $\mathscr{S}$ in $\mathscr{C}$ is denoted $\mathbf{Mod}(\mathscr{S}, \mathscr{C})$.

A model of a sketch $\mathscr{S}$ in **Set** is (among other things) a set indexed by the nodes of the graph of $\mathscr{S}$ as discussed in 2.6.11. If M is a model and c is a node of the graph, an element of $M(c)$ is an element indexed by c, or an element of type c.

4.6.4 Definition A **homomorphism of models** of a linear sketch $\mathscr{S}$, both models in the same category $\mathscr{C}$, is a natural transformation between the models. For given $\mathscr{S}$ and $\mathscr{C}$, the models therefore form a category with natural transformations as arrows; this category is a full subcategory of the category of all graph homomorphisms from $\mathscr{G}$ to $\mathscr{C}$ (which in general do not take the diagrams in $\mathscr{D}$ to commutative diagrams).

4.6.5 Example Any category $\mathscr{E}$ can be made into a linear sketch called the **underlying linear sketch** of $\mathscr{E}$, denoted $U(\mathscr{E})$ (concerning this notation, see 4.6.10 below), by taking for the diagrams the collection of all commutative diagrams in the category. A model of the underlying linear sketch $\mathscr{E}$ in some category $\mathscr{C}$ is the same as a functor on the original category: on the one hand, any functor takes any commutative diagram to a commutative diagram and so is a model. On the other hand a model $M : \mathscr{S} \to \mathscr{C}$ of the underlying linear sketch $\mathscr{S}$ of $\mathscr{E}$ preserves in particular all commutative diagrams of the form



(4.35)

and so preserves composition as well as every commutative diagram consisting of a single node and no arrows, and so preserves identities (see 4.1.6).

4.6.6 Example The linear sketch for u-structures has the graph with one node and one arrow as its graph, and no diagrams.

4.6.7 Example The construction in 4.2.7 gives the **sketch for graphs**. Its graph is

$$a \underset{t}{\overset{s}{\rightrightarrows}} n \tag{4.36}$$

and it has no diagrams.

4.6.8 Example We now consider an example of a linear sketch which has diagrams. Suppose we wanted to consider sets with permutations as structures. This would be a u-structure (S, u) with u a bijection. We can force u to go to a bijection in **Set**-models by requiring that it have an inverse. Thus the sketch $\mathscr{P}$ of sets with permutations has as graph the graph $\mathscr{G}$ with one node e and two arrows u and v, together with this diagram D:

$$e \underset{v}{\overset{u}{\rightleftarrows}} e \tag{4.37}$$

based on the shape graph

$$i \underset{y}{\overset{x}{\rightleftarrows}} j \tag{4.38}$$

A model M of this sketch in **Set** must have $M(e)$ a set, $M(u)$ and $M(v)$ functions from $M(e)$ to itself (since $D(i) = D(j) = e$), and because Diagram (4.37) must go to a commutative diagram, it must have

$$M(u) \circ M(v) = M(v) \circ M(u) = \mathrm{id}_{M(e)}$$

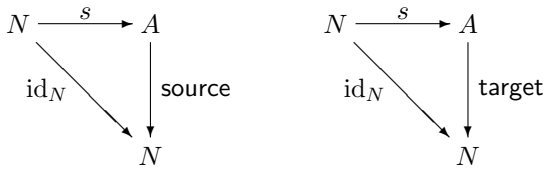
This says that $M(u)$ and M(v) are inverses to each other, so that they are permutations. Note that a model in any category is an object of that category together with an isomorphism of the object with itself and the inverse of that isomorphism.

If we had used as the only diagram the diagram with *one* node e and both arrows u and v, the result would have been a sketch in which any model M had the property that $M(u)$ and $M(v)$ are the identity.

4.6.9 Example Suppose we wanted to have a linear sketch for graphs which have at least one loop at every node. We could try the following construction, which contains a mild surprise. The sketch has the graph (4.19), page 102 as its graph, with arrows $s : N \longrightarrow A$ and $\mathrm{id}_N : N \longrightarrow N$ added. The diagrams are

(4.39)

and



(4.40)

In a model M in sets, if n is a node, that is, $n \in M(N)$, then $M(s)(n)$ is an arrow. The commutativity of Diagram (4.39) forces $M(\mathrm{id}_N)$ to be $\mathrm{id}_{M(N)}$ (see 4.1.6), and the commutativity of the diagrams (4.40) forces the source and target of $M(s)(n)$ to be n, so that $M(s)(n)$ is a loop on n.

The surprise is that a homomorphism $\alpha : M \longrightarrow M'$ of models of this sketch must take the particular loop $M(s)(n)$ to $M'(s)(\alpha N(n))$. Of course, any homomorphism of graphs will take the loop $M(s)(n)$ to *some* loop on $\alpha N(n)$, but our homomorphisms are stricter than that. So what we really have are graphs with a *distinguished* loop at every node and homomorphisms which take distinguished loops to distinguished loops. These are called **reflexive** graphs. A node in a reflexive graph may have other loops but they are not part of the given structure.

If you want a sketch for graphs which have a loop on every node, but not a distinguished loop (so that a homomorphism takes the loop on n to some loop on $\alpha N(n)$, but it does not matter which one), you will have to wait until we can study regular sketches in Section 10.4.

4.6.10 Homomorphisms of linear sketches A **homomorphism of linear sketches** from $\mathscr{S} = (\mathscr{G}, \mathscr{D})$ to $\mathscr{S}' = (\mathscr{G}', \mathscr{D}')$ is a graph homomorphism $\phi : \mathscr{G} \longrightarrow \mathscr{G}'$ with the property that if $D : I \longrightarrow \mathscr{G}$ is a diagram in $\mathscr{D}$, then $\phi \circ D : I \longrightarrow \mathscr{G}'$ is a diagram in $\mathscr{D}'$. It is easy to check that this definition makes linear sketches and their homomorphisms into a category.

Note that here we are defining homomorphisms between possibly different *sketches*, whereas in Definition 4.6.4 we defined homomorphisms between two *models* of a sketch.

If $F : \mathscr{C} \longrightarrow \mathscr{C}'$ is a functor between categories, the underlying linear sketch homomorphism $U(F) : U(\mathscr{C}) \longrightarrow U(\mathscr{C}')$ is F regarded as a homo-

morphism of graphs. Since it takes any commutative diagram in $\mathscr{C}$ to a commutative diagram in $\mathscr{C}'$, it is a homomorphism of linear sketches.

We have already used the symbol $U(\mathscr{E})$ to denote the underlying graph of a category $\mathscr{E}$ in 3.1.10. Here, we use it to denote the underlying linear sketch. In this text, we disambiguate such notation by using phrases such as 'the underlying graph $U(\mathscr{E})$'. In a situation where one needed frequently to refer to several different underlying functors from the same category, one could introduce heavy notation such as $U^{\mathbf{Cat}}_{\mathbf{Grf}}$.

4.6.11 The theory of a linear sketch You can reverse 4.6.5: given a linear sketch $\mathscr{S}$, there is a category $\mathbf{Th}(\mathscr{S})$, the **theory of** $\mathscr{S}$, which has a **universal model** M_0 of $\mathscr{S}$ in this sense: $M_0 : \mathscr{S} \to U(\mathbf{Th}(\mathscr{S}))$ is a model, and for any other model $M : \mathscr{S} \to U(\mathscr{C})$, there is a unique functor $F : \mathbf{Th}(\mathscr{S}) \to \mathscr{C}$ such that $UF \circ M_0 = M$. Since UF is a model of the underlying sketch of the category $\mathbf{Th}(\mathscr{S})$ (see 4.6.5), M_0 induces a bijection between models of $\mathscr{S}$ and models of its theory (functors from $\mathbf{Th}(\mathscr{S})$ to $\mathbf{Set}$). This bijection is in fact part of an equivalence of categories between $\mathbf{Mod}(\mathscr{S}, \mathbf{Set})$ and the functor category $\mathbf{Func}(\mathbf{Th}(\mathscr{S}), \mathbf{Set})$.

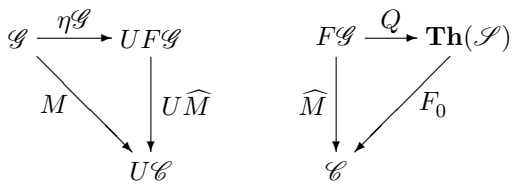
$\mathbf{Th}(\mathscr{S})$ satisfies the following requirements:

LT–1 Every arrow of $\mathbf{Th}(\mathscr{S})$ is a composite of arrows of the form $M_0(a)$ for arrows a of $\mathscr{S}$.

LT–2 M_0 takes every diagram of $\mathscr{S}$ to a commutative diagram in $\mathbf{Th}(\mathscr{S})$.

4.6.12 Construction of the theory of a linear sketch Let $\mathscr{S}$ be a linear sketch. The idea behind the construction of $\mathbf{Th}(\mathscr{S})$ is: 'Freely compose the arrows of $\mathscr{G}$ and impose the diagrams as equations.' Formally, begin with the free category $F(\mathscr{G})$ generated by $\mathscr{G}$ and construct $\mathbf{Th}(\mathscr{S})$ and a functor $Q : F(\mathscr{G}) \to \mathbf{Th}(\mathscr{S})$ which make all the diagrams in $\mathscr{D}$ commute as described in 4.1.13. The universal model $M_0 : \mathscr{G} \to U(\mathbf{Th}(\mathscr{S}))$ is $UQ \circ \eta\mathscr{G}$ (see 3.1.14); it takes each node to itself and each arrow to its congruence class in $\mathbf{Th}(\mathscr{S})$.

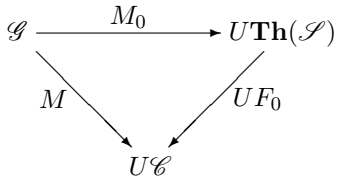
That M_0 has the property claimed in 4.6.11 follows by considering this diagram for a given model $M : \mathscr{G} \to U\mathscr{C}$:



(4.41)

For a given M, it follows by Proposition 3.1.15 that there is a unique functor $\widehat{M}$ for which the left triangle commutes. Then by Proposition 3.5.4 (see

also 4.1.13) there is a unique functor $F_0 : \mathbf{Th}(\mathscr{S}) \to \mathscr{C}$ making the right hand triangle commute. Apply U to the right hand triangle, put them together and let $M_0 = UQ \circ \eta\mathscr{G}$. Then we have a commutative triangle:



(4.42)

This shows the existence and we leave the uniqueness to the reader.

The construction of a semantics functor for a functional programming language illustrated in 3.5.9 is a special case of the construction just given.

4.6.13 Examples The theory which is generated by the underlying linear sketch of a category is isomorphic to the category itself (Exercise 3). The sketch for u-structures (see 4.2.5 and 4.6.6) generates a category with one node u_0, its identity arrow, and arrows e, $e \circ e$, $e \circ e \circ e$, and so on, all different because there are no equations to make them the same. In other words, it has one arrow e^n for each natural number n.

The theory for the sketch for graphs (its graph is 1.3.9 and it has no diagrams) has only two arrows besides those in the graph 1.3.9, namely the identity arrows in a and n. That is because the two arrows of the graph do not compose with each other.

The sketch for permutations in 4.6.8 is more complicated. It has one node e and arrows u^n for all positive and *negative* integers n. This is essentially because v must be u^{-1} in the theory; see 4.7.13 for more details.

The reader may wonder why the term 'linear sketch' is appropriate. One way of thinking of linear sketches is that they are exactly the sketches for which the following are true:

(i) If M_1 and M_2 are models in the category of sets, then there is a model $M_1 + M_2$ defined by setting $(M_1 + M_2)(a) = M_1(a) + M_2(a)$, $a \in G_0(\mathscr{S})$. ($S + T$ is the disjoint union of sets S and T.)

(ii) If M is a model and X is a set, there is a model $X \times M$ defined by setting $(X \times M)(a) = X \times M(a)$, $a \in G_0$.

The linear sketches with constants that we will consider in Section 4.7 below lack these 'linearity' properties. They should perhaps be called affine sketches. For the reader familiar with equational theories we observe that linear sketches have only unary operations while linear sketches with constants have, in addition, nullary operations.

4.6.14 Exercises

1. Find a linear sketch $\mathscr{S}$ for which $\mathbf{Mod}(\mathscr{S}, \mathbf{Set})$ is isomorphic to **Set**.

2. Describe a linear sketch whose models in **Set** are sets S with two functions from S to S which commute with each other on composition.

3. Prove that if a category $\mathscr{C}$ is regarded as a linear sketch as in 4.6.5, then $\mathbf{Th}(\mathscr{C})$ is isomorphic to $\mathscr{C}$.

4. Let M and N be models of the sketch in 4.6.8. Show that $f : M(e) \longrightarrow N(e)$ is (the only component of) a homomorphism of models if and only if $f \circ M(u) = N(u) \circ f$.

5. Describe the theory of the linear sketch whose graph is

$$0 \underset{v}{\overset{u}{\rightleftarrows}} 1 \tag{4.43}$$

with diagram

$$\begin{array}{ccc} 0 & \xrightarrow{\;u\;} & 1 \\ {\scriptstyle u}\downarrow & & \downarrow{\scriptstyle v} \\ 1 & \xleftarrow[\;u\;]{} & 0 \end{array} \tag{4.44}$$

4.7 Linear sketches with constants: initial term models

In this section, we describe a semantics for linear sketches which is essentially a special case of Goguen's initial algebra semantics. We will treat a version equivalent to the general case in Section 7.6. We construct a specific model of the sketch in **Set** by using the ingredients of the sketch, in a similar spirit to mathematical logic wherein models of a theory are constructed using the expressions in the language.

4.7.1 Definition A model of a sketch in a category $\mathscr{C}$ is called **initial** if it has exactly one arrow (natural transformation) to every model in $\mathscr{C}$.

Thus an initial model is an initial object in the category of models, as the name suggests. Any two initial objects of any category are isomorphic by a unique isomorphism, so initial models in a given category are unique up to isomorphism.

In the case of a linear sketch, to describe the initial model in the category of sets is easy; it is the model $M : \mathscr{S} \longrightarrow \mathbf{Set}$ for which $M(a) = \emptyset$ for every

node a of $\mathscr{S}$. When we discuss further sketches we will see that the initial models will be more interesting. At this point, we have to complicate things a bit to get interesting models.

4.7.2 Definition By a **linear sketch with constants** we mean a triple $\mathscr{S} = (\mathscr{G}, \mathscr{D}, C)$ where $(\mathscr{G}, \mathscr{D})$ is a linear sketch and C is a collection of constants indexed (as in 2.6.11) by the set of nodes of $\mathscr{G}$. We let type : $C \longrightarrow G_0$ be the indexing function, where G_0 denotes the set of nodes of $\mathscr{G}$. We will not formalize this further because we will have a more systematic way of doing this in Chapter 7. In particular, we could discuss models on categories other than that of sets, but there is no purpose in doing so.

The sketch $(\mathscr{G}, \mathscr{D})$ is called the **underlying linear sketch** of the linear sketch with constants.

4.7.3 Definition A **model** of a linear sketch with constants in the category of sets is a model $M : (\mathscr{G}, \mathscr{D}) \longrightarrow \mathbf{Set}$ together with a function $M : C \longrightarrow \bigcup_{a \in G_0} M(a)$ such that for $x \in C$, $M(x) \in M(\text{type}(x))$.

(Note that we have overloaded M, and that M need not be injective.) Thus a model of a linear sketch with constants is a model of the underlying linear sketch together with elements chosen in the models of various types.

The existence of the constants means that the values need no longer be empty. In particular, the initial models may be interesting.

4.7.4 Example We give a somewhat arbitrary example which illustrates what happens. Let us add three constants to the sketch for graphs, f of type a and x and y of type n. A set model of this sketch with constants will be a model M in **Set** of the sketch for graphs – in other words, a graph – with a distinguished arrow $M(f)$ and two distinguished nodes $M(x)$ and $M(y)$. It may happen that $M(x) = M(y)$ and there is no requirement that $M(x)$ or $M(y)$ be the source or target of $M(f)$.

4.7.5 Definition A **homomorphism of models** of a linear sketch with constants is a natural transformation between two models of the sketch that takes the values of the constants in the first model to the values in the second.

In other words, if M_1 and M_2 are models and $\phi : M_1 \longrightarrow M_2$ is a homomorphism of models of the underlying linear sketch, then to be a homomorphism of models of the linear sketch with constants it must also satisfy the requirement that for any $x \in C$ of type a, $\phi a(M_1(x)) = M_2(x)$. Thus in 4.7.4, a homomorphism $\alpha : M_1 \longrightarrow M_2$ of models of that sketch with constants has to have $\alpha A(M_1(f)) = M_2(f)$, $\alpha N(M_1(x)) = M_2(x)$, and $\alpha N(M_1(y)) = M_2(y)$.

4.7.6 Term models A model M of a sketch with constants is called a **term model** if for every node a of the underlying graph, every element of $M(a)$ is reachable by beginning with constants and applying various operations (arrows of the sketch). The constants you begin with do not have to be of type a, but the final operation will, of course, have to be one that produces an element of type a. The significance of this condition from the computational point of view is that elements that cannot be produced in this way might as well not be there.

4.7.7 Example Let us consider the linear sketch of u-structures with one constant which we will call zero. Let the graph have node n and call the only arrow succ. There are no diagrams. As this nomenclature suggests, one model of this sketch in **Set** is the natural numbers with the successor operation; the constant is 0. Other models are the integers and the integers modulo a fixed number k (in both cases, take the successor of x to be $x + 1$). However, the natural numbers are the unique (up to unique isomorphism) *initial* model.

To see this, suppose M is any other model. Let us use the same letter M to denote $M(n)$ since there are no other nodes (common practice when the sketch has only one node). Also, let $t : M \to M$ denote the value of M at succ and m_0 the value at zero. We let $\mathbf{N}$, succ and 0 denote the values of these things in the natural numbers. To show that $\mathbf{N}$ is the initial model we must define a natural transformation $f : \mathbf{N} \to M$ and show that it is the only one.

Define f as follows: let $f(0) = m_0$, as required if f is to be an arrow between linear sketches with constants. Then since f must commute with succ, we must have that $f(1) = t(m_0)$, $f(2) = t(t(m_0))$, and so on. This defines f inductively on the whole of $\mathbf{N}$. It is clearly unique and immediate to see that it is an arrow between models.

In Section 5.5, we base the definition of natural numbers object in an arbitrary category on this sketch.

4.7.8 Initiality and induction Since $\mathbf{N}$ is the initial model for the linear sketch of u-structures with one constant, it follows that, *as a u-structure*, $\mathbf{N}$ has no proper substructures (Proposition 2.8.7). (Of course it does have proper substructures for example as a semigroup on addition.) This can be reworded as follows: If S is a subset of $\mathbf{N}$ with the property that $0 \in S$ and for any $x \in S$ also $\mathrm{succ}(x) \in S$, then $S = \mathbf{N}$. This is the principle of mathematical induction.

In general, an initial model of any sketch has no proper subobjects and so produces a principle of structural induction appropriate to that type of structure. (See 8.1.8.)

4.7.9 Example The set of all integers is a model but not a term model of the linear sketch of u-structures with one constant. For imagine you have a computer that can store integers, but the only operation that can be carried out on them is that of increment (successor). Suppose, further, that the only natural number whose existence you are certain of is 0. Then you can certainly produce, in addition, 1, 2, ..., but no negative numbers. Therefore, they may as well not be there. You can get them by, for example, adding a decrement operation, but as it stands they are inaccessible. They are what J. Goguen and J. Meseguer have called 'junk'.

4.7.10 Example The set $\mathbf{Z}_k$ of natural numbers (mod k) is a term model of the sketch of 4.7.7, but not an initial model. For example, there is no arrow from the natural numbers (mod k) to the natural numbers. In the first, the successor of $k-1$ is 0, while in the second it is nonzero. Thus no arrow could preserve successor at that point. What has happened here is that the model satisfies an additional equation $k = 0$ not required by the diagrams. This is an example of what Goguen and Meseguer call 'confusion'.

4.7.11 Construction of initial term models Linear sketches with constants always have initial models. When the sketch is finite, an initial model can always be constructed recursively as a term model. ('Finite' means finite number of nodes and arrows.) We now give this construction.

Let $\mathscr{S} = (\mathscr{G}, \mathscr{D}, C)$ be a linear sketch with constants. We define a model $I : \mathscr{S} \longrightarrow \mathbf{Set}$ recursively as the model constructed by the following requirements I–1 through I–3. (I–1 through I–3 can be see to define an operator, with the model I as a fixed point of the operator.)

The elements of $I(a)$ for a node a of $\mathscr{G}$ are congruence classes of **terms** of $\mathscr{G}$ (composable strings of arrows, including constants, of $\mathscr{G}$); $[x]$ denotes the congruence class of a term x by the congruence relation generated by the relation $\sim$ constructed recursively in the model. By 'congruence relation', we mean congruence on the free category generated by $\mathscr{G}$ as defined in 3.5.1. In particular, if (g, f) and (g', f') are both composable pairs and $[f] = [f']$ and $[g] = [g']$, then $[g \circ f] = [g' \circ f']$.

I–1 If a is a node of $\mathscr{G}$ and x is a constant of type a, then $[x] \in I(a)$.

I–2 If $f : a \longrightarrow b$ is an arrow of $\mathscr{G}$ and $[x]$ is an element of $I(a)$, then $[fx] \in I(b)$ and $I(f)[x] = [fx]$. (Note that this constructs both an element and a value of the function $I(f)$ simultaneously.)

I–3 If $(f_1, \ldots, f_m)$ and $(g_1, \ldots, g_k)$ are paths in a diagram in $\mathscr{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \ldots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \ldots \circ Ig_k)[x]$$

in $I(b)$.

4.7.12 By 'the model constructed by' these requirements, we mean that

(i) no element is in $I(a)$ except congruence classes of the terms constructed in I–1 and I–2, and

(ii) two terms are equivalent if and only if they are forced to be equivalent by the congruence relation generated by I–3.

Requirement (i) means that the models have no elements not nameable in the theory ('no junk') and (ii) means that elements not provably the same are different ('no confusion'). Concerning (ii), see Exercise 2. It follows from requirement (ii) that if $[x] = [y]$ in $I(a)$ and $f : a \longrightarrow b$ is an arrow of $\mathscr{G}$, then $[fx] = [fy]$ in $I(b)$.

Note that the models in 4.7.10 have terms giving the same element which are not forced to be equivalent by I–3.

4.7.13 Example Let us work out the initial term model of the sketch from 4.6.8 with one constant called x added. Since the sketch has only one node, the model has only one type. Thus in this case, there is only one set, call it S, and the arrows of the sketch lead to functions from S to S.

Then S has elements in accordance with the following rules:

Mod–1 There is an element $[x] \in S$.

Mod–2 If $[y] \in S$, then there are elements $[uy], [vy] \in S$.

Mod–3 If $[y] = [z]$, then $[uy] = [uz]$ and $[vy] = [vz]$.

Mod–4 For any $[y] \in S$, $[uvy] = [vuy] = [y]$.

It is clear that the set of all 'words' $[w_1 w_2 \dots w_k x]$, where each w_i is either u or v, satisfies the first two rules above. In order to satisfy all four, we have to impose the equalities they force. In order to gain some insight into this, let us calculate some of the elements of S.

We observe that there must be elements

$$[x_0] = [x],\ [x_1] = [ux],\ [x_2] = [uux],\ \dots\ ,\ [x_n] = [\underbrace{uu\cdots u}_{n \text{ copies}} x]$$

as well as elements we will denote

$$[x_{-1}] = [vx],\ [x_{-2}] = [vvx],\ \cdots\ ,\ [x_{-n}] = [\underbrace{vv\cdots v}_{n \text{ copies}} x]$$

We first explain why these elements exhaust S. We will not give a formal proof, but let us see which element is represented by an element chosen more or less at random, $[y] = [uvuvuuvux]$. Since $[vux] = [x]$, we have that $[y] = [uvuvuux] = [uvuvx_2]$. Since $[uvx_2] = [x_2]$, it follows that $[y] = [uvx_2]$ and then $[y] = [x_2]$, by another application of the same identity.

This kind of reasoning can be used to show that any application of u's and v's to $[x]$ gives the element $[x_k]$ where k is the number of u's less the number of v's.

In particular, $[ux_k] = [x_{k+1}]$ and $[vx_k] = [x_{k-1}]$ so that the set $\{[x_k] \mid -\infty < k < \infty\}$ is carried into itself by both u and v. It contains $[x] = [x_0]$ and so must be all of S.

There remains the question of all the $[x_k]$ being distinct; that is whether or not there are any identities among the $[x_k]$. There is a standard way of resolving this question: if there is an equation among two combinations of arrows from the sketch, that equation must hold in every model. Thus if the equation fails in any one model, it cannot be a consequence of the identities in the sketch. In this case, there is an easy model, namely the set $\mathbf{Z}$ of all integers. In the set $\mathbf{Z}$, we let u act by addition of the number 1 and v act by subtracting 1. Then any combination of actions by u and v is just addition of k, the difference between the number of u's and v's (which may be negative).

The discussion above suggests how to construct a bijection between S and $\mathbf{Z}$ which is an isomorphism of models. We must choose an element to correspond to $[x]$. A plausible, but by no means necessary, choice is to correspond $[x]$ to 0. If we do that then we must correspond $[x_1] = [ux]$ to $[u0] = 1$, $[x_2] = [uux]$ to $[uu0] = 2$ and so on to correspond $[x_k]$ to k, for $k > 0$. For $k < 0$, the argument is similar, replacing u by v, to show that we correspond $[x_k]$ to k in that case as well.

The isomorphism just constructed takes each $[x_k]$ to the integer k, which implies that if $k \neq k'$, then $[x_k] \neq [x_{k'}]$. Thus S consists of precisely the distinct classes $[x_k]$, one for each integer $k \in \mathbf{Z}$.

4.7.14 Example The initial term model for the sketch for graphs with two constants x and y of type n and one constant f of type a given in Example 4.7.4 can be constructed using the method of the preceding example (but it is much easier). The result is a model I with

$$I(n) = \{x, y, \mathsf{source}(f), \mathsf{target}(f)\}$$

and $I(a) = \{f\}$. It is a graph with four nodes and an arrow between two of them.

4.7.15 Given the construction in 4.7.11 and any **Set** model M of the same sketch, the unique homomorphism $\alpha : I \longrightarrow M$ is constructed inductively as follows:

M–1 If x is a constant of type a, then $\alpha a[x] = M(x)$.

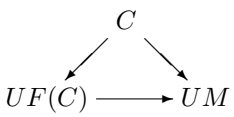
M–2 If $f : a \to b$ in $\mathscr{G}$ and $[x] \in I(a)$, then $\alpha b([fx]) = M(f)([M(x)])$.

It is a straightforward exercise to show that this is well defined and is a homomorphism of models. It is clearly the only possible one.

The construction in 4.7.11 can be seen as the least fixed point of an operator on models of the sketch (without the constants) in the category of sets and *partial* functions. To any such model M, the operator adjoins an element $f(x)$ to $M(b)$ for any arrow $f : a \to b$ and any element $x \in M(a)$ for which $M(f)(x)$ is not defined. It forces $f(x)$ to be the same as some other element of $M(b)$ if the diagrams force that to happen (we leave the formal description of this to you). To get the model for a particular set of constants, you start with the model obtained by applying only I–1 (so that the sorts have only constants in them and all the arrows have empty functions as models). The least fixed point of this operator is the model in 4.7.11, up to isomorphism.

4.7.16 Properties of model categories We have seen that the category of models of a linear sketch with constants always has an initial object. The fact that the category of nonempty semigroups and semigroups homomorphisms does not have an initial object (see 2.7.17) thus means that that category is not the category of models of a linear sketch with constants. This is our first example of a theorem in model theory of a very typical sort, saying that the category of models of a certain kind of sketch or theory has to have certain properties, so that if a category $\mathscr{C}$ does not have one of those properties, it is not the category of models of a sketch of that kind. Generally, the more expressive the sketch, the fewer restrictions are imposed on the possible categories of models. Such theorems are covered in detail in [Barr and Wells, 1985] and [Adámek and Rosický, 1994].

4.7.17 Free models Let $\mathscr{S}$ be a fixed linear sketch with just one node. With each set C we can associate a linear sketch with the set C of constants. Let us call it $\mathscr{S}(C)$. Let $F(C)$ denote an initial model of $\mathscr{S}(C)$. A model of $\mathscr{S}(C)$ is a model M of $\mathscr{S}$ together with a function $C \to M$. Here, as above, we will use the name of the model to denote the value at the single node of the sketch. To say that $F(C)$ is an initial model of $\mathscr{S}(C)$ is to say that given any model M of $\mathscr{S}$ together with a function $C \to M$, there is a unique arrow $F(C) \to M$ in the category of models for which



commutes. This property is summarized by saying that $F(C)$ is the **free model** of $\mathscr{S}$ generated by C. Note the similarity with Proposition 3.1.15. Freeness is given a unified treatment in Definition 13.2.1.

4.7.18 This notion of free models can be generalized to the case of many nodes. We indicate briefly how this can be done. Let $\mathscr{S}$ be a linear sketch and G_0 be the nodes of its graph. By a **G_0-indexed set**, we mean a set C together with a function $C \longrightarrow G_0$ (see 2.6.11). Given any such set we can form a sketch $\mathscr{S}(C)$ which is the linear sketch with the set C of constants with the given function as type function. An initial model of this sketch is called the free model generated by the G_0-indexed set C.

Example 4.7.7 can now be seen as describing the free u-structure with one constant. Another example is the free graph on two nodes and one arrow. As we saw in Example 4.7.14, it has *four* nodes, and one arrow connecting two of them.

We describe sketches with more expressive power in Chapters 7, 8 and 10.

4.7.19 Exercises

1. Find the initial term model of the linear sketch of Exercise 5 of Section 4.6 with two constants added, x of type 0 and y of type 1.

2. Show that requirement (ii) of 4.7.12 is equivalent to the following statement: two terms in the model I are equivalent if and only if every model of $\mathscr{S}$ takes them to the same function.

3. Show that if $\mathscr{S}$ is any linear sketch with constants, then $\mathbf{Mod}(\mathscr{S}, \mathbf{Set})$ has a terminal object.

4.8 2-categories

The category **Cat** of small categories and functors is a category, but it has more structure than a category since between two functors with the same domain and the same codomain there are natural transformations. A 2-category $\mathbf{C}$ can be thought of as a category $\mathbf{C}^b$ (its base category) with in addition 'maps from arrows to arrows' called 2-cells that have many of the properties of natural transformations, for example the interchange law.

We give a definition here (4.8.1, 4.8.3, 4.8.7) that in effect defines a 2-category without assuming the existence of the base category, which is then discovered as part of the structure; then we give a second more conceptual definition (4.8.10) that expresses a 2-category as a kind of 'enriched' category. These definitions follow [Power and Wells, 1992].

The most accessible introduction to 2-categories is that of [Kelly and Street, 1973]; much more is in [Gray, 1974] and [Kelly, 1982a]. [Baez, 1997] is an exposition with many references to the literature of current work on generalizations to n-categories and even further. Applications to computer science are discussed in [Seely, 1987b], [Gray, 1988], [Ji-Feng and Hoare,

1990], [Power, 1990b], [Power and Wells, 1992] and [Corradini and Gadducci, 1997].

4.8.1 Definition of 2-category, part I A 2-category $\mathbf{C}$ consists of three sets $\mathbf{C}_0$, $\mathbf{C}_1$ and $\mathbf{C}_2$ subject to certain requirements listed in TC–1 through TC–6 below. We first establish some notation.

(a) The elements of $\mathbf{C}_i$ for $i = 0, 1, 2$ are called **i-cells**.

(b) Elements of $\mathbf{C}_0$ are denoted by capital script letters and may also be called **objects**.

(c) Elements of $\mathbf{C}_1$ are denoted by capital Roman letters and may also be called **arrows**.

(d) Elements of $\mathbf{C}_2$ are denoted by lowercase Greek letters.

This definition is continued in 4.8.3 below.

4.8.2 Example The category **Cat** has a 2-category structure as follows: the 0-cells are the small categories, the 1-cells are the functors, and the 2-cells are the natural transformations. It will greatly aid your understanding of 2-categories if you will read the following definitions with **Cat** in mind.

4.8.3 Definition of 2-category, part II A 2-category $\mathbf{C}$ has three category structures on it, called the **base category**, the **horizontal category** and the **vertical category**. The structure is defined in terms of the last two and the base category is then derived from them.

We continue the definition of 2-categories in 4.8.7 below.

4.8.4 Example In **Cat**, the base category has small categories as objects and functors as arrows. The vertical category has functors as objects and natural transformations as arrows (although, of course, there are no arrows between two functors unless those functors have the same domain and the same codomain). The horizontal category has categories as objects and natural transformations as morphisms. In the horizontal category, the domain of a natural transformation is the domain of its domain functor (or of its codomain functor, for that matter; they have to be the same) and the codomain is the codomain of its domain (or codomain) functor.

We will amplify these remarks in 4.8.8.

4.8.5 Notation We will systematically use the words 'base', 'horizontal' and 'vertical' before the words 'composition', 'domain', 'codomain', and 'identity' to indicate the category structure being considered. We will also use superscript b, h and v for these purposes, except for composition. For

example, $\mathrm{dom}^v\,\alpha$ is the vertical domain of α and $\mathrm{id}^h\mathscr{A}$ is the horizontal identity of $\mathscr{A}$. If $\mathrm{dom}^v\,\alpha = F$, $\mathrm{cod}^v\,\alpha = G$, $\mathrm{dom}^h\,\alpha = \mathscr{A}$ and $\mathrm{cod}^h\,\alpha = \mathscr{B}$, this is summed up by writing $\alpha : F \longrightarrow G : \mathscr{A} \longrightarrow \mathscr{B}$.

The base and horizontal composition are denoted $*$ (it turns out that this overloading cannot cause confusion) and the vertical composition is denoted $\circ$.

4.8.6 The base category We give the definition of the base category here, even though we have not yet stated the axioms for a 2-category, because it is useful to illustrate the axioms with certain diagrams that assume knowledge of the base category.

The **base category** of a 2-category $\mathbf{C}$ has the 0-cells of $\mathbf{C}$ as objects and the 1-cells as arrows. Its structure is determined by the following rules.

BC–1 The domain and codomain are given by $\mathrm{dom}^b(F) = \mathrm{dom}^h(\mathrm{id}^v(F))$ and $\mathrm{cod}^b(F) = \mathrm{cod}^h(\mathrm{id}^v(F))$.

BC–2 The base identity $\mathrm{id}^b\mathscr{A}$ for a 0-cell $\mathscr{A}$ is $\mathrm{id}^b\mathscr{A} = \mathrm{dom}^v(\mathrm{id}^h\mathscr{A})$, which is the same as $\mathrm{cod}^v(\mathrm{id}^h\mathscr{A})$.

BC–3 The composition is denoted $*$ and is defined by

$$G * F = \mathrm{dom}^v(\mathrm{id}^v\,G * \mathrm{id}^v\,F)$$

We write $F : \mathscr{A} \longrightarrow \mathscr{B}$ if $\mathrm{dom}^b\,F = \mathscr{A}$ and $\mathrm{cod}^b\,F = \mathscr{B}$.

4.8.7 Definition of 2-category, part III The structure of a 2-category $\mathbf{C}$ is defined as follows:

TC–1 The horizontal category $\mathbf{C}^h$ has $\mathbf{C}_0$ as its set of objects and $\mathbf{C}_2$ as its set of arrows.

TC–2 The vertical category $\mathbf{C}^v$ has $\mathbf{C}_1$ as its set of objects and $\mathbf{C}_2$ as its set of arrows.

TC–3 A 2-cell α goes between arrows whose vertical identities have the same horizontal domains and codomains; precisely,

$$\mathrm{dom}^h\,\alpha = \mathrm{dom}^h\,\mathrm{id}^v(\mathrm{dom}^v\,\alpha) = \mathrm{dom}^h\,\mathrm{id}^v(\mathrm{cod}^v\,\alpha)$$

and

$$\mathrm{cod}^h\,\alpha = \mathrm{cod}^h\,\mathrm{id}^v(\mathrm{dom}^v\,\alpha) = \mathrm{cod}^h\,\mathrm{id}^v(\mathrm{cod}^v\,\alpha)$$

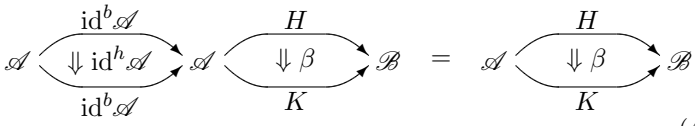
We can illustrate this rule by the following diagram.

$$\begin{array}{ccc} & F & \\ \mathscr{A} & \Downarrow \alpha & \mathscr{B} \\ & G & \end{array} \qquad (4.45)$$

TC–4 A 2-cell that is a horizontal identity must also be a vertical identity; precisely, for a 0-cell $\mathscr{A}$,

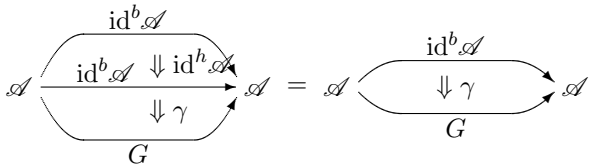
$$\mathrm{id}^h\mathscr{A} = \mathrm{id}^v(\mathrm{dom}^v(\mathrm{id}^h\mathscr{A}))$$

which is necessarily also $\mathrm{id}^v(\mathrm{cod}^v(\mathrm{id}^h\mathscr{A}))$. Thus for all 2-cells $\beta : H \to K : \mathscr{A} \to \mathscr{B}$,



(4.46)

Also for all $\gamma : \mathrm{id}^b\mathscr{A} \to G : \mathscr{A} \to \mathscr{A}$,



(4.47)

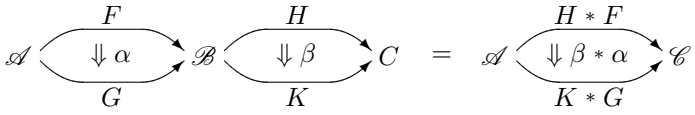
TC–5 For 2-cells α and β with $\mathrm{cod}^h\,\alpha = \mathrm{dom}^h\,\beta$ (so they are horizontally composable)

$$\mathrm{id}^v\,\mathrm{dom}^v(\beta * \alpha) = \mathrm{id}^v\,\mathrm{dom}^v\,\beta * \mathrm{id}^v\,\mathrm{dom}^v\,\alpha$$

and

$$\mathrm{id}^v\,\mathrm{cod}^v(\beta * \alpha) = \mathrm{id}^v\,\mathrm{cod}^v\,\beta * \mathrm{id}^v\,\mathrm{cod}^v\,\alpha$$

The composites on the right of each equation exist by TC–3. This is illustrated this way:

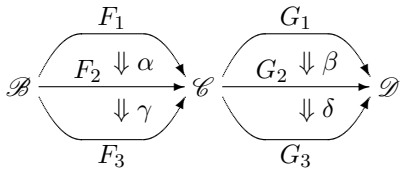


(4.48)

TC–6 (The interchange law) For 2-cells α, β, γ and δ for which the composites $\beta * \alpha$, $\delta * \gamma$, $\gamma \circ \alpha$ and $\delta \circ \beta$ are all defined,

$$(\delta * \gamma) \circ (\beta * \alpha) = (\delta \circ \beta) * (\gamma \circ \alpha)$$

The interchange law applies to a situation like this:



(4.49)

It is necessary to show that the composites $(\delta * \gamma) \circ (\beta * \alpha)$ and $(\delta \circ \beta) * (\gamma \circ \alpha)$ in the interchange law are defined. We verify the second and leave the first (which is a bit easier) as an exercise. Suppose the composites $\beta * \alpha$, $\delta * \gamma$, $\gamma \circ \alpha$ and $\delta \circ \beta$ are all defined. We must show that $\mathrm{cod}^h(\gamma \circ \alpha) = \mathrm{dom}^h(\delta \circ \beta)$. But

$$\begin{array}{rcll} \mathrm{cod}^h(\gamma \circ \alpha) & = & \mathrm{cod}^h \mathrm{id}^v \mathrm{dom}^v(\gamma \circ \alpha) & \text{TC–3} \\ & = & \mathrm{cod}^h \mathrm{id}^v \mathrm{dom}^v \gamma & \text{C–1 of 2.1.3} \\ & = & \mathrm{cod}^h \gamma & \text{TC–3} \\ & = & \mathrm{dom}^h \delta & \delta * \gamma \text{ is defined} \\ & = & \mathrm{dom}^h \mathrm{id}^v \mathrm{dom}^v \delta & \text{TC–3} \\ & = & \mathrm{dom}^h \mathrm{id}^v \mathrm{dom}^v(\delta \circ \beta) & \text{C–1 of 2.1.3} \\ & = & \mathrm{dom}^h(\delta \circ \beta) & \text{TC–3} \end{array} \tag{4.50}$$

as required.

Note that you can read the information about the way the domains and codomains match directly from Diagram (4.49): the diagram makes the fact just verified obvious. Nevertheless, the facts must be verified in the first place without the diagram – that is what makes it possible to draw the diagram as shown.

> We have chosen our notation so that the various entities in a 2-category look like those in **Cat**. This is quite different from most of the literature, where 0-cells are usually written with uppercase Roman letters and 1-cells with lowercase Roman letters. Also, in much of the literature, the composite of 2-cells α and γ is denoted by $\gamma \circ \alpha$ (presumably because their domains compose in the base category to give the domain of the composite) and the vertical composite of α and β by $\beta\alpha$. In 4.2.11, we denoted the vertical composite of natural transformations by $\beta \circ \alpha$ because when functors are thought of as models the vertical composite is the composite of the corresponding homomorphisms. We follow that usage here.

4.8.8 Example We now reexamine **Cat** as a 2-category in more detail, using the notation we have developed.

In the 2-category **Cat**, the base category is what we have called **Cat** all along. Recall that for any two small categories $\mathscr{A}$ and $\mathscr{B}$, $\mathbf{Func}(\mathscr{A}, \mathscr{B})$ is

a category with functors from $\mathscr{A}$ to $\mathscr{B}$ as objects and natural transformations between those functors as arrows. Then the vertical category for **Cat** is the disjoint union of all the categories $\mathbf{Func}(\mathscr{A}, \mathscr{B})$; vertical composition is the ordinary composition of natural transformations defined in 4.2.11. The horizontal category for **Cat** has categories as objects and natural transformations as arrows (the horizontal domain of a natural transformation $\alpha : F \longrightarrow G : \mathscr{A} \longrightarrow \mathscr{B}$ is $\mathscr{A}$, and the horizontal codomain is $\mathscr{B}$), with horizontal composition of natural transformations (4.4.4) as composition.

Now we can see in more detail how **Cat** satisfies the definition of a 2-category. We consider TC–4 through TC–6 in detail. The other parts are easier.

A horizontal identity in **Cat** is the identity natural transformation on the functor $\mathrm{id}_{\mathscr{A}}$ for some category $\mathscr{A}$. The component of this transformation at any object A of $\mathscr{A}$ is simply id_A, so it is immediate that the transformation acts as the identity for vertical composition (which is defined as componentwise composition), so that TC–4 is satisfied.

TC–5 follows from the definition of horizontal composition (4.4.4) of natural transformations, which requires that if $\alpha : F \longrightarrow G$ and $\beta : H \longrightarrow K$, then $\beta * \alpha : H \circ F \longrightarrow G \circ K$. Diagram (4.30) for the identity natural transformations on the vertical domains of these functors becomes

$$\begin{array}{ccc} (H \circ F)A & \xrightarrow{(H\,\mathrm{id}_F)A} & (H \circ F)A \\ {\scriptstyle (\mathrm{id}_H\,F)A}\downarrow & & \downarrow{\scriptstyle (\mathrm{id}_H\,F)A} \\ (H \circ F)A & \xrightarrow[(H\,\mathrm{id}_F)A]{} & (H \circ F)A \end{array} \tag{4.51}$$

This diagram commutes since all the arrows in the diagram are $\mathrm{id}_{(H\circ F)(A)}$. In particular, by Definitions 4.2.13 and 4.4.3,

$$(H\,\mathrm{id}_F)A = h(\mathrm{id}_F\,A) = H(\mathrm{id}_{F(A)}) = \mathrm{id}_{H(F(A))}$$

and by Definitions 4.2.13 and 4.4.2,

$$(\mathrm{id}_H\,F)A = (\mathrm{id}_H)(F(A)) = \mathrm{id}_{H(F(A))}$$

A similar argument works for the vertical codomain.

Finally, TC–6 is just Godement's rule G–5, the interchange law.

4.8.9 Conceptual definition of 2-category We now give a more conceptual definition of 2-category that shows that it is an example of an 'enriched' category. The primary source for enriched categories is [Kelly, 1982a].

4.8.10 Definition A 2-category $\mathbf{C}$ consists of the data given by CTC–1 to CTC–4 below, subject to the requirement that Diagrams (4.52) and (4.53) commute.

CTC–1 A collection $\mathbf{C}_0$ of **objects** or 0**-cells** of $\mathbf{C}$.

CTC–2 For every pair of 0-cells $\mathscr{A}$ and $\mathscr{B}$, a small category $\mathbf{C}(\mathscr{A},\mathscr{B})$. The objects of $\mathbf{C}(\mathscr{A},\mathscr{B})$ for all 0-cells A and B are called the 1**-cells** of $\mathbf{C}$, and the arrows of $\mathbf{C}(\mathscr{A},\mathscr{B})$ are called the 2**-cells** of $\mathbf{C}$.

CTC–3 For every triple of 0-cells $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$, a functor

$$\mathrm{comp} : \mathbf{C}(\mathscr{B},\mathscr{C}) \times \mathbf{C}(\mathscr{A},\mathscr{B}) \to \mathbf{C}(\mathscr{A},\mathscr{C})$$

called **composition**. (The domain of comp is the product of the categories $\mathbf{C}(\mathscr{B},\mathscr{C})$ and $\mathbf{C}(\mathscr{A},\mathscr{B})$ as defined in 2.6.6.)

CTC–4 For every object $\mathscr{A}$ of $\mathbf{C}$, a functor $\mathrm{unit} : \mathscr{T} \to \mathbf{C}(\mathscr{A},\mathscr{A})$, where $\mathscr{T}$ is the terminal object in **Cat**, the category with one object 0 and its identity arrow. The value unit(0) is denoted by $\mathrm{id}_{\mathscr{A}}$.

The following diagrams must commute.

$$\begin{array}{ccc}
\mathbf{C}(\mathscr{C},\mathscr{D})\times\mathbf{C}(\mathscr{B},\mathscr{C})\times\mathbf{C}(\mathscr{A},\mathscr{B}) & \xrightarrow{\mathrm{comp}\times\mathrm{id}} & \mathbf{C}(\mathscr{B},\mathscr{D})\times\mathbf{C}(\mathscr{A},\mathscr{B}) \\
\downarrow{\scriptstyle \mathrm{id}\times\mathrm{comp}} & & \downarrow{\scriptstyle \mathrm{comp}} \\
\mathbf{C}(\mathscr{C},\mathscr{D})\times\mathbf{C}(\mathscr{A},\mathscr{C}) & \xrightarrow[\mathrm{comp}]{} & \mathbf{C}(\mathscr{A},\mathscr{D})
\end{array} \tag{4.52}$$

$$\begin{array}{ccccc}
 & & \mathbf{C}(\mathscr{A},\mathscr{B}) & & \\
 & \swarrow^{\mathrm{id}\times\mathrm{unit}} & \downarrow{\scriptstyle \mathrm{id}} & \searrow^{\mathrm{unit}\times\mathrm{id}} & \\
\mathbf{C}(\mathscr{B},\mathscr{B})\times\mathbf{C}(\mathscr{A},\mathscr{B}) & \xrightarrow{\mathrm{comp}} & \mathbf{C}(\mathscr{A},\mathscr{B}) & \xleftarrow{\mathrm{comp}} & \mathbf{C}(\mathscr{A},\mathscr{B})\times\mathbf{C}(\mathscr{A},\mathscr{A})
\end{array} \tag{4.53}$$

4.8.11 Equivalence of the definitions To show that the definitions in 4.8.1 (as continued in 4.8.3 and 4.8.7) and 4.8.10 are equivalent is mostly straightforward. Given 4.8.10, the composition function $\mathrm{comp} : \mathbf{C}(\mathscr{B},\mathscr{C}) \times \mathbf{C}(\mathscr{A},\mathscr{B}) \to \mathbf{C}(\mathscr{A},\mathscr{C})$ produces the horizontal composite, and the vertical composite is the composition of the category $\mathbf{C}(\mathscr{A},\mathscr{B})$. The fact that unit is a functor means that it must take id_0 to the identity 2-cell on $\mathrm{id}_{\mathscr{A}}$ in $\mathbf{C}(\mathscr{A},\mathscr{A})$, which is a vertical identity, and that 2-cell is the horizontal identity because Diagram (4.53) commutes, so that TC–4 follows. The fact that comp is a

functor immediately implies TC–5 (because functors preserve identities) and the interchange law TC–6.

Given 4.8.1, 4.8.3 and 4.8.7, comp may be defined to take two 2-cells to their composite. The interchange law implies that comp preserves horizontal composition, and it preserves identities by TC–5. Define unit to take id_0 (the sole arrow of $\mathscr{T}$) to $\mathrm{id}^h\mathscr{A}$; then it preserves the identity because $\mathrm{id}^h\mathscr{A} = \mathrm{id}^v\,\mathrm{id}^b\mathscr{A}$, and it trivially preserves composition. Diagrams (4.52) and (4.53) then commute because the horizontal category is a category by definition, so that the composition is associative and has identities.

4.8.12 2-functors If $\mathbf{C}$ and $\mathbf{D}$ are 2-categories, a **2-functor** $F : \mathbf{C} \to \mathbf{D}$ consists of three maps $F_i : \mathscr{C}_i \to \mathscr{D}_i$ that are simultaneously functors from $\mathscr{C}^h \to \mathscr{D}^h$ and from $\mathscr{C}^v \to \mathscr{D}^v$. (It follows that they are also functors from $\mathscr{C}^b \to \mathscr{D}^b$.)

4.8.13 Example A type of category that has been considered in connection with program refinement is a **poset-enriched** category. This is a category $\mathscr{C}$ together with a partial ordering on every hom-set $\mathrm{Hom}_{\mathscr{C}}(A, B)$ with the property that for every triple A, B and C of objects, the composition function $\mathrm{comp} : \mathrm{Hom}(A, B) \times \mathrm{Hom}(B, C) \to \mathrm{Hom}(A, C)$ is monotone (preserves order). In other words, if $f \leq f' : A \to B$ and $g \leq g' : B \to C$, then $g \circ f \leq g' \circ f' : A \to C$.

A poset-enriched category $\mathscr{C}$ can be given a 2-category structure by requiring that there be exactly one 2-cell from f to g (where $f, g : A \to B$) if and only if $f \leq g$, and otherwise no 2-cells from f to g. Thus for each pair of objects A and B we construct the category $C(\mathrm{Hom}(A, B))$ corresponding to the poset $\mathrm{Hom}(A, B)$ as in 2.3.1. It is straightforward to verify that this structure satisfies the conceptual definition of 2-category (4.8.10).

If $\mathscr{C}$ and $\mathscr{D}$ are poset-enriched, it is a sufficient condition for a functor $F : \mathscr{C} \to \mathscr{D}$ to be a 2-functor that, for $f, g : A \to B$, $f \leq g \Rightarrow Ff \leq Fg$.

Three categories that we have already discussed have a natural poset-enriched structure (Exercises 2, 3 and 4).

An **order-enriched category** is usually defined to be a poset-enriched category in which each poset $\mathrm{Hom}(A, B)$ is a strict ω-CPO (see 2.4.4) in which every pair of elements has a greatest lower bound. Applications of this concept may be found in [Ji-Feng and Hoare, 1990], [Edalat and Smyth, 1993] and [Martin, Hoare and He, 1991].

4.8.14 Rewrite systems Let A be a finite alphabet. We will denote letters in A by a, b, $c \ldots$ and strings in the Kleene closure A^* by u, v, $w \ldots$. A **rewrite system** G consists of a finite set of **productions**, which are symbols of the form $y \to z$ where $y, z \in A^*$.

Given a rewrite system G, a string x can be **directly derived** from w, written $w \Rightarrow x$, if there are (possibly empty) strings u and v in A^* and a production $y \longrightarrow z$ such that $w = uyv$ and $x = uzv$. For example, if G has a production $ab \longrightarrow baa$, then $baba \Rightarrow bbaaa$, $abb \Rightarrow baab$ and $ab \Rightarrow baa$.

The operation $\overset{*}{\Rightarrow}$ ('derives to') is defined to be the reflexive transitive closure of $\Rightarrow$: for any string w, $w \overset{*}{\Rightarrow} w$, and $w \overset{*}{\Rightarrow} x$ if and only if there is a string w' such that $w \Rightarrow w'$ and $w' \overset{*}{\Rightarrow} x$.

A **context free rewrite system** is a rewrite system with the property that in every production $w \longrightarrow x$, w is a string of length 1 (essentially a single letter). A **context free grammar** G is a context free rewrite system with additional structure. First, the alphabet A is the union of disjoint sets T, the set of **terminals**, and N, the set of **nonterminals**. The terminals are traditionally (in academic treatments of theoretical computer science such as [Lewis and Papadimitriou, 1981]) denoted by lowercase letters and the nonterminals by uppercase letters. There is a distinguished nonterminal S called the **start symbol**, and every production has the form $V \longrightarrow w$ where V is a single nonterminal and w is any string in A^* (terminals, nonterminals, or both). The **language generated by** G is the set $\{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$.

An example is the grammar G with one terminal a and one nonterminal S, the start symbol, with productions

$$\begin{aligned} S &\longrightarrow SS \\ S &\longrightarrow aS \\ S &\longrightarrow a \end{aligned} \tag{4.54}$$

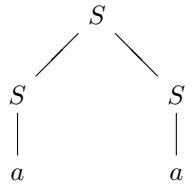
It turns out that context free rewrite systems are easily seen to have the structure of a 2-category. The additional structure that makes a context free grammar plays no role in this; we mentioned them because they are the most familiar example of context free rewrite systems.

To see how this works, consider an example of a derivation. One can derive aa in the grammar (4.54) by (at least) three different routes:

$$\begin{aligned} &S \Rightarrow SS \Rightarrow aS \Rightarrow aa \\ &S \Rightarrow SS \Rightarrow Sa \Rightarrow aa \\ &S \Rightarrow aS \Rightarrow aa \end{aligned} \tag{4.55}$$

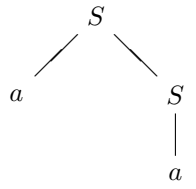
We can represent each of these by a **derivation tree**. This is a rooted tree with branches ordered left to right. The root is S (traditionally put at the top), and the children of each node are the symbols of the string derived from that node in order. For example, in the first derivation in (4.55), the first S derives to SS, so that its two children are S and S. The second S

derives to a, which is therefore its only child, and the third S also derives to a. Thus its tree is drawn this way:



(4.56)

The second derivation in (4.55) has the same tree as the first, whereas the derivation tree of the third is



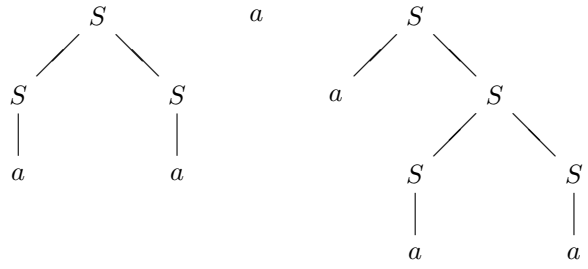
(4.57)

The idea is that the first two derivations are really 'the same, but carried out in a different order'.

A more general derivation, for example

$$SaS \Rightarrow SSaS \Rightarrow aSaS \Rightarrow aaaS \Rightarrow aaaaS \Rightarrow aaaaSS \Rightarrow aaaaaS \Rightarrow aaaaaa \tag{4.58}$$

has a **derivation forest** consisting of one tree for each symbol in the starting string (SaS in this case), with that symbol as its root; each tree consists of the part of the derivation performed on its root. The derivation forest of (4.58) is



(4.59)

The derivation

$$SaS \Rightarrow SSaS \Rightarrow aSaS \Rightarrow aSaaS \Rightarrow aaaaS \Rightarrow aaaaSS \Rightarrow aaaaaS \Rightarrow aaaaaa$$

has the same forest, but this derivation has a different forest, as you can check:

$$SaS \Rightarrow aaS \Rightarrow aaaS \Rightarrow aaaaS \Rightarrow aaaaaS \Rightarrow aaaaaa$$

Again, if two derivations have the same derivation forest, they are essentially the same except for the order of application of the productions.

These examples suggest the following definition.

4.8.15 Definition Let $\mathscr{W}$ be a context free rewrite system on the alphabet A. The 2-category $\mathbf{C}(\mathscr{W})$ associated with $\mathscr{W}$ is defined as follows:

RS–1 $\mathbf{C}(\mathscr{W})$ has one 0-cell.

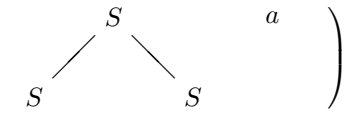
RS–2 The 1-cells of $\mathbf{C}(\mathscr{W})$ are the strings in A^*.

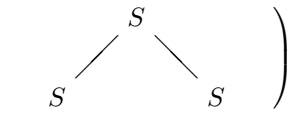
RS–3 A 2-cell $\alpha : w \longrightarrow x$ is the derivation forest of a derivation of x from w using the productions of $\mathscr{W}$.

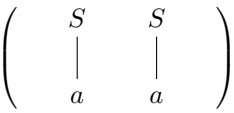
RS–4 If $\alpha : w \longrightarrow x$ and $\beta : w' \longrightarrow x'$ are 2-cells, then $\alpha * \beta : ww' \longrightarrow xx'$ is the forest corresponding to deriving ww' to xw' using α, then deriving xw' to xx' using β. Note that deriving in the opposite order gives the same forest.

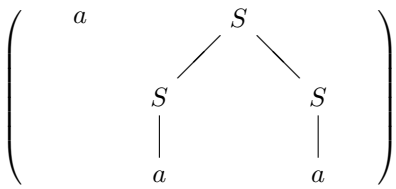
RS–5 If $\alpha : w \longrightarrow x$ and $\gamma : x \longrightarrow y$ are derivation forests, then $\alpha \circ \gamma : w \longrightarrow y$ is the forest obtained by using α, then β. Note that in this case the order does matter. In general, the opposite order won't give a composition at all.

To prove that $\mathbf{C}(\mathscr{W})$ is indeed a 2-category is straightforward but messy. We illustrate the interchange law. Consider these four derivations:

$\alpha =$ 

$\beta =$ 

$\gamma =$ 

$\delta =$ 

It is clear that the derivation shown in (4.59) is both $(\gamma * \delta) \circ (\beta * \alpha)$ and $(\gamma \circ \alpha) * (\delta \circ \beta)$.

4.8.16 Rewrite systems in general Rewrite systems in general have a 2-categorical structure of the same kind as context free rewrite systems. The problem comes in defining when two derivations should be regarded as the same. A reasonable answer is that they are the same when they correspond to the same **pasting scheme**, a 2-categorical notion which specializes to the concept of derivation forest in the case of context free rewrite systems. A pasting scheme is essentially a complex of 0, 1 and 2-cells that compose horizontally and vertically in a unique way to a particular 2-cell. Diagram (4.49) is an example of a pasting scheme: no matter in what order you compose 2-cells in that diagram, you get the same 2-cell. Pasting schemes are characterized in [Johnson, 1989], [Power, 1990a] and [Power, 1991]. Definition 4.8.15 works for that case as well, using pasting schemes for 2-cells instead of derivation forests.

This insight that rewrite systems and grammars are instances of 2-categories suggests that one study rewrite systems for paths in the free category generated by a graph as a way of modeling rewriting systems in typed functional languages as illustrated in Section 2.2.

4.8.17 Exercises

1. Suppose the composites $\beta * \alpha$, $\delta * \gamma$, $\gamma \circ \alpha$ and $\delta \circ \beta$ are all defined. Show that the composite $(\delta * \gamma) \circ (\beta * \alpha)$ is defined.

2. Let $f, g : (A, \leq) \longrightarrow (B, \leq)$ be monotone functions between posets. Define $f \leq g$ to mean that for every $x \in A$, $f(x) \leq g(x)$. Show that this definition makes the category of posets and monotone maps into a poset-enriched category.

3. Show that **Rel** (see 2.1.14) with inclusion of relations as the ordering is a poset-enriched category.

4. Let $f, f' : A \longrightarrow B$ be partial functions (see 2.1.13) with f defined on A_0 and f' defined on A'_0. Define $f \leq f'$ to mean that $A_0 \subseteq A'_0$ and that for $x \in A_0$, $f'(x) = f(x)$. Show that this definition makes **Pfn** a poset-enriched category.

5
Products and sums

This chapter introduces products, which are constructions allowing the definition of operations of arbitrary arity, and sums, which allow the specification of alternatives. In **Set**, the product is essentially the cartesian product, and the sum is disjoint union.

Sections 5.1 through 5.3 introduce products, and Section 5.4 introduces sums. These ideas are used to define the important concept of natural numbers object in Section 5.5. Section 5.6 describes a way to regard formal languages and formal deductive systems as categories. Products and sums then turn out to be familiar constructions. Thus in programming languages products are records with fields, and in deductive systems product becomes conjunction. Finally, Section 5.7 discusses distributive categories (roughly, categories in which products distribute over sums), which are categories with properties that one would expect deductive systems to have.

Except for the last two sections, all the sections of this chapter are used in many places in the rest of the book. The concepts from the last two sections are used only in examples. The last three sections of this chapter are independent of each other.

5.1 The product of two objects in a category

5.1.1 Definition If S and T are sets, the **cartesian product** $S \times T$ is the set of all ordered pairs with first coordinate in S and second coordinate in T; in other words, $S \times T = \{(s,t) \mid s \in S \text{ and } t \in T\}$. The coordinates are functions $\text{proj}_1 : S \times T \longrightarrow S$ and $\text{proj}_2 : S \times T \longrightarrow T$ called the **coordinate projections**, or simply **projections**.

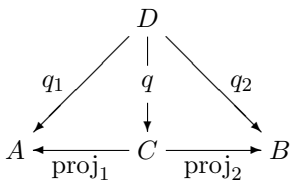
We give a specification of product of two objects in an arbitrary category which will have the cartesian product in **Set** as a special case. This specification is given in terms of the coordinate projections, motivated by these two facts:

(i) you know an element of $S \times T$ by knowing what its two coordinates are, and

(ii) given any element of S and any element of T, there is an element of $S \times T$ with the given element of S as first coordinate and the given element of T as second coordinate.

5.1.2 The product of two objects Let A and B be two objects in a category $\mathscr{C}$. By a (*not* the) **product** of A and B, we mean an object C *together with* arrows $\text{proj}_1 : C \to A$ and $\text{proj}_2 : C \to B$ that satisfy the following condition.

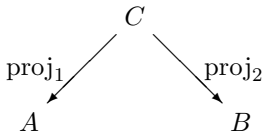
5.1.3 For any object D and arrows $q_1 : D \to A$ and $q_2 : D \to B$, there is a unique arrow $q : D \to C$:



(5.1)

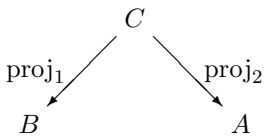
such that $\text{proj}_1 \circ q = q_1$ and $\text{proj}_2 \circ q = q_2$.

5.1.4 Product cones The specification above gives the product as C together with proj_1 and proj_2. The corresponding diagram



(5.2)

is called a **product diagram** or **product cone**, and the arrows proj_i are called the **projections**. These projections are indexed by the set $\{1, 2\}$. The **base** of the cone is the diagram $D : \mathscr{I} \to \mathscr{C}$, where $\mathscr{I}$ is the discrete graph with two nodes 1 and 2 and no arrows. This amounts to saying that the base of the cone is the ordered pair (A, B). The diagram



(5.3)

is regarded as a different product cone since its base is the diagram D with $D(1) = B$ and $D(2) = A$.

By a type of synecdoche, one often says that an object (such as C above) 'is' a product of two other objects (here A and B), leaving the projections implicit, but the projections are nevertheless part of the structure we call 'product'.

Products can be based on discrete graphs with other shape graphs, having more elements (Section 5.3) or having other sets of nodes, for example attributes of a data base such as $\{\texttt{NAME}, \texttt{SALARY}\}$ (see 5.3.14). In this case the projections would be $\text{proj}_{\text{NAME}}$ and $\text{proj}_{\text{SALARY}}$.

The existence of the unique arrow q with the property given in 5.1.3 is called the **universal mapping property** of the product. Any object construction which is defined up to a unique isomorphism (see Theorem 5.2.2) in terms of arrows into or out of it is often said to be defined by a universal mapping property.

5.1.5 Products in Set If S and T are sets, then the cartesian product $S \times T$, together with the coordinate functions discussed in 5.1.1, is indeed a product of S and T in **Set**. For suppose we have a set V and two functions $q_1 : V \longrightarrow S$ and $q_2 : V \longrightarrow T$. The function $q : V \longrightarrow S \times T$ defined by

$$q(v) = (q_1(v), q_2(v))$$

for $v \in V$ is the unique function satisfying 5.1.3. Since $\text{proj}_i(q(v)) = q_i(v)$ by definition, q makes (5.1) commute with $U = S \times T$, and it must be the only such function since the commutativity of (5.1) determines that its value at v *must* be $(q_1(v), q_2(v))$.

We discuss products in **Rel** and in **Pfn** in 5.4.7.

5.1.6 Products in categories of sets with structure In many, but not all, categories of sets with structure, the product can be constructed by endowing the product set with the structure in an obvious way.

5.1.7 Example If S and T are semigroups, then we can make $S \times T$ into a semigroup by defining the multiplication

$$(s_1, t_1)(s_2, t_2) = (s_1 s_2, t_1 t_2)$$

We verify associativity by the calculation

$$\begin{aligned} [(s_1, t_1)(s_2, t_2)](s_3, t_3) &= (s_1 s_2, t_1 t_2)(s_3, t_3) \\ &= ((s_1 s_2)s_3, (t_1 t_2)t_3) \\ &= (s_1(s_2 s_3), t_1(t_2 t_3)) \\ &= (s_1, t_1)(s_2 s_3, t_2 t_3) \\ &= (s_1, t_1)[(s_2, t_2)(s_3, t_3)] \end{aligned} \tag{5.4}$$

Furthermore, this structure together with the coordinate projections satisfies the definition of product in the category of semigroups. To see this requires showing two things:

(a) The arrows $\text{proj}_1 : S \times T \longrightarrow S$ and $\text{proj}_2 : S \times T \longrightarrow T$ are homomorphisms of semigroups.

(b) If q_1 and q_2 are semigroup homomorphisms, then so is the arrow q determined by 5.1.3.

It is necessary to show both because the definition of product in a category $\mathscr{C}$ requires that the arrows occurring in Diagram (5.1) be arrows of the category, in this case, **Sem**.

Requirement (a) follows from this calculation:

$$\begin{aligned}\text{proj}_1((s_1, t_1)(s_2, t_2)) &= \text{proj}_1(s_1 s_2, t_1 t_2) = s_1 s_2 \\ &= \text{proj}_1(s_1, t_1)\,\text{proj}_1(s_2, t_2)\end{aligned}$$

and similarly for proj_2.

As for requirement (b), let R be another semigroup and $q_1 : R \longrightarrow S$ and $q_2 : R \longrightarrow T$ be homomorphisms. Then

$$\begin{aligned}\langle q_1, q_2\rangle(r_1 r_2) &= (q_1(r_1 r_2), q_2(r_1 r_2)) = (q_1(r_1)q_1(r_2), q_2(r_1)q_2(r_2)) \\ &= (q_1(r_1), q_2(r_1))(q_1(r_2), q_2(r_2)) \\ &= \langle q_1, q_2\rangle(r_1)\langle q_1, q_2\rangle(r_2)\end{aligned}$$

A construction for products similar to that for semigroups works for most other categories of sets with structure. Also, the product of categories as defined in 2.6.6 is the product of the categories in **Cat**. One example of a category of sets with structure which lacks products is the category of fields. We discuss this in 8.2.3.

5.1.8 Products in posets We have already seen in 2.3.1 that any poset (partially ordered set) has a corresponding category structure $C(P)$. Let P be a poset and x and y two objects of $C(P)$ (that is, elements of P). Let us see what, if anything, is their product. A product must be an element z together with a pair of arrows $z \longrightarrow x$ and $z \longrightarrow y$, which is just another way of saying that $z \leq x$ and $z \leq y$. The definition of product also requires that for any $w \in P$, given an arrow $w \longrightarrow x$ and one $w \longrightarrow y$, there is an arrow $w \longrightarrow z$.

This translates to

$$w \leq x \text{ and } w \leq y \text{ implies } w \leq z$$

which, together with the fact that $z \leq x$ and $z \leq y$, characterizes z as the **infimum** of x and y, often denoted $x \wedge y$. Thus the existence of products in such a category is equivalent to the existence of infimums. In particular, we see that products generalize a well-known construction in posets. Note that a poset that lacks infimums provides an easy example of a category without products.

5.1.9 Exercises

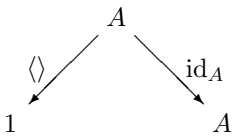
1. Show that the product of two categories, as in 2.6.6, is the product in the category of categories and functors.

2. Describe the product of two monoids in the category of monoids and monoid homomorphisms.

3. Describe the product of two posets in the category of posets and monotone functions.

4. Let $\mathscr{G}$ and $\mathscr{H}$ be two graphs. Show that the product $\mathscr{G} \times \mathscr{H}$ in the category of graphs and homomorphisms is defined as follows: $(\mathscr{G} \times \mathscr{H})_0 = G_0 \times H_0$. An arrow from (g, h) to (g', h') is a pair (a, b) with $a : g \longrightarrow g'$ in $\mathscr{G}$ and $b : h \longrightarrow h'$ in $\mathscr{H}$. The projections are the usual first and second projections.

5. Show that if A is an object in a category with a terminal object 1, then



is a product diagram.

6. Give an example of a product diagram in a category in which at least one of the projections is not an epimorphism.

5.2 Notation for and properties of products

5.2.1 Consider sets $S = \{1, 2, 3\}$, $T = \{1, 2\}$ and $U = \{1, 2, 3, 4, 5, 6\}$. Define $\text{proj}_1 : U \longrightarrow S$ and $\text{proj}_2 : U \longrightarrow T$ by this table:

u	$\mathrm{proj}_1(u)$	$\mathrm{proj}_2(u)$
1	2	1
2	1	1
3	3	1
4	2	2
5	1	2
6	3	2

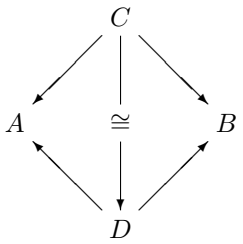
Since the middle and right columns give every possible combination of a number 1, 2 or 3 followed by a number 1 or 2, it follows that U, together with proj_1 and proj_2, is a product of S and T. For example, if $q_1 : V \longrightarrow S$ and $q_2 : V \longrightarrow T$ are given functions and $q_1(v) = 1$, $q_2(v) = 2$ for some v in V, then the unique function $q : V \longrightarrow U$ satisfying 5.1.3 must take v to 5.

In effect, proj_1 and proj_2 code the ordered pairs in $S \times T$ into the set U. As you can see, any choice of a six-element set U and any choice of proj_1 and proj_2 which gives a different element of U for each ordered pair in $S \times T$ gives a product of S and T.

This example shows that the categorical concept of product gives a more general construction than the cartesian product construction for sets. One cannot talk about 'the' product of two objects, but only of 'a' product. However, the following theorem says that two products of the same two objects are isomorphic in a strong sense.

5.2.2 Theorem *Let $\mathscr{C}$ be a category and let A and B be two objects of $\mathscr{C}$. Suppose*

are both product diagrams. Then there is an arrow, and only one, from C to D such that



(5.5)

commutes and this arrow is an isomorphism.

The proof we give is quite typical of the kind of reasoning common in category theory and is worth studying, although not necessarily on first reading.

Proof. Let the projections be $p_1 : C \longrightarrow A$, $p_2 : C \longrightarrow B$, $q_1 : D \longrightarrow A$ and $q_2 : D \longrightarrow B$. In accordance with 5.1.3, there are unique arrows $p : C \longrightarrow D$ and $q : D \longrightarrow C$ for which

$$\begin{array}{rcl} p_1 \circ q & = & q_1 \\ p_2 \circ q & = & q_2 \\ q_1 \circ p & = & p_1 \\ q_2 \circ p & = & p_2 \end{array} \tag{5.6}$$

Thus we already know there is exactly one arrow (namely p) making Diagram (5.5) commute; all that is left to prove is that p is an isomorphism (with inverse q).

The arrow $q \circ p : C \longrightarrow C$ satisfies

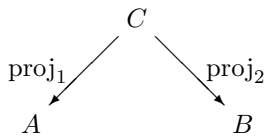
$$p_1 \circ q \circ p = q_1 \circ p = p_1 = p_1 \circ \mathrm{id}_C$$

$$p_2 \circ q \circ p = q_2 \circ p = p_2 = p_2 \circ \mathrm{id}_C$$

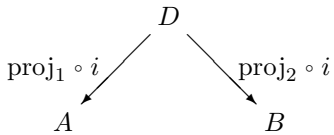
and by the uniqueness part of 5.1.3, it follows that $q \circ p = \mathrm{id}_C$. If we exchange the p's and q's, we similarly conclude that $p \circ q = \mathrm{id}_D$ and hence that p and q are isomorphisms which are inverse to each other. □

The following proposition is a converse to Theorem 5.2.2. Its proof is left as an exercise.

5.2.3 Proposition *Let*



be a product diagram, and suppose that an object D is isomorphic to C by an isomorphism $i : D \longrightarrow C$. Then



is a product diagram.

5.2.4 Categorists specify the product of two sets by saying that all they care about an element of the product is what its first coordinate is and what its second coordinate is. Theorem 5.2.2 says that two structures satisfying this specification are isomorphic in a unique way.

The name '(s,t)' represents the element of the product with first coordinate s and second coordinate t. In a different realization of the product, '(s,t)' represents the element of *that* product with first coordinate s and second coordinate t. The isomorphism of Theorem 5.2.2 maps the representation in the first realization of the product into a representation in the second. Moreover, the universal property of product says that any name '(s,t)' with $s \in S$ and $t \in T$ represents an element of the product: it is the unique element $x \in S \times T$ with $\mathrm{proj}_1(x) = s$ and $\mathrm{proj}_2(x) = t$.

In traditional approaches to foundations, the concept of ordered pair (hence the product of two sets) is defined by giving a specific model (or what a computer scientist might call an implementation) of the specification. Such a definition makes the product absolutely unique instead of unique up to an isomorphism. We recommend that the reader read the discussion of this point in [Halmos, 1960], Section 6, who gives a beautiful discussion of (what in present day language we call) the difference between a specification and an implementation.

In categories other than sets there may well be no standard implementation of products, so the specification given is necessary. In Chapter 15, we will discuss a category known as the category of modest sets in which any construction requires the choice of a bijection between $\mathbf{N}$ and $\mathbf{N} \times \mathbf{N}$. There are many such, and there is no particular reason to choose one over another.

5.2.5 Notation for products It is customary to denote a product of objects A and B of a category as $A \times B$. Precisely, the name $A \times B$ applied to an object means there is a product diagram

$$\begin{array}{ccc} & A \times B & \\ {\scriptstyle \mathrm{proj}_1}\swarrow & & \searrow{\scriptstyle \mathrm{proj}_2} \\ A & & B \end{array} \tag{5.7}$$

Using the name '$A \times B$' implies that there are specific, but unnamed, projections given for the product structure.

If $A = B$, one writes $A \times A = A^2$ and calls it the **cartesian square** of A.

The notations $A \times B$ and A^2 may be ambiguous, but because of Theorem 5.2.2, it does not matter *for categorical purposes* which product the symbol refers to.

Even in the category of sets, you do not really know which set $A \times B$ is unless you pick a specific definition of ordered pair, and the average mathematician does not normally need to give any thought to the definition because what really matters is the universal property that says that an ordered pair is determined by its values under the projections.

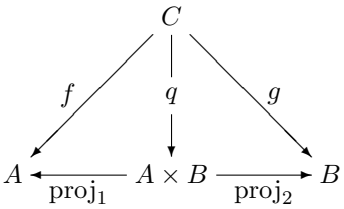
5.2.6 Binary operations A **binary operation** on a set S is a function from $S \times S$ to S. An example is addition on the natural numbers, which is a function $+ : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$. This and other familiar binary operations are usually written in infix notation; one writes $3+5 = 8$, for example, instead of $+(3,5) = 8$. In mathematics texts, the value of an arbitrary binary operation m at a pair (x, y) is commonly denoted xy, without any symbol at all.

Using the concept of categorical product, we can now define the concept of binary operation on any object S of any category provided only that there is a product $S \times S$: a **binary operation** on S is an arrow $S \times S \to S$.

The associative law $(xy)z = x(yz)$ can be described using a commutative diagram as illustrated in 4.1.11. In that section, the diagram is a diagram in **Set**, but now it has a meaning in any category with products. (The meaning of expressions such as mult $\times S$ in arbitrary categories is given in 5.2.17 below.)

The more general concept of function of two variables can now be defined in a categorical setting: an arrow $f : A \times B \to C$ can be thought of as the categorical version of a function of two variables. This has the consequence that a categorist thinks of a function such as $f : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ defined by $f(x, y) = x^2 + y^2$ as a function of *one* variable, but that variable is a structured variable (an ordered pair). The notation we have been using would suggest that one write this as $f((x, y))$ instead of $f(x, y)$, but no one does.

5.2.7 Suppose we are given a product diagram (5.7). For each pair of arrows $f : C \to A$ and $g : C \to B$ requirement 5.1.3 produces a unique $q : C \to A \times B$ making the following diagram commute.



(5.8)

In other words, it produces a function

$$\pi C : \mathrm{Hom}_{\mathscr{C}}(C, A) \times \mathrm{Hom}_{\mathscr{C}}(C, B) \to \mathrm{Hom}_{\mathscr{C}}(C, A \times B)$$

Thus $q = \pi C(f, g)$.

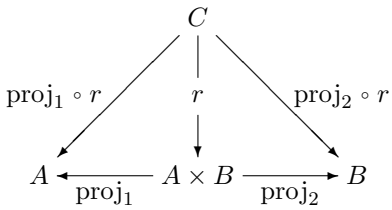
5.2.8 Proposition *The function πC is a bijection.*

Proof. πC is injective, since if (f, g) and (f', g') are elements of

$$\mathrm{Hom}_{\mathscr{C}}(C, A) \times \mathrm{Hom}_{\mathscr{C}}(C, B)$$

both of which produce the same arrow q making Diagram (5.8) commute, then $f = \mathrm{proj}_1 \circ q = f'$, and similarly $g = g'$.

It is also surjective, since if $r : C \to A \times B$ is any arrow of $\mathscr{C}$, then it makes



(5.9)

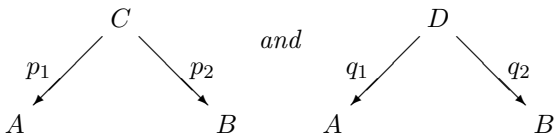
commute, and so is the image of the pair $(\mathrm{proj}_1 \circ r, \mathrm{proj}_2 \circ r)$ under πC. □

5.2.9 It is customary to write $\langle f, g\rangle$ for $\pi C(f, g)$. The *arrow* $\langle f, g\rangle$ internally represents the *pair* of arrows (f, g) of the category $\mathscr{C}$. Proposition 5.2.8 says that the representation is good in the sense that $\langle f, g\rangle$ and (f, g) each determine the other. Proposition 5.2.14 below says that the notation $\langle f, g\rangle$ is compatible with composition.

We have already used the notation '$\langle f, g\rangle$' in the category of sets in 1.2.8.

5.2.10 In the case of two products for the same pair of objects, the isomorphism of 5.2.2 translates the arrow named $\langle f, g\rangle$ for one product into the arrow named $\langle f, g\rangle$ for the other, in the following precise sense.

5.2.11 Proposition *Suppose*



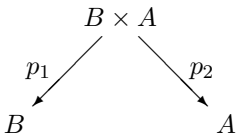
are two product diagrams and $\phi : C \to D$ is the unique isomorphism given by Theorem 5.2.2. Let $f : E \to A$ and $g : E \to B$ be given and let $u : E \to C$, $v : E \to D$ be the unique arrows for which $p_1 \circ u = q_1 \circ v = f$ and $p_2 \circ u = q_2 \circ v = g$. Then $\phi \circ u = v$.

Note that in the statement of the theorem, both u and v could be called '$\langle f, g\rangle$', as described in 5.2.9. The ambiguity occurs because the pair notation does not name which product of A and B is being used. It is rare in practice to have two different products of the same two objects under consideration at the same time.

Proof. By 5.2.2, $p_i = q_i \circ \phi$, $i = 1, 2$. Using this, we have $q_1 \circ \phi \circ u = p_1 \circ u = f$ and similarly $q_2 \circ \phi \circ u = g$. Since v is the unique arrow which makes $q_1 \circ v = u$ and $q_2 \circ v = g$, it follows that $\phi \circ u = v$. □

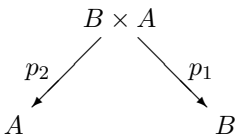
This theorem provides another point of view concerning elements (s, t) of a product $S \times T$. As described in 2.7.19, the element s may be represented by an arrow $s : 1 \longrightarrow S$, and similarly t by $t : 1 \longrightarrow T$. Then the arrow $\langle s, t\rangle : 1 \longrightarrow S \times T$ represents the ordered pair (s, t) whichever realization of $S \times T$ is chosen.

5.2.12 The switch map Our notation $A \times B$ means that $A \times B$ is the vertex of a product cone with base the discrete diagram D with $D(1) = A$ and $D(2) = B$. Then $B \times A$ denotes the product given by the diagram



(5.10)

where we use p_1 and p_2 to avoid confusing them with the arrows proj_1 and proj_2 of Diagram (5.7). (Of course, this is an *ad hoc* solution. If one had to deal with this situation a lot it would be necessary to introduce notation such as $\mathrm{proj}_1^{A,B}$ and $\mathrm{proj}_1^{B,A}$.) Then this is a product diagram:



(5.11)

It follows from Theorem 5.2.2 that there is an isomorphism $\langle p_2, p_1\rangle : B \times A \longrightarrow A \times B$ (called the **switch map**) that commutes with the projections. Its inverse is $\langle \mathrm{proj}_2, \mathrm{proj}_1\rangle : A \times B \longrightarrow B \times A$.

5.2.13 To show that the notation $\langle q_1, q_2\rangle$ is compatible with composition, we will show that the arrows πC defined in 5.2.7 are the components of a natural isomorphism. To state this claim formally, we need to make $\mathrm{Hom}_{\mathscr{C}}(C, A) \times \mathrm{Hom}_{\mathscr{C}}(C, B)$ into a functor. This is analogous to the definition of the contravariant hom functor. A and B are fixed and the varying object is C, so we define the functor $\mathrm{Hom}_{\mathscr{C}}(-, A) \times \mathrm{Hom}_{\mathscr{C}}(-, B)$ as follows:

(i) $[\mathrm{Hom}_{\mathscr{C}}(-,A)\times\mathrm{Hom}_{\mathscr{C}}(-,B)](C)=\mathrm{Hom}_{\mathscr{C}}(C,A)\times\mathrm{Hom}_{\mathscr{C}}(C,B)$, the set of pairs (g,h) of arrows $g:C\longrightarrow A$ and $h:C\longrightarrow B$.

(ii) For $f:D\longrightarrow C$, let $\mathrm{Hom}_{\mathscr{C}}(f,A)\times\mathrm{Hom}_{\mathscr{C}}(f,B)$ be the arrow

$$\mathrm{Hom}_{\mathscr{C}}(C,A)\times\mathrm{Hom}_{\mathscr{C}}(C,B)\longrightarrow\mathrm{Hom}_{\mathscr{C}}(D,A)\times\mathrm{Hom}_{\mathscr{C}}(D,B)$$

that takes a pair (g,h) to $(g\circ f,h\circ f)$.

Now we can state the proposition.

5.2.14 Proposition

$$\pi C:\mathrm{Hom}_{\mathscr{C}}(C,A)\times\mathrm{Hom}_{\mathscr{C}}(C,B)\longrightarrow\mathrm{Hom}_{\mathscr{C}}(C,A\times B)$$

constitutes a natural isomorphism

$$\pi:\mathrm{Hom}(-,A)\times\mathrm{Hom}(-,B)\longrightarrow\mathrm{Hom}(-,A\times B)$$

Proof. We give a proof in detail of this proposition here, but you may want to skip it on first reading, or for that matter on fifteenth reading. We will not always give proofs of similar statements later (of which there are many).

Let $P_{A,B}$ denote the functor $\mathrm{Hom}(-,A)\times\mathrm{Hom}(-,B)$. The projections $p_1:A\times B\longrightarrow A$ and $p_2:A\times B\longrightarrow B$ from the product form a pair $(p_1,p_2)\in P_{A,B}(A\times B)$.

5.2.15 Lemma *The pair (p_1,p_2) is a universal element for $P_{A,B}$.*

Proof. The pair fits the requirements of Proposition 4.5.12 by definition of product: if $(q_1,q_2)\in P_{A,B}(V)$, in other words if $q_1:C\longrightarrow A$ and $q_2:C\longrightarrow B$, there is a unique arrow $q:C\longrightarrow A\times B$ such that $p_i\circ q=q_i$ for $i=1,\,2$. By 5.2.13(ii), $P_{A,B}(q)(p_1,p_2)=(q_1,q_2)$ as required.

Note that this gives an immediate proof of Theorem 5.2.2. See 13.2.3 for another point of view concerning (p_1,p_2).

Continuing the proof of Proposition 5.2.14, from Equation (4.33) it follows that the natural isomorphism α from $\mathrm{Hom}_{\mathscr{C}}(-,A\times B)$ to $P_{A,B}$ induced by this universal element takes $q:V\longrightarrow A\times B$ to $(p_1\circ q,p_2\circ q)$, which is $P_{A,B}(q)(p_1,p_2)$. Then by definition of π we have that $\alpha C=(\pi C)^{-1}$, so that π is the inverse of a natural isomorphism and so is a natural isomorphism. □

It is not hard to give a direct proof of Proposition 5.2.14 using Proposition 5.2.8 and the definition of natural transformation. That definition

requires that the following diagram commute for each arrow $f : C \longrightarrow D$:

$$\begin{array}{ccc} \mathrm{Hom}(C,A) \times \mathrm{Hom}(C,B) & \xrightarrow{\pi D} & \mathrm{Hom}(C, A \times B) \\ \uparrow & & \uparrow \\ \mathrm{Hom}(D,A) \times \mathrm{Hom}(D,B) & \xrightarrow[\pi D]{} & \mathrm{Hom}(D, A \times B) \end{array} \tag{5.12}$$

In this diagram, the left arrow is defined as in Section 5.2.13 and the right arrow is defined as in Section 3.1.21. We leave the details as Exercise 3.

5.2.16 If $f : C \longrightarrow D$ and $q_1 : D \longrightarrow A$ and $q_2 : D \longrightarrow B$ determine $\langle q_1, q_2 \rangle : D \longrightarrow A \times B$, then the commutativity of (5.12) says exactly that

$$\langle q_1 \circ f, q_2 \circ f \rangle = \langle q_1, q_2 \rangle \circ f \tag{5.13}$$

In this sense, the $\langle f, g \rangle$ notation is compatible with composition.

Category theorists say that the single arrow $\langle q_1, q_2 \rangle$ is the *internal* pair of arrows with first coordinate q_1 and second coordinate q_2. The idea behind the word 'internal' is that the category $\mathscr{C}$ is the workspace; inside that workspace the arrow $\langle q_1, q_2 \rangle$ is the pair (q_1, q_2).

When you think of $\mathscr{C}$ as a structure and look at it from the outside, you would say that the arrow q *represents* the *external* pair of arrows (q_1, q_2).

5.2.17 The cartesian product of arrows The cartesian product construction 1.2.9 for functions in sets can also be given a categorical definition. Suppose that $f : S \longrightarrow S'$ and $g : T \longrightarrow T'$ are given. Then the composite arrows $f \circ \mathrm{proj}_1 : S \times T \longrightarrow S'$ and $g \circ \mathrm{proj}_2 : S \times T \longrightarrow T'$ induce, by the definition of product, an arrow denoted $f \times g : S \times T \longrightarrow S' \times T'$ such that

$$\begin{array}{ccccc} S & \xleftarrow{\mathrm{proj}_1} & S \times T & \xrightarrow{\mathrm{proj}_2} & T \\ \downarrow f & & \downarrow f \times g & & \downarrow g \\ S' & \xleftarrow[\mathrm{proj}_1]{} & S' \times T' & \xrightarrow[\mathrm{proj}_2]{} & T' \end{array} \tag{5.14}$$

commutes. Thus $f \times g = \langle f \circ \mathrm{proj}_1, g \circ \mathrm{proj}_2 \rangle$. It is characterized by the properties

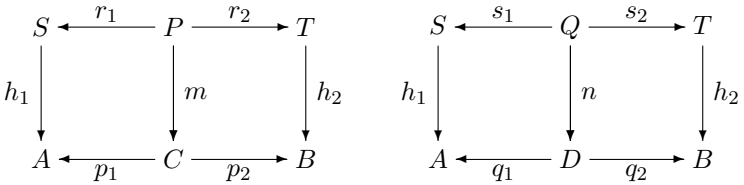
$$\mathrm{proj}_1 \circ (f \times g) = f \circ \mathrm{proj}_1; \quad \mathrm{proj}_2 \circ (f \times g) = g \circ \mathrm{proj}_2$$

Note that we use proj_1 and proj_2 for the product projections among different objects. This is standard and rarely causes confusion since the domains and codomains of the other arrows determine them. We will later call them p_1 and p_2, except for emphasis.

When one of the arrows f or g is an identity arrow, say $f = \mathrm{id}_S$, it is customary to write $S \times g$ for $\mathrm{id}_S \times g$.

An invariance theorem similar to Proposition 5.2.11 is true of cartesian products of functions.

5.2.18 Proposition *Suppose the top and bottom lines of each diagram below are product cones, and that m and n are the unique arrows making the diagrams commute. Let $\psi : P \longrightarrow Q$ and $\phi : C \longrightarrow D$ be the unique isomorphisms given by Theorem 5.2.2. Then $\phi \circ m = n \circ \psi$.*



Proof. For $i = 1, 2$,

$$\begin{aligned} q_i \circ \phi \circ m \circ \psi^{-1} &= p_i \circ m \circ \psi^{-1} \\ &= h_i \circ r_i \circ \psi^{-1} \\ &= h_i \circ s_i \circ \psi \circ \psi^{-1} \\ &= h_i \circ s_i \end{aligned} \tag{5.15}$$

The first equality is the property of ϕ given by Theorem 5.2.2, the second by definition of product applied to C, and the third is the defining property of ψ. It follows that $\phi \circ m \circ \psi^{-1}$ is the unique arrow determined by $h_1 \circ s_1 : Q \longrightarrow A$ and $h_2 \circ s_2 : Q \longrightarrow B$ and the fact that D is a product. But n is that arrow, so that $\phi \circ m \circ \psi^{-1} = n$, whence the theorem. □

5.2.19 Products and composition Let $\mathscr{C}$ be a category with products, and suppose $f_i : A_i \longrightarrow B_i$ and $g_i : B_i \longrightarrow C_i$ for $i = 1, 2$, so that $g_1 \circ f_1$ and $g_2 \circ f_2$ are defined. Then

$$(g_1 \circ f_1) \times (g_2 \circ f_2) = (g_1 \times g_2) \circ (f_1 \times f_2) : A_1 \times A_2 \longrightarrow C_1 \times C_2 \tag{5.16}$$

This follows from the fact that $(g_1 \circ f_1) \times (g_2 \circ f_2)$ is the unique arrow such that

$$\mathrm{proj}_1 \circ ((g_1 \circ f_1) \times (g_2 \circ f_2)) = (g_1 \circ f_1) \circ \mathrm{proj}_1$$

and

$$\mathrm{proj}_2 \circ ((g_1 \circ f_1) \times (g_2 \circ f_2)) = (g_2 \circ f_2) \circ \mathrm{proj}_2$$

Because Diagram (5.14) commutes, we have

$$\begin{aligned} \mathrm{proj}_1 \circ (g_1 \times g_2) \circ (f_1 \times f_2) &= g_1 \circ \mathrm{proj}_1 \circ (f_1 \times f_2) \\ &= g_1 \circ f_1 \circ \mathrm{proj}_1 : A_1 \times A_2 \to C_1 \end{aligned}$$

and similarly

$$\mathrm{proj}_2 \circ (g_1 \times g_2) \circ (f_1 \times f_2) = g_2 \circ f_2 \circ \mathrm{proj}_2 : A_1 \times A_2 \to C_2$$

so the result follows from the uniqueness of $(g_1 \circ f_1) \times (g_2 \circ f_2)$.

This fact allows us to see that the product of two objects is the value of a functor. Define $- \times - : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ as follows: choose, for each pair A and B of $\mathscr{C}$, a product object $A \times B$, and let $(- \times -)(A, B) = A \times B$ and $(- \times -)(f, g) = f \times g$. Equation (5.16) shows that this mapping preserves composition and identities.

Another useful equation is the following, where we assume $f : A \to C$, $g : B \to D$, $u : X \to A$ and $v : X \to B$.

$$(f \times g) \circ \langle u, v \rangle = \langle f \circ u, g \circ v \rangle \tag{5.17}$$

The proof is left as an exercise.

5.2.20 Proposition *Let $\mathscr{C}$ and $\mathscr{D}$ be any categories. If $\mathscr{D}$ has products, then the functor category* $\mathbf{Func}(\mathscr{C}, \mathscr{D})$ *also has products.*

Proof. The product is constructed by constructing the product at each value $F(C)$ and $G(C)$. Precisely, given two functors $F, G : \mathscr{C} \to \mathscr{D}$, the product in $\mathbf{Func}(\mathscr{C}, \mathscr{D})$ of F and G is the functor $F \times G$ defined as follows. For an object C of $\mathscr{C}$, $(F \times G)(C) = F(C) \times G(C)$, the product of the sets $F(C)$ and $G(C)$ in $\mathscr{D}$. For an arrow $f : C \to D$, $(F \times G)(f) = F(f) \times G(f)$, the product of the arrows as defined in 5.2.17. The projection $\pi_1 : F \times G \to F$ is the natural transformation whose component at C is $\pi_1 C = p_i : F(C) \times G(C) \to F(C)$, the product projection in $\mathscr{D}$. For any $f : C \to D$ in $\mathscr{C}$, the diagrams

$$\begin{array}{ccc} FC \times GC & \xrightarrow{p_1} & FC \\ {\scriptstyle F(f)\times G(f)}\downarrow & & \downarrow{\scriptstyle F(f)} \\ FD \times GD & \xrightarrow[p_1]{} & FD \end{array} \qquad \begin{array}{ccc} FC \times GC & \xrightarrow{p_2} & GC \\ {\scriptstyle F(f)\times G(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ FD \times GD & \xrightarrow[p_2]{} & GD \end{array} \tag{5.18}$$

commute by definition of $F(f) \times G(f)$ (5.2.17), so that π_1 and π_2 are natural transformations as required.

Given natural transformations $\alpha : H \longrightarrow F$ and $\beta : H \longrightarrow G$, we must define $\langle \alpha, \beta \rangle : H \longrightarrow F \times G$. For an object C, the component $\langle \alpha, \beta \rangle C = \langle \alpha C, \beta C \rangle : H(C) \longrightarrow F(C) \times \beta(C)$. To see that $\langle \alpha, \beta \rangle$ is a natural transformation, we must show that for any arrow $f : C \longrightarrow D$, this diagram commutes:

$$\begin{array}{ccc} H(C) & \xrightarrow{\langle \alpha, \beta \rangle C} & F(C) \times G(C) \\ {\scriptstyle H(f)}\downarrow & & \downarrow{\scriptstyle F(f) \times G(f)} \\ H(D) & \xrightarrow[\langle \alpha, \beta \rangle D]{} & F(D) \times G(D) \end{array} \tag{5.19}$$

This follows from the following calculation:

$$\begin{aligned} (F(f) \times G(f)) \circ \langle \alpha, \beta \rangle C &= (F(f) \times G(f)) \circ \langle \alpha C, \beta C \rangle \\ &= \langle F(f) \circ \alpha C, G(f) \circ \beta C \rangle \\ &= \langle \alpha D \circ H(f), \beta D \circ H(f) \rangle \\ &= \langle \alpha D, \beta D \rangle \circ H(f) \\ &= \langle \alpha, \beta \rangle D \circ H(f) \end{aligned}$$

in which the first and last equalities are by definition of $\langle \alpha, \beta \rangle$, the second is by Equation (5.17), the third because α and β are natural transformations, and the fourth by Equation (5.13). □

Categorists say that this construction shows that $\mathbf{Func}(\mathscr{C}, \mathscr{D})$ has 'pointwise products'. This is the common terminology, but it might be better to say it has 'objectwise products'.

5.2.21 Corollary *The category of models of a linear sketch has products constructed pointwise.*

This follows from the fact that the category of models of a linear sketch $\mathscr{S}$ is equivalent to the functor category $\mathbf{Func}(\mathbf{Th}(\mathscr{S}), \mathbf{Set})$ (see 4.6.11).

5.2.22 Exercises

1. Give explicitly the isomorphism claimed by Theorem 5.2.2 between $S \times T$ and the set $\{1, 2, 3, 4, 5, 6\}$ expressed as the product of $\{1, 2, 3\}$ and $\{1, 2\}$ using the projections in 5.2.1.

2. Given a two-element set A and a three-element set B, in how many ways can the set $\{1, 2, 3, 4, 5, 6\}$ be made into a product $A \times B$? (This refers to 5.2.1.)

3. Prove that Diagram (5.12) commutes.

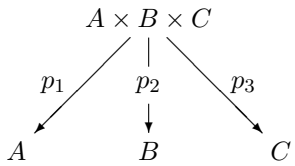
4. Prove Proposition 5.2.3.

5. Let $f : A \longrightarrow C$, $g : B \longrightarrow D$, $u : X \longrightarrow A$ and $v : X \longrightarrow B$. Show that $(f \times g) \circ \langle u, v\rangle = \langle f \circ u, g \circ v\rangle$.

5.3 Finite products

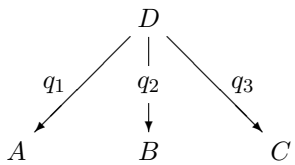
Products of two objects, as discussed in the preceding sections, are called **binary products**. We can define products of more than two objects by an obvious modification of the definition.

For example, if A, B and C are three objects of a category, a product of them is an object $A \times B \times C$ *together with* three arrows:



(5.20)

for which, given any other diagram



there exists a unique arrow $q = \langle q_1, q_2, q_3\rangle : D \longrightarrow A \times B \times C$ such that $p_i \circ q = q_i$, $i = 1, 2, 3$. A diagram of the form (5.20) is called a **ternary product diagram** (or ternary product cone). The general definition of product follows the same pattern.

5.3.1 Definition A **product** of a list $A_1, A_2, \ldots, A_n$ of objects (not necessarily distinct) of a category is an object V together with arrows $p_i : A \longrightarrow A_i$, for $i = 1, \ldots, n$, with the property that given any object B and arrows $f_i : B \longrightarrow A_i$, $i = 1, \ldots, n$, there is a unique arrow $\langle f_1, f_2, \ldots, f_n\rangle : B \longrightarrow A$ for which $p_i \circ \langle f_1, f_2, \ldots, f_n\rangle = f_i$, $i = 1, \ldots, n$.

A product of such a list $A_1, A_2, \ldots, A_n$ is called an **n-ary product** when it is necessary to specify the number of factors. Such a product may be denoted $A_1 \times A_2 \times \cdots \times A_n$ or $\prod_{i=1}^{n} A_i$.

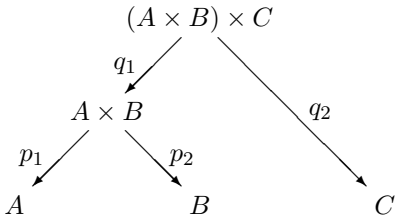
The following uniqueness theorem for general finite products can be proved in the same way as Theorem 5.2.2.

5.3.2 Theorem *Suppose $A_1, A_2, \ldots, A_n$ are objects of a category $\mathscr{C}$ and that A, with projections $p_i : A \longrightarrow A_i$, and B, with projections $q_i : B \longrightarrow A_i$, are products of these objects. Then there is a unique arrow $\phi : A \longrightarrow B$ for which $q_i \circ \phi = p_i$ for $i = 1, \ldots, n$. Moreover, ϕ is an isomorphism.*

Propositions 5.2.3, 5.2.11 and 5.2.18 also generalize in the obvious way to n-ary products.

5.3.3 Binary products give ternary products An important consequence of the definition of ternary product is that in any category with binary products, and any objects A, B and C, either of $(A \times B) \times C$ and $A \times (B \times C)$ can be taken as ternary products $A \times B \times C$ with appropriate choice of projections.

We prove this for $(A \times B) \times C$. Writing p_i, $i = 1, 2$, for the projections which make $A \times B$ a product of A and B and q_i, $i = 1, 2$ for the projections which make $(A \times B) \times C$ a product of $A \times B$ and C, we claim that

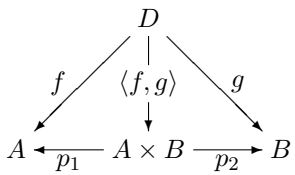


is a product diagram with vertex $(A \times B) \times C$ and projections $p_1 \circ q_1 : (A \times B) \times C \longrightarrow A$, $p_2 \circ q_1 : (A \times B) \times C \longrightarrow B$, and $q_2 : (A \times B) \times C \longrightarrow C$.

Suppose that $f : D \longrightarrow A$, $g : D \longrightarrow B$, and $h : D \longrightarrow C$ are given. We must construct an arrow $u : D \longrightarrow (A \times B) \times C$ with the property that

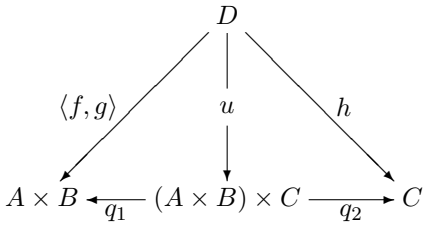
(a) $p_1 \circ q_1 \circ u = f$,
(b) $p_2 \circ q_1 \circ u = g$, and
(c) $q_2 \circ u = h$.

Recall that $\langle f, g \rangle$ is the unique arrow making



(5.21)

commute. This induces a unique arrow $u = \langle\langle f, g\rangle, h\rangle$ making



(5.22)

commute.

The fact that (a) through (c) hold can be read directly off these diagrams. For example, for (a), $p_1 \circ q_1 \circ u = p_1 \circ \langle f, g\rangle = f$.

Finally, if u' were another arrow making (a) through (c) hold, then we would have $p_1 \circ q_1 \circ u' = f$ and $p_2 \circ q_1 \circ u' = g$, so by uniqueness of v as defined by (5.21), $v = q_1 \circ u'$. Since $q_2 \circ u'$ must be h, the uniqueness of u in (5.22) means that $u' = u$.

A generalization of this is stated in Proposition 5.3.10 below.

5.3.4 It follows from the discussion in 5.3.3 that the two objects $(A \times B) \times C$ and $A \times (B \times C)$ are pairwise canonically isomorphic (to each other and to any other realization of the ternary product) in a way that preserves the ternary product structure.

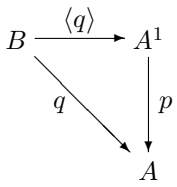
> In elementary mathematics texts the point is often made that 'cartesian product is not associative'. When you saw this you may have thought in your heart of hearts that $(A \times B) \times C$ and $A \times (B \times C)$ are nevertheless really the same. Well, now you know that they are really the same in a very strong sense: they satisfy the *same specification* and so carry exactly the same information. The only difference is in *implementation*.

5.3.5 If all the factors in an n-ary product are the same object A, the n-ary product $A \times A \times \cdots \times A$ is denoted A^n. This suggests the possibility of defining the **nullary product** A^0 and the **unary product** A^1.

5.3.6 For nullary products, the definition is: given no objects of the category $\mathscr{C}$, there should be an object we will temporarily call T, with no arrows from it, such that for any other object B and no arrows from B, there is a unique arrow from B to T subject to no commutativity condition.

When the language is sorted out, we see that a nullary product in $\mathscr{C}$ is simply an object T with the property that every other object of the category has exactly one arrow to T. That is, T must be a terminal object of the category, normally denoted 1. Thus, for any object A of the category, we take $A^0 = 1$. (Compare 4.1.6.)

5.3.7 A unary product A^1 of a single object A should have an arrow $p : A^1 \longrightarrow A$ with the property that given any object B and arrow $q : B \longrightarrow A$ there is a unique arrow $\langle q \rangle : B \longrightarrow A^1$ for which



(5.23)

commutes. The identity $\mathrm{id} : A \longrightarrow A$ satisfies this specification for p; given the arrow $q : B \longrightarrow A$, we let $\langle q \rangle = q : B \longrightarrow A$. The fact that $\mathrm{id} \circ \langle q \rangle = q$ is evident, as is the uniqueness of $\langle q \rangle$. It follows that A^1 can always be taken to be A itself, with the identity arrow as the coordinate arrow.

It is straightforward to show that in general an object B is a unary product of A with coordinate $p : B \longrightarrow A$ if and only if p is an isomorphism (Exercise 1.b).

There is therefore a conceptual distinction between A^1 and A. In the category of sets, A^n is often taken to be the set of strings of elements of A of length exactly n. As you may know, in some computer languages, a string of characters of length one is not the same as a single character, mirroring the conceptual distinction made in category theory.

5.3.8 Definition A category has **binary products** if the product of any two objects exists. It has **canonical binary products** if a specific product diagram is given for each pair of objects. Thus a category with canonical binary products is a category with extra structure given for it. Precisely, a canonical binary products structure on a category $\mathscr{C}$ is a function from $\mathscr{C}_0 \times \mathscr{C}_0$ to the collection of product diagrams in $\mathscr{C}$ which takes a pair (A, B) to a diagram of the form (5.2).

The fact that A and id_A can be taken as a product diagram for any object A of any category means that every category can be given a canonical unary product structure. This is why the distinction between A and A^1 can be and often is ignored.

5.3.9 Definition A category has **finite products** or is a **cartesian category** if the product of any finite number of objects exists. This includes nullary products – in particular, a category with finite products has a terminal object. The category has **canonical finite products** if every finite list of objects has a specific given product.

The following proposition is proved using constructions generalizing those of 5.3.3. (See Exercises 1.c and 6.)

5.3.10 Proposition *If a category has a terminal object and binary products, then it has finite products.*

5.3.11 **Set**, **Grf** and **Cat** all have finite products. In fact, a choice of definition for ordered pairs in **Set** provides canonical products not only for **Set** but also for **Grf** and **Cat**, since products in those categories are built using cartesian products of sets.

5.3.12 Products and initial objects The notation A^0, A^1, A^2 and so on that we have introduced, plus isomorphisms such as $A \times B \cong B \times A$ (5.2.12) and $(A \times B) \times C \cong A \times (B \times C)$ (5.3.4), suggest that other algebraic laws may hold for products. One candidate is $A \times 0 \cong 0$, where 0 denotes the initial object. This is false in general. For example, in **Mon** the initial object is the one-element monoid (which is also the terminal object), and its product with any monoid M is M by Exercise 5 of Section 5.1.

It is important in such areas as programming language semantics and categorical logic that a category have the property $A \times 0 \cong 0$. It is equivalent to another property, as the following proposition states. We include its proof here because it uses several ideas we have introduced.

We denote an initial object by 0 and the unique map to an object A is denoted $! : 0 \longrightarrow A$. As before, a terminal object is denoted 1 and the unique map from an object A is denoted $\langle\rangle : A \longrightarrow 1$.

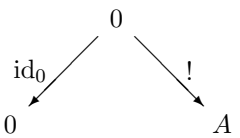
5.3.13 Proposition *Let $\mathscr{C}$ be a category with products and an initial object. Then the following two statements are equivalent.*

(i) *For every object A, if there is an arrow $u : A \longrightarrow 0$, then A is isomorphic to 0.*

(ii) *For every object A, $0 \times A$ is isomorphic to 0.*

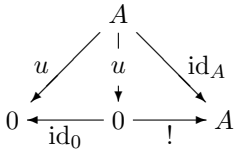
In any category, if the initial object has this property, it is called a **strict initial object**.

Proof. If (i) is true, the map $p_1 : 0 \times A \longrightarrow 0$ serves to force $0 \times A \cong 0$. If (ii) holds, the following must be a product diagram.



(Since $0 \times A$ is isomorphic to 0, there has to be such a product diagram with 0 as vertex, and there is only one possible arrow for each projection.) That

means this diagram must commute:



(5.24)

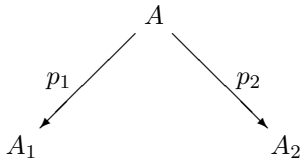
The commutativity of the right triangle says that u is a split monic. Since any arrow to an initial object is epimorphic, the result follows from Proposition 2.9.10. □

5.3.14 Record types To allow operations depending on several variables in a functional programming language L (as discussed in 2.2.1), it is reasonable to assume that for any types A and B the language has a record type P and two field selectors $P \circ A : P \to A$ and $P \circ B : P \to B$. If we insist that the data in P be determined completely by those two fields, it follows that for any pair of operations $f : X \to A$ and $g : X \to B$ there ought to be a unique operation $\langle f, g \rangle : X \to P$ with the property that $P \circ A \circ \langle f, g \rangle = f$ and $P \circ B \circ \langle f, g \rangle = g$. This would make P the product of A and B with the selectors as product projections.

For example, a record type `PERSON` with fields `NAME` and `AGE` could be represented as a product cone whose base diagram is defined on the discrete graph with two nodes `NAME` and `AGE`. If `HUMAN` is a variable of type `PERSON`, then the field selector `HUMAN.AGE` implements the coordinate projection indexed by `AGE`. This example is closer to the spirit of category theory than the cone in Diagram (5.2); there, the index graph has nodes 1 and 2, which suggests an ordering of the nodes (and the projections) which is in fact spurious.

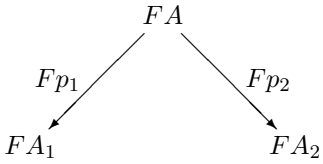
Thus to say that one can always construct record types in a functional programming language L is to say that the corresponding category $C(L)$ has finite products. (See Poigné, [1986].)

5.3.15 Functors that preserve products Let $F : \mathscr{A} \to \mathscr{B}$ be a functor between categories. Suppose that



(5.25)

is a (binary) product diagram in $\mathscr{A}$. We say that F **preserves the product** if



is a product diagram in $\mathscr{B}$. It is important to note that F must preserve the diagram, not merely the object A. It is possible for F to take A to an object which is isomorphic to a product, but do the wrong thing on projections (Exercise 4).

F **preserves canonical products** if $\mathscr{A}$ and $\mathscr{B}$ have canonical products and F preserves the canonical product diagrams.

Similar definitions can be made about a product diagram of any family of objects. A functor is said to **preserve finite products**, respectively **all products** if it preserves every finite product diagram, respectively all product diagrams. Preserving canonical product diagrams is defined analogously (but note Exercise 5.b).

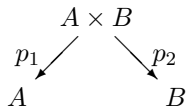
We have a proposition related to Proposition 5.3.10.

5.3.16 Proposition *If a functor preserves terminal objects and binary products, it preserves all finite products.*

Two important examples of this is given by Propositions 5.3.17 and 5.3.18 below.

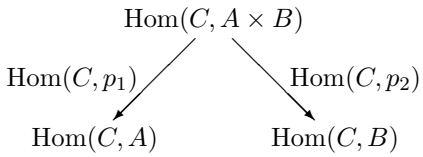
5.3.17 Proposition *Any covariant hom functor preserves products.*

Proof. A covariant hom functor preserves terminal objects by Exercise 11 of Section 3.3. Now suppose



(5.26)

is a product diagram. We must show that for any object C,



(5.27)

is a product diagram in **Set**. In 5.2.7, we defined the function

$$\pi C : \mathrm{Hom}(C, A) \times \mathrm{Hom}(C, B) \longrightarrow \mathrm{Hom}(C, A \times B)$$

which is a bijection that takes (f, g) to $\langle f, g\rangle$. Let $q_1 : \mathrm{Hom}(C, A) \times \mathrm{Hom}(C, B) \longrightarrow \mathrm{Hom}(C, A)$ be the first coordinate function $(f, g) \mapsto f$ in **Set**, and similarly for q_2. Then

$$\mathrm{Hom}(C, p_1)\langle f, g\rangle = p_1 \circ \langle f, g\rangle = f = q_1(f, g)$$

and similarly $\mathrm{Hom}(C, p_2)\langle f, g\rangle = q_2(f, g)$, so $p_i \circ \pi C = q_i$ for $i = 1, 2$. Hence Diagram (5.27) is a product diagram in **Set** by Proposition 5.2.3. □

If F is a functor that preserves products, then any functor isomorphic to it preserves products (Exercise 7), so that it follows from Proposition 5.3.17 that any representable functor preserves products.

5.3.18 Proposition *The second Yoneda embedding*

$$J : \mathscr{C} \longrightarrow \mathbf{Func}(\mathscr{C}^{\mathrm{op}}, \mathbf{Set})$$

(see 4.5.5) preserves products.

Proof. By definition, $J(C) = \mathrm{Hom}(-, C)$ for an object C of $\mathscr{C}$, and if $f : C \longrightarrow D$, $J(f) = \mathrm{Hom}(-, f) : \mathrm{Hom}(-, C) \longrightarrow \mathrm{Hom}(-, D)$. For each X in $\mathscr{C}$,

$$\begin{array}{ccc} \mathrm{Hom}(X, A) \times \mathrm{Hom}(X, B) & & \\ & \searrow^{p_2} \swarrow_{p_1} & \\ \mathrm{Hom}(X, A) & & \mathrm{Hom}(X, B) \end{array}$$

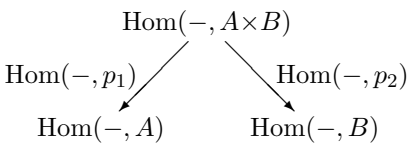
(where p_1 and p_2 are the ordinary projections in **Set**) is a product cone in **Set**. Let $\pi_1 : \mathrm{Hom}(-, A) \times \mathrm{Hom}(-, B) \longrightarrow \mathrm{Hom}(-, A)$ be the natural transformation whose component at X is p_1, and similarly define π_2. Then by the proof of Proposition 5.2.20,

$$\begin{array}{ccc} \mathrm{Hom}(-, A) \times \mathrm{Hom}(-, B) & & \\ & \searrow^{\pi_2} \swarrow_{\pi_1} & \\ \mathrm{Hom}(-, A) & & \mathrm{Hom}(-, B) \end{array}$$

is a product cone in $\mathbf{Func}(\mathscr{C}^{\mathrm{op}}, \mathbf{Set})$. It is easy to show that the diagram

$$\begin{array}{ccc} \mathrm{Hom}(-, A) \times \mathrm{Hom}(-, B) & \xrightarrow{\pi} & \mathrm{Hom}(-, A \times B) \\ & \searrow_{\pi_1} \quad \swarrow_{\mathrm{Hom}(-, p_1)} & \\ & \mathrm{Hom}(-, A) & \end{array}$$

commutes, where π_1 is an instance of the natural transformation defined in Proposition 5.2.14. A similar diagram for π_2 also commutes. It now follows from Proposition 5.2.3 that



is a product diagram in $\mathbf{Func}(\mathscr{C}^{\mathrm{op}}, \mathbf{Set})$, which is what is required to show that J preserves binary products. It is easy to show that it preserves terminal objects. □

5.3.19 Infinite products There is no difficulty in extending the definition of products to allow infinitely many objects. Suppose I is an arbitrary set and $\{A_i\},\ i \in I$ is an indexed set of objects of the category $\mathscr{C}$. (See 2.6.11.) A product of the indexed set is an object P of $\mathscr{A}$ together with an indexed set of arrows $p_i : P \longrightarrow A_i,\ i \in I$, such that given any object A of $\mathscr{A}$ together with arrows $q_i : A \longrightarrow A_i$, for $i \in I$, there is a unique arrow $q = \langle q_i \rangle : A \longrightarrow P$ such that $p_i \circ q = q_i$, for all $i \in I$. The product P is denoted by $\prod_{i \in I} A_i$ or simply $\prod A_i$.

5.3.20 Exercises

1. a. Show that assuming $B \times C$ and $A \times (B \times C)$ (along with the required projections) exist, then the latter with appropriately defined projections is a ternary product $A \times B \times C$.

b. Show that $p : B \longrightarrow A$ is a unary product diagram *if and only if* p is an isomorphism.

c. Show that a category which has binary products and a terminal object has finite products.

2. Let $\mathscr{C}$ be a category with the following properties.

(i) For any two objects A and B there is an object $A \times B$ and arrows $p_1 : A \times B \longrightarrow A$ and $p_2 : A \times B \longrightarrow B$.

(ii) For any two arrows $q_1 : X \longrightarrow A$ and $q_2 : X \longrightarrow B$ there is an arrow $\langle q_1, q_2 \rangle : X \longrightarrow A \times B$.

(iii) For any arrows $q_1 : X \longrightarrow A$ and $q_2 : X \longrightarrow B$, $p_1 \circ \langle q_1, q_2 \rangle = q_1$ and $p_2 \circ \langle q_1, q_2 \rangle = q_2$.

(iv) For any arrow $h : Y \longrightarrow A \times B$, $\langle p_1 \circ h, p_2 \circ h \rangle = h$.

Prove that $\mathscr{C}$ has binary products. (This exercise shows that the property of having binary products can be expressed using rewrite rules.)

3. Show that the underlying functor $U : \mathbf{Cat} \to \mathbf{Grf}$ defined in 3.1.10 preserves products.

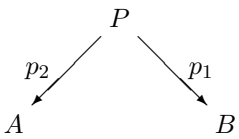
4.† Give an example of categories $\mathscr{C}$ and $\mathscr{D}$ with products and a functor $F : \mathscr{C} \to \mathscr{D}$ which does not preserve products, but for which nevertheless $F(A \times B) \cong F(A) \times F(B)$ for all objects A and B of $\mathscr{C}$. (Hint: Consider the category whose objects are countably infinite sets and arrows are all functions between them.)

5. a. Show that a functor which preserves terminal objects and binary products preserves finite products.

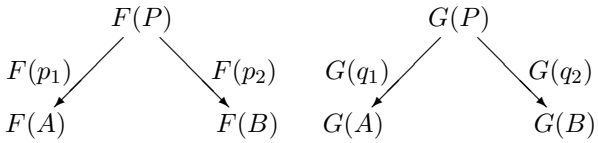
b. Give an example of categories $\mathscr{C}$ and $\mathscr{D}$ with canonical finite products and a functor $F : \mathscr{C} \to \mathscr{D}$ which preserves canonical binary products which does not preserve canonical finite products.

6. Let $\mathbf{N}$ be the set of nonnegative integers with the usual ordering. Show that the category determined by $(\mathbf{N}, \leq)$ has all binary products but no terminal object.

7. Let $F, G : \mathscr{C} \to \mathscr{D}$ be naturally isomorphic functors and suppose



is a product diagram in $\mathscr{C}$. Show that if either of the diagrams



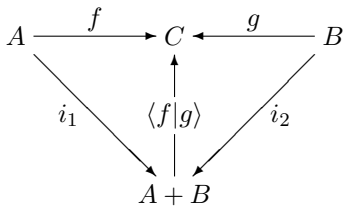
is a product diagram in $\mathscr{D}$ then so is the other. (Hence if a functor preserves products then so does any functor isomorphic to it.)

5.4 Sums

A sum in a category is a product in the dual category. This definition spells that out:

5.4.1 Definition The **sum**, also called the **coproduct**, $A + B$ of two objects in a category consists of an object called $A + B$ together with arrows

$i_1 : A \longrightarrow A + B$ and $i_2 : B \longrightarrow A + B$ such that given any arrows $f : A \longrightarrow C$ and $g : B \longrightarrow C$, there is a unique arrow $\langle f|g\rangle : A + B \longrightarrow C$ for which

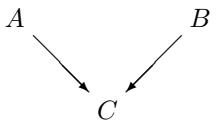


commutes.

The arrows i_1 and i_2 are called the **canonical injections** or the **inclusions** even in categories other than **Set**. These arrows need not be monomorphisms (Exercise 5).

5.4.2 More generally, one can define the sum of any finite or infinite indexed set of objects in a category. The sum of a family $A_1, \ldots, A_n$ is denoted $\sum_{i=1}^{n} A_n$ or $\coprod_{i=1}^{n} A_n$ (the latter symbol is the one typically used by those who call the sum the 'coproduct'). Theorems such as Theorem 5.2.2 and Propositions 5.2.3, 5.2.11 and 5.3.2 are stateable in the opposite category and give uniqueness theorems for sums (this depends on the fact that isomorphisms are isomorphisms in the opposite category). The functor represented by the sum of A and B is $\mathrm{Hom}(A, -) \times \mathrm{Hom}(B, -)$ and the universal element is the pair of canonical injections. (Compare Proposition 5.2.14 and the discussion which follows.)

5.4.3 Definition A **binary discrete cocone** in a category $\mathscr{C}$ is a diagram of the form



It is a **sum cocone** if the two arrows shown make C a sum of A and B.

5.4.4 The notions of a category having sums or of having canonical sums, and that of a functor preserving sums, are defined in the same way as for products. The notion corresponding to $f \times g$ for two arrows f and g is denoted $f + g$ (Exercise 1).

5.4.5 Sums in Set In the category of sets, one can find a sum of two sets in the following way. If S and T are sets, first consider the case that S and T are disjoint. In that case the set $S \cup T$, together with the inclusion

functions $S \longrightarrow S \cup T \longleftarrow T$ is a sum cocone. Given $f : S \longrightarrow C$ and $g : T \longrightarrow C$, then $\langle f|g\rangle(s) = f(s)$ for $s \in S$ and $\langle f|g\rangle(t) = g(t)$ for $t \in T$. In this case, the graph of $\langle f|g\rangle$ is the union of the graphs of f and g.

In the general case, all we have to do is find sets S' and T' isomorphic to S and T, respectively, that are disjoint. The union of those two sets is a sum of S' and T' which are the same, as far as mapping properties, as S and T. The usual way this is done is as follows: let

$$S' = S_0 = \{(s, 0) \mid s \in S\} \quad \text{and} \quad T' = T_1 = \{(t, 1) \mid t \in T\}$$

These sets are disjoint since the first is a set of ordered pairs each of whose second entries is a 0, while the second is a set of ordered pairs each of whose second entries is a 1. The arrow $i_1 : S \longrightarrow S' \cup T'$ takes s to $(s, 0)$, and i_2 takes t to $(t, 1)$. If $f : S \longrightarrow C$ and $g : T \longrightarrow C$, then $\langle f|g\rangle(s, 0) = f(s)$ and $\langle f|g\rangle(t, 1) = g(t)$.

> Note that it will not do to write $S_0 = S \times \{0\}$ and $T_1 = T \times \{1\}$ since our specification of products does not force us to use ordered pairs and, in fact, S is itself a possible product of S with either $\{0\}$ or $\{1\}$.

Our notation $S + T$ for the disjoint union of two sets in **Set** conflicts with a common usage in which S and T are sets of numbers and $S + T$ denotes the set of all their sums. For this reason, many use the notation $S \amalg T$ for what we call $S + T$. We will use the notation $S + T$ here but will remind you by referring to it as the disjoint union.

5.4.6 Sums and products in posets The **least upper bound** or **supremum** of two elements x and y in a poset is an element z with the property that $x \leq z$, $y \leq z$ (z is thus an upper bound) and if for some element w, $x \leq w$ and $y \leq w$, then necessarily $z \leq w$ (z is the – necessarily unique – least upper bound). In the category corresponding to the poset, the supremum of two elements is their categorical sum. We have already seen the dual idea of greatest lower bound or infimum in 5.1.8.

A poset whose corresponding category has all finite sums is called a **sup semilattice** or an **upper semilattice**. The sup of s and t is generally denoted $s \vee t$ and the minimum element (initial object) is denoted 0. In this situation, the minimum element is often called 'bottom'. A poset with all finite products is similarly an **inf semilattice** or **lower semilattice**. A **homomorphism** of sup semilattices is a function such that $f(s \vee t) = f(s) \vee f(t)$. Such a function is monotone because if $s \leq t$ then $t = s \vee t$.

A poset with both finite sups and finite infs is a **lattice**. Some authors assume only binary sups and infs so that a lattice need not have a maximum or minimum element. A good source for the theory of lattices is [Davey and Priestley, 1990].

5.4.7 Sums and products in Rel and Pfn Let S and T be two sets. Their sum in **Rel**, the category of sets and relations (2.1.14) and in **Pfn**, the category of sets and partial functions (2.1.13) is the same as in **Set**: the disjoint union $S + T$. For **Pfn**, the canonical injections are the same as for **Set**, and for **Rel** they are the graphs of the canonical injections for **Set**.

We will verify this for **Rel** (primarily to show the subtleties involved) and leave the claim for **Pfn** as an exercise. Set $i_1 = \{(s, (s, 0)) \mid s \in S\}$ and $i_2 = \{(t, (t, 1)) \mid t \in T\}$. For relations $\alpha : S \longrightarrow X$ and $\beta : T \longrightarrow X$, define

$$\langle \alpha | \beta \rangle = \{((s, 0), x) \mid \text{for all } (s, x) \in \alpha\} \cup \{((t, 1), x) \mid \text{for all } (t, x) \in \beta\}$$

Then

$$\begin{aligned} \langle \alpha | \beta \rangle \circ i_1 &= \{(r, x) \mid \exists z \text{ such that } (r, z) \in i_1 \text{ and } (z, x) \in \langle \alpha | \beta \rangle\} \\ &= \{(r, x) \mid r \in S \text{ and } (r, (r, 0)) \in i_1 \text{ and } ((r, 0), x) \in \langle \alpha | \beta \rangle\} \\ &= \{(r, x) \mid r \in S \text{ and } ((r, 0), x) \in \langle \alpha | \beta \rangle\} \\ &= \{(r, x) \mid r \in S \text{ and } (r, x) \in \alpha\} \\ &= \alpha \end{aligned}$$

and similarly for i_2 and β.

Now suppose we omitted a particular element $((s_0, 0), x_0)$ from $\langle \alpha | \beta \rangle$. Since i_1 has the functional property (see 1.2.3), the only z for which $(s_0, z) \in i_1$ is $(s_0, 0)$, but then $((s_0, 0), x_0) \notin \langle \alpha | \beta \rangle$. On the other hand, if we insert an extra element (r, x) into $\langle \alpha | \beta \rangle$, suppose without loss of generality that $r = (s_0, 0)$ for $s \in S$ and $(s, x) \notin \alpha$. Then $(s, (s_0, 0)) \in i_1$ and $((s_0, 0), x) \in \langle \alpha | \beta \rangle$ but $(s, x) \notin \alpha$. So $\langle \alpha, \beta \rangle$ is the only relation from the disjoint union $S + T$ to X such that $\langle \alpha, \beta \rangle \circ i_1 = \alpha$ and $\langle \alpha, \beta \rangle \circ i_2 = \beta$.

The product of two sets S and T in **Rel** is also the disjoint union $S + T$. It must be that because that is the sum and **Rel** is isomorphic to $\mathbf{Rel}^{\mathrm{op}}$ (Exercise 6 of 3.3). It follows that $p_1 = \{((s, 0), s) \mid s \in S\}$ (the opposite of the graph of i_1) and similarly for p_2.

The product of S and T in **Pfn** is quite different: it is the disjoint union $S \times T + S + T$. The projection p_1 is defined by

$$p_1(r) = \begin{cases} s & \text{if } r = (s, t) \in S \times T \\ r & \text{if } r \in S \\ \text{undefined} & \text{otherwise} \end{cases}$$

5.4.8 Finite sums in programming languages In 5.3.14, we have described products in the category corresponding to a programming language as records. Sums play a somewhat subtler role.

If A and B are types, the sum $A + B$ can be thought of as the free variant or free union of the types A and B. If we are to take this seriously,

we have to consider the canonical structure maps $i_1 : A \longrightarrow A + B$ and $i_2 : B \longrightarrow A + B$. These are type conversions; i_1 converts something of type A to something of the union type. Such type conversions are not explicit in languages such as Pascal or C, because in those languages, the free union is implemented in such a way that the type conversion $i_1 : A \longrightarrow A + B$ can always be described as 'use the same internal representation that it has when it is type A'. Thus in those languages, the effect of an operation with domain '$A + B$' is implementation-dependent.

Thus sums and products provide some elementary constructions for functional programming languages. Still missing are some constructions such as `IF...THEN...ELSE` and `WHILE` loops or recursion which give the language the full power of a Turing machine. `IF...THEN...ELSE` is discussed in Sections 5.7 and 9.6.1. Some approaches to recursion are discussed in 6.6 and 14.2. Wagner [1986a, 1986b] discusses these constructions and others in the context of more traditional imperative programming languages.

5.4.9 Exercises

1. For $f : S \longrightarrow S'$ and $g : T \longrightarrow T'$ in a category with sums, the arrow $f + g : S + T \longrightarrow S' + T'$ is the unique arrow making the opposite of Diagram (5.14) commute. Describe this construction in **Set**.

2. Show that the sum of two elements in a poset is the least upper bound of the elements.

3. Describe the sum of two posets in the category of posets and monotone maps.

4. Show that the sum of two sets S and T in **Pfn** is the disjoint union $S + T$ (see 5.4.5).

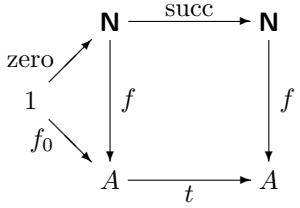
5. Give an example of a category containing a sum with the property that one of the 'canonical injections' is not monic.

5.5 Natural numbers objects

Virtually all the categorical models for programming language semantics suppose there is an object of the category that allows a recursive definition of arrows. To define such an object, we must suppose that the category has a terminal object, which we denote by 1.

5.5.1 Definition An object $\mathbf{N}$, an arrow zero : $1 \longrightarrow \mathbf{N}$ and an arrow succ : $\mathbf{N} \longrightarrow \mathbf{N}$ constitute a **natural numbers object** in a category $\mathscr{C}$ if given any

object A, any arrow $f_0 : 1 \to A$ and any arrow $t : A \to A$, there is then a unique arrow $f : \mathbf{N} \to A$ such that the following diagram commutes:

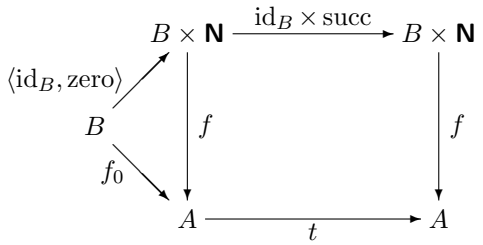


(5.28)

5.5.2 Example A natural numbers object in $\mathscr{C}$ is exactly an initial model in $\mathscr{C}$ of the sketch of Example 4.7.7 (Exercise 2). In particular, the set $\mathbf{N}$ of natural numbers with zero $= 0$ and succ the successor function is a natural numbers object in **Set**. Let the function zero choose the element $0 \in \mathbf{N}$ and let succ be the usual successor function, which we will denote by s. For any set A, a function $1 \to A$ is an element we may call $a_0 \in A$. The commutation of the triangle forces that $f(0) = a_0$ and then we have that $f(1) = f(s0) = t \circ f(0) = t(a_0)$, $f(2) = f(s1) = t \circ f(1) = t^2(a_0)$ and so on. Thus we can show by induction that any function f making the diagram commute must satisfy $f(n) = t^n(a_0)$. This proves both the existence and the uniqueness of f.

In practice, this definition is not strong enough to be really useful and must be strengthened as follows.

5.5.3 Definition An object $\mathbf{N}$ in a category $\mathscr{C}$ together with arrows zero : $1 \to \mathbf{N}$ and succ : $\mathbf{N} \to \mathbf{N}$ is called a **stable** natural numbers object if for all objects A and B and arrows $f_0 : B \to A$ and $t : A \to A$, there is a unique arrow $f : B \times \mathbf{N} \to A$ such that



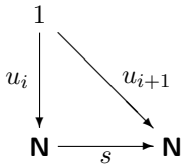
commutes. When $B = 1$, this reduces to the previous definition. We think of B as an object of parameters. For this reason, an $\mathbf{N}$ that satisfies this definition is sometimes called a parametrized natural numbers object, even though it is not $\mathbf{N}$ that is parametrized.

This definition is equivalent to the statement that the product projection $B \times \mathbf{N} \longrightarrow B$ is a natural numbers object in the category $\mathscr{C}/B$ for every object B. Thus we are using 'stable' in its usual sense of 'invariant under slicing'.

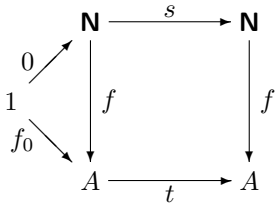
5.5.4 Remark In many sources, what we have called a stable natural numbers object is called, simply, a natural numbers object and the weaker concept is given no name. The reason for this is that the concept was originally defined in cartesian closed categories (the subject of Chapter 6) and in such categories the two notions coincide; see Proposition 6.2.5. In truth, natural numbers objects that are not stable are not very useful.

5.5.5 Theorem *Let $\mathscr{C}$ be a category with a terminal object 1 and $\mathbf{N}$ a sum of countably many copies of 1. Then $\mathbf{N}$ satisfies the specification of a natural numbers object.*

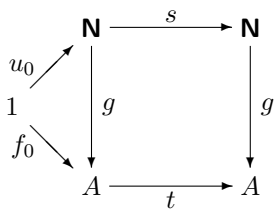
Proof. Let $u_i : 1 \longrightarrow \mathbf{N}$ be the element of the cocone corresponding to the ith copy of the sum. Let $s : \mathbf{N} \longrightarrow \mathbf{N}$ be the unique arrow defined by the formula $s \circ u_i = u_{i+1}$. That is the arrow s is the unique arrow such that for each i, the diagram



commutes. Now suppose $t : A \longrightarrow A$ is an endomorphism and $f_0 : 1 \longrightarrow A$ is a global element. We define a sequence of arrows $f_n : 1 \longrightarrow A$ by ordinary induction by letting f_0 be the given map and $f_{n+1} = t \circ f_n$. Then we use the universal property of the sum to define $f : \mathbf{N} \longrightarrow A$ by $f \circ u_n = f_n$. The commutativity of



is evident. To show uniqueness, suppose $g : \mathbf{N} \longrightarrow A$ is an arrow such that



commutes, then the commutativity of the triangle implies that $g \circ u_0 = f_0$ and if we suppose inductively that $g \circ u_n = f_n = f \circ u_n$, then

$$g \circ u_{n+1} = g \circ s \circ u_n = t \circ g \circ u_n = t \circ f_n = f_{n+1} = f \circ u_{n+1}$$

But then it follows by the uniqueness of arrows from a sum that $f = g$. □

5.5.6 There are two comments we would like to make about Theorem 5.5.5.

First, there is a question of whether assuming countable sums is computationally plausible as a model of programming language semantics. We are not talking about the fact that computers are finite. Despite that, it is a reasonable idealization (and common practice) to allow computational models that, in principle at least, allow indefinitely large objects.

The problem is that these computations, although potentially infinite, should be finitely describable. Thus a function from natural numbers to natural numbers can be imagined computable if it can be described by a finite formula in some sense of the word formula. Since a formula can be described with a finite alphabet in a finite way, one can show that only countably many such formulas exist. On the other hand, if a countable sum of copies of 1 exists, that sum $\mathbf{N}$ can be shown to admit uncountably many functions to itself. It is evident that most of these cannot be given by a formula and make no sense in a model of computation.

Although one cannot actually prove it (owing to imprecision in the notion 'formula'), it seems likely that the class of functions that are computationally reasonable is just the class of recursive functions (Church's thesis). (See [Lewis and Papadimitriou, 1981], Section 5.1, where recursive functions are called μ-recursive.) One should look on a natural numbers object as a computationally meaningful substitute for a countable sum of copies of 1.

The real reason that this theorem is interesting is that it tells us something about the strength of the hypothesis of existence of a natural numbers object; it is weaker than the assumption of existence of countable sums and that is usually considered by category theorists to be fairly weak. Of course, we have just argued above that it is fairly strong, but at any rate the hypothesis of a natural numbers object is weaker.

The second comment is that the construction in Theorem 5.5.5 does not generally give stable natural numbers. Many familiar categories, for example the category of semigroups, have countable (in fact arbitrary) sums and a terminal object and so a natural numbers object which, however, is not stable. In Proposition 6.2.5 and in Section 9.6, Exercise 7, we will see additional assumptions that force the existence of a stable natural numbers object.

In Section 14.2, we study the concept of a locally recursive category, which has a stronger property that implies that it has a natural numbers object.

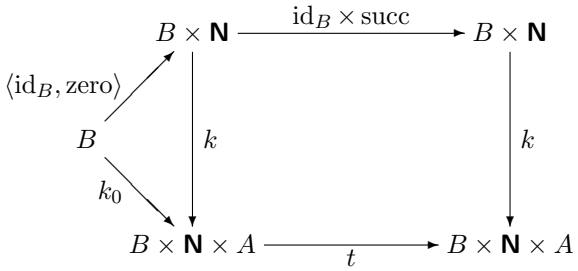
5.5.7 Restricted natural numbers objects James R. Otto studied in his doctoral dissertation, [Otto, 1995], what might be called restricted natural numbers objects. These are objects $\widetilde{N}$, equipped with $\widetilde{\text{zero}} : 1 \to \widetilde{N}$ and $\widetilde{\text{succ}} : \widetilde{N} \to \widetilde{N}$ that have the uniqueness, not necessarily the existence expressed in the Definition 5.5.1. The effect of computation with such restricted natural numbers objects is that only a subset of functions can be computed. With various kinds of restricted NNOs one can find models of restricted recursion in which the constructible functions are, for example, those that can be computed in polynomial time, or that require polynomial space.

5.5.8 Exercises

1. Show that the natural numbers, with zero and the successor function, form a stable natural numbers object in **Set**.

2. Show that a natural numbers object in a category is a model of the sketch of Example 4.7.7 in that category. Show that it is in fact an initial model.

3. Suppose $\mathbf{N}$ is a stable natural numbers object in some category and A and B are arbitrary objects. Show that given any arrows $g : B \to A$ and $h : B \times \mathbf{N} \times A \to A$, there is a unique $f : B \times \mathbf{N} \to A$ such that $f(b, 0) = g(b)$ and $f(b, sn) = h(b, n, f(b, n))$. (This uses variable element notation, as discussed in 2.8.2. It is a great convenience here.) This is the usual formulation of induction, except that B is usually taken to be the cartesian product of a finite number of copies of $\mathbf{N}$. (The way to do this is to find an appropriate arrow t from $B \times \mathbf{N} \times A$ to itself, to define $k_0 : B \to B \times \mathbf{N} \times A$ by $k_0(b) = (b, 0, g(b))$ and then let $k : B \times \mathbf{N} \to B \times \mathbf{N} \times A$ be the unique arrow so that



commutes. Then set $f = p_3 \circ k$.)

5.6 Deduction systems as categories

In this section, we describe briefly the connection between formal logical systems and categories.

5.6.1 A **deduction system** has **formulas** and **proofs**. The informal idea is that the formulas are statements, such as $x \leq 7$, and the proofs are valid lines of reasoning starting with one formula and ending with another; for example, it is valid in high school algebra (which is a deductive system) to prove that if $x \leq 7$ then $2x \leq 14$. We will write a proof p which assumes A and deduces B as $p : A \longrightarrow B$.

Typically, *formal* deduction systems define the formulas by some sort of context free grammar, a typical rule of which might be: 'If A and B are formulas, then so is $A \wedge B$'. The valid proofs are composed of chains of applications of certain specified **rules of inference**, an example of which might be: 'From $A \wedge B$ it is valid to infer A'.

5.6.2 Assumptions on a deduction system As in the case of functional programming languages (see 2.2.4), certain simple assumptions on a deduction system, however it is defined, produce a category.

DS–1 For any formula A there is a proof $\mathrm{id}_A : A \longrightarrow A$.

DS–2 Proofs can be composed: if you have proofs $p : A \longrightarrow B$ and $q : B \longrightarrow C$ then there is a valid proof $p; q : A \longrightarrow C$.

DS–3 If $p : A \longrightarrow B$ is a proof then $p; \mathrm{id}_B$ and $\mathrm{id}_A; p$ are both the *same* proof as p.

DS–4 If p, q and r are proofs, then $(p; q); r$ must be the same proof as $p; (q; r)$, which could be denoted $p; q; r$.

These requirements clearly make a deduction system a category.

One could take a deduction system and impose the minimum requirements just given to produce a category. One could on the other hand go all the way and make any two proofs A to B the same. In that case, each arrow in the category stands for the usual notion of deducible. The choice of an intermediate system of identification could conceivably involve delicate considerations.

5.6.3 Definition A **conjunction calculus** is a deduction system with a formula true and a formula $A \wedge B$ for any formulas A and B, which satisfies CC–1 through CC–5 below for all A and B.

CC–1 There is a proof $A \longrightarrow \mathrm{true}$.

CC–2 If $u : A \longrightarrow \mathrm{true}$ and $v : A \longrightarrow \mathrm{true}$ are proofs, then $u = v$.

CC–3 There are proofs $p_1 : A \wedge B \longrightarrow A$ and $p_2 : A \wedge B \longrightarrow B$ with the property that given any proofs $q_1 : X \longrightarrow A$ and $q_2 : X \longrightarrow B$ there is a proof $\langle q_1, q_2 \rangle : X \longrightarrow A \wedge B$.

CC–4 For any proofs $q_1 : X \longrightarrow A$ and $q_2 : X \longrightarrow B$, $p_1 \circ \langle q_1, q_2 \rangle = q_1$ and $p_2 \circ \langle q_1, q_2 \rangle = q_2$.

CC–5 For any proof $h : Y \longrightarrow A \wedge B$, $\langle p_1 \circ h, p_2 \circ h \rangle = h$.

It is straightforward (Exercise 1) to prove that category corresponding to a conjunction calculus has finite products, with true the terminal object and $\wedge$ the binary product.

Similar constructions may be made using sums to get a disjunction calculus. Cartesian closed categories (the subject of Chapter 6) give an implication operator and quantifiers are supplied in a topos (Chapter 15). These topics are pursued in detail in [Lambek and Scott, 1986], [Makkai and Reyes, 1977] and [Bell, 1988].

5.6.4 Exercises

1. Prove that equations CC–1 through CC–5 make the given deduction system a category with finite products.

2. Prove that in any conjunction calculus, for any objects A, B and C, there are proofs

a. $A \wedge A \longrightarrow A$.
b. $A \longrightarrow A \wedge A$.
c. $A \wedge B \longrightarrow B \wedge A$.
d. $(A \wedge B) \wedge C \longrightarrow A \wedge (B \wedge C)$.

5.7 Distributive categories

Distributive categories, roughly speaking categories in which products distribute over sums, have been proposed as suitable semantics for programs, particularly those aspects involving control (if then else). We introduce them here and describe some examples and applications. Walters [1991] (see also [Walters, 1992]) gives many applications in detail, and Cockett [1993] provides an extensive description of the theory of distributive categories. See also [Kasangian and Vigna, 1991].

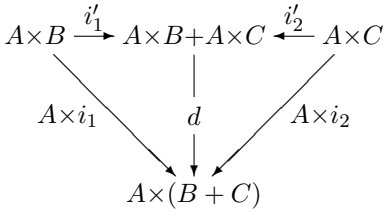
Let $\mathscr{C}$ be a category with binary products and binary sums. Then for any objects A, B and C we have sum cocones

$$B \xrightarrow{i_1} B + C \xleftarrow{i_2} C$$

and

$$A{\times}B \xrightarrow{i'_1} A{\times}B{+}A{\times}C \xleftarrow{i'_2} A{\times}C$$

There is then an arrow $d : A \times B + A \times C \to A \times (B + C)$ that is the unique arrow making



(5.29)

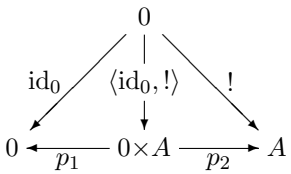
commute.

5.7.1 Definition A **distributive category** is a category with finite sums and finite products in which for all objects A, B and C, the arrow d defined by Diagram (5.29) is an isomorphism.

5.7.2 Remark For an object A in a distributive category, define the functor $A \times -$ as follows: for an object B, the value is $A \times B$; for an arrow f, the value is $\mathrm{id}_A \times f$. The definition of distributive category then says that $A \times -$ preserves binary sums.

The fact that we assume that the category has finite sums and products instead of merely binary sums and products means that it has an initial object and a terminal object (Definition 5.3.9 and its dual). As before, the initial object is denoted 0 and the unique map from 0 to an object A is denoted $! : 0 \to A$. The terminal object is denoted 1 and the unique map to 1 from an object A is denoted $\langle\rangle : A \to 1$. Many authors use $\bot$ for 0 and $\top$ for 1.

Some other consequences of the definition are given by the following proposition. Part (ii) refers to the unique map $\langle \mathrm{id}_0, ! \rangle$ that makes this diagram commute:



(5.30)

5.7.3 Proposition *Let $\mathscr{C}$ be a distributive category. Then*

(i) *Every canonical injection $i_1 : A \to A + B$ and $i_2 : B \to A + B$ is monic.*

(ii) *For every object A, the arrow $\langle \mathrm{id}_0, ! \rangle : 0 \longrightarrow 0 \times A$ is an isomorphism.*

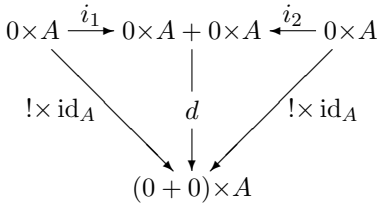
Note that (ii) implies by Proposition 5.3.13 that any distributive category has a strict initial object. In conjunction with 5.7.2 it also shows that the functor $A \times -$ preserves finite sums.

We will prove (ii). This proof gives a good workout of the ideas of this chapter. The proof of (i) uses similar techniques and may be found in [Cockett, 1993].

Proof. We first observe that in the sum diagram

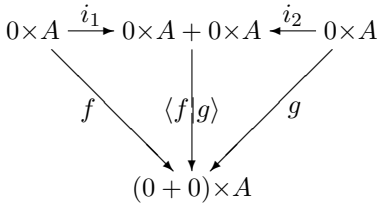
$$0 \longrightarrow 0 + 0 \longleftarrow 0$$

both sum injections are equal to ! because 0 is initial. Now consider



Since d is an isomorphism, it follows that $i_1 = i_2$.

Suppose $f, g : 0 \times A \longrightarrow X$ are arrows. Then the sum diagram



shows that $f = g$. Therefore, there is at most one arrow from $0 \times A$ to any given object. Finally, the composite

$$0 \times A \xrightarrow{\mathrm{proj}_1} 0 \xrightarrow{!} X$$

shows that there is at least one arrow from $0 \times A$ to a given object. Thus $0 \times A$ is an initial object. □

5.7.4 Example **Set** is distributive. If B and C are disjoint, the map $d : A \times B + A \times C \longrightarrow A \times (B + C)$ is the identity map and so has an inverse. If they are not disjoint, then for $a \in A$, $b \in B$ and $c \in C$, using the definition of disjoint union in 5.4.5, $d((a, b), 0) = (a, (b, 0))$ and $d((a, c), 1) = (a, (c, 1))$. Its inverse is then defined by $d^{-1}(a, (b, 0)) = ((a, b), 0)$ and $d^{-1}(a, (c, 1)) = ((a, c), 1)$.

5.7.5 Boolean algebras A poset B is a **Boolean algebra** if

BA–1 For all $x, y \in B$, there is a least upper bound (supremum) $x \vee y$.

BA–2 For all $x, y \in B$, there is a greatest lower bound (infimum) $x \wedge y$.

BA–3 $\wedge$ distributes over $\vee$: for all $x, y, z \in B$,

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

BA–4 B contains two elements 0 and 1 that are the least and greatest elements of B respectively.

BA–5 Each element x has a **complement** $\neg x$, meaning that $x \wedge \neg x = 0$ and $x \vee \neg x = 1$.

It follows that a Boolean algebra has all finite (including empty) infimums and supremums, that both operations $\wedge$ and $\vee$ are commutative and associative and that $\vee$ distributes over $\wedge$. (That these properties follow from BA–1 through BA–5 is not entirely trivial to show.) A Boolean algebra B is **trivial** if $0 = 1$; in that case B contains only one element.

The category corresponding to a Boolean algebra is distributive. Note that the extra distributive law that holds in Boolean algebras does not hold in all distributive categories; in particular, the equation $A + (B \times C) = (A + B) \times (A + C)$ is in general false in **Set**.

5.7.6 The object of truth values Suppose this is a sum cocone

$$1 \xrightarrow{\text{true}} \mathbf{B} \xleftarrow{\text{false}} 1$$

Of course, this makes $\mathbf{B}$ isomorphic to $1 + 1$, but we call it $\mathbf{B}$ because of the role it will play in giving semantics to programs.

Using $\mathbf{B}$, one can simulate 'if–then–else' and similar control functions. Suppose, for example, that we have arrows $f, g : C \longrightarrow D$ and a test function $b : T \longrightarrow \mathbf{B} \cong 1 + 1$. We want to construct an arrow $h : C \times T \longrightarrow D$ such that $h(c, t) = f(c)$ if $b(t) = \text{true}$ and $h(c, t) = g(c)$ if $b(t) = \text{false}$. The composite arrow below has that property, provided one interprets expressions such as $f(c)$ as $f \circ c$ where c is a global element of C.

$$C \times T \xrightarrow{\mathrm{id}_C \times b} C \times \mathbf{B} \xrightarrow{d^{-1}} C \times 1 + C \times 1 \xrightarrow{p_1 + p_2} C + C$$

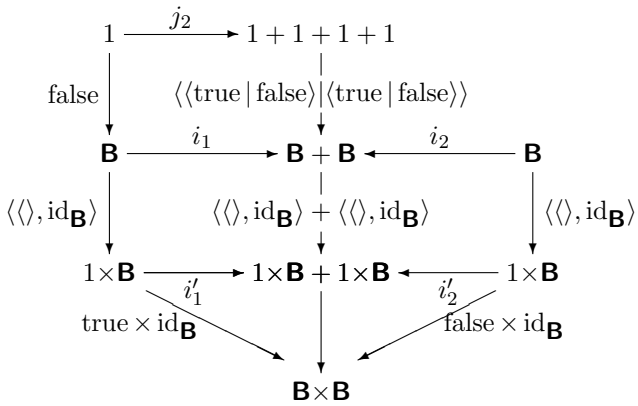
$$\xrightarrow{f + g} D + D \xrightarrow{\langle \mathrm{id}_D \,|\, \mathrm{id}_D \rangle} D$$

(This example is due to Walters [1991].)

5.7.7 Truth tables One can also combine such tests using Boolean operations. In a distributive category, any truth table, for example

P	Q	$P\&Q$
true	true	true
true	false	false
false	true	false
false	false	false

corresponds to an arrow from $\mathbf{B} \times \mathbf{B}$ to $\mathbf{B}$. To see how this works, consider this diagram, in which the i_k, i'_k and j_k are the injections for the sums shown. We have shown only j_2 of the four injections into $1 + 1 + 1 + 1$.



(5.31)

Call the central vertical arrow in this diagram $\phi : 1 + 1 + 1 + 1 \to \mathbf{B} \times \mathbf{B}$. ϕ is a composite of isomorphisms. It is an exercise in diagram chasing to show that $\phi \circ j_2 = \langle \text{true}, \text{false}\rangle : 1 \to \mathbf{B} \times \mathbf{B}$. Similarly, $j_1 = \langle \text{true}, \text{true}\rangle$, $j_3 = \langle \text{false}, \text{true}\rangle$ and $j_4 = \langle \text{false}, \text{false}\rangle$. If we now construct the function $\langle \text{true} \,|\, \text{false} \,|\, \text{false} \,|\, \text{false}\rangle : 1 + 1 + 1 + 1 \to \mathbf{B}$, then

$$q = \langle \text{true} \,|\, \text{false} \,|\, \text{false} \,|\, \text{false}\rangle \circ \phi^{-1} : \mathbf{B} \times \mathbf{B} \to \mathbf{B}$$

satisfies $q \circ \langle \text{true}, \text{true}\rangle = \text{true}$, and q composed with the other three combinations gives false, so that q is essentially the operation of conjunction on $\mathbf{B}$. All the binary operations on $\mathbf{B}$ that correspond to truth tables can be constructed in this way. (Indeed, any function between finite sets can be simulated in this way.) As a result, $\mathbf{B}$ plays the role of a Boolean algebra in the category $\mathscr{C}$.

5.7.8 Using Boolean operations Using the arrow q just defined, we can modify the example in 5.7.6 to employ two Boolean tests b_1 and b_2, where

$h(c, t_1, t_2) = f(c)$ if and only if both $b_1(t_1)$ and $b_2(t_2)$ are true. This arrow does that:

$$C \times T \times T \xrightarrow{\mathrm{id}_C \times b_1 \times b_2} C \times \mathbf{B} \times \mathbf{B} \xrightarrow{\mathrm{id}_c \times q} C \times \mathbf{B}$$

$$\xrightarrow{d_1} C \times 1 + C \times 1 \xrightarrow{p_1 + p_1} C + C$$

$$\xrightarrow{f+g} D + D \xrightarrow{\langle \mathrm{id}_D \,|\, \mathrm{id}_D \rangle} D$$

5.7.9 Exercises

1. Show that **Rel** and **Mon** are not distributive categories.

2. In a category with finite sums, a sum $A + B + C$ is defined using the dual of Diagram (5.20), page 169. Using this notation, show that in a distributive category, $D \times (A + B + C) \cong D \times A + D \times B + D \times C$.

3. This exercise shows how to implement cases in a distributive category. Given an arrow $c : T \to 1 + 1 + 1$ and arrows $f, g, h : C \times T \to D$, show how to implement a function $h : C \times T \to D$ that has value $f(c)$, $g(c)$ or $h(c)$ depending on whether $c(t)$ is in the first, second or third summand of $1 + 1 + 1$.

6
Cartesian closed categories

A cartesian closed category is a type of category that as a formal system has the same expressive power as a typed λ-calculus. In Sections 6.1 and 6.2, we define cartesian closed categories and give some of their properties. In Section 6.3, we describe the concept of typed λ-calculus. Section 6.4 describes the constructions involved in translating from one formalism to the other, without proof. Section 6.5 discusses the issues and technicalities involved in translating between a term in the λ-calculus and an arrow in the corresponding category. Section 6.6 describes the construction of fixed points of endomorphisms of an ω-complete partially ordered object in a cartesian closed category. This is used to provide a semantics for If and While loops. Section 6.6 may be read immediately after Section 6.4.

Sections 6.1 and 6.2 are needed for Chapter 15. Monoidal closed categories, discussed in Chapter 16, are a generalization of cartesian closed categories, but the discussion in Chapter 16 is independent of this chapter. Other than for the chapters just mentioned, this chapter is not needed in the rest of the book.

Most of the cartesian closed categories considered in computer science satisfy a stronger property, that of being 'locally cartesian closed', which is discussed in Section 13.4.

A basic reference for cartesian closed categories and their connection with logic is [Lambek and Scott, 1986]. See also [Huet, 1986] and [Mitchell and Scott, 1989]. The text by Gunter [1992] gives a systematic treatment of programming language semantics in terms of the ideas of this chapter. Substantial applications to computing may be found in [Cousineau, Curien and Mauny, 1985], [Curien, 1986], [Hagino, 1987b] and [Dybkjaer and Melton, 1993].

In this chapter and later, we frequently use the name of an object to stand for the identity arrow on the object: thus 'A' means 'id_A'. This is common in the categorical literature because it saves typographical clutter.

6.1 Cartesian closed categories

6.1.1 Functions of two variables If S and T are sets, then an element of $S \times T$ can be viewed interchangeably as a *pair* of elements, one from S

and one from T, or as a *single* element of the product. If V is another set, then a function $f : S \times T \longrightarrow V$ can interchangeably be viewed as a function of a single variable ranging over $S \times T$ or as a function of two variables, one from S and one from T. Conceptually, we must distinguish between these two points of view, but they are equivalent.

In a more general category, the notion of function of two variables should be understood as meaning an arrow whose domain is a product (see 5.2.6). Under certain conditions, such a function can be converted to a function of one variable with values in a 'function object'. We now turn to the study of this phenomenon.

6.1.2 To **curry** a function of two variables is to change it into a function of one variable whose values are functions of one variable. Precisely, let S, T and V be sets and $f : S \times T \longrightarrow V$ a function. Let $[S \longrightarrow T]$ denote the set of functions from S to T. Then there is a function $\lambda f : S \longrightarrow [T \longrightarrow V]$ defined by letting $\lambda f(s)$ be the function whose value at an element $t \in T$ is $f(s,t)$. The passage from f to λf is called **currying** f. As an example, the definition on arrows of the functor F_α obtained from a monoid action as in 3.2.3 is obtained by currying α.

In the other direction, if $g : S \longrightarrow [T \longrightarrow V]$ is a function, it induces a function $f : S \times T \longrightarrow V$ defined by $f(s,t) = [g(s)](t)$. This construction produces an inverse to λ that determines an isomorphism

$$\mathrm{Hom}_{\mathbf{Set}}(S \times T, V) \cong \mathrm{Hom}_{\mathbf{Set}}(S, [T \longrightarrow V])$$

These constructions can readily be stated in categorical language. The result is a theory that is equivalent to the typed λ-calculus (in the sense of Section 6.3) and has several advantages over it.

6.1.3 Definition A category $\mathscr{C}$ is called a **cartesian closed category** if it satisfies the following:

CCC–1 There is a terminal object 1.

CCC–2 Each pair of objects A and B of $\mathscr{C}$ has a product $A \times B$ with projections $p_1 : A \times B \longrightarrow A$ and $p_2 : A \times B \longrightarrow B$.

CCC–3 For every pair of objects A and B, there is an object $[A \longrightarrow B]$ and an arrow $\mathrm{eval} : [A \longrightarrow B] \times A \longrightarrow B$ with the property that for any arrow $f : C \times A \longrightarrow B$, there is a unique arrow $\lambda f : C \longrightarrow [A \longrightarrow B]$ such that the composite

$$C \times A \xrightarrow{\lambda f \times A} [A \longrightarrow B] \times A \xrightarrow{\mathrm{eval}} B$$

is f.

6.1.4 Terminology Traditionally, $[A \to B]$ has been denoted B^A and called the **exponential object**, and A is then called the exponent. The exponential notation is motivated by the following special case: if $\mathscr{C}$ is the category of sets and $n = \{0, 1, \ldots, n-1\}$, the standard set with n elements, then B^n is indeed the set of n-tuples of elements of B.

In CCC–3, there is a different arrow eval for each pair of objects A and B. When necessary, we will write the arrow eval : $[A \to B] \times A \to B$ as eval^A_B. This collection of arrows for a fixed A forms the counit of an adjunction, as defined in 13.2.5. Because of that, λf is often called the **adjoint transpose** of f.

6.1.5 In view of Proposition 5.3.10, CCC–1 and CCC–2 could be replaced by the requirement that $\mathscr{C}$ have finite products. There is a slight notational problem in connection with this. We have used p_1 and p_2 for the two projections of a binary product. We will now use

$$p_j : \prod A_i \to A_j$$

for $j = 1, \ldots, n$ to denote the jth projection from the n-ary product. This usage should not conflict with the previous usage since it will always be clear what the domain is.

A related problem is this. Condition CCC–3 appears to treat the two factors of $C \times A$ asymmetrically, which is misleading since of course $C \times A \cong A \times C$. Products are of indexed sets of objects (see 5.1.3) not necessarily indexed by an ordered set, even though our notation appears to suggest otherwise. It gets even worse with n-ary products, so we spell out the notation we use more precisely.

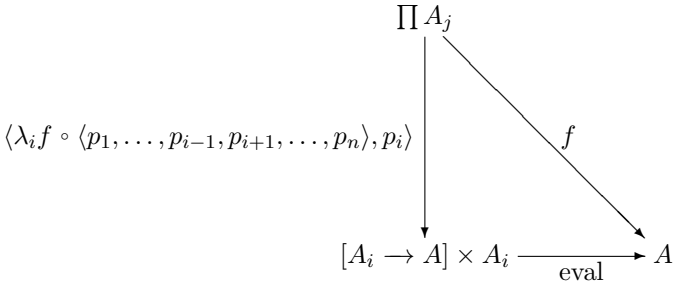
6.1.6 Proposition *For any objects $A_1, \ldots, A_n$ and A of a cartesian closed category and any $i = 1, \ldots, n$, there is an object $[A_i \to A]$ and an arrow*

$$\mathrm{eval} : [A_i \to A] \times A_i \to A$$

such that for any $f : \prod A_j \to A$, there is a unique arrow

$$\lambda_i f : \prod_{j \neq i} A_j \to [A_i \to A]$$

such that the following commutes:



In **Set**, $\lambda_i f$ gives a function from A_i to A for each $(n-1)$-tuple

$$(a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$$

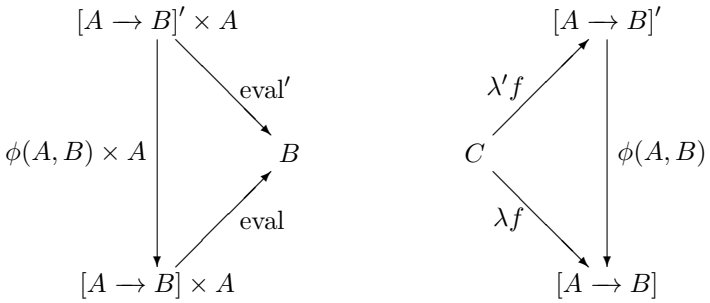
6.1.7 Evaluation as universal element For fixed objects A and B of a cartesian closed category, let $F_{A,B}$ denote the functor $\mathrm{Hom}(- \times A, B)$, so that if $g : D \to C$, $F_{A,B}(g) : \mathrm{Hom}(C \times A, B) \to \mathrm{Hom}(D \times A, B)$ takes $f : C \times A \to B$ to $f \circ (g \times A)$. CCC–3 says that $\mathrm{eval} : [A \to B] \times A \to B$ is a universal element for $F_{A,B}$. This is true for every object A and B and by Proposition 4.5.12 implies that the maps $f \mapsto \lambda f$ form a natural isomorphism of functors:

$$\mathrm{Hom}(- \times A, B) \cong \mathrm{Hom}(-, [A \to B]) \tag{6.1}$$

It follows that a category with finite products can be a cartesian closed category in essentially only one way. The following proposition makes this precise. It is a special case of the fact that adjoints are essentially unique (see 13.3.4).

6.1.8 Proposition *Let $\mathscr{C}$ be a category with finite products. Suppose that for every pair of objects A and B, there are objects $[A \to B]$ and $[A \to B]'$ and arrows* $\mathrm{eval} : [A \to B] \times A \to B$ *and* $\mathrm{eval}' : [A \to B]' \times A \to B$. *Suppose these have the property that for any arrow $f : C \times A \to B$, there are unique arrows $\lambda f : C \to [A \to B]$, $\lambda' f : C \to [A \to B]'$ for which* $\mathrm{eval} \circ (\lambda f \times A) = \mathrm{eval}' \circ (\lambda' f \times A) = f$. *Then for all objects A and B, there is a unique arrow $\phi(A, B) : [A \to B]' \to [A \to B]$ such that for every arrow $f : C \times A \to B$*

the following diagrams commute:



Moreover, ϕ is an isomorphism.

6.1.9 Example The first example of a cartesian closed category is the category of sets. For any sets A and B, $[A \to B]$ is the set of functions from A to B, and eval_B^A is the evaluation or apply function. The meaning of λf is discussed above in 6.1.2. Note that in the case of **Set**, $[A \to B]$ *is* $\mathrm{Hom}_{\mathbf{Set}}(A, B)$.

6.1.10 Example A Boolean algebra (see 5.7.5) B is a poset, so corresponds to a category $C(B)$. This category is a cartesian closed category. The terminal object of a Boolean algebra B is 1, so $C(B)$ satisfies CCC–1. Since B has the infimum of any two elements, $C(B)$ has binary products (see 5.1.8) and so satisfies CCC–2.

To prove CCC–3, define $[a \to b]$ to be $\neg a \vee b$. To show that eval_b^a exists requires showing that $[a \to b] \wedge a \leq b$. This can be seen from the following calculation, which uses the distributive law:

$$\begin{aligned}[a \to b] \wedge a &= (\neg a \vee b) \wedge a \\ &= (\neg a \wedge a) \vee (b \wedge a) \\ &= 0 \vee (b \wedge a) = b \wedge a \leq b\end{aligned}$$

The existence of the λ function requires showing that if $c \wedge a \leq b$ then $c \leq [a \to b]$ (the uniqueness of λf and the fact that $\mathrm{eval} \circ \lambda f \times A = f$ are automatic since no hom set in the category determined by a poset has more than one element). This follows from this calculation, assuming $c \wedge a \leq b$:

$$\begin{aligned}c &= c \wedge 1 = c \wedge (a \vee \neg a) \\ &= (c \wedge a) \vee (c \wedge \neg a) \\ &\leq b \vee (c \wedge \neg a) \leq b \vee \neg a = [a \to b]\end{aligned}$$

6.1.11 Example A poset with all finite (including empty) infs and sups that is cartesian closed as a category is called a **Heyting algebra**. A Heyting algebra is thus a generalization of a Boolean algebra. Heyting algebras correspond to intuitionistic logic in the way that Boolean algebras correspond to classical logic. A Heyting algebra is **complete** if it has sups of all subsets. We use this concept in Chapter 15 (see Exercise 1 of Section 15.5). A thorough exposition of the elementary properties of Heyting algebras may be found in [Mac Lane and Moerdijk, 1992], section I.8.

6.1.12 Example The category whose objects are graphs and arrows are homomorphisms of graphs is also cartesian closed. If $\mathscr{G}$ and $\mathscr{H}$ are graphs, then the exponential $[\mathscr{G} \to \mathscr{H}]$ (which must be a graph) can be described as follows. Let No denote the graph consisting of a single node and no arrows and Ar the graph with one arrow and two distinct nodes, the nodes being the source and the target of the arrow.

There are two embeddings of No into Ar, which we will call $s, t : \mathrm{No} \to \mathrm{Ar}$, that take the single node to the source and target of the arrow of Ar, respectively. Then $[\mathscr{G} \to \mathscr{H}]$ is the graph whose set of nodes is the set $\mathrm{Hom}(\mathscr{G} \times \mathrm{No}, \mathscr{H})$ of graph homomorphisms from $\mathscr{G} \times \mathrm{No}$ to $\mathscr{H}$ and whose set of arrows is the set $\mathrm{Hom}(\mathscr{G} \times \mathrm{Ar}, \mathscr{H})$ with the source and target functions given by $\mathrm{Hom}(\mathscr{G} \times s, \mathscr{H})$ and $\mathrm{Hom}(\mathscr{G} \times t, \mathscr{H})$, respectively. In Exercise 3, we give a more elementary description of the cartesian closed structure. The description given here is a special case of a general result on categories of set-valued functors in 6.1.13 below.

Note that in the case of graphs, the object $[\mathscr{G} \to \mathscr{H}]$ is *not* the set $\mathrm{Hom}(\mathscr{G}, \mathscr{H})$ with the structure of a graph imposed on it in some way. Since $[\mathscr{G} \to \mathscr{H}]$ is an object of the category, it must be a graph; but neither its set of objects nor its set of arrows is $\mathrm{Hom}(\mathscr{G}, \mathscr{H})$. In particular, to prove that a category of sets with structure is not cartesian closed, it is not enough to prove that an attempt to put a structure on the hom set must fail.

6.1.13 Example If $\mathscr{C}$ is a small category then $\mathbf{Func}(\mathscr{C}, \mathbf{Set})$ is a category; the objects are functors and the arrows are natural transformations. (See 4.3.1.) $\mathbf{Func}(\mathscr{C}, \mathbf{Set})$ is a cartesian closed category. If F, $G : \mathscr{C} \to \mathbf{Set}$ are two functors, the product $F \times G$ was defined in 5.2.20. The exponential object $[F \to G]$ is the following functor. For an object C, $[F \to G](C)$ is the set of natural transformations from the functor $\mathrm{Hom}(C, -) \times F$ to G. For $f : C \to D$ an arrow of $\mathscr{C}$ and $\alpha : \mathrm{Hom}(C, -) \times F \to G$ a natural transformation, $[F \to G](f)(\alpha)$ has component at an object A that takes a pair (v, x) with $v : D \to A$ and $x \in F(A)$ to the element $\alpha A(v \circ f, x)$ of $G(A)$. See [Mac Lane and Moerdijk, 1992], page 46 for a proof of a more general theorem of which this is a special case.

Example 6.1.12 is a special case of this example. This also implies, for example, that the category of u-structures described in 4.2.5 is cartesian closed, as is the arrow category of **Set** (see 4.2.17).

6.1.14 Example The category **Cat** of small categories and functors is cartesian closed. In this case, we have already given the construction in 4.3.1: for two categories $\mathscr{C}$ and $\mathscr{D}$, $[\mathscr{C} \longrightarrow \mathscr{D}]$ is the category whose objects are functors from $\mathscr{C}$ to $\mathscr{D}$ and whose arrows are natural transformations between them.

If $F : \mathscr{B} \times \mathscr{C} \longrightarrow \mathscr{D}$ is a functor, then $\lambda F : \mathscr{B} \longrightarrow [\mathscr{C} \longrightarrow \mathscr{D}]$ is the functor defined this way: if B is an object of $\mathscr{B}$, then for an object C of $\mathscr{C}$, $\lambda F(B)(C) = F(B, C)$, and for an arrow $g : C \longrightarrow C'$, $\lambda F(B)(g)$ is the arrow $F(B, g) : (B, C) \longrightarrow (B, C')$ of $\mathscr{D}$. For an arrow $f : B \longrightarrow B'$ of $\mathscr{B}$, $\lambda F(f)$ is the natural transformation from $\lambda F(B)$ to $\lambda F(B')$ with component $\lambda F(f)(C) = F(f, C)$ at an object C of $\mathscr{C}$. It is instructive to check that this does give a natural transformation.

6.1.15 Example The category consisting of ω-CPOs (2.4.3) and continuous functions is a cartesian closed category, although the subcategory of strict ω-CPOs and strict continuous maps is not. The category of continuous lattices and continuous functions between them is cartesian closed; a readable proof is in [Scott, 1972], Section 3. Many other categories of domains have been proposed for programming language semantics and many, but not all, are cartesian closed. More about this is in [Scott, 1982], [Smyth, 1983], [Dybjer, 1986] and [Gunter, 1992].

6.1.16 Example In 2.2.2 we described how to regard a functional programming language as a category. If such a language is a cartesian closed category, then for any types A and B, there is a type $[A \longrightarrow B]$ of functions from A to B. Since that is a type, one can apply programs to data of that type: that is, functions can be operated on by programs. This means that functions are on the same level as other data, often described by saying 'functions are first class objects'.

Proposition 6.1.8 puts strong constraints on making functions into first class objects; if you make certain reasonable requirements on your types $[A \longrightarrow B]$, there is essentially only one way to do it.

6.1.17 Example When a deduction system (Section 5.6) is a cartesian closed category, the constructions giving the exponential turn out to be familiar rules of logic.

Thus, if A and B are formulas, $[A \longrightarrow B]$ is a formula; think of it as A implies B. Then eval is a proof allowing the deduction of B from A and $[A$

$\longrightarrow B]$; in other words, it is *modus ponens*. Given $f : C \times A \longrightarrow B$, that is, given a proof that C and A together prove B, λf is a proof that deduces $[A \longrightarrow B]$ from C. In some logic texts, this is called the rule of detachment.

6.1.18 Exercises

1. Check that the constructions in 6.1.14 make **Cat** a cartesian closed category.

2. In the notation of 6.1.12, show that for any graph $\mathscr{G}$, $\mathscr{G} \times \mathrm{No}$ (the product in the category of graphs and graph homomorphisms) is essentially the set of nodes of $\mathscr{G}$, regarded as a graph with no arrows. (Precisely, find graph homomorphisms from the set of nodes of $\mathscr{G}$ to $\mathscr{G}$ and to No that make the set of nodes the product.)

3.† This exercise provides a description of the cartesian closed structure on the category of graphs and graph homomorphisms that is distinct from that given in 6.1.12. The preceding exercise gives part of the connection between the two descriptions. For two graphs $\mathscr{G}$ and $\mathscr{H}$, define the exponential $[\mathscr{G} \longrightarrow \mathscr{H}]$ in the category of graphs and graph homomorphisms as follows. A node of $[\mathscr{G} \longrightarrow \mathscr{H}]$ is a function from G_0 to H_0 (the nodes of $\mathscr{G}$ and $\mathscr{H}$, respectively). An arrow consists of three functions $f_1 : G_0 \longrightarrow H_0$, $f_2 : G_0 \longrightarrow H_0$ and $f_3 : G_1 \longrightarrow H_1$ that satisfy the condition that for any arrow n of $\mathscr{G}$, $\mathrm{source}(f_3(n)) = f_1(\mathrm{source}(n))$ and $\mathrm{target}(f_3(n)) = f_2(\mathrm{target}(n))$. The source and target of this arrow (f_1, f_2, f_3) are f_1 and f_2, respectively. Define $\mathrm{eval} : [\mathscr{G} \longrightarrow \mathscr{H}] \times \mathscr{G} \longrightarrow \mathscr{H}$ by

$$\begin{cases} \mathrm{eval}_0(f : G_0 \longrightarrow H_0, n) &= f(n) \\ \mathrm{eval}_1((f_1, f_2, f_3), a) &= f_3(a) \end{cases}$$

If $f : \mathscr{C} \times \mathscr{G} \longrightarrow \mathscr{H}$ is a graph homomorphism, define $\lambda f : \mathscr{C} \longrightarrow [\mathscr{G} \longrightarrow \mathscr{H}]$ to be the homomorphism that takes a node c of $\mathscr{C}$ to the function $\lambda f(c)$ for which $\lambda f(c)(g) = f(c, g)$ for any node g of $\mathscr{G}$, and that takes an arrow $a : c \longrightarrow d$ to the arrow $(\lambda f(c), \lambda f(d), f_a)$, where for an arrow u of $\mathscr{G}$, $f_a(u) = f(a, u)$. Show that these constructions provide a cartesian closed structure on the category of graphs and graph homomorphisms.

4.† Prove Proposition 6.1.8.

6.2 Properties of cartesian closed categories

Many nice properties follow from the assumption that a category is cartesian closed. Some of them are easy consequences of theorems concerning adjunctions (Chapter 13). Others can be proved using the Yoneda embedding. We now state some of these basic properties, with some of the proofs outlined.

Given $f : B \to C$ in a cartesian closed category, we have the composite

$$[A \to B] \times A \xrightarrow{\text{eval}_B^A} C \xrightarrow{f} C$$

Given $g : B \to A$ we have

$$[A \to C] \times B \xrightarrow{[A \to C] \times g} [A \to C] \times A \xrightarrow{\text{eval}_C^A} C$$

These are used in the following proposition.

6.2.1 Proposition *Let A be an object of a cartesian closed category $\mathscr{C}$. There are functors $F : \mathscr{C} \to \mathscr{C}$ and $G : \mathscr{C}^{\text{op}} \to \mathscr{C}$ for which*

(i) *$F(B) = [A \to B]$ for an object B, and for an arrow $f : B \to C$,*

$$F(f) = \lambda(f \circ \text{eval}_B^A) : [A \to B] \to [A \to C]$$

(ii) *$G(C) = [A \to C]$ for an object C, and for an arrow $g : B \to A$,*

$$G(g) = \lambda(\text{eval}_C^A \circ [A \to C] \times g) : [A \to C] \to [B \to C]$$

The value of F at $f : B \to C$ is normally written

$$[A \to f] : [A \to B] \to [A \to C]$$

and the value of G at $g : B \to A$ is normally written

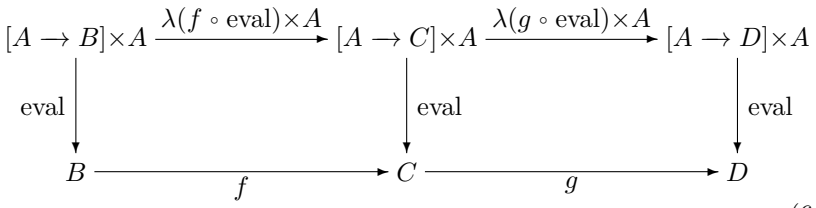
$$[g \to C] : [A \to C] \to [B \to C]$$

These functors are called the **internal hom functors** of the cartesian closed category.

Proof. Preservation of the identity is left as an exercise (Exercise 1). We will prove that F preserves composition. The proof for G is similar. We must show that, for $f : B \to C$ and $g : C \to D$, $[A \to f] \circ [A \to g] = [A \to g \circ f]$. By Definition 6.1.3, for any $h : X \to Y$, $[A \to h]$ is the unique arrow such that

$$\begin{array}{ccc} [A \to X] \times A & \xrightarrow{\lambda(h \circ \text{eval}) \times A} & [A \to Y] \times A \\ \Big\downarrow{\scriptstyle \text{eval}} & & \Big\downarrow{\scriptstyle \text{eval}} \\ X & \xrightarrow[h]{} & Y \end{array}$$

commutes. But then both squares in the following diagram commute, so the outer rectangle commutes, so that $[A \longrightarrow g] \circ [A \longrightarrow f] = [A \longrightarrow g \circ f]$.



(6.2)

□

Now that we have Proposition 6.2.1, we can regard both $\mathrm{Hom}(C \times A, B)$ and $\mathrm{Hom}(C, [A \longrightarrow B])$ as functors of any one of the three variables. Thus for fixed A and B, there are contravariant functors $\mathrm{Hom}(- \times A, B)$ and $\mathrm{Hom}(-, [A \longrightarrow B])$ (we don't need Proposition 6.2.1 to define these), and we have already seen in 6.1.7 that these functors are naturally isomorphic. Because $C \times A \cong A \times C$, A can be treated in the same way as C. For fixed A and C there are covariant functors $\mathrm{Hom}(C \times A, -)$ and $\mathrm{Hom}(C, [A \longrightarrow -])$, all defined as composites of functors. The following proposition now follows from Theorem 13.3.5. It is also not difficult to prove directly.

6.2.2 Proposition *The function $f \mapsto \lambda f$ defines a natural isomorphism*

$$\mathrm{Hom}(C \times A, -) \longrightarrow \mathrm{Hom}(C, [A \longrightarrow -])$$

The next proposition follows from Theorem 13.3.7 (see 5.7.2). It is based on the fact that the definition of cartesian closed category can be reworded as stating that the functor $[A \longrightarrow -]$ defined by Proposition 6.2.1 is the right adjoint to the functor $A \times -$ that takes an object B to $A \times B$ and an arrow $f : B \longrightarrow C$ to $\mathrm{id}_A \times f$.

6.2.3 Proposition *Any cartesian closed category with finite sums is a distributive category.*

The proposition below collects the ways in which $[A \longrightarrow -]$ behaves like a hom functor.

6.2.4 Proposition *The following isomorphisms hold for any objects A, B and C in a cartesian closed category. The last two isomorphisms hold whenever the requisite initial object or sum exists.*

(i) $[A \longrightarrow 1] \cong 1$.

(ii) $[1 \to A] \cong A$.
(iii) $[A \times B \to C] \cong [A \to [B \to C]]$.
(iv) $[A \to B] \times [A \to C] \cong [A \to B \times C]$.
(v) $[0 \to A] \cong 1$.
(vi) $[A + B \to C] \cong [A \to C] \times [B \to C]$.

It is instructive to rewrite these isomorphisms using the notation B^A for the object $[A \to B]$, and to rewrite them as rules of deduction following the construction of 6.1.17.

These isomorphisms are 'natural in all the variables'. This means: fix all but one variable. Then both sides of the isomorphism are functors in the remaining variable in a way analogous to what we did for Proposition 6.2.2, and the isomorphism is a natural isomorphism between those functors. For example, fixing A and C in item (iv), we have naturally isomorphic functors $[A \to -] \times [A \to C]$ and $[A \to (- \times C)]$. Given $f : B \to B'$, $[A \to f]$ is an arrow as defined in Proposition 6.2.1, and so $[A \to f] \times [A \to C]$, meaning $[A \to f] \times \mathrm{id}_{[A \to C]}$, is the product of two arrows as in 5.2.17. $[A \to (- \times C)]$ is similarly defined as a composite of functors.

Each of the isomorphisms in Proposition 6.2.4 can be proved by using fullness of the Yoneda embedding (Theorem 4.5.3). We will prove (iii). For fixed objects B and C we have functors $[- \times B \to C]$ and $[- \to [B \to C]]$ from $\mathscr{C}^{\mathrm{op}}$ to $\mathscr{C}$. Then we have the following chain of natural isomorphisms.

$$\begin{aligned} \mathrm{Hom}(1, [- \times B \to C]) &\cong \mathrm{Hom}(- \times B, C) \\ &\cong \mathrm{Hom}(-, [B \to C]) \\ &\cong \mathrm{Hom}(1, [- \to [B \to C]]) \end{aligned}$$

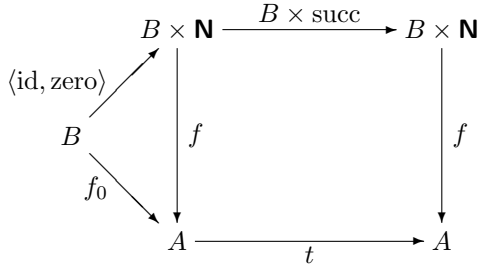
The second one is that of the isomorphism (6.1), page 198, and the first and third are straightforward. Now Theorem 4.5.3 gives an isomorphism from $[- \times B \to C]$ to $[- \to [B \to C]]$.

The final property we consider concerns natural numbers objects (Section 5.5).

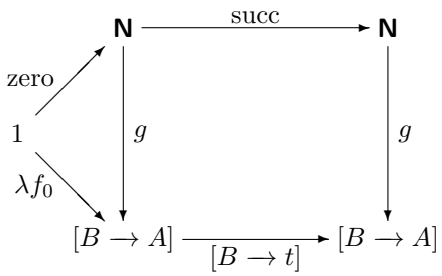
6.2.5 Proposition *Let $\mathscr{C}$ be a cartesian closed category with a natural numbers object. Then the natural numbers object is stable.*

Proof. Note that in a cartesian closed category, to construct an arrow $B \times \mathbf{N} \to A$ is equivalent to constructing one $\mathbf{N} \to [B \to A]$. Let $\mathbf{N}$ be a natural numbers object and suppose $f_0 : B \to A$ and $t : A \to A$. We must find a

unique $f : B \times \mathbf{N} \to A$ for which



commutes. By definition of natural numbers object, we know that there is a unique $g : \mathbf{N} \to [B \to A]$ for which



commutes, where $[B \to t]$ is the arrow defined in Exercise 5.c. Then by CCC–3 and the second diagram,

$$\text{eval} \circ (B \times g) \circ (B \times \text{zero}) = \text{eval} \circ B \times (g \circ \text{zero}) = \text{eval} \circ (B \times \lambda f_0)$$

which is f_0, so that defining $f = \text{eval} \circ B \times g$ will make the triangle in the first diagram commute. (Note that we are using Proposition 6.1.6 here to evaluate on the first coordinate instead of the second.) As for the square, $f \circ (B \times \text{succ}) = \text{eval} \circ (B \times (g \circ \text{succ})) = \text{eval} \circ (B \times ([B \to t] \circ g)) = \text{eval} \circ ((B \times [B \to t]) \circ (B \times g))$ because the square in the second diagram above commutes. By Exercise 5.d, this is

$$\text{eval} \circ (B \times \lambda(t \circ \text{eval})) \circ (B \times g) = t \circ \text{eval} \circ (B \times g) = t \circ f$$

as required. By the uniqueness property of eval, f is unique. □

6.2.6 Exercises

1. Let $\mathscr{C}$ be a cartesian closed category with objects A and B. Show that

$$\lambda(\text{eval}) = \text{id}_{[A \longrightarrow B]} : [A \longrightarrow B] \longrightarrow [A \longrightarrow B]$$

2. Let $g : C' \longrightarrow C$ and $f : C \times A \longrightarrow B$ in a cartesian closed category. Show that

$$\lambda(f \circ (g \times A)) = \lambda f \circ g : C' \longrightarrow [A \longrightarrow B]$$

3. Give a direct proof of Proposition 6.2.2.

4. Show that for any cartesian closed category $\mathscr{C}$, the global elements of $[A \longrightarrow B]$ (see 2.7.19) are in one to one correspondence with the elements of $\text{Hom}_{\mathscr{C}}(A, B)$.

5. This exercise shows how to use the Yoneda embedding to define the functor $[A \longrightarrow f]$. It is intended to be done without using any of the results of this section, since those results depend on Proposition 6.2.1, which is the aim of this problem to prove.

For any objects A, B, C in a cartesian closed category, let ΛB be the map

$$f \mapsto \lambda f : \text{Hom}(C \times A, B) \longrightarrow \text{Hom}(C, [A \longrightarrow B])$$

a. Show that ΛB is a bijection.

b. For fixed A and C and $h : B \longrightarrow B'$, define the function $\widehat{h}C : \text{Hom}(C, [A \longrightarrow B]) \longrightarrow \text{Hom}(C, [A \longrightarrow B'])$ by

$$\widehat{h}C = \Lambda B' \circ \text{Hom}(C \times A, h) \circ (\Lambda B)^{-1}$$

Show that this makes $\widehat{h} : \text{Hom}(-, [A \longrightarrow B]) \longrightarrow \text{Hom}(-, [A \longrightarrow B'])$ a natural transformation of contravariant functors to **Set**.

c. For fixed A and $h : B \longrightarrow B'$, let $[A \longrightarrow h] : [A \longrightarrow B] \longrightarrow [A \longrightarrow B']$ (we must pretend not to know of the definition of $[A \longrightarrow h]$ in the text) be the unique arrow induced by Corollary 4.5.4 and the preceding part of this exercise. Show that Λ is a natural isomorphism from $\text{Hom}(C \times A, -)$ to $\text{Hom}(C, [A \longrightarrow -])$. (This is an instance of Theorem 13.3.2.)

d. Prove that for any A and $h : B \longrightarrow B'$,

$$[A \longrightarrow h] = \lambda(h \circ \text{eval}) : [A \longrightarrow B] \longrightarrow [A \longrightarrow B']$$

6.† Describe the one to one correspondence of Exercise 4 for the category of graphs and graph homomorphisms.

6.3 Typed λ-calculus

In order to describe the connection between cartesian closed categories and typed λ-calculus, we give a brief description of the latter. The material is essentially adapted from [Lambek and Scott, 1984] and [Lambek and Scott, 1986]. It differs from the latter reference in that we do not suppose the existence of a (weak) natural numbers object. The brief discussion in [Lambek, 1986] may be helpful as an overview. The standard reference on the λ-calculus is [Barendregt, 1984], which emphasizes the untyped case.

There is a more general idea of typed λ-calculus in the literature that includes the idea presented here as one extreme. See [Hindley and Seldin, 1986], pp. 168ff.

6.3.1 Definition A **typed λ-calculus** is a formal theory consisting of **types, terms, variables** and **equations**. To each term a, there corresponds a type A, called the type of a. We will write $a \in A$ to indicate that a is a term of type A. These are subject to the following rules:

TL–1 There is a type 1.

TL–2 If A and B are types, then there are types $A \times B$ and $[A \longrightarrow B]$.

TL–3 There is a term $*$ of type 1.

TL–4 For each type A, there is a countable set of terms x_i^A of type A called **variables of type** A.

TL–5 If a and b are terms of type A and B, respectively, there is a term (a, b) of type $A \times B$.

TL–6 If c is a term of type $A \times B$, there are terms $\mathrm{proj}_1(c)$ and $\mathrm{proj}_2(c)$ of type A and B, respectively.

TL–7 If a is a term of type A and f is a term of type $[A \longrightarrow B]$, then there is a term $f\text{`}a$ of type B.

TL–8 If x is a variable of type A and $\phi(x)$ is a term of type B, then $\lambda_{x \in A}\phi(x) \in [A \longrightarrow B]$.

The intended meaning of all the term-forming rules should be clear, with the possible exception of $f\text{`}a$ which is to be interpreted as f applied to a. The notation $\phi(x)$ for the term in TL–8 means that ϕ is a term that *might* contain the variable x. We will use the usual mathematical notation for substitution: if we have called a term $\phi(x)$, then $\phi(a)$ is the term obtained by substituting a for every occurrence of x in the term. In the literature on the λ-calculus, more elaborate notations for substitution are used because they are necessary in complex calculations.

In much of the literature, the symbol ` is omitted; $f\text{`}a$ would be written fa.

6.3.2 Before stating the equations, we need some definitions. We omit, as usual, unnecessary subscripts.

If x is a variable, then x is **free** in the term x. If an occurrence of x is free in either of the terms a or b, then that occurrence of x is free in (a, b). If x occurs freely in either f or a, then that occurrence of x is free in $f\,\text{`}a$. On the other hand, every occurrence of x is not free (and is called **bound**) in $\lambda_x \phi(x)$.

The term a is **substitutable** for x in $\phi(x)$ if no occurrence of a variable in a becomes bound when every occurrence of x in $\phi(x)$ (if any) is replaced by a. A term is called **closed** if no variable is free in it.

6.3.3 The equations take the form $a =_X a'$, where a and a' are terms of the same type and X is a finite set of variables among which are all variables occurring freely in either a or a'.

TL–9 The relation $=_X$ is reflexive, symmetric and transitive.

TL–10 If a is a term of type 1, then $a =_{\{\}} *$.

TL–11 If $X \subseteq Y$, then $a =_X a'$ implies $a =_Y a'$.

TL–12 $a =_X a'$ implies $f\,\text{`}a =_X f\,\text{`}a'$.

TL–13 $f =_X f'$ implies $f\,\text{`}a =_X f'\,\text{`}a$.

TL–14 $\phi(x) =_{X \cup \{x\}} \phi'(x)$ implies $\lambda_x \phi(x) =_X \lambda_x \phi'(x)$.

TL–15 $\text{proj}_1(a, b) =_X a$; $\text{proj}_2(a, b) =_X b$, for $a \in A$ and $b \in B$.

TL–16 $c =_X (\text{proj}_1(c), \text{proj}_2(c))$, for $c \in A \times B$.

TL–17 $\lambda_x \phi(x)\,\text{`}a =_X \phi(a)$, if a is substitutable for x in $\phi(x)$ and $\phi(a)$ is gotten by replacing every free occurrence of x by a in $\phi(x)$.

TL–18 $\lambda_{x \in A}(f\,\text{`}x) =_X f$ provided $x \notin X$ (and is thus not free in f).

TL–19 $\lambda_{x \in A} \phi(x) =_X \lambda_{x' \in A} \phi(x')$ if x' is substitutable for x in $\phi(x)$ and x' is not free in $\phi(x)$ and vice versa.

It is important to understand that the expression $a =_X a'$ does not imply that the terms a and a' are equal. Two terms are equal only if they are identical. The symbol '$=_X$' may be understood as meaning that in any formal interpretation of the calculus, the denotation of the two terms must be the same.

It should be emphasized that these type and term-forming rules and equations are not exhaustive. There may and generally will be additional types, terms and equations. It is simply that a typed λ-calculus must have at least the types, terms and equations described above.

We will usually abbreviate proj_1 and proj_2 by p_1 and p_2.

6.3.4 Exercises

1. Let c, c' be terms of type $A \times B$. Show that $c =_X c'$ implies both that $\mathrm{proj}_1(c) =_X \mathrm{proj}_1(c')$ and that $\mathrm{proj}_2(c) =_X \mathrm{proj}_2(c')$.

2. Let a, a' be terms of type A and b, b' terms of type B. Show that if $a =_X a'$ and $b =_X b'$ then $(a, b) =_X (a', b')$.

6.4 λ-calculus to category and back

Both of the concepts of typed λ-calculus and cartesian closed category are adapted to understanding the calculus of functions of several variables, so it is not surprising that they are equivalent. In this section and the next, we describe the constructions which give this equivalence.

6.4.1 Definition of the category Given a typed λ-calculus $\mathscr{L}$, the objects of the category $C(\mathscr{L})$ are the types of $\mathscr{L}$. An arrow from an object A to an object B is an equivalence class of terms of type B with one free variable of type A (which need not actually occur in the terms).

The equivalence relation is the least reflexive, symmetric, transitive relation induced by saying that two such terms $\phi(x)$ and $\psi(y)$ are equivalent if ϕ and ψ are both of the same type, x and y are both of the same type, x is substitutable for y in ψ, and $\phi(x) =_{\{x\}} \psi(x)$, where $\psi(x)$ is obtained from $\psi(y)$ by substituting x for every occurrence of y.

The reason we need equivalence classes is that any two variables of the same type must correspond to the same arrow, the identity, of that object to itself. If $\lambda_{x \in A} x : 1 \to [A \to A]$ is to name the identity arrow of A for any variable $x \in A$, as is intuitively evident, then the arrow corresponding to a variable x of type A must be the identity of A.

The equivalence relation also makes two terms containing a variable of type 1 equivalent (because of TL–10), thus ensuring that 1 will be a terminal object of the category.

6.4.2 Suppose ϕ is a term of type B with at most one free variable x of type A and ψ is a term of type C with at most one free variable y of type B. Note that by replacing, if necessary, x by a variable that is not bound in ψ, we can assume that ϕ is substitutable for y in ψ. We then define the composite of the corresponding arrows to be the arrow which is the equivalence class of the term $\psi(\phi)$ obtained by substituting ϕ for x in ψ.

6.4.3 Proposition *The category $C(\mathscr{L})$ is a cartesian closed category.*

We will not prove this here. The construction we have given follows Lambek and Scott [1986], who give a proof.

We do not need to say what the cartesian closed structure on $C(\mathscr{L})$ is by virtue of Proposition 6.1.8. Nevertheless, the construction is the obvious one. $A \times B$ with proj_1 and proj_2 is the product of A and B, and $[A \to B]$ is the exponential object. If $\phi(x)$ determines an arrow $f : C \times A \to B$, then x must be a variable of type $C \times A$. Using TL–15, we can substitute (z, y) for x in ϕ, getting $\phi(z, y)$ where z is of type C and y is of type A. Then λf is the equivalence class of $\lambda_z \phi(z, y)$.

Note if z and y actually occur in $\phi(z, y)$, then it is not in any equivalence class, since it has two free variables.

6.4.4 Example We will exhibit a calculation that verifies one of the properties of a cartesian closed category as an example of how the definition of $C(\mathscr{L})$ works.

Let $\mathscr{L}$ be a typed λ-calculus. Define

$$\Lambda : \mathrm{Hom}_{C(\mathscr{L})}(C \times A \to B) \to \mathrm{Hom}_{C(\mathscr{L})}(C \to [A \to B])$$

as follows: for $[\phi(u)] : C \times A \to B$ (so that u is a variable of type $C \times A$), $\Lambda([\phi(u)]) = \lambda_x \phi((z, x))$, where z is a variable of type C and x is a variable of type A. Define

$$\Gamma : \mathrm{Hom}_{C(\mathscr{L})}(C \to [A \to B]) \to \mathrm{Hom}_{C(\mathscr{L})}(C \times A \to B)$$

as follows: for $[\psi(z)] : C \to [A \to B]$, $\Gamma([\psi(z)]) = \psi(\mathrm{proj}_1\, u)\text{`}\,\mathrm{proj}_2\, u$, where u is a variable of type $C \times A$. Then Λ and Γ are inverse functions.

We will show one direction of the verification and leave the other as an exercise. The following calculation uses TL–18 and the fact that x does not occur in $\phi(z)$ because by definition of arrow in $C(\mathscr{L})$, $\phi(z)$ contains only one variable and that is not x.

$$\begin{aligned}
\Lambda(\Gamma(\psi(z))) &= \Lambda(\psi(\mathrm{proj}_1\, u)\text{`}\,\mathrm{proj}_2\, u) \\
&= \lambda_x(\psi(\mathrm{proj}_1(z, x))\text{`}\,\mathrm{proj}_2(z, x)) \\
&=_X \lambda_x \psi(z)\text{`}x \\
&=_X \psi(z)
\end{aligned}$$

6.4.5 Cartesian closed category to λ-calculus Let $\mathscr{C}$ be a cartesian closed category. We will suppose that it has been equipped with given finite products (including, of course, their projections). This means that with each finite indexed set of objects $\{A_i\}$, $i \in I$, there is a given product cone with its projections $p_i : \prod_{i \in I} A_i \to A_i$.

The **internal language** of $\mathscr{C}$ is a typed λ-calculus $\mathbf{L}(\mathscr{C})$. We will describe this λ-calculus by following Definition 6.3.1.

The types of $\mathbf{L}(\mathscr{C})$ are the objects of $\mathscr{C}$. The types required by TL–1 and TL–2 are the objects 1, $A \times B$ and $[A \to B]$. We will assume there is a countable set of variables x_i^A of type A for each object A as required by TL–4. The terms are defined by TL–3 through TL–8, and equality by TL–9 through TL–19.

6.4.6 Theorem *Let $\mathscr{C}$ be a cartesian closed category with internal language $\mathscr{L}$. Then $C(\mathscr{L})$ is a category equivalent to $\mathscr{C}$.*

Lambek and Scott [1986] prove much more than this. They define what it means for languages to be equivalent and show that if you start with a typed λ-calculus, construct the corresponding category, and then construct the internal language, the language and the original typed λ-calculus are equivalent. They state this in a more powerful way in the language of adjunctions.

6.4.7 Exercises

1. Let A and B be types in a λ-calculus $\mathscr{L}$. Show that in $C(\mathscr{L})$, eval : $[A \to B] \times A \to B$ can be taken to be the equivalence class of the term $(\mathrm{proj}_1 u)\text{`}(\mathrm{proj}_2 u)$, where u is a variable of type $[A \to B] \times A$.

2. In the notation of Example 6.4.4, prove that $\Gamma(\Lambda(\phi(u))) = \phi(u)$.

6.5 Arrows vs. terms

It may be instructive to compare what a simple function would look like when defined according to the two kinds of formalism, cartesian closed categories and the λ-calculus.

6.5.1 Example Consider the function $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ that we would traditionally define by the equation

$$f(x, y) = x^2 + 3xy$$

In the typed λ-calculus, this would appear as:

$$f = \lambda_{x \in \mathbf{N}} \lambda_{y \in \mathbf{N}}\ x^2 + 3xy$$

which is directly related to the traditional way of doing it. A rendition in a cartesian closed category is likely to look something like this (in which $*$ is

multiplication):

$$\mathbf{N}\times\mathbf{N}\xrightarrow{\langle p_1,p_1,p_1,p_2,3\rangle}\mathbf{N}\times\mathbf{N}\times\mathbf{N}\times\mathbf{N}\times\mathbf{N}\longrightarrow\cdots$$

$$\cdots\xrightarrow{*\times *\times \mathrm{id}}\mathbf{N}\times\mathbf{N}\times\mathbf{N}\xrightarrow{\mathrm{id}\times *}\mathbf{N}\times\mathbf{N}\xrightarrow{+}\mathbf{N}$$

which appears to be more complicated. However, the categorical formula could be reformulated as:

$$p_1^2+3p_1p_2$$

with an obvious semantics.

Thus we see that the only apparent difference between the two systems, at least for a simple example like this, is that we write p_1 and p_2 instead of x and y. The real difference is whether one chooses the formula or the computational process. It is worth noting that the categorical notation exposes clearly that two of the multiplications, at least, can be carried out in parallel. In fact, by emphasizing the process over the result, the categorical approach would appear to offer a natural way of exploring questions like that.

6.5.2 The time has come to explain one point that we have blurred up to now. If a is a term of type A in variables $x_1\in A_1,\ldots,x_n\in A_n$, and b is a term of type B in variables $y_1\in B_1,\ldots,y_m\in B_m$, then we described (a,b) as a term of type $A\times B$ in variables $x_1,\ldots,x_n,y_1,\ldots,y_m$. A problem arises with this if there is overlap among the variables. For example, if x is a variable of type A, then (x,x) is a term of type $A\times A$. But there is only one free variable in the term. Thus the arrow $\langle x,x\rangle$ corresponds to an arrow $A\longrightarrow A\times A$. The arrow in question is the diagonal arrow $\langle \mathrm{id},\mathrm{id}\rangle$.

The method, using the notation of the preceding paragraph, is to rename and divide the variables into three classes according to whether they are free in both a and b, free only in a or free only in b.

Suppose that $x_1,\ldots,x_n$ of type $A_1,\ldots,A_n$ respectively are free in a alone, $y_1,\ldots,y_m$ of type $B_1,\ldots,B_m$ respectively are free in b alone and $z_1,\ldots,z_r$ of types $C_1,\ldots,C_r$, respectively are free in both. Then

$$\langle\widetilde{a},\widetilde{b}\rangle : A_1\times\cdots\times A_n\times B_1\times\cdots\times B_m\times C_1\times\cdots\times C_r\longrightarrow A\times B$$

is defined as the composite given in (6.3).

$$
\begin{array}{c}
A_1 \times \cdots \times A_n \times B_1 \times \cdots \times B_m \times C_1 \times \cdots \times C_r \\
\Big\downarrow \\
\langle p_1, \ldots, p_n, p_{n+m+1}, \ldots, p_{n+m+r}, p_{n+1}, \ldots, p_{n+m}, p_{n+m+1}, \ldots \rangle \\
\Big\downarrow \\
A_1 \times \cdots \times A_n \times C_1 \times \cdots \times C_r \times B_1 \times \cdots \times B_m \times C_1 \times \cdots \times C_r \\
\Big\downarrow \langle a, b \rangle \\
A \times B
\end{array}
\tag{6.3}
$$

In other words, a generalized diagonal is used on the repeated entries. It must be emphasized that this diagonal is used only when the variables are the same, not merely of the same type. If x and y are distinct free variables of type A, then the term (x, y) corresponds to an arrow $A \times A \longrightarrow A \times A$, namely the identity arrow.

A formal description of a conversion process similar to this in the case of signatures and equations is described in 7.7.6.

A similar identification must be made in the case of function application. The construction is essentially the same and need not be repeated.

6.5.3 The advantages of cartesian closed categories Of all the advantages of cartesian closed categories, the most important is that there being no variables, we never have to worry about any clash of variables. Consider the rule that says that under certain circumstances,

$$\lambda_{x \in A} \phi(x) =_X \lambda_{y \in A} \phi(y)$$

The categorical interpretation of this rule is simply that for

$$f : A_1 \times \cdots \times A_n \times A \longrightarrow B$$

then

$$\lambda f = \lambda f : A_1 \times \cdots \times A_n \longrightarrow [A \longrightarrow B]$$

which is identically true. The rule in λ-calculus has to be circumscribed by conditions. If x and y are two variables of type A and $\phi(x) = y$, then $\lambda_x \phi(x) \neq \lambda_y \phi(y)$. The left hand side still has y free, while the right hand side has no free variables. The real problem comes from the use of variables as what are really only place holders.

Another advantage of the categorical approach, perhaps the most important in the long run, is that the composition is built in. This puts the entire structure of category theory, the machinery of commutative diagrams, etc. at the service of the enterprise.

6.5.4 Exercise

1. Translate the function f into an arrow to $\mathbf{N}$ with domain as shown:
a. $f(x, y) = x^2 + y^2$, domain $\mathbf{N} \times \mathbf{N}$.
b. $f(x, y, z) = x^2 + y^2$, domain $\mathbf{N} \times \mathbf{N} \times \mathbf{N}$.
c. $f(x, y) = 5$, domain $\mathbf{N} \times \mathbf{N}$.

6.6 Fixed points in cartesian closed categories

The main result in this section is that the concept of ω-CPO can be defined in any cartesian closed category, and that the construction of fixed points for such objects can be carried out as in Section 2.4.8. This allows cartesian closed categories to be used as semantic models for functional programming languages. The assumption that *every* object in a cartesian closed category has fixed points is inconsistent with other desirable assumptions on the category [Huwig and Poigné, 1990]. See also [Backhouse *et al.*, 1995].

6.6.1 One of the oft-cited reasons that theoretical computer scientists have studied untyped λ-calculus is the existence of a fixed point combinator. This is an element Y with the property that $x\,`(Y\,`x) = Y\,`x$; that is, $Y\,`x$ is a fixed point of x. This can hardly make sense in the typed λ-calculus. If, for example, there is a type of natural numbers, one could not expect the successor function to have a fixed point. On the other hand, without some kind of fixed point operator, there is no way of interpreting such a functional form as (in the notation of [Backus, 1981a])

$$f = p \Rightarrow q; H(f) \tag{6.4}$$

where $H(f)$ is some kind of function of f. This may be read, 'If p Then q Else $H(f)$'.

It should be noted that although we use Backus' notation for convenience, Backus is interested in actual convergence. Here we are studying the syntactic question; the fixed point combinator is certainly not guaranteed to give a terminating function. In the case of the usual fixed point combinator Y, the fixed point of the identity function is the function $(\lambda x \circ xx)\,`(\lambda x \circ xx)$, the typical example of a nonterminating loop.

In order to describe a program (in some pseudo-language) that actually implements this form, we have to know something about the nature of H. For example, if $H(f) = g \circ f \circ h$, then f in equation (6.4) stands for the infinite program:

```
If p(x) Then Output q(x)
Else If p ∘ h(x) Then Output g ∘ q ∘ h(x)
Else If p ∘ h ∘ h(x) Then Output g ∘ g ∘ q ∘ h ∘ h(x)
...
Else If p ∘ h^n(x) Then Output g^n ∘ q ∘ h^n(x)
...
```

Of course, if the condition is never satisfied, then the program runs forever. This can be programmed in closed form as

```
Proc f(x)
If p(x)
Then Output q(x)
Else Output H(f)(x)
EndIf
EndProc
```

One way around the dilemma mentioned at the beginning of this section is to recognize that one does not need all functions to have a fixed point, only certain ones. Then the question becomes to recognize the ones that do and show that there are enough of them to provide a description of all loops you will need. We turn to the first point first.

6.6.2 Partially ordered objects Suppose D is a partially ordered object in a cartesian closed category. This means that on every hom set $\mathrm{Hom}(A, D)$, there is a partial order relation such that for any $f : B \longrightarrow A$ and any $g, h : A \longrightarrow D$, we have that $g \leq h$ in the hom set $\mathrm{Hom}(A, D)$ implies $g \circ f \leq h \circ f$ in the hom set $\mathrm{Hom}(B, D)$. We will similarly say that D is an **ω-complete partial ordered object** or ω-CPO object if it is a partial ordered object and each hom set is an ω-complete partial ordered set, meaning that every increasing countable chain has a least upper bound.

> There is also a notion of an internally ω-complete partial order. Throughout this section, we understand all references as being to the external notion.

If D and D' are ω-complete partial orders, an arrow $f : D \longrightarrow D'$ is ω-continuous if, for any object A and any sequence $g_0 \leq g_1 \leq \ldots$ of arrows $A \longrightarrow D$ with supremum g, the arrow $f \circ g$ is the supremum of the sequence $f \circ g_0,\ f \circ g_1, \ldots$. Since $f \leq g$ if and only if the supremum of the sequence $f,\ g,\ g,\ g, \ldots$ is g, it easily follows that an ω-continuous arrow is order-preserving.

A partially ordered object D is strict if there is an arrow $\perp : 1 \longrightarrow D$ such that for any object A and any arrow $f : A \longrightarrow D$, we have $\perp \circ \langle\rangle \leq f$. In other words $\perp$ is the least element of D.

6.6.3 Proposition *Suppose D is a strict ω-CPO object and $f : D \longrightarrow D$ is an ω-continuous arrow. Then there is an element $\mathrm{fix}(f) : 1 \longrightarrow D$ such that $f \circ \mathrm{fix}(f) = \mathrm{fix}(f)$. The element $\mathrm{fix}(f)$ is the least element of D with this property.*

Proof. Since $\perp$ is the least element of D, we have that $\perp \leq f \circ \perp$ (the terminal arrow on 1 is id_1). Since f is monotone, $f(\perp) \leq f \circ f(\perp)$. An obvious induction allows us to prove that $f^n(\perp) \leq f^{n+1}(\perp)$, where f^n denotes the n-fold composite of f with itself. Thus we get the sequence

$$\perp \leq f(\perp) \leq f^2(\perp) \leq \cdots \leq f^n(\perp) \leq f^{n+1}(\perp) \leq \cdots$$

We define $\mathrm{fix}(f)$ to be the least upper bound of this sequence. The fact that this is fixed is an immediate consequence of the fact that f preserves the least upper bound of ω-sequences. It is the least fixed point, for if $d : 1 \longrightarrow D$ is another fixed point, one shows by induction beginning from $\perp \leq d$ that $f^n(\perp) \leq d$. □

6.6.4 Application of fixed point theory to programs It is not immediately evident what the above fixed point theory has to do with the interpretation of programs. Although most familiar data types have a partial order on them and may even be thought to be ω-complete, most functions do not preserve the order. For example, squaring does not preserve the natural order on the set of integers.

It is not the basic data types to which we want to apply the above constructions. It is rather the arrow types. We want to find a fixed point, not to the successor function, but rather to the function that assigns to each function f the function $p \Rightarrow q; H(f)$ as in Formula (6.4). Moreover, in thinking about this example, one sees that the appropriate order relation is the one in which $f \leq g$ if the domain of definition of f is included in that of g *and* if they agree on that domain. (Compare the discussion in 2.4.8.) This is how we get the fixed point by successively enlarging the domain. But once an element gets into the domain of definition, no further processing is applied to it. Here is one way of arranging this.

Consider some data type D, that is some object of our category of types. Forget about any order that may exist on D. Define a new data type we will denote $D_\perp$. It is just the object $D + \{\perp\}$ and the order relation is simply that $\perp \leq d$ for all $d \in D$ and for $d \neq \perp$ and $d \neq d' \in D$, it is never the case that $d \leq d'$. A CPO with this property (it has a bottom element but no two other elements are related to each other) is called a **flat CPO**.

We will suppose that $D_\perp$ is ω-complete. If we make certain assumptions about our category (which are true in particular in **Set**) that we will discuss in Section 9.6, it can be shown that this is so. These questions are addressed

in [Barr, 1990], where the results of this section are 'internalized', that is to say, interpreted in the internal language of the category.

If we do make this assumption, then the data type $[A \longrightarrow D_\perp]$ is also an ω-CPO object. In fact an increasing sequence of arrows $B \longrightarrow [A \longrightarrow D_\perp]$ is equivalent to an increasing sequence of arrows $B \times A \longrightarrow D_\perp$ and that has a supremum by assumption. It follows that each continuous endoarrow on $[A \longrightarrow D_\perp]$ has a fixed point.

To apply this to Backus' operator, we have only to show that the functional H is continuous. One interpretation of an arrow $A \longrightarrow D_\perp$ is as a partial arrow from a complemented subobject (see 9.6.2 for the definition) of A to D. This is extended to an arrow that takes the value $\perp$ on the complementary subobject. An arrow $\phi : [A \longrightarrow D] \longrightarrow [A \longrightarrow D]$ preserves order if and only if whenever g extends f, then $\phi(g)$ extends $\phi(f)$. As already mentioned, the order relation is that of extension of domain. So to apply Proposition 6.6.3, we have to show that if f is a restriction of g to a smaller domain, then $H(f)$ is a restriction of $H(g)$. It must also be shown that if f is the least upper bound of the increasing sequence

$$f_0 \leq f_1 \leq \cdots \leq f_n \leq \cdots$$

then $H(f)$ is the least upper bound of

$$H(f_0) \leq H(f_1) \leq \cdots \leq H(f_n) \leq \cdots$$

Now Backus considers the following possible functional forms for H (it is a little hard to apply this directly, since for Backus, there is just one data type, list):

FF–1 $H(f) = r$ (a constant; r is not constant, H is).

FF–2 $H(f) = f_i$ where $f = \langle f_1, \ldots, f_n \rangle$ (here he uses the fact that everything is a list).

FF–3 $H(f) = \Gamma \circ \langle E_1(f), \ldots, E_n(f) \rangle$, where $E_1, \ldots, E_n$ are (simpler) functional forms and Γ is a function.

All three of these rules generate ω-continuous functions on $[D \longrightarrow D]$ and so are covered by Proposition 6.6.3. Of course, there are many other possibilities for H. It must be emphasized that Backus is mainly concerned about convergence and therefore puts other constraints on H, whereas we are dealing with programs.

7
Finite product sketches

A formal theory in mathematical logic is a specification method based on strings of symbols as the formal structure. A signature, with equations, is a specification based on tuples as the formal structure. A sketch is a specification based on graphs as the formal structure. As such, sketches are the intrinsically categorical way of providing a finite specification of a possibly infinite mathematical object or class of models.

In Section 4.6 we described a weak form of sketch, linear sketches, to illustrate the constructions involved. In this chapter we generalize the notion of linear sketch to allow operations with more than one argument. These are called FP (finite product) sketches and are defined in Section 7.1. The example in Section 7.2 shows the connection with signatures. A notation resembling the notation for signatures and equations is developed in Section 7.3, but this notation is intended to be informal. The formal object is the sketch, not the notation. The connection between FP sketches and the method of signatures and equations is made more explicit in Section 7.7 (which, by the way, is not needed in the rest of the book).

The semantics for a sketch are given by certain graph homomorphisms. The nature of FP sketches is such that models can be taken to be in an arbitrary category with finite products. We describe this in Section 7.4.

Each FP sketch generates an FP theory (Section 7.5) which is a category with finite products. A model of an FP sketch in a category that has finite products is a functor that preserves finite products from the theory to the category. The theory is analogous to the formal language of mathematical logic, and the sketch to the recursive rules which define the language.

Besides the functorial semantics described above, an FP sketch has an initial algebra semantics which is described in Section 7.6. We also discuss the construction of free algebras for an FP sketch; the free monoid construction is a special case of this.

The concept of sketch is due to Charles Ehresmann and has been highly developed by his students in France. Their formalism is different from ours and is described in [Bastiani and Ehresmann, 1972] and, using more up-to-date notation, by Coppey and Lair [1984, 1988]. Guitart and Lair [1980] describe the connections with other formalisms. Much of their discussion is updated in [Adámek and Rosický, 1994]. Bibliographies may be found in [Permvall, 1991] and [Wells, 1993].

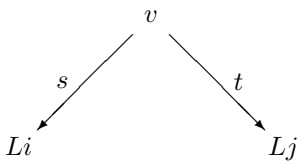
This chapter and Chapters 8 and 10 form a sequence that introduces successively more powerful forms of sketches. The concepts in this sequence are not used in the rest of the book, except that Section 9.7 requires the concept of FP sketch and Section 15.7 requires the sketch for categories given in 10.1.5.

7.1 Finite product sketches

We first define discrete cones, which are used to specify that an object of a sketch is a product.

7.1.1 A **finite discrete cone** in the graph $\mathscr{G}$ consists of a finite discrete graph $\mathscr{I}$, a graph homomorphism $L : \mathscr{I} \longrightarrow \mathscr{G}$, a node n of $\mathscr{G}$ and a collection of arrows $p_i : n \longrightarrow Li$, one for each node $i \in \mathscr{I}$. The node n is called the **vertex** of the cone and the diagram L the **base** of the cone. Since $\mathscr{I}$ is discrete, the base of the cone is an indexed family of nodes. In particular it is possible to have $Li = Lj$ for $i \neq j$.

We will often use other labels for the nodes and arrows. Thus in the discrete cone



(7.1)

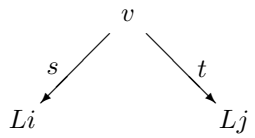
v is the vertex, s is the arrow p_i and t is the arrow p_j. Note that if Li and Lj are the same object but $i \neq j$, s and t are not required to be the same.

> We could have defined cone in a graph the way we defined cone in a category in 5.1.4 and in Section 5.3, in particular saying that the base is indexed by a set instead of saying it is a graph homomorphism from a discrete graph (that is, a discrete diagram). When we study general limits we will introduce cones with arrows between the nodes of the base, so that the base is itself a diagram. Then, the construction here will be seen as a special case – the discrete case. Hence the name 'discrete cone'.

7.1.2 Definition A cone in a *category* is called a **product cone** if it is a product diagram in the category.

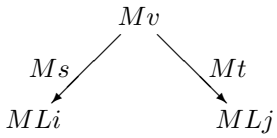
7.1.3 Definition A **finite product sketch** or **FP sketch** $\mathscr{S}$ is a triple $(\mathscr{G}, \mathscr{D}, \mathscr{L})$ where $\mathscr{G}$ is a finite graph, $\mathscr{D}$ a finite set of finite diagrams in $\mathscr{G}$ and $\mathscr{L}$ a finite set of finite discrete cones in $\mathscr{G}$.

7.1.4 Definition Let $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$ be an FP sketch and $\mathscr{C}$ be a category. By a **model** of $\mathscr{S}$ in $\mathscr{C}$, or an $\mathscr{S}$**-algebra** in $\mathscr{C}$, we mean a model M of the linear sketch $(\mathscr{G}, \mathscr{D})$ in $\mathscr{C}$ that has the additional property that for any cone $L : \mathscr{I} \longrightarrow \mathscr{G}$ in $\mathscr{L}$, the composite $M \circ L$ is a product cone. For example if the cone looks like



(7.2)

then our condition requires that



(7.3)

be a product cone in $\mathscr{C}$. If the cone is

$$v$$

(7.4)

meaning that the base is empty, then the cone

$$Mv$$

(7.5)

should also be a product cone, which means precisely that Mv should be a terminal object of $\mathscr{C}$.

When we refer to a model of a sketch without mentioning the category $\mathscr{C}$, we mean a model in **Set**.

7.1.5 A simple example of an FP sketch In 4.7.7, we described a linear sketch with constants for the natural numbers. Here we do the same thing with an FP sketch. The sketch has two nodes we will name 1 and n. There is just one cone, namely the cone with 1 as vertex and empty base.

As we mentioned above, the result is that in any model M, $M(1)$ must be the terminal object of the value category. The graph has just two arrows,

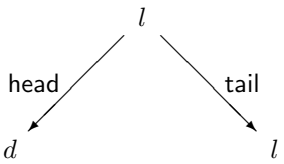
$$\mathsf{zero} : 1 \longrightarrow n \qquad \text{and} \qquad \mathsf{succ} : n \longrightarrow n$$

The sketch has no diagrams.

Since $M(1)$ is terminal in any model, in the category of sets $M(1)$ is a one-element set, so that $M(\mathsf{zero})$ is a function from a one-element set into $M(n)$, i.e. an element of $M(n)$. Note that this observation is independent of which one-element set $M(1)$ actually is. In an arbitrary category, $M(1)$ is terminal, and $M(\mathsf{zero})$ is then a global element or constant (see 2.7.19).

$M(\mathsf{succ})$ is any arrow from $M(n)$ to $M(n)$, so that a model of this sketch in an arbitrary category is an object together with an endoarrow and a constant. Even in the category of sets, there are many models of this sketch that are not the natural numbers. How to pick out those that are will be explored in Section 7.6.

7.1.6 Infinite lists Let $\mathscr{S}$ be the sketch with three nodes, 1, d and l, arrows $a, b : 1 \longrightarrow d$, $\mathsf{head} : l \longrightarrow d$ and $\mathsf{tail} : l \longrightarrow l$, no diagrams, and two cones, one empty with 1 at the vertex and the other C given by

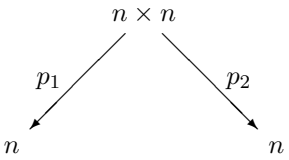


One model of this sketch is the model M with $M(l)$ the set of all infinite sequences of a's and b's. $M(\mathsf{head})$ gives the first entry of a sequence and $M(\mathsf{tail})$ gives the sequence obtained by deleting the first entry of the sequence.

If you try to define a model M of this sketch in which $M(l)$ is the set of finite lists of a's and b's and head and tail have their usual meaning, you will run into trouble eventually. The reason is that if a list is finite, then repeated applications of $M(\mathsf{tail})$ to it will eventually give the empty list, but the empty list has no head, so there is no obvious way to allow it to be in $M(l)$. Various ways of handling this type of error have been suggested in the literature, for example [Goguen, 1978] and [Wells and Barr, 1988] (the latter is described in Section 8.1).

7.1.7 We can introduce addition into the sketch in 7.1.5 by adding a binary operation, using our new ability to specify that an object be a cartesian

product. We need an object $n \times n$ and a diagram



(7.6)

The notation $n \times n$ indicates our intention that the node become a product in a model. It cannot itself be a product, since it is a node in a graph, not an object of a category and products are defined only for categories.

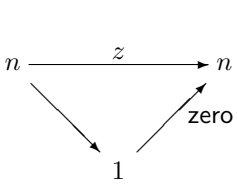
A systematic method for describing this situation is to use the word 'formal': the diagram (7.6) is a **formal product diagram**, meaning that its value in a model must be a product diagram. Similarly, the node v in Diagram (7.4) is a **formal terminal object**.

We will also need new arrows:

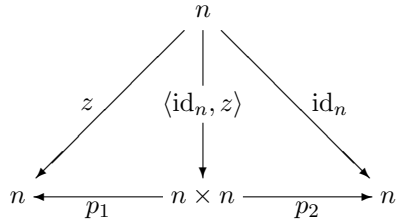
$$\begin{aligned} z &: n \longrightarrow n \\ \mathrm{id}_n &: n \longrightarrow n \\ \langle \mathrm{id}_n, z \rangle &: n \longrightarrow n \times n \\ + &: n \times n \longrightarrow n \\ \mathrm{id}_n \times \mathsf{succ} &: n \times n \longrightarrow n \times n \end{aligned}$$

Definition 7.1.4 will force $M(n \times n)$ to be isomorphic to $M(n) \times M(n)$ for a model M. To simplify the discussion, we assume $M(n \times n)$ is actually identical to $M(n) \times M(n)$. In 7.2.7 and 7.2.8 below we will show how this assumption is avoided.

The following diagrams ensure that this operation will satisfy the inductive definition of addition for integers:

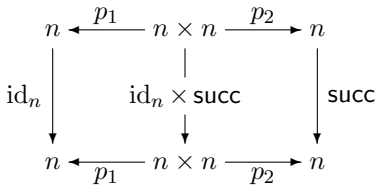


(a)

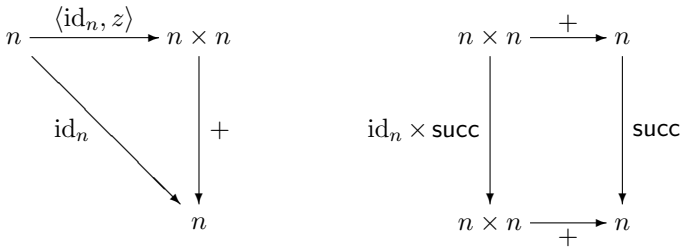


(b)

(c)



(d)



(e) (f)

In a model M, the commutativity of (a) forces the arrow $M(z)$ to be the constant map which takes any element of $M(n)$ to the element determined by zero. The commutativity of (b) forces the center arrow to go to the product suggested by its name: it forces

$$M\langle \mathrm{id}_n, z \rangle = \langle M(\mathrm{id}_n), M(z) \rangle = \langle \mathrm{id}_{M(n)}, M(z) \rangle$$

The commutativity of (c) forces

$$M(\mathrm{id}_n) = \mathrm{id}_{M(n)}$$

and that of (d) forces

$$M(\mathrm{id}_n \times \mathsf{succ}) = \mathrm{id}_{M(n)} \times M(\mathsf{succ})$$

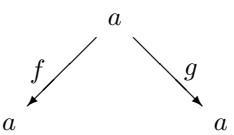
The last two force the inductive definition of addition to be true in a model: $k + M(\mathsf{zero}) = k$ and $k + M(\mathsf{succ})(m) = M(\mathsf{succ})(k + m)$.

7.1.8 A model of the resulting FP sketch that takes n to the set of natural numbers, succ to the successor function and zero to 0 is forced by the diagrams to take $+$ to the usual addition function, simply because the inductive definition determines addition uniquely. However, there is no way, using an FP sketch, to force the model to take n to the natural numbers in the first place. We will look at this again when we consider FD sketches.

7.1.9 Exercises

1. Prove the claim concerning $+$ made in 7.1.8.

2. Let $\mathscr{S}$ be a sketch with one node a and two arrows $f, g : a \longrightarrow a$. It has one cone



Show that in any model M in **Set**, $M(a)$ is either empty or is a singleton set or is infinite, and that the sketch does indeed have an infinite model.

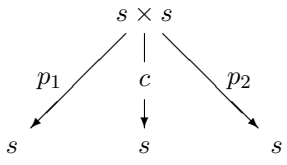
3. Explain how to adjoin a multiplication to Example 7.1.7 so that it becomes the usual multiplication when the model is actually **N**.

7.2 The sketch for semigroups

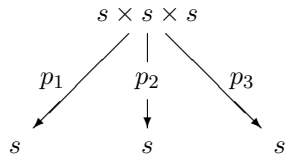
We now develop a substantial example of an FP sketch of a type of mathematical structure, namely the sketch for semigroups. It exhibits many of the issues and subtleties concerning presentation of structures by sketches. The presentation is quite lengthy; later we introduce a notation which makes such presentations shorter and easier to read. In 7.3.2, we describe how to extend this sketch to a sketch for monoids.

7.2.1 We define an FP sketch $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$. The graph $\mathscr{G}$ has three nodes we will give the names s, $s \times s$ and $s \times s \times s$. As in 7.1.7, these three nodes are not, indeed cannot be, products (which are defined only in categories). They are formal products, as the names suggest.

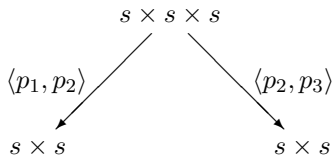
The arrows in the graph are:



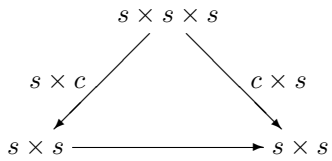
(a)



(b)



(c)



(d)

We have given the graph in four pictures, but it is just one graph with three nodes and ten arrows. Note that there are two arrows labeled p_1 and two labeled p_2. This is overloaded terminology, used by long convention. For $i = 1, 2$, the two arrows named p_i are necessarily different because their domains and codomains are different.

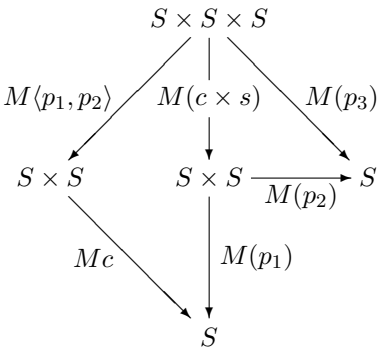
7.2.2 Before giving the details of the diagrams and cones, we pause to explain the intent of this example. Up till now, we have described a graph. What is a model of this graph in the category **Set**? We need sets we call $S = M(s)$, $S^2 = M(s \times s)$ and $S^3 = M(s \times s \times s)$. For the moment, the exponents are simply superscripts. In addition we require functions $M(p_i) : S^2 \to S$, $i = 1,\ 2$, $M(p_i) : S^3 \to S$, $i = 1,\ 2,\ 3$, $M(c) : S^2 \to S$ and $M(s \times c)$ and $M(c \times s)$ from S^3 to S^2. So far, this is nothing familiar, but if we now suppose, as suggested by the notation, that S^2 and S^3 are actually the cartesian square and cube of S and if we make certain subsidiary assumptions given by diagrams to be described later, these data cause S to be a semigroup whose multiplication map is given by $M(c) : S \times S \to S$.

The subsidiary hypotheses are

(i) The various $M(p_i)$ are indeed the projections suggested by the notation.

(ii) $M(\langle p_1, p_2 \rangle) = \langle M(p_1), M(p_2) \rangle$ and similarly for $\langle p_2, p_3 \rangle$.

(iii) $M(c \times s) : S \times S \times S \to S \times S$ is the unique function (guaranteed by the specification for products) for which the diagram

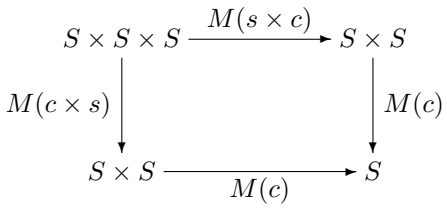


(7.7)

commutes. This diagram merely expresses the fact that $M(c \times s) = M(c) \times \mathrm{id}_S$, which we need to get associativity (in Diagram (d) below).

(iv) A similar diagram expresses the fact that $M(s \times c) = \mathrm{id}_S \times M(c)$.

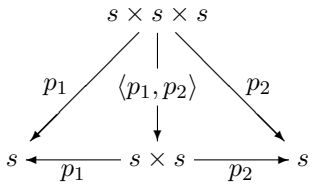
(v) Finally, if we want a semigroup, we must express the associative law of the multiplication. This is done by saying that the diagram



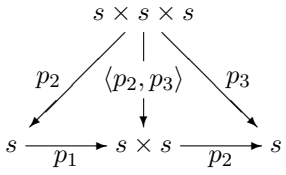
(7.8)

commutes.

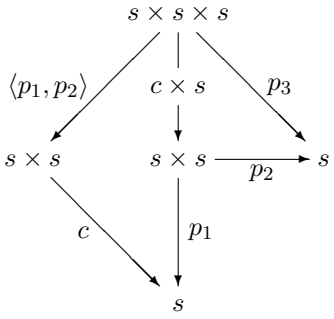
7.2.3 Our task will be to express these requirements in our sketch. This is done as follows. We let $\mathscr{D}$ consist of the following diagrams:



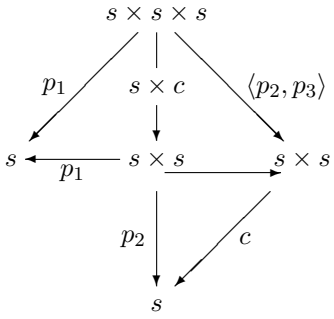
(a)



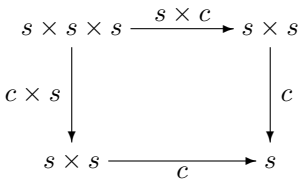
(b)



(c)



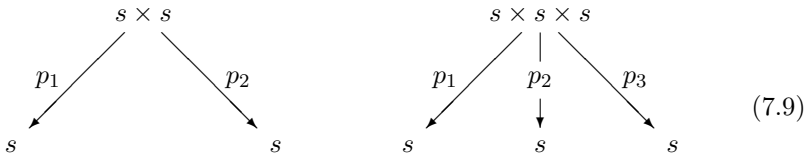
(d)



(e)

These five diagrams have (e) as their main statement; (c) and (d) are needed to define arrows which occur in (e), and (a) and (b) are needed to define arrows which occur in (c) and (d). This construction is reminiscent of the way you construct progressively higher level procedures in Pascal culminating in the procedure which actually does what you want.

7.2.4 The set $\mathscr{L}$ of cones consists of the following:



(7.9)

Then we say that the model M of $\mathscr{S}$ is a model of the sketch if all the diagrams in $\mathscr{D}$ become commutative diagrams when M is applied and if all the cones in $\mathscr{L}$ become product cones. In particular, up to a technicality to be discussed below, diagram (c) in $\mathscr{D}$ becomes (7.7), and diagram (e) becomes (7.8).

7.2.5 Models in Set A model of this FP sketch in **Set** is essentially a semigroup. We will now spell out the precise relationship (which is fairly subtle) between a model of the sketch for semigroups and the usual way of describing a semigroup: a semigroup is a set S with an associative binary operation $m : S \times S \longrightarrow S$. We will discuss models in an arbitrary category in more detail in Section 7.4.4.

7.2.6 Every semigroup determines a model of $\mathscr{S}$ in Set Let S be a semigroup with operation $m : S \times S \longrightarrow S$. We construct a model M of the sketch $\mathscr{S}$ of semigroups developed above. We take $M(s) = S$, $M(s \times s) = S \times S$, and $M(s \times s \times s) = S \times S \times S$. For the arrows of $\mathscr{G}$, we define the following arrows. The diagram numbers refer to 7.2.3.

(i) $M(c) = m$.

(ii) $M(p_i) = p_i : S \times S \longrightarrow S$, for $i = 1, 2$.

(iii) $M(p_i) = p_i : S \times S \times S \longrightarrow S$, for $i = 1, 2, 3$.

(iv) $M\langle p_1, p_2\rangle$ in Diagram 7.2.3(a) is the function which takes (r, t, u) to (r, t). This function, of course, is normally denoted $\langle p_1, p_2\rangle$, which is why the arrow in Diagram 7.2.3(a) above is so labeled.

(v) $M\langle p_2, p_3\rangle$ in Diagram 7.2.3(b) to be the function which takes (r, t, u) to (t, u).

(vi) $M(c \times s)$ in Diagram 7.2.3(c) to be the function $m \times \mathrm{id}_S$ which takes (r, t, u) to (rt, u).

(vii) $M(s \times c)$ in Diagram 7.2.3(d) to be the function $\mathrm{id}_S \times m$ which takes (r, t, u) to (r, tu), i.e., $(r, m(t, u))$.

Since we have required the arrows labeled p_i to be actual product projections, the two cones in the sketch both become product cones in **Set**. Similarly, all five diagrams in 7.2.3 become commutative diagrams; in particular, Diagram 7.2.3(e) is just the associative law.

As an example of how one shows the diagrams become commutative, we will check the commutativity of the right half of Diagram 7.2.3(d). This translates into proving that

$$M(p_2) \circ M(s \times c)(r,t,u) = M(c) \circ M\langle p_2, p_3\rangle(r,t,u)$$

for any triple (r,t,u) of elements of S. Applying the definition of M on arrows, this requires that $p_2(r,tu) = M(c)(t,u)$, that is, that $tu = tu$ since $M(c) = m$, the multiplication of the semigroup. The other verifications are similar.

7.2.7 Every model of $\mathscr{S}$ in Set determines a semigroup If M is a model of $\mathscr{S}$ in **Set**, then $M(c) : M(s^2) \longrightarrow M(s)$ determines a binary operation on $M(s)$, since $M(s^2)$, with the functions $M(p_i) : M(s^2) \longrightarrow M(s)$ for $i = 1, 2$, is a product $M(s) \times M(s)$ in **Set**. (See the discussion in 5.2.6.)

Since M takes every diagram of $\mathscr{S}$ to a commutative diagram in **Set**, Diagram (7.8) commutes, and because diagrams (c) and (d) of 7.2.3 commute, we have that the following diagram commutes. Here we write $S \times S$ and $S \times S \times S$ for $M(s \times s)$ and $M(s \times s \times s)$, respectively. Because of the unique isomorphisms given by Theorem 5.2.2, it does not matter which particular product is used since if the diagram commutes with one, it will commute with any other one (see 5.2.1 and Proposition 5.2.18).

$$\begin{array}{ccc} S \times S \times S & \xrightarrow{M(s) \times M(c)} & S \times S \\ {\scriptstyle M(c) \times M(s)} \downarrow & & \downarrow {\scriptstyle M(c)} \\ S \times S & \xrightarrow[M(c)]{} & S \end{array} \tag{7.10}$$

But this is precisely the associativity of the multiplication.

7.2.8 Subtleties The preceding constructions exhibit the subtle relationship between the usual informal method of describing a mathematical structure and a model of a sketch of the structure.

In the first place, models do not have to take vertices of cones to canonical products; thus $M(s \times s)$ need not be the same as $M(s) \times M(s)$. This was just discussed in connection with Diagram (7.10). In any case, one can regard the map $M(c) : M(s \times s) \longrightarrow M(s)$ as a binary operation on $M(s)$ defined on a

coded form of the cartesian product $M(s) \times M(s)$, as discussed in 5.2.1. The unique isomorphism between $M(s \times s)$ and the actual set of ordered pairs guarantees that the binary operation is uniquely defined on $M(s) \times M(s)$.

A deeper difference is that there are no distinguished nodes or operations in a sketch. The graph of the sketch for semigroups, for example, has three nodes, no one singled out, whereas in the usual definition of semigroup, the underlying set S (corresponding to $M(s)$ in a model of the sketch) is singled out and other things are defined in terms of it. Similarly, in the graph there are various arrows; c is just one of them.

The sketch approach requires you to construct everything you need explicitly from a few basic constants and operations. Consider the traditional definition of semigroup. You are given a set S and a binary operation on S. The set $S \times S$ is regarded as there for you to use in giving the binary operation. Similarly, when you state the associative law you use $S \times S \times S$ (or more traditionally, triples of variables from S) without comment: you *already have it available.*

By contrast, when you use a sketch to describe semigroups, you have to give explicitly everything you use and every property it has. Nothing exists until you construct it, as in lazy evaluation, where a datum is not constructed until it is needed. Thus to define the binary operation, you need a node to become the cartesian product of $M(s)$ with itself, so you put it in the graph and you put in a cone making it a cartesian product. To state the associative law, you need $M(c) \times M(\mathrm{id}_{M(s)})$, so you have to put in the arrow $c \times s$ and Diagram 7.2.3(c); to construct *that*, you need $\langle p_1, p_2 \rangle$ and Diagram 7.2.3(a); and so on.

The result looks much more complicated than the usual way of defining a semigroup, even if you use the elaborate formal notation of signatures and equations. That is because everything you need is constructed from scratch. The sketch is the formal object corresponding to the informal specification for semigroups, and it is appropriate for the formal object to expose all the girders and braces, so to speak. *Sketches are not designed as notation, but as a mathematical structure embodying the formal syntax.* We introduce a usable informal notation in Section 7.3 below which allows the efficient description of sketches. The fact that the girders and braces are exposed is what allows you to define models in arbitrary categories. They would also be necessary ingredients of a computer model of a sketch.

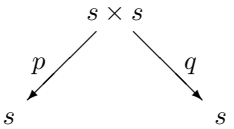
The derived sorts and operations not needed in a sketch are still there potentially, just as they are for signatures. Where they exist is in the theory of the sketch, discussed in Section 7.5 below.

7.2.9 Exercises

1. Let $\mathscr{S}$ be a sketch that has two nodes s and $s \times s$ and three arrows $p, q, c : s \times s \longrightarrow s$. It has one diagram

$$s \times s \underset{p}{\overset{c}{\rightrightarrows}} s$$

and one cone



Show that in a model M in **Set**, $M(c)$ is an associative binary operation on $M(s)$. (Nevertheless, most semigroups are not models of this sketch.) We look at this sketch again in Exercise 1 of Section 7.4.

2. A semigroup S is **idempotent** if for every $x \in S$, $x \cdot x = x$. Add arrows and diagrams to the sketch for semigroups that force the models to be idempotent semigroups.

3.† What modifications to the sketch for semigroups would have to be made so that a model is a real (or complex) vector space? (This sketch is not finite: you will need one unary operation of scalar multiplication for each scalar.)

7.3 Notation for FP sketches

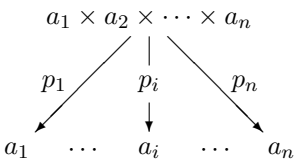
The reader will see that the cones and most of the diagrams used in the sketch for semigroups are required to specify that nodes are products of other nodes and that certain arrows into products are specified by their projections. Accordingly, we will adopt notation that will allow us in most cases to avoid writing down the cones and most of the diagrams. Although these are just notational conventions, they are extremely important for *they are what make the use of sketches feasible.*

In later sections, we will increase the expressive power of sketches in various ways, and extend the list of notations which begins below to cover those cases.

7.3.1 These conventions are as follows:

N–1 It is understood that in all the conventions below, a^n is a shorthand for $a \times a \times \cdots \times a$, with n copies of the node a.

N–2 If the sketch has nodes called $a_1,\ a_2, \ldots, a_n$ and $a_1 \times a_2 \times \cdots \times a_n$, then it is understood that there is a cone

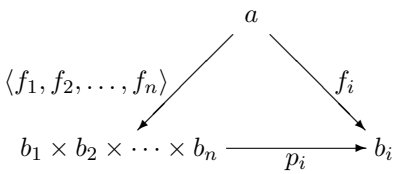


N–3 A node labeled 1 is assumed to imply the existence of a cone whose vertex is that node and whose base is the empty diagram.

N–4 If an arrow has the form

$$\langle f_1, f_2, \ldots, f_n \rangle : a \longrightarrow b_1 \times b_2 \times \cdots \times b_n$$

where $f_i : a \longrightarrow b_i,\ i = 1, \ldots, n$, then there are assumed to be diagrams

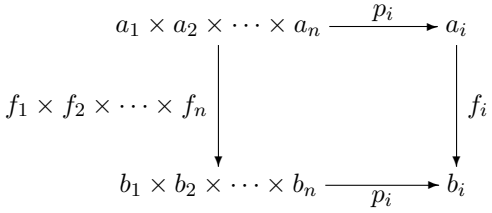


for $i = 1, \ldots, n$.

N–5 If an arrow has the form

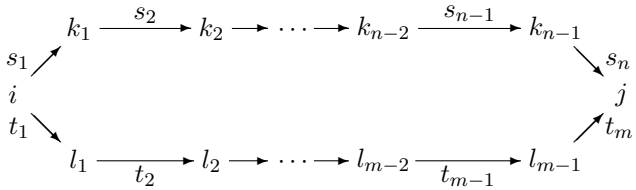
$$f_1 \times f_2 \times \cdots \times f_n : a_1 \times a_2 \times \ldots \times a_n \longrightarrow b_1 \times b_2 \times \cdots \times b_n$$

where $f_i : a_i \longrightarrow b_i,\ i = 1, \ldots, n$ then there are assumed to be diagrams



for $i = 1, \ldots, n$.

N–6 A diagram of the form



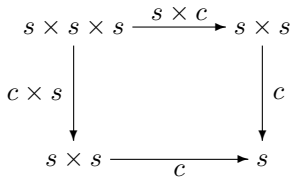
(7.11)

may, if convenient, be specified by the notation

$$s_n \circ s_{n-1} \circ \cdots \circ s_1 = t_m \circ t_{m-1} \circ \cdots \circ t_1$$

The reader should note that this is only a formal equation; it has no meaning except as part of our convention since there is no composition of arrows in graphs.

The result of these notational conventions is that the sketch for semigroups may be specified by the graph as in 7.2.1, plus the single diagram

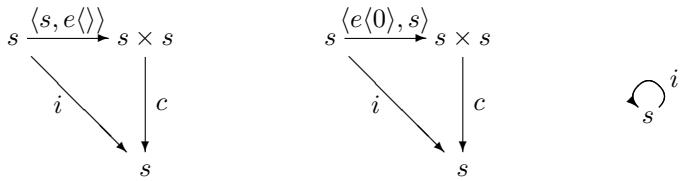


Equivalently, we may specify an equation

$$c \circ (s \times c) = c \circ (c \times s)$$

which is the usual form of the associative law.

7.3.2 There is at least one case in which the proper interpretation of some of these conventions may not be transparent; namely when $n = 0$. For example, we can extend the sketch for semigroups to become a sketch for monoids by adjoining a unary operation $e : 1 \longrightarrow s$ and the following diagrams expressing the identities $ex = xe = x$:



The rightmost diagram is seen to be an example of Diagram (7.11) with $i = j = s$, $m = 0$, $n = 1$ and $s_1 = \mathrm{id}$. We generally use one more notational convention to express this:

N–7 A commutative diagram of the form of Diagram (7.11) with $m = 0$ may, if convenient, be specified by the notation

$$s_n \circ s_{n-1} \circ \cdots \circ s_1 = \mathrm{id}_i$$

7.4 Arrows between models of FP sketches

If M and M' are two models of an FP sketch $\mathscr{S}$ in a category $\mathscr{C}$, a homomorphism $\alpha : M \longrightarrow M'$ is defined to be a natural transformation between $M \longrightarrow M'$, considered as models of the underlying graphs. Just as in 4.6.4, all the models of an FP sketch in $\mathscr{C}$ form a category $\mathbf{Mod}(\mathscr{S}, \mathscr{C})$ which is a full subcategory of the category of all graph homomorphisms from the graph of $\mathscr{S}$ to $\mathscr{C}$. We will discuss the details for models in **Set** first and then consider models in an arbitrary category in 7.4.4.

7.4.1 Let N be the model of the sketch in 7.1.5 for which $N(1) = \{*\}$, $N(n) = \mathbf{N}$ (the set of natural numbers), $N(\mathsf{succ})$ is the function taking n to $n+1$, and $N(\mathsf{zero})$ is the function picking out 0. Let M be the model which is the same for 1, for which $M(n) = \{0, 1, 2, 3\}$, $M(\mathsf{zero})$ is 0, and $M(\mathsf{succ})$ takes 0 to 1, 1 to 2, and both 2 and 3 to 3. Then there is a homomorphism of models $\alpha : N \longrightarrow M$ in which $\alpha n : N(n) \longrightarrow M(n)$ takes 0, 1 and 2 to themselves and all other natural numbers to 3. What $\alpha 1$ does is forced.

The only nontrivial requirement forced by the definition of homomorphism is that for all natural numbers k,

$$M(\mathsf{succ})[\alpha n(k)] = \alpha n[N(\mathsf{succ})(k)]$$

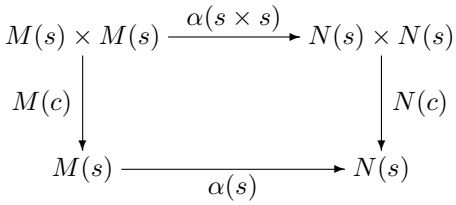
which you should check.

7.4.2 In the case of sketches which specify mathematical structures, these homomorphisms are in most cases essentially the same as homomorphisms as usually defined. For example, we have the following proposition.

7.4.3 Proposition *Let M and N be two models of the sketch $\mathscr{S}$ for semigroups in* **Set** *(so that $M(s)$ and $N(s)$ are semigroups, see 7.2.7). Then if $\alpha : M \longrightarrow N$ is a natural transformation, the component $\alpha s : M(s) \longrightarrow N(s)$ is a semigroup homomorphism (see 2.5.1). Conversely, a semigroup homomorphism $h : M(s) \longrightarrow N(s)$ induces a unique natural transformation $\alpha : M \longrightarrow N$ for which $\alpha s = h$.*

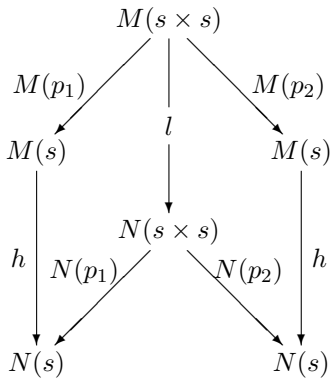
Proof. For convenience, we will suppose that M and N take the cones in the diagram to canonical cones, but this can be avoided as described in 7.2.7. If M and N are models a natural transformation α between the sketches is given by three functions $\alpha(s) : M(s) \longrightarrow N(s)$, $\alpha(s \times s) : M(s \times s) \longrightarrow N(s \times s)$ and $\alpha(s \times s \times s) : M(s \times s \times s) \longrightarrow N(s \times s \times s)$. There is a commutativity

condition imposed for each arrow of the graph. The crucial one says that



commutes. We can now see that the commutativity of the diagram above is exactly the definition of semigroup homomorphism: write h for $\alpha(s)$ and, for $(x, y) \in M(s) \times M(s)$, write xy for $M(c)(x, y)$. Then going south and east in the diagram gives $h(xy)$ and going east and south gives $N(c)[h \times h](x, y) = N(c)(h(x), h(y)) = h(x) \cdot h(y)$.

As for the converse, what must be shown is that a natural transformation $\alpha : M \longrightarrow N$ with a homomorphism h as component at s must have $h \times h$ and $h \times h \times h$ as components at $s \times s$ and $s \times s \times s$ respectively. Let l be the component of α at $s \times s$. Then l must make the following diagram commute simply because α is a natural transformation.

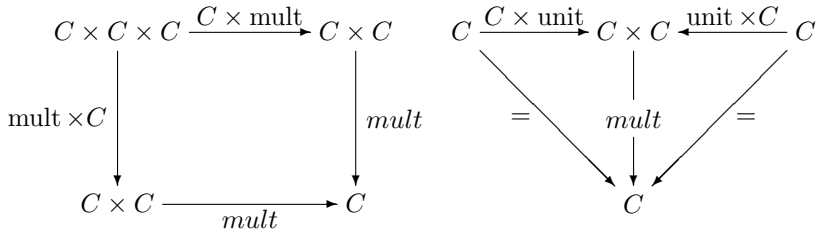


But each $M(p_i)$ and $N(p_i)$ is a product projection in **Set** because models must take cones in $\mathscr{L}$ to product cones. It then follows from 5.2.17 that l is $h \times h$. The proof for $s \times s \times s$ is similar. □

7.4.4 Models of a sketch in an arbitrary category We have defined the concept of model of an FP sketch $\mathscr{S}$ in any category. Here, we will explain in more detail what a model of the sketch for monoids (see Section 7.2 and 7.3.2) in an arbitrary category $\mathscr{C}$ must look like.

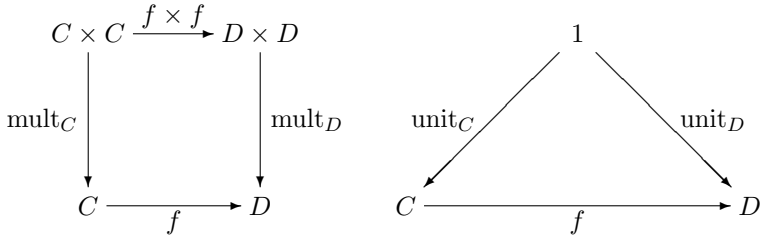
Let M be such a model, and let $C = M(s)$, mult $= M(c)$ and unit $= M(e)$ (e is defined in 7.3.2). It follows that the structure in $\mathscr{C}$ corresponding to M

consists of an object C of $\mathscr{C}$ together with an arrow mult : $C \times C \to C$ of $\mathscr{C}$ and an arrow unit : $1 \to C$ such that the following diagrams commute:



The category $\mathscr{C}$ can be quite arbitrary. Of course, if the category fails to have certain structures, there may be very few models. For example, there can be no models of the monoid sketch if the category lacks a terminal object. Similarly, in order for there to be a model with underlying object C, both $C \times C$ and $C \times C \times C$ must exist.

Suppose that $(C, \text{mult}_C, \text{unit}_C)$ and $(D, \text{mult}_D, \text{unit}_D)$ are two models of the sketch for monoids in the category $\mathscr{C}$. An arrow $f : C \to D$ is a homomorphism of models if the diagrams

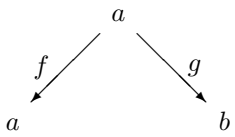


commute. The result is a category **Mon**($\mathscr{C}$) of monoids of $\mathscr{C}$. This category comes equipped with an underlying functor **Mon**($\mathscr{C}$) $\to \mathscr{C}$. In many categories $\mathscr{C}$, there is a functor which behaves like the free monoid functor in **Set** discussed in 13.1.2.

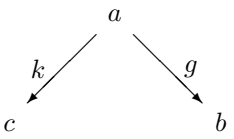
7.4.5 Exercises

1. Let $\mathscr{S}$ be the sketch of Exercise 1 of Section 7.2. Show that the category of models of $\mathscr{S}$ in **Set** is isomorphic to **Set**.

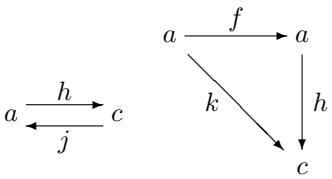
2. Let $\mathscr{S}$ be a sketch with two nodes a and b, two arrows $f : a \to a$ and $g : a \to b$, and one cone



Let $\mathscr{T}$ be a sketch with three nodes a, b and c, five arrows $f : a \to a$, $g : a \to b$, $h : a \to c$, $j : c \to a$ and $k : a \to c$, one cone



and two diagrams



Show that the category of models of $\mathscr{S}$ and of $\mathscr{T}$ in any category $\mathscr{C}$ with finite products are equivalent.

7.5 The theory of an FP sketch

In 4.6.11, we constructed, for each linear sketch $\mathscr{S}$, a category $\mathbf{Th}(\mathscr{S})$ with the property that there is a model $M_0 : \mathscr{S} \to \mathbf{Th}(\mathscr{S})$ and such that for each category $\mathscr{C}$ and each model $M : \mathscr{S} \to \mathscr{C}$, there is a functor $F : \mathbf{Th}(\mathscr{S}) \to \mathscr{C}$ such that $F \circ M_0 = M$. The analog for FP sketches is given by the following.

7.5.1 Theorem *Given any FP sketch $\mathscr{S}$, there is a category $\mathbf{Th_{FP}}(\mathscr{S})$ with finite products and a model $M_0 : \mathscr{S} \to \mathbf{Th_{FP}}(\mathscr{S})$ such that for any model $M : \mathscr{S} \to \mathscr{C}$ into a category with finite products, there is a functor $F : \mathbf{Th_{FP}}(\mathscr{S}) \to \mathscr{C}$ that preserves finite products for which*

(i) *$F \circ M_0 = M$, and*

(ii) *if $F' : \mathbf{Th_{FP}}(\mathscr{S}) \to \mathscr{C}$ is another functor that preserves finite products for which $F' \circ M_0 = M$, then F and F' are naturally isomorphic.*

We do not prove Theorem 7.5.1 here. A proof that constructs $\mathbf{Th_{FP}}(\mathscr{S})$ as a full subcategory of a very special kind of category called a topos (which implies useful properties of $\mathbf{Th_{FP}}(\mathscr{S})$) may be found in [Barr and Wells, 1985], Section 4.3, Theorem 1 (p. 150), where it is also shown that the category of models of $\mathscr{S}$ is equivalent to the category of models of $\mathbf{Th_{FP}}(\mathscr{S})$. Toposes are discussed in Chapter 15. References concerning the construction of theories of sketches of various kinds are given in Section 10.4.

7.5.2 Terminology A category $\mathscr{T}$ with finite products is called an **FP theory**, and a **model** of an FP theory in a category $\mathscr{C}$ which has finite products is a functor $F : \mathscr{T} \longrightarrow \mathscr{C}$ that preserves finite products. Theorem 7.5.1 can be described as saying that every sketch $\mathscr{S}$ has a **universal model** M_0 in an FP theory $\mathbf{Th_{FP}}(\mathscr{S})$ with the property that every model of $\mathscr{S}$ induces a model (uniquely determined up to natural isomorphism) of $\mathbf{Th_{FP}}(\mathscr{S})$.

7.5.3 Theorem *Let $M, M' : \mathscr{S} \longrightarrow \mathscr{C}$ be models of an FP sketch in a category $\mathscr{C}$ with finite products. Then any homomorphism of models $\alpha : M \longrightarrow M'$ extends uniquely to a natural transformation $\widehat{\alpha} : F \longrightarrow F'$, where F and F' are the models of $\mathbf{Th_{FP}}(\mathscr{S})$ in $\mathscr{C}$ given by Theorem 7.5.1 that correspond respectively to M and M'. The passage from M to F and from α to $\widehat{\alpha}$ is a functor, and in fact is an equivalence of categories between $\mathbf{Mod}(\mathscr{S}, \mathscr{C})$ and $\mathbf{Mod}(\mathbf{Th_{FP}}(\mathscr{S}), \mathscr{C})$.*

We omit the proof, but point out that $\widehat{\alpha}$ is induced from α by a generalization of the way in which a semigroup homomorphism from S to S' induces a map from $S \times S$ to $S' \times S'$, as described in the proof of the converse part of Proposition 7.4.3.

7.5.4 The theory $\mathbf{Th_{FP}}(\mathscr{S})$ is defined up to equivalence by the following properties:

FPT–1 $\mathbf{Th_{FP}}\mathscr{S}$ has all finite products.

FPT–2 M_0 takes every diagram of $\mathscr{S}$ to a commutative diagram in $\mathbf{Th_{FP}}\mathscr{S}$.

FPT–3 M_0 takes every cone of $\mathscr{S}$ to a product cone of $\mathbf{Th_{FP}}\mathscr{S}$.

FPT–4 No proper subcategory of $\mathbf{Th_{FP}}\mathscr{S}$ includes the image of M_0 and satisfies FPT–1, FPT–2 and FPT–3.

Two theories of an FP sketch $\mathscr{S}$ need not be isomorphic because, for example, in one of them $A \times B$ might be the same as $B \times A$, whereas in another the two might be different, although, of course, $A \times B$ and $B \times A$ are always isomorphic. As just stated, however, two theories of $\mathscr{S}$ are always *equivalent*. A category theorist normally regards the question as to whether two isomorphic objects are actually the same as irrelevant, and therefore will not usually care which of several equivalent categories she is working in. For this reason we customarily refer to the theory $\mathbf{Th_{FP}}(\mathscr{S})$ as 'the' theory of $\mathscr{S}$.

7.5.5 In general, giving an explicit description of the theory of a sketch is recursively unsolvable, since the general word problem for groups or semigroups can be expressed as the question of whether two arrows in the free category generated by the graph of a sketch generate the same arrow of a theory. However, we can give some explicit examples of theories.

For example, the theory of 7.1.5 has two objects, 1, which is a terminal object, and n. It has an arrow $0 : 1 \longrightarrow n$, an arrow $\langle\rangle : n \longrightarrow 1$, and a family $\mathsf{succ}^k : n \longrightarrow n$ of arrows for $k = 1, 2, \ldots$, which are the n-fold composites of the arrow $\mathsf{succ} : n \longrightarrow n$. And those, with the identity arrows, are all the arrows, and they are all different. With this description, M_0 is an inclusion.

In Section 4.7, we discussed linear sketches with constants. We can now see that these are special types of FP theories, representing each constant as an arrow from an object 1 which must be the vertex of a cone with empty base. We did not discuss the theory of a linear sketch with constants, but it is now clear that such sketches, since they are FP sketches, do have theories.

7.5.6 The theory of the sketch 7.2.1 for semigroups is more complicated. We will again describe it in such a way that the universal model M_0 is an inclusion.

The theory consists of objects s^n for $n = 0, 1, \ldots$. It has as arrows all the projection arrows necessary to make s^n an n-fold product of s, the arrow $c : s^2 \longrightarrow s$ for the binary operation, all the composites of these arrows with each other, and all induced arrows such as $\langle c, c\rangle : s^2 \longrightarrow s^2$, $c \times p_1 : s^3 \longrightarrow s^2$, and so on. Thus it has all the arrows you need to give all the diagrams in 7.2.3 and a great many more. Moreover, all the diagrams in 7.2.3 commute.

It is easy to construct sketches for which M_0 is not injective on arrows (for example, include a diagram forcing two arrows to be the same), so cannot be treated as an inclusion, but this is not true of most interesting sketches – if you have two different arrows in a sketch, you probably already know a model where they are different, and then the universality of M_0 implies that M_0 has to take them to different arrows.

7.6 Initial term models for FP sketches

Just as in the case of a linear sketch with constants (see 4.7.11), a set-valued model of an FP sketch is called a **term model** if each element of the value at each node is forced to be there by the sketch. In the first instance, the node forced to be a terminal object is forced to have a unique element. But then by applying various operations, other elements are forced to exist. A term model has only those elements forced to exist in this way. This is spelled out precisely below in 7.6.5.

7.6.1 Definition A model of an FP sketch in an arbitrary category $\mathscr{C}$ is called an **initial model** or **initial algebra** if it has a unique homomorphism of models to each other model. It is thus an initial object in the category of models of the sketch in $\mathscr{C}$.

7.6.2 A particular example is a natural numbers object in a category $\mathscr{C}$ with terminal object. Using the characterization given in Section 5.5, it is not hard to see that a natural numbers object is an initial model of the sketch given in 7.1.5.

7.6.3 As with linear models with constants, an initial model of an FP sketch in the category of sets is necessarily a term model. For let M_0 be an initial model and suppose it is not a term model. Then the typed set of elements that are reachable beginning from the constants is certainly a model of the theory. It admits the constants and is closed, by definition, under the operations. Thus there is a term model $M_1 \subseteq M_0$. (The same argument, by the way, shows that every model includes a least submodel and that is a term model.)

Now we have an arrow (unique, actually) $f : M_0 \longrightarrow M_1$ by the definition of initial model. That arrow, composed with the inclusion, gives an arrow $M_0 \longrightarrow M_0$. But there is just one arrow from M_0 to itself, the identity (because M_0 is an initial model). Thus the composite must be the identity. But the image of f is included in M_1, so that $M_1 = M_0$, which means that M_0 is a term model, as claimed.

A general version of the preceding argument shows that term models, hence in particular initial models, have no junk in the sense of Example 4.7.9.

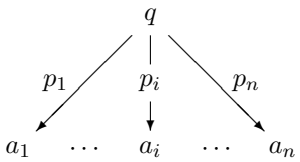
7.6.4 Another property any initial model must have is: if t and u are two terms definable starting with constants, then in an initial model M_0, $M_0(t) = M_0(u)$ if and only if $M(t) = M(u)$ in every model M. The nontrivial direction of that statement follows from the observation that if $M(t) \neq M(u)$ in some model, then necessarily $N(t) \neq N(u)$ in any model N for which there is a homomorphism $N \longrightarrow M$.

Thus initial models have no confusion (see 4.7.10). They are in fact characterized up to isomorphism by having no junk and no confusion (see 4.7.12).

7.6.5 Construction of initial models for finite FP sketches FP sketches always have initial models (see [Barr, 1986b] for a categorical proof). We now revise the construction of 4.7.11 to construct initial models for finite FP sketches.

Let $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$ be an FP sketch. In the construction of the initial model of a linear sketch with constants, we constructed terms as strings of arrows of $\mathscr{G}$. We will now allow tuples of arrows in these strings. The set $A_{\mathscr{S}}$ consisting of all arrows of $\mathscr{G}$, all tuples (of finite length) of such arrows and the cones C of $\mathscr{L}$ is called the **alphabet** of the sketch $\mathscr{S}$. The rules construct an initial model I recursively using these data. The rules apply to

each cone C in $\mathscr{L}$ of the form



(7.12)

FP–1 If $f : a \longrightarrow b$ is an arrow of $\mathscr{G}$ and $[x]$ is an element of $I(a)$, then $[fx] \in I(b)$ and $I(f)[x] = [fx]$.

FP–2 If $(f_1, \ldots, f_m)$ and $(g_1, \ldots, g_k)$ are paths in a diagram in $\mathscr{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \ldots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \ldots \circ Ig_k)[x]$$

in $I(b)$.

FP–3 If for $i = 1, \ldots, n$, $[x_i]$ is an element in $I(a_i)$ (in Diagram (7.12)), then $[C(x_1, \ldots, x_n)]$ is an element of $I(q)$. (Note that $C(x_1, \ldots, x_n)$ is a string consisting of the cone C followed by a tuple of arrows.) In particular, if $n = 0$, there is a single element $[C()]$ in the empty product.

FP–4 If for $i = 1, \ldots, n$, $[x_i]$ and $[y_i]$ are elements in $I(a_i)$ for which $[x_i] = [y_i]$, $i = 1, \ldots, n$, then

$$[C(x_1, \ldots, x_n)] = [C(y_1, \ldots, y_n)]$$

FP–5 For $i = 1, \ldots, n$, we require

$$[p_i C(x_1, \ldots, x_n)] = [x_i]$$

FP–6 For $x \in I(q)$, we require

$$[x] = [C(p_1 x, \ldots, p_n x)]$$

Thus FP–3 forces the vertex of the cone to contain an element representing each tuple of elements in the factors $a_1, \ldots, a_n$, and FP–5 forces the p_i to be the coordinate projections. Because it applies to empty cones, FP–3 subsumes I–1 of 4.7.11, which has no direct counterpart here. Observe that it follows from FP–1 and FP–5 that

$$I(p_i)[C(x_1, \ldots, x_n)] = [x_i]$$

for each i. FP–4 may be regarded as an extension of the definition of 'congruence relation' to cover the case of tuples.

7.6.6 Examples If M is the initial term model for an FP sketch $\mathscr{S}$, then for each sort g of the graph $\mathscr{G}$ of the sketch, $M(g)$ contains just those elements forced to be there by applying operations to constants. Indeed, up to isomorphism of models the elements *are* the formal applications of operations to constants, identifying those which the diagrams force to be the same.

Thus for the sketch in 7.1.5, $I(1)$ must be a singleton set, and $I(\mathsf{zero})$ applied to that single element is an element of $I(n)$ which we may call 0. Then the elements of the sort $I(n)$ are just 0, $\mathsf{succ}(0)$, $\mathsf{succ}(\mathsf{succ}(0))$, $\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(0)))$, and so on. These can be identified as the natural numbers, starting at zero.

From this point of view, 7.1.5 is a simple data type description, and the initial model is then the set of possible values of that type.

Many other data types have been described using initial models using signatures and equations rather than FP sketches. (See the references at the end of this section.) We will present some of these in Section 8.1 after we have introduced finite sums, which allow us to distinguish exceptional cases such as an empty stack or a null node on a tree.

7.6.7 Sketching context free grammars A context free grammar (defined in 4.8.14) is equivalent to a sketch whose initial model consists of derivations; the languages corresponding to each nonterminal form another model. Given a grammar with set T of terminals, N of nonterminals and P of productions, the sketch $\mathscr{S}$ corresponding to the grammar can be defined as follows:

CFG–1 The nodes consist of all the nonterminals and all the terminals (which are *not* formal terminal objects), as well as a formal terminal object 1 and a node r_p for the right hand side of each production p.

CFG–2 The arrows consist of one constant $1 \longrightarrow t$ for each terminal t, an arrow $r_p \longrightarrow V$ for each production $p = V \longrightarrow w$ (the arrow goes the opposite way from the production), and projection arrows for the cones listed below.

CFG–3 There are no diagrams.

CFG–4 There is an empty cone with vertex 1.

CFG–5 For each production $p = V \longrightarrow w$, let $w = a_1 a_2 \cdots a_n$ where each a_i is a terminal or a nonterminal. Then there is a cone with base $\{a_1, a_2, \ldots, a_n\}$ and with vertex r_p.

If $\mathscr{I}$ is the initial algebra of the sketch and V is a nonterminal, then $\mathscr{I}(V)$ consists of all derivations with root V and terminals at the leaves, and if t is a terminal, then $\mathscr{I}(t)$ is a singleton.

In a model $\mathscr{M}$, each production $p = V \longrightarrow a_1 a_2 \cdots a_n$ can be regarded as the n-ary operation that takes a tuple of elements of $\mathscr{M}(a_1) \times \mathscr{M}(a_2) \times \cdots \times \mathscr{M}(a_n)$ to $\mathscr{M}(V)$. In the case of the initial model $\mathscr{I}$ this attaches the trees in $\mathscr{M}(a_i)$ in left to right order as subtrees of a tree in $\mathscr{I}(V)$.

There is another model $\mathscr{L}$ of this sketch. It is the same as $\mathscr{I}$ on terminals. If V is a nonterminal, the elements of $\mathscr{L}(V)$ are the strings of terminals that can be derived from V. (Thus if S is the start symbol, $\mathscr{L}(S)$ is the set of strings usually called the language generated by the grammar.) A production $p = V \longrightarrow a_1 a_2 \cdots a_n$ concatenates the strings in $\mathscr{L}(a_1) \times \mathscr{L}(a_2) \times \cdots \times \mathscr{L}(a_n)$ to produce a string in V. The unique map from $\mathscr{I}$ to $\mathscr{L}$ takes a derivation tree to its string of leaves. This map is an isomorphism if and only if the grammar is unambiguous (this can be taken as the definition of unambiguity).

More information about this construction may be found in [Wells and Barr, 1988].

7.6.8 Free algebras Let $\mathscr{S}$ be a sketch with set S of nodes. Recall from 2.6.11 that an S-indexed set is a set X together with a typing function $\tau : X \longrightarrow S$. Our point of view is that the nodes of the sketch represent types and that X is a set of typed constants. If $\tau : X \longrightarrow S$ and $\tau' : X' \longrightarrow S$ are sets typed by the same sketch $\mathscr{S}$, then a function $f : X \longrightarrow X'$ is a **typed function** if $\tau = \tau' \circ f$. Sets typed by S and typed functions form the slice category $\mathbf{Set}/S$.

A particular example of a typed set is any model M of $\mathscr{S}$ in **Set** for which $M(c)$ and $M(d)$ have no elements in common for distinct nodes c and d. Any model is isomorphic to such a model, obtained by taking the disjoint union of the values of M at the different nodes of $\mathscr{S}$. A model is thus a family of sets, indexed by S, but we can as well think of it as a single set (the union) typed by S. In any case, the underlying (family of) set(s) of a model is an object of the slice category $\mathbf{Set}/S$.

Now given an S-indexed set X, let $\mathscr{S}_X$ be the sketch constructed by adding to the graph of $\mathscr{S}$ a set of arrows $x : 1 \longrightarrow s$ for each element $x \in X$ of type s. These are *in addition* to any constants of type S already given in the sketch. An initial model of $\mathscr{S}_X$, if one exists, is called the **free algebra** generated by the typed set X. We use the definite article because, although not unique, it is unique up to a unique isomorphism that preserves the set X for the same reason that initial algebras are always unique.

The following theorem gives the main existence result.

7.6.9 Theorem *Let $\mathscr{S}$ be an FP sketch. Then for any set X typed by the set of nodes of $\mathscr{S}$, there is a free algebra generated by X.*

The free algebra on X is denoted $F(X)$. This theorem follows from the fact that any FP sketch, in fact any sketch with cones but no cocones (see Chapter 10), has an initial algebra, and of course $\mathscr{S}_X$ is an FP sketch. An accessible proof of this fact is in [Barr, 1986b].

7.6.10 The map lifting property Free algebras have the map lifting property described in 3.1.14. This topic is continued in Section 10.2 and Section 13.2.

7.6.11 Theorem *Let $\mathscr{S}$ be an FP sketch, X a typed set and M a model of $\mathscr{S}$ in sets. Then any typed function $f : X \to M$ has a unique extension to an arrow between models $F(X) \to M$.*

7.6.12 Example Let us see how our definition of free model allows us to discover that the free semigroup on a one element set is isomorphic to the semigroup $(\mathbf{N}^+, +)$.

Let the set be $\{a\}$. Thus we must construct the initial term model I of the sketch for semigroups with one arrow $1 \xrightarrow{a} s$ added; let us denote this sketch by $\mathscr{S}_a$. We write $c(x, y)$ as xy. It follows that $I(s)$ must have elements $[a]$, $[a][a]$, $[a][a][a]$, and so on. Let us call them simply a, a^2, a^3, $\ldots a^n, \ldots$. Now $(\mathbf{N}^+, +)$ is a model M of $\mathscr{S}_a$ with $M(s) = \mathbf{N}^+$, $M(c) = +$ and $M(a) = 0$. ($\mathbf{N}$ is also a model of $\mathscr{S}_a$ in other ways, but this is the way in which it is free on one generator.) Therefore by initiality there must be a unique homomorphism $h : I(s) \to (\mathbf{N}^+, +)$ that takes a to 0. Then h necessarily takes a^n to n for each $n \in \mathbf{N}^+$. Since $a^m a^n = a^{m+n}$ (see 2.3.5), there is also a homomorphism $k : (\mathbf{N}^+, +) \to I(s)$ that takes n to a^n. It follows that h is an isomorphism with inverse k.

Note that it follows from this that there are no elements of $I(s)$ other than those of the form a^n. It is instructive to think about how you would prove this by a direct analysis of the sketch $\mathscr{S}_a$.

7.6.13 Specifying pairs The sketch $\mathscr{S}$ for semigroups has other nodes besides s. For example, one can construct a free algebra F with an element t of type $s \times s$ by constructing the initial algebra for the sketch $\mathscr{S}_t$, where $t : 1 \to s \times s$ is a new arrow adjoined to $\mathscr{S}$. $F(s)$ would necessarily have two elements $a = F(p_1)(t)$ and $b = F(p_2)(t)$, where $p_1, p_2 : s \times s \to s$ are the product projections. On the other hand, if you constructed a free algebra G for $\mathscr{S}$ with two elements a, b of type s, then $G(s \times s)$ would have to have an element $C(a, b)$. This is a reflection of the fact that, as a product, there is a bijection between elements of $G(s \times s)$ (which is $G(s) \times G(s)$) and pairs of elements of $G(s)$. It follows that any free algebra is equivalent to one in which all the elements that are specified are elements of s. A more general form of this observation is used in 9.7.

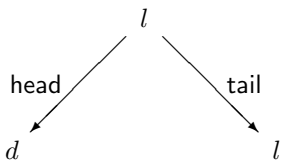
7.6.14 Initial algebra semantics in the literature The initial algebra semantics of multisorted signatures and equations is equivalent to the initial algebra semantics of FP sketches. This approach has been developed extensively by Goguen and others. The idea is to give an algebraic specification of a data type (using a signature or an FP sketch) which explicitly specifies all the behavior of the type accessible to the program writer. The corresponding initial algebra is then an implementation of this specification: the terms in the algebra (arrows) are in fact the data inhabiting that type. The point is that the terms in the initial algebra, as we have seen in 7.6.5, can be constructed entirely from the given description, thus constructing the semantics out of the syntax.

This is discussed in the sketch case by [Gray, 1987] and [Wells and Barr, 1988]. The approach via signatures is described in [Goguen, Thatcher and Wagner, 1978], [Zilles, Lucas and Thatcher, 1982] and [Meseguer and Goguen, 1985]. In many cases, extensions to the idea have been proposed, for example in [Goguen *et al.*, 1977], [Goguen, Jouannaud and Meseguer, 1985] and [Lellahi, 1989]. More references are given at the end of the next section.

7.6.15 Exercises

1. What is the initial model of the sketch of 7.1.6? (Note: It is *not* the model M described in 7.1.6.)

2. Let $\mathscr{S}$ be the sketch with three nodes, 1, d and l, three arrows $u : 1 \to l$, $\mathsf{head} : l \to d$ and $\mathsf{tail} : l \to l$, no diagrams, and two cones, one empty with 1 at the vertex and the other C given by



What is the initial model of $\mathscr{S}$? ($\mathscr{S}$ is the sketch of 7.1.6 without the constants a and b but with a constant u of type l.)

3. Let X be a set with two elements. Explain how the free model of Example 7.1.5 on X can be thought of as the disjoint union of *three* copies of $\mathbf{N}$.

7.7 Signatures and FP sketches

We will describe a formal version of the method of signatures and equations similar to that used by many authors in the literature and show how to produce an FP sketch with essentially the same models in the sense of 7.2.8.

7.7.1 Definition A **signature** $S = (\Sigma, \Omega)$ is a pair of finite sets Σ and Ω. The elements of Σ are **sorts**, which will usually be denoted by lowercase Greek letters. The elements of Σ^* (the set of words of finite length in Σ – including the empty word) are called **arities**. The elements of Ω are **operations**, denoted by lowercase English letters. Each operation $f \in \Omega$ has an arity in Σ^* and a sort in Σ. The arity gives the input type and the sort is the output type.

A **model** M of the signature S is a collection of sets M_σ indexed by Σ and a collection of functions M_f indexed by Ω with the property that if $\sigma_1\sigma_2\ldots\sigma_n$ is the arity of f and τ is the sort of f, then $M_f : M_{\sigma_1} \times M_{\sigma_2} \times \cdots \times M_{\sigma_n} \longrightarrow M_\tau$. When $n = 0$, M_f is interpreted to mean a function from the one-element set to M_τ, that is an element of M_τ.

7.7.2 Terms In general, one is interested in structures defined by a signature which satisfy certain equations. For example, to describe a semigroup, we need a binary operation that sends (x, y) to $x * y$ and we also have to express the associative law that says $x * (y * z) = (x * y) * z$, that is that two terms are equal in any model. To do this in a formal way requires that we define terms for a signature.

Given a set V of variables $x, y, \ldots$ sorted by Σ (meaning that each variable is assigned a specific sort in Σ) we can define a set of **terms** of S recursively by T–1 and T–2 below.

Each term t has an **arity** $\mathbf{A}(t) = w \in \Sigma^*$, a **sort** $\mathbf{S}(t) = \sigma \in \Sigma$, and a **variable list** $\mathbf{VL}(t) = v \in V^*$. T–1 and T–2 define the arity, sort and variable list. Maintaining the information about the arity is redundant (but convenient) since the arity of the term is the list of sorts (in order) of the variables in the variable list and so is recoverable from the variable list. The reverse is not true: the arity does not indicate which variables are repeated.

T–1 If x is a variable of sort σ, then x is a term of arity $\mathbf{A}(x) = `\sigma'$, sort $\mathbf{S}(x) = \sigma$ and variable list $\mathbf{VL}(x) = `x'$.

T–2 If for $i = 1, \ldots, n$, e_i is a term of arity $w_i \in \Sigma^*$, sort $\sigma_i \in \Sigma$ and variable list $l_i \in V^*$, and f is an operation of arity $\sigma_1\sigma_2\ldots\sigma_n$ and sort τ in Ω, then $t = f(e_1, e_2, \ldots, e_n)$ is a term of arity $\mathbf{A}(t) = w_1w_2\cdots w_n$ (this is the concatenate of the *strings* w_i), sort $\mathbf{S}(t) = \tau$ and variable list $\mathbf{VL}(t) = l_1l_2\ldots l_n$.

Terms stand in much the same relation to signatures as arrows of the theory of the sketch do to the graph of the sketch.

7.7.3 Example To give you a clue as to what this definition does, here is an example: if we have a signature S with

(i) sorts σ and τ in Σ,

(ii) operations m of arity '$\sigma\sigma$' and sort τ and f of arity '$\sigma\tau\tau$' and sort σ, and

(iii) variables x, y of sort σ and z of sort τ in V,

then $f(y, m(x, y), z)$ is a term of arity '$\sigma\sigma\sigma\tau$', sort σ and variable list '$yxyz$'.

7.7.4 Equations An **equation** for the signature S is a formula of the form $t = u$, where t and u are terms of the same sort. They do not have to have the same arity or variable list.

An equation has an obvious interpretation in any model of the signature. Such a model M **satisfies** a set E of equations if for every equation $t = u$ in E, t and u have the same interpretations in M. The models for a signature satisfying a set of equations form an **equational variety**.

7.7.5 The sketch corresponding to a signature with equations If S is a signature with equations E, we can construct a sketch $\mathscr{S}(S, E)$ whose models in **Set** which take products to canonical products are exactly the models of (S, E).

To begin with, the graph of $\mathscr{S}(S, E)$ should have a node labeled σ for each sort σ of S, plus a node labeled $\sigma_1 \times \cdots \times \sigma_n$ for each string $\sigma_1 \cdots \sigma_n$ which is the arity of an operation. For each such arity, $\mathscr{S}(S, E)$ must have a cone with vertex $\sigma_1 \times \cdots \times \sigma_n$ and base $\sigma_1, \ldots, \sigma_n$. There must be an arrow with the proper source and target for each operation of S.

The equations in E must be specified by diagrams of $\mathscr{S}(S, E)$, a complicated process which in general also requires adding arrows and nodes to its graph. There are two problems with this construction. The first is that in any equation $t = u$, the variables that appear in t and u are not necessarily the same. The second is that a variable may appear more than once in a term (compare 6.5.2). It is interesting to note that these two possibilities are what distinguish ordinary logic – based on cartesian product – from Girard's linear logic [Girard, 1987], [Girard, Taylor and Lafont, 1989], which is based on another kind of product.

7.7.6 In order to describe how to express the equations in a sketch, we will associate with each term t a path in a certain graph. Since there are infinitely many terms, there will be infinitely many such paths. On the other hand, it is convenient to keep the graph finite. Thus in the sketch associated to a signature with equations, we will add only such nodes and arrows as are necessary to state the given equations. If there are just a finite set of operations and equations, then the resultant sketch will be finite.

In the descriptions below are a number of nodes and arrows which involve formal products, identities and projections. We will suppose without mention that the appropriate cones and diagrams to express the fact that

these become actual products, identities and projections are included in the sketch.

Our approach will be to associate with each term t a 'pseudo-path' $\mathbf{Q}(t)$ that is the same as the path of the term with the same operations but no repeated variables. Then we show how to turn these pseudo-paths into diagrams in a way which takes into account the repeated variables. Note that the actual path of a term t can be recovered by applying this construction to the equation $t = t$, which results in a diagram with two identical paths, each the path of t.

Now let t be a term of sort σ, arity '$\sigma_1 \cdots \sigma_n$' and variable list '$v_1 \cdots v_n$'. We will describe a path $\mathbf{Q}(t)$ that starts at $\sigma_1 \times \cdots \times \sigma_n$ and ends at σ. This is defined inductively as follows. If x is a variable of type τ, then $\mathbf{Q}(x) = \mathrm{id} : \tau \longrightarrow \tau$. If $t_1, \ldots, t_m$ are terms of sort $\tau_1, \ldots, \tau_m$, respectively, and if the concatenate of their arities is '$\sigma_1 \cdots \sigma_n$', and if f is an operation of arity '$\tau_1 \cdots \tau_m$' and sort σ, then $\mathbf{Q}(f(t_1, \ldots, t_m))$ is the composite path

$$\sigma_1 \times \cdots \times \sigma_n \xrightarrow{\mathbf{Q}(t_1) \times \cdots \times \mathbf{Q}(t_m)} \tau_1 \times \cdots \times \tau_m \xrightarrow{f} \sigma$$

Now suppose that we have an equation $t = u$. Suppose that the arities of t and u are '$\sigma_1 \cdots \sigma_n$' and '$\tau_1 \cdots \tau_m$' respectively. Let $X = \{x_1, \ldots, x_k\}$ be the set of variables that appear anywhere in t or u and suppose that x_i has sort ρ_i. The sketch will have an arrow

$$v : \rho_1 \times \cdots \times \rho_k \longrightarrow \sigma_1 \times \cdots \times \sigma_n$$

and, for each $i = 1, \ldots k$, a diagram

$$\begin{array}{ccc} \rho_1 \times \cdots \times \rho_k & \xrightarrow{v} & \sigma_1 \times \cdots \times \sigma_n \\ & \searrow^{\rho_i} & \downarrow{\scriptstyle \rho_i} \\ & & \sigma \end{array}$$

where j is chosen so that the jth variable in the variable list of t is x_i. We similarly have an arrow

$$w : \rho_1 \times \ldots \times \rho_k \longrightarrow \tau_1 \times \ldots \times \tau_m$$

and a similar set of diagrams. Finally, we add the diagram $\mathbf{Q}(t) \circ v = \mathbf{Q}(u) \circ w$ to the set of diagrams. It is this diagram that expresses the equation that $t = u$.

7.7.7 An example The above may appear rather complicated; here is how the construction looks when applied to a specific case. Let the signature S and the operations f and m be as in 7.7.3. Let t be the term $f(y, m(x, y), z)$. Let u be the term x of arity σ and sort 'σ' and variable list 'x'. We will work out the diagram corresponding to the equation $t = u$, that is, $f(y, m(x, y), z) = x$. The path $\mathbf{Q}(t)$ is

$$\sigma \times \sigma \times \sigma \times \tau \xrightarrow{\mathrm{id} \times m \times \mathrm{id}} \sigma \times \tau \times \tau \xrightarrow{f} \sigma$$

The path of u is $\mathrm{id} : \sigma \longrightarrow \sigma$. The variable list of t is the string '$yxyz$' and that of u is the string 'x'. Thus the set of all variables that appear in either list is $\{x, y, t\}$. The sorts of the variables are σ, σ and τ, respectively. Accordingly, we form the diagram

$$\begin{array}{ccc}
\sigma \times \sigma \times \tau & \xrightarrow{v = \langle p_1, p_2, p_1, p_3 \rangle} & \sigma \times \sigma \times \sigma \times \tau \\
 & & \downarrow{\scriptstyle \mathrm{id} \times m \times \mathrm{id}} \\
\downarrow{\scriptstyle w = p_1} & & \sigma \times \tau \times \tau \\
 & & \downarrow{\scriptstyle f} \\
\sigma & \xrightarrow[id]{} & \sigma
\end{array}$$

The top arrow $\langle p_1, p_2, p_1, p_3 \rangle$ is exactly what is needed to transform the variable set $\{x, y, t\}$ into the variable list $xyxt$. Of course a set is not ordered, but our notation forces us to choose an ordering of it. If we had chosen to write the set as $\{t, x, y\}$, then the top arrow would have been

$$\langle p_2, p_3, p_2, p_1 \rangle : \tau \times \sigma \times \sigma \longrightarrow \tau \times \sigma \times \sigma \times \sigma$$

The bottom arrow labeled identity could have been omitted – or rather replaced by the null arrow. However, this would have made the inductive step harder.

In applications it is generally necessary to associate a list of variables with each term or equation, as we did in Section 6.3. This is done in detail in [Bagchi and Wells, 1997b].

7.7.8 FP sketches and multisorted algebraic theories The discussion in this section shows informally that FP sketches have the same expressive power as multisorted algebraic theories, at least as far as models in **Set** are concerned, and the considerable literature on applications of multisorted algebra to computer science provides examples of applications of FP sketches. See in particular [Ehrig and Mahr, 1985], [Wagner, Bloom and

Thatcher, 1985], [Bloom and Wagner, 1985], [Gray, 1987], [Gray, 1990] and [Ehrich, 1986]. Other references, concentrating on initial algebra semantics, were mentioned in 7.6.14. The approach via sketches makes it natural to consider models in an arbitrary category, for example a category such as modest sets designed specifically for programming language semantics. This is an advantage over the usual formulation of multisorted universal algebra, which considers only models in sets.

> One would expect that a reformulation of universal algebra which puts models in other categories on the same footing as models in sets would wind up looking rather like FP sketches. In particular, the formulation would necessarily carry explicit information about the projection arrows of cones.

7.7.9 Exercise

1. A binary operation $*$ is said to **distribute** over a binary operation $+$ if $x * (y + z) = x * y + x * z$ holds for all x, y and z. Express this law using the techniques of this section.

8
Finite discrete sketches

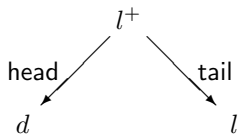
This chapter discusses FD (finite discrete) sketches, which are a more general kind of sketch than that described in Chapter 7. The sketches described here allow the specification of objects which are sums as well as products. Section 8.1 introduces these sketches. Section 8.2 gives a detailed description of the sketch for fields as an example of an FD sketch. In Section 8.3, we show how to modify initial algebra construction so that it works for FD sketches. FD sketches have theories, too, but discussion of that is postponed until Chapter 10.

8.1 Sketches with sums

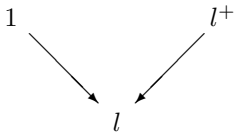
Using sum cocones as well as product cones in a sketch allows one to express alternatives as well as n-ary operations.

8.1.1 A **finite discrete sketch**, or **FD sketch**, $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$, consists of a finite graph $\mathscr{G}$, a finite set of finite diagrams $\mathscr{D}$, a finite set $\mathscr{L}$ of discrete cones with finite bases, and a finite set $\mathscr{K}$ of discrete cocones with finite bases. A **model** of the FD sketch $\mathscr{S}$ in **Set** is a graph homomorphism $M : \mathscr{G} \to \mathbf{Set}$ which takes the diagrams in $\mathscr{D}$ to commutative diagrams, the cones in $\mathscr{L}$ to product cones, and the cocones in $\mathscr{K}$ to sum cocones.

8.1.2 The sketch for lists The FD sketch for finite lists illustrates how FD sketches can be used to specify a data type which has an exceptional case for which an operation is not defined. (Compare 7.1.6.) The graph has nodes 1, d (the data), l (the lists) and l^+ (the nonempty lists). There are no diagrams. There is a cone with empty base expressing the fact that 1 must become a terminal object in a model, another cone



and one cocone



These can be summed up in the expressions $l^+ = d \times l$ (a nonempty list has a datum as head and a list as tail) and $l = 1 + l^+$: (a list is either the empty list – the single member of the terminal set – or a nonempty list). The fact that a sum becomes a disjoint union in a model in **Set** enables us to express alternatives in the way exhibited by this description. We thereby avoid the problem mentioned in 7.1.6: we can define head for only nonempty lists because we have a separate sort for nonempty lists.

If S is any finite set, there is a model M of this sketch for which $M(d) = S$, $M(1)$ is the singleton set containing only the empty list $\langle\rangle$, $M(l)$ is the set of finite lists of elements of S, and $M(l^+)$ is the set of finite nonempty lists of elements of S. $M(\mathsf{head})$ and $M(\mathsf{tail})$ pick out the head and tail of a nonempty list. This is an initial term algebra (to be defined in Section 8.3) for the sketch extended by one constant for each element of S.

8.1.3 Given a finite set S, the model M just mentioned is certainly not the only model of the sketch for which $M(d)$ is S. Another model has $M(l)$ all finite and infinite lists of elements of S, with $M(l^+)$ the nonempty lists and $M(\mathsf{head})$ and $M(\mathsf{tail})$ as before. This model contains elements inaccessible by applying operations to constants: you cannot get infinite lists by iterating operations starting with constants. They are junk as defined in 4.7.9.

8.1.4 The sketch for natural numbers We will now modify the sketch 7.1.5 in a way which shows how FD sketches can model overflow.

The graph contains nodes we will call 1, n, n_{over} and $n + n_{\text{over}}$, which we interpret, respectively, as a one-element set, a set of natural numbers (of which there may be only finitely many), an overflow element and the set of all natural numbers. The only cone is the one implicit in the name 1: the cone has 1 as the vertex and the empty base. There is similarly a (discrete) cocone implicit in the name $n + n_{\text{over}}$. There is a constant operation $\mathsf{zero} : 1 \longrightarrow n$ and a unary operation $\mathsf{succ} : n \longrightarrow n + n_{\text{over}}$. There are no diagrams.

There are many models of this sketch in **Set**. Here are three of them. They are all term models.

8.1.5 The first model is the set of natural numbers. The value of zero is 0 and $\mathsf{succ}(i) = i + 1$. The model takes n_{over} to the empty set. This model illustrates the fact that models are permitted to take the empty set as value

on one or more nodes. Classical model theorists have not usually allowed sorts in a model to be empty.

8.1.6 The second model takes n to the set of integers up to and including an upper bound N. The value at n_{over} is a one- element set we will call ∞, although any other convenient label (including $N+1$) could be used instead. As above, the value of zero is 0. As for succ, it is defined by $\mathsf{succ}(i) = i+1$ for $i < N$, while $\mathsf{succ}(N) = \infty$. Note that the domain of succ does not include ∞.

8.1.7 The third model takes for the value of n also the set of integers up to a fixed bound N. The value of zero is again 0 and of n_{over} is empty. The operation succ is defined by $\mathsf{succ}(i) = i+1$, for $i < N$ while $\mathsf{succ}(N) = 0$. This is arithmetic modulo n.

This third model could be varied by letting $\mathsf{succ}(N) = N$ or, for that matter, any intermediate value. This shows that these models really have at least two parameters: the value of N and what happens to the successor of N.

8.1.8 Finite state machines It is instructive to create an FD sketch for finite state machines as described in Section 3.2.5. We assume a fixed alphabet A and we assume that the states are divided into acceptor states and non-acceptor states; we do not distinguish a start state (thus each state q of the machine determines a language, the language of all strings that drive the machine from state q to an acceptor state). We will produce a sketch whose models include machines with an infinite number of states, so we call its models simply "state machines". Note that all models will have the same alphabet A.

For this purpose we describe a state machine M as a tuple

$$\mathscr{M} = (A, S, \phi, Y)$$

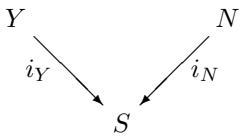
where A is an alphabet, S is a set of states, $\phi : A \times S \to S$ is the transition function, and Y is the subset of A of acceptor states.

The sketch $\mathscr{S}$ for state machines can be described as follows:

SM–1 The graph of $\mathscr{S}$ has three nodes, S, Y and N.

SM–2 The graph has a set of arrows $\phi_a : S \to S$ indexed by the fixed alphabet A. (In a model, ϕ_a will become the curried function $\phi(a, -)$.) In addition there are arrows $i_Y : Y \to S$ and $i_N : N \to S$.

SM–3 There are no cones and one cocone:



It is clear that the models of $\mathscr{S}$ are state machines with alphabet A. There is an uninteresting initial model of $\mathscr{S}$, with empty set of states. Much more interesting is the terminal object in the category of models (the **final algebra**). A state of the final algebra is a subset of A^* (a language in A). If $a \in A$ and $L \subseteq A^*$, then $\phi(a, L) = \{w \in A^* \mid aw \in L\}$. The acceptor states of the final algebra are those languages that contain the empty string. The unique map from a machine $\mathscr{M} = (A, S, \phi, Y)$ to the final algebra takes a state in S to the set of strings in A^* that drive the machine to an acceptor state (the language determined by the state). This map is injective if and only if $\mathscr{M}$ is a minimal machine in the sense of the Myhill-Nerode theorem [Lewis and Papadimitriou, 1981], page 96.

This example is based on [Rutten, 1996]; [Rutten, 1998]. Those papers describe the analog of induction for final algebras (see 4.7.8), which is called the principle of coinduction. It is based on the fact that a final object in a category has no proper quotients.

8.1.9 Exercises

1. Construct an FD sketch whose only model is the two-element Boolean algebra.

2. Give an example of a sketch which has no models in **Set**. (Hint: In **Set**, $1 + 1 \neq 1$.)

8.2 The sketch for fields

Another example of an FD sketch is the sketch for the mathematical structure known as a **field**. The concept of field abstracts the properties of the arithmetic of numbers. We will describe it in some detail here. This section is used later only as an example in Section 8.3. Rather than define 'field' we will describe the sketch and say that a field is a model of the sketch in sets.

The nodes are 1, u, f, $f \times f$ and $f \times f \times f$. There are operations

FO–1 $0 : 1 \to f$.

FO–2 $1 : 1 \to u$.

FO–3 $+ : f \times f \to f$.

FO–4 $* : f \times f \to f$.

FO–5 $- : f \to f$.

FO–6 $(\ )^{-1} : u \to u$.

FO–7 $j : u \to f$.

The reader should note that the two 1's in FO–2 above are, of course, different. One of them is the name of a node and the other of an operation. This would normally be considered inexcusable. The reasons we do it anyway are (a) each usage is hallowed by long tradition and (b) because they are of different type, there is never any possibility of actual clash. The reader may think of it as an early example of overloading.

The diagrams are:

FE–1 (associativity of $+$): $+ \circ (\mathrm{id} \times +) = + \circ (+ \times \mathrm{id}) : f \times f \times f \to f$.

FE–2 (associativity of $*$): $* \circ (\mathrm{id} \times *) = * \circ (* \times \mathrm{id}) : f \times f \times f \to f$.

FE–3 (commutativity of $+$): $+ \circ \langle p_2, p_1 \rangle = + : f \times f \to f$.

FE–4 (commutativity of $*$): $* \circ \langle p_2, p_1 \rangle = * : f \times f \to f$.

FE–5 (additive unit): $+ \circ \langle \mathrm{id}, 0 \circ \langle\rangle \rangle = + \circ \langle 0 \circ \langle\rangle, \mathrm{id} \rangle = \mathrm{id} : f \to f$.

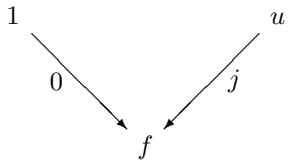
FE–6 (multiplicative unit): $* \circ \langle j, j \circ 1 \circ \langle\rangle \rangle = * \circ \langle j \circ 1 \circ \langle\rangle, j \rangle = j : u \to f$.

FE–7 (additive inverse): $+ \circ \langle \mathrm{id}, - \rangle = + \circ \langle -, \mathrm{id} \rangle = 0 \circ \langle\rangle : f \to f$.

FE–8 (multiplicative inverse): $* \circ (j \times j) \circ \langle \mathrm{id}, ()^{-1} \rangle = * \circ (j \times j) \circ \langle ()^{-1}, \mathrm{id} \rangle = 1 \circ \langle\rangle : u \to u$.

FE–9 (distributive): $+ \circ (* \times *) \circ \langle p_1, p_2, p_1, p_3 \rangle = * \circ (\mathrm{id} \times +)$;
$+ \circ (* \times *) \circ \langle p_1, p_3, p_2, p_3 \rangle = * \circ (+ \times \mathrm{id}) : f \times f \times f \to f$.

There are cones defined implicitly and one cocone:



Intuitively, this says that each element of a field is either zero or else is an element of u. u is interpreted as the set of multiplicatively invertible elements in a model.

8.2.1 An example of a field is the set of ordinary rational numbers. Expressed as a model M of the sketch, the numbers $M(0)$ and $M(1)$ are the usual ones, $M(f)$ and $M(u)$ are the sets of all rationals and nonzero rationals, respectively, and $M(j)$ is the inclusion. The arithmetic operations are the usual ones. Other familiar examples are the real and complex numbers.

8.2.2 In building a model of this sketch, we would have to start with the elements 0 and 1 and then begin adding and multiplying them to get new elements. For example, we could let $2 = 1+1$. Now since $M(f) = M(u)+\{0\}$, we have to decide whether $2 \in M(u)$ or $2 \in \{0\}$, that is, whether or not $2 = 0$. Quite possibly, the only fields the reader has seen have the property that $2 \neq 0$. Even so, one cannot exclude out of hand the possibility that $2 = 0$ and it is in fact possible: there is a model of this sketch whose value at f is $\{0, 1\}$ with the values of the operations being given by the tables:

$+$	0	1
0	0	1
1	1	0

$*$	0	1
0	0	0
1	0	1

(These tables are addition and multiplication (mod 2).) If, on the other hand, $2 \neq 0$, we can form $3 = 1 + 2$. The same question arises: is $3 = 0$ or not? It is not hard to write down a field in which $3 = 0$.

Suppose that neither 2 nor 3 is 0? We would define $4 = 1 + 3$ (and prove, by the way, that $2+2 = 2*2 = 4$) and ask whether $4 = 0$. It turns out that if $2 \neq 0$ then $4 \neq 0$. In fact, it is not hard to show that any product of elements of u also lies in u. Thus the first instance of an integer being 0 must happen at a prime. But, of course, it need never happen, since no number gotten by adding 1 to itself a number of times is zero in the rational, real or complex field.

8.2.3 A homomorphism between fields must preserve all operations. Since it preserves the operation $(\ )^{-1}$, no nonzero element can be taken to zero. Consequently, two distinct elements cannot go to the same element since their difference would be sent to zero. Thus field homomorphisms are injective.

From this, it follows that there can be no field that has a homomorphism both to the field of two elements and to the rationals, since in the first $1 + 1 = 0$, which is not true in the second. This implies that the category of fields and field homomorphisms does not have products.

8.2.4 Exercises

1. Prove that in a field, if two elements each have multiplicative inverses, then so does their product. Deduce that if $4 = 0$, then $2 = 0$.

2. Prove that the next to last sentence of 8.2.3 implies that the category of fields and field homomorphisms does not have products.

8.3 Term algebras for FD sketches

A complication arises in trying to extend the construction of initial term algebras to FD sketches. As we see from the examples of natural numbers and fields, an operation taking values in the vertex of a discrete cocone forces us to choose in which summand the result of any operation shall be. The choice, in general, leads to nonisomorphic term models which are nevertheless initial in a more general sense which we will make precise. Diers [1980a], [1980b] investigates these models (not in the language of sketches) as generalized adjoints.

8.3.1 Dæmons How to make the choice? Clearly there is no systematic way. One way of dealing with the problem is to take *all* choices, or at least to explore all choices. In the example in Section 8.2 of fields, we mentioned that not all choices are possible; once $2 \neq 0$, it followed that also $4 \neq 0$. (Recall that in that example, saying that something is zero is saying that it is in one of two summands.)

More generally, suppose we have an FD sketch and there is an operation $s : a \longrightarrow b$ and a cocone expressing $b = b_1 + b_2 + \ldots + b_n$. If we are building a model M of this sketch and we have an element $x \in M(a)$, then $M(s)(x) \in M(b)$, which means that we must have a unique i between 1 and n for which $M(s)(x) \in M(b_i)$. (For simplicity, this notation assumes that $M(b)$ is the actual union of the $M(b_i)$.)

Now it may happen that there is some equation that forces it to be in one rather than another summand, but in general there is no such indication. For example, the result of a push operation on a stack may or may not be an overflow, depending on the capacity of the machine or other considerations. Which one it is determines the particular term model we construct and it is these choices that determine which term model we will get.

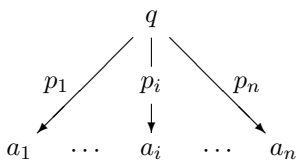
Our solution is basically to try all possible sequences of choices; some such sequences will result in a model and others will abort. Thus as we explore all choices, some will eventually lead to a model; some will not. The theoretical tool we use to carry out this choice we call a **dæmon**. Just as a Maxwell Dæmon chooses, for each molecule of a gas, whether it goes into one chamber or another, our dæmon chooses, for each term of a model, which summand it goes into. The following description spells this out precisely.

8.3.2 Definition Let $\mathscr{S}$ be an FD sketch and suppose the maximum number of nodes in the base of any cocone is κ. A **dæmon** for $\mathscr{S}$ is a function d from the set of all strings in the alphabet $A_{\mathscr{S}}$ (see 7.6.5) of the underlying FP sketch (in other words, forget the cocones) to the initial segment $\{1..\kappa\}$ of the positive integers.

8.3.3 We will use a dæmon this way. We assume that the nodes in the base of each cocone of $\mathscr{S}$ are indexed by $1, 2, \ldots, k$ where $k \leq \kappa$ is the number of nodes in that cocone. In constructing an initial algebra, if a string w must be in a sort which is the vertex of a cocone (hence in the model it must be the disjoint union of no more than κ sorts), we will choose to put it in the $D(w)$th summand. If $D(w) > k$, the construction aborts. We will make this formal.

8.3.4 Construction of initial term models for FD sketches This construction includes the processes in 4.7.11 and 7.6.5; we repeat them here modified to include the effects of a dæmon D. The alphabet is the same as in 7.6.5.

If a node b is the vertex of a cocone with $k = k(b)$ summands, the summands will be systematically denoted $b^1, \ldots, b^k$ and the inclusion arrows $u^i : b^i \longrightarrow b$ for $i = 1, \ldots, k$. If b is not the vertex of a cocone, then we take $k(b) = 1$, $b^1 = b$ and $u^1 = \mathrm{id}_b$. We denote the congruence relation by $\sim$ and the congruence class containing the element x by $[x]$. Rules FD–3 through FD–6 refer to a cone C in $\mathscr{L}$ of the form:



FD–1 If $u^i : a^i \longrightarrow a$ is an inclusion in a cocone and $[x] \in I(a^i)$, then $[u^i x] \in I(a)$ and $I(u^i)[x] = [u^i x]$. (Thus we ignore the wishes of the dæmon in this case.)

FD–2 Suppose $f : a \longrightarrow b$ in $\mathscr{G}$, f is not an arrow of the form u^i, and $[x] \in I(a)$. Let $j = D(fx)$ (we ask the dæmon what to do). If $j > k(b)$, the construction aborts. Otherwise, we let $[fx] \in I(b^j)$ and $I(f)[x] = [u^j fx]$.

FD–3 For $i = 1, \ldots, n$, let $[x_i]$ be a term in $I(a_i)$. Let

$$j = D(C(x_1, \ldots, x_n))$$

If $j > k(q)$, the construction aborts. If not, put $[C(x_1, \ldots, x_n)]$ in $I(q^j)$.

FD–4 If for $i = 1, \ldots, n$, $[x_i]$ and $[y_i]$ are elements in $I(a_i)$ for which $[x_i] = [y_i]$, $i = 1, \ldots, n$, then

$$[C(x_1, \ldots, x_n)] = [C(y_1, \ldots, y_n)]$$

FD–5 For $i = 1, \ldots, n$, $I(p_i)([C(x_1, \ldots, x_n)]) = [x_i]$.

FD–6 For $x \in I(q)$,

$$[x] = [C(p_1 x, \ldots, p_n x)]$$

FD–7 If $\langle f_1, \ldots, f_m \rangle$ and $\langle g_1, \ldots, g_k \rangle$ are paths in a diagram in $\mathscr{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \ldots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \ldots \circ Ig_k)[x]$$

in $I(b)$. If

$$D(f_1 f_2 \ldots f_m x) \neq D(g_1 \ldots g_k x)$$

(causing $(If_1 \circ If_2 \circ \ldots \circ If_m)[x]$ and $(Ig_1 \circ Ig_2 \circ \ldots \circ Ig_k)[x]$ to be in two different summands of $I(b)$), then the construction aborts.

This construction gives a term model if it does not abort. It is an initial model for only part of the category of models, however. To make this precise, we recall the definition of connected component from 4.3.11. It is easy to see that each connected component is a full subcategory of the whole category of models.

8.3.5 Proposition *For each dæmon for which the construction in FD–1 to FD–6 does not abort, the construction is a recursive definition of a model I of $\mathscr{S}$. Each such model is the initial model of a connected component of the category of models of $\mathscr{S}$, and there is a dæmon giving the initial model for each connected component.*

We will not prove this theorem here. However, we will indicate how each model determines a dæmon which produces the initial model for its component. Let M be a model of an FD sketch $\mathscr{S}$. Every string w which determines an element of a sort $I(a)$ in a term model as constructed above corresponds to an element of $M(a)$. That element must be in a unique summand of a; if it is the ith summand, then define $D(w) = i$. On strings not used in the construction of the term models, define $D(w) = 1$, not that it matters.

Our definition of dæmon shows that one can attempt a construction of an initial model without already knowing models. In concrete cases, of course, it will often be possible to characterize which choices give initial models and which do not.

8.3.6 Confusion maybe, junk no The slogan, 'No junk, no confusion' (see 4.7.12 and 7.6.4) is only half true of the initial models for FD theories. The 'No junk' half of the slogan expresses exactly what we mean when we say that every element is reachable. There are no extraneous elements. 'No confusion' means no relations except those forced by the equations in the

theory. As we will show by example it may happen that some initial models have confusion and others not. Later we give an example of a sketch that has more than one unconfused initial model and one that has no unconfused initial (or noninitial) model.

If there is just one unconfused initial model, that one may be thought of as a 'generic' model. The others remain nonetheless interesting. In fact, it is likely that the generic model is the one that cannot be accurately modeled on a real machine.

8.3.7 Example A typical example of a sketch with many initial models is the sketch for natural numbers with overflow. The generic model is easily seen to be the one in 8.1.5 in which the overflow state is empty. The models with overflow in 8.1.6 are all initial algebras for some component of the category of models, but they have confusion, since nothing in the sketch implies that the successor of any element can be the same element. None of these models have junk.

The modular arithmetic models of 8.1.7 are not initial models; in fact they are all in the same component as the natural numbers since the remainder map $(\mathrm{mod}\, N)$ is a morphism of models. They also have no junk.

8.3.8 Example Here is a simple sketch with no generic model. It has two initial models, each satisfying an equation the other one does not. There are five nodes $a = b + c$, d and 1. There is one constant x of type d, and a single operation $s : d \longrightarrow a$. The initial models have one element – the constant – of type d. One of the initial models has an element of type b and the other an element of type c.

By modifying this example, we can get forced confusion. Add constants y and z of type b and c, respectively, and cones forcing b and c to be terminal. Now there are two initial models, one in which $s(x) = y$ and another in which $s(x) = z$. Since there is a model in which $s(x) \neq y$, there can be no equation that forces $s(x) = y$ and there is similarly no equation that forces $s(x) = z$. But one or the other equation must hold in any model.

8.3.9 Example It is well known and proved in abstract algebra texts that the initial fields are (a) the rational numbers and (b) the integers mod p for each prime p. (The word for initial model in these texts is 'prime field'.) A field is in the component of the integers mod p if and only if $1 + 1 + \ldots + 1$ (sum of p 1's) is zero. These fields have confusion. Otherwise the field is in the component of the rational numbers, which have no confusion (nor junk). Duval and Reynaud [1994a, 1994b] show how to implement simultaneous computation in the initial algebras for a finite discrete sketch for fields.

The real numbers and the complex numbers form fields with the usual operations. The irrational real numbers constitute junk.

8.3.10 Example The example in 8.1.2 has only one component and hence a single initial model in which all sorts are empty except the singleton 1. If you add constants to d, the initial model is just the set of lists of finite length of elements of d. In the model discussed there which also has all infinite sequences, the infinite sequences are junk.

8.3.11 Binary trees We now describe a sketch for ordered rooted binary trees (called trees in this discussion). 'Binary' means that each node has either no children or two children, and 'ordered' means that the children are designated left and right. This is an example which uses cocones to treat exceptional cases, in this case the empty tree.

Trees are parametrized by the type of data that are stored in them. We will say nothing about this type of data, supposing only that it is a type for which there is an initial model. The way in which the parametrized data type is filled in with a real one is described, for example, in Section 11.2. We would like to thank Adam D. Barr for helpful discussions on how binary trees operate (especially their error states) on real machines.

The sketch will have sorts 1, t, s, d. Informally, t stands for tree, s for nonempty tree and d for datum. We have the following operations:

$\mathsf{empty} : 1 \longrightarrow t$

$\mathsf{incl} : s \longrightarrow t$

$\mathsf{val} : s \longrightarrow d$

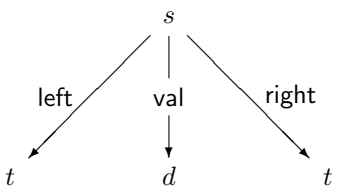
$\mathsf{left} : s \longrightarrow t$

$\mathsf{right} : s \longrightarrow t$

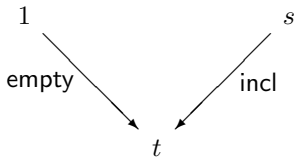
The intended meaning of these operations is as follows.

Empty$\langle\rangle$ is the empty tree; incl is the inclusion of the set of nonempty trees in the set of trees; $\mathsf{val}(S)$ is the datum stored at the root of S; $\mathsf{left}(S)$ and $\mathsf{right}(S)$ are the right and left branches (possibly empty) of the nonempty tree S, respectively.

We require that



be a cone and that



be a cocone.

There are no diagrams.

The cocone says that every tree is either empty or nonempty. This cocone could be alternatively expressed $t = s + \{\mathsf{empty}\}$. The cone says that every nonempty tree can be represented uniquely in the form of a triplet $(\mathsf{left}(S), \mathsf{val}(S), \mathsf{right}(S))$, and that every such triplet corresponds to a tree. Note that this implies that left, val and right become coordinate projections in a model.

Using this, we can define subsidiary operations on trees. For example, we can define an operation of left attachment, $\mathsf{lat} : t \times s \longrightarrow s$ by letting

$$\mathsf{lat}(T, (\mathsf{left}(S), \mathsf{val}(S), \mathsf{right}(S))) = (T, \mathsf{val}(S), \mathsf{right}(S))$$

This can be done without elements: lat is defined in any model as the unique arrow making the following diagram commute (note that the horizontal arrows are isomorphisms):

$$\begin{array}{ccc} M(t) \times M(s) & \xrightarrow{M(t) \times \langle \mathsf{left}, \mathsf{val}, \mathsf{right} \rangle} & M(t) \times M(t) \times M(d) \times M(t) \\ \Big\downarrow{\scriptstyle \mathsf{lat}} & & \Big\downarrow{\scriptstyle \langle p_1, p_3, p_4 \rangle} \\ M(s) & \xrightarrow[\langle \mathsf{left}, \mathsf{val}, \mathsf{right} \rangle]{} & M(t) \times M(d) \times M(t) \end{array}$$

In a similar way, we can define right attachment as well as the insertion of a datum at the root node as operations definable in any tree. These operations are implicit in the sketch in the sense that they occur as arrows in the theory generated by the sketch (see 4.6.11), and therefore are present in every model.

8.3.12 Proposition *Supposing there is an initial algebra for the data type, then the category of binary trees of that type has an initial algebra. If the data type has (up to isomorphism) a unique initial algebra, then so does the corresponding category of binary trees.*

Proof. We construct the initial algebra recursively according to the rules:

(i) The empty set is a tree;

(ii) If T_l and T_r are trees and D is element of the initial term algebra for the data type, then (T_l, D, T_r) is a nonempty tree;

(iii) Nothing else is a tree.

This is a model M_0 defined by letting $M_0(s)$ be the set of nonempty trees, $M_0(t) = M_0(s) + \{\emptyset\}$ and $M_0(d)$ be the initial model of the data type. Here '+' denotes disjoint union. It is clear how to define the operations of the sketch in such a way that this becomes a model of the sketch.

Now let M be any model with the property that $M(d)$ is a model for the data type. Then there is a unique morphism $f(d) : M_0(d) \longrightarrow M(d)$ that preserves all the operations in the data type. We also define $f(t)\{\emptyset\}$ to be the value of $M(\mathsf{empty}) : 1 \longrightarrow M(t)$. Finally, we define

$$f(s)((T_l, D, T_r)) = (f(t)(T_l), f(d)(D), f(t)(T_r))$$

where $f(t)$ is defined recursively to agree with $f(s)$ on nonempty trees. It is immediately clear that this is a morphism of models and is unique. In particular, if the data type has, up to isomorphism, only one initial model then M_0 is also unique up to isomorphism. □

8.3.13 In Pascal textbooks a definition for a tree type typically looks like this:

```
type TreePtr = ^Tree;
     Tree = record LeftTree, RightTree : TreePtr;
         Datum : integer
     end;
```

Note that from the point of view of the preceding sketch, this actually defines nonempty trees. The empty tree is referred to by a null pointer. This takes advantage of the fact that in such languages defining a pointer to a type D actually defines a pointer to what is in effect a variant record (union structure) which is either of type D or of 'type' null.

8.3.14 Exercise

1.† **a.** Show that if $\mathscr{S}$ is an FD sketch and $f : M \longrightarrow N$ is a homomorphism between models in the category of sets, then the image of f is a submodel of N.

b. Show that every model in the category of sets of an FD sketch has a smallest submodel.

9
Limits and colimits

A limit is the categorical version of the concept of an equationally defined subset of a product. For example, a circle of radius 1 is the subset of $\mathbf{R} \times \mathbf{R}$ ($\mathbf{R}$ is the set of real numbers) satisfying the equation $x^2 + y^2 = 1$. Another example is the set of pairs of arrows in a category for which the target of the first is the source of the second: this is the set of pairs for which the composition operation is defined.

A different kind of example is division of one integer by another, which requires that the second argument be nonzero. This can be made an equational condition by building in a Boolean type and a test; the equation then becomes $[y = 0] = \text{false}$. (This can also be handled using finite sums: see Section 5.7 and Exercise 1 of Section 8.2.)

A colimit is similarly the categorical version of a quotient of a sum by an equivalence relation. The quotient category constructed in Section 3.5 is an example of a colimit in the category of categories.

This chapter discusses finite limits and colimits in detail, concentrating on certain useful special cases. Infinite limits and colimits are widely used in mathematics, but they are conceptually similar to the finite case, and are not described here.

Sections 9.1, 9.2 and 9.3 discuss limits. Sections 9.4 and 9.5 describe colimits. Section 9.6 describes certain properties of sums which are desirable in programming language semantics. Section 9.7 shows how unification (as used in logic programming) can be described as a kind of colimit. Section 9.8 describes some of the interaction between limits, colimits and factorizations. Section 9.7 is not used in the rest of the book, and Section 9.8 is used only in Chapter 16.

Limits and colimits are used in all the remaining chapters (but Chapter 12 requires only pullbacks). In particular, they are used in Chapter 10 to describe more expressive types of sketches.

This chapter may be read right after Chapter 5, except that Section 9.7 requires familiarity with FP sketches (Section 7.1).

9.1 Equalizers

If S and T are sets and $f, g : S \longrightarrow T$ are functions, it is a familiar fact that we can form the subset $\mathrm{Eq}(f, g) \subseteq S$ consisting of all elements $s \in S$ for which $f(s) = g(s)$. This concept can be made into a categorical concept by changing it to a specification which turns out to determine a subobject. We will look more closely at the construction in **Set** to see how to make the general categorical construction.

Suppose that $j : \mathrm{Eq}(f, g) \longrightarrow S$ is the inclusion function: $j(u) = u$ for $u \in \mathrm{Eq}(f, g)$. Let $h : V \longrightarrow S$ be a function. Then h factors through $j : \mathrm{Eq}(f, g) \longrightarrow S$ if and only if the image of the function h lies in the subset $\mathrm{Eq}(f, g)$ (see 2.8.9). This is the key to the categorical specification of $\mathrm{Eq}(f, g)$, Definition 9.1.2 below.

9.1.1 Definition Two arrows $f : A \longrightarrow B$ and $g : A \longrightarrow B$ of a category (having the same domain and the same codomain) are a **parallel pair** of arrows. An arrow h with the property that $f \circ h = g \circ h$ is said to **equalize** f and g.

9.1.2 Definition Let $\mathscr{C}$ be a category and $f, g : A \longrightarrow B$ be a parallel pair of arrows. An **equalizer** of f and g is an object E *together with* an arrow $j : E \longrightarrow A$ with the following properties:

EQ–1 $f \circ j = g \circ j$.

EQ–2 For each arrow $h : C \longrightarrow A$ such that $f \circ h = g \circ h$, there is an arrow $k : C \longrightarrow E$, and only one, such that $j \circ k = h$.

Frequently, E is referred to as an equalizer of f and g without referring to j. Nevertheless, j is a crucial part of the data.

9.1.3 Examples of equalizers In **Set**, an equalizer E of the functions $(x, y) \mapsto x^2 + y^2$ and $(x, y) \mapsto 1$ from $\mathbf{R} \times \mathbf{R}$ to $\mathbf{R}$ is the circle $x^2 + y^2 = 1$. The arrow $j : E \longrightarrow \mathbf{R} \times \mathbf{R}$ is the inclusion.

Given a graph $\mathscr{G}$, the inclusion of the set of loops in the graph is an equalizer of the source and target functions. This equalizer will be empty if the graph has no loops.

In a monoid M regarded as a category, any two elements of the monoid form a parallel pair of arrows which may or may not have an equalizer. (See Exercise 4.)

A theorem like Theorem 5.2.2 is true of equalizers as well.

9.1.4 Proposition *If $j : E \longrightarrow A$ and $j' : E' \longrightarrow A$ are both equalizers of $f, g : A \longrightarrow B$, then there is a unique isomorphism $i : E \longrightarrow E'$ for which $j' \circ i = j$.*

Proof. We give two proofs. The first uses the concept of universal element. Let $\mathscr{C}$ be the category containing the equalizers given. Let $F : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ be the functor for which $F(C) = \{u : C \longrightarrow A \mid f \circ u = g \circ u\}$, the set of arrows from C which equalize f and g. If $h : D \longrightarrow C$ let $F(h)(u) = u \circ h$. This makes sense, because if $u \in F(C)$, then $f \circ u \circ h = g \circ u \circ h$, so $u \circ h \in F(D)$. Note that this makes F a subfunctor of the contravariant hom functor $\mathrm{Hom}(-, A)$ as in Exercise 3 of Section 4.3. The definition of an equalizer of f and g can be restated this way: an equalizer of f and g is an element $j \in F(E)$ for some object E such that for any $u \in F(C)$ there is a unique arrow $k : C \longrightarrow E$ such that $F(k)(j) = u$. This means j is a universal element of F, so by Corollary 4.5.13, any two equalizers $j \in F(E)$ and $j' \in F(E')$ are isomorphic by a unique arrow $i : E \longrightarrow E'$ such that $F(i)(j') = j$. That is, $j = j' \circ i$, as required.

This method of constructing a functor of which the given limit is a universal element is a standard method in category theory. We have already seen it used in the proof of Proposition 5.2.14 and in the discussion of the uniqueness of eval in 6.1.7.

Here is a direct proof not using universal elements: the fact that $f \circ j' = g \circ j'$ implies the existence of a unique arrow $h : E' \longrightarrow E$ such that $j' = j \circ h$. The fact that $f \circ j = g \circ j$ implies the existence of a unique arrow $h' : E \longrightarrow E'$ such that $j = j' \circ h'$. Then $j \circ h \circ h' = j' \circ h' = j = j \circ \mathrm{id}_E$ and the uniqueness part of the definition of equalizer implies that $h \circ h' = \mathrm{id}_E$. By symmetry, $h' \circ h = \mathrm{id}_{E'}$. □

9.1.5 Proposition *Let $j : E \longrightarrow A$ be an equalizer of the pair of arrows $f, g : A \longrightarrow B$. Then j is a monomorphism. Moreover, any two equalizers of f and g belong to the same subobject of A.*

Proof. To see that j is monic, suppose $h, k : C \longrightarrow E$ with $j \circ h = j \circ k = l$. Then $f \circ l = f \circ j \circ k = g \circ j \circ k$ so there is a unique arrow $m : C \longrightarrow E$ with $j \circ m = l$. But both h and k are such arrows and so $h = k$.

Now suppose j and j' are equalizers of f and g. In the notation of Proposition 9.1.4, i and i^{-1} are the arrows required by the definition of subobject in 2.8.11, since $j' \circ i = j$ and $j \circ i^{-1} = j'$. □

9.1.6 More generally, if $f_1, \ldots, f_n$ are all arrows from A to B, then an object E together with an arrow $j : E \longrightarrow A$ is an equalizer of $f_1, \ldots, f_n$ if it has the property that an arbitrary arrow $h : C \longrightarrow A$ factors uniquely

through j if and only if $f_1 \circ h = \cdots = f_n \circ h$. Having equalizers of parallel pairs implies having equalizers of all finite lists (Exercise 2).

9.1.7 Regular monomorphisms A monomorphism $e : S \longrightarrow T$ in a category is **regular** if e is an equalizer of a pair of arrows. Monomorphisms in **Set** are regular (Exercise 5) but not all monomorphisms in **Cat** are regular (Exercise 6).

9.1.8 Proposition *An arrow in a category that is both an epimorphism and a regular monomorphism is an isomorphism.*

Proof. Let $f : A \longrightarrow B$ be both an epimorphism and an equalizer of $g, h : B \longrightarrow C$. Since $g \circ f = h \circ f$ and f is epi, $g = h$. Then $g \circ \mathrm{id}_B = h \circ \mathrm{id}_B$ so there is a $k : B \longrightarrow A$ such that $f \circ k = \mathrm{id}_B$. But then $f \circ k \circ f = f = f \circ \mathrm{id}_A$. But f being mono can be cancelled from the left to conclude that $k \circ f = \mathrm{id}_A$. □

9.1.9 Exercises

1. Recall that an arrow $f : A \longrightarrow B$ in a category is a split monomorphism if there is a $g : B \longrightarrow A$ with $g \circ f = \mathrm{id}_A$. Show that a split monomorphism is regular.

2. Show that a category with equalizers of all parallel pairs of arrows has equalizers of every finite list $f_1, \ldots, f_n : A \longrightarrow B$ of arrows.

3.† Prove that in the category of monoids every pair of parallel arrows has an equalizer.

4. a. Prove that if M is a free monoid, then in $C(M)$ no two different elements have an equalizer.

b. Prove the same statement for finite monoids. (Hint: See Exercise 3 of Section 2.9.)

5. Prove that every monomorphism in **Set** is regular.

6. Suppose that $\mathscr{A}$ and $\mathscr{B}$ are categories with two objects and whose only nonidentity arrows are as shown:

$\mathscr{A}$ $\mathscr{B}$

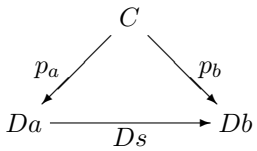
Show that the inclusion functor of $\mathscr{A}$ into $\mathscr{B}$ is an epimorphism in **Cat**. Conclude it cannot be a regular monomorphism. (Hints: The arrows u and v are inverse to each other. Use Proposition 9.1.8.)

9.2 The general concept of limit

Products and equalizers are both examples of the general concept of limit.

9.2.1 Definition Let $\mathscr{G}$ be a graph and $\mathscr{C}$ be a category. Let $D : \mathscr{G} \to \mathscr{C}$ be a diagram in $\mathscr{C}$ with shape $\mathscr{G}$. A **cone** with base D is an object C of $\mathscr{C}$ together with a family $\{p_a\}$ of arrows of $\mathscr{C}$ indexed by the nodes of $\mathscr{G}$, such that $p_a : C \to Da$ for each node a of $\mathscr{G}$. The arrow p_a is the **component** of the cone at a.

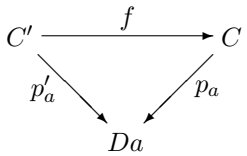
The cone is **commutative** if for any arrow $s : a \to b$ of $\mathscr{G}$, the diagram



commutes. Note: The diagram D is *not* assumed to commute.

Such a diagram will also be termed a commutative cone **over** D (or **to** D or with **base** D) with vertex C. We will write it as $\{p_a\} : C \to D$ or simply $p : C \to D$. It is clear that if $p : C \to D$ is a cone over D and $f : C' \to C$ is an arrow in $\mathscr{C}$, then there is a cone $p \circ f : C' \to D$ whose component at a is $p_a \circ f$. Moreover $p \circ f$ is commutative if p is. Note that a cone over a discrete diagram (see 5.1.3) is vacuously commutative.

9.2.2 Definition If $p' : C' \to D$ and $p : C \to D$ are cones, a **morphism** from p' to p is an arrow $f : C' \to C$ such that for each node a of $\mathscr{G}$, the diagram



commutes.

9.2.3 Definition A commutative cone over the diagram D is called **universal** if every other commutative cone over the same diagram has a unique arrow to it. A universal cone, if such exists, is called a **limit** of the diagram D.

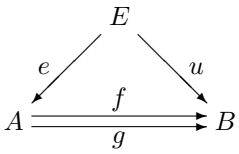
9.2.4 It is worth spelling out in some detail the meaning of a limit. To say that $p : C \to D$ is a limit means that there is given a family $p_a : C \to Da$, indexed by the nodes of $\mathscr{G}$, for which

L–1 Whenever $s : a \to b$ is an arrow of $\mathscr{G}$, then $Ds \circ p_a = p_b$.

L–2 If $p' : C' \longrightarrow D$ is any other such family with the property that $Ds \circ p'_a = p'_b$ for every $s : a \longrightarrow b$ in $\mathscr{G}$, then there is one and only one arrow $f : C' \longrightarrow C$ such that for each node a of $\mathscr{G}$, $p_a \circ f = p'_a$.

9.2.5 Examples A limit cone over a finite discrete diagram is a product cone: here, L–1 and the commutativity condition $Ds \circ p'_a = p'_b$ in L–2 are vacuous.

Equalizers are also limits. Let $f, g : A \longrightarrow B$ be parallel arrows in a category. An equalizer $e : E \longrightarrow A$ is part of a cone



(9.1)

where $u = f \circ e = g \circ e$. This cone is commutative, and it is a limit cone if and only if e is an equalizer of f and g. Since u is determined uniquely by f and e (or by g and e), it was not necessary to mention it in Definition 9.1.2.

9.2.6 Equivalent definitions of limit There are two more equivalent ways to define limits. Let $D : \mathscr{G} \longrightarrow \mathscr{C}$ be a diagram. We define the category cone(D) as follows. An object of this category is a commutative cone $\{p_i : C \longrightarrow Di\}$ and an arrow is a morphism of cones (Definition 9.2.2). It is evident that the identity is a morphism and that the composite of two morphisms is another one. A terminal object in cone(D), if one exists, is a commutative cone over D to which every other commutative cone over D has a unique morphism. Thus we have shown that the existence of a limit is equivalent to the existence of a terminal object of cone(D).

The second equivalent construction of limits associates to D a functor we call cone$(-, D) : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$. For an object C of $\mathscr{C}$, cone(C, D) is the set of commutative cones $p : C \longrightarrow D$. If $f : C' \longrightarrow C$, cone(f, D) : cone(C, D) $\longrightarrow$ cone(C', D) is defined by cone$(f, D)(p) = p \circ f$. It is straightforward to verify that this is a functor. A universal element of this functor is an object C and an element $p \in$ cone(C, D) such that for any C' and any element $p' \in$ cone(C', D) there is a unique arrow $f : C' \longrightarrow C$ such that $p \circ f = p'$. But this is just the definition of a limit cone. (Compare the proof of Proposition 9.1.4.)

We have now sketched the proof of the following.

9.2.7 Theorem *The following three are equivalent for a diagram D in a category $\mathscr{C}$:*

(i) *a limit of D;*

(ii) *a terminal object of* cone(D)*;*

(iii) *a universal element of the functor* $\mathrm{cone}(-, D)$.

In particular, one can see products and equalizers as terminal objects or as universal elements. One immediate consequence of this, in light of 4.5.13, is that a limit of a diagram is characterized uniquely up to a unique isomorphism in the same way that a product is.

9.2.8 Theorem *Let D be a diagram and $p : C \longrightarrow D$ and $p' : C' \longrightarrow D$ limits. Then there is a unique arrow $i : C \longrightarrow C'$ such that for every node a of the shape graph of D, $p'_a \circ i = p_a$ and i is an isomorphism.*

9.2.9 Definition A category is said to **have all limits** or to be **complete** if every diagram has a limit. It is said to **have all finite limits** or to be **finitely complete** if every diagram whose domain is a finite graph has a limit.

The following theorem gives a useful criterion for the existence of all limits or all finite limits. Another way of getting all finite limits is given by Proposition 9.3.7.

9.2.10 Theorem *Let $\mathscr{C}$ be a category in which every set (respectively, every finite set) of objects has a product and every parallel pair of arrows has an equalizer. Then every diagram (respectively, every finite diagram) in $\mathscr{C}$ has a limit.*

The proof of the main claim is given in Exercise 3. The case for finite limits is exactly the same, specialized to that case. As an immediate corollary, using Exercise 1.c of Section 5.3, we have the following.

9.2.11 Corollary *A category $\mathscr{C}$ with the following properties*

(i) *$\mathscr{C}$ has a terminal object;*

(ii) *every pair of objects has a product; and*

(iii) *every parallel pair of arrows has an equalizer*

has all finite limits.

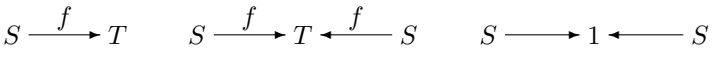
9.2.12 Definition A functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ **preserves limits** if it takes every limit cone in $\mathscr{C}$ to a limit cone in $\mathscr{D}$. It **preserves finite limits** if it takes every limit cone over a graph with a finite diagram in $\mathscr{C}$ to a limit cone in $\mathscr{D}$.

It follows from Corollary 9.2.11 that a functor that preserves terminal objects, products of pairs of objects, and equalizers of parallel pairs of arrows preserves all finite limits.

9.2.13 Exercises

1. Prove that Diagram (9.1) is a limit cone if and only if e is an equalizer of f and g.

2. Let $f : S \to T$ be a set function. Construct limits of each of these diagrams in **Set**. In (c) the arrows are the unique functions to the terminal object.



(a) (b) (c)

3. In this exercise, $\mathscr{C}$ is a category in which every set of objects has a product and every pair of arrows with the same source and target has an equalizer. The purpose of this exercise is to prove Theorem 9.2.10.

a. Suppose that $\mathscr{I}$ is a graph that may have an arbitrary set of nodes, but just one nonidentity arrow $a : j \to k$. Show that for any $D : \mathscr{I} \to \mathscr{C}$, an equalizer

$$E \to \prod_{i \in \mathscr{I}} Di \underset{\text{proj}_k}{\overset{Da \circ \text{proj}_j}{\rightrightarrows}} Dk$$

is a limit of D.

b. Suppose that $\mathscr{I}$ has just two nonidentity arrows $a : j \to k$ and $b : l \to m$ (no assumption of distinctness is made). Let $r_1 = Da \circ \text{proj}_j : \prod Di \to Dk$, $s_1 = \text{proj}_k : \prod Di \to Dk$, $r_2 = Db \circ \text{proj}_l : \prod Di \to Dm$ and $s_2 = \text{proj}_m : \prod Di \to Dm$. Let $r = \langle r_1, r_2 \rangle : \prod Di \to Dk \times Dm$ and $s = \langle s_1, s_2 \rangle : \prod Di \to Dk \times Dm$. Show that if

$$E \to \prod Di \underset{s}{\overset{r}{\rightrightarrows}} Dk \times Dm$$

is an equalizer, then E is a limit of D.

c. Let $\mathscr{I}$ be an arbitrary graph and $D : \mathscr{I} \to \mathscr{C}$ a diagram in $\mathscr{C}$. Let $A = \prod Di$, taken over the objects of $\mathscr{I}$ and $B = \prod(D(\text{target}\, a))$, the product taken over the set of all arrows of $\mathscr{I}$. Let $r : A \to B$ be such that $\text{proj}_a \circ r = Da \circ \text{proj}_{\text{source}(a)}$ and $s : A \to B$ be such that $\text{proj}_a \circ s = \text{proj}_{\text{target}(a)}$. Let

$$E \to A \underset{s}{\overset{r}{\rightrightarrows}} B$$

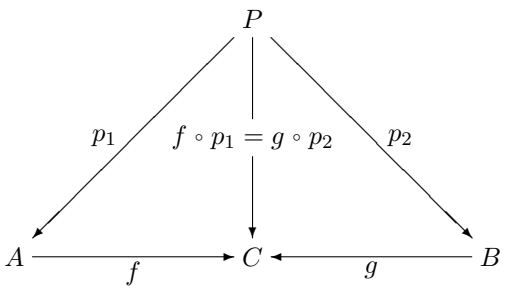
be an equalizer. Then E is a limit of D.

9.3 Pullbacks

Here is another example of a finite limit that will be important to us. Consider an object P and arrows $p_1 : P \to A$ and $p_2 : P \to B$ such that the diagram

$$\begin{array}{ccc} P & \xrightarrow{p_1} & A \\ {\scriptstyle p_2}\downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[g]{} & C \end{array} \tag{9.2}$$

commutes. This does not appear directly to be a cone, because there is no arrow from the vertex P to C. It is understood to be a commutative cone with the arrow from P to C being the composite $f \circ p_1$, which by definition is the same as the composite $g \circ p_2$. It is common to omit such forced arrows in a cone. (Compare the discussion after Diagram (9.1).) It is more cone-like if we redraw the diagram as



However, the square shape of (9.2) is standard.

If this commutative cone is universal, then we say that P together with the arrows p_1 and p_2 is a **pullback** or **fiber product** of the pair. We also say that p_2 is the **pullback of** f **along** g, and that (9.2) is a **pullback diagram**. We often write $P = A \times_C B$, although this notation omits the arrows which are as important as the object C.

9.3.1 Example In **Set**, if $f : S \to T$ and $g : U \to T$ are functions, then the pullback

$$\begin{array}{ccc} P & \xrightarrow{p_1} & S \\ {\scriptstyle p_2}\downarrow & & \downarrow{\scriptstyle f} \\ U & \xrightarrow[g]{} & T \end{array} \tag{9.3}$$

is constructed by setting

$$P = \{(s, u) \mid f(s) = g(u)\}$$

with $p_1(s, u) = s$ and $p_2(s, u) = u$.

In one way this example is characteristic of pullbacks in any category with products: in any such category, P is a subobject of $S \times U$ (Exercise 4).

9.3.2 Example The inverse image of a function is a special case of a pullback. Suppose $g : S \to T$ is a set function and $A \subseteq T$. Let $i : A \to T$ be the inclusion. Let $g^{-1}(A) = \{x \in S \mid g(x) \in A\}$ be the inverse image of A, j be its inclusion in S, and h be the restriction of g to $g^{-1}(A)$. Then the following is a pullback diagram.

$$\begin{array}{ccc} g^{-1}(A) & \xrightarrow{h} & A \\ {\scriptstyle j}\downarrow & & \downarrow{\scriptstyle i} \\ S & \xrightarrow[g]{} & T \end{array} \tag{9.4}$$

Observe that Example 9.3.1 gives a general construction for pullbacks in sets, and the present example gives a construction for a special type of pullback, but this construction *is not a special case* of the construction in 9.3.1. By Theorem 9.2.8, there must be a unique function $u : \{(s, a) \mid g(s) = a\} \to g^{-1}(A)$ for which $g(j(u(s, a))) = i(h(u(s, a)))$. By the definitions of the functions involved, $u(s, a)$ must be s.

The reader may wish to know the origin of the term 'fiber product'. Consider a function $f : S \to T$. Of the many ways to think of a function, one is as determining a partition of S indexed by the elements of T. (See 2.6.11.) For $t \in T$, let

$$S_t = f^{-1}(t) = \{s \in S \mid f(s) = t\}$$

This is a family of disjoint subsets of S (some of which may be empty) whose union is all of S. This is sometimes described as a **fibration** of S and S_t is called the **fiber** over t. Now if $S \to T$ and $U \to T$ are two arrows and $S \times_T U$ is a pullback, then it is not hard to see that

$$S \times_T U = \bigcup \{S_t \times U_t \mid t \in T\}$$

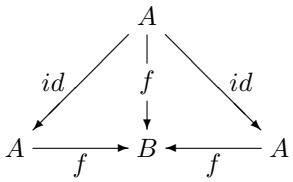
In other words, $S \times_T U$ is the fibered set whose fiber over any $t \in T$ is the product of the fibers. This is the origin of the term and of the notation. We will not use the term in this book, but the notation has become standard and is too useful to abandon. Fibrations can be constructed for categories as well as sets; they are considered in Section 12.1.

Pullbacks can be used to characterize monomorphisms in a category.

9.3.3 Theorem *These three conditions are equivalent for any arrow $f : A \longrightarrow B$ in a category $\mathscr{C}$:*

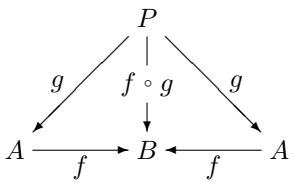
(a) *f is monic.*

(b) *The diagram*



is a limit cone.

(c) *There is an object P and an arrow $g : P \longrightarrow A$ for which*



is a limit cone.

(The operative condition in (c) is that the same arrow g appears on both slant lines.)

Another connection between pullbacks and monomorphisms is given by the following.

9.3.4 Proposition *In Diagram (9.2), if the arrow f is monic, then so is p_2.*

This proposition is summed up by saying, 'A pullback of a monic is a monic.' Contrast it with the situation for epis given in Exercises 7 and 8.

The proofs of Theorem 9.3.3 and Proposition 9.3.4 are left as exercises.

9.3.5 Weakest precondition as pullback A widely used approach to program verification is to attach preconditions and postconditions to program fragments. An example is

$$\begin{array}{ll} 1 \circ & \{\mathtt{X} < 3\} \\ 2 \circ & \mathtt{X} := \mathtt{X} + 1; \\ 3 \circ & \{\mathtt{X} < 24\} \end{array} \tag{9.5}$$

where X is an integer variable.

Statement 1 is called a **precondition** and statement 3 is a **postcondition**. The whole expression (9.5) is an **assertion** about the program: if the precondition is true before the program fragment is executed, the postcondition must be true afterward.

Clearly (9.5) is correct but a stronger assertion can be made. For the given postcondition, the weakest possible precondition is $\{\texttt{X} < \texttt{23}\}$. In a very general setting, there is a weakest precondition for every postcondition.

This can be placed in a categorical setting. Let D be the set of possible inputs and E be the set of possible outputs. Then the program fragment, provided it is deterministic and terminating, can be viewed as an arrow $f : D \longrightarrow E$. Any condition C on a set X can be identified with the subset $X_0 = \{x \mid x \text{ satisfies } C\}$. In particular, the postcondition is a subset $S \subseteq E$ and the weakest precondition for that postcondition is the inverse image $f^{-1}(S)$, which is the unique subobject of D for which

$$\begin{array}{ccc} f^{-1}(S) & \rightarrowtail & D \\ \downarrow & & \downarrow \\ S & \rightarrowtail & E \end{array}$$

is a pullback. In the example, $f(x) = x + 1$, $S = \{x \mid x < 24\}$ and $f^{-1}(S) = \{x \mid x + 1 < 24\} = \{x \mid x < 23\}$.

This is easily checked in the case of **Set** and is a plausible point of view for other categories representing data and program fragments.

In the general case, one would expect that an assertion would be some special kind of subobject. For example, Manes [1986] requires them to be 'complemented', a concept to be discussed in 9.6.2 below, and Wagner [1987] requires them to be regular (see 9.1.7). Naturally, if one requires that assertions be subobjects with a certain property, one would want to work in a category in which pullbacks of such subobjects also had the property. In a topos (toposes will be discussed in Chapter 15), for example, pullbacks of complemented subobjects are complemented, and all monomorphisms are regular.

9.3.6 Constructing all finite limits revisited Corollary 9.2.11 described a class of limits whose existence guarantees the existence of all finite limits. A similar construction is possible for pullbacks.

9.3.7 Proposition *A category that has a terminal object and all pullbacks has all finite limits.*

The proof is left as an exercise.

9.3.8 Exercises

1. Prove that (9.4) is a pullback diagram.

2. Prove Theorem 9.3.3.

3. Prove Proposition 9.3.4.

4. Show that if Diagram (9.3) is a pullback in any category with products, then $\langle p_1, p_2 \rangle : P \longrightarrow S \times U$ is monic. (Here p_1 and p_2 refer to the arrows so labeled in the diagram.)

5. Show that equalizers can be constructed out of products and pullbacks. (Hint: Try to work it out in the category of sets and then do it in general categories.)

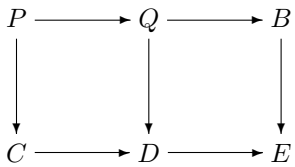
6. Prove Proposition 9.3.7.

7. Prove that in **Set** if f in Diagram (9.2) is an epimorphism, then so is p_2. (One says that a pullback of an epimorphism is an epimorphism in **Set**.)

8.† Give an example of a category with an epimorphism whose pullback along at least one arrow is not an epimorphism.

9. Prove that a pullback of a split epimorphism is a split epimorphism in any category.

10. Show that if the left and right squares below are pullbacks then so is the outer rectangle. What does this say about weakest preconditions?



9.4 Coequalizers

In 9.1.2 we introduced the notion of equalizer. The dual notion is called coequalizer. Explicitly, consider $f, g : A \rightrightarrows B$. An arrow $h : B \longrightarrow C$ is called a **coequalizer** of f and g provided $h \circ f = h \circ g$ and for any arrow $k : B \longrightarrow D$ for which $k \circ f = k \circ g$, there is a unique arrow $l : C \longrightarrow D$ such that $l \circ h = k$.

The way to think about the coequalizer of f and g is as the quotient object made by forcing f and g to be equal. In the category of sets, this observation becomes the construction that follows.

9.4.1 Coequalizers in Set Let $f, g : S \rightrightarrows T$ be a pair of arrows in the category of sets. The conditions a coequalizer $h : T \longrightarrow U$ must satisfy are that $h \circ f = h \circ g$ and that given any arrow $k : T \longrightarrow V$, the equation $k \circ f = k \circ g$ implies the existence of a unique arrow $l : U \longrightarrow V$ such that $l \circ h = k$. (If such an arrow l exists, then necessarily $k \circ f = k \circ g$.)

The pair f and g determine a relation $R \subseteq T \times T$ which consists of

$$\{(f(s), g(s)) \mid s \in S\}$$

The equation $k \circ f = k \circ g$ is equivalent to the statement that $k(t_1) = k(t_2)$ for all $(t_1, t_2) \in R$.

In general R is not an equivalence relation, but it can be completed to one by forming the reflexive, symmetric and transitive closures of R in that order. The reflexive, symmetric closure is the union $R_1 = R \cup \Delta \cup R^{\mathrm{op}}$ where $\Delta = \{(t, t) \mid t \in T\}$ and $R^{\mathrm{op}} = \{(t_2, t_1) \mid (t_1, t_2) \in R\}$. If we inductively define R_{i+1} to be the set

$$\{(t_1, t_2) \mid \exists t \in T((t_1, t) \in R_1 \text{ and } (t, t_2) \in R_i)\}$$

for each $i \in \mathbf{N}$ and define $\widetilde{R} = \bigcup R_n$, then $\widetilde{R}$ is the transitive closure of R_1 and is the least equivalence relation containing R.

When S and T are finite, the least equivalence relation of the previous paragraph can be computed efficiently by Warshall's Algorithm ([Sedgewick, 1983], p. 425).

Let U be the set of equivalence classes modulo the equivalence relation $\widetilde{R}$ and let $h : T \longrightarrow U$ be the function that sends an element of T to the class that contains that element.

9.4.2 Proposition *The arrow h is a coequalizer of f and g.*

Proof. Since $R \subseteq \widetilde{R}$, the fact that $h(t_1) = h(t_2)$ for all $(t_1, t_2) \in \widetilde{R}$ implies, in particular, that the same equation is satisfied for all $(t_1, t_2) \in R$; this means that for all $s \in S$, $h(f(s)) = h(g(s))$, so that $h \circ f = h \circ g$.

Now let $k : T \longrightarrow V$ satisfy $k \circ f = k \circ g$. We claim that for $(t_1, t_2) \in \widetilde{R}$, $k(t_1) = k(t_2)$. In fact, this is true for $(t_1, t_2) \in R$. It is certainly true for $t_1 = t_2$, i.e. $(t_1, t_2) \in \Delta$ implies that $k(t_1) = k(t_2)$. Since $k(t_1) = k(t_2)$ implies that $k(t_2) = k(t_1)$, we now know that the assertion is true for $(t_1, t_2) \in R_1$. Since $k(t_1) = k(t_2)$ and $k(t_2) = k(t_3)$ imply $k(t_1) = k(t_3)$, one may show by induction that the assertion is true for all the R_n and hence for $\widetilde{R}$. This shows that k is constant on equivalence classes and hence induces a function $l : U \longrightarrow V$ such that $l \circ h = k$. Since h is surjective, l is unique. □

9.4.3 Regular epimorphisms In Proposition 9.1.5, it is asserted that all equalizers are monomorphisms. The dual is of course also true; all coequalizers are epimorphisms. The converse is not true; not every epimorphism is a coequalizer of some pair of arrows. For example, in the category of monoids, the inclusion of the nonnegative integers into the set of all integers is an epimorphism but is not the coequalizer of any pair of arrows. Those epimorphisms that are the coequalizer of some pair of arrows into their domain are special and this is marked by giving them a special name: they are called **regular epimorphisms**.

In the category of sets, every epimorphism is regular (Exercise 2). Because of Exercise 7 of Section 9.3, it follows that in the category of sets, a pullback of a regular epi is regular.

A category is called a **regular category** if it has finite limits, if every parallel pair of arrows has a coequalizer, and if the arrow opposite a regular epimorphism in a pullback diagram is a regular epimorphism. A functor between regular categories is called a **regular functor** if it preserves finite limits and regular epis.

It is an old theorem (stated in different language) that all categories of models of FP sketches, in other words, all varieties of multisorted algebraic structures, are regular ([Barr and Wells, 1985], Theorem 1 of Section 8.4). Because of such examples, regular categories have seen considerable theoretical study, and regular epimorphisms have played a larger role in category theory than regular monos have.

9.4.4 Exercises

1. Show that any coequalizer is an epimorphism.

2. Prove that an epimorphism in the category of sets is regular. (Hint: Its image is isomorphic to its quotient.)

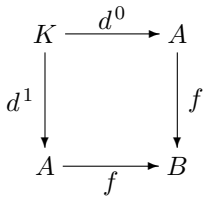
3. Let $\mathbf{N}$ denote the monoid of natural numbers on addition and $\mathbf{Z}$ the monoid of integers on addition. Show that the inclusion function is an epimorphism (in the category of monoids) which is not regular.

4.† Prove that in the category of monoids an epimorphism f is regular if and only if $U(f)$ is surjective, where U is the underlying functor to **Set**.

5. If $f : A \longrightarrow B$ is an arrow in a category, a kernel pair

$$K \underset{d^1}{\overset{d^0}{\rightrightarrows}} A$$

is a limit



Show that if f has a kernel pair d^0, d^1, then f is a regular epimorphism if and only if f is the coequalizer of its kernel pair.

9.5 Cocones

Just as there is a general notion of limit of which products and equalizers are special cases, so there is a general notion of colimit.

9.5.1 Definition A **cocone** is a cone in the dual graph. Spelling the definition out, it is a diagram $D : \mathscr{G}_0 \longrightarrow \mathscr{G}$, a node g of $\mathscr{G}$ together with a family $u_a : Da \longrightarrow g$ of arrows indexed by the objects of $\mathscr{G}_0$. The diagram is called the **base** of the cocone. If the diagram is discrete, it is called a **discrete cocone**.

A cocone in a category is called **commutative** if it satisfies the dual condition to that of commutative cone. Explicitly, if the diagram is $D : \mathscr{G} \longrightarrow \mathscr{C}$ and the cone is $\{u_a : Da \longrightarrow C \mid a \in \mathrm{Ob}(\mathscr{G})\}$, then what is required is that for each arrow $s : a \longrightarrow b$ of $\mathscr{G}$, $u_b \circ Ds = u_a$. The commutative cocone is called a **colimit** cocone if it has a unique arrow to every other commutative cocone with the same base.

9.5.2 Cocompleteness A category **has all colimits** or is **cocomplete** if every diagram in the category has a colimit. It **has finite colimits** or is **finitely cocomplete** if every diagram with a finite graph has a colimit.

Categories of algebraic structures are complete and cocomplete. If we disallowed the empty semigroup (see 2.7.17) we would have to say that the category of semigroups is 'complete and cocomplete except that it does not have an initial object'. We would also have to say that the intersection of any set of subsemigroups of a semigroup is either a subsemigroup or empty.

9.5.3 Pushouts For example, dual to the notion of pullback is that of pushout. In detail, a commutative square

$$\begin{array}{ccc} C & \xrightarrow{f} & A \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle q_1} \\ B & \xrightarrow[q_2]{} & Q \end{array} \tag{9.6}$$

is called a **pushout** if for any object R and any pair of arrows $r_1 : A \longrightarrow R$ and $r_2 : B \longrightarrow R$ for which $r_1 \circ f = r_2 \circ g$ there is a unique arrow $r : Q \longrightarrow R$ such that $r \circ q_i = r_i$, $i = 1, 2$.

9.5.4 Pushouts as amalgamations If the effect of coequalizers is to force identifications, that of pushouts is to form what are called amalgamated sums. We illustrate this with examples.

First, consider the case of a set S and three subsets S_0, S_1 and S_2 with $S_0 = S_1 \cap S_2$ and $S = S_1 \cup S_2$. Then the diagram

$$\begin{array}{ccc} S_0 & \longrightarrow & S_1 \\ \downarrow & & \downarrow \\ S_2 & \longrightarrow & S \end{array}$$

with all arrows inclusion, is a pushout. (It is a pullback as well; such diagrams are often referred to as **Doolittle diagrams**.) The definition of pushout translates to the obvious fact that if one is given functions $f_1 : S_1 \longrightarrow T$ and $f_2 : S_2 \longrightarrow T$ and $f_1|S_0 = f_2|S_0$, then there is a unique function $f : S \longrightarrow T$ with $f|S_1 = f_1$ and $f|S_2 = f_2$.

Now consider a slightly more general situation. We begin with sets S_0, S_1 and S_2 and functions $g_1 : S_0 \longrightarrow S_1$ and $g_2 : S_0 \longrightarrow S_2$. If g_1 and g_2 are injections, then, up to isomorphism, this may be viewed as the same as the previous example. Of course, one cannot then form the union of S_1 and S_2, but rather first the disjoint sum and then, for $s \in S_0$, identify the element $g_1(s) \in S_1$ with $g_2(s) \in S_2$. In this case, the pushout is called an **amalgamated sum** of S_1 and S_2.

More generally, the g_i might not be injective and then the amalgamation might identify two elements of S_1 or of S_2, but the basic idea is the same; *pushouts are the way you identify part of one object with a part of another.*

9.5.5 Equations, equalizers and coequalizers One way of thinking about an equalizer is as the largest subobject on which an equation or set of equations is true. A coequalizer, by contrast, is the least destructive identification necessary to force an equation to be true on the equivalence classes.

Here is an instructive example. There are two ways of defining the rational numbers. They both begin with the set $\mathbf{Z} \times \mathbf{N}^+$ of pairs of integers (a, b) for which $b > 0$. For the purpose of this illustration, we will write (a/b) instead of (a, b). In the first, more familiar, construction, we identify (a/b) with (c/d) when $a * d = b * c$. One way of describing this is as the coequalizer of two arrows

$$T \rightrightarrows \mathbf{Z} \times \mathbf{N}^+$$

where T is the set of all $(a, b, c, d) \in \mathbf{Z} \times \mathbf{N}^+ \times \mathbf{Z} \times \mathbf{N}^+$ such that $a * d = b * c$ and the first arrow sends (a, b, c, d) to (a/b) while the second sends it to (c/d). The effect of the coequalizer is to identify (a/b) with (c/d) when $a * d = b * c$.

The second way to define the rationals is the set of pairs that are relatively prime (that is, in lowest terms). This can be realized as an equalizer as follows. Let $\gcd : \mathbf{Z} \times \mathbf{N}^+ \to \mathbf{N}^+$ take a pair of numbers to their positive greatest common divisor. Let $1 \circ \langle\rangle : \mathbf{Z} \times \mathbf{N}^+ \to \mathbf{N}^+$ denote the function that is constantly 1. Then an equalizer of gcd and $1 \circ \langle\rangle$ is exactly the set of relatively prime pairs.

9.5.6 Exercises

1. Show that in any category, if both squares in

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array} \qquad \begin{array}{ccc} A' & \xrightarrow{f'} & B' \\ {\scriptstyle g'}\downarrow & & \downarrow{\scriptstyle h'} \\ C' & \xrightarrow[k']{} & D' \end{array}$$

are pushouts, then so is

$$\begin{array}{ccc} A + A' & \xrightarrow{f + f'} & B + B' \\ {\scriptstyle g + g'}\downarrow & & \downarrow{\scriptstyle h + h'} \\ C + C' & \xrightarrow[k + k']{} & D + D' \end{array}$$

2. Let A, B, C be sets and $f : C \to A$, $e : C \to B$ functions with e injective. Let $e(C)$ denote the image of e and $X = B - e(C)$ the complement of the image in B. Let $i : A \to A + X$ and $j : X \to A + X$ be the canonical injections.

a. Show that the diagram below is a pushout.

$$\begin{array}{ccc} C & \xrightarrow{e} & B = \mathrm{e(C)+X} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle i \circ f \circ e^{-1} + j} \\ A & \xrightarrow[i]{} & A + X \end{array}$$

b. Conclude that in **Set** *the pushout of a monomorphism is a monomorphism.*

3. a. Show that $\mathbf{Z}$ can be constructed from $\mathbf{N}$ as the coequalizer of two arrows $g, h : \mathbf{N} \times \mathbf{N} \times \mathbf{N} \to \mathbf{N} \times \mathbf{N}$, defined by $g(a, b, c) = (b, c)$ and $h(a, b, c) = (a + b, a + c)$. (Hint: Addition is coordinatewise; 1 and -1 are the classes containing $(1, 0)$ and $(0, 1)$, respectively.)

b. Show that the integers can also be constructed as a subset of $\mathbf{N} \times \mathbf{N}$ consisting of all (n, m) such that $m = 0$ or $n = 0$. The sum is the usual sum followed by the application of the function $r(n, m) = (n \mathbin{\dot{-}} m, m \mathbin{\dot{-}} n)$, where $\dot{-}$ is subtraction at 0. This function is the additive analog of reduction to lowest terms by dividing by the greatest common divisor.

4. Let $f, g : \mathbf{N}^+ \times \mathbf{Z} \times \mathbf{N}^+ \to \mathbf{Z} \times \mathbf{N}^+$ be defined by $f(a, b, c) = (b, c)$ and $g(a, b, c) = (ab, ac)$. Show that the coequalizer of f and g can also be thought of as the rational numbers (see 9.5.5).

5. Show that in the category of sets, if both squares in

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array} \qquad \begin{array}{ccc} A' & \xrightarrow{f'} & B' \\ {\scriptstyle g'}\downarrow & & \downarrow{\scriptstyle h'} \\ C' & \xrightarrow[k']{} & D' \end{array}$$

are pullbacks, then so is

$$\begin{array}{ccc} A + A' & \xrightarrow{f + f'} & B + B' \\ {\scriptstyle g + g'}\downarrow & & \downarrow{\scriptstyle h + h'} \\ C + C' & \xrightarrow[k + k']{} & D + D' \end{array}$$

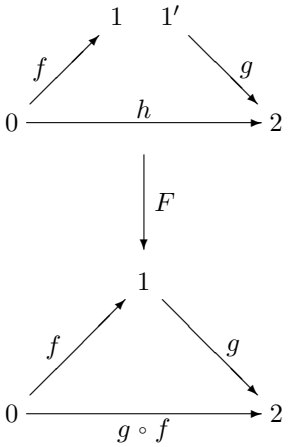
6.† **a.** Let **2** be the category with two objects and one nonidentity arrow between them and **N** be the category with one object and the natural numbers as its set of arrows, with 0 the identity arrow and $n \circ m = n + m$. Show that the functor $q : \mathbf{2} \longrightarrow \mathbf{N}$ that takes the nonidentity arrow of **2** to 1 is a regular epi, even though it is not surjective on arrows. Note that this shows that the underlying arrow functor from **Cat** to **Set** does not preserve coequalizers, since coequalizers in **Set** are surjective.

b. Let $t : \mathbf{N} \longrightarrow \mathbf{N}$ be the functor that assigns to the integer n the number $2n$. Let $\mathbf{1} + \mathbf{1}$ be the category with two objects and two identity arrows and none other. Show that there is a pullback diagram

$$\begin{array}{ccc} \mathbf{1}+\mathbf{1} & \longrightarrow & \mathbf{N} \\ \downarrow & & \downarrow{\scriptstyle t} \\ \mathbf{2} & \xrightarrow[q]{} & \mathbf{N} \end{array}$$

and that the top arrow is not a regular epi.

c. Let **3** be the category with three objects, say the numbers 0, 1 and 2 and three nonidentity arrows, one from 0 to 1, one from 1 to 2 and the composite of them from 0 to 2. Let $\mathscr{C}$ be the category with four objects we will call 0, 1, $1'$ and 2 and three nonidentity arrows, one from 0 to 1, one from $1'$ to 2 and one from 0 to 2. Show that the functor $F : \mathscr{C} \longrightarrow \mathbf{3}$ pictured in Diagram (9.7) is not a regular epi, even though it is surjective on both objects and arrows.



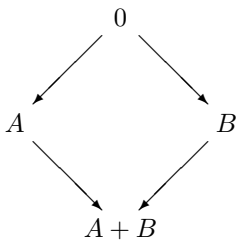
(9.7)

d. Show that a regular epi in **Cat** is **stable**, that is every pullback of it is a regular epi, if and only if it induces a surjection on the set of composable pairs of arrows.

9.6 More about sums

Sums in the category of sets have special properties they do not have in most other categories, including most categories of mathematical structures. Two of these special properties, that sums are disjoint and that they are universal, are widely assumed in the study of categories suitable for programming language semantics. They are defined in this section.

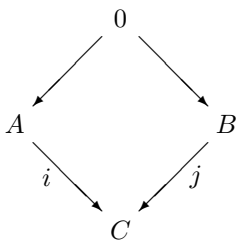
9.6.1 Disjoint sums Suppose that A and B are two objects in a category with an initial object 0 and a sum $A + B$. Then we have a commutative diagram



(9.8)

If this diagram is a pullback, and the canonical injections $A \longrightarrow A + B$ and $B \longrightarrow A + B$ are monic, then the sum is said to be **disjoint**. It is easy to see that sums in the category of sets are disjoint. In fact, that is essentially how they are defined.

9.6.2 Diagram (9.8) is a pushout in any case. If a diagram



(9.9)

is both a pullback and a pushout, then C is the sum of A and B with the arrows i and j as canonical injections. If i and j are monic, then A and B are subobjects of C and the sum is disjoint. In this situation B is said to be the **complement** of A in C (and conversely), and A and B are said to be **complemented subobjects**.

9.6.3 Example In **Set**, every subobject is complemented: the complement is what is usually meant by complement.

9.6.4 Example In **Mon** there are many subobjects which are noncomplemented; for example, the set of even integers is a submonoid of the monoid of all integers on addition, but it has no complement. Its set-theoretic complement, the set of odd integers, is not a submonoid: it does not contain the identity element (zero) and the sum of two odd integers is *never* odd. Even if its set-theoretic complement were a submonoid (impossible since only one of the two could contain the unit element), it still would not necessarily be the complement in the category.

A semigroup *can* have two subsemigroups whose underlying sets are complements but which is not the sum in the category of semigroups of those two subsemigroups. See Exercise 5.

9.6.5 Computable subsets More interesting from a computational point of view is the fact that some computable subsets of the natural numbers do not have computable complements. That is, there are subsets A of $\mathbf{N}$ with both these properties:

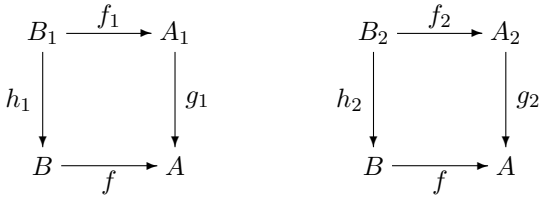
(a) There is an algorithm that halts on n if and only if $n \in A$.

(b) There is no algorithm to determine if an arbitrary element $n \in A$.

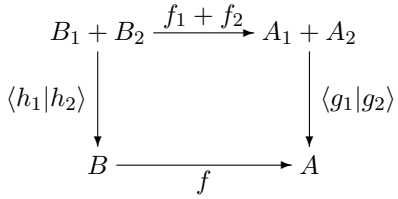
A set A is said to be **computable** or **recursively enumerable** if it has property (a). If A also has property (b) then the complement of A cannot be computable. (See [Lewis and Papadimitriou, 1981].) Thus in the category of computable subsets of the natural numbers and computable maps between them, there are noncomplemented subobjects. An expression such as `If` n `In A Then Do P` should have a *computable* meaning only if A is complemented.

The category of Boolean algebras (defined in 5.7.5) is an example of a category in which sums are not disjoint. Although many books do not allow the one-element Boolean algebra, we do. It is the terminal Boolean algebra. In the category it is characterized by the fact that it admits no arrows to any Boolean algebra but itself. This is because in this algebra true = false and an arrow between Boolean algebras must take true to true and false to false. That being the case, the only possible sum of 1 and 1 is 1 itself since the sum must admit an arrow from each constituent. From this it is clear that the pullback in question is 1 which is not initial (actually 2 is initial).

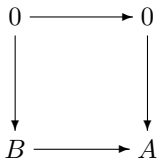
9.6.6 Universal sums Finite sums in a category are **universal** if whenever



are pullbacks, then so is



In addition it is required that if 0 denotes the initial object of the category, then for any arrow $B \longrightarrow A$, the diagram



must be a pullback. The meaning of these two conditions together can be summarized by the statement that pullbacks preserve finite sums. A finite sum is either of no objects, one object, or two or more. The sum of no objects is the initial object; of one it is the object in question and the condition is vacuous since the identity functor preserves everything and the condition for more than two follows immediately from the one for two.

Although finite (and infinite) sums are universal in the category of sets, they are not in many familiar categories. In the category of monoids, for example, the initial monoid 0 is the monoid with one element. Let us consider $\mathbf{Z}$ as a monoid with one generator x; in this guise, its elements are all powers x^n for n positive, negative or zero, and multiplication is defined by $x^m x^n = x^{m+n}$. Let $\mathbf{Z}'$ be a copy of $\mathbf{Z}$ with generator y. The sum $\mathbf{Z} + \mathbf{Z}'$ consists of all finite words in powers of x and powers of y, with multiplication illustrated by

$$x^2 y^{-2} x^4 \cdot x^{-1} y^3 = x^2 y^{-2} x^3 y^3$$

and

$$y^3 x^{-2} \cdot y^{-2} x^6 = y^3 x^{-2} y^{-2} x^6$$

Define the arrow $\mathbf{Z} + \mathbf{Z}' \to \mathbf{Z}$ by letting the generators x and y both go to x. The words in powers of x and y that go to the identity element x^0 are all those whose exponents add up to 0, for example any word of the form $x^n y^{-n}$. On the other hand, in each summand the only word going to x^0 is the identity x^0 or y^0. The pullback of

$$\begin{array}{ccc} & & \mathbf{Z} + \mathbf{Z} \\ & & \downarrow \\ 0 & \longrightarrow & \mathbf{Z} \end{array}$$

consists of all the elements of $\mathbf{Z} + \mathbf{Z}'$ that go to x^0 and so is not the initial object 0. Thus this pullback does not preserve finite sums. (See Exercise 4.)

9.6.7 Definition A category is called an **FLS category** (for finite limits and sums) if it has finite limits and finite sums and the sums are disjoint and universal.

9.6.8 Coherent categories A category is called **coherent** if it is both FLS and regular. A functor between coherent categories is called **coherent** if it preserves finite limits, finite sums and regular epimorphisms. Coherent categories correspond to an important type of logic called **positive logic**. See [Makkai and Reyes, 1977].

9.6.9 Infinite universal sums In a category with pullbacks, an infinite sum $A = \sum A_i$ with transition arrows $u_i : A_i \longrightarrow A$ is universal if for any arrow $B \longrightarrow A$, we have that the natural arrow $\sum B \times_A A_i \longrightarrow B$ is an isomorphism. To explain this in more detail, for each i, let

$$\begin{array}{ccc} B_i & \longrightarrow & A_i \\ {\scriptstyle v_i}\downarrow & & \downarrow{\scriptstyle u_i} \\ B & \longrightarrow & A \end{array}$$

be a pullback. Then the assertion that the sum is universal is equivalent to the assertion that the cocone with elements $v_i : B_i \longrightarrow B$ is a sum.

9.6.10 Exercises

1. Prove that Diagram (9.8) is a pushout in any category.

2. Prove that if Diagram (9.9) is a pushout then C is the sum of A and B with canonical injections i and j.

3. Prove that **Set** has universal sums.

4.† Let E denote the one-element semigroup. Note that it is a monoid. Show that in the category of semigroups, $E + E$ is infinite, but in the category of monoids, $E + E = E$. (Hint for the first part: Let $u : \mathbf{N} \to \mathbf{N}$ denote the function which adds 1 to even integers but leaves odd integers alone, and $v : \mathbf{N} \to \mathbf{N}$ the function which adds 1 to odd integers but leaves even integers alone. Note that u and v are both idempotent, but that the composites $(u \circ v)^n$, $n = 1, \ldots, n$ are all different.)

5.† Let S be the semigroup with two elements 0 and 1 under the usual multiplication of numbers. Show that even though both $\{0\}$ and $\{1\}$ are subsemigroups and each is the set-theoretic complement of the other, they are not complementary subobjects of S and that, in fact, their sum in the category of semigroups is infinite.

6.† (For those who know something about computability.)

a. Prove that there is a category whose only object is the set of natural numbers and whose arrows are the computable (recursive) functions with the usual composition of functions as composition.

b. Show that in this category there are subobjects without complements.

7.† Show that a category with 1 and universal countable sums has a stable natural numbers object. (See Section 5.5 and especially Theorem 5.5.5.)

9.7 Unification as coequalizer

A **unification** of two expressions is a common substitution which results in the two expressions becoming the same. This may not be possible, but when it is, there is a least such expression. We shall formulate this as the existence of a coequalizer (Theorem 9.7.5 below).

9.7.1 As an example, suppose we have a structure with three sorts, A, C and S, and two operations $m : A \times A \to S$ and $t : C \times C \to A$. If $a, b \in A$ and $u, v, w \in C$, then we can construct terms

$$\begin{array}{rcl} x_1 & = & m(t(u,v),a) \\ y_1 & = & m(a,a) \\ x_2 & = & m(b,t(u,w)) \\ y_2 & = & m(b,a) \end{array} \tag{9.10}$$

all of type S. As it happens, the substitutions

$$\begin{array}{rcl} w & \leftarrow & v \\ a & \leftarrow & t(u,v) \end{array} \tag{9.11}$$

in (9.10) produce

$$\begin{array}{rcl} x_1' & = & m(t(u,v),t(u,v)) \\ y_1' & = & m(t(u,v),t(u,v)) \\ x_2' & = & m(b,t(u,v)) \\ y_2' & = & m(b,t(u,v)) \end{array} \tag{9.12}$$

so that $x_i' = y_i'$ for $i = 1, 2$. Moreover, *any other substitution in (9.10) which causes those equations to be true can be obtained by a further substitution in (9.12)*, so that the substitution in (9.10) is a unification of (x_1, y_1) with (x_2, y_2).

9.7.2 Free sketches Let us say that an FP sketch is a **free FP sketch** if it has no diagrams and if any node that is the vertex of a cone appears in that cone *only*. The nodes look like $s_1 \times s_2 \times \cdots \times s_n$ and although the operations generally have the form

$$f : s_1 \times \cdots \times s_n \longrightarrow t_1 \times \cdots \times t_m$$

we can without loss of generality suppose that the target of any operation is actually not a product node. The reason is that a function to a product is, by Definition 5.3.1, completely determined by the functions obtained by composing it with the projections, so you can always replace a function to a product of n objects by n functions, one to each object in the product.

If we have a cone with base nodes $s_1, \ldots, s_n$ and vertex s, which we generally denote $s_1 \times \cdots \times s_n$, then a free algebra with a generator x of type s and possibly other generators, contains the elements $p_1(x), \ldots, p_n(x)$ of type $s_1, \ldots, s_n$, respectively. It is clear that it is isomorphic to a free algebra generated by elements $x_1, \ldots, x_n$ of types $s_1, \ldots, s_n$, respectively. Thus every free algebra is isomorphic to a free algebra in which all the given elements have types that are not vertices of any cone. Let us temporarily call this a *basic* free algebra. Let us also say that a term in a free algebra that has no projection in it is a *proper term.*

9.7.3 Proposition *In a basic free algebra, every element has a unique expression as a proper term.*

Proof. As noted above, we can suppose that each operation in the sketch has a base node as its codomain. Then we can apply the construction of 7.6.5 using only FP–1 and FP–3. In this way we build up words in which no

instance of a projection appears. From FP–1, it follows that the elements we construct are closed under the operations of the sketch and from FP–3 the elements of the product types are products of those of the base types. This is a model of the sketch since no equations are required and it is free since there are no identifications. □

A set of terms in a free algebra can be viewed as a map from a free algebra. For example, the pair of terms $\{x_1, x_2\}$ in 9.7.1 can be seen as an arrow between free algebras:

$$x : F(\{1,2\}) \rightrightarrows F(\{a,b,u,v,w\})$$

Here 1 and 2 both represent variables of type S and x is the unique arrow that takes 1 to $m(t(u,v),a)$ and 2 to $m(b,t(u,w))$. Similarly, $\{y_1, y_2\}$ can be viewed as the unique arrow y that takes 1 to $m(a,a)$ and 2 to $m(b,a)$. A unification is then an arrow which coequalizes these two arrows. In particular, in this example the arrow d in the theorem below is the substitution (9.11).

9.7.4 Proposition *Let $h : F(X) \longrightarrow F(W)$ be a map in $\mathscr{C}$. Then either there is a subset $W_0 \subseteq W$ that factors h or h is an epimorphism in $\mathscr{C}$.*

Proof. Assume that $F(X)$ and $F(W)$ are basic free algebras. For each $x \in X$, $h(x)$ is either an element $w \in W$ or has the form $f(u_1, \ldots, u_n)$ where f is an n-ary operation in the sketch and f is not a projection and $u_1, \ldots, u_n$ are elements of $F(W)$ less complicated than $f(u_1, \ldots, u_n)$. If each $u_1, \ldots, u_n$ is represented by a proper term, then $f(u_1, \ldots, u_n)$ is also a proper term. We say that $w \in W$ is involved in $h(x)$ either if $h(x) = w$ or, by induction, if it is involved in one of $u_1, \ldots, u_n$. If some element of W is not involved in any of the $h(x)$ for $x \in X$, then the first alternative holds. So suppose that every element of W is involved in at least one $h(x)$. Suppose that $k \neq l : F(W) \longrightarrow F(T)$ where we may also suppose that $F(T)$ is a basic free algebra. Then for some $w \in W$, we must have $k(w) \neq l(w)$ since two homomorphisms that agree on generators are equal. If we suppose, as inductive hypothesis, that at least one $k(u_i) \neq l(u_i)$ it follows that

$$\begin{aligned} k(f(u_1, \ldots, u_n)) &= f(k(u_1), \ldots, k(u_n)) \\ &\neq f(l(u_1), \ldots, l(u_n)) \\ &= l(f(u_1, \ldots, u_n)) \end{aligned}$$

since homomorphisms cannot introduce projections and the representations are unique. But this shows that $k \neq l$ implies that $k \circ h \neq l \circ h$ so that h is an epimorphism in $\mathscr{C}$. □

9.7.5 Theorem *Let $\mathscr{S}$ be a free FP sketch and $\mathscr{C}$ be the full subcategory of free algebras in the category of models of $\mathscr{S}$ in* **Set**. *Suppose X, Y and Z are finite sets typed by the objects of $\mathscr{S}$. Suppose there is a diagram in $\mathscr{C}$,*

$$F(X) \underset{d^1}{\overset{d^0}{\rightrightarrows}} F(Y) \xrightarrow{d} F(Z) \tag{9.13}$$

with $d \circ d^0 = d \circ d^1$. Then the arrows d^0 and d^1 have a coequalizer in $\mathscr{C}$ (so that this coequalizer is also a free algebra).

Proof. This proof is adapted from that of [Rydeheard and Burstall, 1985]. The proof makes use of the fact that F preserves sums, which follows from Theorem 13.3.7.

We can suppose that the free algebras involved are basic. The proof will be by a double induction. The first induction is on the complexity of the arrows d^0 and d^1 and the second on the number of elements of X. By the depth of the induction, we mean the largest number of steps necessary to express each $d^0(x)$ and each $d^1(x)$, for $x \in X$ as proper terms using FP–1 and FP–3 as explained above.

We consider first the case that X consists of a single element x of some type. Let $d^0(x) = f^0(a_1, \ldots, a_n)$ and $d^1(x) = f^1(b_1, \ldots, b_m)$ where f^0 and f^1 are operations of the sketch and $a_1, \ldots, a_n, b_1, \ldots, b_m$ are proper terms. Note that it is possible that $n = m = 0$.

Now we have

$$d \circ d^0(x) = df^0(a_1, \ldots, a_n) = f^0(da_1, \ldots, da_n)$$

and similarly,

$$d \circ d^1(x) = f^1(db_1, \ldots, db_m)$$

It can be seen by induction that da_1, $\ldots$, da_n and db_1, $\ldots$, db_m are still proper since the value simply percolates upward from the values on the generators $y \in Y$ and projections are never introduced. Setting the terms equal and using Proposition 9.7.3, we conclude that

$$m = n, \qquad f^0 = f^1 \qquad \text{and} \qquad da_i = db_i \text{ for } i = 1, \ldots, n \tag{9.14}$$

Now d preserves type so the type of a_i is the same as that of b_i. Let W be a set $\{w_1, \ldots, w_n\}$ where w_i is a variable of the same type as a_i. (If a_i and a_j are of the same type for some distinct i and j, w_i and w_j are nevertheless distinct variables.) We have arrows $e^0, e^1 : F(W) \rightrightarrows F(Y)$ given by $e^0(w_i) = a_i$ and $e^1(w_i) = b_i$. This gives us a commutative diagram

$$F(W) \underset{e^1}{\overset{e^0}{\rightrightarrows}} F(Y) \xrightarrow{d} F(Z) \tag{9.15}$$

The complexities of e^0 and e^1 are evidently less than those of d^0 and d^1 and so we can suppose that there is a coequalizer diagram in $\mathscr{C}$

$$F(W) \underset{e^1}{\overset{e^0}{\rightrightarrows}} F(Y) \xrightarrow{e} F(V) \tag{9.16}$$

Let $h : F(\{x\}) \longrightarrow F(W)$ take x to $f^0(w_1, \ldots, w_n) = f^1(w_1, \ldots, w_n)$ by (9.14). Then h is epimorphic by Proposition 9.7.4, so by Exercise 1,

$$F(\{x\}) \underset{e^1 \circ h}{\overset{e^0 \circ h}{\rightrightarrows}} F(Y) \xrightarrow{e} F(V) \tag{9.17}$$

is a coequalizer diagram. This completes the proof when X has one element.

Now we suppose that X has more than one element and that the conclusion is true for any set with fewer elements. Let $x \in X$ be arbitrary. Since we are supposing that $d \circ d^0 = d \circ d^1$, we can compose both arrows with the inclusion $g : F(\{x\}) \longrightarrow F(X)$ induced by the inclusion of $\{x\}$ into X and use the single element case to get a coequalizer

$$F(\{x\}) \underset{d^1 \circ g}{\overset{d^0 \circ g}{\rightrightarrows}} F(Y) \xrightarrow{e} F(V) \tag{9.18}$$

Now we let $h : X - \{x\} \longrightarrow X$ be the inclusion and invoke the second induction on the number of elements of X to conclude that there is also a coequalizer diagram

$$F(X - \{x\}) \underset{e \circ d^1 \circ h}{\overset{e \circ d^0 \circ h}{\rightrightarrows}} F(V) \xrightarrow{c} F(U) \tag{9.19}$$

It now follows from this, the fact that F preserves sums, and Exercise 3 that $c \circ e$ is the coequalizer of the original pair of arrows. □

This proof is made into a formal algorithm in [Rydeheard and Burstall, 1986]. See also [Goguen, 1988].

It should be noted that the coequalizer in $\mathscr{C}$ constructed in the preceding proof need not be the coequalizer in the category of all models of $\mathscr{S}$.

9.7.6 Exercises

1. Show that in any category if

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{c} C$$

is a coequalizer diagram and $e : D \longrightarrow A$ is an epimorphism, then

$$D \underset{g \circ e}{\overset{f \circ e}{\rightrightarrows}} B \xrightarrow{c} C$$

is a coequalizer diagram.

2. Show that in any category in which the sums exist, if for $i = 1, 2$

$$A_i \underset{g_i}{\overset{f_i}{\rightrightarrows}} B_i \xrightarrow{c_i} C_i$$

are both coequalizer diagrams, then so is

$$A_1 + A_2 \underset{g_1 + g_2}{\overset{f_1 + f_2}{\rightrightarrows}} B_1 + B_2 \xrightarrow{c_1 + c_2} C_1 + C_2$$

3. Prove: If d^0, $d^1 : A_1 + A_2 \longrightarrow B$ is a parallel pair, with $u_i : A_i \longrightarrow A_1 + A_2$ the coproduct injections for $i = 1, 2$, if $e : B \longrightarrow C$ is a coequalizer of $d^0 \circ u_1$ and $d^1 \circ u_1$ and if $c : C \longrightarrow D$ is a coequalizer of $e \circ d^0 \circ u_2$ and $e \circ d^1 \circ u_2$, then $c \circ e : B \longrightarrow D$ is a coequalizer of d^0 and d^1.

9.8 Properties of factorization systems

Recall the definition of factorization systems from Section 2.10 of Chapter 2. In this section, we use a kind of limit to infer the existence of one kind of factorization system and also an important property of factorization systems.

9.8.1 Intersections If C is an object of $\mathscr{C}$ and $C_1 \subseteq C$ and $C_2 \subseteq C$ are two subobjects, then the **intersection** of C_1 and C_2 is defined to be a pullback, if it exists:

$$\begin{array}{ccc} C_0 & \longrightarrow & C_1 \\ \downarrow & & \downarrow \\ C_2 & \longrightarrow & C \end{array}$$

This is not the only possible definition. One might be tempted to define the intersection as the meet, if it exists, in the poset of subobjects of C. When the intersection exists, it is the meet in the subobject poset, but it is possible for that meet to exist without being the pullback, see Exercise 2. But the meet is not useful if it does not have the universal mapping property of the pullback.

Similarly, if $\{C_i \mid i \in I\}$ is any collection of subobjects of C, we define the intersection as the limit, if it exists, of the following diagram. Let $\mathscr{G}$ denote the graph with one node i for each $i \in I$ together with one more node we will call ∞. The only arrows are one arrow $i \to \infty$ for each $i \in I$. Define the diagram $D : \mathscr{G} \to \mathscr{C}$ by $Di = C_i$ for $i \in I$, $D\infty = C$ and $D(i \to \infty)$ is the inclusion. As before, a limit of such a diagram, if it exists, is the meet in the poset of subobjects of C, but such a meet could exist without being a limit.

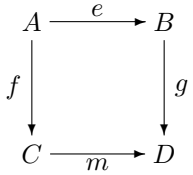
We say that $\mathscr{C}$ has **wide intersections** if for every object C and every class of subobjects $\{C_i\}$ of C, there is an intersection.

The reader may have noticed that we did not say every *set* of subobjects, but every *class*, allowing for the possibility that the collection of subobjects of some object might not be a set (see Section 1.1.3). In the familiar categories that arise in mathematics and computer science every object has only a set of subobjects and the distinction we are making here is empty. Readers will lose little if they replace "class" by "set" in the definition above. They will have to add to the theorem below the requirement that each object have a set of subobjects.

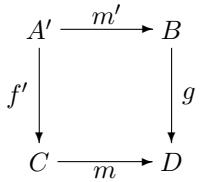
9.8.2 Theorem *Suppose $\mathscr{C}$ is a category with wide intersections in which every finite diagram has a limit. Then there is a factorization system $\mathscr{E}/\mathscr{M}$ in which $\mathscr{E}$ consists of the class of extremal epimorphisms and $\mathscr{M}$ consists of the class of monomorphisms.*

Proof. Given $f : A \to B$, let $C \subseteq B$ denote the intersection of the class of all subobjects of $B_i \subseteq B$ for which there is a factorization of f as $A \to B_i \to B$. The universal mapping property of intersection implies that there is a factorization of f as $A \xrightarrow{e} C \xrightarrow{m} B$. If the arrow $e : A \to C$ were not epic, there would be two arrows, say $g \neq h : C \to D$ with $g \circ e = h \circ e$. But then the equalizer of g and h would be a subobject of B strictly smaller than C that factors e and hence f, a contradiction. The fact that e is extremal follows since if it factored as $e = n \circ g$ with n monic and not an isomorphism, then the image of $m \circ n$ would be a subobject of B even smaller than B_0 that factored f. This shows that every arrow factors as an extremal epic

followed by a monic. Now suppose that



is a commutative square with $e \in \mathscr{E}$ and $m \in \mathscr{M}$. Let

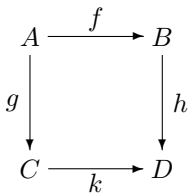


be a pullback. Then m' is monic since a pullback of a monic is always monic (see 9.3.4). There is a unique arrow $h : A \longrightarrow A'$ such that $m' \circ h = e$ and $f' \circ h = f$. The fact that e is an extremal epic implies that m' is an isomorphism and then $f' \circ m'^{-1}$ is the required diagonal fill-in since $f' \circ m'^{-1} \circ e = f' \circ m'^{-1} \circ m' \circ h = f' \circ h = f$ and $m \circ f' \circ m^{-1} = g \circ m' \circ m'^{-1} = g$. If k were another diagonal fill-in, then from $m \circ k = g$ and $m \circ f' \circ m^{-1} = g$ we can cancel the monic m to conclude that $k = f' \circ m^{-1}$. □

Of course, there is a dual theorem with epics and extremal monics in a wide cocomplete category.

The proof of the theorem above used the fact that a pullback of a monic is monic. This property is valid for the arrows in the $\mathscr{M}$ part of any factorization system.

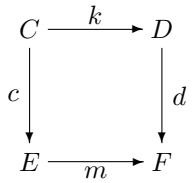
9.8.3 Proposition *Suppose $\mathscr{C}$ is a category and $\mathscr{E}/\mathscr{M}$ is a factorization system on $\mathscr{C}$. Then in any commutative square*



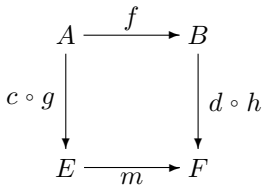
(9.20)

if the square is a pullback, then $k \in \mathscr{M}$ implies $f \in \mathscr{M}$; if the square is a pushout, then $f \in \mathscr{E}$ implies $k \in \mathscr{E}$.

Proof. Assume that diagram (9.20) is a pushout and that $f \in \Sigma$. We prove the second assertion using Proposition 2.10.8; the first is dual. Suppose we have a commutative square



with $m \in \mathscr{M}$. The diagonal fill-in in the square



implies the existence of an arrow $l : B \longrightarrow E$ such that $l \circ f = c \circ g$ and $m \circ l = d \circ h$. The latter equation, in conjunction with the universal mapping properties of pushouts implies the existence of an arrow $b : D \longrightarrow E$ such that $b \circ h = l$ and $b \circ k = c$. The latter equation is one of the diagonal triangles. We show the second, $m \circ b = d$ by composing both sides with h and k, using the uniqueness of arrows from a pushout. We have $m \circ b \circ h = m \circ l = d \circ h$ and $m \circ b \circ k = m \circ c = d \circ k$. □

9.8.4 Exercises

1. Show that when $\mathscr{C}$ has cokernel pairs, the converse of Proposition2.10.7 of Chapter 2 is valid; that is, the left cancellability of $\mathscr{M}$ implies that every arrow in $\mathscr{E}$ is an epimorphism.

2. a. Show that an intersection, if it exists, of a collection of subobjects of an object is also its meet in the subobject poset.

b. Show by counter-example (a finite category should suffice) that the meet of subobjects can exist without being their intersection.

10
More about sketches

This chapter develops the concept of sketch in several ways. The first three sections describe a generalization of FP theories called FL theories which allow the use of equalizers, pullbacks and other limits in the description of a structure. These theories have expressive power which includes that of universal Horn theories.

The last section gives further generalizations of the concept of sketch. These generalizations are described without much detail since they do not appear to have many applications (yet!) in computer science.

This chapter is needed only for Chapter 11, except that the sketch for categories of Section 10.1.5 is required for Section 15.7.

10.1 Finite limit sketches

A cone is called finite if its shape graph is finite, meaning that it has only finitely many nodes and arrows.

10.1.1 Definition A **finite limit** or **FL sketch** $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$ is a finite graph together with a finite set $\mathscr{D}$ of diagrams and a finite set $\mathscr{L}$ of finite cones. A **model** of $\mathscr{S}$ is a model of $\mathscr{G}$ that takes all the diagrams in $\mathscr{D}$ to commutative diagrams and all the cones in $\mathscr{L}$ to limit cones.

For historical reasons, FL sketches are also known as **left exact sketches** or **LE sketches**.

In 7.2.8 we described the way the choice of products and terminal objects in a model to represent the vertices of cones in a sketch were irrelevant but could result in the technicality that, for example, the set representing the product might not actually be a set of ordered pairs. The same sort of statement is true of models of FL theories. In particular, in a model the equalizer of parallel arrows need not be a subset of their common domain.

FL sketches allow the specification of structures or data types with sorts that include equationally specified subsorts, by using equalizers. More generally, you can specify an operation whose domain, in a model, will be an equationally defined subobject of another sort. FL theories can express anything expressible by universal Horn theories, but in general FL theories are

more powerful (see [Barr, 1989] and [Adámek and Rosičky, 1994], pages 209-210). For example, small categories and functors form the category of models of an FL theory (which we give in 10.1.5 below) but not of a universal Horn theory. Categories of models of FL sketches can be described axiomatically as **locally finitely presentable categories** (see [Gabriel and Ulmer, 1971] and [Adámek and Rosičky, 1994]).

In practice it is generally sufficient to restrict the types of cones to a few simple types, products, pullbacks and equalizers. In principle, an FL sketch can always be replaced by one which has an equivalent category of models and which has only these three types of cones, but there might be some case in which this is not the most efficient approach.

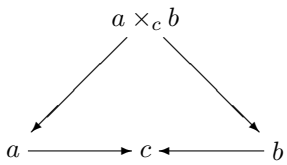
10.1.2 Other approaches In the introduction to Chapter 7, we mentioned three approaches to formalization: logical theories, signatures and equations, and sketches. Systems equivalent to FL sketches have been developed for both the other approaches. Coste [1976], Cartmell [1986] and McLarty [1986] generalize logical systems and Reichel [1987] generalize signatures. The book by Reichel has many examples of applications to computer science.

10.1.3 Notation for FL sketches We extend the notational conventions in Section 7.3 to cover products, pullbacks and equalizers.

First, the notation of N–2 in 7.3.1 using product projections is extended to cover the arrows from the limit to any of the nodes in the diagram: that is they are all denoted by an appropriate p; moreover, the notation of N–4 in 7.3.1 is extended to arrows into the limit in terms of the composite with the projections. See Example 10.1.5 below of the sketch of categories to see how this is used. Of course, it is sometimes necessary to be more explicit than this.

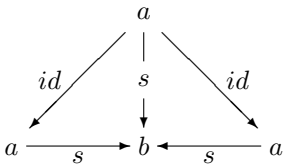
We add the following to the notational conventions of Section 7.3.

N–8 If we have a node labeled $a \times_c b$ this implies the existence of a cone



Of course, this notation is not self-contained since it is necessary to specify the arrows $a \to c \leftarrow b$. In many cases, these arrows are clear, but it may be necessary to specify them explicitly.

N–9 If we have an arrow labeled $s : a \rightarrowtail b$ (recall from 2.8.2 that in a category this notation means an arrow that is a monomorphism), then we are implicitly including a cone

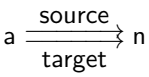


This notation makes sense because of Theorem 9.3.3.

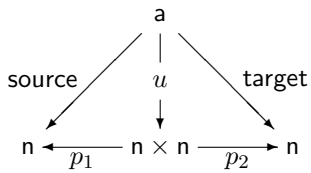
We could go on and turn our notational conventions into a formal language, but have chosen not to do so because we do not know what it would add to the theory. For us, they remain a set of notational conventions; the real object of study is the whole sketch or else the theory it generates.

10.1.4 The sketch for simple graphs Here is a simple example that shows how the added power of FL sketches can be used to define a familiar type of structure. A **simple graph** is a graph with the property that for any pair (a, b) of nodes there is no more than one arrow with source a and target b.

The sketch for graphs has the graph



and no cones or diagrams. We can modify it to get a sketch for simple graphs by adding an object $\mathsf{n} \times \mathsf{n}$ and the necessary discrete cone to make it a formal product, an arrow $u : \mathsf{a} \rightarrowtail \mathsf{n} \times \mathsf{n}$ (hence adding a formal pullback as in N–9 making it formally monic) and also the diagram (*not* cone)



The effect of this is to make the monic arrow u become $\langle \mathsf{source}, \mathsf{target} \rangle$ in a model, so that no two arrows can have the same source-target pair.

10.1.5 The sketch for categories We give here an FL sketch whose models in the category of sets are small categories and arrows between the models are functors. Models in an arbitrary category $\mathscr{C}$ with finite limits are called **category objects** in $\mathscr{C}$. These are used in Section 15.7.

The sketch has nodes c_0, c_1, c_2 and c_3 which stand for the objects, the arrows, the composable pairs and the composable triples of arrows respectively. The arrows of the sketch are:

$u : c_0 \longrightarrow c_1$

$s, t : c_1 \longrightarrow c_0$

p_1, p_2, $c : c_2 \longrightarrow c_1$

p_1, p_2, $p_3 : c_3 \longrightarrow c_1$

$\langle p_1, p_2\rangle$, $\langle p_2, p_3\rangle$, $\langle p_1, c\rangle$, $\langle c, p_3\rangle : c_3 \longrightarrow c_2$

$\langle u \circ t, \mathrm{id}\rangle$, $\langle \mathrm{id}, u \circ s\rangle : c_1 \longrightarrow c_2$

The intention is that in a **Set**-model, the arrows of the sketch will be interpreted as follows:

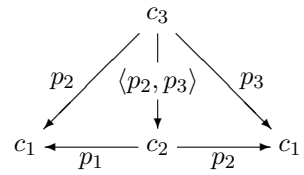
u is the unit function which assigns to each object its identity,

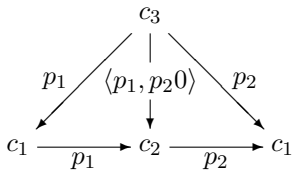
s and t are the source and target functions from the set of arrows to the set of objects, and

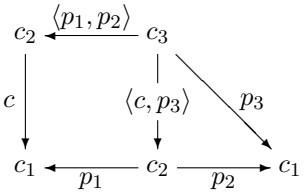
c is the function which takes a composable pair of arrows to its composite.

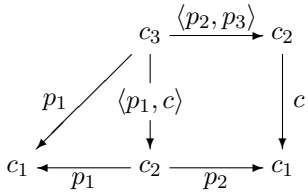
The remaining arrows of the sketch are projections from a limit or are interpreted as arrows to a limit with specified projections.

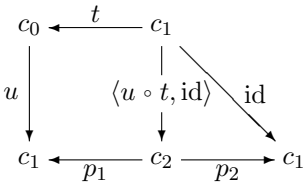
The diagrams are

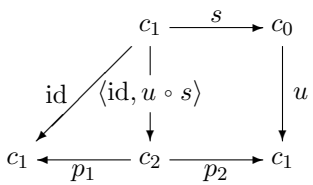


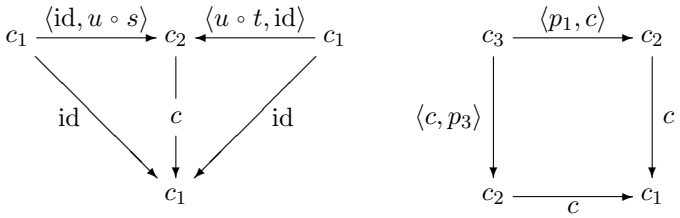




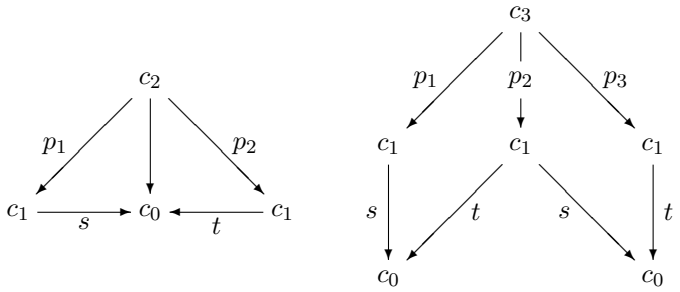








Finally, there are two cones that say that c_2 and c_3 are interpreted as the objects of composable pairs and triples or arrows, respectively.



This sketch is the **sketch of categories** and is one of the simpler FL sketches around.

The category of models (in **Set**) of an FP sketch must be regular (see [Barr and Wells, 1985], Theorem 1 of Section 8.4) and it can be proved that the category of categories and functors is not regular (see Exercise 6 of Section 9.6). It follows that categories and functors cannot be the models of an FP sketch.

Lawvere [1966] described another structure whose models are categories. Let $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$ and $\mathbf{4}$ denote the total orders with one, two, three and four elements, considered as categories in the usual way a poset is. Let $\mathscr{L}$ denote the opposite category. In that opposite category, it turns out that $\mathbf{3} = \mathbf{2} \times_{\mathbf{1}} \mathbf{2}$ and $\mathbf{4} = \mathbf{2} \times_{\mathbf{1}} \mathbf{2} \times_{\mathbf{1}} \mathbf{2}$. Then **Cat** is the category of limit-preserving functors of $\mathscr{L}$ into **Set** with natural transformations as arrows. It follows that the sketch whose objects and arrows are those of $\mathscr{L}$, whose diagrams are the commutative diagrams of $\mathscr{L}$ and cones are the limit cones of $\mathscr{L}$ is another sketch, closely related in fact to the one above, whose category of models is **Cat**.

We show how binary trees can be described as models of an FL sketch in the next section.

10.1.6 Exercises

1. A **groupoid** is a category in which every arrow is an isomorphism. Explain how to modify the sketch for categories to get a sketch for groupoids.

2. Show that in the category of sets, the definition of homomorphism between two models of the sketch for categories gives the usual definition of functor.

10.2 Initial term models of FL sketches

Like FP sketches, FL sketches always have initial models. The construction 7.6.5 produced an initial term algebra for each finite FP sketch. A modification of the last three rules in 7.6.5 is sufficient to construct an initial term algebra for each finite FL sketch. This will be described below.

A different construction is in [Barr, 1986b], where it is proved that in fact FL sketches have free algebras on any typed set X (see Section 13.2). Volger [1987] gives a logic-based proof for the special case of Horn theories (see [Volger, 1988] for applications).

The modification of 7.6.5 is required by the fact that the base diagram D of a cone in $\mathscr{L}$ need not be discrete in the case of general limits; that is, the shape graph $\mathscr{I}$ of the diagram D in Section 9.2 may have nontrivial arrows $u : i \longrightarrow j$. A limit of such a cone in the category of sets is not just any tuple of elements of the sets corresponding to the nodes of $\mathscr{I}$, but only tuples which are compatible with the arrows of $\mathscr{I}$.

10.2.1 Precisely, suppose $E : \mathscr{I} \longrightarrow \mathbf{Set}$ is a finite diagram (we use the letter E instead of D to avoid confusion below). A **compatible family** of elements of E is a sequence $(x_1, \ldots, x_n)$ indexed by the nodes of $\mathscr{I}$ for which

C–1 $x_i \in E(i)$ for each node i.

C–2 If $u : i \longrightarrow j$ in $\mathscr{I}$, then $E(u)(x_i) = x_j$.

An initial term algebra of an FL sketch $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$ is then the least model satisfying the following requirements.

FL–1 If $f : a \longrightarrow b$ is an arrow of $\mathscr{G}$ and $[x]$ is an element of $I(a)$, then $[fx] \in I(b)$ and $I(f)[x] = [fx]$.

FL–2 If $(f_1, \ldots, f_m)$ and $(g_1, \ldots, g_k)$ are paths in a diagram in $\mathscr{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \ldots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \ldots \circ Ig_k)[x]$$

in $I(b)$.

FL–3 If $D : \mathscr{I} \to \mathscr{G}$ is a diagram, $([x_1], \ldots, [x_n])$ is a compatible family of elements of $I \circ D$ and $p : q \to D$ is a cone over D in $\mathscr{L}$, then $[C(x_1, \ldots, x_n)]$ is an element of $I(q)$.

FL–4 If $p : q \to D$ and $([x_1], \ldots, [x_n])$ are as in FL–3, and $x'_1, \ldots, x'_n$ is a compatible family of elements of $I \circ D$ for which $[x_i] = [x'_i]$ for $i = 1, \ldots, n$, then

$$[C(x_1, \ldots, x_n)] = [C(x'_1, \ldots, x'_n)]$$

FL–5 If $p : q \to D$ and $([x_1], \ldots, [x_n])$ are as in FL–3, then for $i = 1, \ldots, n$

$$[p_i C(x_1, \ldots, x_n)] = [x_i]$$

FL–6 If $p : q \to D$ is a cone over D in $\mathscr{L}$ and $x \in I(q)$, then

$$[x] = [C(p_1 x, \ldots, p_n x)]$$

Compatibility implies that for any arrow $u : i \to j$ of $\mathscr{I}$,

$$I(D(u))([x_i]) = [x_j]$$

As in 7.6.5, it follows that

$$I(p_i)([C(x_1, \ldots, x_n)]) = [x_i]$$

for each i.

10.2.2 Binary trees We describe an FL sketch whose initial term algebra is the set of binary trees of integers. We gave an FD sketch for binary trees in 8.3.11; the sketch given here illustrates a different approach to the problem that operations such as taking the datum at the root or producing the left or right subtrees are not defined on the empty tree.

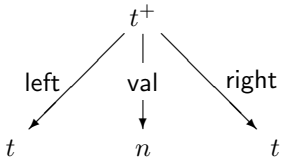
We have the following basic nodes in the sketch: 1, t, t^+, b, n. These should be thought of as representing the types of binary trees, nonempty binary trees, the Boolean algebra 2 and the natural numbers, respectively. We have the following operations:

$\mathsf{empty} : 1 \to t$	$\mathsf{empty?} : t \to b$	$\mathsf{incl} : t^+ \to t$
$\mathsf{val} : t^+ \to n$	$\mathsf{left} : t^+ \to t$	$\mathsf{right} : t^+ \to t$
$\mathsf{zero} : 1 \to n$	$\mathsf{succ} : n \to n$	$\mathsf{true} : 1 \to b$
$\mathsf{and} : b \times b \to b$	$\mathsf{not} : b \to b$	

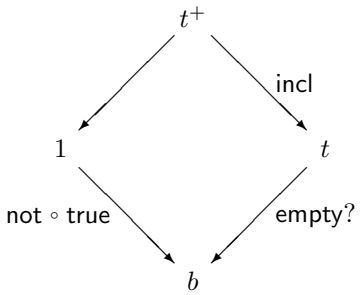
The intended meaning of these operations is as follows: the constant $\mathsf{empty}\langle\rangle$ is the empty tree; empty? is the test for whether a tree is the empty tree; incl is the inclusion of the set of nonempty trees in the set of trees;

$\mathsf{val}(T)$ is the datum stored at the root of the nonempty tree T; $\mathsf{left}(T)$ and $\mathsf{right}(T)$ are the right and left branches (possibly empty) of the nonempty tree T, respectively. The remaining operations are the standard operations appropriate to the natural numbers and the Boolean algebra 2.

We require that

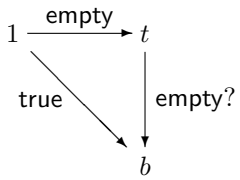


and



(10.1)

be cones and that



be a diagram.

In the cone (10.1), there should be an arrow from the vertex to the node b. It will appear in a model as either of the two (necessarily equal) composites. Since its value is forced, it is customary to omit it from the cone; however, there actually does have to be such an arrow there to complete the cone (and the sketch). In omitting it, we have conformed to the standard convention of showing explicitly only what it is necessary to show.

As in 8.3.11, the existence of the first cone says that every nonempty tree can be represented uniquely as a triplet

$$(\mathsf{left}(T), \mathsf{val}(T), \mathsf{right}(T))$$

The fact that (10.1) is a cone requires that in a model M, $M(t^+)$ be exactly the subset of $M(t)$ of those elements which evaluate to false under $M(\mathsf{empty?})$.

10.2.3 Exercise

1. In the sketch for binary trees, show that for any model M, $M(\mathsf{incl})$ is an injective function. (Hint: use Diagram (10.1).)

10.3 The theory of an FL sketch

Just as in the case of linear and FP sketches, every FL sketch generates a category with finite limits in which it has a universal model.

10.3.1 Theorem *Given any FL sketch $\mathscr{S}$, there is a category $\mathbf{Th_{FL}}(\mathscr{S})$ with finite limits and a model $M_0 : \mathscr{S} \longrightarrow \mathbf{Th_{FL}}(\mathscr{S})$ such that for any model $M : \mathscr{S} \longrightarrow \mathscr{E}$ into a category with finite limits, there is a functor $F : \mathbf{Th_{FL}}(\mathscr{S}) \longrightarrow \mathscr{E}$ that preserves finite limits for which*

(i) *$F \circ M_0 = M$, and*

(ii) *if $F' : \mathbf{Th_{FL}}(\mathscr{S}) \longrightarrow \mathscr{E}$ is another functor that preserves finite limits for which $F' \circ M_0 = M$, then F and F' are naturally isomorphic.*

$\mathbf{Th_{FL}}(\mathscr{S})$ is defined up to equivalence by the following properties:

FLT–1 $\mathscr{T}$ has all finite limits.

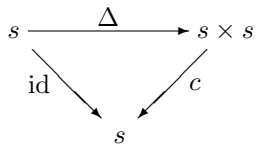
FLT–2 M_0 takes every diagram of $\mathscr{S}$ to a commutative diagram in $\mathscr{T}$.

FLT–3 M_0 takes every cone of $\mathscr{S}$ to a limit cone of $\mathscr{T}$.

FLT–4 No proper subcategory of $\mathscr{T}$ includes the image of M_0 and satisfies FLT-1, FLT-2 and FLT-3.

The comments concerning Theorem 7.5.1 apply here too. See [Barr and Wells, 1985], Section 4.4, Theorem 2 (p. 156). The constructions in [Ehresmann, 1968] and [Bastiani and Ehresmann, 1972] are more general since they allow cones over infinite diagrams.

10.3.2 Example The sketch for semigroups (see Section 7.2) is an FP sketch and therefore an FL sketch whose cones all go to discrete diagrams. The FL theory of this sketch contains a node v that is the limit of this diagram:



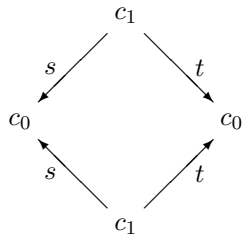
(10.2)

Since M_0 is injective, we write s for $M_0(s)$, $s \times s$ for $M_0(s \times s)$, and so on.

A model M of the sketch induces a limit preserving functor F which must take v to a set. Since v is the equalizer of id_s and $c \circ \Delta$, $F(v)$ must be

the set of elements e of $M(s)$ that satisfy the requirement $ee = e$, that is, the set of idempotents in the semigroup $M(s)$. (This set is not in general a subsemigroup.)

10.3.3 Example The FL theory of the sketch for categories (see 10.1.5) contains a node v that is the limit of the diagram

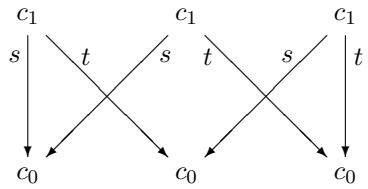


(10.3)

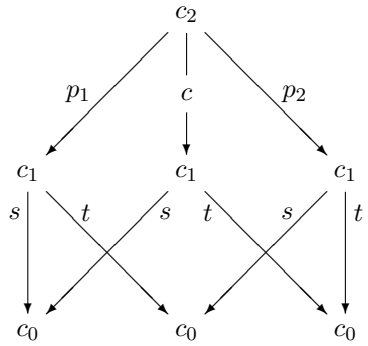
If $\mathscr{C}$ is a small category, hence a model of the sketch, the induced limit-preserving functor F takes v to the set of all parallel pairs of arrows in $\mathscr{C}$.

10.3.4 Exercises

1. The following diagram in the sketch for categories has a limit v in the FL theory of that sketch. What is its value in a model?



2. Same question as Exercise 1 for the following diagram:



3. Show that any model M of a sketch $\mathscr{S}$ is isomorphic to a model M' for which $M'(c)$ and $M'(d)$ have no elements in common if $c \neq d$.

10.4 General definition of sketch

Although we will not be using the most general notion of sketch, we give the definition here.

10.4.1 Definition A **sketch** $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ consists of a graph $\mathscr{G}$, a set $\mathscr{D}$ of diagrams in $\mathscr{G}$, a set $\mathscr{L}$ of cones in $\mathscr{G}$ and a set $\mathscr{K}$ of cocones in $\mathscr{G}$.

10.4.2 Definition A model M of a sketch $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ in a category $\mathscr{A}$ is a homomorphism from $\mathscr{G}$ to the underlying graph of $\mathscr{A}$ that takes every diagram in $\mathscr{D}$ to a commutative diagram, every cone in $\mathscr{L}$ to a limit cone and every cocone in $\mathscr{K}$ to a colimit cocone.

10.4.3 Definition Let M, N be models of a sketch $\mathscr{S}$ in a category $\mathscr{A}$. A **homomorphism of models** $\alpha : M \to N$ is a natural transformation from M to N.

Sketches in general do not have initial algebras, or even families of initial algebras. What they do have is 'locally free diagrams' as described by Guitart and Lair [1981, 1982].

10.4.4 Regular sketches Chapter 8 of [Barr and Wells, 1985] describes special classes of sketches with cones and cocones that have in common the fact that their theories are embedded in a topos, and so inherit the nice properties of a topos. (Toposes are discussed in Chapter 15.) These are the regular sketches, coherent sketches and geometric sketches. We describe regular sketches here to illustrate the general pattern. (Note: The French school uses the phrase "regular sketch" for a sketch in which no node is the vertex of more than one cone.)

10.4.5 An arrow $f : A \to B$ is a regular epimorphism if and only if there is a cocone diagram

$$C \underset{h}{\overset{g}{\rightrightarrows}} A \xrightarrow{f} B$$

It follows that the predicate of being a regular epimorphism can be stated within the semantics of a finite limits sketch. Note that an arrow can be required to become an epimorphism (not necessarily regular) in a model by using the dual of Theorem 9.3.3.

Let us say that a **regular cocone** is one of the form

$$c \rightrightarrows a \to b$$

10.4.6 Definition A **regular sketch** $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ consists of a graph, a set of diagrams, a set of finite cones and a set of regular cocones.

There is one undesirable feature to the definition above. The introduction of a regular sketch requires the introduction of a sort c and two arrows $c \rightrightarrows a$. A model will have to provide a value for c as well as the two arrows. This is generally irrelevant information that one would not normally want to have to provide. Worse, the definition of natural transformation is such that arrows between models will have to preserve this additional information and that is definitely undesirable. The way in which this is usually dealt with is by adding, with each cocone

$$c \underset{d^1}{\overset{d^0}{\rightrightarrows}} a \xrightarrow{d} b$$

the cone

$$\begin{array}{ccc} c & \xrightarrow{d^0} & a \\ {\scriptstyle d^1}\downarrow & & \downarrow{\scriptstyle d} \\ a & \xrightarrow[d]{} & b \end{array}$$

It follows from Exercise 5 of Section 9.4 that if d has a kernel pair, then d is a regular epimorphism if and only if it is the coequalizer of that kernel pair. The addition of this cone thus adds no new data, at least in the case of models in a category with finite limits and coequalizers. The preservation of the kernel pair and the two arrows by homomorphisms of models is automatic, given the universal mapping properties of limits.

The point of these considerations is that a regular sketch can be described as one in which one can specify any kind of finite limits and that any particular arrow is a coequalizer of some pair, without having to specify (or preserve in an arrow between models) what pair of arrows it coequalizes.

> One of the things we want a sketch to do is minimize the number of items – or amount of information – that has to be specified and maximize the amount that is implicit in the sketch. Thus we want to be able to say that an arrow is a regular epimorphism (that is, coequalizer of some pair of arrows) without specifying what pair it coequalizes. In the preceding discussion, we described a way to do this: to observe that a regular epi is a coequalizer of its own kernel pair and that the kernel pair can be calculated. There is a price to pay in solving the problem in this way, since this works only when you form models in a category that has that kernel pair. An alternative approach would be to simply add regular epi to the list of primitives that a sketch can make use of, so that one could require than an arrow be a regular epi, without having to specify the arrows it coequalizes, in a category that may or may not have pullbacks. This is an example of expanding the concept of sketch to allow specification of constructions other than commutative diagrams, limits

and colimits. (References to generalized sketches are given at the end of this chapter.)

As an example, we construct a sketch for weakly reflexive graphs – graphs with the property that there is at least one loop on each node. Morphisms will be ordinary morphisms of graphs. (So the models should be the full subcategory of **Grf** of weakly reflexive graphs.) The problem that arose in 4.6.9 that homomorphisms had to take certain specific loops to other specific loops does not arise in this construction.

We take the sketch for graphs with nodes n and a and arrows $\mathsf{source} : \mathsf{a} \longrightarrow \mathsf{n}$ and $\mathsf{target} : \mathsf{a} \longrightarrow \mathsf{n}$. We add a new type r with a cone

$$r \longrightarrow \mathsf{a} \underset{\mathsf{target}}{\overset{\mathsf{source}}{\rightrightarrows}} \mathsf{n}$$

and a cocone to express that the composite $r \longrightarrow \mathsf{a} \longrightarrow \mathsf{n}$ (the latter arrow being either source or target) is regular epic. In a model, r becomes the set of loops. The fact that the arrow $r \longrightarrow \mathsf{a} \longrightarrow \mathsf{n}$ maps surjectively to the nodes implies that there is at least one loop at each node. A homomorphism from M to M' must take $M(r)$ to $M'(r)$ as a set, but there is no particular loop in $M(r)$ that is distinguished by the construction.

10.4.7 Theories of sketches with colimits. The reader may have noticed that in Chapter 7, we discussed FP sketches, their initial algebras and their theories, and in Chapter 8 we discussed FD sketches and their initial algebras. We did not discuss theories for FD sketches. In fact, theories exist for FD sketches, and indeed for all sketches. This is because sketches can themselves be modeled by an FL theory, and the theory of a sketch can then be realized as an initial algebra (see Section 10.2). See [Wells, 1990], [Bagchi and Wells, 1997a]. The theories of many types of sketches can be constructed as subcategories of toposes; this is done in [Barr and Wells, 1985], Chapters 4 and 8. Other constructions of theories for special kinds of sketches are given in [Ehresmann, 1968], [Bastiani and Ehresmann, 1972], [Peake and Peters, 1972], [Kelly, 1982b]. [Bagchi and Wells, 1997a].

Here, we state the universal properties for theories of FD sketches for models in categories with finite products and finite disjoint universal sums. The resulting theory has these properties, too, and is embedded in a topos. The proof by initial algebras just mentioned does not give such an embedding.

10.4.8 Theorem *Let $\mathscr{S}$ be an FD sketch. Then there is a category denoted $\mathbf{Th_{FD}}(\mathscr{S})$ with finite disjoint universal sums and a model $M_0 : \mathscr{S} \longrightarrow \mathbf{Th_{FD}}(\mathscr{S})$ such that for any model $M : \mathscr{S} \longrightarrow \mathscr{E}$ into a category with finite*

products and finite disjoint universal sums, there is a functor $F : \mathbf{Th_{FD}}(\mathscr{S}) \longrightarrow \mathscr{E}$ *that preserves finite products and finite sums for which*

(i) $F \circ M_0 = M$, *and*

(ii) *If* $F' : \mathbf{Th_{FD}}(\mathscr{S}) \longrightarrow \mathscr{E}$ *is another functor that preserves finite products and finite sums for which* $F' \circ M_0 = M$, *then* F *and* F' *are naturally isomorphic.*

A proof may be found in [Barr and Wells, 1985], Proposition 1 of Section 8.2 (FD sketches are called FS sketches there) and in [Bagchi and Wells, 1997a].

The situation is similar for regular sketches.

10.4.9 Theorem *Let* $\mathscr{S}$ *be a regular sketch. Then there is a regular category denoted* $\mathbf{Th_{Reg}}(\mathscr{S})$ *and a model* $M_0 : \mathscr{S} \longrightarrow \mathbf{Th_{Reg}}(\mathscr{S})$ *such that, for any model* $M : \mathscr{S} \longrightarrow \mathscr{E}$ *into a regular category, there is a regular functor* $F : \mathbf{Th_{Reg}}(\mathscr{S}) \longrightarrow \mathscr{E}$ *such that*

(i) $F \circ M_0 = M$, *and*

(ii) *if* $F' : \mathbf{Th_{Reg}}(\mathscr{S}) \longrightarrow \mathscr{E}$ *is another regular functor for which* $F' \circ M_0 = M$, *then* F *and* F' *are naturally isomorphic.*

10.4.10 Categories of set-valued models of general sketches can be axiomatized as **accessible categories** (see [Makkai and Paré, 1990], [Adámek and Rosický, 1994].) A precise statement of the relationship between categories of models of first order theories and categories of models of sketches is given in [Adámek and Rosický, 1994] (Theorems 5.35 and 5.44).

Generalizations of the concept of sketch are given by Lair [1987], Wells [1990], Power and Wells [1992], and Makkai [1997]. See also [Barr and Wells, 1992] and [Bagchi and Wells, 1997a].

11
The category of sketches

The first section of this chapter defines the concept of homomorphism of sketches, yielding a category of sketches. In Section 11.2 we describe a formalism for defining parametrized data types using sketch homomorphisms. In Section 11.3 we develop the theory of sketches further, showing that a homomorphism of sketches induces a contravariant functor between the model categories and making contact with Goguen and Burstall's concept of institution.

Section 11.3 requires only Section 11.1 to read. Nothing in this chapter is needed later in the book.

Much more is known about the category of sketches than is mentioned here. It is cartesian closed, for example. A basic study in English of the category of sketches which is oriented toward computer science is given by Gray [1989].

11.1 Homomorphisms of sketches

11.1.1 Let $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ and $\mathscr{S}' = (\mathscr{G}', \mathscr{D}', \mathscr{L}', \mathscr{K}')$ be sketches. A graph homomorphism $F : \mathscr{G} \to \mathscr{G}'$ takes a diagram $D : \mathscr{I} \to \mathscr{G}$ to a diagram $F \circ D : \mathscr{I} \to \mathscr{G}'$ which is called the **image** of D in $\mathscr{G}'$. It takes a cone $p : v \to D$ to a cone $F(p) : F(v) \to F \circ D$ where, for each node a of the shape graph of D, $F(p)_a : F(v) \to F(D(a))$ is defined to be $F(p_a)$. It is defined on cocones similarly.

An arrow or **homomorphism of sketches** $F : \mathscr{S} \to \mathscr{S}'$ is a graph homomorphism from the graph $\mathscr{G}$ to the graph $\mathscr{G}'$ for which, if D is a diagram in $\mathscr{D}$ then $F \circ D$ lies in $\mathscr{D}'$; if $p : v \to D$ is a cone in $\mathscr{L}$ then $F(p)$ is a cone in $\mathscr{L}'$; and if $c : D \to v$ is a cocone in $\mathscr{K}$ then $F(c)$ is a cocone in $\mathscr{K}'$.

Note that if in the sketch $\mathscr{S}$ the sets of diagrams, cones and cocones are all empty, then an arrow from $\mathscr{S}$ to $\mathscr{S}'$ is precisely a graph homomorphism from $\mathscr{G}$ to $\mathscr{G}'$.

11.1.2 Example Let $\mathscr{E}$ denote the impoverished sketch which has one node we will call e and no arrows, diagrams, cones or cocones. Any assignment of e to a node of any sketch is a morphism of sketches.

11.1.3 Example The sketch for graphs has the graph

$$\mathsf{a} \underset{\mathsf{target}}{\overset{\mathsf{source}}{\rightrightarrows}} \mathsf{n}$$

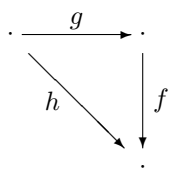
and no cones or diagrams. There is a graph homomorphism from the graph of the sketch of 4.2.17 to the graph just given that takes 0 to a, 1 to n, and u to source. This is a sketch homomorphism since there are no diagrams, cones or cocones in either sketch. (Of course, there is another sketch homomorphism taking u to target.)

11.1.4 Example In 10.1.4, we modifed the sketch for graphs by adding a cone making it the sketch for simple graphs. The inclusion of the sketch for graphs into the sketch for simple graphs is a homomorphism of sketches.

11.1.5 Example A field has an associative binary operation of addition on it, so one would suspect that there is a homomorphism from the sketch for semigroups in 7.2.1 to the sketch for fields in 8.2. That is the case. The homomorphism takes s to f, $s \times s$ to $f \times f$ and $s \times s \times s$ to $f \times f \times f$, and the arrow c to the arrow $+$. The homomorphism must take projection arrows of cones to corresponding projection arrows, and of course the associativity diagram (7.8) then is mapped to the diagram implied by FE–1 of 8.2.

11.1.6 The sketch underlying a category Let $\mathscr{C}$ be a category. There is an underlying sketch of $\mathscr{C}$, call it $\mathrm{Sk}(\mathscr{C})$, whose graph consists of the objects of $\mathscr{C}$ as nodes and the arrows of $\mathscr{C}$ as arrows. This underlying sketch is not in general finite or even small. The commutative diagrams of this sketch are all diagrams that are commutative in $\mathscr{C}$. Similarly we take for cones all those that are limit cones in $\mathscr{C}$ and for cocones all those that are colimit cocones. An arrow from $\mathscr{S}$ to $\mathrm{Sk}(\mathscr{C})$ is then exactly what we have called a model of $\mathscr{S}$ in $\mathscr{C}$ in 10.4.2.

We note that although the composition in $\mathscr{C}$ has been forgotten, it can be completely recovered from the knowledge of which diagrams commute. For example, the information $f \circ g = h$ is equivalent to the information that



commutes so that we could recover the category $\mathscr{C}$ entirely from the underlying sketch (in fact, just from the graph and the commutative triangles).

11.1.7 The category of sketches With the definition of homomorphism of sketches given above, sketches themselves form a category which we will call **Sketch**. By restricting the shape graphs for the diagrams, cones and cocones one obtains many full subcategories of **Sketch**, for example the category of FP sketches, the category of FD sketches, and so on.

One consequence of this is that all constructions that we can carry out in any category make sense in the category of sketches. The particular construction that interests us here is the formation of colimits, particularly pushouts.

11.1.8 Exercise

1. Let $\mathscr{E}$ be the sketch with one node and no arrows, diagrams, cones or cocones. Show that the category of models of $\mathscr{E}$ in a category $\mathscr{C}$ is isomorphic to $\mathscr{C}$. In particular, the category of models of $\mathscr{E}$ in **Set** is isomorphic to **Set**.

11.2 Parametrized data types as pushouts

One thing a theory of data types should do is give a way of saying how the data types of stacks of integers, say, is like the type of stacks of reals or for that matter of stacks of arrays of trees of characters. In other words, we need a way of talking about an abstract stack in a way that leaves it open to fill in the blank corresponding to the thing that it is stacks of. The data type is the parameter.

The way we do this is to describe a sketch for stacks of d where d stands for an abstract data type and then use a pushout construction to identify d with a concrete data type in any particular application. The point is that pushouts are the general way we use to identify things. We give several illustrations.

11.2.1 Abstract stacks Consider the sketch whose graph consists of nodes s, t, d, $d \times s$ and 1. The idea is that s stands for the set of stack configurations, t the set of nonempty stack configurations, and d the data. There are operations $\mathsf{push} : d \times s \longrightarrow t$, $\mathsf{pop} : t \longrightarrow d \times s$, $\mathsf{empty} : 1 \longrightarrow s$ and $\mathsf{incl} : t \longrightarrow s$. In order to state the equations we also have two arrows $\mathrm{id}_{d\times s}$ and id_t which will be forced to be identity arrows.

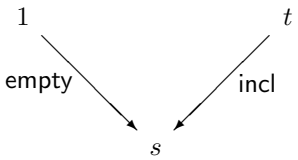
There are four equations (diagrams). Two of them, $\mathrm{id}_{d\times s} = ()_{d\times s}$ and $\mathrm{id}_t = ()_t$, force those arrows to be identities, and the following two, which express the essence of being a stack:

$$\mathsf{pop} \circ \mathsf{push} = \mathrm{id}_{d\times s}$$

and

$$\mathsf{push} \circ \mathsf{pop} = \mathrm{id}_t$$

There are the cones to express 1 as terminal and $d \times s$ as a product, and one cocone



So far, this sketch is not very interesting, since the type of d is still undetermined. In fact its initial model I has $I(d) = I(t) = \emptyset$ and $I(s) = \{\mathsf{empty}\}$. Note that $\{\mathsf{empty}\}$ is not empty. It contains one element which we interpret as representing the empty stack. Since the type d of data is empty, the empty stack is the only kind of stack there is in the model I.

If there were constants in the data type d, then $I(s)$ would be the set of all possible configurations of a stack of data of type $I(d)$, and $I(t)$ would be the set of all possible configurations other than the empty stack.

11.2.2 Stacks of natural numbers In 4.7.7, we described the sketch with two nodes, 1 and n, two operations $\mathsf{zero} : 1 \longrightarrow n$ and $\mathsf{succ} : n \longrightarrow n$. There are no equations (diagrams), just the cone describing 1 as terminal and no cocones. This sketch, which we called **Nat**, has models that can be reasonably viewed as describing natural numbers. We described a more elaborate sketch in 8.1.4 which could also be used in what follows.

Let us consider the following diagram in the category of sketches, where $\mathscr{E}$ is the trivial sketch defined in Example 11.1.2 and the arrows F and G are defined by letting $Fe = n$ and $Ge = d$.

$$\begin{array}{ccc} \mathscr{E} & \xrightarrow{F} & \mathbf{Nat} \\ {\scriptstyle G}\downarrow & & \\ \mathbf{Stack} & & \end{array} \tag{11.1}$$

A pushout of this diagram is a sketch **Stack**(**Nat**) made by forming the union of **Nat** and **Stack** and then identifying the node n of **Nat** with the node d of **Stack**.

From the definition of pushout, it follows that a model of this sketch is a model of the sketch for natural numbers that is simultaneously a model of the sketch for stacks whose value at the node d of **Stack** is the same as its value at the node n of **Nat**. Models of this sketch can be identified as

stacks of natural numbers. The effect of this construction is to fill in the data parameter in the sketch for stacks with an actual data type. Of course, this data type of natural numbers could be replaced by any data type desired (including stacks of natural numbers!) by replacing $F : \mathscr{E} \longrightarrow \mathbf{Nat}$ by an appropriate sketch homomorphism.

Although we think of this sketch as being that of stacks of natural numbers, the pushout construction is completely symmetric and any asymmetry is imposed by our way of looking at it. It is caused by our (quite reasonable) perception that the node d of **Stack** is an input parameter and the node s represents the actual data type.

11.2.3 Binary trees, revisited again Here is a second example. In 8.3.11, we described a sketch called **BinTree** and then another sketch of the same name in 10.2.2. Here we describe yet another sketch we call **BinTree**. This third and final approach is fully parametrized and represents exactly what we mean by binary trees, no more and no less. We use nodes 1, t^+, t and d. Now the only operations we put in are empty, incl, val, left and right. We also need the cones and cocones necessary to say that $t^+ = 1 + t$ with inclusions empty and incl and that $t^+ = t \times d \times t$ with projections left, val and right.

Then if $F : \mathscr{E} \longrightarrow \mathbf{Nat}$ is as above and $H : \mathscr{E} \longrightarrow \mathbf{BinTree}$ is defined by $H(e) = d$, then the pushout of

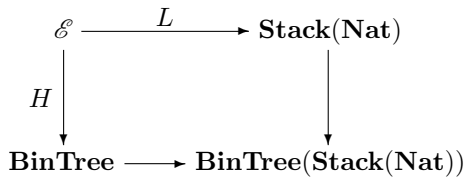
$$\begin{array}{ccc} \mathscr{E} & \xrightarrow{F} & \mathbf{Nat} \\ {\scriptstyle H}\downarrow & & \downarrow \\ \mathbf{BinTree} & \rightarrowtail & \mathbf{BinTree}(\mathbf{Nat}) \end{array}$$

is a sketch whose models can be interpreted as binary trees of natural numbers.

11.2.4 Further combinators This operation can be iterated. For example, we can form the pushout

$$\begin{array}{ccc} \mathscr{E} & \xrightarrow{K} & \mathbf{BinTree}(\mathbf{Nat}) \\ {\scriptstyle G}\downarrow & & \downarrow \\ \mathbf{Stack} & \longrightarrow & \mathbf{Stack}(\mathbf{BinTree}(\mathbf{Nat})) \end{array}$$

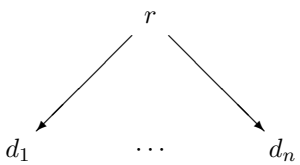
with $Ke = t$ or even



with $Le = s$. Models of these types will be interpreted as stacks of binary trees of natural numbers, respectively binary trees of stacks of natural numbers. Clearly what is at issue here is a notion of a type having an input node and an output node. However, things are not quite so simple, as the following example shows.

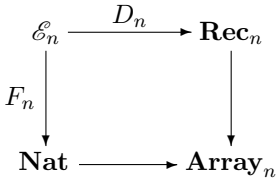
11.2.5 The basic idea of this type is that of n-place records (Pascal) or structures (C). We leave aside here the use of variant records (which can be adequately handled by judicious use of finite sums as C's union type shows). Another question we do not tackle is that of parametrizing n. At this point, we make no attempt to relate the sketches for two-place and three-place records, say, although it seems clear that in a mature theory this will be done. Parametrizing n is discussed in [Wagner, 1986b] and [Gray, 1989].

So we describe a sketch we will call $\mathbf{Rec}_n$. It has nodes r and d_1, d_2 ..., d_n and one cone

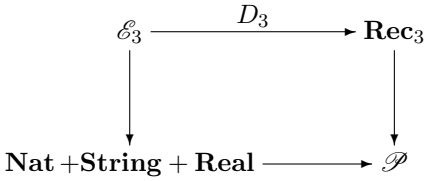


As it stands, the initial model of this sketch has all values empty. It is really a shell of a sketch. To give it content, we must fill in values for the data. To do this we must first recognize that the sketch has n input types and these must all be specified. This could be done in n steps, but it is more coherent to do it all at once.

Let $\mathscr{E}_n$ denote the discrete graph with nodes $e_1, \ldots, e_n$. There is an obvious arrow $D_n : \mathscr{E}_n \longrightarrow \mathbf{Rec}_n$ given by $De_i = d_i$. If, for example, we formed the pushout



where $F_n e_i = n,\ i = 1, \ldots, n$, the resultant sketch can be evidently interpreted as arrays of natural numbers. On the other hand, the pushout



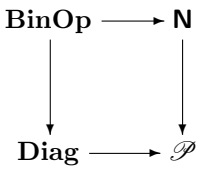
(where we suppose that sketches for **String** and **Real** have already been defined) is a sketch for three-place records, of which the first field is natural numbers, the second is strings and the third is floating point reals.

The ideas in this section are discussed without the explicit use of sketches in [Thatcher, Wagner and Wright, 1982], [Ehrig *et al.*, 1984] and [Wagner, 1986b].

11.2.6 Parametrizing operations The sketch $\mathscr{E}$ in the upper left corner of the pushout diagrams can contain operations. For example, let $\mathbf{Nat}^+$ denote the sketch for natural numbers with an added operation $+ : n \times n \longrightarrow n$ with diagrams forcing it to be addition in the initial model, and $\mathbf{Nat}^*$ a similar sketch with an added operation $* : n \times n \longrightarrow n$ forced to be multiplication there. Let **Diag** be the sketch whose graph contains operations

$$n \xrightarrow{\Delta} n \times n \xrightarrow{m} n$$

a cone with arrows $p_i : n \times n \longrightarrow n$, $i = 1, 2$, to force $n \times n$ to be the indicated product and a diagram to force Δ to be the diagonal map in a model. Let **BinOp** be the sketch containing $n \times n \longrightarrow n$ and similar data making $n \times n$ the indicated product. **BinOp** is included in both **Diag** and in each of $\mathbf{Nat}^+$ and $\mathbf{Nat}^*$. A pushout



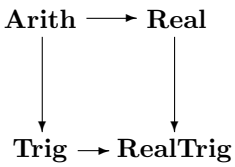
with $\mathbf{N} = \mathbf{Nat}^+$ would add a doubling operator to $\mathbf{Nat}^+$ and with $\mathbf{N} = \mathbf{Nat}^*$ would add a squaring operator to $\mathbf{Nat}^*$.

11.2.7 Arithmetic One last, more speculative example will show that we have barely begun to explore this idea. Let us suppose that we have defined two sketches called **Real** and **Complex** that implement the arithmetic operations (addition, subtraction, multiplication and division) of real and complex numbers, respectively, as well as absolute value. While it is true that

real numbers are a special case of complex numbers and therefore do not, in principle, have to be treated differently, it is also true that real arithmetic is much easier and faster and virtually every language that implements a complex number type treats them separately.

Suppose, further, there is a sketch **Trig** whose operations implement the trigonometric functions by means of power series or other approximations which have the same algorithm for real and for complex numbers. This sketch will use four formal arithmetic operations on an abstract data type d which admits the arithmetic operations and absolute value, but is not otherwise specified. These five operations are determined by a sketch **Arith** which has an arrow to all of **Real**, **Complex**, **Trig**.

If we now form the pushout



we get a sketch for the real type with trigonometric functions and if we replace **Real** by **Complex**, we get a sketch for complex trigonometry.

11.2.8 Exercise

1. If $\mathscr{S}$ is the pushout of Diagram (11.1), describe precisely the nodes in $\mathscr{S}$ which must become a singleton in a model in **Set**.

11.3 The model category functor

Sketch is a category whose objects are sketches. If $\mathscr{C}$ is a fixed category, then each sketch produces a category of models of the sketch in $\mathscr{C}$ (see 7.4.4). The category of models in $\mathscr{C}$ of a sketch $\mathscr{S}$ we will call $\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$. ($\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ might very well be the empty category.)

> Most of the examples in this book have had $\mathscr{C} = \mathbf{Set}$, which regrettably obscures one of the major advantages sketches have over standard logical theories: the fact that their models can be in any suitable category.

11.3.1 For each sketch homomorphism $F : \mathscr{S} \to \mathscr{T}$ we will now describe a functor $\mathbf{Mod}_{\mathscr{C}}(F) : \mathbf{Mod}_{\mathscr{C}}(\mathscr{T}) \to \mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ (note the reversal). $\mathbf{Mod}_{\mathscr{C}}(F)$ is also denoted F^*, and is called the **functor induced by** F. F^* is defined by MF–1 and MF–2 below. If M is a model of $\mathscr{T}$ then $F^*(M)$ is a model of $\mathscr{S}$.

MF–1 The object function of F^* is defined by $F^*(M) = M \circ F$. Thus if g is a node or arrow of the graph of $\mathscr{S}$ and M is a model of $\mathscr{T}$, then $F^*(M)(g) = M(F(g))$.

MF–2 If $\alpha : M \longrightarrow N$ is a homomorphism of models of $\mathscr{T}$, define $F^*(\alpha)$ to be the natural transformation αF as defined in Section 4.4. Thus at a node g of the graph of $\mathscr{S}$,

$$F^*(\alpha)(g) = (\alpha F)g = \alpha F(g) : M(F(g)) \longrightarrow N(F(g))$$

11.3.2 Proposition *For each model M of $\mathscr{T}$ and homomorphism $F : \mathscr{S} \longrightarrow \mathscr{T}$, $F^*(M)$ is a model of $\mathscr{S}$.*

Proof. Since M is among other things a graph homomorphism and so is F, and the composite of graph homomorphisms is a graph homomorphism, MF–1 makes $F^*(M)$, which is $M \circ F$, respect the source and target of arrows of the graph, so that it is a graph homomorphism to $\mathscr{C}$ as a model should be.

If D is a diagram of $\mathscr{S}$, then because F is a homomorphism of sketches, $F \circ D$ is a diagram of $\mathscr{T}$. Since M is a model of $\mathscr{T}$, $F^*(M) \circ D = M \circ F \circ D$ commutes.

Suppose $v \longrightarrow D$ is a cone of $\mathscr{S}$. Then $F(v) \longrightarrow F \circ D$ is a cone of $\mathscr{T}$ which must be taken to a limit cone by M. Since $F^*(M)(v \longrightarrow D) = M(F(v)) \longrightarrow M \circ F \circ D$, $F^*(M)$ takes $v \longrightarrow D$ to a limit cone. A similar argument deals with cocones. □

11.3.3 Proposition *For each homomorphism $\alpha : M \longrightarrow N$ of models of $\mathscr{T}$ and homomorphism $F : \mathscr{S} \longrightarrow \mathscr{T}$ of sketches, $\mathbf{Mod}_{\mathscr{C}}(F)(\alpha)$ as defined by MF–2 is a homomorphism of models of $\mathscr{S}$.*

Proof. By MF–2, $F^*(\alpha)$ is the natural transformation αF defined in 4.4.1. Since its domain and codomain are models, it is a homomorphism of models by definition. □

For any sketch $\mathscr{S}$, let $\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ be the category of models of $\mathscr{S}$ in $\mathscr{C}$ (we called this $\mathbf{Mod}(\mathscr{S}, \mathscr{C})$ in Section 10.1. If $F : \mathscr{S} \longrightarrow \mathscr{T}$ is a homomorphism of sketches, we have defined a functor $\mathbf{Mod}_{\mathscr{C}}(F) : \mathbf{Mod}_{\mathscr{C}}(\mathscr{T}) \longrightarrow \mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$.

11.3.4 Proposition $\mathbf{Mod}_{\mathscr{C}} : \mathbf{Sketch}^{\mathrm{op}} \longrightarrow \mathbf{Cat}$ *is a functor.*

The proof involves some simple checking and is left as an exercise.

11.3.5 Example In Section 3.1, we gave several examples of underlying set functors $U : \mathscr{C} \longrightarrow \mathbf{Set}$. In general, such functors are induced by a homomorphism of sketches from the trivial sketch $\mathscr{E}$ defined in 11.1.2 to a particular node of the sketch $\mathscr{C}$. For example, the underlying set functor $U : \mathbf{Sem} \longrightarrow \mathbf{Set}$ (see 3.1.8) is induced by the unique sketch homomorphism from $\mathscr{E}$ to the sketch for semigroups that takes the only node of $\mathscr{E}$ to s. This follows directly from MF–1 and MF–2 and the fact that a model of $\mathscr{E}$ in **Set** is essentially a set (Exercise 1 of Section 11.2). Similarly the underlying arrow and node functors $A : \mathbf{Grf} \longrightarrow \mathbf{Set}$ and $N : \mathbf{Grf} \longrightarrow \mathbf{Set}$ (see 3.1.9) are induced by the sketch homomorphisms from $\mathscr{E}$ to the sketch for graphs that take e to a and to n respectively.

11.3.6 Example The sketch homomorphism of Example 11.1.3 induces the functor from the category of graphs to the arrow category of **Set** that takes a graph to its source function. The homomorphism of Example 11.1.4 that includes the sketch for graphs into the sketch for simple graphs induces the underlying functor that forgets that a graph is simple. Similarly the functor of 11.1.5 that includes the sketch for semigroups into the sketch for fields induces the underlying functor from fields to semigroups that takes a field to its additive semigroup.

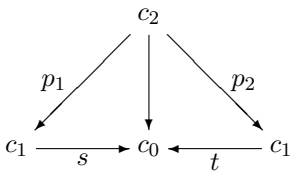
11.3.7 Example Sketches usually include only the minimal information that is needed to describe a theory. There is a cost to this in that they may omit crucial information needed to define morphisms. Here is an uncontrived example that illustrates the point. We begin by observing that in any category, if 1 is a terminal object (vertex of a cone with empty base) and $S \times S$ is a product of two copies of S, then the square

$$\begin{array}{ccc} S \times S & \longrightarrow & S \\ \downarrow & & \downarrow \\ S & \longrightarrow & 1 \end{array}$$

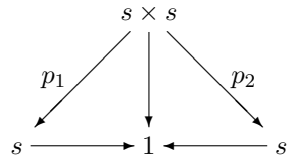
is a pullback. Now consider the functor from the category **Mon** of monoids to the category **Cat** of small categories that takes a monoid to the corresponding category with one object (see 2.3.12). One expects that this functor would be induced by a sketch homomorphism from the sketch for categories given by 10.1.5 to the sketch for monoids given by 7.2.1 as augmented by 7.3.2. Since the underlying set of the monoid is the set of arrows of the corresponding category, the sketch homomorphism should take the node c_1 of 10.1.5 to the node s of 7.2.1, and the arrow c of 10.1.5 (which we will call comp here to avoid confusion) should go to the multiplication

c of 7.2.1. Since comp has domain c_2 and c has domain $s \times s$, c_2 should go to $s \times s$. Unfortunately, $s \times s$ is the vertex of a discrete cone and c_2 is the vertex of a pullback cone. And besides, where should c_0 go?

The key is to map c_0 to the object 1 of the sketch for monoids, so that s and t are forced to go to the unique map from s to 1. This would force the pullback cone



to go to the cone



which unfortunately is not a cone in the sketch for monoids. However, it is a cone in the FL theory for the sketch for monoids, so it would appear that, although we cannot realize the functor in question as induced by a sketch homomorphism from the sketch for categories to the sketch for monoids, it is nevertheless induced by a homomorphism to a sketch – the underlying sketch of the theory of monoids.

It is clear that, as the preceding example illustrates, one would normally want to consider homomorphisms of theories (functors that preserve finite limits in this case) rather than homomorphisms of sketches. In practice, one would construct a homomorphism from a sketch $\mathscr{S}$ to the theory of a sketch $\mathscr{T}$ since such a homomorphism would extend essentially uniquely to a mapping between theories. An alternative would be to adjoin just what you need to $\mathscr{T}$ to make the homomorphism work (the cone of Diagram (11.3.7) – and others – in this case), but that seems excessively clumsy when the whole theory exists in any case.

11.3.8 It is almost immediate that Propositions 11.3.2, 11.3.3 and 11.3.4 remain true if the category **Sketch** is replaced by the category of sketches with diagrams based on a specific class of shape graphs, cones to specific class of shape graphs not necessarily the same as that of the diagrams, and cocones from a specific class of shape graphs not necessarily the same as either of the other two. For example, taking any shapes for the diagrams, finite discrete shapes for the cones, and no shapes for the cocones gives the

category of FP sketches and homomorphisms between them, so all these propositions are true of FP sketches.

11.3.9 Sentences For this subsection only, we will say a **sentence** in a sketch $\mathscr{S}$ is a diagram, cone or cocone in the graph of $\mathscr{S}$, *not* necessarily one in the specified set of diagrams, cones or cocones. The sentence is **satisfied** in a model M of $\mathscr{S}$ if it commutes when M is applied (if it is a diagram) or if it is a (co)limit (co)cone when M is applied (if it is a (co)cone).

If $F : \mathscr{S} \to \mathscr{T}$ is a sketch homomorphism, and σ is a sentence of $\mathscr{S}$, then $F(\sigma)$ is a sentence of $\mathscr{T}$ where $F(\sigma)$ is defined by composition: this follows from the definition of sketch homomorphism.

11.3.10 Proposition *If $F : \mathscr{S} \to \mathscr{T}$ is a sketch homomorphism and σ is a sentence of $\mathscr{S}$, then $F(\sigma)$ is satisfied in a model M of $\mathscr{T}$ in a category $\mathscr{C}$ if and only if σ is satisfied in* $\mathbf{Mod}_{\mathscr{C}}(F)(M)$.

This follows directly from the definitions and is left as an exercise.

Propositions 11.3.4 and 11.3.10 imply that the category of sketches with designated shape graphs for each of the diagrams, cones and cocones, together with sentences, form a 'simple institution' in the sense of [Goguen and Burstall, 1986].

11.3.11 The results of this section show the way in which sketches are a very different method for describing mathematical structures from the theories of traditional mathematical logic. Of course, sketches use graphs, diagrams, cones and cocones instead of variables, symbols, expressions and formulas, but that difference, although significant, is not the biggest difference. When you use sketches, you factor the entire descriptive process differently. We have already discussed the differences to some extent for FP sketches in 7.2.8.

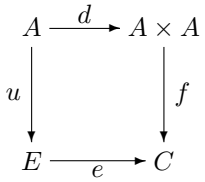
An example of the subtlety of the difference is exemplified by our word 'sentence' above. Diagrams correspond to universally quantified sentences, and the correspondence, which is not entirely trivial (see Section 7.7), is nevertheless exact. But to use a cone as a sentence corresponds to a statement *about a type* in multisorted logic, rather than statements about terms. (McLarty [1986] makes this explicit.) Such statements have not played a big role in classical logic. In classical logic, you do not say (with an axiom) that a type is (for example) a product type; the product type is implicit in the existence of some given term which has a sequence of variables of specific types.

On the other hand, our sentences do not provide a way of giving universal Horn clauses in (for example) an FL sketch. When using sketches, universal Horn clauses are usually implicit. Such a clause which must be satisfied in

all models is given by an arrow built into the graph of the sketch which causes an operation to factor through some limit type. Thus if you wanted to require

$$a = b \Rightarrow g(f(a,b)) = h(f(a,b))$$

where a and b are of type A, f is an operation of type C and g and h are operations of type D, you build the graph of your sketch with arrows $f : A \times A \to C$, $g, h : C \to D$, $e : E \to C$, $u : A \to E$ and $d : A \to A \times A$, with cones requiring $A \times A$ to be the required product, e to be the equalizer of g and h, d to be the diagonal map, and this diagram



which requires $f \circ d$ to factor through the equalizer of g and h. It is built into the graph in a way analogous to the way in classical logic you build the product types implicitly by incorporating specific terms.

There is presumably a theory of sentences built in this way which makes FL sketches an institution, but it will require work to produce it: it does not fit well with the ingredients of a sketch. In the same way, building an institution out of classical logic using a definition of sentence which allows you to state that a type is a certain equalizer or pullback requires work because it does not fit the traditional way of doing things. (It is *not* impossible, it merely does not fit well.) *The classical approach and the sketch approach make different things easy.*

11.3.12 Exercises

1. In Example 4.3.6 we defined a functor $U \times U : \mathbf{Mon} \to \mathbf{Set}$. Show that it is induced by a homomorphism of sketches.

2. Prove that MF–2 makes F^* a natural transformation.

3. Prove Proposition 11.3.4.

4. Prove Proposition 11.3.10.

12
Fibrations

A category is both a generalized poset and a generalized monoid. Many constructions in category theory can be understood in terms of the constructions in posets that they generalize, so that it is generally good advice when learning about a new categorical idea to see what it says about posets. Seeing what a construction says about monoids has not usually been so instructive.

However, certain concepts used to study the algebraic structure of monoids generalize to categories in a natural way, and often the theorems about them remain true. In addition, applications of monoids to the theory of automata have natural generalizations to categories, and some work has been done on these generalized ideas.

In this chapter we describe some aspects of categories as generalized monoids. We begin in Section 12.1 with the concept of fibration, which has been used in recent research on polymorphism. One way of constructing fibrations is by the Grothendieck construction, described in Section 12.2, which is a generalization of the semidirect product construction for monoids. Section 12.3 gives an equivalence between certain types of fibrations and category-valued functors. Section 12.4 describes the wreath product of categories, a generalization of the concept of the same name for monoids; some applications of the construction are mentioned.

The Grothendieck construction is used in Section 15.7. The rest of the material is not used elsewhere in the book.

12.1 Fibrations

In this section, we describe fibrations, which are special types of functors important in category theory and which have been proposed as useful in certain aspects of computer science.

The next section gives a way of constructing fibrations from set or category-valued functors.

12.1.1 Fibrations and opfibrations Let $P : \mathscr{E} \to \mathscr{C}$ be a functor between small categories, let $f : C \to D$ be an arrow of $\mathscr{C}$, and let $P(Y) = D$. An arrow $u : X \to Y$ of $\mathscr{E}$ is **cartesian** for f and Y if

CA–1 $P(u) = f$.

CA–2 For any arrow $v : Z \longrightarrow Y$ of $\mathscr{E}$ and any arrow $h : P(Z) \longrightarrow C$ of $\mathscr{C}$ for which $f \circ h = P(v)$, there is a unique $w : Z \longrightarrow X$ in $\mathscr{E}$ such that $u \circ w = v$ and $P(w) = h$.

Similarly, if $f : C \longrightarrow D$ and $P(X) = C$, then an arrow $u : X \longrightarrow Y$ is **opcartesian** for f and X if

OA–1 $P(u) = f$.

OA–2 For any arrow $v : X \longrightarrow Z$ of $\mathscr{E}$ and any arrow $k : D \longrightarrow P(Z)$ for which $k \circ f = P(v)$, there is a unique $w : Y \longrightarrow Z$ in $\mathscr{E}$ for which $w \circ u = v$ and $P(w) = k$.

If $P : \mathscr{E} \longrightarrow \mathscr{C}$ is a functor, categorists often think of $\mathscr{E}$ as being above $\mathscr{C}$. (This is also common for functions between spaces, which is what originally suggested the ideas in this section.) For example, if $P(Y) = D$, one says that Y **lies over** D. Similar terminology is used for arrows. Thus a cartesian arrow for f must lie over f (CA–1) and one refers to CA–2 as a 'unique lifting' property.

12.1.2 Definition A functor $P : \mathscr{E} \longrightarrow \mathscr{C}$ is a **fibration** if there is a cartesian arrow for every $f : C \longrightarrow D$ in $\mathscr{C}$ and every object Y of $\mathscr{E}$ for which $P(Y) = D$. P is an **opfibration** if there is an opcartesian arrow for every $f : C \longrightarrow D$ in $\mathscr{C}$ and every object X of $\mathscr{E}$ for which $P(X) = C$. It follows that $P : \mathscr{E} \longrightarrow \mathscr{C}$ is a fibration if and only if $P^{\mathrm{op}} : \mathscr{E}^{\mathrm{op}} \longrightarrow \mathscr{C}^{\mathrm{op}}$ is an opfibration.

If $P : \mathscr{E} \longrightarrow \mathscr{C}$ is a fibration, one also says that $\mathscr{E}$ is **fibered over** $\mathscr{C}$. In that case, $\mathscr{C}$ is the **base category** and $\mathscr{E}$ is the **total category** of the fibration.

12.1.3 Definition A **cleavage** for a fibration $P : \mathscr{E} \longrightarrow \mathscr{C}$ is a function γ that takes an arrow $f : C \longrightarrow D$ and object Y such that $P(Y) = D$ to an arrow $\gamma(f, Y)$ of $\mathscr{E}$ that is cartesian for f and Y. Similarly an opcleavage κ takes $f : C \longrightarrow D$ and X such that $P(X) = C$ to an arrow $\kappa(f, X)$ that is opcartesian for f and X.

The cleavage γ is a **splitting** of the fibration if it satisfies the following two requirements.

SC–1 Let D be an object of $\mathscr{C}$ and let Y be an object of $\mathscr{E}$ for which $P(Y) = D$. Then $\gamma(\mathrm{id}_D, Y) = \mathrm{id}_Y$.

SC–2 Suppose $f : C \longrightarrow D$ and $g : D \longrightarrow E$ in $\mathscr{C}$ and suppose Y and Z are objects of $\mathscr{E}$ for which $P(Z) = E$ and Y is the domain of $\gamma(g, Z)$. Then

$$\gamma(g, Z) \circ \gamma(f, Y) = \gamma(g \circ f, Z)$$

Note that under the assumptions in SC–2, $P(Y) = D$, so that $\gamma(f, Y)$ and $\gamma(g, Z) \circ \gamma(f, Y)$ are defined.

A fibration is **split** if it has a splitting.

Similarly, an opcleavage κ is a splitting of an opfibration $P : \mathscr{E} \longrightarrow \mathscr{C}$ if $\kappa(\mathrm{id}_C, X) = \mathrm{id}_X$ whenever $P(X) = C$ and

$$\kappa(g, Y) \circ \kappa(f, X) = \kappa(g \circ f, X)$$

whenever $f : C \longrightarrow D$ and $g : D \longrightarrow E$ in $\mathscr{C}$, $P(X) = C$ and Y is the codomain of $\kappa(f, X)$. A split opfibration is again one which has a splitting.

12.1.4 Example Let $\mathscr{A}$ and $\mathscr{C}$ be any categories. Then the second projection $p_2 : \mathscr{A} \times \mathscr{C} \longrightarrow \mathscr{C}$ is both a split fibration and a split opfibration. To see that it is a fibration, suppose that $f : C \longrightarrow C'$ in $\mathscr{C}$ and let $Y = (A, C')$ be an object of $\mathscr{A} \times \mathscr{C}$. Then we can take $\gamma(f, Y)$ to be the arrow $(\mathrm{id}_A, f) : (A, C) \longrightarrow (A, C')$. If $(g, h) : (A', C'') \longrightarrow (A, C')$ and $u : C'' \longrightarrow C$ satisfy $f \circ u = h$ (note that $p_2(g, h) = h$), then the unique arrow from (A', C'') to (A, C') required by CA–2 is (g, u).

12.1.5 Example If $\mathscr{C}$ is a category, the **arrow category** of $\mathscr{C}$ (which we have already mentioned in 4.2.17) has as objects the *arrows* of $\mathscr{C}$. An arrow from $f : A \longrightarrow B$ to $g : C \longrightarrow D$ is a pair (h, k) of arrows with $h : A \longrightarrow C$, $k : B \longrightarrow D$ for which

$$\begin{array}{ccc} A & \xrightarrow{h} & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & D \end{array} \tag{12.1}$$

commutes.

If $\mathscr{A}$ is the arrow category of $\mathscr{C}$, there is a functor $P : \mathscr{A} \longrightarrow \mathscr{C}$ which takes $f : A \longrightarrow B$ to B and $(h, k) : f \longrightarrow g$ to k. If $\mathscr{C}$ has pullbacks, this functor is a fibration. For a given $f : C \longrightarrow D$ in $\mathscr{C}$ and object $k : B \longrightarrow D$ of $\mathscr{A}$, a cartesian arrow for f and k is any (u, f) given by a pullback

$$\begin{array}{ccc} P & \xrightarrow{u} & B \\ {\scriptstyle u'}\downarrow & & \downarrow{\scriptstyle k} \\ C & \xrightarrow[f]{} & D \end{array} \tag{12.2}$$

The verification is left as an exercise (Exercise 4).

12.1.6 Fibers For any functor $P : \mathscr{E} \to \mathscr{C}$, the **fiber over** an object C of $\mathscr{C}$ is the set of objects X for which $P(X) = C$ and arrows f for which $P(f) = \mathrm{id}_C$. It is easy to verify that this fiber is a subcategory of $\mathscr{E}$ (Exercise 1).

In the case of Example 12.1.4, the fibers are all the same: each one is isomorphic to the category $\mathscr{A}$. This suggests thinking of an arbitrary fibration as a type of generalized product, in which the first coordinates come in general from varying sets depending on the second coordinate. This observation can also be made concerning the relationship between a set product $S \times T$ and a general T-indexed set.

On the other hand, the fiber of the fibration in Example 12.1.5 over an object A of $\mathscr{C}$ is the slice category $\mathscr{C}/A$. Since an object of $\mathscr{C}/A$ can be thought of as an indexed family of objects of $\mathscr{C}$, indexed by A, this example has been referred to as $\mathscr{C}$ 'fibered over itself'.

12.1.7 Cleavages induce functors If $\mathscr{E}$ is fibered over $\mathscr{C}$, then the fibers form an indexed set of categories (indexed by the objects). Given a cleavage, the arrows of $\mathscr{C}$ induce functors between the fibers. In this way fibrations or opfibrations give a concept like that of indexed sets, in which the indexing takes into account the arrows of the underlying categories as well as the objects. Propositions 12.1.8 and 12.1.9 below spell this out.

An alternative approach to these ideas which follows the indexed set analogy more explicitly is the concept of **indexed category** (see [Johnstone and Paré, 1978], [Tarlecki, Burstall and Goguen, 1991]). Rosebrugh and Wood [1992] apply indexed categories to relational databases and Cockett and Spencer [1992] use them in studying datatypes.

In the rest of this chapter, when $F : \mathscr{C} \to \mathbf{Cat}$ or $F : \mathscr{C} \to \mathbf{Set}$ is a functor, we will normally write Ff for $F(f)$.

Let $P : \mathscr{E} \to \mathscr{C}$ be an opfibration with opcleavage κ. Define $F : \mathscr{C} \to \mathbf{Cat}$ by

FF–1 $F(C)$ is the fiber over C for each object C of $\mathscr{C}$.

FF–2 For $f : C \to D$ in $\mathscr{C}$ and X an object of $F(C)$, $Ff(X)$ is defined to be the codomain of the arrow $\kappa(f, X)$.

FF–3 For $f : C \to D$ in $\mathscr{C}$ and $u : X \to X'$ in $F(C)$, $Ff(u)$ is the unique arrow from $Ff(X)$ to $Ff(X')$ given by OA–2 for which

$$Ff(u) \circ \kappa(f, X) = \kappa(f, X') \circ u$$

12.1.8 Proposition *Let $P : \mathscr{E} \to \mathscr{C}$ be an opfibration with cleavage κ. For any arrow $f : C \to D$ in $\mathscr{C}$, $Ff : F(C) \to F(D)$ as defined by FF–1 through FF–3 is a functor. Moreover, if κ is a splitting, then F is a functor from $\mathscr{C}$ to* **Cat**.

Proof. Let $u : X \to X'$ and $v : X' \to X''$ in $F(C)$. Then

$$\begin{aligned} Ff(v) \circ Ff(u) \circ \kappa(f, X) &= Ff(v) \circ \kappa(f, X') \circ u \\ &= \kappa(f, X'') \circ v \circ u \end{aligned} \tag{12.3}$$

by two applications of FF–3. But then by the uniqueness part of FF–3, $Ff(v) \circ Ff(u)$ must be $Ff(v \circ u)$. This proves Ff preserves composition. We leave the preservation of identities to you.

Now suppose κ is a splitting. Let $f : C \to D$ and $g : D \to E$ in $\mathscr{C}$, and let $u : X \to X'$ in $F(C)$. Then $F(g \circ f)(u)$ is the unique arrow from $F(g \circ f)(X)$ (the codomain of $\kappa(g \circ f, X)$) to $F(g \circ f)(X')$ (the codomain of $\kappa(g \circ f, X')$) for which

$$F(g \circ f)(u) \circ \kappa(g \circ f, X) = \kappa(g \circ f, X') \circ u$$

Since κ is a splitting, this says

$$F(g \circ f)(u) \circ \kappa(g, Ff(X)) \circ \kappa(f, X) = \kappa(g, Ff(X')) \circ \kappa(f, X') \circ u$$

By FF–3, the right side is

$$\kappa(g, Ff(X')) \circ Ff(u) \circ \kappa(f, X)$$

Applying FF–3 with g and $Ff(u)$ instead of f and u, this is the same as

$$Fg[Ff(u)] \circ \kappa(g, Ff(X)) \circ \kappa(f, X)$$

which is

$$Fg[Ff(u)] \circ \kappa(g \circ f, X)$$

because κ is a splitting. Using the uniqueness requirement in FF–3, this means $F(g \circ f)(u) = Fg[Ff(u)]$, so that F preserves composition. Again, we leave preservation of the identity to you. □

In a similar way, split fibrations give functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Cat}$. Let $P : \mathscr{E} \to \mathscr{C}$ be a fibration with cleavage γ. Define $F : \mathscr{C}^{\mathrm{op}} \to \mathbf{Cat}$ by

FF′–1 $F(C)$ is the fiber over C for each object C of $\mathscr{C}$.

FF′–2 For $f : C \to D$ in $\mathscr{C}$ and Y an object of $F(D)$, $Ff(Y)$ is defined to be the domain of the arrow $\gamma(f, Y)$.

FF′–3 For $f : C \to D$ in $\mathscr{C}$ and $u : Y \to Y'$ in $F(D)$, $Ff(u)$ is the unique arrow from $Ff(Y)$ to $Ff(Y')$ given by CA–2 for which

$$\gamma(f, Y') \circ Ff(u) = u \circ \gamma(f, Y)$$

12.1.9 Proposition *Let $P : \mathscr{E} \to \mathscr{C}$ be a fibration with cleavage κ. For any arrow $f : C \to D$ in $\mathscr{C}$, $Ff : F(C) \to F(D)$ as defined by FF–1 through FF–3 is a functor. Moreover, if γ is a splitting, then F is a functor from $\mathscr{C}^{\mathrm{op}}$ to* **Cat**.

The proof is similar to those of Proposition 12.1.8 and is left as an exercise.

12.1.10 Exercises

1. Verify that for any functor $P : \mathscr{E} \to \mathscr{C}$ and object C of $\mathscr{C}$, the fiber over an object C is a subcategory of $\mathscr{E}$.

2. Prove Proposition 12.1.9.

3. Let $\phi : \mathbf{Z}_4 \to \mathbf{Z}_2$ be the homomorphism defined in Exercise 2 of Section 2.9.

a. Show that the functor from $C(\mathbf{Z}_4)$ to $C(\mathbf{Z}_2)$ induced by ϕ is a fibration and an opfibration. (If you know about groups, this is an instance of the fact that every surjective group homomorphism is a fibration and an opfibration.)

b. Show that ϕ is not a split fibration or opfibration.

4. Let $\mathscr{C}$ be a category with pullbacks and $\mathscr{A}$ its arrow category. For an arrow $f : A \to B$ (object of $\mathscr{A}$) let $P(f) = B$. For an arrow $(h, k) : f \to g$ (where $g : C \to D$ in $\mathscr{C}$) in $\mathscr{A}$, let $P(h, k) = k$.

a. Show that $P : \mathscr{A} \to \mathscr{C}$ is a functor.

b. Show that P is a fibration.

12.2 The Grothendieck construction

The Grothendieck construction is a way of producing fibrations. It generalizes the semidirect product construction for monoids, which is defined here. Hyland and Pitts [1989] use the Grothendieck construction to construct categories that are models of the calculus of constructions, a system due to Coquand and Huet [1988] that provides a way of handling polymorphism essentially by quantifying over types. (See also [Coquand, 1988], [Ehrhard, 1988] and [Asperti and Martini, 1992].)

The construction can be applied to either set-valued functors or category-valued functors. Given such a functor $F : \mathscr{C} \to \mathbf{Set}$ or $F : \mathscr{C} \to \mathbf{Cat}$, it constructs a category $\mathbf{G}(\mathscr{C}, F)$ and a functor from $\mathbf{G}(\mathscr{C}, F)$ to $\mathscr{C}$. When F is set-valued we will write $\mathbf{G}_0$ instead of $\mathbf{G}$. We will look at the set-valued case first, since it is simpler.

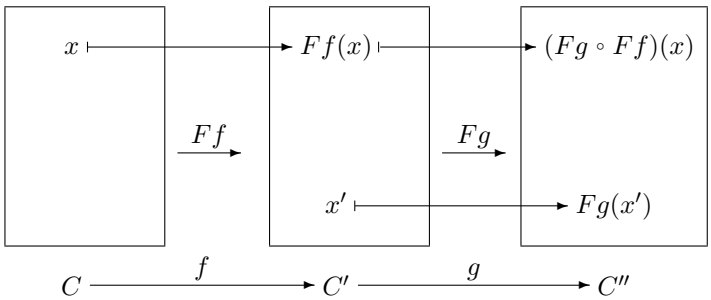
12.2.1 Let $\mathscr{C}$ be a small category and let $F : \mathscr{C} \to \mathbf{Set}$ be a functor. For each object C of $\mathscr{C}$, $F(C)$ is a set, and for each arrow $f : C \to D$, $F(f) : F(C) \to f(D)$ is a set function. There is no set of all sets or set of all set functions, but, since $\mathscr{C}$ is small, there certainly is a set consisting of all the elements of all the sets $F(C)$, and similarly there is a set consisting of all the functions $F(f)$. In other words, although **Set** is large, the description of $F : \mathscr{C} \to \mathbf{Set}$ requires only a small amount of data. ('Small' and 'large' are used here in the technical sense, referring to whether or not a set of data is involved. See 1.3.4.)

By contrast, to describe a functor $G : \mathbf{Set} \to \mathscr{C}$ would require a large amount of data – an object of $\mathscr{C}$ for each set, and so on.

We will formalize these observations about $F : \mathscr{C} \to \mathbf{Set}$ by taking the disjoint union of all the sets of the form $F(C)$ for all objects C of $\mathscr{C}$. The elements of this disjoint union can be represented as pairs (x, C) for all objects C of $\mathscr{C}$ and elements $x \in F(C)$. (Thus we construct the disjoint union of sets by labeling the elements. The disjoint union is the construction in **Set** corresponding to the categorical concept of 'sum', discussed in Section 5.4.)

We must do more than this to capture the *functorial nature* of F – what it does to arrows of $\mathscr{C}$. The category $\mathbf{G}_0(\mathscr{C}, F)$ constructed by the Grothendieck construction does capture this structure, and its set of objects is the disjoint union just mentioned.

12.2.2 If we were to draw a picture to explain what F does, the result might be Diagram (12.4), in which $f : C \to C'$ and $g : C' \to C''$ are arrows of $\mathscr{C}$ and x and x' are elements of $F(C)$ and $F(C')$ respectively. The box



(12.4)

over each object C of $\mathscr{C}$ represents the elements of $F(C)$. The arrows from x to $Ff(x)$ and from x' to $Fg(x')$ are there informally to illustrate what the set functions Ff and Fg do. *These informal arrows become actual arrows in* $\mathbf{G}_0(\mathscr{C}, F)$.

12.2.3 Definition $\mathbf{G}_0(\mathscr{C}, F)$ is the category defined as follows.

GS–1 An object of $\mathbf{G}_0(\mathscr{C}, F)$ is a pair (x, C) where C is an object of $\mathscr{C}$ and x is an element of $F(C)$ (as observed, the C occurs in the pair (x, C) because we want the disjoint union of the values of F).

GS–2 An arrow is a pair of the form $(x, f) : (x, C) \to (x', C')$ where f is an arrow $f : C \to C'$ of $\mathscr{C}$ for which $Ff(x) = x'$.

GS–3 If $(x, f) : (x, C) \longrightarrow (x', C')$ and $(x', g) : (x', C') \longrightarrow (x'', C'')$, then $(x', g) \circ (x, f) : (x, C) \longrightarrow (x'', C'')$ is defined by

$$(x', g) \circ (x, f) = (x, g \circ f)$$

Note in GS–3 that indeed $F(g \circ f)(x) = x''$ as required by GS–2.

The reason that we use the notation (x, f) is the requirement that an arrow must determine its source and target. The source of (x, f) is (x, C), where C is the source of f and x is explicit, while its target is (x', C'), where C' is the target of f and $x' = Ff(x)$. In the literature, (x, f) is often denoted simply f, so that the same name f may refer to many different arrows – one for each element of $F(C)$. We used lacunary notation of this sort in defining slice categories in 2.6.10.

Projection on the second coordinate defines a functor

$$\mathbf{G}_0(F) : \mathbf{G}_0(\mathscr{C}, F) \longrightarrow \mathscr{C}$$

$\mathbf{G}_0(\mathscr{C}, F)$ together with $\mathbf{G}_0(F)$ is called the **split discrete opfibration** induced by F, and $\mathscr{C}$ is the **base category** of the opfibration.

If C is an object of $\mathscr{C}$, the inverse image under $\mathbf{G}_0(F)$ of C is simply the set $F(C)$, although its elements are written as pairs so as to form a disjoint union.

This discrete opfibration is indeed an opfibration, in fact a split opfibration. If $f : C \longrightarrow C'$ in $\mathscr{C}$ and (x, C) is an object of $\mathbf{G}_0(\mathscr{C}, F)$, then an opcartesian arrow is $(x, f) : (x, C) \longrightarrow (Ff(x), C')$ (Exercise 2). The word 'discrete' refers to the fact that the fibers are categories in which the only arrows are identity arrows; such categories are essentially the same as sets.

12.2.4 Semidirect products We now describe a more general version of the Grothendieck construction that has the semidirect product of monoids as a special case. We first define the semidirect product of monoids: it is constructed from two monoids, one of which acts on the other.

12.2.5 Definition If M and T are monoids, an **action** of M on T is a function $\alpha : M \times T \longrightarrow T$ for which

MA–1 $\alpha(m, 1_T) = 1_T$ for all $m \in M$.

MA–2 $\alpha(m, tu) = \alpha(m, t)\alpha(m, u)$ for all $m \in M$ and $t, u \in T$.

MA–3 $\alpha(1_M, t) = t$ for all $t \in T$.

MA–4 $\alpha((mn), t) = \alpha(m, \alpha(n, t))$ for all $m, n \in M$ and $t \in T$.

If we curry α as in 6.1.2, we get a family of functions $\phi(m) : T \longrightarrow T$ with the properties listed in MA′–1 through MA′–4 below.

MA′–1 $\phi(m)(1_T) = 1_T$ for all $m \in M$.

MA′–2 $\phi(m)(tu) = \phi(m)(t)\phi(m)(u)$ for all $m \in M$ and $t, u \in T$.

MA′–3 $\phi(1_M)(t) = t$ for all $t \in T$.

MA′–4 $\phi(mn)(t) = \phi(m)[\phi(n)(t)]$ for all $m, n \in M$ and $t \in T$.

Thus we see that an alternative formulation of monoid action is that it is a monoid homomorphism $\phi : M \longrightarrow \mathbf{End}(T)$ ($\mathbf{End}(T)$ being the monoid of endomorphisms of T). MA′–1 and MA′–2 say that each function $\phi(m)$ is an endomorphism of T, and MA′–3 and MA′–4 say that ϕ is a monoid homomorphism.

12.2.6 Definition The **semidirect product** of M and T with the given action α as just defined is the monoid with underlying set $T \times M$ and multiplication defined by

$$(t, m)(t', m') = (t\alpha(m, t'), mm')$$

To see the connection with the categorical version below you may wish to write this definition using the curried version of α.

12.2.7 The categorical construction corresponding to a monoid acting on a monoid is a functor which takes values in **Cat** rather than in **Set**. A functor $F : \mathscr{C} \longrightarrow \mathbf{Cat}$ can be regarded as an action of $\mathscr{C}$ on a *variable category* which plays the role of T in the definition just given.

In the case of a monoid action defined by MA–1 through MA–4, the variable category is actually not varying: it is the category $C(T)$ determined by the monoid T. The functor F in that case takes the single object of M to the single object of T, and, given an element $m \in M$, $F(m)$ is the endomorphism of T which takes $t \in T$ to mt: in other words, $F(m)(t) = mt$. Thus F on the arrows is the curried form of the action α.

A set-valued functor is a special case of a category-valued functor, since a set can be regarded as a category with only identity arrows. Note that this is different from the monoid case: an action by a monoid on a set is *not* in general a special case of an action by the monoid on a monoid. It is, however, a special case of the action of a monoid on a category – a discrete category.

12.2.8 Given a functor $F : \mathscr{C} \longrightarrow \mathbf{Cat}$, the Grothendieck construction in this more general setting constructs the opfibration induced by F, a category $\mathbf{G}(\mathscr{C}, F)$ defined as follows:

GC–1 An object of $\mathbf{G}(\mathscr{C}, F)$ is a pair (x, C) where C is an object of $\mathscr{C}$ and x is an object of $F(C)$.

GC–2 An arrow $(u, f) : (x, C) \longrightarrow (x', C')$ has $f : C \longrightarrow C'$ an arrow of $\mathscr{C}$ and $u : Ff(x) \longrightarrow x'$ an arrow of $F(C')$ (note that by definition $Ff(x)$ is an object of $F(C')$).

GC–3 If $(u, f) : (x, C) \longrightarrow (x', C')$ and $(v, g) : (x', C') \longrightarrow (x'', C'')$, then $(v, g) \circ (u, f) : (x, C) \longrightarrow (x'', C'')$ is defined by

$$(v, g) \circ (u, f) = (v \circ Fg(u), g \circ f)$$

12.2.9 Theorem *Given a functor $F : \mathscr{C} \longrightarrow \mathbf{Cat}$, $\mathbf{G}(\mathscr{C}, F)$ is a category and the second projection is a functor $P : \mathbf{G}(\mathscr{C}, F) \longrightarrow \mathscr{C}$ which is a split opfibration with splitting*

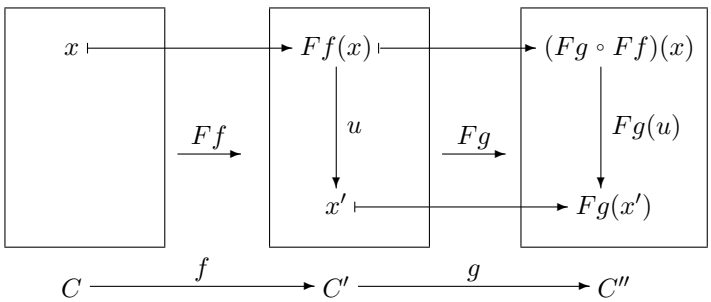
$$\kappa(f, X) = (\mathrm{id}_{Ffx}, f) : (x, C) \longrightarrow (Ffx, C')$$

for any arrow $f : C \longrightarrow C'$ of $\mathscr{C}$ and object (x, C) of $\mathbf{G}(\mathscr{C}, F)$.

We omit the proof of this theorem. $\mathbf{G}(\mathscr{C}, F)$ is called the **crossed product** $\mathscr{C} \times F$ by some authors.

It is instructive to compare this definition with the discrete opfibration constructed from a set-valued functor. In the case that F is set-valued, the first component u of an arrow $(u, f) : (x, C) \longrightarrow (x', C')$ has to be an identity arrow and it has to be $\mathrm{id}_{Ff(x)}$. Thus the only arrows are of the form $(\mathrm{id}_{Ff(x)}, f) : (x, C) \longrightarrow (x', C)$. Such an arrow is denoted (x, f) in GS–1 through GS–3.

To visualize the **Cat**-valued Grothendieck construction, we can modify the picture in Diagram (12.4) to get Diagram (12.5). The arrows from inside



(12.5)

one box to inside another, such as the arrow from x to $Ff(x)$, are parts of arrows of $\mathbf{G}(f)$, which are now (in contrast to the discrete case) allowed to miss the target and be rescued by an internal arrow of the codomain category.

Thus in the picture above there is an arrow from x to $Ff(x)$ and $Ff(x)$ is not necessarily x'; the gap is filled by the arrow $u : Ff(x) \longrightarrow x'$ of $F(C')$. The arrow $(u, f) : (x, C) \longrightarrow (x', C')$ of $\mathbf{G}(\mathscr{C}, F)$ may be pictured as the arrow from x to $Ff(x)$ followed by u. Observe that the definition of composition says that the square in the picture with corners $Ff(x)$, $(Fg \circ Ff)(x)$, x' and $Fg(x')$ 'commutes'.

As before, one writes (x, C) for x and (x', C') for x' only to ensure that the union of all the categories of the form $F(C)$ is a *disjoint* union.

12.2.10 An analogous construction, also called the Grothendieck construction (in fact this is the original one), produces a split fibration $\mathbf{F}(\mathscr{C}, G)$ given a functor $G : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Cat}$.

FC–1 An object of $\mathbf{F}(\mathscr{C}, G)$ is a pair (C, x) where C is an object of $\mathscr{C}$ and x is an object of $G(C)$.

FC–2 An arrow $(f, u) : (C, x) \longrightarrow (C', x')$ has $f : C \longrightarrow C'$ an arrow of $\mathscr{C}$ and $u : x \longrightarrow Gf(x')$ an arrow of $G(C)$.

FC–3 If $(f, u) : (C, x) \longrightarrow (C', x')$ and $(g, v) : (C', x') \longrightarrow (C'', x'')$, then the composite $(g, v) \circ (f, u) : (C, x) \longrightarrow (C'', x'')$ is defined by

$$(g, v) \circ (f, u) = (g \circ f, Gf(v) \circ u)$$

12.2.11 In line with the concept that a category is a mathematical workspace, one could ask to construct objects in a suitably rich category which themselves are categories. The Grothendieck construction provides a way to describe functors from such a category object to the ambient category which is worked out in 15.7.2.

12.2.12 Exercises

1. Verify that GS–1 through GS–3 define a category.

2. Show that for any functor $F : \mathscr{C} \longrightarrow \mathbf{Set}$, $\mathbf{G}_0(F) : \mathbf{G}_0(\mathscr{C}, F) \longrightarrow \mathscr{C}$ is a split opfibration.

3. Verify that GC–1 through GC–3 define a category.

4. Show that for any functor $F : \mathscr{C} \longrightarrow \mathbf{Cat}$, $\mathbf{G}(F) : \mathbf{G}(\mathscr{C}, F) \longrightarrow \mathscr{C}$ is a split opfibration.

5. Verify that the definition of the semidirect product in 12.2.6 makes $T \times M$ a monoid.

6. Let $F : \mathscr{C} \longrightarrow \mathbf{Cat}$ be a functor. Show that for each object C of $\mathscr{C}$, the arrows of the form $(u, \mathrm{id}_C) : (x, C) \longrightarrow (y, C)$ (for all arrows $u : x \longrightarrow y$ of $F(C)$) (and their sources and targets) form a subcategory of the opfibration $\mathbf{G}(\mathscr{C}, F)$ which is isomorphic to $F(C)$.

12.3 An equivalence of categories

In this section, we describe how the construction of a functor from an opfibration given in Proposition 12.1.8 (in one direction) produces an equivalence of categories (with the Grothendieck construction as pseudo-inverse) between a category of functors and a suitably defined category of split opfibrations.

12.3.1 Cat-valued functors For a category $\mathscr{C}$, $\mathbf{Func}(\mathscr{C}, \mathbf{Cat})$ is the category whose objects are functors from $\mathscr{C}$ to the category of categories, and whose arrows are natural transformations between them.

If $F : \mathscr{C} \longrightarrow \mathbf{Cat}$ is such a functor and $f : C \longrightarrow D$ is an arrow of $\mathscr{C}$, then $F(C)$ and $F(D)$ are categories and $Ff : F(C) \longrightarrow F(D)$ is a functor. If also $G : \mathscr{C} \longrightarrow \mathbf{Cat}$ and $\alpha : F \longrightarrow G$ is a natural transformation, then for each object of C, $\alpha C : F(C) \longrightarrow G(C)$ is a functor and the following diagram is a commutative diagram of categories and functors:

$$\begin{array}{ccc} F(C) & \xrightarrow{\alpha C} & G(C) \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ F(D) & \xrightarrow[\alpha D]{} & G(D) \end{array} \tag{12.6}$$

12.3.2 The category of split opfibrations of $\mathscr{C}$ Let $P : \mathscr{E} \longrightarrow \mathscr{C}$ and $P' : \mathscr{E}' \longrightarrow \mathscr{C}$ be two split opfibrations of the same category $\mathscr{C}$ with splittings κ and κ' respectively. A **homomorphism of split opfibrations** is a functor $\zeta : \mathscr{E} \longrightarrow \mathscr{E}'$ for which

HSO–1 The diagram

$$\begin{array}{ccc} \mathscr{E} & \xrightarrow{\zeta} & \mathscr{E}' \\ & {\scriptstyle P}\searrow \quad \swarrow{\scriptstyle P'} & \\ & \mathscr{C} & \end{array} \tag{12.7}$$

commutes.

HSO–2 For any arrow $f : C \longrightarrow D$ in $\mathscr{C}$ and object X of $\mathscr{E}$ such that $P(X) = C$,

$$\zeta(\kappa(f, X)) = \kappa'(f, \zeta(X))$$

Thus a homomorphism of split fibrations 'takes fibers to fibers' and 'preserves the splitting'.

12.3.3 Definition Split opfibrations of $\mathscr{C}$ and homomorphisms between them form a category $\mathbf{SO}(\mathscr{C})$.

We will show that $\mathbf{SO}(\mathscr{C})$ is equivalent to $\mathbf{Func}(\mathscr{C}, \mathbf{Cat})$. We do this by defining two functors

$$\mathsf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}, \mathbf{Cat})$$

and

$$\mathsf{G} : \mathbf{Func}(\mathscr{C}, \mathbf{Cat}) \to \mathbf{SO}(\mathscr{C})$$

so that F is an equivalence with pseudo-inverse G as defined in Section 3.4.

12.3.4 Definition For a category $\mathscr{C}$, define the functor

$$\mathsf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}, \mathbf{Cat})$$

as follows:

FI–1 If $P : \mathscr{E} \to \mathscr{C}$ is a split opfibration with splitting κ, then $\mathsf{F}(P, \kappa) : \mathscr{C} \to \mathbf{Cat}$ is the functor F satisfying FF–1 through FF–3 defined in 12.1.7.

FI–2 If $\zeta : (P, \kappa) \to (P', \kappa')$ is a homomorphism of opfibrations, $\mathsf{F}\zeta : \mathsf{F}(P, \kappa) \to \mathsf{F}(P', \kappa')$ is the natural transformation whose component at an object C of $\mathscr{C}$ is the functor ζ restricted to $P^{-1}(C)$.

To show that $F\zeta$ is a natural transformation, it is necessary to show that for every $f : C \to D$ in $\mathscr{C}$ the following diagram commutes:

$$\begin{array}{ccc} \mathsf{F}(P,\kappa)(C) & \xrightarrow{\mathsf{F}\zeta C} & \mathsf{F}(P',\kappa')(C) \\ \Big\downarrow{\scriptstyle \mathsf{F}(P,\kappa)(f)} & & \Big\downarrow{\scriptstyle \mathsf{F}(P',\kappa')(f)} \\ \mathsf{F}(P,\kappa)(D) & \xrightarrow[\mathsf{F}\zeta D]{} & \mathsf{F}(P',\kappa')(D) \end{array} \tag{12.8}$$

Let $u : X \to X'$ be in $\mathsf{F}(P, \kappa)(C)$. Note that u is an arrow in the inverse image $P^{-1}C$, so $Pu = \mathrm{id}_C$. Moreover, ζu is an arrow for which $P'(\zeta u) = \mathrm{id}_C$.

By definition of cleavage, there are unique arrows $\widetilde{u}$ such that $P(\widetilde{u}) = \mathrm{id}_D$ and $\widehat{u}$ such that $P'(\widehat{u}) = \mathrm{id}_D$, for which

$$\widetilde{u} \circ \kappa(f, X) = \kappa(f, X') \circ u$$

and

$$\widehat{u} \circ \kappa'(f, \zeta(X)) = \kappa(f, \zeta(X')) \circ \zeta u$$

The top route in Diagram (12.8) takes u to $\widehat{u}$ and the bottom route takes it to $\zeta\widetilde{u}$. The following calculation shows that the diagram commutes:

$$\begin{array}{rcl} \widehat{u} \circ \kappa'(f, \zeta(X)) & = & \kappa'(f, \zeta(X')) \circ \zeta(u) \\ & = & \zeta(\kappa(f, X')) \circ \zeta(u) \\ & = & \zeta(\kappa(f, X') \circ u) \\ & = & \zeta(\widetilde{u} \circ \kappa(f, X)) \\ & = & \zeta(\widetilde{u}) \circ \zeta(\kappa(f, X)) \\ & = & \zeta(\widetilde{u}) \circ \kappa'(f, \zeta(X)) \end{array} \tag{12.9}$$

so that $\widehat{u} = \zeta(\widetilde{u})$ by the uniqueness requirement in the definition of $\widehat{u}$.

12.3.5 The Grothendieck functor To define the functor going the other way we extend the Grothendieck construction.

12.3.6 Definition For a category $\mathscr{C}$, define the functor

$$\mathsf{G} : \mathbf{Func}(\mathscr{C}, \mathbf{Cat}) \to \mathbf{SO}(\mathscr{C})$$

as follows:

GR–1 For $F : \mathscr{C} \to \mathbf{Cat}$, $\mathsf{G}(F) = \mathsf{G}(\mathscr{C}, F)$.

GR–2 For a natural transformation $\alpha : F \to G : \mathscr{C} \to \mathbf{Cat}$,

$$\mathsf{G}\alpha(x, C) = (\alpha C x, C)$$

for (x, C) an object of $\mathsf{G}(\mathscr{C}, F)$ (so that C is an object of $\mathscr{C}$ and x is an object of FC), and

$$\mathsf{G}\alpha(u, f) = (\alpha C' u, f)$$

for (u, f) an arrow of $\mathsf{G}(\mathscr{C}, F)$ (so that $f : C \to C'$ in $\mathscr{C}$ and $u : Ffx \to x'$ in FC').

Note that in GR–2, $\alpha C' u$ has domain $\alpha C'(Ffx)$, which is $Gf(\alpha C x)$ because α is a natural transformation. The verification that $\mathsf{G}\alpha$ is a functor is omitted.

12.3.7 Theorem *The functor* $\mathsf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}, \mathbf{Cat})$ *defined in 12.3.4 is an equivalence of categories with pseudo-inverse* G.

There is a similar equivalence of categories between split fibrations and contravariant functors. The details are in [Nico, 1983]. Moreover, the non-split case for both fibrations and opfibrations corresponds in a precise way to 'pseudo-functors', which are like functors except that identities and composites are preserved only up to natural isomorphisms. See [Gray, 1966] (the terminology has evolved since that article). A generalization of this equivalence is described in [Pavlović and Abramsky, 1997], page 151, with references to the literature for other generalizations.

12.3.8 Exercises

1. Verify that $\mathbf{G}\alpha$ as defined by GR–2 is a functor.

2. Verify that $\mathbf{G}$ as defined by GR–1 and GR–2 is a functor.

12.4 Wreath products

In this section, we introduce the idea of the wreath product of categories (and of functors), based on an old construction originating in group theory. In the monoid case, this construction allows a type of series-parallel decomposition of finite state machines (the Krohn–Rhodes Theorem). This section is not needed later.

12.4.1 Let $\mathscr{A}$ and $\mathscr{B}$ be small categories and $G : \mathscr{A} \longrightarrow \mathbf{Cat}$ a functor. With these data we define the **shape functor** $S(G, \mathscr{B}) : \mathscr{A}^{\mathrm{op}} \longrightarrow \mathbf{Cat}$ as follows. If A is an object of $\mathscr{A}$, then $S(G, \mathscr{B})(A)$ is the category of functors from the category $G(A)$ to $\mathscr{B}$ with natural transformations as arrows.

Thus an object of $S(G, \mathscr{B})(A)$ is a functor $P : G(A) \longrightarrow \mathscr{B}$ and an arrow from $P : G(A) \longrightarrow \mathscr{B}$ to $P' : G(A) \longrightarrow \mathscr{B}$ is a natural transformation from P to P'. It is useful to think of $S(G, \mathscr{B})(A)$ as the category of diagrams of shape $G(A)$ (or models of $G(A)$) in $\mathscr{B}$; the arrows between them are homomorphisms of diagrams, in other words natural transformations.

> Embedding or modeling a certain shape (diagram, space, structure, etc.) into a certain workspace (category, topological space, etc.) in all possible ways is a tool used all over mathematics. In particular, what we have called the shape functor is very reminiscent of the singular simplex functors in algebraic topology.

We must say what $S(G, \mathscr{B})$ does to arrows of $\mathscr{A}^{\mathrm{op}}$. If $f : A \longrightarrow A'$ is an arrow of $\mathscr{A}$, then $S(G, \mathscr{B})(f) : \mathbf{Func}(G(A'), \mathscr{B}) \longrightarrow \mathbf{Func}(G(A), \mathscr{B})$ takes a functor $H : G(A') \longrightarrow \mathscr{B}$ to $H \circ Gf : G(A) \longrightarrow \mathscr{B}$ and it takes a natural transformation $\alpha : H \longrightarrow H' : G(A') \longrightarrow \mathscr{B}$ to the natural transformation $\alpha Gf : H \circ Gf \longrightarrow H' \circ Gf : G(A) \longrightarrow \mathscr{B}$ whose component at an object X of A is the component of α at $Gf(X)$. This is the usual action of a functor on diagrams in a category.

12.4.2 Since $S(G, \mathscr{B}) : \mathscr{A}^{\mathrm{op}} \longrightarrow \mathbf{Cat}$ is a category-valued functor, we can use the Grothendieck construction to form the split fibration of $\mathscr{A}$ by the functor $S(G, \mathscr{B})$. This fibration consists of a category denoted $\mathscr{A} \operatorname{wr}^G \mathscr{B}$, called the **wreath product** of $\mathscr{A}$ by $\mathscr{B}$ with given action G, and a functor $\Pi : \mathscr{A} \operatorname{wr}^G \mathscr{B} \longrightarrow \mathscr{A}$.

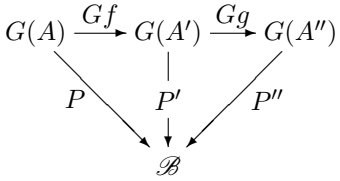
We now unwind what this implies to give an elementary definition of the wreath product.

12.4.3 Definition Given small categories $\mathscr{A}$ and $\mathscr{B}$ and a functor $G : \mathscr{A} \longrightarrow \mathbf{Cat}$, the wreath product $\mathscr{A}\ \mathrm{wr}^G\ \mathscr{B}$ is a category defined as follows:

WP–1 The objects of $\mathscr{A}\ \mathrm{wr}^G\ \mathscr{B}$ are pairs (A, P), where A is an object of $\mathscr{A}$ and $P : G(A) \longrightarrow \mathscr{B}$ is a functor.

WP–2 An arrow $(f, \lambda) : (A, P) \longrightarrow (A', P')$ of $\mathscr{A}\ \mathrm{wr}^G\ \mathscr{B}$ has $f : A \longrightarrow A'$ an arrow of $\mathscr{A}$ and $\lambda : P \longrightarrow P' \circ Gf$ a natural transformation.

WP–3 If $(f, \lambda) : (A, P) \longrightarrow (A', P')$ and $(g, \mu) : (A', P') \longrightarrow (A'', P'')$ are arrows of $\mathscr{A}\ \mathrm{wr}^G\ \mathscr{B}$, as in



then

$$(g, \mu) \circ (f, \lambda) = (g \circ f, \mu \circ Gf \circ \lambda) : (A, P) \longrightarrow (A'', P'')$$

To see the meaning of WP–3, observe that $\lambda : P \longrightarrow P' \circ Gf$ and $\mu : P' \longrightarrow P'' \circ Gg$ are natural transformations. Then

$$\mu \circ Gf : P' \circ Gf \longrightarrow P'' \circ Gg \circ Gf = P'' \circ G(g \circ f)$$

is the natural transformation whose component at an object x of $G(A)$ is the component of μ at $Gf(x)$ (this was described in Section 4.4). Then

$$\mu \circ Gf \circ \lambda : P \longrightarrow P'' \circ G(g \circ f)$$

is the usual composite of natural transformations (see 4.2.11); it is the natural transformation whose component at an object x of $G(A)$ is the composite of the components $(\mu \circ Gf(x)) \circ \lambda x$.

It follows from WP–3 that there is a **projection functor**

$$\Pi : \mathscr{A}\ \mathrm{wr}^G\ \mathscr{B} \longrightarrow \mathscr{A}$$

taking (A, P) to A and (f, λ) to f.

12.4.4 Special cases of the wreath product If the functor G in definition 12.4.3 is set-valued, then one obtains the **discrete wreath product** of $\mathscr{A}$ by $\mathscr{B}$ with action G. When $\mathscr{A}$ and $\mathscr{B}$ are both monoids, the *discrete* wreath product is also a monoid. (The general case need not be a monoid.)

12.4.5 Definition For any small category $\mathscr{C}$, the **right regular representation** of $\mathscr{C}$ is the functor $R_{\mathscr{C}} : \mathscr{C} \longrightarrow \mathbf{Set}$ defined as follows:

RR–1 If C is an object of $\mathscr{C}$, then $R_{\mathscr{C}}(C)$ is the set of arrows of $\mathscr{C}$ with codomain C.

RR–2 If $f : C \longrightarrow C'$ in $\mathscr{C}$ and $g \in R_{\mathscr{C}}(C)$, then $R_{\mathscr{C}}(f)(g) = f \circ g$.

For small categories $\mathscr{A}$ and $\mathscr{B}$, the **standard wreath product** $\mathscr{A} \operatorname{wr} \mathscr{B}$ is the wreath product $\mathscr{A} \operatorname{wr}^{R_{\mathscr{A}}} \mathscr{B}$. This is a generalization of what is called the standard wreath product for groups and monoids. It is the wreath product used in [Rhodes and Tilson, 1989]. They also have a two-sided version of the wreath product.

12.4.6 The action induced by a wreath product Given small categories $\mathscr{A}$ and $\mathscr{B}$ and functors $G : \mathscr{A} \longrightarrow \mathbf{Cat}$ and $H : \mathscr{B} \longrightarrow \mathbf{Cat}$, there is an induced functor $G \operatorname{wr} H : \mathscr{A} \operatorname{wr}^G \mathscr{B} \longrightarrow \mathbf{Cat}$ defined as follows:

WF–1 For an object (A, P) of $\mathscr{A} \operatorname{wr}^G \mathscr{B}$, $(G \operatorname{wr} H)(A, P)$ is the split opfibration induced by $H \circ P : G(A) \longrightarrow \mathbf{Cat}$.

WF–2 If (h, λ) is an arrow of $\mathscr{A} \operatorname{wr}^G \mathscr{B}$ with domain (A, P), and (t, x) is an object of $(G \operatorname{wr} H)(A, P)$, so that x is an object of $G(A)$ and t is an object of $H(P(x))$, then

$$(G \operatorname{wr} H)(h, \lambda)(t, x) = (H\lambda x(t), Gh(x))$$

WF–3 If $(u, f) : (t, x) \longrightarrow (t', x')$ is an arrow of $(G \operatorname{wr} H)(P, A)$, then

$$(G \operatorname{wr} H)(h, \lambda)(u, f) = (H(\lambda x')(u), Gh(f))$$

WF–1 can be perceived as saying that $G \operatorname{wr} H$ is obtained by composing the shapes given by G (see the discussion in 12.4.1) with H. Indeed, G. M. Kelly, who invented this concept [1974] called what we call the wreath product the 'composite' of the categories. That is in some ways a better name: the word 'product' suggests that the two factors are involved in the product in symmetric ways, which is not the case, as the next subsection describes.

12.4.7 The action $G \operatorname{wr} H$ of $\mathscr{A} \operatorname{wr}^G \mathscr{B}$ just defined is said to be **triangular** because it is a precise generalization of the action of a triangular matrix. For example, the action

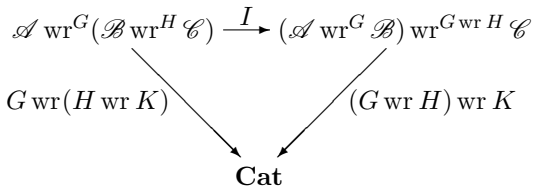
$$\begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cy \end{bmatrix}$$

can be described this way: the effect on the first coordinate depends on both the first and second coordinates, but the effect on the second coordinate depends only on the second coordinate.

The dependency of the action on the coordinates given in WF–2 and WF–3 is analogous to the dependency for the matrices in the example just given.

The wreath product can be generalized to many factors, using the following theorem, proved in [Kelly, 1974], Section 7. This theorem allows one to think of the wreath product as generalizing triangular matrices bigger than 2×2.

12.4.8 Proposition *Let $G : \mathscr{A} \to \mathbf{Cat}$, $H : \mathscr{B} \to \mathbf{Cat}$ and $K : \mathscr{C} \to \mathbf{Cat}$ be functors. Then there is an isomorphism of categories I making this diagram commute.*



Note that the *standard* wreath product is not associative.

12.4.9 Applications of the wreath product It is natural to wish to simulate complicated state transition systems using systems built up in some way from a small stock of simpler ones. This requires a precise notion of simulation. This is defined in various ways in the literature. In some cases one says a functor $F : \mathscr{C} \to \mathscr{D}$ is a simulation of $\mathscr{D}$ (the word 'cover' is often used) if it has certain special properties. Other authors have the functor going the other way.

The Krohn–Rhodes Theorem for monoids says that every finite monoid action is simulated by an iterated wreath product of finite simple groups and certain very small monoids. The original Krohn–Rhodes Theorem was stated for semigroups. Discussions are in [Wells, 1976] and [Eilenberg, 1976]; the latter uses a different definition of cover. Wells [1980] proved a generalization of a weak form of the Krohn–Rhodes Theorem for finite categories and set-valued functors (which generalize the concept of action by a monoid, as discussed in Section 3.2). Wells [1988a, 1988b] describes how to use some of these decomposition techniques for category-valued functors.

Rhodes and Tilson use the idea of 'division' of categories, which is related to the notion of cover, as the basic idea of an extensive study of varieties of semigroups and complexity. See See [Rhodes and Tilson, 1989, Rhodes and Weil, 1989].

Nico [1983] defines a category induced by any functor called the 'kernel category' of the functor and proves a theorem which embeds the domain

of the functor into the standard wreath product of the codomain and the kernel. This generalizes an old theorem of Kaloujnine and Krasner. It follows that every functor factors as a full and faithful functor which is injective on objects, followed by a fibration. Street and Walters [1973] have a related theorem.

In the case of groups, the kernel category of a homomorphism is a category equivalent, but not isomorphic, to the actual kernel of the homomorphism. In the case that $\mathscr{C}$ and $\mathscr{D}$ are monoids, the kernel category (which is not a monoid in general) is called the **derived category** of F. Rhodes and Tilson [1989] have a tighter definition obtained by imposing a congruence on the kernel. A theorem analogous to Nico's theorem is true for the tighter definition, as well. It is stated for semigroups and relations, not merely monoids and homomorphisms, so it is neither more nor less general than Nico's theorem. In the semigroup case, the 'category' is replaced by a 'semigroupoid', which is like a category but does not have to have identity arrows.

12.4.10 Exercises

1. Show that the discrete wreath product of two monoids is a monoid.

2. Show that the wreath product of two groups (monoids in which every element is invertible) regarded as categories is a category in which every arrow is an isomorphism.

13
Adjoints

Adjoints are about the most important idea in category theory, except perhaps for the closely related notion of representable functors. Many constructions made earlier in this book are examples of adjoints, as we will describe. Moreover, toposes are most conveniently described in terms of adjoints.

This chapter develops enough of the basic theory of adjoints to read the last two chapters of the book. Section 13.1 revisits the concept of free monoids, describing them in a way which suggests the general definition of adjoint. Adjoints are described and some basic properties developed in Sections 13.2 and 13.3. Section 13.4 describes locally cartesian closed categories, which are best described using adjoints.

More detail concerning adjoints can be found in [Barr and Wells, 1985] and [Mac Lane, 1971]. Diers [1980a], [1980b] describes a generalization which includes the initial families of FD sketches (see Section 8.3). Hagino [1987a], [1987b] has a general approach to type constructors for functional programming languages based on adjoints (see also [Chen and Cockett, 1989]).

Much of this chapter can be read after completing Section 4.5. Some of the material in the chapter requires the concepts of limit and colimit from Chapters 5 and 9. One major example of an adjoint, cartesian closed categories, requires Sections 5.1 and 6.1. The concept of adjoint is used heavily in the rest of the book, but Section 13.4 is not needed later.

13.1 Free monoids

In 3.1.14, we gave the universal property of the free monoid. We now state it more carefully than we did there, paying closer attention to the categories involved. Note that when we speak of a sub*set* of a monoid, we are mixing two things. A monoid is not just a set and most of its subsets are not submonoids. To describe a monoid, you must give three data: the set of elements, the operation and the identity element. From this point of view the phrase 'subset of a monoid' does not make sense. It is actually a subset of the underlying set of the monoid. Insistence on this is not pointless pedantry; it is the key to understanding the situation.

Let **Mon** denote the category of monoids and let $U : \mathbf{Mon} \to \mathbf{Set}$ denote the underlying set functor. We define a monoid M to be a triple $(UM, \cdot, 1)$,

where UM is a set called the **underlying set** of M, $\cdot$ is a binary operation on UM and $1 \in UM$ is the identity element for that operation. A homomorphism from a monoid M to a monoid N is a function $f : UM \longrightarrow UN$ which preserves the operation and the identity element. The underlying functor is defined on homomorphisms by $Uf = f$.

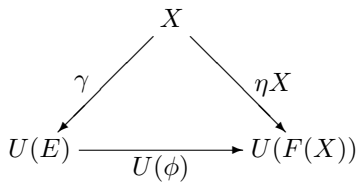
13.1.1 Characterization of the free monoid Given a set X, the free monoid $F(X) = (X^*, \cdot, \langle\rangle)$ is characterized by the property given in the following proposition; the property is called the **universal mapping property** of the free monoid. In the proposition, $\eta X : X \longrightarrow X^* = UF(X)$ takes $x \in X$ to the string $\langle x \rangle$ of length 1. We systematically distinguish the free monoid $F(X)$ from its underlying set, the Kleene closure $X^* = U(F(X))$.

13.1.2 Proposition *Let X be a set. For any monoid M and any function $u : X \longrightarrow U(M)$, there is a unique monoid homomorphism $g : F(X) \longrightarrow M$ such that $u = Ug \circ \eta X$.*

Proof. Define $g(\langle\rangle)$ to be the identity element of M, $g(\langle x \rangle) = u(x)$ for $x \in X$, and $g(\langle x_1 \cdots x_n \rangle)$ to be the product $u(x_1) \cdot \cdots \cdot u(x_n)$ in M. That makes g a monoid homomorphism and $u = Ug \circ \eta X$; the details are left as an exercise. □

Just as the theory of a sketch is the universal model of that sketch, the free monoid generated by a set is universal for functions from that set into the set underlying a monoid.

This proposition says that the arrow ηX is a universal element for the functor $\mathrm{Hom}(X, U-)$ that takes a monoid M to $\mathrm{Hom}(X, U(M))$ and a monoid homomorphism $f : M \longrightarrow N$ to $\mathrm{Hom}(X, U(f)) : \mathrm{Hom}(X, U(M)) \longrightarrow \mathrm{Hom}(X, U(N))$. It follows that the universal mapping property characterizes the free monoid up to a unique isomorphism. Precisely, let X be a set and $\gamma : X \longrightarrow U(E)$ a function to the underlying set of some monoid E with the property that if $u : X \longrightarrow U(M)$, where M is any monoid, there is a unique monoid homomorphism $f : E \longrightarrow M$ such that $u = U(f) \circ \gamma$. We now show that there is a unique isomorphism $\phi : E \longrightarrow F(X)$ for which



commutes. As we have noted above, ηX is a universal element of the functor $\mathrm{Hom}(X, U(-))$. The assumption on γ says that it is also a universal element

for the same functor. Now Corollary 4.5.13 says there is a unique isomorphism $\phi : E \longrightarrow F(X)$ for which $\mathrm{Hom}(X, U(\phi))(\gamma) = \eta X$. The result follows because $\mathrm{Hom}(X, U(\phi))(\gamma) = U(\phi) \circ \gamma$ by definition.

13.1.3 The free monoid functor Each set X generates a free monoid $F(X)$. In 3.1.12, we extended this to a functor $F : \mathbf{Set} \longrightarrow \mathbf{Mon}$. In this section, we will prove this (as part of the proof of Proposition 13.1.4 below) using only the universal mapping property of free monoids; thus the argument will work in complete generality.

Let ηY denote the arrow from Y into $Y^* = UFY$ described in 13.1.1 and let $f : X \longrightarrow Y$ be a function. Then $\eta Y \circ f : X \longrightarrow Y^*$ is a function from X into the set underlying a monoid. The universal property of $F(X)$ gives a unique monoid homomorphism $F(f) : F(X) \longrightarrow F(Y)$ such that if $f^* = UF(f)$, then

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle \eta X}\downarrow & & \downarrow{\scriptstyle \eta Y} \\ X^* & \xrightarrow[f^*]{} & Y^* \end{array} \tag{13.1}$$

commutes.

13.1.4 Proposition *$F : \mathbf{Set} \longrightarrow \mathbf{Mon}$ is a functor and, for any set X, ηX is the component at X of a natural transformation $\eta : \mathrm{id}_{\mathbf{Set}} \longrightarrow U \circ F$.*

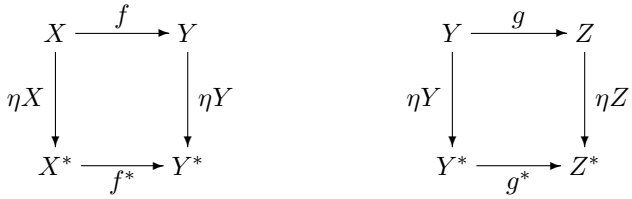
Proof. Once F is shown to be a functor, that η is a natural transformation will follow from Diagram (13.1).

First note that if $\mathrm{id} : X \longrightarrow X$ is the identity, then $F(\mathrm{id})$ is the unique arrow $h : F(X) \longrightarrow F(X)$ such that

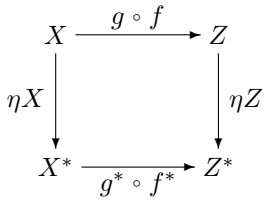
$$\begin{array}{ccc} X & \xrightarrow{\mathrm{id}} & X \\ {\scriptstyle \eta X}\downarrow & & \downarrow{\scriptstyle \eta X} \\ X^* & \xrightarrow[Uh]{} & X^* \end{array}$$

commutes. But if we replace h in the diagram above by id_{FX}, the diagram still commutes. The uniqueness property of the free monoid implies that $h = \mathrm{id}_{FX}$.

Similarly, if $g : Y \to Z$ is a function, the commutativity of both squares in



implies that



commutes. But $g^* \circ f^* = UF(g) \circ UF(f) = U(F(g) \circ F(f))$ since U is a functor. This means that the arrow $F(g) \circ F(f)$ satisfies the same equation that characterizes $F(g \circ f)$ uniquely and hence that they are equal. This shows that F is a functor and that η is a natural transformation. □

Another example of functors F and U satisfying this is the free category given by a graph as described in 2.6.16. This very same proof shows that the free category construction is really the object part of a functor.

13.1.5 Exercise

1. Show that the function g defined in the proof of Proposition 13.1.2 is a monoid homomorphism from $F(X)$ to M.

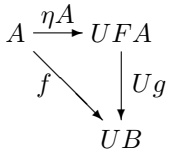
13.2 Adjoints

The relationship between the free monoid functor and the underlying set functor is an example of a very general situation, an adjunction, defined below.

In this section, we have several notational shortcuts to prevent clutter of parentheses and composition circles. This notation is quite standard in the literature on adjoints. For example, an expression $UFUA$ is shorthand for $(U \circ F \circ U)(A)$, which is the same as $U(F(U(A)))$.

13.2.1 Definition Let $\mathscr{A}$ and $\mathscr{B}$ be categories. If $F : \mathscr{A} \to \mathscr{B}$ and $U : \mathscr{B} \to \mathscr{A}$ are functors, we say that F is **left adjoint** to U and U is **right adjoint** to F provided there is a natural transformation $\eta : \mathrm{id} \to UF$ such

that for any objects A of $\mathscr{A}$ and B of $\mathscr{B}$ and any arrow $f : A \longrightarrow UB$, there is a unique arrow $g : FA \longrightarrow B$ such that



(13.2)

commutes.

This definition asserts that there is a functional way to convert any arrow $f : A \longrightarrow UB$ to an arrow $g : FA \longrightarrow B$ in such a way that g solves the equation $f = U(?) \circ \eta A$, and that the solution is unique.

The property of η given in the last sentence of Definition 13.2.1 is called its **universal mapping property**. The existence of the unique arrow $g : FA \longrightarrow B$ such that $f = Ug \circ \eta A$ is the map-lifting property of Section 3.1.14. Just as in the discussion after Proposition 13.1.2, for each object A, the arrow ηA is a universal element for the functor $\mathrm{Hom}(A, U(-))$.

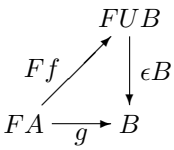
It is customary to write $F \dashv U$ to denote the situation described in the definition. The data (F, U, η) constitute an **adjunction**. The transformation η is called the **unit** of the adjunction.

In some texts the adjunction is written as a rule of inference, like this:

$$\frac{A \longrightarrow UB}{FA \longrightarrow B} \tag{13.3}$$

The definition appears to be asymmetric in F and U. This is remedied in the next proposition, whose proof is deferred to the next section, after the proof of Theorem 13.3.2.

13.2.2 Proposition *Suppose $F : \mathscr{A} \longrightarrow \mathscr{B}$ and $U : \mathscr{B} \longrightarrow \mathscr{A}$ are functors such that $F \dashv U$. Then there is a natural transformation $\epsilon : FU \longrightarrow \mathrm{id}_{\mathscr{B}}$ such that for any $g : FA \longrightarrow B$, there is a unique arrow $f : A \longrightarrow UB$ such that*



(13.4)

The transformation ϵ is called the **counit** of the adjunction.

It is an immediate consequence of categorical duality that this proposition is reversible and the adjunction is equivalent to the existence of either natural transformation η or ϵ with its appropriate universal mapping property.

13.2.3 Examples of adjoints We have of course the example of free monoids that introduced this chapter. In general, let $\mathscr{C}$ be a category of sets with structure and functions which preserve the structure, and let $U : \mathscr{C} \longrightarrow \mathbf{Set}$ be the underlying set functor. If U has a left adjoint F and S is a set, then $F(S)$ is the **free structure** on S. This description fits structures you may know about, such as free groups, free Abelian groups and free rings.

We now give three related examples that illustrate how widespread adjoints are.

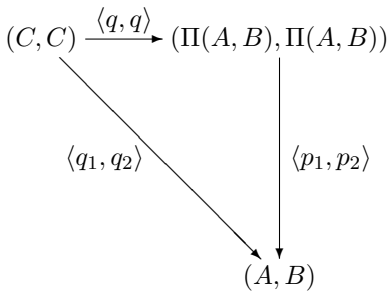
13.2.4 Example Let $\mathscr{C}$ be a category. Consider the category $\mathscr{C} \times \mathscr{C}$ which has as objects pairs of objects (A, B) of $\mathscr{C}$ and in which an arrow $(A, B) \longrightarrow (A', B')$ is a pair (f, g) of arrows $f : A \longrightarrow A'$ and $g : B \longrightarrow B'$. There is a functor $\Delta : \mathscr{C} \longrightarrow \mathscr{C} \times \mathscr{C}$ given by $\Delta(A) = (A, A)$ and $\Delta(f) = (f, f)$. Let us see what a right adjoint to this functor is.

Assuming there is a right adjoint Π to Δ, there should be an arrow we call

$$\langle p_1, p_2 \rangle : \Delta\Pi(A, B) = (\Pi(A, B), \Pi(A, B)) \longrightarrow (A, B)$$

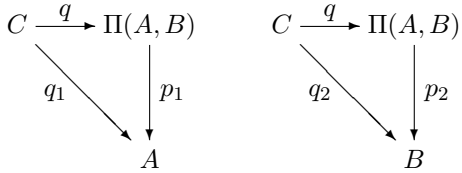
(this is the counit of the adjunction) with the following universal property:

PP For any object C of $\mathscr{C}$ and any arrow $\langle q_1, q_2 \rangle : \Delta C = (C, C) \longrightarrow (A, B)$ there is a unique arrow $q : C \longrightarrow \Pi(A, B)$ such that



commutes.

A diagram in a product category commutes if and only if each component does. Breaking the triangle into components the adjunction asserts the existence of a unique map $q : C \longrightarrow \Pi(A, B)$ such that each of the triangles



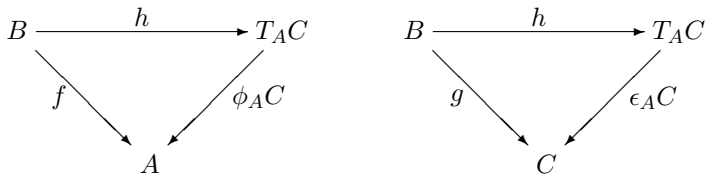
commutes. If you write $A \times B$ instead of $\Pi(A, B)$ you will recover the categorical definition of the product. Thus a right adjoint to Δ is just a functor $\Pi : \mathscr{C} \times \mathscr{C} \longrightarrow \mathscr{C}$ that chooses a product for each pair of objects of $\mathscr{C}$. Such a functor Π is called a **binary product functor**.

13.2.5 Example Now let us suppose we are given a binary product functor Π. We can fix an object A of $\mathscr{C}$ and consider the functor denoted $- \times A : \mathscr{C} \longrightarrow \mathscr{C}$ whose value at an object B is the object $B \times A$ and at an arrow $f : B \longrightarrow C$ is the arrow $f \times \mathrm{id}_A : B \times A \longrightarrow C \times A$ (see 5.2.17).

A right adjoint to the functor $- \times A$, if one exists, can be described as follows. Denote the value at C of this adjoint by $R_A(C)$. Then there is an arrow $e : R_A(C) \times A \longrightarrow C$ with the universal property that for any arrow $f : B \times A \longrightarrow C$, there is a unique arrow we may call $\lambda f : B \longrightarrow R_A(C)$ such that $e \circ (\lambda f \times A) = f$. But this is precisely the universal property that describes the exponential $[A \longrightarrow C]$ (see 6.1.3). The counit e is the arrow eval defined there. The essential uniqueness of adjoints (see 13.3.4 below) implies that this adjunction property determines the exponential, if it exists, up to isomorphism.

Thus the two main defining properties defining cartesian closed categories, namely the existence of binary products and the existence of exponentials, can be (and commonly are) described in terms of adjoints.

13.2.6 Example We can describe the functor $- \times A$ of 13.2.5 in a slightly different way. For a category $\mathscr{C}$ and object A of $\mathscr{C}$ we described the slice category $\mathscr{C}/A$ in 2.6.10 and functor $U_A : \mathscr{C}/A \longrightarrow \mathscr{C}$ in 3.1.11. If U_A has a right adjoint $P_A : \mathscr{C} \longrightarrow \mathscr{C}/A$, then P_A must associate to each object C of $\mathscr{C}$ an object $\phi_A C = P_A(C) : T_A C \longrightarrow A$ of $\mathscr{C}/A$ and an arrow (the counit) $\epsilon_A C : T_A C \longrightarrow C$. This object and arrow must have the universal mapping property that for any other object $f : B \longrightarrow A$ of $\mathscr{C}/A$, and any arrow $g : B \longrightarrow C$ there is a unique arrow $h : B \longrightarrow T_A C$ such that



commute. The left triangle must commute in order to have an arrow in $\mathscr{C}/A$ and the right hand triangle must commute for its universal mapping property. It is evident from this description that $T_A C$ is $C \times A$ and $\epsilon_A C$ and $\phi_A C$ are the first and second projections. Thus $P_A(C)$ is the object $p_2 : C \times A \longrightarrow A$ of $\mathscr{C}/A$.

We return to this topic in Exercise 5 of Section 13.3 and in Section 13.4.

13.2.7 Adjoints to the inverse image functor Here is another example of an adjoint. If S is a set, the set of subsets of S is a poset ordered by inclusion. This becomes a category in the usual way. That is, if S_0 and S_1 are subsets of S, then there is exactly one arrow $S_0 \longrightarrow S_1$ if and only if $S_0 \subseteq S_1$. You should not view this arrow as representing a function, but just a formal arrow that represents the inclusion. We will call this category $\mathrm{Sub}(S)$.

If $f : S \longrightarrow T$ is a function, then with any subset $T_0 \subseteq T$, we get a subset, denoted

$$f^{-1}(T_0) = \{s \in S \mid f(s) \in T_0\}$$

which is called the **inverse image** under f of T_0. If $T_0 \subseteq T_1$, then $f^{-1}(T_0) \subseteq f^{-1}(T_1)$. This means that f^{-1} is a functor from $\mathrm{Sub}(T) \longrightarrow \mathrm{Sub}(S)$. This functor turns out to have both left and right adjoints.

The left adjoint is a familiar one (to mathematicians at least), the so-called **direct image.** For $S_0 \subseteq S$, let

$$f_*(S_0) = \{f(s) \mid s \in S_0\}$$

Then $f_*(S_0) \subseteq T_0$ if and only if $S_0 \subseteq f^{-1}(T_0)$ (proof left as an exercise) which is just the statement that the direct image is left adjoint to the inverse image.

The right adjoint of the inverse image is usually denoted $f_!$ and is defined by saying that $t \in f_!(S_0)$ if and only if $f^{-1}(\{t\}) \subseteq S_0$. Another way of saying this is that *every* element of the inverse image of t is in S_0, while $t \in f_*(S)$ if and only if *some* element of the inverse image of t is in S_0.

We discussed these constructions in 3.1.18 and 3.1.19 for **Set**.

13.2.8 Adjoints in posets Example 13.2.7 is a special case of an adjunction between ordered or even preordered sets. Let P and Q be preordered sets, that is categories in which there is at most one arrow between any pair of objects. We will use $\leq$ for the preorder relation, and the reader may find it more natural to think of this relation as a partial ordering (antisymmetric as well as reflexive and transitive). If $f : P \longrightarrow Q$ and $u : Q \longrightarrow P$ are functors, that is, order preserving functions, then $f \dashv u$ means, when translated into poset language (where uniqueness is not a consideration, since there is at most one arrow between any two objects), that

$$x \leq (u \circ f)(x) \text{ for all } x \in X \tag{13.5}$$

and

$$x \leq u(y) \text{ implies } f(x) \leq y \tag{13.6}$$

for all $x \in X$ and $y \in Y$. We claim that these two conditions together are equivalent to the condition

$$f(x) \leq y \text{ if and only if } x \leq u(y) \tag{13.7}$$

In fact, assuming (13.5) and (13.6), then if $f(x) \leq y$, we have

$$x \leq (u \circ f)(x) \leq u(y)$$

Condition (13.7) follows from this and (13.6). Conversely if (13.7) holds, then (13.6) is immediate and $f(x) \leq f(x)$ implies that $x \leq (u \circ f)(x)$. This is a special case of the Hom set adjunction of the next section (Theorem 13.3.2).

A very common situation involves a pair of contravariant, that is order reversing, functors. In that case, we can dualize either P or Q. Suppose we do the former. Then we have a pair of functors $f : P^{\mathrm{op}} \to Q$ and $u : Q \to P^{\mathrm{op}}$. If $f \dashv u$, then, translating this condition back to P, we see that $f(x) \leq y$ if and only if $u(y) \leq x$. Such a pair of contravariant functors is often called a **Galois connection** between P and Q, named after the first one produced by Galois.

> If you know Galois theory of fields, then you know that E is a Galois extension of F, then there is a one-one order reversing correspondence between subfields of E that include F and subgroups of the Galois group of E over F. This particular Galois correspondence is an equivalence and some people prefer to restrict the use of the term 'Galois connection' to the case of equivalence, while others use it as we have and call the other a Galois equivalence.

13.2.9 Exercises

1. Show that if $f : S \to T$ is a function between sets, then for $S_0 \subseteq S$ and $T_0 \subseteq T$, $S_0 \subseteq f^{-1}(T_0)$ if and only if $f_*(S_0) \subseteq T_0$.

2. Show that a left adjoint to the functor Δ of 13.2.3, if it exists, takes the pair of objects $A,\ B$ to the sum $A + B$.

3. Find the unit of the adjunction $\Delta \dashv \Pi$ described in Section 13.2.3.

4. Let $\mathbf{R}$ and $\mathbf{Z}$ denote the ordered sets of real numbers and integers, respectively, considered as categories. Show that the inclusion $\mathbf{Z} \subseteq \mathbf{R}$ has both a left and a right adjoint and identify them.

5.† Show that the construction of free models of a linear sketch in 4.7.17 gives an example of an adjunction analogous to the free monoid functor.

6. Show that the construction of the path category of a graph in 2.6.16 is left adjoint to the underlying functor from categories to graphs.

7. Find left and right adjoints to the underlying set of objects functor from **Cat** to **Set** (see Example 3.1.10).

8. Show that the underlying arrow set functor of Example 3.1.10 from **Cat** to **Set** has a left adjoint but not a right adjoint.

9. Show that for any set A with other than one element, the functor $- \times A : \mathbf{Set} \to \mathbf{Set}$ defined in 13.2.5 does not have a left adjoint in **Set**. (Hint: $1 \times A$ is isomorphic to A.)

10. In Section 13.2.5, a right adjoint to the functor $- \times A$ is described. The counit of the adjunction is the transformation e defined there. Describe the unit in general and in **Set**.

13.3 Further topics on adjoints

13.3.1 Hom set adjointness There is an alternative formulation of adjointness which is often found in the categorical literature.

13.3.2 Theorem *If $\mathscr{A}$ and $\mathscr{B}$ are categories and $U : \mathscr{B} \to \mathscr{A}$ and $F : \mathscr{A} \to \mathscr{B}$ are functors, then $F \dashv U$ if and only if* $\mathrm{Hom}(F-,-)$ *and* $\mathrm{Hom}(-,U-)$ *are naturally isomorphic as functors $\mathscr{A}^{\mathrm{op}} \times \mathscr{B} \to$ **Set**.*

Proof. Let $F \dashv U$, and let A and B be objects of $\mathscr{A}$ and $\mathscr{B}$ respectively. Define $\beta_{A,B} : \mathrm{Hom}(FA, B) \to \mathrm{Hom}(A, UB)$ by $\beta_{A,B}(g) = Ug \circ \eta A$, and $\gamma_{A,B} : \mathrm{Hom}(A, UB) \to \mathrm{Hom}(FA, B)$ by requiring that $\gamma_{A,B}(f)$ be the unique arrow g such that $f = Ug \circ \eta A$ given by Definition 13.2.1. Then $\gamma_{A,B}(\beta_{A,B}(g)) = g$ by the uniqueness requirement of Definition 13.2.1, and $\beta_{A,B}(\gamma_{A,B}(f)) = f$ by definition of $\beta_{A,B}$ and $\gamma_{A,B}$. Thus $\beta_{A,B}$ is an isomorphism with inverse $\gamma_{A,B}$. The naturality is left as an exercise (Exercise 3).

To go in the other direction, suppose we have the natural isomorphism. Then let A be an arbitrary object of $\mathscr{A}$ and $B = FA$. We then get $\mathrm{Hom}(FA, FA) \cong \mathrm{Hom}(A, UFA)$. Let $\eta A \in \mathrm{Hom}(A, UFA)$ be the arrow that corresponds under the isomorphism to the identity arrow of FA. Now for an arrow $f : A \to UB$, that is $f \in \mathrm{Hom}(A, UB)$, let $g \in \mathrm{Hom}(FA, B)$ be the arrow that corresponds under the isomorphism. Naturality of the isomorphism implies that we have a commutative diagram

$$\begin{array}{ccc} \mathrm{Hom}(FA, FA) & \xrightarrow{\cong} & \mathrm{Hom}(A, UFA) \\ \Big\downarrow{\scriptstyle \mathrm{Hom}(FA, g)} & & \Big\downarrow{\scriptstyle \mathrm{Hom}(A, Ug)} \\ \mathrm{Hom}(FA, B) & \xrightarrow[\cong]{} & \mathrm{Hom}(A, UB) \end{array}$$

If we follow the identity arrow of FA around the clockwise direction, we get first the arrow ηA by definition, and then $\mathrm{Hom}(A, Ug)(\eta A) = Ug \circ \eta A$. In the other direction, we get $\mathrm{Hom}(FA, g)(\mathrm{id}) = g \circ \mathrm{id} = g$ and that corresponds under the isomorphism to f. Thus we conclude that $f = Ug \circ \eta A$. As for the

uniqueness, if $f = Uh \circ \eta A$, then both g and h correspond to f under the isomorphism, so $g = h$. □

13.3.3 Proof of Proposition 13.2.2 The hypotheses of the above theorem are symmetric in F and U and therefore the conclusion must be too. Thus categorical duality implies the existence of ϵ and the requisite universal mapping property. □

13.3.4 Uniqueness of adjoints If $U : \mathscr{B} \longrightarrow \mathscr{A}$ is a functor, then a left adjoint to U, if one exists, is unique up to natural isomorphism. The reason is that if both F and F' are left adjoint to U, then for any object A of $\mathscr{A}$, the Hom functors $\mathrm{Hom}_{\mathscr{B}}(FA, -)$ and $\mathrm{Hom}_{\mathscr{B}}(F'A, -)$ are each naturally isomorphic to $\mathrm{Hom}_{\mathscr{A}}(A, U-)$ and hence to each other. It follows from the Yoneda embedding, Theorem 4.5.3, that $FA \cong F'A$. The naturality of the latter isomorphism follows from the next theorem.

13.3.5 Theorem *Let $\mathscr{A}$ and $\mathscr{B}$ be categories and $U : \mathscr{B} \longrightarrow \mathscr{A}$ be a functor. Suppose for each object A of $\mathscr{A}$ there is an object FA of $\mathscr{B}$ such that* $\mathrm{Hom}_{\mathscr{B}}(FA, -)$ *is naturally equivalent to* $\mathrm{Hom}_{\mathscr{A}}(A, U-)$ *as a functor from $\mathscr{B}$ to* **Set**. *Then the definition of F on objects can be extended to arrows in such a way that F becomes a functor and is left adjoint to U.*

This theorem is called the **Pointwise Adjointness Theorem**. Its proof generalizes the argument of 13.1.3 but we omit the details. They can be found in [Barr and Wells, 1985], Section 1.9, Theorem 1, page 52. There is a detailed, general discussion of many equivalent definitions of adjunction in [Mac Lane, 1971], Chapter IV.

One application of this theorem is in showing that the definition of cartesian closed categories given in Chapter 6 is equivalent to the assumptions that the functors Δ of 13.2.3 and $- \times A$ of 13.2.5 have adjoints. In each case, the definition given in Chapter 6 can be input to the Pointwise Adjointness Theorem and the output is the adjoint described in the previous section.

This theorem has a simple formulation in terms of the universal elements of Section 4.5, as follows.

13.3.6 Proposition *A functor $U : \mathscr{B} \longrightarrow \mathscr{A}$ has a left adjoint if and only if for each object A of $\mathscr{A}$, the functor* $\mathrm{Hom}(A, U-) : \mathscr{B} \longrightarrow$ **Set** *has a universal element.*

If $b : A \longrightarrow UB$ is the universal element, then $FA = B$ and $b : A \longrightarrow UB = UFA$ is the component at A of the natural transformation η that appears in 13.2.1.

As an example of how this proposition can be used, one can use it to deduce from Proposition 13.1.2 that $X \mapsto F(X)$ is the object map of a left

adjoint to the underlying functor $U : \mathbf{Mon} \to \mathbf{Set}$. Of course, we proved this directly in 13.1.3.

13.3.7 Theorem *Let $F : \mathscr{A} \to \mathscr{B}$ be left adjoint to $U : \mathscr{B} \to \mathscr{A}$. Then U preserves limits and F preserves colimits.*

Proof. We sketch the proof that U preserves limits; the details are easy, but the notation is slightly unpleasant. Recall the definition of the cone functor $\mathrm{cone}(-, D) : \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$ from 9.2.6. Then we have, for a diagram $D : \mathscr{I} \to \mathscr{B}$ with limit given by a cone $V \to D$, the following equivalences of contravariant functors:

$$\mathrm{cone}(-, UD) \cong \mathrm{cone}(F-, D) \cong \mathrm{Hom}(F-, V) \cong \mathrm{Hom}(-, UV)$$

which shows that UV is a limit of UD. The first isomorphism in that equation should be verified, since we extended the isomorphism of Theorem 13.3.2 from the hom functor to the cone functor, but that verification is straightforward. The second isomorphism follows from the fact that the cone $V \to D$ is a universal element of $\mathrm{cone}(-, D)$, so that $\mathrm{cone}(-, D) \cong \mathrm{Hom}(-, V)$. The third isomorphism follows from Theorem 13.3.2. □

The interesting question is the extent to which the converse is true. The basic fact is that the converse is false. First, the category $\mathscr{B}$ may not have enough limits for the condition to be meaningful. The really interesting case is the one in which every (small) diagram in $\mathscr{B}$ has a limit. Even in that case, there is still what can basically be described as a size problem. To go into this in more detail is beyond the scope of this book. The best result is Freyd's Adjoint Functor Theorem. See [Barr and Wells, 1985], Section 1.9, Theorem 3, page 54 or [Mac Lane, 1971], Section V.6. Another reference for the Adjoint Functor Theorem (with a different point of view) is [Freyd and Scedrov, 1990], pages 144–146.

13.3.8 Exercises

1. Suppose that we have categories $\mathscr{A}$ and $\mathscr{B}$ and functors $T : \mathscr{A} \to \mathscr{B}$ and $L, R : \mathscr{B} \to \mathscr{A}$ such that $L \dashv T \dashv R$. Show that $LT \dashv RT$ as endofunctors on $\mathscr{A}$ and $TL \dashv TR$ as endofunctors on $\mathscr{B}$.

2. a. Suppose that X is a set of real numbers, considered as an ordered set, considered as a category. Show that a limit of the inclusion of $X \subseteq \mathbf{R}$, if any, is the infimum of X, if that exists, and similarly that the colimit of that inclusion is the supremum of X.

b. Referring to Exercise 4 of the previous section, show that the ceiling function preserves supremum, but not necessarily infimum, and the floor function preserves infimum, but not necessarily supremum.

3. Show that $\beta_{A,B}$, as defined in the proof of Theorem 13.3.2, is a natural transformation from $\mathrm{Hom}_{\mathscr{B}}(F-,-)$ to $\mathrm{Hom}_{\mathscr{A}}(-,U-)$, both functors from $\mathscr{A}^{\mathrm{op}} \times \mathscr{B}$ to **Set**.

4. For any functor $G : \mathscr{C} \to \mathscr{D}$, let $G^{\mathrm{op}} : \mathscr{C}^{\mathrm{op}} \to \mathscr{D}^{\mathrm{op}}$ be the functor which is the same as G on objects and arrows. Show that, for any functors F and U, if F is left adjoint to U then F^{op} is right adjoint to U^{op}.

5.† This is a series of exercises designed to show that if the category $\mathscr{C}$ has finite limits, then the existence of a right adjoint R_A to the functor P_A of 13.2.6 is equivalent to the existence of an exponential object $[A \to -]$ in $\mathscr{C}$. This is a functor that satisfies the universal mapping property that $\mathrm{Hom}(A \times B, C) \cong \mathrm{Hom}(B, [A \to C])$ for all objects B and C of $\mathscr{C}$. (See Section 6.1.)

a. Let A be an object of $\mathscr{C}$, $P_A : \mathscr{C} \to \mathscr{C}/A$ the functor defined in 13.2.6 which takes an object C to $p_2 : C \times A \to A$, and $L_A : \mathscr{C}/A \to \mathscr{C}$ the underlying object functor which takes $f : B \to A$ to B. Show that L_A is left adjoint to P_A.

b. Show that if $P_A : \mathscr{C} \to \mathscr{C}/A$ has a *right* adjoint R_A, then the functor $A \times - : \mathscr{C} \to \mathscr{C}$ does as well.

c. Suppose that $\mathscr{C}$ is cartesian closed. Show that for each object of the form $P_A(C)$ in $\mathscr{C}/A$, the object $\Phi(P_A(C)) = [A \to C]$ in $\mathscr{C}$ has the basic adjunction property:

$$\mathrm{Hom}(B, \Phi(P_A(C))) \cong \mathrm{Hom}_A(P_A(B), P_A(C))$$

d. Show that every arrow $f : P_A(C) \to P_A(D)$ in $\mathscr{C}/A$ induces an arrow $\Phi(f) : \Phi(C) \to \Phi(D)$ such that

$$\begin{array}{ccc} \mathrm{Hom}(B, \Phi(C)) & \xrightarrow{\cong} & \mathrm{Hom}(P_A(B), P_A(C)) \\ \Big\downarrow{\scriptstyle \mathrm{Hom}(B, \Phi(f))} & & \Big\downarrow{\scriptstyle \mathrm{Hom}(B, f)} \\ \mathrm{Hom}(B, \Phi(D)) & \xrightarrow[\cong]{} & \mathrm{Hom}(P_A(B), P_A(D)) \end{array}$$

commutes. (Hint: Use the Yoneda Lemma.)

e. Show that if $f : C \to A$ is an arbitrary object of $\mathscr{C}/A$, then there is an equalizer diagram

$$f \xrightarrow{d} P_A(C) \underset{d^1}{\overset{d^0}{\rightrightarrows}} P_A L_A P_A(C)$$

(Hint: Use elements.)

f. Show that if we let, for an object $f : C \to A$ of $\mathscr{C}/A$,

$$\Phi(f) \to \Phi(P_A(C)) \underset{\Phi(d^1)}{\overset{\Phi(d^0)}{\rightrightarrows}} \Phi(P_A(P_A(C)))$$

be an equalizer, then for any object B of $\mathscr{C}$,

$$\mathrm{Hom}(B, \Phi(f)) \cong \mathrm{Hom}_A(P_A(B), f)$$

g. Show that P_A has a right adjoint.

13.4 Locally cartesian closed categories

A locally cartesian closed category is a special type of cartesian closed category which has properties desirable for modeling polymorphism. Its definition, which we give here, makes intrinsic use of adjunctions.

13.4.1 The pullback functor Let $\mathscr{C}$ be a category with pullbacks. Suppose $f : A \to B$ is an arrow. Then for any object $u : X \to A$ of the slice category $\mathscr{C}/A$, there is an object $f \circ u : \mathscr{C}/A \to \mathscr{C}/B$ that is the object part of a functor we denote $\Sigma_f : \mathscr{C}/A \to \mathscr{C}/B$. If $v : Y \to A$ is another object and $g : X \to Y$ an arrow, that is a map in $\mathscr{C}$ such that $v \circ g = u$, then $f \circ v \circ g = f \circ u$ so that g is also an arrow in $\mathscr{C}/B$. This defines Σ_f on arrows and it is immediate that it is a functor.

We now show, using Theorem 13.3.5, that Σ_f has a right adjoint $f^* : \mathscr{C}/B \to \mathscr{C}/A$. If $x : X \to B$ is an object of $\mathscr{C}/B$, $f^*(x) = p_1 : P \to A$, where

$$\begin{array}{ccc} P & \xrightarrow{p_2} & X \\ {\scriptstyle p_1}\downarrow & & \downarrow{\scriptstyle x} \\ A & \xrightarrow[f]{} & B \end{array} \tag{13.8}$$

is a pullback diagram.

To see what f^* does on arrows, suppose $f^*(x : X \to B) = p_1 : P \to A$ as above, and $f^*(x' : X' \to B) = p_1' : P' \to A$, the latter given by the pullback

$$\begin{array}{ccc} P' & \xrightarrow{p_2'} & X' \\ {\scriptstyle p_1'}\downarrow & & \downarrow{\scriptstyle x'} \\ A & \xrightarrow[f]{} & B \end{array} \tag{13.9}$$

Suppose $t : x \longrightarrow x'$ is an arrow of $\mathscr{C}/B$. Then

$$x' \circ t \circ p_2 = x \circ p_2 = f \circ p_1$$

because t is an arrow in $\mathscr{C}/B$ and Diagram (13.8) commutes. Thus P with $t \circ p_2$ and p_1 is a commutative cone over the pullback diagram (13.9), so induces an arrow $s : P \longrightarrow P'$ such that $p'_1 \circ s = p_1$ and $p'_2 \circ s = t \circ p_2$. The first equation says that s is an arrow from p_1 to p'_1 in $\mathscr{C}/A$, and so $f^*(t) = s$. This shows what f^* must be on arrows. Then Theorem 13.3.5 puts it all together into a functor.

13.4.2 Proposition *Let $\mathscr{C}$ be a category with pullbacks and $f : A \longrightarrow B$ an arrow of $\mathscr{C}$. Then Σ_f as defined above is left adjoint to f^*.*

The proof is omitted. The construction in 13.2.6 is a special case of this construction, at least in the case where $\mathscr{C}$ has a terminal object. To see this, let f be the unique arrow from A to 1 and note that the pullback becomes a product and $\mathscr{C}/1$ is isomorphic to $\mathscr{C}$.

The pullback functor need not have a *right* adjoint, in fact that is a rather rare occurrence. The following theorem characterizes those categories in which it does have a right adjoint.

13.4.3 Theorem *The following are equivalent for any category $\mathscr{C}$.*

(a) *$\mathscr{C}$ has pullbacks and for each arrow $f : A \longrightarrow B$, the pullback functor $f^* : \mathscr{C}/B \longrightarrow \mathscr{C}/A$ has a right adjoint.*

(b) *For every object A of $\mathscr{C}$, the slice category $\mathscr{C}/A$ is cartesian closed.*

The right adjoint to the pullback functor f^* is denoted Π_f ($\forall_f$ in some texts).

We sketch the proof. The proof hinges on the following proposition, whose easy proof we omit.

13.4.4 Proposition *For any arrows $u : X \longrightarrow A$ and $v : Y \longrightarrow A$ in any category, the arrow from P to A given by a pullback diagram*

$$\begin{array}{ccc} P & \xrightarrow{p_2} & X \\ {\scriptstyle p_1}\downarrow & & \downarrow{\scriptstyle u} \\ Y & \xrightarrow[v]{} & A \end{array} \tag{13.10}$$

is the product of u and v in the slice category $\mathscr{C}/A$.

Now we give the proof of Theorem 13.4.3. Suppose the conditions in (a) hold. Let us consider arrows $u : X \longrightarrow A$, $v : Y \longrightarrow A$ and $w : Z \longrightarrow A$ of $\mathscr{C}/A$. It is clear from Proposition 13.4.4 that, in $\mathscr{C}/A$,

$$u \times v = u \circ p_2 = \Sigma_u(u^*(v))$$

By hom set adjointness (Theorem 13.3.2), we then have

$$\begin{aligned} \mathrm{Hom}_{\mathscr{C}/A}(u \times v, w) &= \mathrm{Hom}_{\mathscr{C}/A}(\Sigma_u(u^*(v)), w) \\ &\cong \mathrm{Hom}_{\mathscr{C}/X}(u^*(v), u^*(w)) \\ &\cong \mathrm{Hom}_{\mathscr{C}/A}(v, \Pi_u(u^*(w))) \end{aligned}$$

so that defining $[u \longrightarrow w]$ to be $\Pi_u(u^*(w))$ gives the cartesian closed structure on $\mathscr{C}/A$.

For the converse, let $\mathscr{C}$ denote a category with the property that every slice is a cartesian closed category. Since any slice of a slice of the form $(\mathscr{C}/A)/f$, for $f : B \longrightarrow A$ is isomorphic to the slice $\mathscr{C}/B$ (Exercise 12 of Section 3.3), we know that every slice of $\mathscr{C}$ also has the property that every slice is a cartesian closed category.

We now show that every slice has finite limits. Since every slice has products, $\mathscr{C}$ has pullbacks by Proposition 13.4.4. It follows from the remark in the preceding paragraph that every slice has pullbacks as well. Since also every slice has a terminal object ($\mathrm{id} : A \longrightarrow A$ is automatically terminal in $\mathscr{C}/A$), it follows from Proposition 9.3.7 of Section 9.3 that every slice has finite limits.

It now follows from Exercise 5 of Section 13.3 that for any $f : A \longrightarrow B$ in $\mathscr{C}$, the induced $f^* : \mathscr{C}/B \longrightarrow \mathscr{C}/A$ has a right adjoint.

13.4.5 Definition A category $\mathscr{C}$ which satisfies (a) (or equivalently (b)) of Theorem 13.4.3 is a **locally cartesian closed category**.

If a category $\mathscr{C}$ has a terminal object 1, $\mathscr{C}$ is isomorphic as a category to $\mathscr{C}/1$. We then have the following proposition.

13.4.6 Proposition *A locally cartesian category which has a terminal object is cartesian closed.*

Applications are discussed in [Seely, 1984] and [Seely, 1987a].

14
Algebras for endofunctors

In this chapter, we describe some constructions based on an **endofunctor** of a category $\mathscr{C}$, which is a functor from $\mathscr{C}$ to $\mathscr{C}$.

We begin in Section 14.1 to study the concept of fixed point and least fixed point of an endofunctor. A natural structure to be a fixed point is what is called an algebra for the endofunctor, which is defined there. These algebras are suitable for developing a concept of list object in a category (Section 14.2) and a categorical version of state transition machines (14.2.10).

A triple is a structure which abstracts the idea of adjunction; part of the structure is an endofunctor. Triples have turned out to be an important technical tool in category theory. Section 14.3 defines triples and gives some of their basic properties. In Section 14.4 we develop the idea of an algebra for a triple; it is an algebra for the endofunctor part of the triple with certain properties. The last section describes a technique of Smyth and Plotkin to construct Scott domains, which provide models of computations in cartesian closed categories. These models allow modeling recursion by fixed point techniques, and the models are themselves constructed as fixed points. This last section requires only Section 14.1 to be read.

The material in this chapter is not needed for the rest of the book, except for some examples.

14.1 Fixed points for a functor

14.1.1 What are fixed points for a functor? Let $R : \mathscr{A} \to \mathscr{A}$ be a functor. In this section, we will analyze the notions of fixed point and of least fixed point for a functor.

The question of fixed points arises only with an endofunctor. We must understand what sort of structure a fixed point is and what it means to be a least fixed point. For a function on a set, say, we know exactly what a fixed point is. If the set is a partial order, we know exactly what a least fixed point is, if any. The relevant structure for fixed points of functors turns out to be the concept of algebra for the functor.

14.1.2 To see why a fixed point for a functor does not have the obvious definition, consider a possible definition of the natural numbers as the least

fixed point of the functor R on **Set** defined by $R(S) = 1 + S$ for a set S, and for a function $f : S \to T$, $R(f) = \mathrm{id}_1 + f : 1 + S \to 1 + T$. Here, 1 is a terminal object and '+' denotes disjoint union.

Even supposing that we have chosen a functorial definition for $+$, it is not clear that any set is fixed by this functor. If one is, it is not obvious that it will look like the natural numbers. Moreover, this fixed point will depend on the surely irrelevant way in which the sums are defined.

At first it might seem that what is really wanted is a set S which is *isomorphic to* $1 + S$. This is certainly true for $S = \mathbf{N}$; one isomorphism from $1 + \mathbf{N}$ to $\mathbf{N}$ takes the unique element of the terminal object to 0 and each element $n \in \mathbf{N}$ to $n + 1$. This approach is consonant with the categorical perception that any use you could make of an object can equally well be made of any isomorphic copy. Thus a fixed point for a functor need only be fixed *up to isomorphism*, thereby avoiding any dependence on arbitrary choices, for example of which particular categorical sum is used.

This leads to a second problem. Although $\mathbf{N}$, and indeed any infinite set, is now a fixed point, none is *least* in any obvious way. For example, any infinite set has a proper infinite subset, which is also fixed in the same way. To understand what is going on here, we approach the question from another angle.

14.1.3 Algebras for an endofunctor As above, we let $\mathscr{A}$ be a category and $R : \mathscr{A} \to \mathscr{A}$ be an endofunctor. An **R-algebra** is a pair (A, a) where $a : RA \to A$ is an arrow of $\mathscr{A}$. A homomorphism between R-algebras (A, a) and (B, b) is an arrow $f : A \to B$ of $\mathscr{A}$ such that

$$\begin{array}{ccc} RA & \xrightarrow{a} & A \\ {\scriptstyle Rf}\downarrow & & \downarrow{\scriptstyle f} \\ RB & \xrightarrow[b]{} & B \end{array}$$

commutes. This construction gives a category $(R : \mathscr{A})$ of R-algebras.

14.1.4 Definition An object (A, a) of $(R : \mathscr{A})$ for which the arrow a is an isomorphism is a **fixed point** for R.

Based on the perception that a category is a generalized poset, the following definition is reasonable.

14.1.5 Definition A **least fixed point** of a functor $R : \mathscr{A} \to \mathscr{A}$ is an initial object of $(R : \mathscr{A})$.

14.1.6 Example In the case of the CPO $\mathscr{P}$ defined in 2.4.8, the function ϕ induces an endofunctor on the category $C(\mathscr{P})$ corresponding to $\mathscr{P}$. One can prove by induction that the only algebra for the functor ϕ is the factorial function itself, which is both A and $\phi(A)$. The algebra map a is the identity map (the factorial function is less than or equal to itself in the CPO $\mathscr{P}$). Thus this algebra is clearly the initial (because it is the only) object of the category of algebras.

If Definition 14.1.5 is to work, we have to show that an initial object of $(R : \mathscr{A})$ is indeed a fixed point.

14.1.7 Theorem [Lambek, 1970] *Let $R : \mathscr{A} \longrightarrow \mathscr{A}$ be a functor from a category to itself. If (A, a) is initial in the category $(R : \mathscr{A})$, then a is an isomorphism.*

Proof. Suppose (A, a) is initial. (RA, Ra) is an object of the category $(R : \mathscr{A})$ and hence there is a unique arrow $f : (A, a) \longrightarrow (RA, Ra)$. This means that the top square of the diagram

$$\begin{array}{ccc} RA & \xrightarrow{a} & A \\ {\scriptstyle Rf}\downarrow & & \downarrow{\scriptstyle f} \\ R^2A & \xrightarrow{Ra} & RA \\ {\scriptstyle Ra}\downarrow & & \downarrow{\scriptstyle a} \\ RA & \xrightarrow[a]{} & A \end{array}$$

commutes, while the bottom square patently does. Thus the whole rectangle does and so $a \circ f : (A, a) \longrightarrow (A, a)$ is an arrow between R-algebras. But (A, a) is initial and so has only the identity endomorphism. Thus $a \circ f = \mathrm{id}$. Then the commutativity of the upper square gives us that

$$f \circ a = Ra \circ Rf = R(a \circ f) = R(\mathrm{id}) = \mathrm{id}$$

which means that $f = a^{-1}$ and that a is an isomorphism. □

14.1.8 Example Now let us look again at the functor $R : \mathbf{Set} \longrightarrow \mathbf{Set}$ which takes S to $1 + S$. The natural numbers $\mathbf{N}$ form an algebra for R, whose R-algebra structure is the function $(0; s) : 1 + \mathbf{N} \longrightarrow \mathbf{N}$, where 0 is the function picking out 0 and s is the successor function.

This is in fact an initial algebra for R. For suppose $f : 1 + S \longrightarrow S$ is an R-algebra. The required unique R-algebra homomorphism $h : \mathbf{N} \longrightarrow S$ is the function defined inductively by $h(0) = f(*)$ ($*$ is the unique element of the singleton 1) and $h(n + 1) = f(h(n))$ (Exercise 1).

Although **N** has many subsets which are fixed up to isomorphism for R, it has no proper subset that is fixed under the given isomorphism $(0; s)$; that is one sense in which it is the 'least' fixed point for R. Now a proper subset fixed under the isomorphism is the same thing as a proper subobject in the category $(R : \mathscr{A})$, and no initial object of any category has a proper subobject (Exercise 6 of Section 2.8); thus least fixed points of functors all have the property that they have no proper subobjects.

However, initiality is strictly stronger than the property of not having proper subobjects. For example, the only object of the category corresponding to the two-element monoid whose nonidentity element m satisfies $m^2 = m$ is an object with no proper subobjects that is not initial. Thus being the least fixed point of a functor in the sense we have defined is a strong requirement.

Another property of initial objects is that an initial object is determined uniquely up to a unique isomorphism; thus least fixed points in our sense have the desirable property of being uniquely determined in the strongest possible sense of uniqueness consistent with the philosophy of category theory: any two least fixed points are isomorphic in a unique way.

14.1.9 Fixed points of finitary functors Suppose we have a functor R on the category of sets. We say that R is **finitary** if

FF–1 For each set S and each element $x \in RS$, there is a finite subset $S_0 \subseteq S$ and an element $x_0 \in RS_0$ such that $x = Ri_0(x_0)$ where $i_0 : S_0 \longrightarrow S$ is the inclusion.

FF–2 If also S_1 is a finite subset of S with inclusion i_1 and $x_1 \in RS_1$ with $x = Ri_1(x_1)$, then there is a finite subset S_2 containing both S_0 and S_1 with inclusions $j_0 : S_0 \longrightarrow S_2$ and $j_1 : S_1 \longrightarrow S_2$ such that $Rj_0(x_0) = Rj_1(x_1)$.

The real meaning of finitary is that everything is determined by what happens on finite subsets. From the point of view of computer science, this seems quite a reasonable hypothesis.

In fact, in the category of sets, FF–2 is unnecessary. We include it anyway as a condition of this type is needed if this construction is generalized beyond the category of sets. See Exercise 3 below.

14.1.10 Example The functor $R : \mathbf{Set} \longrightarrow \mathbf{Set}$ defined in 14.1.2 is finitary. Suppose $x \in 1 + S$. If $x \in 1$ we can take $S_0 = \emptyset$ with inclusion $i_0 : \emptyset \longrightarrow 1 + \emptyset$. Then the unique element of the terminal object 1 is taken to itself by Ri_0, which is id_1. If $x \in S$ we can take $S_0 = \{x\}$ since $Ri_0 = i_0$ when restricted to the component S_0 of $1 + S_0$.

A finitary functor always has a fixed point (Theorem 14.1.11 below). In fact, the underlying functor from $(R : \mathbf{Set})$ to $\mathbf{Set}$ has an adjoint; see [Barr, 1971] for details and generalizations.

We denote by η the unique arrow $\emptyset \to R\emptyset$. We have a sequence

$$\emptyset \xrightarrow{\eta} R(\emptyset) \xrightarrow{R(\eta)} R^2(\emptyset) \xrightarrow{R^2(\eta)} \cdots \xrightarrow{R^{n-1}(\eta)} R^n(\emptyset) \xrightarrow{R^n(\eta)} \cdots \tag{14.1}$$

The colimit of this sequence (which would be the union if the arrows were inclusions) is a set we will call Z.

14.1.11 Theorem *If R is a finitary endofunctor on* **Set** *and Z is the colimit of (14.1), then there is an R-algebra structure $z : RZ \to Z$ which is an initial R-algebra.*

The crucial point is that when R is finitary, it commutes with the colimit that defines Z so that RZ is the colimit of the sequence

$$R(\emptyset) \to R^2(\emptyset) \to R^3(\emptyset) \cdots$$

which is canonically isomorphic to Z. The structure map $z : RZ \to Z$ is this isomorphism. The proof that this works is fairly technical and is omitted.

14.1.12 Example Let R be the functor of 14.1.2. Then the sequence (14.1) comes down to

$$0 \to 1 \to 2 \to \cdots \to n \to \cdots$$

where by n we mean the sum $1 + 1 + \cdots + 1$ (n summands). The arrow from n to $n + 1$ is given by the injection into the sum. This gives rise to the set $\mathbf{N}$ of natural numbers with successor as the operation.

14.1.13 Example Here is an example of a nonfinitary functor. Let $P : \mathbf{Set} \to \mathbf{Set}$ take a set S to the set of subsets of S. If $f : S \to T$ is a function, then Pf takes the subset $S_0 \subseteq S$ to $f(S_0) \subseteq T$. P is not finitary because when S is infinite, not every subset of S (in fact, no infinite subset of S) is a subset of a finite subset of S.

14.1.14 The reader may wonder why it was necessary to introduce the category of all R-algebras when we were interested only in the fixed points. There are two answers to that question.

First, if $\mathscr{A}$ is complete with an initial object, then $(R : \mathscr{A})$ is complete as well and the underlying functor $U : (R : \mathscr{A}) \to \mathscr{A}$ preserves all limits. This does not actually *prove* that U has an adjoint, but it makes it highly likely.

The second reason is that the construction of $(R : \mathscr{A})$ is part of the construction of the category of algebras for a triple, which is carried out in 14.4.2.

14.1.15 Exercises

1. Show that the function h defined in 14.1.8 is the unique R-algebra homomorphism from $(0; s) : 1 + \mathbf{N} \to \mathbf{N}$ to $f : 1 + S \to S$.

2. Let $(S, \leq)$ be a total order and $f : S \to S$ a monotone function. Let $C(S, \leq)$ be the category determined by $(S, \leq)$ as in 2.3.1. Show that f determines an endofunctor, the objects of $(f : S)$ are the $x \in X$ for which $f(x) \leq x$ and the least fixed point is the same as the one defined here.

3. a. Show that whenever $f : S \to T$ is an injective function in the category of sets and $S \neq \emptyset$, then there is a $g : T \to S$ such that $g \circ f = \mathrm{id}_S$.

b. Show that if $F : \mathbf{Set} \to \mathscr{C}$ is a functor to an arbitrary category and $f : S \to T$ is an injective function in **Set**, then $F(f)$ is a monomorphism.

c. Show that condition FF–2 of 14.1.9 is unnecessary.

14.2 Recursive categories

In this section, we discuss two ways involving algebras for functors in which recursion can be modeled in a category.

14.2.1 Recursion in functional programming languages In 2.2.1, we described how to represent a functional programming language as a category. The major missing piece from that description was how to produce potentially infinite programs. This is done in traditional languages using `while` loops and their relatives, but can also be done in a flexible way using recursion, which is available in cartesian closed categories with a natural numbers object (see Section 5.5). However, the construction of function space objects in cartesian closed categories can lead to noncomputable constructions.

J. R. B. Cockett has proposed axioms on a category which directly allow a limited form of recursion, which in the presence of mild additional hypotheses preserves a form of computability (Theorem 14.2.9). The discussion that follows is based on [Cockett, 1989], [Cockett, 1990].

14.2.2 Let $\mathscr{C}$ be a category with finite products. Then for every object A there is a functor $A \times -$ which takes an object X to $A \times X$ and an arrow $f : X \to Y$ to $\mathrm{id}_A \times f : A \times X \to A \times Y$. An algebra $x : A \times X \to X$ for this functor is an **A-action**. For each object A there is a category $\mathrm{act}(A)$ which is the category of algebras for the functor $A \times -$. There is an underlying functor $U_A : \mathrm{act}(A) \to \mathscr{C}$ which takes $x : A \times X \to X$ to X and an arrow $f : (x : A \times X \to X) \to (y : A \times Y \to Y)$ to f.

14.2.3 Definition A category $\mathscr{C}$ is **recursive** if for every object A the underlying functor U_A has a left adjoint $F_A : \mathscr{C} \longrightarrow \mathrm{act}(A)$.

For an object B, denote $U_A(F_A(B))$ by $\mathrm{rec}(A, B)$; for $f : B \longrightarrow C$, we get an arrow

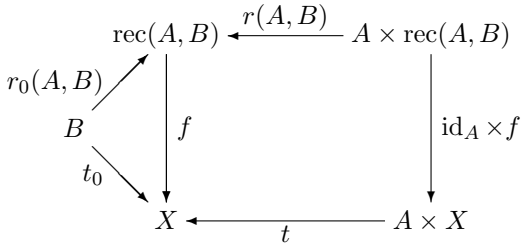
$$\mathrm{rec}(A, f) = U_A(F_A(f)) : \mathrm{rec}(A, B) \longrightarrow \mathrm{rec}(A, C)$$

The algebra $F_A(B)$ is an algebra structure

$$r(A, B) : A \times \mathrm{rec}(A, B) \longrightarrow \mathrm{rec}(A, B)$$

The unit of the adjunction has component $r_0(A, B) : B \longrightarrow \mathrm{rec}(A, B)$ at an object B. In particular, $\mathrm{rec}(1, 1)$ is a natural numbers object (Exercise 2).

The object $\mathrm{rec}(A, B)$ is characterized by the following universal mapping property. If $t_0 : B \longrightarrow X$ and $t : A \times X \longrightarrow X$ are arbitrary arrows, then there is a unique $f : \mathrm{rec}(A, B) \longrightarrow X$ such that the following diagram commutes.

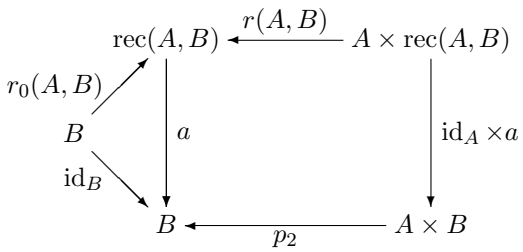


14.2.4 Set is recursive. You can check that in **Set**, $\mathrm{rec}(A, B)$ is (up to isomorphism) the set of pairs of the form (l, b) where l is a list of elements of A (including the empty list) and $b \in B$, and $r(A, B)$ is the function which takes $(a, (l, b))$ to $(\mathrm{cons}(a, l), b)$, where cons is the function which adjoins a to the list l at the front. The unit $r_0(A, B) : B \longrightarrow \mathrm{rec}(A, B)$ takes b to $(\langle\rangle, b)$.

In particular, in **Set**, $A^* = \mathrm{rec}(A, 1)$ is the set of lists of elements of A. It has the right properties to be regarded as lists of A in any category; for example, Cockett [1990] shows how to define the head and tail of a list in recursive categories. Because of this, it is natural to denote $\mathrm{rec}(A, 1)$ by A^* in a recursive category.

14.2.5 Now for objects A and B of a recursive category $\mathscr{C}$, the projection $p_2 : A \times B \longrightarrow B$ is an algebra for $A \times -$. We have the identity arrow of B and the universal mapping property then produces a unique arrow $a : \mathrm{rec}(A, B)$

$\longrightarrow B$ with the property that



commutes. There is then an arrow

$$c(A,B) = \langle \mathrm{rec}(A, \langle\rangle), a\rangle : \mathrm{rec}(A,B) \longrightarrow \mathrm{rec}(A,1) \times B$$

14.2.6 Definition Let $\mathscr{C}$ be a recursive category. If for all A and B, $c(A,B)$ is an isomorphism, then $\mathscr{C}$ has **local recursion**. A **locally recursive category** is a recursive category with local recursion for which every slice category $\mathscr{C}/C$ is recursive. $\mathscr{C}$ is a **locos** if it is coherent (see 9.6.8) and locally recursive.

The definition of locally recursive implies that $\mathrm{rec}(A,B) \cong \mathrm{rec}(A,1) \times B = A^* \times B$. The discussion in 14.2.4 shows that this is true in **Set**.

The axioms on a locos allow many other constructions. Specifically, there is a whole class of functors for which the underlying functor from the algebra category has a left adjoint.

14.2.7 Definition The class of **polynomial endofunctors** on a category with finite sums and products is the least class that contains the constant endofunctors and the identity functor and is closed under the operations of finite products and sums.

14.2.8 Proposition *Let $\mathscr{C}$ be a locos and P a polynomial functor. Then the underlying functor $U : (P : \mathscr{C}) \longrightarrow \mathscr{C}$ has a left adjoint.*

This allows the construction of free objects generated by an object. Thus if D is an object and T is the functor for which

$$T(X) = D + X \times D \times X$$

the free object on the terminal object 1 has a structure map

$$[D + F(1) \times D \times F(1)] \longrightarrow F(1)$$

It can be interpreted as the object of binary trees with data at the root, and the structure map constructs such trees from either a datum or a datum and two given trees. Thus a locos allows a type of initial algebra semantics.

Another property of locoses is that if you start with computable things the constructions of a locos give you computable things in a precise sense. This requires a definition. In any category with finite products, an object D is **decidable** if the diagonal $\Delta : D \longrightarrow D \times D$ (which is a subobject of $D \times D$) has a complement in $D \times D$ (see 9.6.2). Thus the law of the excluded middle applies internally to equality of elements of a decidable object: two elements are either equal (factor through the diagonal) or not equal (factor through the complement of the diagonal).

In a category representing computable objects, one can assume that each object can be generated effectively: in other words, each object is recursively enumerable. Of course, the diagonal is then recursively enumerable – one generates elements of D and for each element forms (d, d) (the word 'element' can be taken in various senses here). If the object is decidable, then the complement of the diagonal exists, so that one can generate all the pairs of unequal elements. Then by a well known theorem one can decide whether two elements are the same: hence the name 'decidable'.

14.2.9 Theorem *If $\mathscr{S}$ is a collection of decidable objects in a locos, and no sublocos of the locos contains $\mathscr{S}$, then every object of the locos is decidable.*

For the proof, see [Cockett, 1989]. Locoses may be described as models of an FL sketch, so that there is an initial locos. The theorem above implies that in the initial locos every object is decidable.

14.2.10 Categorical dynamics An **input process** in a category $\mathscr{C}$ is an endofunctor R for which the underlying functor $(R : \mathscr{C}) \longrightarrow \mathscr{C}$ has a left adjoint. An object of the category $(R : \mathscr{C})$ is called an **R-dynamic**, the category itself is called the category of **R-dynamics** and an arrow in the category is called a **dynamorphism**. A **machine in** $\mathscr{C}$ is a 7-tuple $M = (R, C, c, I, \tau, Y, \beta)$ in which R is an input process, (C, c) is an R-dynamic, the object I of $\mathscr{C}$ is called the initial state object (not to be confused with an initial object of the category) and $\tau : I \longrightarrow C$ the initial state arrow, and Y is called the output object and $\beta : C \longrightarrow Y$ is the output arrow.

14.2.11 Example Let A be an object of any recursive category. Then by definition the functor R defined by $R(C) = A \times C$ and $R(f) = \mathrm{id}_A \times f$ is an input process. Thus finite state machines can be defined in an arbitrary recursive category. Many definitions connected with finite state machines can be given in a recursive category. For example, a machine $M = (R, C, c, I, \tau, Y, \beta)$ is reachable if τ factors through no proper subobject of the algebra (C, c).

In particular, let the recursive category be the category of sets and let A be a finite set thought of as an alphabet. An algebra for R is a function $\delta : A \times C \longrightarrow C$ for some set C. By defining $\delta^*(\langle\rangle, q) = q$ and

$\delta^*(aw, q) = \delta(A, \delta^*(w, q))$, one has an action of the free monoid A^* on C. Setting I to be a one-element set, one obtains the usual definition of state transition machine used in the automata theory literature. Reachability here has its usual meaning, that one can reach any state from the start state by a sequence of operations (if not, the states that *can* be so reached form a proper subalgebra containing the image of τ).

The concept of R-dynamic is much more general than this. For example, one can have several 'start' states by choosing I to be a bigger set. More drastically, one could use a different functor R, for example $R(C) = C \times C$.

This discussion of categorical dynamics is from [Arbib and Manes, 1975], which should be consulted for details. See also [Arbib and Manes, 1980].

14.2.12 Exercises

1. Prove the claims in 14.2.4.

2. Prove that in a recursive category, $\mathrm{rec}(1, 1)$ is a natural numbers object. (Hint: Use the unit of the adjunction for 0.)

14.3 Triples

We now describe a structure based on an endofunctor which has turned out to be an important technical tool in studying toposes and related topics. It is an abstraction of the concept of adjoint and in a sense an abstraction of universal algebra (see the remarks in fine print at the end of 14.4.3 below).

14.3.1 Definition A **triple** $\mathbf{T} = (T, \eta, \mu)$ on a category $\mathscr{A}$ consists of a functor $T : \mathscr{A} \to \mathscr{A}$, together with two natural transformations $\eta : \mathrm{id} \to T$ and $\mu : T^2 \to T$ for which the following diagrams commute.

$$\begin{array}{ccccc} T & \xrightarrow{\eta T} & T^2 & \xleftarrow{T\eta} & T \\ & \searrow^{=} & \downarrow{\scriptstyle \mu} & \swarrow_{=} & \\ & & T & & \end{array} \qquad \begin{array}{ccc} T^3 & \xrightarrow{T\mu} & T^2 \\ {\scriptstyle \mu T}\downarrow & & \downarrow{\scriptstyle \mu} \\ T^2 & \xrightarrow[\mu]{} & T \end{array} \tag{14.2}$$

Here, ηT and $T\eta$ are defined as in 4.4.2 and 4.4.3.

The transformation η is the **unit** of the triple and μ is the **multiplication**. The left diagram constitutes the (left and right) **unitary identities** and the right one the **associative identity**. The reason for these names comes from the analogy between triples and monoids. This will be made clear in 14.3.4.

Another widely used name for triple is 'monad'. However, they have nothing to do with the monads of Robinson's theory of infinitesimals.

14.3.2 The triple arising from an adjoint pair An adjoint pair gives rise to a triple on the domain of the left adjoint.

14.3.3 Proposition *Let $U : \mathscr{B} \longrightarrow \mathscr{A}$ and $F : \mathscr{A} \longrightarrow \mathscr{B}$ be functors such that $F \dashv U$ with $\eta : \mathrm{id} \longrightarrow UF$ and $\epsilon : FU \longrightarrow \mathrm{id}$ the unit and counit, respectively. Then $(UF, \eta, U\epsilon F)$ is a triple on $\mathscr{A}$.*

We leave the proof as an exercise. Note that $U\epsilon F : UFUF \longrightarrow UF$, as required for the multiplication of a triple with functor UF.

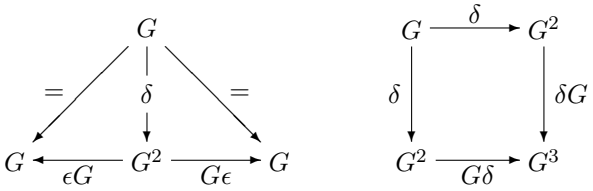
Conversely, every triple arises in that way out of some (generally many) adjoint pair. See Section 14.4 for two ways of constructing such adjoints.

14.3.4 Representation triples Let M be a monoid. The representation triple $\mathbf{T} = (T, \eta, \mu)$ on the category of sets is given by letting $T(S) = M \times S$ for a set S. $\eta S : S \longrightarrow T(S) = M \times S$ takes an element $s \in S$ to the pair $(1, s)$. We define μS by $(\mu S)(m_1, m_2, s) = (m_1 m_2, s)$ for $s \in S,\ m_1, m_2 \in M$. Thus the unit and multiplication of the triple arise directly from that of the monoid. The unitary and associativity equations can easily be shown to follow from the corresponding equations for the monoid.

The standard way of getting this triple from an adjoint pair is by using the underlying and free functors on M-sets (see 3.2.1). If S and T are M-sets, then a function $f : S \longrightarrow T$ is said to be **M-equivariant** if $f(ms) = mf(s)$ for $m \in M,\ s \in S$. For a fixed monoid M, the M-sets and the M-equivariant functions form a category, called the category of M-sets.

The **free M-set** generated by the set S is the set $M \times S$ with action given by $m'(m, s) = (m'm, s)$. Using Theorem 13.3.5, one can show immediately that this determines a functor left adjoint to the underlying set functor on the category of M-sets. The triple associated to this adjoint pair is the one described above.

14.3.5 Cotriples A **cotriple $\mathbf{G} = (G, \epsilon, \delta)$** in a category $\mathscr{A}$ is a triple in $\mathscr{A}^{\mathrm{op}}$. Thus G is an endofunctor of $\mathscr{A}$ and $\epsilon : G \longrightarrow \mathrm{id}$ and $\delta : G \longrightarrow G^2$ are natural transformations such that



(14.3)

Cotriples are used in Chapter 16.

14.3.6 Exercises

1. Let $(P \leq)$ be a poset and $\mathbf{T} = (T, \eta, \mu)$ a triple on the category $C(P, \leq)$. Show that for any $x \in \mathbf{P}$, $x \leq T(x)$ and $T(T(x)) = T(x)$. (Such a function T is called a **closure operator**.)

2. Prove Proposition 14.3.3.

3. Let $T : \mathbf{Set} \longrightarrow \mathbf{Set}$ be the functor which takes a set A to the Kleene closure A^* and a function $f : A \longrightarrow B$ to the function $f^* : A^* \longrightarrow B^*$ defined in Section 2.5.7. Let $\eta A : A \longrightarrow A^*$ take an element a to the one-element string (a), and let $\mu A : A^{**} \longrightarrow A^*$ take a string $(s_1, s_2, \ldots, s_k)$ of strings to the concatenated string $s_1 s_2 \cdots s_n$ in A^* obtained in effect by erasing inner brackets: thus $((a, b), (c, d, e), (), (a, a))$ goes to

$$(a, b)(c, d, e)()(a, a) = (a, b, c, d, e, a, a)$$

In particular, $\mu A((a, b)) = (a, b)$. Show that $\eta : \mathrm{id} \longrightarrow T$ and $\mu : T \circ T \longrightarrow T$ are natural transformations, and that (T, η, μ) is a triple.

14.4 Factorizations of a triple

14.4.1 The Kleisli category for a triple Let $\mathbf{T} = (T, \eta, \mu)$ be a triple on $\mathscr{C}$. We describe here a construction which exhibits the triple as coming from an adjoint. This construction, which is due to Kleisli [1965], has proven to be quite useful in theoretical computer science.

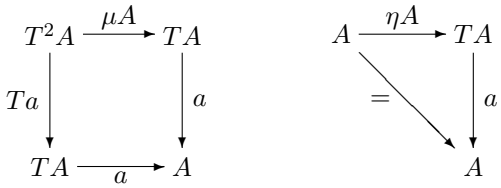
We define a category $\mathscr{K} = \mathscr{K}(\mathbf{T})$ which has the same objects as $\mathscr{C}$. If A and B are objects of $\mathscr{C}$, then an arrow in $\mathscr{K}$ from A to B is an arrow $A \longrightarrow TB$ in $\mathscr{C}$. The composition of arrows is as follows. If $f : A \longrightarrow TB$ and $g : B \longrightarrow TC$ are arrows in $\mathscr{C}$, we let their composite in $\mathscr{K}$ be the following composite in $\mathscr{C}$:

$$A \xrightarrow{f} TB \xrightarrow{Tg} T^2C \xrightarrow{\mu C} TC$$

The identity of the object A is the arrow $\eta A : A \longrightarrow TA$. It can be shown that this defines a category. Moreover, there are functors $U : \mathscr{K}(\mathbf{T}) \longrightarrow \mathscr{C}$ and $F : \mathscr{C} \longrightarrow \mathscr{K}(\mathbf{T})$ defined by $UA = TA$ and $Uf = \mu B \circ Tf$, where B is the codomain of f, and $FA = A$ and for $g : A \longrightarrow B$, $Fg = Tg \circ \eta A$. Then F is left adjoint to U and $T = U \circ F$. The proof is left as an exercise.

14.4.2 Eilenberg–Moore algebras Here is a second way, due to Eilenberg and Moore [1965] of factoring every triple as an adjoint pair of functors. In mathematics, this construction has been much more interesting than the Kleisli construction, but in computer science it has been quite the opposite.

Let $\mathbf{T} = (T, \eta, \mu)$ be a triple on $\mathscr{A}$. A T-algebra (A, a) is called a $\mathbf{T}$-algebra if the following two diagrams commute:



An arrow (homomorphism) between $\mathbf{T}$-algebras is the same as an arrow between the corresponding T-algebras. With these definitions, the $\mathbf{T}$-algebras form a category traditionally denoted $\mathscr{A}^{\mathbf{T}}$ and called the category of $\mathbf{T}$-algebras.

There is an obvious underlying functor $U : \mathscr{A}^{\mathbf{T}} \to \mathscr{A}$ with $U(A, a) = A$ and $Uf = f$. This latter makes sense because an arrow of $\mathscr{A}^{\mathbf{T}}$ *is* an arrow of $\mathscr{A}$ with special properties. There is also a functor $F : \mathscr{A} \to \mathscr{A}^{\mathbf{T}}$ given by $FA = (TA, \mu A)$ and $Ff = Tf$. Some details have to be checked; these are included in the following.

14.4.3 Proposition *The function F above is a functor left adjoint to U. The triple associated to the adjoint pair $F \dashv U$ is precisely $\mathbf{T}$.*

The proof is left as an exercise.

By a theorem of Linton's, every equationally defined category of one-sorted algebraic structures is in fact equivalent to the category of Eilenberg–Moore algebras for some triple in **Set** ([Barr and Wells, 1985], Theorem 5 of Section 4.3). (See Exercise 3.) In fact, the converse is true if infinitary operations are allowed (but then a hypothesis has to be added on the direct part of the theorem that there is only a set of operations of any given arity).

14.4.4 The Kleisli category and free algebras The Kleisli category of a triple $\mathbf{T} = (T, \eta, \mu)$ is equivalent to the full subcategory of free $\mathbf{T}$-algebras. Its definition makes it clear that the arrows are substitutions.

As an example, consider the list triple of Exercise 3, Section 14.3. An arrow $f : A \to B$ (here A and B are sets) of the Kleisli category is a set function $A \to TB$, so that it associates a string of elements of B to each element of A. Suppose $A = \{a, b\}$ and $B = \{c, d, e\}$, and that $f(a) = cddc$ and $f(b) = ec$. Then $Tf : TA \to TTB$ takes, for example, the string $abba$ to $(cddc)(ec)(ec)(cddc)$, the result of substituting $cddc$ for a and ec for b in $abba$. Then μ takes that string to $cddcececcddc$. It is instructive in this situation to think of μ as *carrying out a computation*. In this case the computation is carried out using the (only) monoid operation, since in fact the algebras for this triple are monoids (Exercise 3). Thus one can think of the objects

of the Kleisli category as *computations*. This is more compelling if one uses a triple arising from algebraic structures such as rings that abstract some of the properties of numerical addition and multiplication; then the objects of the free algebra are polynomial expressions and μ evaluates the polynomial.

An important idea for developing this point of view is the notion of 'strong monad' [Kock, 1972], which has been developed by Moggi [1989, 1991b] and others [Cockett and Spencer, 1992], [Mulry, 1992]. Other applications of triples (monads) in computing science can be found in the survey [Rydeheard and Burstall, 1985], as well as [Moggi, 1991a], [Power, 1990b], [Wadler, 1989], [Lüth and Ghani, 1997] and [Wadler, 1992] (the latter has many references to the literature).

14.4.5 Exercises

1. Show that $\eta A : A \longrightarrow TA$ is the identity on A in $\mathscr{K}(\mathbf{T})$.

2. Show that the composition defined for $\mathscr{K}(\mathbf{T})$ is associative.

3.† Let (T, η, u) be the triple in **Set** defined in Exercise 3 of Section 14.3.

a. Show that an algebra for this triple is a monoid: specifically, if $\alpha : T(A) \longrightarrow A$ is an algebra, then the definition $ab = \alpha(a, b)$ makes A a monoid, and up to isomorphisms every monoid arises this way.

b. Show that algebra homomorphisms are monoid homomorphisms, and that every monoid homomorphism arises this way.

4.† Prove Proposition 14.4.3.

14.5 Scott domains

In [Scott, 1972], a general construction of a great many models of the untyped λ-calculus is given. (See also [Scott, 1982].) One version of finding such a model involves finding an object D in a cartesian closed category with the property that $D \cong [D \longrightarrow D]$. This is an example of finding a model for computation in a category other than the category of sets. Another such example involves the category of modest sets (15.8.3).

This section requires only Section 14.1 to be read, and is not needed in the rest of the book.

14.5.1 The difficulty with finding an object $D \cong [D \longrightarrow D]$ using a fixed point of a functor is that the operation that takes D to $[D \longrightarrow D]$ is not a functor since an arrow $D \longrightarrow E$ induces arrows $[D \longrightarrow D]$ to $[D \longrightarrow E]$ and $[E \longrightarrow D]$ to $[D \longrightarrow D]$ as well as others (see Proposition 6.2.1), but no arrow in either direction between $[D \longrightarrow D]$ and $[E \longrightarrow E]$. There is an ingenious trick due to Smyth and Plotkin [1983] to solve this problem.

Some categories have additional structure on their hom sets; the hom set has the structure of an object from some category $\mathscr{V}$ and the hom functors are arrows of $\mathscr{V}$. Such a category is said to be **enriched over** $\mathscr{V}$. (The actual definition of enriched is more abstract than that.) In particular, **Cat**, and any other 2-category, is enriched over **Cat** itself, since its hom sets are themselves categories (with the arrows as objects and the natural transformations as arrows) and the hom functors preserve the extra structure. (See Section 4.8 and [Kelly, 1982a].) We will not define the general concept of enriched here, but will concentrate on a particular type of poset-enriched category (see Example 4.8.13).

14.5.2 Definition A category $\mathscr{C}$ is a **Smyth–Plotkin category** under the following conditions.

SP–1 $\mathscr{C}$ is cartesian closed.

SP–2 $\mathrm{Hom}(A, B)$ is a poset for each pair of objects A, B of $\mathscr{C}$.

SP–3 If $f : A \to B$, $g, h : B \to C$, $k : C \to D$ and $g \leq h$, then both $g \circ f \leq h \circ f$ and $k \circ g \leq k \circ h$.

SP–4 If $g, h : B \to C$ with $g \leq h$, then for any A, both

$$[g \to A] \leq [h \to A] : [C \to A] \to [B \to A]$$

(note that the order is *not* reversed) and

$$[A \to g] \leq [A \to h] : [A \to B] \to [A \to C]$$

SP–5 $\mathscr{C}$ has limits and colimits along countable chains.

SP–6 If $A = \lim A_i$, then the isomorphism

$$\mathrm{Hom}(B, A) \cong \lim \mathrm{Hom}(B, A_i)$$

for each object B is an order isomorphism.

SP–7 If $A = \mathrm{colim}\, A_i$, then the isomorphism

$$\mathrm{Hom}(A, C) \cong \lim \mathrm{Hom}(A_i, C)$$

for each object C is an order isomorphism.

The constructions in this section work if we replace cartesian closed categories by the symmetric monoidal closed categories that are the subject of the next chapter.

14.5.3 To a Smyth–Plotkin category $\mathscr{C}$ we associate two new categories $\mathbf{LA}(\mathscr{C})$ and $\mathbf{RA}(\mathscr{C})$. $\mathbf{LA}(\mathscr{C})$ (for left adjoint) has the same objects as $\mathscr{C}$. A homomorphism $A \to B$ is a pair of arrows of $\mathscr{C}$, $f : A \to B$ and $g : B \to A$ such that $\mathrm{id}_A \leq g \circ f$ and $f \circ g \leq \mathrm{id}_B$. $\mathbf{RA}(\mathscr{C})$ is defined similarly but with inequalities reversed, so that an arrow $A \to B$ is a pair (f, g) with $g \circ f \leq \mathrm{id}_A$ and $\mathrm{id}_B \leq f \circ g$. These really do mean left and right adjoint, respectively, when we consider the partially ordered sets as categories.

If $(f, g) : A \to B$ is in either $\mathbf{LA}(\mathscr{C})$ or $\mathbf{RA}(\mathscr{C})$, then f and g determine each other. For suppose that both (f, g) and (f, h) are arrows of $\mathbf{LA}(\mathscr{C})$, then $h \leq g \circ f \circ h \leq g$ and $g \leq h \circ f \circ g \leq h$, using the definition of homomorphism in $\mathbf{LA}(\mathscr{C})$ as well as SP–3 above. The proof that g determines f and the proofs in $\mathbf{RA}(\mathscr{C})$ are similar. A second proof follows from the fact that right and left adjoints determine each other up to isomorphism (see 13.3.4) and isomorphic objects in a poset are equal.

Next we observe that $\mathbf{LA}(\mathscr{C})^{\mathrm{op}} \cong \mathbf{RA}(\mathscr{C})$. In fact, an arrow in $\mathbf{LA}(\mathscr{C})$ from A to B is a pair (f, g) where $\mathrm{id} \leq g \circ f$ and $f \circ g \leq \mathrm{id}$. This, by definition, is an arrow from $B \to A$ in $\mathbf{LA}(\mathscr{C})^{\mathrm{op}}$. On the other hand an arrow from $B \to A$ in $\mathbf{RA}(\mathscr{C})$ is a pair (g, f) where $g : B \to A$ and $f : A \to B$ satisfy $f \circ g \leq \mathrm{id}$ and $\mathrm{id} \leq g \circ f$. Thus the isomorphism is the one that takes the pair (f, g) to the pair (g, f).

Now suppose that $\{(f_i, g_i) : A_i \to A\}$ is a cocone in $\mathbf{LA}(\mathscr{C})$. Purely formally, this is a colimit if and only if the cone $\{(f_i, g_i) : A \to A_i\}$ is a limit cone in $m\,\mathbf{LA}(\mathscr{C})^{\mathrm{op}}$. By the isomorphism above this is true if and only if the cone $\{(g_i, f_i) : A \to A_i\}$ is a limit cone in $\mathbf{RA}(\mathscr{C})$.

14.5.4 Definition The arrow (f, g) of either $\mathbf{LA}(\mathscr{C})$ or $\mathbf{RA}(\mathscr{C})$ is a **retract** if $g \circ f = \mathrm{id}$ and a **coretract** if $f \circ g = \mathrm{id}$.

14.5.5 Theorem *Let $\mathscr{C}$ be a Smyth–Plotkin category. Suppose that the hom sets in $\mathscr{C}$ have least upper bounds along countable increasing chains and that*

$$A_0 \to A_1 \to A_2 \to \cdots \to A_n \to \cdots \tag{14.4}$$

is a countable retract chain in $\mathbf{LA}(\mathscr{C})$ with arrows $(h_{ji}, k_{ij}) : A_i \to A_j$ for $i \leq j$. Suppose A is the colimit in $\mathscr{C}$ with transition arrows $f_i : A_i \to A$. Then there are (necessarily unique) arrows $g_i : A \to A_i$ such that (f_i, g_i) is a retract in $\mathbf{LA}(\mathscr{C})$ for each i, and such that A is the colimit of (14.4) in $\mathbf{LA}(\mathscr{C})$.

Dually, suppose that

$$\cdots \to A_n \to A_{n-1} \to \cdots \to A_1 \to A_0 \tag{14.5}$$

is a countable coretract chain in $\mathbf{RA}(\mathscr{C})$ with arrows $(k_{ji}, h_{ij}) : A_i \to A_j$ for $j \leq i$. Suppose A is the limit in $\mathscr{C}$ with transition arrows $g_i : A \to A_i$.

Then there are (necessarily unique) arrows $f_i : A_i \longrightarrow A$ such that (g_i, f_i) is a coretract in $\mathbf{RA}(\mathscr{C})$ and such that A is the limit of (14.5) in $\mathbf{RA}(\mathscr{C})$.

Proof. The two statements are dual to each other, as are their proofs. We sketch the proof of the second. To define for each i an arrow $f_i : A_i \longrightarrow A$ using the limit property, we have to give a consistent family of arrows $f_{ji} : A_i \longrightarrow A_j$. We let $f_{ji} = h_{ji}$ for $i < j$, $f_{ji} = k_{ji}$ for $i > j$ and $f_{ii} = \mathrm{id}$. (In point of fact, it is not hard to show that these arrows need be defined only for sufficiently large j so that only the case $i < j$ is important.) To show consistency with the cone, we calculate for $i < j < l$,

$$k_{jl} \circ f_{li} = k_{jl} \circ h_{li} = k_{jl} \circ h_{lj} \circ h_{ji} = \mathrm{id} \circ h_{lj} = f_{lj}$$

The cases where $j < i \leq l$ and $i \leq j < l$ are similar (easier, in fact). The result is an arrow $f_i : A_i \longrightarrow A$ such that $g_j \circ f_i = f_{ji}$. In particular, $g_i \circ f_i = \mathrm{id}$. Finally we have for $i < j$, $g_j \circ f_i \circ g_i = f_{ji} \circ g_i = h_{ji} \circ g_i = h_{ji} \circ k_{ij} \circ g_j \leq g_j = g_j \circ \mathrm{id}$. At this point, we invoke the fact that the partial order on $\mathrm{Hom}(A, A)$ is the limit of those on $\mathrm{Hom}(A, A_j)$ so that it follows that $f_i \circ g_i \leq \mathrm{id}$.

This defines the arrows f_i. To prove that this is a limit in $\mathbf{RA}(\mathscr{C})$ is tedious, but not hard. If B is an object and (r_i, s_i) is a consistent family of arrows in $\mathbf{RA}(\mathscr{C})$, then since A is the limit in $\mathscr{C}$, there is a unique arrow $r : B \longrightarrow A$ such that $r_i = g_i \circ r$ for all i. For $i \leq j$, we have $s_i \circ g_i = s_j \circ h_{ji} \circ k_{ij} \circ g_j \leq s_j \circ g_j$. Thus the sequence of arrows $s_i \circ g_i \in \mathrm{Hom}(A, B)$ is a countable increasing sequence and has a sup we call s. To show that $s \circ r \geq \mathrm{id}$ it is sufficient to show that $g_j \circ s \circ r \geq g_j$ for each j. $g_j \circ s \circ r = g_j \circ \sup s_i \circ g_i = \sup_i g_j \circ s_i \circ g_i$. Now to do something with the sup, it sufficient to stick to $j > i$. For such i, $\sup_i g_j \circ s_i \circ g_i = \sup_i k_{ji} \circ s_i \circ r_i \circ g_i \geq \sup_i k_{ji} \circ g_i = \sup_i g_j = g_j$. Also $s \circ r = \sup s_n \circ g_n \circ r = \sup s_n \circ r_n \leq \mathrm{id}$ since every $s_n \circ r_n \leq 1$. We note that if every (r_n, s_n) is a coretract, that is $r_n \circ s_n = \mathrm{id}$, then the same is true of (r, s). The uniqueness of (r, s) follows from the uniqueness of r in $\mathscr{C}$ since r determines s. □

14.5.6 If $(f, g) : D \longrightarrow E$ is a homomorphism in $\mathbf{LA}(\mathscr{C})$, the diagram

$$\begin{array}{ccc} [D \longrightarrow D] & \xrightarrow{[D,f]} & [D \longrightarrow E] \\ {\scriptstyle [g,D]}\downarrow & & \downarrow{\scriptstyle [g,E]} \\ [E \longrightarrow D] & \xrightarrow[[E,f]]{} & [E \longrightarrow E] \end{array}$$

commutes and determines an arrow $[D, D] \to [E, E]$. This idea can be used directly to find a fixed point for this endoarrow functor. However, it turns out that a slight modification of this idea gives a more useful conclusion.

Let us begin with an object A of $\mathscr{C}$ with $A \neq 1$ and $A \cong A \times A$. Assume further that there is a retract pair $(h, k) : A \to [A \to A]$ in $\mathbf{LA}(\mathscr{C})$. Define a functor $F : \mathbf{LA}(\mathscr{C}) \to \mathbf{LA}(\mathscr{C})$ that takes B to $[B \to A]$ and $(f, g) : B \to C$ to $([g, A], [f, A])$ from $[B \to a]$ to $[C \to a]$. If (f, g) is a retract pair, so is $F(f, g)$.

Consider the sequence

$$A \to F(A) \to F^2(A) \to \cdots \to F^n(A) \to F^{n+1}(A) \to \cdots$$

with $(h_n, k_n) = F^n(h, k) : F^n(A) \to F^{n+1}(A)$. This is a countable retract chain (the arrow from the ith term to the jth is just the composite of the successive arrows). Let B be the colimit of this chain in $\mathscr{C}$. Then we know that this is also the colimit in $\mathbf{LA}(\mathscr{C})$ and that B is also the limit of the sequence

$$\cdots \to F^n(A) \to \cdots \to F(A) \to A$$

in $\mathscr{C}$. We claim that F commutes with this colimit. In fact, when the object C is the colimit in $\mathscr{C}$ of the sequence

$$C_0 \to C_1 \to \cdots \to C_n \to \cdots$$

$[C, A]$ is the limit of the sequence

$$\cdots \to [C_n \to A] \to \cdots \to [C_1 \to A] \to [C_0 \to A] \tag{14.6}$$

because $[-, A] : \mathscr{C}^{\mathrm{op}} \to \mathscr{C}$ is a right adjoint and thus preserves limits (see 13.3.7). But (14.6) being a limit is equivalent to the sequence

$$[C_0 \to A] \to [C_1 \to A] \to \cdots \to [C_n \to A] \to \cdots$$

being a colimit. But this is just

$$F(A_0) \to F(A_1) \to \cdots \to F(A_n) \to \cdots$$

which shows that F preserves the colimit of a countable chain of retracts. It follows that $F(B) \cong B$.

Now we have that $B \cong [B \to A]$. We supposed that $A \cong A \times A$ so that $B \times B \cong [B \to A] \times [B \to A] \cong [B, A \times A] \cong [B \to A] \cong B$. Also, $[B \to B] \cong [B \to [B \to A]] \cong [(B \times B) \to A] \cong [B \to A] \cong B$. Thus B is not only a solution to $B \cong [B \to B]$, but simultaneously to $B \cong B \times B$.

14.5.7 Examples Many categories of posets satisfy Theorem 14.5.5. Let $\mathscr{C}$ be a (not necessarily full) subcategory of posets and order-preserving maps. The set of order-preserving functions between two objects of $\mathscr{C}$ is partially ordered by saying that $f \leq g$ if $f(x) \leq g(x)$ for all x in the domain of f. Whether $\mathscr{C}$ is cartesian closed and satisfies the other conditions of the theorem we have just proved to be useful is something that has to be proved in each special case.

Here is the example that especially interests us. Let $\mathscr{C}$ be the category whose objects are ω-CPOs and whose arrows are functions that preserve countable sups as described in 2.4.3. It is not hard to see that this category is cartesian closed. In fact, we know that there is a one to one correspondence between functions $A \times B \longrightarrow C$ and functions $A \longrightarrow [B \longrightarrow C]$; the thing to do is to show that this isomorphism remains when everything in sight preserves countable sups and that the isomorphism itself preserves countable sups. We leave this as an exercise. The category has limits computed pointwise. Colimits are not quite so simple. The easiest way to compute a colimit is to first compute the colimit as a poset and then add freely the colimits of countable increasing sequences.

We want to find an object $A \neq 1$ with $A \cong A \times A$ and A having an arrow $A \longrightarrow [A \longrightarrow A]$ in $\mathbf{LA}(\mathscr{C})$. Take any object A_0 of $\mathscr{C}$ with $A_0 \neq 1$ and having a least element $\perp$. Then the product A of countably many copies of A_0 will have the property that $A \cong A \times A$. For example, the function that takes the countable sequence $(a_0, a_1, a_2, a_3, \ldots)$ to $((a_0, a_2, a_4, \ldots), (a_1, a_3, a_5, \ldots))$ is an isomorphism.

Since A_0 has a least element, so does A. Let $f : A \longrightarrow [A \longrightarrow A]$ be the arrow that takes an element to the constant function at that element which is certainly an order-preserving function. Let $g : [A \longrightarrow A] \longrightarrow A$ take an order-preserving function to its value at $\perp$. Then it is immediate that $g \circ f = \mathrm{id}$ and that $f \circ g \leq \mathrm{id}$ so we have an arrow of $\mathbf{LA}(\mathscr{C})$.

We can now apply the construction of 14.5.6 to get an ω-CPO D for which $D \cong D \times D \cong [D \longrightarrow D]$.

14.5.8 Exercises

1.† Show that the category of ω-CPOs and functions that preserve countable sups is cartesian closed.

2.† Show that every poset freely generates an ω-CPO. (Let P be a poset. Let $\widehat{P}$ denote the set of all countably increasing chains of P. If $c = \{c_n\}$ and $c' = \{c'_n\}$ are two such chains, say that $c \leq c'$ if every element of c is less than or equal to some element of c'. Say that $c = c'$ if both $c \leq c'$ and $c' \leq c$.)

3.† Show that the category of ω-CPOs has limits (computed pointwise) and colimits. Assume that the category of posets has colimits.

4. A subset of a poset is called **directed** or **filtered** if every pair of elements has a common upper bound in the subset. Show that a poset is countably chain complete if and only if every countable directed subset has a least upper bound.

15
Toposes

A topos is a cartesian closed category with some extra structure which produces an object of subobjects for each object. This structure makes toposes more like the category of sets than cartesian closed categories generally are.

Toposes, and certain subcategories of toposes, have proved attractive for the purpose of modeling computation. A particular reason for this is that in a topos, a subobject of an object need not have a complement. One of the fundamental facts of computation is that it may be possible to list the elements of a subset effectively, but not the elements of its complement (see [Lewis and Papadimitriou, 1981], Theorems 6.1.3 and 6.1.4.). Sets which cannot be listed effectively do not exist for computational purposes, and toposes provide a universe of objects and functions which has many nice set-like properties but which does not force complements to exist. We discuss one specific subcategory of a topos, the category of modest sets, which has been of particular interest in the semantics of programming languages.

Toposes have interested mathematicians for other reasons. They are an abstraction of the concept of sheaf, which is important in pure mathematics. They allow the interpretation of second-order statements in the category in an extension of the language associated to cartesian closed categories in Chapter 6. This fact has resulted in toposes being proposed as an alternative to the category of sets for the foundations of mathematics. Toposes can also be interpreted as categories of sets with an internal system of truth values more general than the familiar two-valued system of classical logic; this allows an object in a topos to be thought of as a variable or time-dependent set, or as a set with various degrees of membership. In particular, most ways of defining the category of fuzzy sets lead to a category which can be embedded in a topos.

Sections 15.1 and 15.2 describe the basic properties of toposes, for the most part without proof. Section 15.3 takes a closer look at an aspect of toposes which make many of them a better model of computation than, for example, **Set**.

Sections 15.4 and 15.5 describe a special case of categories of sheaves which makes the connection with sets with degrees of membership clear. The category of graphs is discussed as an example there. Section 15.6 describes the connection with fuzzy sets.

In Section 15.7 we describe category objects in a category, a notion that is needed in Section 15.8, which is a brief description of the realizability topos and modest sets.

This chapter depends on Chapters 1 through 6, Chapter 9, and Chapter 13. In addition, Section 15.7 needs 10.1.5, 11.1 and 11.2.

Sections 15.1 and 15.2 are needed in all the remaining sections. After that, the chapter consists of four independent units: Section 15.3, Sections 15.4 and 5, Section 15.6 and Sections 15.7 and 8.

Basic properties of toposes are [Johnstone, 1977], [Barr and Wells, 1985], [Lambek and Scott, 1986], [Bell, 1988], [McLarty, 1992], [Mac Lane and Moerdijk, 1992]. None of these are aimed at applications to computer science. We do not discuss the language and logic corresponding to a topos in this book. The most accessible introduction to this is perhaps that of [McLarty, 1992], Chapter 16. Other discussions of the language and the relation with logic are in [Makkai and Reyes, 1977], [Fourman, 1977], [Fourman and Vickers, 1986], [Boileau and Joyal, 1981]. The texts [Makkai and Reyes, 1977] and [Freyd and Scedrov, 1990] discuss toposes and also more general classes of categories that have a rich logical structure. The use of toposes specifically for semantics is discussed in [Hyland, 1982], [Hyland and Pitts, 1989], [Vickers, 1992].

15.1 Definition of topos

15.1.1 The subobject functor Recall from 2.8.11 that if C is an object of a category, a subobject of C is an equivalence class of monomorphisms $C_0 \rightarrowtail C$ where $f_0 : C_0 \rightarrowtail C$ is equivalent to $f_1 : C_1 \rightarrowtail C$ if and only if there are arrows (necessarily isomorphisms) $g : C_0 \to C_1$ and $h : C_1 \to C_0$ such that $f_1 \circ g = f_0$ and $f_0 \circ h = f_1$.

Assuming the ambient category $\mathscr{C}$ has pullbacks, the 'set of subobjects' function is the object function of a functor $\mathrm{Sub} : \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$: precisely, for an object C, $\mathrm{Sub}(C)$ is the set of subobjects of C. We must define Sub on arrows.

If $k : C' \to C$ is an arrow and if $f_0 : C_0 \rightarrowtail C$ represents a subobject of C, then in a pullback

$$\begin{array}{ccc} C'_0 & \xrightarrow{k_0} & C_0 \\ {\scriptstyle f'_0}\downarrow & & \downarrow{\scriptstyle f_0} \\ C' & \xrightarrow[k]{} & C \end{array} \qquad (15.1)$$

the arrow f'_0 is also a monomorphism (see 9.3.4).

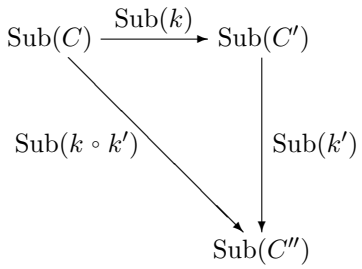
It is left as an exercise to prove, using the universal mapping property of pullbacks, that if the monomorphism $f_0 : C_0 \rightarrowtail C$ is equivalent to $f_1 : C_1 \rightarrowtail C$, then the pullbacks $f_0' : C_0' \rightarrowtail C'$ and $f_1' : C_1' \rightarrowtail C'$ are also equivalent. Thus not only is a pullback of a monomorphism a monomorphism, but also a pullback of a subobject is a subobject.

Thus we can define, for an arrow k as above,

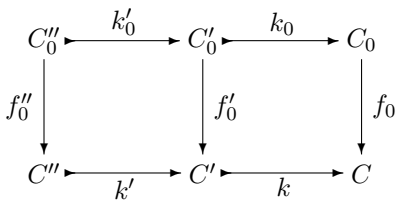
$$\mathrm{Sub}(k) : \mathrm{Sub}(C) \to \mathrm{Sub}(C')$$

to be the function that sends the equivalence class containing f_0 to the equivalence class containing the pullback f_0'.

To show that this is a functor, we must show that the identity arrow induces the identity arrow on subobjects (exercise) and that if $k' : C'' \to C'$, then the diagram



commutes. But the commutativity of this diagram at the subobject represented by f_0 is equivalent to the outer rectangle of the diagram



being a pullback when the two smaller squares are, which is true by Exercise 10 of Section 9.3.

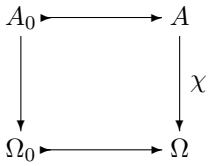
15.1.2 Definition A **topos** is a category which

TOP–1 has finite limits;

TOP–2 is cartesian closed;

TOP–3 has a representable subobject functor.

We know that a functor is representable if and only if it has a universal element (see 4.5.9). A universal element of the subobject functor is an object, usually called Ω, and a subobject $\Omega_0 \subseteq \Omega$ such that for any object A and subobject $A_0 \subseteq A$, there is a unique arrow $\chi : A \longrightarrow \Omega$ such that there is a pullback



It can be proved that Ω_0 is the terminal object and the left arrow is the unique arrow from A_0 ([Barr and Wells, 1985], Proposition 4 of Section 2.3).

The object Ω is called the **subobject classifier** and the arrow from $\Omega_0 = 1 \longrightarrow \Omega$ is usually denoted true. The arrow χ corresponding to a subobject is called the **characteristic arrow** of the subobject.

The fact that the subobject functor is represented by Ω means precisely that there is a natural isomorphism

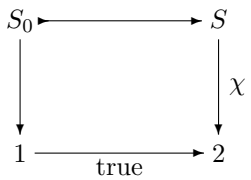
$$\phi : \mathrm{Sub}(-) \longrightarrow \mathrm{Hom}(-, \Omega)$$

which takes a subobject to its characteristic function.

15.1.3 Example The category of sets is a topos. It was shown in 6.1.9 that sets are a cartesian closed category. A two-element set, which we call 2, is a subobject classifier. In fact, call the two elements true and false. Given a set S and subset $S_0 \subseteq S$, define the characteristic function $\chi : S \longrightarrow \{\mathrm{true}, \mathrm{false}\}$ by

$$\chi(x) = \begin{cases} \text{true} & \text{if } x \in S_0 \\ \text{false} & \text{if } x \notin S_0 \end{cases}$$

Then the following square (where the top arrow is inclusion) is a pullback:



Thus 2 is a subobject classifier.

15.1.4 Exercises

1. Referring to Diagram (15.1), show that if the monomorphism $f_0 : C_0 \rightarrowtail C$ is equivalent to $f_1 : C_1 \rightarrowtail C$, then for any $g : C' \to C$, the pullbacks $f'_0 : C'_0 \rightarrowtail C'$ and $f'_1 : C'_1 \rightarrowtail C'$ are also equivalent.

2. Show that the identity arrow $C \to C$ induces the identity arrow $\mathrm{Sub}(C) \to \mathrm{Sub}(C)$.

3. Show that the category of finite sets and all functions between them is a topos.

4.† (Requires some knowledge of infinite cardinals.) Show that the category of finite and countably infinite sets and all functions between them has a subobject classifier but is not cartesian closed. (Hint: the set of subsets of a countable set is not countable.)

15.2 Properties of toposes

We list here some of the properties of toposes, without proof.

15.2.1 In the first place, a topos is not only cartesian closed, it is locally cartesian closed (see Section 13.4). This is Corollary 1.43, p. 36 of [Johnstone, 1977], Corollary 7, p. 182 of [Barr and Wells, 1985] or section 17.2 of [McLarty, 1992].

15.2.2 Power objects In any topos, the object $[A \to \Omega]$ has the property that

$$\mathrm{Hom}(B, [A \to \Omega]) \cong \mathrm{Hom}(A \times B, \Omega) \cong \mathrm{Sub}(A \times B) \qquad (15.2)$$

These isomorphisms are natural when the functors are regarded as functors of either A or of B. (One of them appears, for **Set**, in Example 4.3.7.) The object $[A \to \Omega]$ is often called the **power object** of A and denoted $\mathscr{P}A$. It is the topos theoretic version of the powerset of a set. Theorem 1 of Section 5.4 of [Barr and Wells, 1985] implies that a category with finite limits is a topos if for each object A there is a power object that satisfies (15.2).

The inverse image and universal image constructions in 13.2.7 for the powerset of a set can be made on $[A \to \Omega]$ for any object A in a topos. The left and right adjoints of the pullback functors (they exist because any topos is locally cartesian closed) are related to these images via the diagram in [Johnstone, 1977], Proposition 5.29; this diagram is called the 'doctrinal diagram' and is the basis for introducing elementary (first order) logic into a topos.

15.2.3 Effective equivalence relations Let $d, e : E \rightrightarrows A$ be two arrows in a category. For any object B we have a single function

$$\langle \mathrm{Hom}(B,d), \mathrm{Hom}(B,e)\rangle : \mathrm{Hom}(B,E) \longrightarrow \mathrm{Hom}(B,A) \times \mathrm{Hom}(B,A)$$

which sends f to the pair $(d \circ f, e \circ f)$. If this function is, for each object B, an isomorphism of $\mathrm{Hom}(B,E)$ with an equivalence relation on the set $\mathrm{Hom}(B,A)$, then we say that E is an **equivalence relation** on the object A. This means that the image of $\langle \mathrm{Hom}(B,d), \mathrm{Hom}(B,e)\rangle$ is actually an equivalence relation on $\mathrm{Hom}(B,A)$.

This can be thought of as embodying two separate conditions. First, the function $\langle \mathrm{Hom}(B,d), \mathrm{Hom}(B,e)\rangle$ must be an injection, because we are supposing that it maps $\mathrm{Hom}(B,E)$ isomorphically to a subset of $\mathrm{Hom}(B,A) \times \mathrm{Hom}(B,A)$. Secondly, that subset must satisfy the usual reflexivity, symmetry and transitivity conditions of equivalence relations.

15.2.4 Kernel pairs Here is one case in which this condition is automatic. If $g : A \longrightarrow C$ is an arrow, the pullback of the square

$$\begin{array}{ccc} E & \xrightarrow{d} & A \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle g} \\ A & \xrightarrow[g]{} & C \end{array}$$

is called a **kernel pair** of g. Notation: we will write that

$$E \underset{e}{\overset{d}{\rightrightarrows}} A \xrightarrow{g} C$$

is a kernel pair. For an object B of $\mathscr{C}$, the definition of this limit is that there is a one to one correspondence between arrows $B \longrightarrow E$ and pairs of arrows (h,k) from B to A such that $g \circ h = g \circ k$ and that the correspondence is got by composing an arrow from B to E with d and e, resp. To put it in other words, $\mathrm{Hom}(B,E)$ is isomorphic to

$$\{(h,k) \in \mathrm{Hom}(B,A) \times \mathrm{Hom}(B,A) \mid g \circ h = g \circ k\}$$

which is an equivalence relation.

15.2.5 Suppose that $E \rightrightarrows A$ describes an equivalence relation. We say that the equivalence relation is **effective** if it is a kernel pair of some arrow from A. We say that a category has **effective equivalence relations** if every equivalence relation is effective. We give the following without proof. The interested reader may find the proof in [Barr and Wells, 1985] Theorem 7 of Section 2.3, [Johnstone, 1977], Proposition 1.23, p. 27, or [McLarty, 1992], Section 16.7.

15.2.6 Theorem *In a topos, every equivalence relation is effective.*

15.2.7 Example Equivalence relations in the categories **Set** and **Mon** are effective. An equivalence relation in **Set** is simply an equivalence relation, and the class map to the quotient set is a function that has the equivalence relation as kernel pair. An equivalence relation on a monoid M in **Mon** is a congruence relation on M (see Exercise 6 of Section 3.5); it is effective because a monoid multiplication can be defined on the quotient set of the congruence relation that makes the quotient map a homomorphism (Exercise 3).

There are many categories which lack effective equivalence relations. One is the category of partially ordered sets and monotone maps. Here is a simple example. Let C be a two-element chain $x < y$. Consider the subset E of $C \times C$ consisting of all four pairs (x, x), (x, y), (y, x) and (y, y). The only ordering is that $(x, x) \leq (y, y)$. Then E is an equivalence relation, but is not the kernel pair of any arrow out of C. The least kernel pair that includes E has the same elements as E, but has the additional orderings $(x, x) \leq (x, y) \leq (y, y)$ and $(x, x) \leq (y, x) \leq (y, y)$.

Other important properties of toposes are contained in the following.

15.2.8 Theorem *Let $\mathscr{E}$ be a topos.*

(a) *$\mathscr{E}$ has finite colimits.*

(b) *$\mathscr{E}$ has finite disjoint and universal sums.*

(c) *Every epi in $\mathscr{E}$ is regular, and $\mathscr{E}$ is a regular category.*

An early proof of the fact that a topos has finite colimits ([Mikkelson, 1976]) mimicked the construction in sets. For example, the coequalizer of two arrows was constructed as a set of subsets, which can be identified as the set of equivalence classes mod the equivalence relation generated by the two arrows. However, the argument is difficult. The modern proof (due to Paré) is much easier, but it involves some legerdemain with triples and it is hard to see what it is actually doing. See [Barr and Wells, 1985], Corollary 2 of 5.1 for the latter proof. The rest is found in 5.5 of the same source.

15.2.9 The initial topos There is an FL sketch whose models are toposes. (See [Barr and Wells, 1985], Section 4.4. In that book, FL sketches are called LE sketches.) It follows that there is an initial model of the topos axioms. This topos lacks a natural numbers object. It might be an interesting model for a rigidly finitistic model of computation, but would not allow the modeling of such things as recursion.

The phrase 'initial topos' is usually reserved for the initial model of the axioms for toposes with a natural numbers object (Section 5.5). This category provides an interesting model for computation. The arrows from **N** to **N**

in that category are, not surprisingly, the total recursive functions. In fact, all partial recursive functions are modeled in there as well, but as partial arrows, which we now describe.

15.2.10 Partial arrows In 2.1.13 we discussed partial functions between sets. This concept can be extended to any category. Let A and B be objects. A partial arrow A to B consists of a subobject $A_0 \subseteq A$ and an arrow $f : A_0 \longrightarrow B$. This defines the partial arrow f in terms of a particular representative $A_0 \longrightarrow A$ of the subobject, but given any other representative $A_0' \longrightarrow A$, there is a unique arrow from A_0' to A_0 commuting with the inclusions which determines an arrow from A_0' to B by composition with f. The subobject determined by A_0 is called the **domain** of the partial arrow. If $g : A_1 \longrightarrow B$ is another partial arrow on A we say the $f \leq g$ if $A_0 \subseteq A_1$ and the restriction of g to A_0 is f. If we let $i : A_0 \longrightarrow A_1$ denote the inclusion arrow, then the second condition means simply that $g \circ i = f$. We will say that f and g are the same partial arrow if both $f \leq g$ and $g \leq f$. This means that the domains of f and g are the same subobject of A and that f and g are equal on that domain.

We say that **partial arrows to B are representable** if there is an object $\widetilde{B}$ and an embedding $B \rightarrowtail \widetilde{B}$ such that there is a one to one correspondence between arrows $A \longrightarrow \widetilde{B}$ and partial arrows A to B, the correspondence given by pulling back:

$$\begin{array}{ccc} A_0 & \longrightarrow & A \\ \downarrow & & \downarrow \\ B & \longrightarrow & \widetilde{B} \end{array}$$

In a topos, the arrow true: $1 \longrightarrow \Omega$ represents partial functions to 1. The reason is that since each object has a unique arrow to 1, a partial arrow from A to 1 is equivalent to a subobject of A.

15.2.11 Theorem *In a topos, partial arrows into any object are representable.*

See [Johnstone, 1977], Section 1.26 or [McLarty, 1992], Section 17.1 for the proof.

15.2.12 Exercises

1. Verify the isomorphisms of (15.2) for **Set**.

2. Let S be a set and E be an equivalence relation on S. Then there are two arrows E to S, being the inclusion of E into $S \times S$, followed by the two projections on S. Show that E, together with these two arrows, is the kernel pair of the function that takes each element of S to the equivalence class containing it.

3. a. Let $d, e : E \rightrightarrows M$ be two monoid homomorphisms. Show that they form an equivalence relation in **Mon** if and only if the arrow

$$E \xrightarrow{\langle d, e \rangle} M \times M$$

is a monomorphism and the image of $\langle d, e \rangle$ is a congruence on M (see Exercise 6 of Section 3.5 with d, e the projections).

b. Show that equivalence relations in **Mon** are effective.

4. Show that in any category with kernel pairs, if $f : C \to D$ is a coequalizer, then it is the coequalizer of its kernel pair.

5. Give an example in **Set** of a function with a kernel pair which is not the coequalizer of its kernel pair.

6. Show that in the category of sets, $\widetilde{S}$ can be taken to be $S \cup \{*\}$, where $*$ is an element not in S.

7. Show that in a topos, the subobject classifier is $\widetilde{1}$, the object that represents partial arrows into 1.

15.3 Is a two-element poset complete?

This discussion requires familiarity with the concept of recursive set (there is an algorithm that always halts that tells whether an element is in the set or not) and recursively enumerable set (there is an algorithm that generates all the elements of the set and no others).

We consider a 2 element chain we denote by 2. The word 'complete' in the question in the heading means that it has sups of all subsets. The question seems absurd at first, but it improves with age. In fact, we will show that both in a topos and in realistic models of computation, the answer is 'no'. Moreover the answer is no in both cases for the same reason.

The claim we want to make, and for which this discussion is evidence, is that topos semantics is an appropriate model for computation, or at least a more appropriate one than set theory. This will not mean that topos theory will tell you how to compute something that you could not compute without

it. What happens is that in many toposes certain computationally meaningless constructions cannot be made at all, although they are all possible in **Set**.

15.3.1 Computation models Of course, in a most naive sense, 2 is certainly complete as a poset, but we want to look at this in a more sophisticated way. What we really want to be true if we say that a poset P is complete is that for any other object A, the poset $[A \longrightarrow P]$, with the pointwise order, is complete. In the case of 2, this means that $[A \longrightarrow 2]$ is complete. To see why we want this to be true, recall that an element of a set is essentially the same thing as a function from a one-element set to that set. However, in categories thought of as workspaces, it is more useful to think of *any* arrow with codomain S as an element of S – a *variable* element of S. (A variable elements whose domain is a terminal object is then a constant element or global element (2.7.19)). Categories rich enough to be workspaces for a type of mathematics typically come with an **internal language**; statements in that internal language can often be perceived as statements about elements, but their truth is dependent on 'elements' being interpreted as variable elements.

In the case of ordinary set theory, it is still true that 2 is complete in this internal sense. For computational semantics, the situation is different. Consider the case $A = \mathbf{N}$, although any infinite set would do as well. A function $\mathbf{N} \longrightarrow 2$ is determined by and determines a subset of $\mathbf{N}$. However, a computable function $\mathbf{N} \longrightarrow 2$ is determined by and determines a *recursive* subset of $\mathbf{N}$. And it is well known that the set of recursive subsets of $\mathbf{N}$ is not complete. It is not even countably complete; in fact, the recursively enumerable subsets are characterized as the countable unions of recursive subsets. Of course, arbitrary unions of recursive subsets will be even worse.

15.3.2 Topos models The situation in an arbitrary topos is similar. A topos has an object Ω with the property that for any object A, $\mathrm{Hom}(A, \Omega)$ is the set of subobjects of A. A topos also always has an object $2 = 1 + 1$, and an arrow $A \longrightarrow 2$ does not represent an arbitrary subobject, but rather a complemented subobject (see 9.6.2). Such an arrow $A \longrightarrow 2$ gives a decomposition of A as a sum $A_0 + A_1$ where A_0 and A_1 are the pullbacks shown below, where $\mathrm{true}, \mathrm{false} : 1 \longrightarrow 2$ are the two injections:

$$\begin{array}{ccc} A_0 & \longrightarrow & A \\ \downarrow & & \downarrow \\ 1 & \xrightarrow[\mathrm{false}]{} & 2 \end{array} \qquad \begin{array}{ccc} A_1 & \longrightarrow & A \\ \downarrow & & \downarrow \\ 1 & \xrightarrow[\mathrm{true}]{} & 2 \end{array}$$

The fact that a topos has universal sums implies that from $2 = 1 + 1$ we can conclude that $A = A_0 + A_1$. On the other hand, if $A = A_0 + A_1$, there is a unique arrow $A \longrightarrow 2$ whose restriction to A_0 is false $\circ \langle\rangle$ and to A_1 is true $\circ \langle\rangle$. Thus 2 is internally complete in a topos if and only if the supremum of complemented sets are complemented. It is not hard to show that this is not true in general. In fact, there is a topos in which the arrows from $\mathbf{N}$ to 2 are just the total recursive two-valued functions (and in fact, the arrows from $\mathbf{N}$ to itself are also the total recursive functions). In that topos a complemented subset of $\mathbf{N}$ is exactly a recursive subset (one which, with its complement, has a recursive characteristic function) and the fact that a union of recursive subsets is not necessarily recursive finishes the argument.

In 6.6.4 we constructed a semantics for an `If` loop by constructing a supremum in $[A \longrightarrow D_\perp]$, where $D_\perp$ is a flat CPO (of which 2 is an example). However, the last paragraph shows that in general this semantics does not make computational sense. The reason is that, although you can write down the sup as a formal thing, it will not be guaranteed to give a terminating program. The sup of partial functions exists, but the domain of such a partial function is not generally computable.

In a topos model of computational semantics, in which all arrows $\mathbf{N} \longrightarrow \mathbf{N}$ are given by recursive functions, a subobject of $\mathbf{N}$ is determined by an arrow into Ω. The maps into Ω cannot usually be computed. The one special case in which they can is that of an arrow that factors through the subobject $\langle \text{true}; \text{false} \rangle : 2 \longrightarrow \Omega$. This corresponds to the traditional distinction between recursively enumerable and recursive subsets. In classical set theory, all subsets are classified by arrows into 2, but here only recursive subsets are.

15.4 Presheaves

15.4.1 Definition Let $\mathscr{C}$ be a category. A functor $E : \mathscr{C}^{\text{op}} \longrightarrow \mathbf{Set}$ is called a **presheaf** on $\mathscr{C}$. Thus a presheaf on $\mathscr{C}$ is a contravariant functor. The presheaves on $\mathscr{C}$ with natural transformations as arrows forms a category denoted $\mathbf{Psh}(\mathscr{C})$.

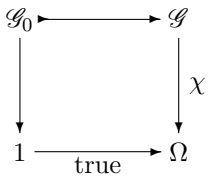
We considered presheaves as actions in Section 3.2. They occur in other guises in the categorical and computer science literature, too. For example, a functor $F : A \longrightarrow \mathbf{Set}$, where A is a set treated as a discrete category, is a 'bag' of elements of A. If $a \in A$, the set $F(a)$ denotes the multiplicity to which a occurs in A. See [Taylor, 1989] for an application in computing science.

15.4.2 Proposition *The category of presheaves on a category $\mathscr{C}$ form a topos.*

The proof may be found in [Barr and Wells, 1985] Section 2.1, Theorem 4. That proof uses a different, but equivalent, definition of topos.

15.4.3 Example Consider the category we will denote by $0 \rightrightarrows 1$. It has two objects and four arrows, two being the identities. A contravariant set-valued functor on this category is a pair of objects G_0 and G_1 and a pair of arrows we will call source, target : $G_1 \to G_0$. The two identities are sent to identities. Thus the category of presheaves on this category is the category of graphs, which is thereby a topos. This category is the theory of the sketch for graphs given in 4.6.7.

We described the exponential object for graphs in 6.1.12. It is instructive to see what the subobject classifier is. We have to find a graph Ω and an arrow true : $1 \to \Omega$ such that for any graph $\mathscr{G}$ and subgraph $\mathscr{G}_0 \subseteq \mathscr{G}$, there is a unique graph homomorphism $\chi : \mathscr{G} \to \Omega$ such that the diagram



commutes.

We define the graph Ω as follows. It has five arrows we will call 'all', 'source', 'target', 'both' and 'neither'. The reason for these names should become clear shortly. It has two nodes we will call 'in' and 'out'. The arrows 'all' and 'both' go from 'in' to itself, 'neither' goes from 'out' to itself. The arrow 'source' goes from 'in' to 'out' and 'target' from 'out' to 'in'. The terminal graph, which has one arrow and one node, is embedded by the function true that takes the arrow to 'all' and the node to 'in'.

Now given a graph $\mathscr{G}$ and a subgraph $\mathscr{G}_0$ we define a function $\chi : \mathscr{G} \to \Omega$ as follows. For a node n of $\mathscr{G}$, $\chi(n)$ is 'in' or 'out', according to whether n is in $\mathscr{G}_0$ or not. For an arrow a, we let $\chi(a)$ be 'all' if $a \in \mathscr{G}_0$ (whence its source and target are as well). If not, there are several possibilities. If the source but not the target of a belongs to $\mathscr{G}_0$, then $\chi(a) =$'source'. Similarly, if the target but not the source is in $\mathscr{G}_0$, it goes to 'target'. If both the source and target are in it, then $\chi(a) =$'both' and if neither is, then it goes to 'neither'.

15.4.4 Exercises

1. Show that the graphs No and Ar discussed in 6.1.12 are actually the contravariant set-valued functors on $0 \rightrightarrows 1$ represented by the objects 0 and 1, respectively.

2. Show that the object Ω in the category of graphs can be described as follows. The nodes are the subgraphs of No and the arrows are the subgraphs of Ar and the source and target are induced by s and t (defined in 6.1.12), respectively.

3. Let $\mathscr{C}$ be a category, and let M and N be two objects of $\mathbf{Psh}(\mathscr{C})$. Use the Yoneda Lemma and the adjunction defining cartesian closed categories to show that for any object X of $\mathscr{C}$, $[M \to N](X)$ must be the set of natural transformations from $\mathrm{Hom}(-, X) \times M$ to N, up to isomorphism. Note that this does not prove that $\mathbf{Psh}(\mathscr{C})$ *is* cartesian closed. That is true, but requires ideas not given here (see [Mac Lane and Moerdijk, 1992], page 46).

4. Let $\mathscr{C}$ be a category and X an object of $\mathscr{C}$. Let Ω be the subobject classifier of $\mathbf{Psh}(\mathscr{C})$. Show that $\Omega(X)$ is, up to isomorphism, the set of subfunctors (see Exercise 3 of Section 4.3) of $\mathrm{Hom}(-, X)$.

15.5 Sheaves

The general definition of sheaves requires a structure on the category called a **Grothendieck topology**. The most accessible and detailed discussion of Grothendieck topologies is that of [Mac Lane and Moerdijk, 1992]. Here we will discuss the special case of sheaves in which the category is a partial order.

15.5.1 Let P be a partially ordered set. From the preceding section, a presheaf E on P assigns to each element $x \in P$ a set $E(x)$ and whenever $x \leq y$ assigns a function we will denote $E(x, y) : E(y) \to E(x)$ (note the order; *x precedes y, but the arrow is from $E(y)$ to $E(x)$*). This is subject to two conditions. First, that $E(x, x)$ be the identity function on $E(x)$ and second that when $x \leq y \leq z$, that $E(x, y) \circ E(y, z) = E(x, z)$. The arrows $E(x, y)$ are called **restriction functions**.

15.5.2 Heyting algebras We make the following supposition about P.

HA–1 There is a top element, denoted 1, in P.

HA–2 Each pair of elements $x,\ y \in P$ has an infimum, denoted $x \wedge y$.

HA–3 Every subset $\{x_i\}$ of elements of P has a supremum, denoted $\bigvee x_i$.

HA–4 For every element $x \in P$ and every subset $\{x_i\} \subseteq P$, $x \wedge (\bigvee x_i) = \bigvee(x \wedge x_i)$.

A poset that satisfies these conditions is called a **complete Heyting algebra**.

15.5.3 If $\{x_i\}$ is a subset with supremum x, and E is a presheaf, there is given a restriction function $e_i : E(x) \longrightarrow E(x_i)$ for each i. The universal property of product gives a unique function $e : E(x) \longrightarrow \prod_i E(x_i)$ such that $p_i \circ e = e_i$. In addition, for each pair of indices i and j, there are functions $c_{ij} : E(x_i) \longrightarrow E(x_i \wedge x_j)$ and $d_{ij} : E(x_j) \longrightarrow E(x_i \wedge x_j)$ induced by the relations $x_i \geq x_i \wedge x_j$ and $x_j \geq x_i \wedge x_j$. This gives two functions $c, d : \prod_i E(x_i) \longrightarrow \prod_{ij} E(x_i \wedge x_j)$ such that

$$\begin{array}{ccc} \prod_i E(x_i) & \xrightarrow{\ c\ } & \prod_{ij} E(x_i \wedge x_j) \\ \downarrow{\scriptstyle p_i} & & \downarrow{\scriptstyle p_{ij}} \\ E(x_i) & \xrightarrow[c_{ij}]{} & E(x_i \wedge x_j) \end{array}$$

and

$$\begin{array}{ccc} \prod_i E(x_i) & \xrightarrow{\ d\ } & \prod_{ij} E(x_i \wedge x_j) \\ \downarrow{\scriptstyle p_i} & & \downarrow{\scriptstyle p_{ij}} \\ E(x_i) & \xrightarrow[d_{ij}]{} & E(x_i \wedge x_j) \end{array}$$

commute.

15.5.4 Definition A presheaf is called a **sheaf** if it satisfies the following additional condition:

$$x = \bigvee x_i$$

implies

$$E(x) \xrightarrow{e} \prod_i E(x_i) \underset{d}{\overset{c}{\rightrightarrows}} \prod_{ij} E(x_i \wedge x_j)$$

is an equalizer.

15.5.5 Theorem *The category of sheaves on a Heyting algebra is a topos.*

As a matter of fact, the category of sheaves for any Grothendieck topology is a topos (see any of the texts [Johnstone, 1977], [Barr and Wells, 1985], [Mac Lane and Moerdijk, 1992], [McLarty, 1992]).

15.5.6 Constant sheaves A presheaf E is called **constant** if for all $x \in P$, $E(x)$ is always the same set and for all $x \leq y$, the function $E(y, x)$ is the identity function on that set.

The constant presheaf at a one-element set is always a sheaf. This is because the sheaf condition comes down to a diagram

$$1 \to 1 \rightrightarrows 1$$

which is certainly an equalizer. No constant presheaf whose value is a set with other than one element can be a sheaf. In fact, the 0 (bottom) element of P is the supremum of the empty set and the product of the empty set of sets is a one-element set (see 5.3.6). Hence the sheaf condition on a presheaf E is that

$$E(0) \to \prod_{\emptyset} \rightrightarrows \prod_{\emptyset}$$

which is

$$E(0) \to 1 \rightrightarrows 1$$

and this is an equalizer if and only if $E(0) = 1$.

15.5.7 A presheaf is said to be **nearly constant** if whenever $0 < x \leq y$ in P, the restriction $E(y) \to E(x)$ is an isomorphism. It is interesting to inquire when a nearly constant presheaf is a sheaf. It turns out that every nearly constant presheaf over P is a sheaf over P if and only if the meet of two nonzero elements of P is nonzero.

To see this, suppose E is a nearly constant presheaf whose value at any $x \neq 0$ is S and that $x = \bigvee x_i$. In the diagram

$$E(x) \to \prod E(x_i) \rightrightarrows \prod E(x_i \wedge x_j)$$

every term in which $x_i = 0$ contributes nothing to the product since $1 \times Y \cong Y$. An element of the product is a string $\{s_i\}$ such that $s_i \in S$. The condition of being an element of the equalizer is the condition that the image of s_i under the induced function $E(x_i) \to E(x_i \wedge x_j)$ is the same as the image of s_j under $E(x_j) \to E(x_i \wedge x_j)$. But in a nearly constant sheaf, all these sets are the same and all the functions are the identity, so this condition is simply that $s_i = s_j$. But this means that an element of the equalizer must be the same in every coordinate, hence that diagram is an equalizer.

15.5.8 Interpretation of sheaves Let E be a sheaf on P. The reader will want to know how E is to be interpreted. Start by thinking of P as an algebra of truth values. Whereas propositions (assertions) in ordinary logic are either true or false (so that ordinary logic is often called 2-valued), in

P-valued logic, the truth of an assertion is an element of P. In the next section, we will use this idea to discuss time sheaves in which propositions can be true at some times and not at others. If p is some proposition, let us write $[\![p]\!]$ to denote the element of P that is the truth value.

A sheaf E is a set in this logic. For $x \in P$, the (ordinary) set $E(x)$ could, as a first approximation, be thought of as the set of all entities a for which $[\![a \in E]\!]$ is at least x. If $y < x$, then $[\![a \in E]\!] \geq x$ implies that $[\![a \in E]\!] \geq y$ so that $E(x) \subseteq E(y)$. This is only a first approximation and what we have described is actually a P-valued fuzzy set (see 15.6). The reason is that equality is also a predicate and it may happen, for example, that $[\![a = b]\!]$ could lie between x and y so that the entities a and b are not discernably equal at level x, but are equal at level y. The result is that rather than an inclusion, we have a restriction function $E(x) \longrightarrow E(y)$ for $y \leq x$.

15.5.9 Time sheaves, I Here is a good example of a topos in which one can see that the restriction arrows should not be expected to be injective. Consider the partially ordered set whose elements are intervals on the real line with inclusion as ordering. It is helpful to think of these as time intervals.

Now consider any definition of a naive set. Some possible definitions will be time invariant, such as the set of mathematical statements about, say, natural numbers, that are true. Others of these 'sets' change with time; one example might be the set of all books that are currently in print; another the set of statements currently known to be true about the natural numbers. These may conveniently be thought of as the presheaves whose value on some time interval is the set of books that were in print over the entire interval and the set of statements about natural numbers known to be true during that entire interval. The restriction to a subinterval is simply the inclusion of the set of books in print over the whole interval to that (larger) set of those in print over that subinterval or the restriction of the knowledge from the interval to the subinterval. In this example, the restrictions are injective.

Instead of books in print, we could take the example of businesses in operation. Because of the possibility of merger, divestment and the like, two businesses which are actually distinct over a large interval might coincide over a smaller subinterval. Thus for this example of a "set", the restriction function is not injective in general.

Another situation in which the restriction functions are not necessarily injective arises from the set of variables in a block-structured programming language. The presheaf is this: the set for a certain time interval during the running of the program is the quotient of the set of variables which exist over the whole time interval, modulo the equivalence relation that identifies two variables in an interval if they should happen to have the same value over the whole interval. Two variables may not be equivalent over a large

interval, whereas they may be equivalent over a smaller one; in that case the restriction function would not be injective.

In general, any property describes the set of all entities that have that property over the entire interval. The restriction to a subinterval arises out of the observation that any entity that possesses a property over an interval possesses it over any subinterval.

The sheaf condition in this case reduces to this: if the interval I is a union of subintervals I_k (where k ranges over an index set which need not be countable) and an entity possesses that property over every one of the subintervals I_k, then it does over the whole interval. This condition puts a definite restriction on the kinds of properties that can be used. For example, the property of being **grue**, that is blue over the first half of the interval and green on the second half, is not allowed. The properties that are allowed are what might be called **local**, that is true when they are true in each sufficiently small interval surrounding the point in time. This statement is essentially a tautology, but it does give an idea of what the sheaf condition means.

15.5.10 Time sheaves, II Here is another topos, rather different from the one above, that might also be considered to be time sheaves. Unlike the one above which is symmetric to forward and reverse time, this one definitely depends on which way time is flowing. This is not to say that one is better or worse, but they are different and have different purposes. In this one, the elements of the Heyting algebra are the times themselves. In other words, we are looking at time indexed sets, as opposed to sets indexed by time intervals.

We order this set by the reverse of the usual order. So a presheaf in this model is a family $\{X(t)\}$ of sets, t a real number, together with functions $f(s,t) : X(t) \longrightarrow X(s)$ for $t \leq s$, subject to the condition that $f(t,t)$ is the identity and $f(r,s) \circ f(s,t) = f(r,t)$ for $t \leq s \leq r$. The sheaf condition of Definition 15.5.4 is a bit technical, but can easily be understood in the special case of a presheaf all of whose transition arrows $f(s,t)$ are inclusions. In that case, the condition is that when $t = \bigwedge t_i$ (so that t is the greatest lower bound of the t_i), then $X(t) = \bigcap X(t_i)$.

An example, which might be thought typical, of such a sheaf might be described as the 'sheaf of states of knowledge'. At each time t we let $X(t)$ denote the set of all things known to the human race. On the hypothesis (which might be disputed) that knowledge is not lost, we have a function $X(t) \longrightarrow X(s)$ for $t \leq s$. In common parlance, we might consider this function to be an inclusion, or at least injective, but it is possible to modify it in various ways that will render it not so. We might introduce an equivalence relation on knowledge, so that two bits of knowledge might be considered the

same. In that case, if at some future time we discover two bits of knowledge the same, then bits not equal at one time become equal at a later time and the transition arrow is not injective.

For example, consider our knowledge of the set of complex numbers. There was a time in our history when all the numbers e, i, π and -1 were all known, but it was not known that $e^{i\pi} = -1$. In that case, the number $e^{i\pi}$ and -1 were known separately, but not the fact that they were equal. See [Barr, McLarty and Wells, 1985]. The sheaf condition is this: if $\{t_i\}$ is a set of times and t is their infimum, then anything known at time t_i for every i is known at time t.

15.5.11 Exercise

1. Let $\mathscr{H}$ be a complete Heyting algebra. Define the binary operation $\Rightarrow$: $Hsc \times Hsc \longrightarrow Hsc$ by requiring that $a \Rightarrow b$ is the join of all elements c for which $a \wedge c \leq b$.

a. Prove that $a \wedge c \leq b$ if and only if $c \leq a \Rightarrow b$.

b. Prove that when $\mathscr{H}$ is regarded as a category in the usual way, it is cartesian closed with $\Rightarrow$ as internal hom.

15.6 Fuzzy sets

Fuzzy set theory is a more-or-less categorical idea that some claim has application to computer modeling. It appears to be closely related to topos theory. In fact, it appears to us that the interesting core of the subject is already implicit in topos theory. We will not go deeply into the subject, but only give a few definitions and point out the connections. More details can be found in [Barr, 1986a], [Pitts, 1982].

15.6.1 At this point, it would be appropriate to give some definitions. One of the problems with fuzzy sets is that the meaning of the term has been left vague (one might say fuzzy). Rather than attempt to give all possible definitions, we content ourselves with a definition that is common and for which the connection with topos theory is as simple as possible.

15.6.2 Definition Let P be a complete Heyting algebra. A **P-valued set** is a pair (S, σ) consisting of a set S and a function $\sigma : S \longrightarrow P$. A **category of fuzzy sets** is the category of P-valued sets (in other words, the slice category $\mathbf{Set}/P$) for a fixed complete Heyting algebra P.

In practice, fuzzy sets are defined with P being the closed interval of real numbers from 0 to 1, which is a complete Heyting algebra with the usual ordering. Think of $\sigma(s)$ as being the degree of membership of s in the fuzzy

set. If $\sigma(s) = 1$, then s is fully in the fuzzy set, while if $\sigma(s) = 0$, then s is not in the fuzzy set at all.

Actually that last statement is not quite true; we will return to this point later; pretend for the moment that it is.

15.6.3 Let (S, σ) and (T, τ) be fuzzy sets. An **arrow** $f : (S, \sigma) \longrightarrow (T, \tau)$ is an arrow $f : S \longrightarrow T$ such that $\sigma \leq \tau \circ f$. Thus the degree of membership of s in (S, σ) cannot exceed that of $f(s)$ in (T, τ). With this definition and the obvious identity arrows, the fuzzy sets based on P form a category $\mathbf{Fuzz}(P)$.

The hypothesis actually made on P was that both P and the opposite order P^{op} were Heyting algebras ([Goguen, 1974]). The hypothesis on P^{op} plays no role in the theory and so we have omitted it.

15.6.4 Once we have defined the category of fuzzy sets, the definition of subset of a fuzzy set emerges. For $f : (S, \sigma) \rightarrowtail (T, \tau)$ to be monic it is necessary that f be injective. In particular, we can think of a subset of (T, τ) as being a fuzzy set (T_0, τ_0) where $T_0 \subseteq T$ and $\tau | T_0 \leq \tau_0$.

15.6.5 More ado about nothing Consider the following two fuzzy subsets of (S, σ). The first is the set $(\emptyset, \langle\rangle)$ and the second is the set $(S, 0)$ where $\langle\rangle$ is the unique function of $\emptyset$ to P and 0 stands for the function that is constantly zero. One is the empty set and the other is the set in which every element is not there. There is seemingly no difference between these two sets as neither actually contains any elements. In fact, in fuzzy set theory, these two sets (and sets in between) are not considered to be equal. This results in the class of fuzzy set theories being curiously restricted (see 15.6.10).

15.6.6 Fuzzy sets and sheaves The reader may suspect (from the title of this section, if nothing else) that there is a connection between fuzzy sets and toposes. Both are generalizations of set theory to introduce lattices more general than the two-element lattice as truth values.

One of the two differences has just been mentioned; the different treatment of the null set. Actually, this difference is relatively minor. The second one is not. Suppose (S, σ) is a fuzzy set. We can define a presheaf E by letting

$$E(x) = \{s \in S \mid \sigma(s) \geq x\}$$

as suggested in our informal discussion. Clearly, if $y \leq x$, then $E(x) \subseteq E(y)$ and using these inclusions, we get a presheaf on P. It is almost never a sheaf, however. The essential reason for this is that $E(0) = S$, while we have seen in 15.5.6 that $E(0) = 1$ when E is a sheaf.

15.6.7 It turns out there is a very simple way to make E into a sheaf, but not on P. Let P^+ denote the poset constructed from P by adding a new bottom element. Let us call the new bottom element $\perp$ to distinguish it from the old one we called 0. Now given a P indexed fuzzy set, define a presheaf on P^+ by letting $E(x)$ be defined as above for $x \in P$ and $E(\perp) = 1$.

15.6.8 Proposition *The presheaf E just defined is a sheaf. It is a subsheaf of the near constant sheaf C defined by $C(x) = S$ for $x \neq \perp$ and $C(\perp) = 1$.*

Proof. We first observe that P^+ obviously has the property that the meet of two nonzero elements is nonzero because P has finite meets. Thus C is a sheaf. A diagram chase shows that if C is a sheaf and E a subpresheaf, then E is a sheaf if and only if for each $x = \bigvee x_i$, the diagram

$$\begin{array}{ccc} E(x) & \longrightarrow & \prod E(x_i) \\ \downarrow & & \downarrow \\ C(x) & \longrightarrow & \prod C(x_i) \end{array}$$

is a pullback. The vertical arrows are just the inclusions. As we saw in 15.5.7, the lower horizontal function is just the inclusion of $C(x)$ into the set of constant strings. It follows that this is essentially what the upper horizontal arrow is. Now in order that a string of elements $\{s_i\} \in \prod E(x_i)$ be constant, it is necessary and sufficient that all the s_i be the same element s and that $s \in E(x_i)$ for all i which means that $\sigma(s) \geq x_i$ for all i. But this is just what is required to have $\sigma(s) \geq x$ and $s \in E(x)$. □

Continuing in this vein, it is possible to show the following.

15.6.9 Theorem *For any Heyting algebra P, the category of fuzzy sets based on P is equivalent to the full subcategory of the category of P^+ sheaves consisting of the sheaves that are subsheaves of the near constant sheaves.*

15.6.10 The introduction of P^+ instead of P is directly traceable to the failure the two kinds of empty sets as mentioned in 15.6.5 to be the same. The fact that the sheaves are subsheaves of the near constant sheaves is really a reflection of the fact that in fuzzy set theory only one of the two predicates of set theory is made to take values in P (or P^+).

This shows up in the fact that in fuzzy set theory there is no fuzzy set of fuzzy subsets of a fuzzy set. In other words, the $\mathscr{P}$ construction is missing. Here's why. Suppose S is a set, considered as a fuzzy set with $\sigma(s) = 1$ for all $s \in S$. (Such a fuzzy set is called a **crisp** set.) Let $x < y$ be two elements

of P and consider the subsets $S_x = (S, \sigma_x)$ and $S_y = (S, \sigma_y)$, with σ_x and σ_y being the functions which are constant at x and y respectively. Then of course, $S_x \neq S_y$ (actually S_x is a proper subset of S_y), but it is clear that when looking only at degrees of membership at level x or below, the two subsets are equal. In fact, in the topos, the degree to which S_x equals S_y is just x. But this predicate cannot be stated in the language of fuzzy sets and the result is that there are not and cannot be power objects (unless P has just one element).

The point is that there are two predicates in set theory, membership and equality. In topos theory, both may be fuzzy, but in fuzzy set theory, only membership is allowed to be. But $\mathscr{P}$ converts membership into equality as explained in the preceding paragraph and so cannot be defined in fuzzy set theory. Thus fuzzy set theory, as currently implemented, lacks a certain conceptual consistency.

One can try to refine the definition of fuzzy set so as to allow fuzzy equality. The obvious way to proceed is to define as objects triplets (S, σ, η), with (S, σ) as above and $\eta : S \times S \to P$, interpreted as fuzzy equality. These must be subject to the condition that the degree to which two elements are equal cannot exceed the degree to which either one is defined. The resultant category is equivalent to the topos of sheaves on P^+.

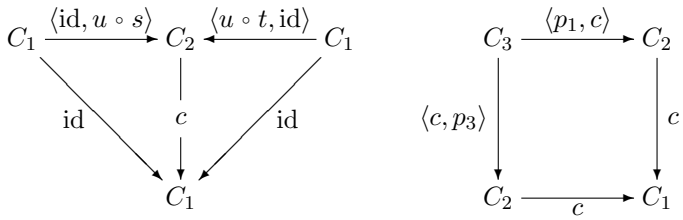
15.7 External functors

15.7.1 Category objects in a topos One of the tools proposed for programming language semantics is the category of modest sets, which we will describe in the next section. The category of modest sets is not a category in the sense we have been using the word up until now: it is a category *object* in another category, called the effective topos.

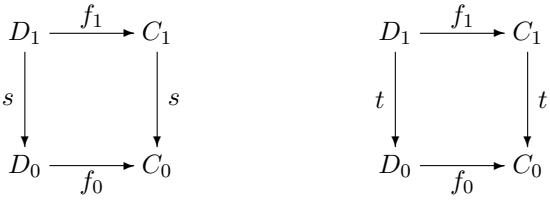
Recall the sketch for categories that was described in detail in 10.1.5. A model of this sketch in the category of sets is, of course, a category. A model in a category $\mathscr{C}$ is called a **category object** in $\mathscr{C}$. A homomorphism between such category objects is called an **internal functor** between those category objects.

Referring to the sketch, we see that a category object consists of four objects C_0, C_1, C_2 and C_3 such that $C_2 \cong C_1 \times_{C_0} C_1$ and $C_3 \cong C_1 \times_{C_0} C_1 \times_{C_0} C_1$. There are arrows in $\mathscr{C}$ corresponding to unit, source, target and

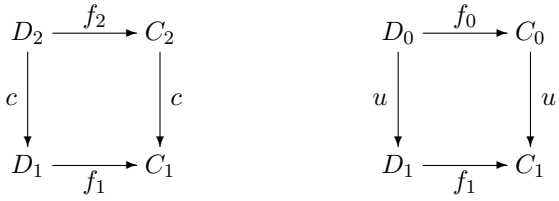
composition. The crucial commutative diagrams are:



We may denote such a category object by $\mathbf{C} = \langle C_0, C_1, u_{\mathbf{C}}, s_{\mathbf{C}}, t_{\mathbf{C}}, c_{\mathbf{C}} \rangle$ since the remaining data are determined by these. As a matter of convenience, we usually omit the indexes unless they are necessary for comprehension. If $\mathbf{D} = \langle D_0, D_1, u, s, t, c \rangle$ is another category object, then an internal functor $f : \mathbf{D} \longrightarrow \mathbf{C}$ is given by homomorphisms $f_0 : D_0 \longrightarrow C_0$ and $f_1 : D_1 \longrightarrow C_1$ such that the following diagrams commute, where $f_2 : D_2 \longrightarrow C_2$ is the unique arrow for which $p_i \circ f_2 = f_1 \circ p_i, \quad i = 1, 2$.



(15.3)



15.7.2 External functors The notion of a functor from a category into the category of sets can be extended to describe a functor from a category in $\mathscr{E}$ to $\mathscr{E}$ itself. Such a functor is called an **external functor**. This construction is based on the construction in Section 12.2 for **Set**. Theorem 12.3.7 gives an equivalence of categories between external functors and the split opfibrations which are produced by the Grothendieck construction, and so justifies the representation by split opfibrations of external functors defined on a category object.

Essentially, what we will do is use objects and arrows representing sets and functions involved in the Grothendieck construction, find enough cones

and commutative diagrams to characterize it, and take that as the definition of external functor in an arbitrary category.

15.7.3 Let $\mathscr{E}$ be a category, and let

$$\mathbf{C} = (C_0, C_1, \text{source}, \text{target}, \text{unit}, \text{comp})$$

be a category object in $\mathscr{E}$. An external functor $\mathbf{C} \longrightarrow \mathscr{E}$ consists of data

$$(D_0, D_1, D_2, d^0, d^1, u, \pi_0, \pi_1, p_1, p_2, c)$$

for which the D_i are objects of $\mathscr{E}$ and the arrows have sources and targets as indicated:

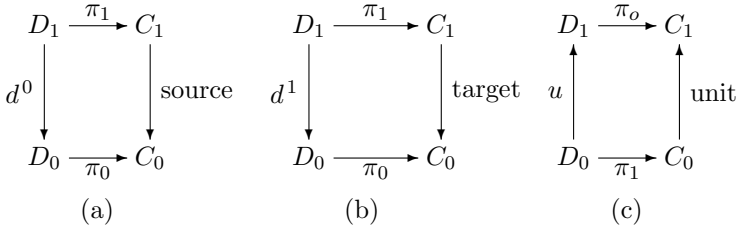
$$\begin{aligned} d^i &: D_1 \longrightarrow D_0, \quad i = 1, 2 \\ u &: D_0 \longrightarrow D_1 \\ \pi_i &: D_i \longrightarrow C_i, \quad i = 0, 1 \\ p_i &: D_2 \longrightarrow D_1, \quad i = 1, 2 \\ c &: D_2 \longrightarrow D_1 \end{aligned}$$

D_0 is the object corresponding to the disjoint union of the values of the functor. Note that we are not *given* a functor F as we were in Section 12.2 – we are being guided by Theorem 12.3.7 and the details of the Grothendieck construction to *define* an external functor, and π_0 represents the projection (taking (x, C) to C in the case of the Grothendieck construction). D_1 is the object corresponding to the arrows of the category $\mathbf{G}(C, F)$ as defined in Section 12.2, and π_1 takes (x, f) to f. Thus π_0 and π_1 together describe the functor $\mathbf{G}(F)$ in the case of the Grothendieck construction.

d^0 and d^1 are the source and target maps of that category, c is the composition and u picks out the identities.

The data are subject to the requirements E–1 through E–4 below.

E–1 All three diagrams below must commute and (a) must be a pullback:

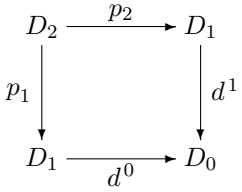


(15.4)

That (a) is a pullback says in the case of the Grothendieck construction that, up to unique isomorphism, D_1 consists of elements of the form (x, f) with f an arrow of $\mathscr{E}$ and $x \in F(C)$ where C is the source of f. This is part of

GS–2 (see Section 12.2). The commutation of (b) says that $d_1(x, f) \in F(C')$ where C' is the target of F; that follows from GS–1 and GS–2 (there, x' must be in $F(C')$). That of (c) says that $u(x, C)$ must be (id_C). In the case of the Grothendieck construction that follows from the fact that F is given as a functor.

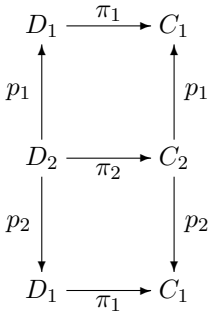
E–2 The following diagram is a pullback:



(d)

In the case of the Grothendieck construction this forces D_2 to be the set of composable pairs of arrows of $\mathbf{G}(C, F)$.

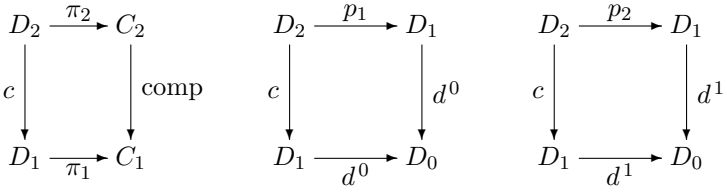
E–3 The following diagram must commute:



(e)

π_2 is defined by Diagram (15.3) (called f_2 there). In the case of the Grothendieck construction this follows from GS–3: (x, g') composes with (x, f) only if g composes with f (but not conversely!).

E–4 The following diagrams must commute.



(f) (g) (h) (15.5)

In the case of the Grothendieck construction, (f) says that π_1 preserves composition and (g) and (h) say that the composite has the correct source and target.

As we have presented these diagrams, we have observed that they are all true in the case of the Grothendieck construction in **Set**. In the case of the Grothendieck construction,

$$(D_0, D_1, \text{source}, \text{target}, \text{unit}, \text{comp})$$

is actually a category, hence a category object in **Set**. Moreover, $\pi = (\pi_0, \pi_1)$ is a functor, namely $\mathbf{G}(F)$. That is actually true in any category, as seen in the following.

15.7.4 Proposition *Let*

$$(D_0, D_1, D_2, d^0, d^1, u, \pi_0, \pi_1, p_1, p_2, c)$$

be an external functor to a category object

$$\mathscr{C} = (C_0, C_1, \text{source}, \text{target}, \text{unit}, \text{comp})$$

in a category $\mathscr{E}$. Then

$$(D_0, D_1, \text{source}, \text{target}, \text{unit}, \text{comp})$$

is a category object in $\mathscr{E}$ and (π_0, π_1) is a functor in $\mathscr{E}$.

The proof is a lengthy series of diagram chases.

The converse is true, too: up to some technicalities, an external functor in **Set** arises from a functor to **Set** from a small category. That means we said enough about it in E–1 through E–4 to characterize it.

15.7.5 Theorem *Let*

$$(\mathscr{D}_0, \mathscr{D}_1, \mathscr{D}_2, d^0, d^1, u, \pi_0, \pi_1, p_1, p_2, c)$$

be an external functor to a category

$$\mathscr{C} = (\mathscr{C}_0, \mathscr{C}_1, \text{source}, \text{target}, \text{unit}, \text{comp})$$

in **Set**. *Define $F : \mathscr{C} \longrightarrow$ **Set** by*

F–1 *If C is an object of $\mathscr{C}$, then $F(C) = \pi_1^{-1}(C)$.*

F–2 *If $f : C \longrightarrow C'$ is an arrow of $\mathscr{C}$ and $x \in F(C)$, then $F(f)(x)$ is the target of the unique arrow α of $\mathscr{D}$ for which $\pi_1(\alpha) = f$ and $d^0(\alpha) = x$.*

*Then $F : \mathscr{C} \longrightarrow$ **Set** is a functor, and there is a unique isomorphism $\beta : \mathscr{D} \longrightarrow \mathbf{G}(\mathscr{C}, F)$ for which $\mathbf{G}(F) \circ \beta = \pi$.*

This theorem is essentially the discrete case of Theorem 12.3.7.

F–2 makes use of the fact that Diagram (15.4)(a) is a pullback. A seasoned categorist will simply *name* $\alpha \in F^2$ as (x, f), since he knows that between any two pullbacks defined by (15.4)(a) there is a unique isomorphism respecting the projections π_1 and d^0.

15.8 The realizability topos

The category of modest sets has been proposed as a suitable model for the polymorphic λ-calculus. It is a subcategory of a specific topos, the realizability topos. Space limitations prevent us from giving a full exposition of this topic, Here we describe how the realizability topos and the subcategory of modest sets are constructed. Further references are [Carboni, Freyd and Scedrov, 1988], [Rosolini, 1990], [Gray, 1991].

15.8.1 Realizability sets In the discussion below, we will be writing $f(x)$ where f is a partial function. It will be understood that $f(x)$ is defined when we write this.

A **realizability set** is a pair $\mathbf{A} = (A, =_{\mathbf{A}})$ where A is a set called the **carrier** of $\mathbf{A}$ and $=_{\mathbf{A}}$: $A \times A \longrightarrow \mathscr{P}(N)$ is a function to sets of natural numbers (thought of as reasons that two elements are equal; in fact, they can be thought of as the set of Gödel numbers of proofs that they are). We denote the value of this function at $(a_1, a_2) \in A \times A$ by $[\![a_1 =_{\mathbf{A}} a_2]\!]$, often abbreviated to $[\![a_1 = a_2]\!]$. This is subject to the following conditions:

REAL–1 There is a partial recursive function f such that for any a_1, $a_2 \in A$ and $n \in [\![a_1 = a_2]\!]$, $f(n) \in [\![a_2 = a_1]\!]$.

REAL–2 There is a partial recursive function of two variables g with the property that if $n \in [\![a_1 = a_2]\!]$ and $m \in [\![a_2 = a_3]\!]$, then $g(n, m) \in [\![a_1 = a_3]\!]$.

This is made into a category by defining an arrow from $\mathbf{A} = (A, =_{\mathbf{A}})$ to $\mathbf{B} = (B, =_{\mathbf{B}})$ to be an equivalence class of partial functions $\phi : A \longrightarrow B$ such that there is a partial recursive function f such that $n \in [\![a_1 = a_2]\!]$ implies that $f(n) \in [\![\phi a_1 = \phi a_2]\!]$. Two such arrows ϕ and ψ are equal if there is a partial recursive f such that $n \in [\![a = a]\!]$ implies that $f(n) \in [\![\phi a = \psi a]\!]$.

Note that the last definition implies that f is defined on all elements $a \in A$ for which $[\![a = a]\!] \neq \emptyset$. The other elements of A are irrelevant.

15.8.2 This category is a topos. We will not prove this, but simply describe some of the constructions required. To find products, for example, first choose a recursive bijection $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$. If $\mathbf{A} = (A, =_{\mathbf{A}})$ and $\mathbf{B} = (B, =_{\mathbf{B}})$ are realizability sets, the product is $(A \times B, =_{\mathbf{A}} \times =_{\mathbf{B}})$, where the latter is defined by

$$[\![(a_1, b_1) = (a_2, b_2)]\!] = \{f(n, m) \mid n \in [\![a_1 = a_2]\!], m \in [\![b_1 = b_2]\!]\}$$

Notice that a choice of f is required to show that products exist. The products thereby exist, but, owing to the uniqueness of categorical products, do not depend in any way on the choice of the function.

Equalizers can be defined as follows. If $\phi, \psi : \mathbf{A} \to \mathbf{B}$ are two arrows their equalizer is given by $(A, =_{\phi,\psi})$ where $[\![a_1 =_{\phi,\psi} a_2]\!] = [\![\phi a_1 = \psi a_2]\!]$.

Finally power objects can be constructed as follows. Let f denote a pairing as above and also choose an enumeration g_e of all the partial recursive functions. The carrier of $\mathscr{P}\mathbf{A}$ is $[A \to \mathscr{P}\mathbf{N}]$, all functions from A to sets of natural numbers. If P and Q are two such functions, then we will say that $f(d, e) \in [\![P = Q]\!]$ if for all $n \in P(a)$ and $m \in [\![a = b]\!]$, we have $g_d(f(n, m)) \in Q(b)$ and for all $n \in Q(a)$ and $m \in [\![a = b]\!]$, we have $g_e(f(n, m)) \in P(b)$.

This topos is called the **realizability** or **effective** topos. It is due to Martin Hyland [1982], although the basic idea goes back to S. Kleene. See also [Rosolini, 1987].

15.8.3 Modest sets 'Modest sets' is used more-or-less interchangeably to denote either a certain full subcategory of the category of realizability sets or an internal category object of that topos. A great deal of effort has been put into describing the connection between the two. We will begin with the former. The category of modest sets can be described directly and then embedded into the realizability topos.

A **modest set** consists of $\mathbf{A} = (\mathbf{N}, =_{\mathbf{A}})$ where as usual $\mathbf{N}$ denotes the natural numbers and $=_{\mathbf{A}}$ is a **partial equivalence relation** or **PER** on $\mathbf{N}$, which means it is symmetric and transitive, but not necessarily reflexive. We think of $\mathbf{A}$ as a quotient of a subobject of $\mathbf{N}$; the subobject is the set of n for which $n =_{\mathbf{A}} n$ modulo the relation $=_{\mathbf{A}}$.

The category of modest sets has arrows from $\mathbf{A}$ to $\mathbf{B}$ defined as partial recursive functions f defined on all n such that $n =_{\mathbf{A}} n$ and such that $n =_{\mathbf{A}} m$ implies that $f(n) =_{\mathbf{A}} f(m)$. Note that since the relation is symmetric and transitive, as soon as there is some m with $n =_{\mathbf{A}} m$, it is also the case that $n =_{\mathbf{A}} n$ and so $f(n)$ and $f(m)$ are defined.

The modest sets form a cartesian closed category. Choose, as above, an enumeration g_e of the partial recursive functions. Then $[\mathbf{A} \to \mathbf{B}]$ is the PER

with relation given by $d =_{[\mathbf{A}\to\mathbf{B}]} e$ if and only if whenever $n =_{\mathbf{A}} m$ then $g_d(n) =_{\mathbf{B}} g_e(m)$. The modest sets do not form a topos.

We embed the modest sets into the realizability sets by choosing a pairing f and associating to the modest set $\mathbf{A} = (\mathbf{N}, =_{\mathbf{A}})$ the realizability set with carrier $\mathbf{N}$ and $e \in [\![n = m]\!]$ if and only if $n =_{\mathbf{A}} m$ and $e = f(n, m)$. Although this appears to depend on the choice of a pairing, it is easy to see that up to isomorphism it does not.

In fact it cannot. The pairing is used only to show that products *exist*. But their properties – in particular their isomorphism class – are independent of the particular construction used to prove their existence. Similar remarks apply to the cartesian closed structure, which does not depend on a particular enumeration of the partial recursive functions.

15.8.4 The internal category of modest sets This is a very brief glance at how one might internalize the description of modest sets to produce a category object inside the category of realizability sets. Except by way of motivation, this category object has no real connection with the category of modest sets. Nevertheless, we will call it the **internal category of modest sets**.

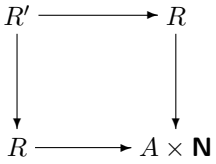
This internal category has the remarkable property that *every* internal diagram in it has a limit. This is not possible for ordinary categories (unless they are just posets), but the proof that it is not possible requires a property of set theory which is not valid for the realizability sets (namely the axiom of choice).

The construction is an exercise in internal expression. A modest set is a relation on $\mathbf{N}$ with certain properties. The natural numbers object in the category of realizability sets is the object $(\mathbf{N}, =_{\mathbf{N}})$, where $e \in [\![n =_{\mathbf{N}} m]\!]$ if and only if $n = m = e$. That is $[\![n =_{\mathbf{N}} m]\!] = \{n\}$ if $n = m$ and is empty otherwise. We will denote it by $\mathbf{N}$. Then a modest set is a subset of $\mathbf{N} \times \mathbf{N}$ with certain properties.

The set of objects of the internal category of modest sets is then a certain set of subsets of $\mathbf{N} \times \mathbf{N}$, which is the same as a subset of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$. We want to describe this subset as consisting of those relations that are symmetric and transitive and double negation closed. The trick is to define endoarrows s and t of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$ that associate to a relation R the relations R^{op} and $R \circ R$, respectively. For s, this is easy. There is an arrow $\langle p_2, p_1\rangle : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \times \mathbf{N}$ that switches the coordinates and we let $s = \mathscr{P}(\langle p_2, p_1\rangle)$.

For t, it is a little harder. By Yoneda, an arrow $\mathscr{P}(\mathbf{N} \times \mathbf{N})$ to itself can be defined by describing a natural transformation $\mathrm{Hom}(-, \mathscr{P}(\mathbf{N} \times \mathbf{N}))$ to itself. An element of $\mathrm{Hom}(A, \mathscr{P}(\mathbf{N} \times \mathbf{N}))$ is, by the defining property of $\mathscr{P}$, a subobject $R \subseteq A \times \mathbf{N} \times \mathbf{N}$. Given such an R, we define R' to be the pullback

in the following diagram:



The two maps from $R \longrightarrow A \times \mathbf{N}$ are the inclusion into $A \times \mathbf{N} \times \mathbf{N}$ followed by $\langle p_1, p_2 \rangle$ and $\langle p_1, p_3 \rangle$, respectively. You should think of R as being a set of triples (a, n_1, n_2) and then $R' \subseteq A \times \mathbf{N} \times \mathbf{N} \times \mathbf{N}$ is the set of 4-tuples (a, n_1, n_2, n_3) such that $(a, n_1, n_2) \in R$ and $(a, n_2, n_3) \in R$. Then the inclusion of R' into $A \times \mathbf{N} \times \mathbf{N} \times \mathbf{N}$ followed by $\langle p_1, p_2, p_4 \rangle$ gives us an arrow $R' \longrightarrow A \times \mathbf{N} \times \mathbf{N}$ whose image we denote by $t(R)$. It is interpreted as the set of (a, n_1, n_3) such that there is an n_2 with both $(a, n_1, n_2) \in R$ and $(a, n_2, n_3) \in R$. We omit the proof that this construction from R to $t(R)$ is natural in A, but assuming that, it follows from the Yoneda Lemma that this results from an endomorphism t of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$.

Finally, the object M_0 of objects is defined as the limit of

$$\mathscr{P}(\mathbf{N} \times \mathbf{N}) \underset{t}{\overset{s}{\rightrightarrows}} \mathscr{P}(\mathbf{N} \times \mathbf{N})$$

where the middle arrow is the identity. This is to be interpreted as the set of relations that are fixed under both s and t.

The object M_0 is the object of objects of the internal category of modest sets. The object of arrows is defined as a subobject of the object $[\mathbf{N} \longrightarrow \mathbf{N}]^{\sim}$ of partial arrows of $\mathbf{N}$ to $\mathbf{N}$; namely those that satisfy the internal version of the definition of the ordinary of modest sets. This construction is tedious, but not difficult, and we omit it.

16

Categories with monoidal structure

Many of the categories that arise in computer science (and elsewhere) have a binary operation on objects and arrows. This operation is rarely actually associative, but is usually assumed associative up to natural isomorphism and subject to certain **coherence** laws as we will explain. We also suppose there is a unit object, which is not actually a unit, but is so up to a natural coherent isomorphism. Since the associativity and existence of a unit characterize monoids, these categories are called monoidal.

This chapter may be read immediately after Chapter 13.

16.1 Closed monoidal categories

16.1.1 Definition A **monoidal** category is a category $\mathscr{C}$ equipped with an object $\top$ and a functor $\otimes : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$. We use infix notation so that the value at (A, B) is written $A \otimes B$. A monoidal category must have, in addition, the natural isomorphisms a, r and l for all objects A, B and C listed in MC–1 through MC–3 below, and these must make Diagrams MC–5 and MC–6 commute. The category is a **symmetric monoidal category** if in addition it has the natural isomorphism SMC–4 and the diagrams SMC–7 through SMC–9 commute.

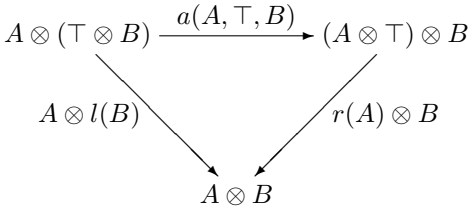
MC–1 $a(A, B, C) : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$;

MC–2 $rA : A \otimes \top \to A$;

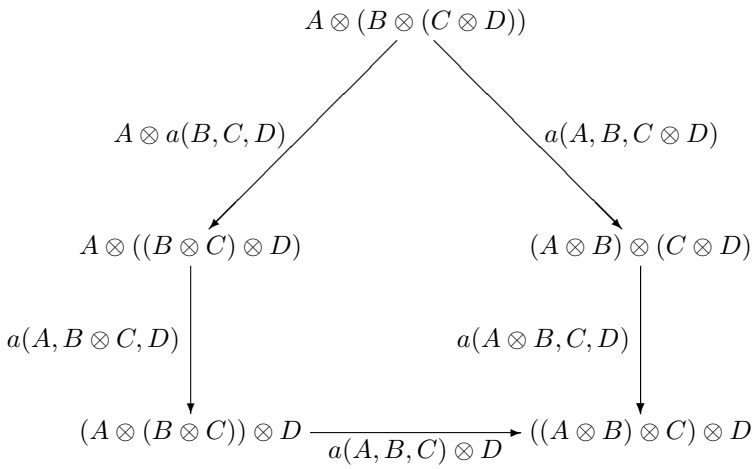
MC–3 $lA : \top \otimes A \to A$.

SMC–4 $s(A, B) : A \otimes B \to B \otimes A$.

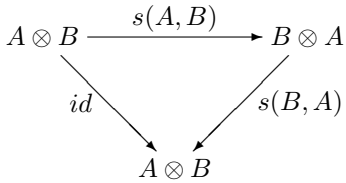
MC–5



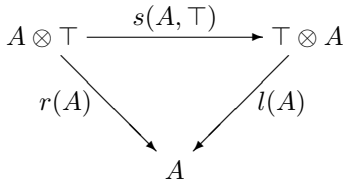
MC–6



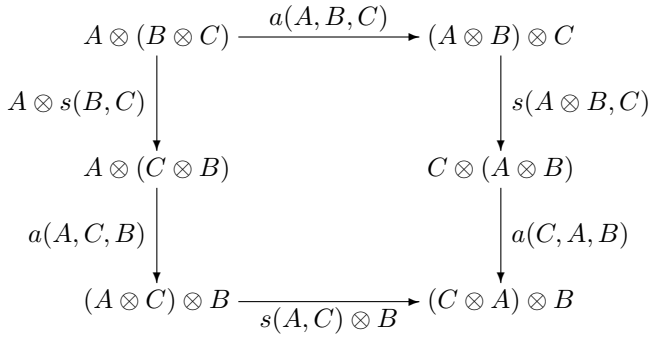
SMC–7



SMC–8



SMC–9



Each of the diagrams listed above asserts the commutativity of a diagram formed by composing isomorphisms built from instances of $\otimes$, a, r and l (and s in the symmetric case). These diagrams are chosen to be the axioms because assuming that they commute implies that *any* two parallel isomorphisms, provided that they are constructed from $\otimes$, a, r and l and s using $\otimes$ and composition, are the same. A precise statement of this fact is in Chapter VII.2 of [Mac Lane, 1971].

In any monoidal category $\mathscr{C}$, there is a functor $A \otimes - : \mathscr{C} \to \mathscr{C}$ that takes B to $A \otimes B$ and $f : B \to C$ to $A \otimes f : A \otimes B \to A \otimes C$.

16.1.2 Definition A monoidal category $\mathscr{C}$ is said to be **closed** if for each object A of $\mathscr{C}$, the functor $A \otimes -$ has a right adjoint.

If we denote the value at C of the right adjoint by $A \multimap C$ (A appears as a parameter since this is defined for each object A), then the defining condition is an isomorphism, natural in B and C,

$$\mathrm{Hom}(A \otimes B, C) \xrightarrow{\cong} \mathrm{Hom}(B, A \multimap C) \tag{16.1}$$

In the symmetric case, it follows that

$$\mathrm{Hom}(A \otimes B, C) \cong \mathrm{Hom}(A, B \multimap C)$$

In general, the latter isomorphism fails in the non-symmetric case; there we may want to suppose that there is, for each B a functor $C \mapsto C \circ\!\!- B$ such that $\mathrm{Hom}(A \otimes B, C) \cong \mathrm{Hom}(A, C \circ\!\!- B)$. We will not explore this point further, but will suppose from now on that our monoidal structure is symmetric.

16.1.3 Example A category that has finite products is automatically a symmetric monoidal category, using the categorical product for $\otimes$ and the terminal object for $\top$. The various isomorphisms can be constructed using the universal mapping property of products. For example, $a(A, B, C) : A \times (B \times C) \to (A \times B) \times C$ is the map that is denoted $\langle\langle \mathrm{proj}_1, \mathrm{proj}_1 \circ \mathrm{proj}_2\rangle, \mathrm{proj}_2 \circ \mathrm{proj}_2\rangle$ (see 5.2.9 for an explanation of this notation). The coherence is also automatic, essentially because of the uniqueness of the universal mappings. In Diagram MC–6, for example, both paths give the map

$$\langle\langle\langle p_1, p_1 \circ p_2\rangle, p_1 \circ p_2 \circ p_2\rangle, p_2 \circ p_2 \circ p_2\rangle : A \times (B \times (C \times D)) \\ \to ((A \times B) \times C) \times D$$

where we have written p instead of proj.

Cartesian closed categories are, of course, just monoidal closed categories in which the product is the cartesian product. Thus any cartesian closed category is monoidal closed, but there are many others.

A category with finite sums is also monoidal, using the sum for $\otimes$ and the initial object for $\top$. The verification is essentially the same as for the case of products. **Set** (and many other categories with both products and sums) is therefore an example of a category with two inequivalent monoidal structures on it.

16.1.4 Example In 2.1.14, we constructed the category **Rel** of relations, whose objects are sets S, T, ..., and in which an arrow $S \longrightarrow T$ is a subset $\alpha \subseteq S \times T$. The cartesian product of sets is not the product in this category – the disjoint sum is both sum and product – but it is a functor of two variables, that is a monoidal structure on **Rel**. We will denote it $S \otimes T$ to avoid confusion with the categorical product. An arrow $S \otimes T \longrightarrow U$ in this category is a subset of $(S \otimes T) \otimes U \cong T \otimes (S \otimes U)$. Thus $\mathrm{Hom}(S \otimes T, U) \cong \mathrm{Hom}(T, S \otimes U)$. Thus **Rel** is a closed monoidal category with $S \multimap T = S \otimes T$.

Observe that this is an example of a closed monoidal category in which $A \multimap B$ is not a structure built on $\mathrm{Hom}(A, B)$: $A \multimap B = A \otimes B$ is the cartesian product (in **Set**, not **Rel**!) of A and B, whereas $\mathrm{Hom}(A, B)$ is the powerset of that cartesian product.

We return to this example in Example 16.3.3.

16.1.5 Subsets of a monoid Here is an example that is a poset viewed as a category. Let M be a monoid and $\mathscr{C}$ denote the category whose objects are the subsets of M with inclusions as the only morphisms. If A and B are subsets of M, let $A \otimes B = \{ab \mid a \in A \text{ and } b \in B\}$. Define $A \multimap C = \{b \in M \mid ab \in C \text{ for all } a \in A\}$. Then it is easy to verify that $A \otimes B \subseteq C$ if and only if $B \subseteq A \multimap C$. If we define $C \circ\!\!- B = \{a \in M \mid ab \in C \text{ for all } b \in B\}$, it is equally easy to see that $A \otimes B \subseteq C$ if and only if $A \subseteq C \circ\!\!- B$.

16.1.6 Sup semilattices Another example is the category of sup semilattices, (see 5.4.6), which also illustrates another principle, that it is often much easier to give an explicit description of the $\multimap$ operation than of the $\otimes$. If K and L are sup semilattices, then $K \multimap L$ is defined to be the set of all sup-semilattice homomorphisms from K to L. If f and g are such homomorphisms, then their sup is defined by $(f \vee g)(x) = f(x) \vee g(x)$. The bottom element is the constant function at the bottom element of L. It is easily seen that with these definitions $K \multimap L$ is a sup semilattice.

To get the monoidal structure, we will construct the left adjoint to the functor $K \multimap -$. The existence of the left adjoint also follows from the adjoint functor theorem (references to this are given after Theorem 13.3.7).

First, form the free sup semilattice generated by $K \times L$. It has elements of the form $(x_1, y_1) \vee (x_2, y_2) \vee \cdots \vee (x_n, y_n)$ with $x_1, x_2, \ldots, x_n \in K$ and $y_1, y_2, \ldots, y_n \in L$. These specifically include the empty sup 0. We then form the quotient semilattice gotten by all identifications of the form $(x_1, y) \vee (x_2, y) = (x_1 \vee x_2, y)$ and $(x, y_1) \vee (x, y_2) = (x, y_1 \vee y_2)$. In addition, we must identify the bottom element of the semilattice, which is the empty sup, with $(x, 0)$ for any $x \in K$ and with $(0, y)$ for any $y \in L$ These identifications have to be made compatible with the semilattice structure so that whenever two elements are identified, their joins with any third element are also identified.

It is instructive to see why any $(x, 0)$ has to be identified with 0. First, let us denote the image of (x, y) in $K \otimes L$ by $x \otimes y$. Not every element of $K \otimes L$ has the form $x \otimes y$, but it is generated by such elements. The isomorphism $\mathrm{Hom}(K \otimes L, M) \cong \mathrm{Hom}(L, K \multimap M)$ can be described as follows. If $f : K \otimes L \longrightarrow M$ corresponds to $\widetilde{f} : L \longrightarrow K \multimap M$, then for $y \in L$, $\widetilde{f}(y) : K \longrightarrow M$ is defined by $\widetilde{f}(y)(x) = f(x \otimes y)$. In particular, $\widetilde{f}(0)$ is required to be the 0 homomorphism, in order to be a homomorphism of sup semilattices. Thus $f(x \otimes 0) = \widetilde{f}(0)(x) = 0$. In particular, take $M = K \otimes L$ and choose $f = \mathrm{id}$ to conclude that $x \otimes 0 = 0$. A similar argument will show that $0 \otimes y = 0$ as well, but in fact, it is not hard to show symmetry of $\otimes$ directly.

> The crucial property of sup semilattices exploited here is that sup semilattices are determined by operations, $\vee$ and 0 that are homomorphisms of the sup semilattice structure. Thus when L is a sup semilattice, $\vee : L \times L \longrightarrow L$ is not just a function but a semilattice homomorphism. It is fairly uncommon for an equational theory to have the property that its operations are homomorphisms of the theory, but when it does, constructions analogous to the above will always give a monoidal closed structure.

Many other examples of closed monoidal categories are given in [Mac Lane, 1971]. Applications are discussed in [Asperti and Longo, 1991], [Corradini and Asperti, 1993], [Marti Oliet and Meseguer, 1991] and [Bartha, 1992].

16.2 Properties of $A \multimap C$

Monoidal closed categories exemplify one of the earliest perceptions of category theory, that a category is quite special if the set of arrows between any two objects is, in some natural way, also an object of the category. Thus, although it is much commoner to have some kind (or many kinds) of monoidal structure on a category than to have a closed structure, it is usually easier to define the closed structure, when it exists. This is well illustrated in the example of semilattices given in 16.1.6.

16.2.1 The internal hom Another aspect of this perception is that $A \multimap C$ is called the **internal hom** of A to C and indeed it has properties similar to those of the 'external' (that is the set-valued) hom functor $\mathrm{Hom}(A, C)$. For example, for each C, $A \mapsto A \multimap C$ is also functorial in A, albeit contravariantly. (For each A, $C \mapsto A \multimap C$ is a functor in C by definition.) This means that there is a natural way of defining, for each map $f : A' \longrightarrow A$ a map $f \multimap C : A \multimap C \longrightarrow A' \multimap C$ in such a way that the isomorphism (16.1) is natural in all three arguments. This generalizes Proposition 6.2.1.

Another aspect is illustrated by the following, which internalizes the adjunction. It is a generalization of isomorphism (6.1) and Proposition 6.2.2.

16.2.2 Proposition *In any monoidal closed category, there is a natural equivalence (of functors of three variables)*

$$(A \otimes B) \multimap C \cong B \multimap (A \multimap C)$$

Proof. The proof uses the Yoneda Lemma (Theorem 4.5.3 of Chapter 4) and the associativity isomorphism. The naturality will appear as Exercise 1. At this point, we confine ourselves to constructing the isomorphism. We have, for any object D,

$$\begin{aligned} \mathrm{Hom}(D, (A \otimes B) \multimap C) &\cong \mathrm{Hom}((A \otimes B) \otimes D, C) \\ &\cong \mathrm{Hom}(A \otimes (B \otimes D), C) \\ &\cong \mathrm{Hom}(B \otimes D, A \multimap C) \\ &\cong \mathrm{Hom}(D, B \multimap (A \multimap C)) \end{aligned}$$

Each of these isomorphisms is easily seen to be natural in D and thus we conclude from the Yoneda Lemma that $(A \otimes B) \multimap C \cong A \multimap (B \multimap C)$. □

16.2.3 Example Let M be a commutative monoid. Recall from 3.2.1 that an M-action is a set S together with a function $\alpha : M \times S \longrightarrow S$ subject to certain conditions. We generally denote $\alpha(m, s)$ by ms and never mention α explicitly. Suppose from now on that M is commutative. If S and T are M-actions, then there are two functions $M \times S \times T \longrightarrow S \times T$, one taking (m, s, t) to (ms, t) and the other taking that element to (s, mt). The coequalizer of these two maps is denoted $S \otimes_M T$ or simply $S \otimes T$ if M is understood fixed. If we denote the class containing (s, t) by $s \otimes t$, then $S \otimes T$ can be thought of as consisting of elements $s \otimes t$, $s \in S$ and $t \in T$, subject to the relation that $ms \otimes t = s \otimes mt$ for $m \in M$. It is left as an exercise to show that this gives a monoidal structure on M-act with unit action the multiplication of M on itself. It is also a closed structure, given by defining $S \multimap T$ to be the set of equivariant maps S to T with mf defined by $(mf)(s) = m(f(s)) = f(ms)$, the latter equality from the fact that f is equivariant.

16.2.4 Evaluation and composition There are certain derived morphisms in a monoidal closed category, using the adjunction. For any objects A and B, the identity $A \multimap B \to A \multimap B$ corresponds to an arrow $e(A,B) : A \otimes (A \multimap B) \to A$ called **evaluation** since it internalizes the evaluation map. This is natural in A and B. We can then derive an arrow $c(A,B,C)$ defined as the map $(A \multimap B) \otimes (B \multimap C) \to A \multimap C$ that corresponds under the adjointness to the map

$$A \otimes (A \multimap B) \otimes (B \multimap C) \xrightarrow{e(A,B) \otimes \mathrm{id}} B \otimes (B \multimap C) \xrightarrow{e(B,C)} C$$

The arrow $c(A,B,C)$, which is also natural in all three variables, internalizes the composition arrow.

16.2.5 Cotriples in closed monoidal categories Let $\mathbf{G} = (G, \epsilon, \delta)$ be a cotriple on $\mathscr{C}$. The Kleisli category (often called the co-Kleisli category) $\mathscr{K} = \mathscr{K}(\mathbf{G})$ has the same objects as $\mathscr{C}$ and the hom sets are defined by $\mathscr{K}(A,B) = \mathscr{C}(GA, B)$. The composite of $f : GA \to B$ and $g : GB \to C$ is $g \circ Gf \circ \delta A$. This is the dual construction to the Kleisli category of a triple and the verifications are the same.

> Note that we are using the name of the category to denote the hom sets. This notation is especially useful when we are dealing with two categories that have the same objects.

One of the most important properties of the Kleisli category of a cotriple is the following. It is used to construct models of classical logic inside linear logic.

16.2.6 Theorem *Suppose $\mathbf{G} = (G, \epsilon, \delta)$ is a cotriple on a symmetric monoidal closed category $\mathscr{C}$ with the property that for all objects A and B of $\mathscr{C}$, there are natural isomorphisms between the two variable functors $G(- \times -) \cong G- \otimes G-$. Then the Kleisli category for the cotriple is cartesian closed.*

Proof. Let $\mathscr{K}$ denote the Kleisli category. Then we have, for any objects A, B and C of $\mathscr{C}$ (and therefore of $\mathscr{K}$),

$$\begin{aligned} \mathscr{K}(A, B \times C) &\cong \mathscr{C}(GA, B \times C) \cong \mathscr{C}(GA, B) \times \mathscr{C}(GA, C) \\ &\cong \mathscr{K}(A, B) \times \mathscr{K}(A, C) \end{aligned}$$

This isomorphism is clearly natural in A, which shows that, in $\mathscr{K}$, $B \times C$ is a product of B and C. (The same argument would work for any limit.) Then we have, again for any objects A, B and C of $\mathscr{C}$,

$$\begin{aligned} \mathscr{K}(A \times B, C) &\cong \mathscr{C}(G(A \times B), C) \cong \mathscr{C}(GA \otimes GB, C) \\ &\cong \mathscr{C}(GB, GA \multimap C) \cong \mathscr{K}(B, GA \multimap C) \end{aligned}$$

so that the functor $GA \multimap -$ is right adjoint to $A \times -$ in $\mathscr{K}$. □

16.2.7 Exercises

1. Show that the isomorphism of Proposition 16.2.2 is natural.

2. Show that when M is a commutative monoid, the category M-act as described in 16.2.3 is a monoidal closed category.

Note: The following exercise is more of a project than an exercise. Anyone who completes it should understand functors and natural transformations quite thoroughly, when finished.

3. a. Suppose $F : \mathrm{Ob}(\mathscr{A}) \times \mathrm{Ob}(\mathscr{B}) \to \mathrm{Ob}(\mathscr{C})$ is an object function that we want to complete to a functor. Show that what has to be done is to find,

1. for each object A of $\mathscr{A}$, a functor $F(A, -) : \mathscr{B} \to \mathscr{C}$;
2. for each object B of $\mathscr{B}$, a functor $F(-, B) : \mathscr{A} \to \mathscr{C}$; such that
3. for each $f : A \to A'$ of $\mathscr{A}$ and $g : B \to B'$ of $\mathscr{B}$ the square

$$\begin{array}{ccc} F(A,B) & \xrightarrow{F(A,g)} & F(A,B') \\ {\scriptstyle F(f,B)}\downarrow & & \downarrow{\scriptstyle F(f,B')} \\ F(A',B) & \xrightarrow[F(A',g)]{} & F(A',B') \end{array}$$

commutes.

b. Show that the third condition above is equivalent to each of the two statements

1. for each map $f : A \to A'$, $F(f, -) : F(A, -) \to F(A', -)$ is a natural transformation;
2. for each map $g : B \to B'$, $F(-, g) : F(-, B) \to F(-, B')$ is a natural transformation.

c. Show that a transformation α between functors $F, G : \mathscr{A} \times \mathscr{B} \to \mathscr{C}$ is natural if and only if for each object A of $\mathscr{A}$, the transformation $\alpha(A, -) : F(A, -) \to G(A, -)$ is natural between functors $\mathscr{B} \to \mathscr{C}$ and similarly for each object B of $\mathscr{B}$, the transformation $\alpha(-, B)$ is natural.

d. Show that in a monoidal closed category there is a natural way of defining, for each map $f : A' \to A$, a map $f \multimap C : A \multimap C \to A' \multimap C$ and for each map $h : C \to C'$ a map $A \multimap h : A \multimap C \to A \multimap C'$ in such a way that the isomorphism $\mathrm{Hom}(A \otimes B, C) \cong \mathrm{Hom}(B, A \multimap C)$ is natural in all three arguments. (Hint: Use the Yoneda lemma.)

16.3 $*$-autonomous categories

16.3.1 Definition Let $\mathscr{C}$ be a symmetric monoidal closed category. A functor $(-)^* : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathscr{C}$ is a **duality functor** if there is an isomorphism $d(A,B) : A \multimap B \xrightarrow{\cong} B^* \multimap A^*$, natural in A and B, such that for all objects A, B and C,

$$\begin{array}{ccc} (A \multimap B) \otimes (B \multimap C) & \xrightarrow{\quad c(A,B,C) \quad} & A \multimap C \\ \Big\downarrow{\scriptstyle d(A,B) \otimes d(B,C)} & & \Big\downarrow{\scriptstyle d(A,C)} \\ (B^* \multimap A^*) \otimes (C^* \multimap B^*) & \xrightarrow[c(C^*,B^*,A^*) \circ s]{} & C^* \multimap A^* \end{array}$$

commutes. In the bottom arrow, $s = s(B^* \multimap A^*, C^* \multimap B^*)$. Then we define a **$*$-autonomous** category to be a symmetric monoidal closed category with a given duality functor.

The discussion of $*$-autonomous categories in this section omits many details. The basic theory is in [Barr, 1979]. More recent developments are given in [Barr, 1996a] and [Barr, 1995]. $*$-autonomous categories form a model of linear logic [Barr, 1991], which has become important in modeling of parallel processes and is closely related to Petri nets and event structures. A good introduction, with references, to linear logic, is Yves Lafont's Appendix B of [Girard, Taylor and Lafont, 1989]. The connection between $*$-autonomous categories and linear logic is described in [Asperti and Longo, 1991], sections 4.4 and 5.5, and $*$-autonomous categories are used in modelling processes in [Pavlović and Abramsky, 1997].

16.3.2 Properties of $*$-autonomous categories There is much redundancy in the data specifying a $*$-autonomous category. Given the duality, it is not hard to show that the $\otimes$ and $\multimap$ are related by the equations:

$$A \otimes B \cong (A \multimap B^*)^* \qquad A \multimap B \cong (A \otimes B^*)^*$$

so that only the $(-)^*$ and either $\multimap$ or $\otimes$ need be given explicitly. Of course, various isomorphisms, maps and coherences must still be given. The details can be found (for the not necessarily symmetric case) in [Barr, 1995]. It turns out that it is usually most convenient to describe the $\multimap$, since when the objects are structured sets, $A \multimap B$ usually turns out to be the set of structure preserving functions of $A \longrightarrow B$, equipped with a certain structure itself.

There is a second monoidal structure in a $*$-autonomous category, which we will denote by $\oplus$, defined by $A \oplus B = (A^* \otimes B^*)^*$. The unit for this

monoidal structure is $\top^*$, which we will denote $\bot$, since $A \oplus \bot = (A^* \otimes \bot^*)^* \cong (A^* \otimes \top)^* \cong A^{**} \cong A$. Since $A^* \oplus C \cong A \multimap C$, there is a one-one correspondence between arrows $A \otimes B \longrightarrow C$ and arrows $B \longrightarrow A^* \oplus C$. This should be thought of as being analogous to the equivalence between the logical statements $a \wedge b \Rightarrow c$ and $b \Rightarrow \neg a \vee c$.

It is also possible to describe a $*$-autonomous category in terms of a symmetric closed monoidal category with an object $\bot$ subject to the condition that for every A, the map $A \longrightarrow ((A \multimap \bot) \multimap \bot)$ that corresponds under the adjunction to

$$(A \multimap \bot) \otimes A \xrightarrow{s(A \multimap \bot, A)} A \otimes (A \multimap \bot) \xrightarrow{e(A, \bot)} \bot$$

is an isomorphism. The reason is that we have

$$A^* \cong \top \multimap A^* \cong A \multimap \top^* \cong A \multimap \bot$$

which shows that the duality is determined by $\multimap$ and $\bot$. We say that $\bot$ is the **dualizing object** for the duality. It is clearly determined up to isomorphism as the dual of $\top$.

However, one of the most important ways (for computing science) of constructing $*$-autonomous categories does not work this way in that the $\multimap$ and $\otimes$ are constructed together. That is the so-called Chu construction, described in 16.4 below. We first present a familiar example.

16.3.3 Example The category **Rel** of sets and relations is a $*$-autonomous category. We saw that it was closed monoidal in Example 16.1.4. Since $\mathrm{Hom}(S,T) \cong \mathrm{Hom}(T,S)$ (both being the powerset of the cartesian product of S and T), it follows that we have a duality by letting $S^* = S$. This gives a $*$-autonomous category. In fact, it is an example of what is called a **compact** $*$-autonomous category, that is one in which $A \multimap B \cong A^* \otimes B$.

16.3.4 Exercise

1. a. Show that a finite sup semilattice is actually a lattice; that is, there is a greatest element and every pair of elements has an inf.

b. Show that if $f : L \longrightarrow M$ is a sup preserving map of finite semilattices, and if $g : M \longrightarrow L$ is defined by $g(y) = \bigvee\{x \mid f(x) \leq y\}$, then g preserves inf.

c. For a semilattice L, let L^{op} denote the same set of elements with the directions of $\leq$ reversed. The preceding implies that if L is a finite semilattice, so is L^{op}. Show that with $L^* = L^{\mathrm{op}}$, the category of finite semilattices is a $*$-autonomous category.

16.4 The Chu construction

Chu [1978, 1979] describes a construction that begins with a symmetric monoidal closed category and an object in it to use as dualizing object. He constructs from this a $*$-autonomous category that is closely related to the original category and has the chosen object as dualizing object. There is a non-symmetric version of this construction, but it is much more complicated [Barr, 1995], see also [Barr, 1996b].

Suppose that $\mathscr{V}$ is a symmetric monoidal closed category and let D denote a fixed object of $\mathscr{V}$. We define a category $\mathscr{A}$ that has as objects triples $(V, V', \langle -,- \rangle)$ where V and V' are objects of $\mathscr{V}$ and $\langle -,- \rangle : V \otimes V' \to D$ is an arrow of $\mathscr{V}$. We call $\langle -,- \rangle$ a pairing and we usually omit it in the notation, writing (V, V') for an object of $\mathscr{A}$. This notation means we have in mind a pairing from $V \otimes V'$ to D, and moreover this pairing will always be denoted $\langle -,- \rangle$. An arrow $(f, f') : (V, V') \to (W, W')$ of $\mathscr{A}$ consists of a pair of arrows $f : V \to W$ and $f' : W' \to V'$ (note the direction of the second arrow) such that

$$\begin{array}{ccc} V \otimes W' & \xrightarrow{f \otimes W'} & W \otimes W' \\ {\scriptstyle V \otimes f'}\downarrow & & \downarrow{\scriptstyle \langle -,- \rangle} \\ V \otimes V' & \xrightarrow[\langle -,- \rangle]{} & D \end{array}$$

commutes. This says that the arrows $(V, V') \to (W, W')$ consist of all pairs

$$(f, f') \in \mathrm{Hom}(V, V') \times \mathrm{Hom}(W', W)$$

that give the same element of $\mathrm{Hom}(V \otimes W', D)$ or that

$$\begin{array}{ccc} \mathrm{Hom}((V, V'), (W, W')) & \longrightarrow & \mathrm{Hom}(V, W) \\ \downarrow & & \downarrow \\ \mathrm{Hom}(W', V') & \longrightarrow & \mathrm{Hom}(V \otimes W', D) \end{array}$$

is a pullback. This observation is the key to the construction of the monoidal closed structure in $\mathscr{A}$.

The next step in the construction is to make the set of arrows into an

object of $\mathscr{V}$. We define $\mathscr{V}((V,V'),(W,W'))$ so that

$$\begin{array}{ccc} \mathscr{V}((V,V'),(W,W')) & \longrightarrow & V \multimap W \\ \downarrow & & \downarrow \\ W' \multimap V' & \longrightarrow & (V \otimes W') \multimap D \end{array}$$

is a pullback. Here the arrow on the right side is described as follows. The arrow $\langle -,- \rangle : W \otimes W' \to D$ corresponds to an arrow $W \to W' \multimap D$ (the symmetry is involved here too) and then we have a composite arrow

$$V \multimap W \to V \multimap (W' \multimap D) \xrightarrow{\cong} (V \otimes W') \multimap D$$

The lower arrow in the diagram is similar.

If (V,V') is an object of $\mathscr{A}$, then so is (V',V) using $V' \otimes V \xrightarrow{s(V',V)} V \otimes V' \xrightarrow{\langle -,- \rangle} D$. We denote it by $(V,V')^*$. It is clear that

$$\mathrm{Hom}((V,V'),(W,W')) \cong \mathrm{Hom}((W,W')^*,(V,V')^*)$$

so that $(-)^*$ is a duality on $\mathscr{A}$. Now we can give the main definitions. We let

$$(V,V') \multimap (W,W') = (\mathscr{V}((V,V'),(W,W')), V \otimes W')$$
$$(V,V') \otimes (W,W') = (V \otimes W, \mathscr{V}((V,V'),(W,W')^*))$$

16.4.1 Separated and extensional Chu objects Suppose that $\mathscr{E}/\mathscr{M}$ is a factorization system (see 2.10 in Chapter 2) on $\mathscr{V}$. Then we say that the Chu object (V,V') is **$\mathscr{M}$-separated** if the arrow $V \to V' \multimap D$ belongs to $\mathscr{M}$ and is **$\mathscr{M}$-extensional** if the arrow $V' \to V \multimap D$ belongs to $\mathscr{M}$. When the $\mathscr{M}$ remains fixed, we often omit it and speak of separated and extensional objects.

The reason for these names is this. We often think of a pair (V,V') as consisting of an object and a set of descriptions of scalar-valued functions on it. Separability is the existence of enough such descriptions in V' to distinguish (or separate) the elements of V, while extensionality is the word used to name the property of functions that two are equal if their values on all elements of their domain are equal. Functions have this property, but programs for computing them do not necessarily.

We denote the full subcategories of separated objects by

$$\mathrm{Chu}_s = \mathrm{Chu}_s(\mathscr{V}, D)$$

and of extensional objects by

$$\mathrm{Chu}_e = \mathrm{Chu}_e(\mathscr{V}, D)$$

We follow Vaughan Pratt in denoting the full subcategory of objects that are both separated and extensional by $\mathrm{chu} = \mathrm{chu}(\mathscr{V}, D)$. Since Chu_s is evidently the dual of Chu_e, it is immediate that chu is self-dual. It is useful to ask if chu is also $*$-autonomous.

We also suppose that the factorization satisfies the following two conditions

FC–1 Every arrow in $\mathscr{E}$ is an epimorphism;

FC–2 if $m \in \mathscr{M}$, then for any object A of $\mathscr{V}$, the induced $A \multimap m$ is in $\mathscr{M}$.

We will show that under these conditions, the category $\mathrm{chu}(\mathscr{V}, D)$ of separated, extensional objects (with respect to $\mathscr{M}$) in $\mathrm{Chu}(\mathscr{V}, D)$ is a $*$-autonomous category.

16.4.2 We begin by treating a somewhat common situation in a $*$-autonomous category. Suppose $\mathscr{A}$ is a such a category and $\mathscr{A}_l$ is a reflective full subcategory with reflector l. The full subcategory $\mathscr{A}_r = \mathscr{A}_l{}^{\perp}$ is quite evidently a coreflective subcategory with $rA = (lA^{\perp})^{\perp}$ as coreflector. Let $\mathscr{A}_{rl} = \mathscr{V}_r \cap \mathscr{A}_l$. Assume that when A is an object of $\mathscr{A}_r$, then so lA, which is then an object of $\mathscr{A}_{rl}$. This is readily seen to imply that when A is an object of $\mathscr{A}_l$, so is rA. It is also immediate that under these conditions, l and r induce left and right adjoints, resp., to the inclusions of $\mathscr{A}_{rl} \longrightarrow \mathscr{A}_r$ and $\mathscr{A}_{rl} \longrightarrow \mathscr{A}_l$.

16.4.3 Proposition *Under the above assumptions $\mathscr{A}_r \otimes \mathscr{A}_r \subseteq \mathscr{A}_r$ if and only if $\mathscr{A}_r \multimap \mathscr{A}_l \subseteq \mathscr{A}_l$.*

Proof. These follow immediately from the identities, valid in any $*$-autonomous category

$$A \multimap B \cong (A \otimes B^{\perp})^{\perp}$$

$$A \otimes B = (A \multimap B^{\perp})^{\perp} \qquad \square$$

16.4.4 Theorem *Suppose $\mathscr{A}$ is a $*$-autonomous category and $\mathscr{A}_l$ is a reflective subcategory whose reflector l leaves $\mathscr{A}_l{}^{\perp}$ invariant as in 16.4.2. Suppose that the unit object belongs to $\mathscr{A}_r$ and that the equivalent conditions of 16.4.3 are satisfied. Then $\mathscr{A}_{rl}$ is a $*$-autonomous category with unit object $l\top$, dualizing object rD, tensor $A \boxtimes B = l(A \otimes B)$ and $A \multimap\!\!\square\, B = r(A \multimap B)$.*

Proof. We have, for any objects A and B of $\mathscr{A}_{rl}$, using the facts that B and $A \multimap B$ belong to $\mathscr{A}_l$,

$$\begin{aligned}\mathrm{Hom}(l\top \boxtimes A, B) &\cong \mathrm{Hom}(l(l\top \otimes A), B) \cong \mathrm{Hom}(l\top \otimes A, B)\\ &\cong \mathrm{Hom}(l\top, A \multimap B) \cong \mathrm{Hom}(\top, A \multimap B) \cong \mathrm{Hom}(A, B)\end{aligned}$$

and so, by the Yoneda lemma, we conclude that $l\top \boxtimes A \cong A$.

We have, for any objects A, B, C, and D of $\mathscr{A}_{rl}$,

$$\begin{aligned}\mathrm{Hom}(A, (B \boxtimes C) \mathbin{-\!\!\square} D) &\cong \mathrm{Hom}(A, r((B \boxtimes C) \multimap D))\\ &\cong \mathrm{Hom}(A, (B \boxtimes C) \multimap D)\\ &\cong \mathrm{Hom}(A \otimes (B \boxtimes C), D) \cong \mathrm{Hom}((B \boxtimes C), A \multimap D)\\ &\cong \mathrm{Hom}(l(B \otimes C), A \multimap D) \cong \mathrm{Hom}(B \otimes C, A \multimap D)\\ &\cong \mathrm{Hom}(A \otimes B \otimes C, D)\end{aligned}$$

and

$$\begin{aligned}\mathrm{Hom}(A, B \mathbin{-\!\!\square} (C \mathbin{-\!\!\square} D)) &\cong \mathrm{Hom}(A, r(B \multimap (C \mathbin{-\!\!\square} D)))\\ &\cong \mathrm{Hom}(A, B \multimap (C \mathbin{-\!\!\square} D))\\ &\cong \mathrm{Hom}(A \otimes B, C \mathbin{-\!\!\square} D)\\ &\cong \mathrm{Hom}(A \otimes B, r(C \multimap D))\\ &\cong \mathrm{Hom}(A \otimes B, C \multimap D) \cong \mathrm{Hom}(A \otimes B \otimes C, D)\end{aligned}$$

By letting $A = l\top$, we see that $\mathrm{Hom}(B \boxtimes C, D) \cong \mathrm{Hom}(B, C \mathbin{-\!\!\square} D)$ and by Yoneda we get $(B \boxtimes C) \mathbin{-\!\!\square} D \cong B \mathbin{-\!\!\square} (C \mathbin{-\!\!\square} D)$, which is the internal version and implies the associativity of the tensor. Thus we have a $*$-autonomous category.

16.4.5 The separated extensional subcategory We apply the above construction to show that under the conditions FC–1 and FC–2 on a factorization system $\mathscr{E}/\mathscr{M}$, the full subcategory of $\mathscr{M}$-separated and $\mathscr{M}$-extensional objects forms a $*$-autonomous category.

16.4.6 Theorem *Suppose the category $\mathscr{V}$ has pullbacks and $\mathscr{E}/\mathscr{M}$ is a factorization system that satisfies FC–1 and FC–2. Then for any object D, the category* $\mathrm{chu}(\mathscr{V}, D)$ *of $\mathscr{M}$-separated, and $\mathscr{M}$-extensional objects of* Chu *is $*$-autonomous.*

We give the proof as a series of propositions.

16.4.7 Proposition *The inclusion of* $\mathrm{Chu}_s \longrightarrow \mathrm{Chu}$ *has a left adjoint.*

Proof. Let $A = (V, V')$ be an object of Chu. Factor $V \longrightarrow V'^{\perp}$ as $V \longrightarrow\!\!\!\!\!\rightarrow \widetilde{V} \rightarrowtail V'^{\perp}$, where we adopt the usual notation of writing $\longrightarrow\!\!\!\!\!\rightarrow$ for an arrow of $\mathscr{E}$ and $\rightarrowtail$ for an arrow of $\mathscr{M}$. Then $sA = (\widetilde{V}, V')$ is a separated object and we have an obvious map $A \longrightarrow sA$. If $(f_1, f_2) : (V, V') \longrightarrow (W, W')$ is a map in $\mathrm{Chu}(\mathscr{V}, K)$ and (W, W') is separated, the unique diagonal fill-in in the square

$$\begin{array}{ccc} V & \longrightarrow\!\!\!\!\!\rightarrow & \widetilde{V} \\ \downarrow & & \downarrow \\ & & V'^{\perp} \\ & & \downarrow \\ W & \rightarrowtail & W'^{\perp} \end{array}$$

gives the required map $sA \longrightarrow B$. □

It follows that the inclusion $\mathrm{Chu}_e \longrightarrow \mathrm{Chu}$ has a right adjoint, which we denote e.

16.4.8 Proposition *FC–1 implies that when the object* (V, V') *of the category* $\mathrm{Chu}(\mathscr{V}, D)$ *is separated, so is* $e(V, V')$ *and similarly if* (V, V') *is extensional, so is* $s(V, V')$.

Proof. Suppose that $A = (V, V')$ is a separated object. Then $eA = (V, \widetilde{V}')$ where $V' \longrightarrow\!\!\!\!\!\rightarrow \widetilde{V}' \rightarrowtail V^{\perp}$. By transposing, this gives $V \longrightarrow (\widetilde{V}')^{\perp} \longrightarrow V'^{\perp}$ whose composite belongs to $\mathscr{M}$ and so, by Proposition 2.10.7, does the first factor $V \longrightarrow (\widetilde{V}')^{\perp}$ which means that eA is still separated. The proof that sA is extensional when A is is similar (also dual). □

16.4.9 Proposition *FC–2 implies that when* A *and* B *are extensional, so is* $A \otimes B$.

Proof. Let $A = (V, V')$ and $B = (W, W')$ be extensional. The second component of $A \otimes B$ is the pullback

$$\begin{array}{ccc} P & \longrightarrow\!\!\!\!\!\rightarrow & V \multimap W' \\ \downarrow & & \downarrow \\ W \multimap V' & \longrightarrow & W \multimap V^{\perp} \cong V \multimap W^{\perp} \cong (V \otimes W)^{\perp} \end{array}$$

It follows from FC–2 that both arrows $W \multimap V' \longrightarrow W \multimap V^{\perp}$ and $V \multimap W' \longrightarrow V \multimap W^{\perp}$ lie in $\mathscr{M}$. But it is a general property of factorization systems that $\mathscr{M}$ is closed under pullback and composition, whence the arrow $P \longrightarrow (V \otimes W)^{\perp}$ also lies in $\mathscr{M}$. □

The results of Section 16.4.4 now prove the theorem.

16.4.10 Examples The first example is Chu(**Set**, 2). An object is simply a pair (S, S') of sets, together with a function $S \times S' \longrightarrow 2$. Equivalently, it is a subset of $S \times S'$, otherwise known as a relation between S and S'. The extensional objects are those for which S' is, up to isomorphism, a set of subsets of S and then $\langle s, s' \rangle = 1$ if $s \in s'$ and 0 otherwise. The separated ones are those for which, given $s_1,\ s_2 \in S$, there is at least one $s' \in S'$ with either $s_1 \in s'$ and $s_2 \notin s'$ or vice versa. A set together with a set of subsets is the beginnings of a topological space and the separation condition is the same as that of a T_1 topological space. This example has been intensively studied by Vaughan Pratt and his students.

A second example is given by taking $\mathscr{V}$ to be the category of vector spaces over a field F. The category of finite dimensional vector spaces is already a $*$-autonomous category with the set of linear transformations between two spaces as internal hom and the usual vector space dual as the duality operator. It is easily seen to be a $*$-autonomous subcategory of Chu$(\mathscr{V}, F)$. It is also a $*$-autonomous subcategory of the separated extensional subcategory chu$(\mathscr{V}, F)$. There is one distinctive feature of the separated extensional subcategory worth noting. In that case (and so far as is known only in that case), the tensor product and internal hom of separated extensional objects is already separated and extensional and there is no reason to apply the reflector and coreflector, respectively (see [Barr, 1996c]).

Solutions to the exercises

Solutions for Chapter 1

Section 1.2

1. Suppose h is surjective, and suppose $\mathrm{Hom}(h,T)(f) = \mathrm{Hom}(h,T)(g)$. Then $f \circ h = g \circ h$. Let $x \in S$, and let $w \in W$ satisfy $h(w) = x$ (using surjectivity of h). Then $f(x) = f(h(w)) = g(h(w)) = g(x)$. Since x was arbitrary, this shows that $f = g$, so that $\mathrm{Hom}(h,T)$ is injective. Conversely, suppose that h is not surjective. This means there is a particular $x \in S$ which is not $h(w)$ for any $w \in W$. Let $t, u \in T$ be two distinct elements of T. Let $f : S \to T$ be the constant function with value t. Let $g : S \to T$ take x to u and all other elements to t. Then $\mathrm{Hom}(h,T)(f) = \mathrm{Hom}(h,T)(g)$, so $\mathrm{Hom}(h,T)$ is not injective.

2. a. To show that the mapping is injective, suppose $\langle f, g\rangle = \langle f', g'\rangle$. Then for any $x \in X$,

$$(f(x), g(x)) = \langle f, g\rangle(x) = \langle f', g'\rangle(x) = (f'(x), g'(x))$$

The coordinates of these pairs must be the same, so $f(x) = f'(x)$ and $g(x) = g'(x)$ for all $x \in X$. Hence $f = f'$ and $g = g'$, so $(f, g) = (f', g')$. To get surjectivity, suppose that $q : X \to S \times T$ is any function. Define $f : X \to S$ by requiring that $f(x)$ be the first coordinate of the pair $q(x)$ for all $x \in X$; in other words, $f(x) = \mathrm{proj}_1(q(x))$. Similarly set $g(x) = \mathrm{proj}_2(q(x))$. Then the mapping of the problem takes (f, g) to q, so it is surjective.

b. The pair $(\mathrm{proj}_1, \mathrm{proj}_2)$.

3. a. Define $\psi : \mathrm{Hom}(S \cup T, V) \to \mathrm{Hom}(S,V) \times \mathrm{Hom}(T,V)$ by $\psi(h : S \cup T \to V) = (h|S, h|T)$, where $h|S$ denotes the restriction of h to the subset S of $S \cup T$. It is easy to see that ψ is the inverse function of ϕ.

b. It is (i, j), where i is the inclusion of S in $S \cup T$ and j is the inclusion of T in $S \cup T$.

4. a. Define $\psi : \mathrm{Hom}(A, \mathscr{P}B) \to \mathrm{Rel}(A, B)$ by

$$\psi(f : A \to \mathscr{P}B) = \{(a, b) \mid b \in f(a)\}$$

Then ψ is inverse to ϕ.

b. The map $a \mapsto \{a\}$.

c. The opposite of the 'element of' relation: If Y is a subset of B and $b \in B$ then Y is related to b if and only if $b \in Y$.

Section 1.3

1. The graph

$$\mathsf{a} \underset{\mathsf{t}}{\overset{\mathsf{s}}{\rightrightarrows}} \mathsf{n} \xrightarrow{\mathsf{label}} \mathsf{l}$$

expresses the concept. Interpret n as nodes and l as labels; then the arrow called 'label' takes a node to its label.

2. By definition, $\mathscr{G}$ is simple if and only if for any two distinct arrows f and g, either $\text{source}(f) \neq \text{source}(g)$ or $\text{target}(f) \neq \text{target}(g)$. Since for any arrow x,

$$\langle \text{source}, \text{target} \rangle(x) = (\text{source}(x), \text{target}(x))$$

$\mathscr{G}$ is simple if and only if for any two distinct arrows f and g,

$$\langle \text{source}, \text{target} \rangle(f) \neq \langle \text{source}, \text{target} \rangle(g)$$

But that says that $\langle \text{source}, \text{target} \rangle$ is injective by definition.

Section 1.4

1. Let $\phi : \mathscr{G} \to \mathscr{H}$ be a graph homomorphism with $\mathscr{H}$ simple. Let $f : a \to b$ be an arrow of $\mathscr{G}$. Then $\phi_1(f)$ has to be an arrow from $\phi_0(a)$ to $\phi_0(b)$. But there is at most one such arrow because $\mathscr{H}$ is simple. Thus $\phi_1(f)$ is determined by $\phi_0(a)$ and $\phi_0(b)$.

2. We must show that if $u : m \to n$ is an arrow of $\mathscr{G}$, then

$$\psi_1(\phi_1(u)) : \psi_0(\phi_0(m)) \to \psi_0(\phi_0(n))$$

in $\mathscr{K}$. Since ϕ is a graph homomorphism, $\phi_1(u) : \phi_0(m) \to \phi_0(n)$ in $\mathscr{H}$. The required equation then follows because ψ is a graph homomorphism.

3. a. Let $h : c \to d$ be an arrow in $\mathscr{H}$. We must show that $\psi_1(h) : \psi_0(c) \to \psi_0(d)$ in $\mathscr{G}$. Suppose $\psi_1(h) : a \to b$. Then, by definition of ψ and the fact that ϕ is a graph homomorphism,

$$h = \phi_1(\psi_1(h)) : \phi_0(a) \to \phi_0(b)$$

in $\mathscr{H}$. Thus $\phi_0(a) = c$ and $\phi_0(b) = d$, so $\psi_0(c) = a$ and $\psi_0(d) = b$ as required.

Here is a second proof that fits a pattern we use over and over in this book in showing that if an invertible arrow has a certain property, then its inverse does too.

$$\begin{aligned} \text{source}(\psi_1(h)) &= \psi_0(\phi_0(\text{source}(\psi_1(h)))) \\ &= \psi_0(\text{source}(\phi_1(\psi_1(h)))) \quad = \quad \psi_0(\text{source}(h)) \end{aligned}$$

and similarly for target.

b. By definition, we must prove the following equations:

$$\begin{aligned} \psi_0 \circ \phi_0 &= (\text{id}_{\mathscr{G}})_0 \\ \psi_1 \circ \phi_1 &= (\text{id}_{\mathscr{G}})_1 \\ \phi_0 \circ \psi_0 &= (\text{id}_{\mathscr{H}})_0 \\ \phi_1 \circ \psi_1 &= (\text{id}_{\mathscr{H}})_1 \end{aligned}$$

For any graph $\mathscr{F}$, $(\text{id}_{\mathscr{F}})_0 = \text{id}_{F_0}$ and $(\text{id}_{\mathscr{F}})_1 = \text{id}_{F_1}$ (see Example 1.4.3). The result then follows from the fact that $\psi_i = (\phi_i)^{-1}$ for $i = 1, 2$.

Solutions for Chapter 2

Section 2.1

1. Functional composition is associative, and the identity functions satisfy C–3 and C–4, so it is only necessary to show that the identity functions are injective and that composite of injective functions is injective. If S is a set and $x, y \in S$ with $x \neq y$, then $\mathrm{id}_S(x) = x \neq y = \mathrm{id}_S(y)$, so id_S is injective. Let $f : S \to T$ and $g : T \to V$. If $x, y \in S$ and $x \neq y$, then $f(x) \neq f(y)$ because f is injective. But then $g(f(x)) \neq g(f(y))$ because g is injective. Hence $g \circ f$ is injective.

2. It is necessary to show that id_S is surjective for each set S and that if $f : S \to T$ and $g : T \to V$ are surjective then so is $g \circ f$. If $x \in S$, then $\mathrm{id}_S(x) = x$ so id_S is surjective. If $v \in V$, then there is a $t \in T$ for which $g(t) = v$ since g is surjective. There is $x \in S$ for which $f(x) = t$ since f is surjective. Then $g(f(x)) = v$, so $g \circ f$ is surjective.

3. Let α be a relation from A to B, β a relation from B to C and γ a relation from C to D. By definition of composite, for $x \in A$ and $u \in D$, $(x, u) \in (\gamma \circ \beta) \circ \alpha$ if and only if there is an element $y \in B$ for which $(x, y) \in \alpha$ and $(y, u) \in \gamma \circ \beta$. But $(y, u) \in \gamma \circ \beta$ if and only if there is $z \in C$ such that $(y, z) \in \beta$ and $(z, u) \in \gamma$. Thus $(x, u) \in (\gamma \circ \beta) \circ \alpha$ if and only if there are elements $y \in B$ and $z \in C$ such that $(x, y) \in \alpha$, $(y, z) \in \beta$ and $(z, u) \in \gamma$.

On the other hand, (x, u) is in $\gamma \circ (\beta \circ \alpha)$ if and only if there is $z \in C$ such that $(x, z) \in \beta \circ \alpha$ and $(z, u) \in \gamma$. But $(x, z) \in \beta \circ \alpha$ if and only if there is $y \in B$ such that $(x, y) \in \alpha$ and $(y, z) \in \beta$. Thus $(x, u) \in \gamma \circ (\beta \circ \alpha)$ if and only if there are $y \in B$ and $z \in C$ such that $(x, y) \in \alpha$, $(y, z) \in \beta$ and $(z, u) \in \gamma$. Hence $(\gamma \circ \beta) \circ \alpha = \gamma \circ (\beta \circ \alpha)$.

4. a. $\mathrm{id}_A \circ u = u$ by definition of id_A, but $\mathrm{id}_A \circ u = \mathrm{id}_A$ by assumption. Hence $u = \mathrm{id}_A$. This is an example of a useful heuristic: when a property is given in terms of all arrows and you want to prove something about the property, see what it says for identity arrows.

b. $u \circ \mathrm{id}_A = u$ by definition of id_A, but $u \circ \mathrm{id}_A = \mathrm{id}_A$ by assumption. Hence $u = \mathrm{id}_A$.

Section 2.2

1. This requires an arrow `nonzero` : `NAT` $\longrightarrow$ `BOOLEAN` and the following equations: `nonzero` $\circ\, 0 =$ `false` and `nonzero` $\circ$ `succ` $=$ `true`. Since an arrow to `NAT` has to be a composite ending in 0 or `succ`, this takes care of all possibilities.

Section 2.3

1. A must be empty or have only one element.

2. Composition is certainly a binary operation on $\mathrm{Hom}(A, A)$ since all the arrows involved have the same source and target, so that every pair of arrows is composable. It is associative by C–2 and id_A is an identity by C–4.

3. If a semigroup S has two identity elements e and e', then $e = ee' = e'$. The first equation is true because e' is an identity and the second one is true because e is an identity.

Section 2.4

1. a. If (S, α) is a set with relation, then $\mathrm{id}_S : (S, \alpha) \to (S, \alpha)$ is a homomorphism and satisfies C–3 and C–4. We must show that the composite of homomorphisms is a homomorphism; since they are set functions, C–1 and C–2 will follow. Suppose $f : (S, \alpha) \to (T, \beta)$ and $g : (T, \beta) \to (V, \gamma)$ are homomorphisms. If $x \alpha y$, then $f(x) \beta f(y)$ because f is a homomorphism. Then $g(f(x)) \gamma g(f(y))$ since g is a homomorphism. Hence $g \circ f$ is a homomorphism.

b. This is an immediate consequence of the definition of monotone in 2.4.2.

2. The identity functions are clearly continuous and strict (if relevant), so we need only show that the composite of continuous functions is continuous and the composite of strict functions is strict. Let $f : S \to T$ and $g : T \to V$ be continuous functions. Let s be the supremum of a chain $\mathscr{C} = (c_0, c_1, c_2, \ldots)$ in S. Then the image $f(\mathscr{C}) = \{f(c_i) \mid i \in \mathbf{N}\}$ is a chain in T because a continuous function is monotone. Moreover, $f(s)$ is the supremum of $f(\mathscr{C})$. Since g is continuous, $g(f(s))$ is the supremum of $g(f(\mathscr{C}))$, so that $g \circ f$ is continuous. Finally, if f and g are strict, so is $g \circ f$, since then $g(f(D)) = g(D) = D$.

3. The nonnegative integers form a chain in $\mathbf{R}^+$. There is no element in $\mathbf{R}^+$ satisfying SUP–1, since such an element would be a real number which is bigger than any integer.

4. Let $\mathscr{C} = (C_0, C_1, C_2, \ldots)$ be a chain in $(\mathscr{P}(S), \subseteq)$. Let V be the union $\bigcup_{i=1}^{\infty} C_i$. Then for every i, $C_i \subseteq V$, so SUP–1 is satisfied. Let $C_i \subseteq W$ for every i and suppose $v \in V$. Then $v \in C_i$ for at least one i since V is the union of all the C_i. Thus $v \in W$, which proves that $V \subseteq W$ so that SUP–2 holds. The bottom element is $\emptyset$.

5. Let $S = \mathbf{Z} \cup \{\top, \widehat{\top}\}$, where $\top$ and $\widehat{\top}$ are distinct and are not integers. Let the ordering $\leq$ on S be the usual ordering on $\mathbf{Z}$, with $\top \leq \widehat{\top}$ and for any integer n, $n \leq \top$ and $n \leq \widehat{\top}$. Then $(S, \leq)$ is an ω-CPO. The function $f : (S, \leq) \to (S, \leq)$ for which $n \mapsto n$ for $n \in \mathbf{Z}$, $\top \mapsto \widehat{\top}$ and $\widehat{\top} \mapsto \widehat{\top}$ is monotone, but not continuous, since it does not preserve the sup of the chain $\mathbf{Z}$: that sup is $\top$ but $f(\top) = \widehat{\top}$.

6. For a partial function h, define $\psi(h)(0) = 1$, and require that $\psi(h)(n)$ is defined if and only if $h(n-1)$ is defined, and then $\psi(h)(n) = 2\,h(n-1)$. Then the unique fixed point of ψ is g.

7. For a partial function h, define $\psi(h)$ by requiring that $\psi(h)(0) = \psi(h)(1) = 1$ and that for $n > 1$, if $h(n-1)$ and $h(n-2)$ are defined, then $\psi(h)(n)$ is defined and $\psi(h)(n) = h(n-1) + h(n-2)$. Then the Fibonacci function is the unique fixed point of ψ.

Section 2.5

1. Let $f : S \to T$ and $g : T \to V$ be semigroup homomorphisms. Then for any two elements $s, s' \in S$, $g(f(s))g(f(s')) = g(f(s)f(s')) = g(f(ss'))$, the first equation because g is a homomorphism and the second because f is a homomorphism. Thus $g \circ f$ is a homomorphism.

In the monoid case, we also have $g \circ f(1) = g(f(1)) = g(1) = 1$, so that again $g \circ f$ is a homomorphism.

2. For $m \in \mathbf{Z}_k$, $m = m + 0 = 0 \cdot k + m$, so $m +_k 0 = m$ and similarly $0 +_k m = m$. To verify the associative law let m, n and $p \in \mathbf{Z}_k$. Define r_i, q_i, $i = 1, 2, 3, 4$ by

$$\begin{aligned} m + n &= q_1 k + r_1 \\ n + p &= q_2 k + r_2 \\ r_1 + p &= q_3 k + r_3 \\ m + r_2 &= q_4 k + r_4 \end{aligned}$$

and $0 \leq r_i < k$ for $i = 1, 2, 3, 4$. We must show that $r_3 = r_4$. It follows from the equations just given that $r_3 - r_4 = (q_2 + q_4 - q_1 - q_3)k$. Since $0 \leq r_i < k$ for each i, the difference $r_3 - r_4$ has to be between $-k$ and k not inclusive. Since it is a multiple of k, it must be 0; hence $r_3 = r_4$.

3. Call the function ϕ, so that for some q, $n = qk + \phi(n)$ and $0 \leq \phi(n) < k$. Since $0 = 0 \cdot k + 0$, $\phi(0) = 0$. For arbitrary m and n in $\mathbf{Z}$, let q_1 and q_2 be the integers for which $m + n = q_1 k + \phi(m + n)$ and $\phi(m) + \phi(n) = q_2 k + \phi(m) +_k \phi(n)$. Then

$$\phi(m + n) - (\phi(m) +_k \phi(n)) = (q_1 - q_2)k + (m - \phi(m)) + (n - \phi(n))$$

so that the left hand side of that equation is a multiple of k. Since it is the difference of two numbers whose absolute values are less than k, the difference must be 0, so that $\phi(m) +_k \phi(n) = \phi(m + n)$.

4. It is easy to show that there is a one to one correspondence between homomorphisms ϕ from that first monoid to any given monoid M and elements $x \in M$ such that $x^4 = 1$. The correspondence takes such an element x to the homomorphism ϕ which take the elements 0, 1, 2 and 3 to 1, x, x^2 and x^3, respectively. This clearly preserves the identity and the equation $\phi(m + n) = \phi(m)\phi(n)$ is interesting only when there is a carry mod 4, and in that case reduces to the requirement that $x^4 = 1$. A ϕ constructed this way is injective if and only if $x^n \neq 1$ for $n = 1$, 2 or 3. The second monoid is easily seen to have two such elements with that property, namely 2 and 3.

5. Since f^* is a homomorphism by 2.5.7, it is necessary by 2.5.5 only to show that it is bijective. We are given that f is a bijection (isomorphisms in **Set** are bijections). If a and a' are lists in A^* and they are different, then they differ in some coordinate; suppose the ith coordinates a_i and a'_i are different. The ith coordinates of $f^*(a)$ and $f^*(a')$ are then $f(a_i)$ and $f(a'_i)$, which must be different because f is injective. Thus f^* is injective. If b is a list in B, let a be a list in A of the same length for which $f(a_i) = b_i$ for each coordinate i. There is such a list since f is surjective, and then $f^*(a) = b$, so f^* is surjective.

Section 2.6

1. $C(M)^{\mathrm{op}}$ has exactly one object $*$ because $C(M)$ does. Define M^{op} to be $\mathrm{Hom}_{C(M)^{\mathrm{op}}}(*, *)$. If x, y, $z \in M$ and $xy = z$ in M then $yx = z$ in M^{op}. Clearly $C(M^{\mathrm{op}}) = C(M)^{\mathrm{op}}$.

2. The ordering in all posets in this answer will be written '$\leq$'. If P is a poset then between any two objects in $C(P)^{\mathrm{op}}$ there is at most one arrow because that is true in $C(P)$. Thus $C(P)^{\mathrm{op}}$ is the category determined by a poset called P^{op}; $x \leq y$ in P^{op} if and only if $y \leq x$ in P.

3. Let P be the set $\{1,2\}$ with discrete ordering (no two different elements are related). Let Q be the same set with the usual ordering ($1 \leq 2$). Let R be the set $\{1,2,3\}$ with the discrete ordering. $C(P)$ is wide in $C(Q)$ (they have the same elements) but not full ($1 \leq 2$ in Q but not in P). $C(P)$ is full in $C(R)$ because in both $x \leq y$ holds only when $x = y$, but it is not wide since they do not have the same elements.

4. Let M be the two-element monoid $M = \{1, e\}$ with identity element 1 and $ee = e$. Then in $C(M)$, the only object is $*$ and the arrows are $1 : * \to *$ and $e : * \to *$. Consider the category $\mathscr{D}$ with one object $*$ and one arrow e. Note that e is the identity arrow for $*$ in $\mathscr{D}$ but not in $\mathscr{C}$. $\mathscr{D}$ satisfies requirements S–1, S–2 and S–4 but not S–3. Note that $\mathscr{D}$ is even a category; it is just not a subcategory.

Section 2.7

1. We have $f^{-1} \circ g^{-1} \circ g \circ f = f^{-1} \circ f = \mathrm{id}_A$ and similarly $g \circ f \circ f^{-1} \circ g^{-1} = \mathrm{id}_C$. Thus $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$ because the inverse is unique (2.7.3).

2. (a) Let P_1 be the set $\{1,2\}$ with discrete ordering (no two different elements are related). Then $P_1 = {P_1}^{\mathrm{op}}$ since $x \leq y$ if and only if $x = y$.

(b) Let P_2 be the same set with the usual ordering ($1 \leq 2$). Then $P_2 \neq {P_2}^{\mathrm{op}}$ since $1 \leq 2$ in P_2 but not in ${P_2}^{\mathrm{op}}$. However the map $1 \mapsto 2$, $2 \mapsto 1$ is an isomorphism from P_2 to ${P_2}^{\mathrm{op}}$.

(c) Finally, let $P_3 = \{\{1\}, \{2\}, \{1,2\}\}$ ordered by inclusion. Then P_3 is not isomorphic to ${P_3}^{\mathrm{op}}$ since P_3 has an upper bound, namely $\{1,2\}$, but ${P_3}^{\mathrm{op}}$ has no upper bound.

3. First we construct a monoid M for which $M \neq M^{\mathrm{op}}$ but M is isomorphic to M^{op}: let $A = \{x, y\}$ and $M = A^*$. Then in M, $(x)(y) = (x, y)$, while in M^{op}, $(x)(y) = (y, x)$ so that $M \neq M^{\mathrm{op}}$. On the other hand the function $\phi : M \to M^{\mathrm{op}}$ defined by $\phi(a_1, \ldots, a_n) = (a_n, \ldots, a_1)$ is easily seen to be an isomorphism of M with M^{op}.

For a monoid M that is not isomorphic to M^{op}, we use a construction that works for any set S: The 'right monoid' generated by S is defined on the set $S \cup \{1\}$ (1 is some element not in S) by $xy = y$ for $x, y \in S$ and $x1 = 1x = x$ for $x \in S \cup \{1\}$. M^{op} (the 'left monoid') is not isomorphic to M so long as S has two or more elements.

4. Let $(P, \leq)$ and $(Q, \leq)$ be posets and $\phi : P \to Q$ an isomorphism. Suppose P is totally ordered. Let q, $q' \in Q$. Since P is totally ordered, we may assume (after perhaps renaming) that $\phi^{-1}(q) \leq \phi^{-1}(q')$. Then $q = \phi(\phi^{-1}(q) \leq \phi(\phi^{-1}(q'))) = q'$ so Q is totally ordered.

5. If x and y are isomorphic objects, then there is an arrow from x to y and one from y to x (the isomorphism and its inverse). Then $x \leq y$ and $y \leq x$, so $x = y$ by antisymmetry.

6. Let $\phi : S \to T$ be an isomorphism of semigroups. Since it a set function with an inverse, it is a bijection by Proposition 2.7.9. Conversely, suppose that $\phi : S \to T$ is a bijective homomorphism. Then its inverse is a homomorphism by 2.5.5.

7. Suppose f satisfies (a). Then for any x, $y \in A$, $f(x) = k(*) = f(y)$. Conversely, suppose f satisfies (b). If $A = B = \emptyset$, then $\langle\rangle = \mathrm{id}_\emptyset$ and we can take $k = \mathrm{id}_\emptyset$ to get $f = \mathrm{id}_\emptyset = \mathrm{id}_\emptyset \circ \mathrm{id}_\emptyset$. Otherwise, define an element b of B as follows. If $A = \emptyset$, let b be any element of B. If not, let b be the element such that $f(x) = f(y) = b$ for all x, $y \in A$. Define $k : 1 \to A$ to be $* \mapsto b$ where $*$ is the unique element of 1. Then $f = k \circ \langle\rangle$.

8. Let $\mathscr{E}$ be the graph with one node e and one arrow $f : e \to e$. Let $\mathscr{G}$ be any graph. Define $\phi : \mathscr{G} \to \mathscr{E}$ by $\phi_0(g) = e$ for any node g and $\phi_1(u) = f$ for any arrow $f : g \to h$. Then $\phi_1(u) : \phi_0(g) \to \phi_0(h)$, so ϕ is a graph homomorphism. It is clearly the only possible one.

9. Suppose $f : S \to S$ is an idempotent set function. Let x be an element of the image, so there is some $s \in S$ such that $f(s) = x$. Then $f(x) = f(f(s)) = f(s) = x$, so x is a fixed point. Conversely, if $f(x) = x$, then x is in the image of f by definition.

Conversely, suppose the image of f is the same as its set of fixed points. Then for any $x \in S$, $f(f(x)) = f(x)$, so that $f \circ f = f$.

10. a. Let $f : A \to A$ be an idempotent set function. Let B be the image of f and $g : A \to B$ the corestriction of f. Let $h : B \to A$ be the inclusion of B in A. Then for any $x \in A$, $h(g(x)) = g(x) = f(x)$, so $h \circ g = f$. And for any $b \in B$, $g(h(b)) = g(b) = f(b) = b$ by the result of the preceding problem, so $g \circ h = \mathrm{id}_B$.

b. Let $\mathscr{E}$ be the category with one object $*$ and two arrows 1, e with $1 = \mathrm{id}_*$ and $e \circ e = e$. There are no arrows g, h for which $h \circ g = e$ and $g \circ h = 1$ since in this category, composition is commutative. (Note that 1 is split. The identity is split in any category.)

11. (i) is false. The category with two distinct objects and only identity arrows is a counterexample.

(ii) is false. Take the category of the preceding example and add one arrow u going from one of the objects, say A, to itself, with $u \circ u = \mathrm{id}_A$.

(iii) is true. Suppose $f : A \to B$ is an arrow of $C(P)$. Its inverse goes from B to A so $A \leq B$ and $B \leq A$ in P. Hence $A = B$. Since there is never more than one arrow with the same source and target in $C(P)$, f must be id_A.

Section 2.8

1. a. Let $f : A \to B$ be a monomorphism in $\mathscr{C}$ and let $\mathscr{D}$ be a subcategory of $\mathscr{C}$. Then if $f \circ g = f \circ h$ in $\mathscr{D}$ the same equation is true in $\mathscr{C}$, so that $g = h$ in $\mathscr{C}$. Hence f is a monomorphism in $\mathscr{D}$.

b. Let $\mathscr{C}$ be the full subcategory of **Set** determined by the sets $A = \{1\}$ and $B = \{1, 2\}$. Let $k : B \to A$ be the (only possible) function. Let $\mathscr{D}$ be the subcategory of $\mathscr{C}$ consisting of id_A, id_B, and k. Then k is a monomorphism in $\mathscr{D}$ since there are not two distinct arrows from B to B, but it is not a monomorphism in $\mathscr{C}$ since k composed with any of the four functions from B to B gives k again.

2. Suppose $(g \circ f) \circ x = (g \circ f) \circ y$. Then $g \circ (f \circ x) = g \circ (f \circ y)$. Because g is a monomorphism, $f \circ x = f \circ y$. Then $x = y$ as required because f if a monomorphism.

3. This happens in **Set**. Let f to f be the inclusion of $\{1,2\}$ into $\{1,2,3\}$ and $g : \{1,2,3\} \to \{1,2\}$ be $1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 2$. Then $g \circ f$ (which is $\mathrm{id}_{\{1,2\}}$) is monic but g is not.

4. In this answer we use these facts repeatedly:

(a) An isomorphism in the category of sets is a bijection and conversely.

(b) If $i : X \to S$ and $j : Y \to S$ are injective functions and $\beta : X \to Y$ is a bijection for which $j \circ \beta = i$ then $i \circ \beta^{-1} = j$.

(a)(i) Let $s = m(a)$ be in the image of m and let $\beta : A \to B$ be the bijection for which $m = n \circ \beta$. Then $s = n(\beta(a))$ so s is in the image of n. A symmetric argument using β^{-1} shows that the image of n is included in the image of m, so the images are equal.

(a)(ii) Let $m : A \to S$ be an injection in $\mathscr{O}$. Define $\beta : A \to I$ to be the corestriction of m to I. β is injective because m is and surjective because I is the image of m. Hence it is bijective. Clearly $i \circ \beta = m$.

(a)(iii) Let $\beta : J \to I$ be the bijection for which $i \circ \beta = j$. Since i and j are both inclusions, for any $x \in J$, $i(x) = x = j(x) = i(\beta(x))$, so $i \circ \beta = i$. Since i is injective, $\beta(x) = x$ for all $x \in J$. Since β is surjective, $I = J$ and $\beta = \mathrm{id}_I$.

(a)(iv) Immediate from (i), (ii) and (iii).

(b)(i) Immediate from the definition of subobject.

(b)(ii) Immediate from properties of equivalence relations.

(b)(iii) Immediate from (i) and (ii).

5. We must find arrows $k : D \to C$ and $k' : C \to D$ such that $\mathrm{id}_D = h \circ k$ and $h = \mathrm{id}_D \circ k'$. These requirements are satisfied by $k = h^{-1}$ and $k' = h$.

6. Let $m : C \to 0$ be a monomorphism into an initial object. By Proposition 2.8.7, m is an isomorphism. By Problem 5, it is equivalent to id_0, so that the subobject it inhabits is not proper by definition of proper.

7. Let $m : A \to 1$ be monic and let $f, g : B \to A$. Then $m \circ f = m \circ g : B \to 1$ by definition of terminal object, and m is monic, so $f = g$. In particular, for any arrow $h : A \to C$, $h \circ f = h \circ g$ implies that $f = g$; hence h is monic.

8. a. The terminal object is a one-element set. By the remarks in 2.8.12, such a set has two subobjects because it has two subsets: the set itself and the empty set. (Note in connection with the preceding exercise that there is at most one function from any set B to the empty set: none at all if B is nonempty, and the identity function if B is empty.)

b. The terminal graph has one node and one arrow. It has three subgraphs: itself, the graph with one node and no arrows, and the graph with no nodes or arrows. These are in one to one correspondence with the subobjects.

c. The terminal monoid is the monoid with one element. It has only itself as a submonoid.

9. A relation $M : A \to B$ is a monomorphism in **Rel** if and only if the following condition holds: for any two subsets U and V of A, if U and V have the same images under M, then $U = V$. (In other words, the induced function from $\mathscr{P}A$ to $\mathscr{P}B$ is injective.) The image of U under M is the set ${}_b\exists a \in U(aMb)$; we will denote it by UM.

To prove this, first suppose M satisfies the condition and suppose $R, S : C \to A$ are relations such that $M \circ R = M \circ S$. For any c, the images $c(M \circ R)$ and $c(M \circ S)$ are the same, so the sets cR and cS have the same image under M. By the condition, $cR = cS$. But that is for every c, which implies that $R = S$.

Suppose M does not satisfy the condition. Let U and V be different subsets of A with the same image under M. Define relations $R, S : \{1\} \to A$ by $1Ra \equiv a \in U$ and $1Sa \equiv a \in V$. Then $1R = U$ and $1S = V$, and U and V have the same images under M, so $M \circ R = M \circ S$.

Section 2.9

1. Let $f : S \to T$ be a surjective monoid homomorphism and let g, $h : T \to V$ be monoid homomorphisms. Suppose that $g \circ f = h \circ f$. Let t be any element of T. Then there is an element $x \in S$ for which $f(x) = t$. Then $g(t) = g(f(x)) = h(f(x)) = h(t)$. Since t was arbitrary, $g = h$ so f is an epimorphism.

2. That ϕ is a homomorphism requires verifying that $\phi(0) = 0$ (which is true by definition) and that $\phi(m +_4 n) = \phi(m) +_2 \phi(n)$ for all m, $n \in \mathbf{Z}_4$: for example, $\phi(2 +_4 3) = \phi(1) = 1$ and $\phi(2) +_2 \phi(3) = 0 +_2 1 = 1$. (The comments in the answer to Exercise 4 of Section 2.5 apply here too.) It is surjective since it has both 0 and 1 as values.

Now suppose $\psi : \mathbf{Z}_2 \to \mathbf{Z}_4$ is a monoid homomorphism satisfying $\phi \circ \psi = \mathrm{id}_{\mathbf{Z}_2}$. Since $\psi(0) = 0$ necessarily, there are only four possibilities for ψ, determined by what it does to 1. If $\psi(1) = 0$ then $\phi(\psi(1)) = \phi(0) = 0 \neq 1$. If $\psi(1) = 1$ then $\psi(1 +_2 1) = \psi(0) = 0$ but $\psi(1) +_4 \psi(1) = 2$, so ψ is not a homomorphism. If $\psi(1) = 2$ then $\phi(\psi(1)) \neq 1$. Finally, if $\psi(1) = 3$, then again ψ is not a homomorphism because $\psi(1 +_2 1) = 0 \neq 2 = \psi(1) +_4 \psi(1)$.

3. a. The statement that $m \in M$ is a monomorphism says that $mn = mp$ implies $n = p$ (m is left cancellable). That it is an epimorphism says that $nm = pm$ implies $n = p$ (m is right cancellable) If m is an isomorphism then m is a monomorphism and an epimorphism by Proposition 2.9.10. To see the converse, let m be a monomorphism. Let $M = \{m_1, m_2, \ldots, m_k\}$ be finite. The elements $mm_1, mm_2, \ldots, mm_k$ are all different, so there are k of them, so one of them, call it mn, is 1. But then $mnm = m = m \cdot 1$ and m is a monomorphism, so $nm = 1$. Hence m is an isomorphism. A symmetric argument shows that an epimorphism is an isomorphism.

Warning: In a finite *semigroup*, a left cancellable element need not be right cancellable.

b. Every nonzero element of the nonnegative integers with addition as operation is both a monomorphism and an epimorphism but not an isomorphism. (In other words, $m + n = m + p$ implies $n = p$ and similarly on the other side, but if $m \neq 0$ then m has no inverse.)

4. If $f : A \to B$ has a splitting $g : B \to A$, then in P, $A \leq B$ and $B \leq A$ so $A = B$. The only arrow from A to itself is id_A.

5. $h \circ g$ is an idempotent because $h \circ g \circ h \circ g = h \circ \mathrm{id} \circ g = h \circ g$. It is split because h and g fit the requirements for the h and g of the definition in the exercise mentioned: $h \circ g = h \circ g$ and $g \circ h = \mathrm{id}$.

6. (i) $\mathrm{Hom}(A, f)(g) = \mathrm{Hom}(A, f)(h)$ if and only if $f \circ g = f \circ h$ by definition of $\mathrm{Hom}(A, f)$. The result is the immediate from the definitions of injective and monomorphism.

(ii) $\mathrm{Hom}(f, D)(u) = \mathrm{Hom}(f, D)(v)$ if and only if $u \circ f = v \circ f$. Again, the result follows from the definition of injective and epimorphism.

(iii) Suppose f is a split monomorphism with splitting g, so $g \circ f = \mathrm{id}_B$. Suppose $u : B \to D$. Then $\mathrm{Hom}(f, D)(u \circ g) = u \circ g \circ f = u$, so $\mathrm{Hom}(f, D)$ is surjective. Conversely suppose $\mathrm{Hom}(f, D)$ is surjective for every object D. Then there is a $g : C \to B$ such that $\mathrm{hom}(f, B)(g) = \mathrm{id}_B$; that is, $g \circ f = \mathrm{id}_B$, so f is split.

(iv) Suppose f is a split epimorphism, with $f \circ g = \mathrm{id}_C$. Let $u : A \to C$. Then we have $\mathrm{Hom}(A, f)(g \circ u) = f \circ g \circ u = u$ so $\mathrm{Hom}(A, f)$ is surjective. Conversely, suppose $\mathrm{Hom}(A, f)$ is surjective for every A. Let $g : C \to B$ satisfy $\mathrm{Hom}(B, f)(g) = \mathrm{id}_C$, that is, $f \circ g = \mathrm{id}_C$, so f is a split epimorphism.

(v) An isomorphism is a split epimorphism and a split monomorphism because it is split by its inverse. Thus an isomorphism satisfies (a) and (b). In particular an isomorphism is an epimorphism and a monomorphism. It then follows from (i) through (iv) that an isomorphism satisfies (c) and (d). Conversely, suppose f is a split epi, split by $g : C \to B$, so $f \circ g = \mathrm{id}_C$. Then $f \circ g \circ f = f = f \circ \mathrm{id}_B$. If f is also mono then $g \circ f = \mathrm{id}_B$. Hence (a) implies that f is an isomorphism. That also follows from (b) by doing the same proof in the opposite category (note that the dual of the concept of isomorphism is isomorphism). Finally, (c) implies (a) by (i) and (iv) and (d) implies (b) by (ii) and (iii).

7. (i) Suppose an arrow $x : m \to n$ in $\mathscr{H}$ is not in the image of f_1. Let $\mathscr{U}$ be the graph with one node a and two arrows $u, v : a \to a$. Let $g : \mathscr{H} \to \mathscr{U}$ take all nodes to a and all arrows to u, and let h take all nodes to a and all arrows except x to u, with $h(x) = v$. Both g and h are graph homomorphisms. Then $g \circ f = h \circ f$ but $g \neq h$, showing that f is not epi.

If all the arrows of $\mathscr{H}$ are in the image of f_1 but there is a node n not in the image of f_0, then there can be no arrows with n as source or target. Let $\mathscr{V}$ be the graph with two nodes a and b and one arrow $u : a \to a$. Let $g : \mathscr{H} \to \mathscr{V}$ take all nodes to a and all arrows to u; let $h : \mathscr{H} \to \mathscr{V}$ take all nodes except n to a, n to b and all arrows to u. Again, g and h are graph homomorphisms and $g \neq h$ but $g \circ f = h \circ f$.

Conversely, suppose that f_0 and f_1 are both surjective. Let g, $h : \mathscr{H} \to \mathscr{K}$ satisfy $g \circ f = h \circ f$. Then for every node *or* arrow x of $\mathscr{H}$ there is a node or arrow m of $\mathscr{G}$ for which $f_i(m) = x$ ($i = 0$ if x is a node, $i = 1$ if x is an arrow). Then $g_i(x) = g_i(f_i(m)) = h_i(f_i(m)) = h_i(x)$, so $g = h$ and f is epi.

(ii) Suppose f is monic and suppose u and v are arrows of $\mathscr{G}$ for which $f_1(u) = f_1(v)$. Let $\mathscr{F}$ denote the graph with two nodes a and b and one arrow $x : a \to b$, and define $g : \mathscr{F} \to \mathscr{G}$ to take x to u and $h : \mathscr{F} \to \mathscr{G}$ to take x to v. What g and h do on nodes is then forced. Then $f \circ g = f \circ h$ but $g \neq h$, so f is not monic. If f is monic and f_1 is injective but there are nodes m and n of $\mathscr{G}$ for which $f_0(m) = f_0(n)$, then we carry out the same trick except that we take $\mathscr{F}$ to be the graph with one node and no arrows and let g and h take that node to m and n respectively.

Conversely, suppose f_0 and f_1 are injective. Suppose $g : \mathscr{F} \to \mathscr{G}$ and $h : \mathscr{F} \to \mathscr{G}$ are graph homomorphisms such that $f \circ g = f \circ h$. Let x be any node or arrow of $\mathscr{F}$. Then $f_i(g_i(x)) = f_i(h_i(x))$ ($i = 0$ if x is a node, $i = 1$ if x is an arrow), so because f_i is injective, $g_i(x) = h_i(x)$. Hence $g = h$ so f is monic.

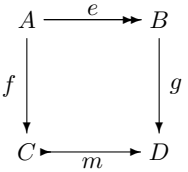
(iii) This is an immediate consequence of Exercise 3 of Section 1.4 and the fact that an isomorphism in the category of sets is a bijection.

8. a. Let $f : A \to B$, $g : B \to A$, $f \circ g = id_B$, so that f is split by g. Suppose that $f = m \circ h$ with m monic. We have $m \circ h \circ g = f \circ g = \mathrm{id}_B$. Hence $m \circ h \circ g \circ m = m$, so $h \circ g \circ m = \mathrm{id}_A$ since m is monic. Hence m is an isomorphism with inverse $h \circ g$.

b. In $\mathscr{C}$, if f is an epimorphism and $f = m \circ h$ with m monic, then m is epic by Proposition 2.9.4. Hence by the assumption, m is an isomorphism.

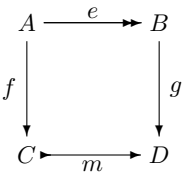
Section 2.10

1. FS–1: The requirement that $\mathscr{M} \circ \mathscr{I} \subseteq \mathscr{M}$ follows from the Shoe-Sock Theorem (Exercise 1 of Section 2.7) and $\mathscr{I} \circ \mathscr{E} \subseteq \mathscr{E}$ because arrows are closed under composition. FS–2: If $f : A \to B$ is an arrow, then $f = \mathrm{id}_B \circ f$, and id_B is an isomorphism. FS–3: Given the commutative square



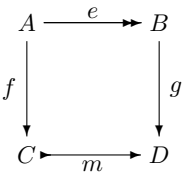
with $e \in \mathscr{E}$ and $m \in \mathscr{M}$, let $h = m^{-1} \circ g$. Then because the square commutes, $h \circ e = m^{-1} \circ g \circ e = m^{-1} \circ m \circ f = f$, and $m \circ h = m \circ m^{-1} \circ g = g$.

2. Suppose that every arrow in $\mathscr{M}$ is monic and that



is commutative. Suppose that $h : B \to C$, $h' : B \to C$ satisfy $m \circ h = m \circ h' = g$. Then $h = h'$ becuase m is monic. The other proof is dual.

3. Every isomorphism in **Set** is a bijection, hence an epimorphism and a monomorphism by Theorems 2.8.3 and 2.9.2. Given an arrow $f : A \to B$, let C denote the image of f, $e : A \to C$ the corestriction of f to C, and $i : C \to B$ the inclusion. Then $f = i \circ e$. Finally, suppose that



is a commutative square with $e \in \mathscr{E}$ and $m \in \mathscr{M}$. Define $h : B \to C$ as follows: If $b \in B$, then there is $a \in A$ for which $e(a) = b$ since e is surjective. Let $h(b) = f(a)$. We must show that this is well defined, that is, that if $e(a) = e(a') = b$ then $f(a) = f(a')$. Suppose that $e(a) = e(a') = b$. Then $m(f(a)) = g(e(a)) = g(e(a')) = m(f(a'))$ because the square commutes, so $f(a) = f(a')$ since m is monic. Finally, we must show that $h \circ e = f$ and $m \circ h = g$. The first is by definition. As for the second, we have $m \circ h \circ e = g \circ e = m \circ f = m \circ h \circ e$, so $m \circ h = g$ because e is an epimorphism.

4. If $h : \mathbf{Z} \to \mathbf{N}$ were a diagonal fill-in, then we must have $i(h(-1)) = -1$, but -1 is not in the image of i.

Solutions for Chapter 3

Section 3.1

1. Since functors preserve the operations of domain and codomain, the fact that $g \circ f$ is defined implies that the domain of g is the codomain of f. But then the domain of $F_1(g)$, which is F_0 applied to the domain of g, is the same as the codomain of $F_1(f)$. Hence $F_1(g) \circ F_1(f)$ is defined in $\mathscr{D}$.

2. The initial category has no objects and, therefore, no arrows. It clearly has exactly one functor to every other category. The terminal category has just one object and the identity arrow of that object. To any category $\mathscr{C}$ there is just one functor that takes every object to that single object and every arrow to that one arrow.

3. If $f : A \to B$ is a function between sets, the existential powerset functor takes f to f_* defined by $f_*(A_0) = \{f(a) \mid a \in A_0\}$. If $g : B \to C$, then $c \in (g_* \circ f_*)(A_0)$ if and only if there is an $a \in A_0$ such that $c = g(f(a))$. This is the same condition that $c \in (g \circ f)_*(A_0)$. It is evident that when f is the identity, so is f_*.

The universal powerset functor $f_!$ is defined by $b \in f_!(A_0)$ if and only if $b = f(a)$ implies that $a \in A_0$. If f is the identity, then $b = f(a)$ if and only if $b = a$ so that $f_!(A_0) = A_0$ (the identity arrow). If $f : A \to B$ and $g : B \to C$, then $c \in (g_! \circ f_!)(A_0)$ if and only if $c = g(b)$ implies that $b \in f_!(A_0)$, which is true if and only if $b = f(a)$ implies $a \in A_0$. Putting these together, we see that $c \in (g_! \circ f_!)(A_0)$ if and only if $c = f(g(a))$ implies $a \in A_0$. But this is the condition that $c \in (g \circ f)_!(A_0)$.

4. a. Since a functor is determined by its value on objects and arrows, a functor that is injective on objects and arrows is certainly a monomorphism. (Compare Exercise 7 of Section 2.9.) The other way is less obvious. If F is not injective on objects, say $C \neq C'$ but $F(C) = F(C')$ then let 1 be the category with one object and only the identity arrow. Let $G : 1 \to \mathscr{C}$ take that object to C, while $G' : 1 \to \mathscr{C}$ takes it to C'. Then $G \neq G'$, while $F \circ G = F \circ G'$, whence F is not a monomorphism. Next suppose F is injective on objects, but not on arrows. If $f \neq g$ are two arrows with $F(f) = F(g)$, then $F(f)$ and $F(g)$ have the same domain and the same codomain. Since F is injective on objects, it must be that f and g have the same domain and the same codomain. Say that $f, g : C \to C'$. Now let $\mathbf{2}$ be the category with two objects and one nonidentity arrow between them:

$$0 \to 1$$

Let $G : \mathbf{2} \to \mathscr{C}$ take 0 to C, 1 to C' and the nonidentity arrow to f, while G' is the same on objects, but takes the nonidentity arrow to g. Clearly $G \neq G'$, while $F \circ G = F \circ G'$.

b. The functor simply forgets the existence of the identity element. Two distinct monoids must differ in either their sets of elements or multiplication and in either case must differ as semigroups. Thus the functor is injective on objects. For similar reasons, it is injective on arrows. We have just seen that a functor that is injective on objects and arrows is a monomorphism.

5. a. It being evident that e is a two-sided identity, it is necessary only to show that the multiplication is associative. In any equation $x(yz) = (xy)z$ as soon as any of the variables is e, both sides reduce to a binary product of the remaining two terms (one or both of which might also be e). If none of them is e, then this is an equation involving terms from S and so is valid because it is in S.

b. If we denote by e_S and e_T the elements added to S and T respectively, then we define $F(f) = f^1 : S^1 \to T^1$ by

$$f^1(x) = \begin{cases} f(x) & \text{if } x \in S \\ e_T & \text{if } x = e_S \end{cases}$$

In verifying that $f^1(xy) = f^1(x)f^1(y)$ it is necessary to consider cases. If neither x nor y is e_S, then it follows from the fact that f is a homomorphism of semigroups. If, say, $x = e_S$, then both sides reduce to $f^1(y)$ and similarly if $y = e_S$. Finally, we must show that F is a functor. It is clear that if $f : S \to S$ is the identity, then $f^1 : S^1 \to S^1$ is the identity as well. It is also clear that if $g : T \to R$ is another monoid homomorphism, then

$$(g^1 \circ f^1)(x) = \begin{cases} (g \circ f)(x) & \text{if } x \in S \\ e_R & \text{if } x = e_S \end{cases} = (g \circ f)^1(x)$$

c. Since F is injective on objects and arrows, it is a monomorphism (see preceding exercise).

6. Define $\alpha : \mathrm{Hom}_{\mathbf{Mon}}(F(A), M) \to \mathrm{Hom}_{\mathbf{Set}}(A, U(M))$ by $\alpha(g)(a) = g(a)$. The fact that $(\alpha(\beta(f)))(a) = f(a)$ is clear. On the other hand, if $g : F(A) \to M$ is a monoid homomorphism, then

$$\begin{aligned}(\beta(\alpha(g)))(a_1, a_2, \cdots, a_n) &= \alpha(g)(a_1)\alpha(g)(a_2)\cdots\alpha(g)(a_n) \\ &= g(a_1)g(a_2)\cdots g(a_n) = g(a_1, a_2, \cdots, a_n)\end{aligned}$$

the last equality coming from the fact that g is a monoid homomorphism. Thus β is invertible and hence bijective.

7. In Exercise 5, we showed that $\gamma(h)$ (which would have been called h^1 there) is a monoid homomorphism. To see that γ is a bijection, we define

$$\delta : \mathrm{Hom}_{\mathbf{Mon}}(F(S), M) \to \mathrm{Hom}_{\mathbf{Sem}}(S, U(M))$$

by $\delta(g)(x) = g(x)$ for $x \in S$. Since a monoid homomorphism is also a semigroup homomorphism, this is well defined. Clearly $\delta(\gamma(h))(x) = h(x)$ for $x \in S$ and $h : S \to U(M)$. To go the other way, suppose $g : F(S) \to M$. For $x \in S$, $\gamma(\delta(g))(x) = \delta(g)(x) = g(x)$. For e_S, we have $\gamma(\delta(g))(e_S) = 1$ by definition of γ, but $g(e_S) = 1$ since g is a monoid homomorphism. Thus $\delta = \gamma^{-1}$ and γ is a bijection.

8. This is an immediate consequence of Exercises 4.b and 5.c.

9. It is obvious that **Set** is a subcategory of the category of partial functions. To get a functor in the other direction, let $F(S) = S \cup \{S\}$. If $f : S \to T$ is a partial function, let $F(f) : F(S) \to F(T)$ be the total function defined by

$$F(f)(x) = \begin{cases} f(x) & \text{if } x \in S \text{ and } f(x) \text{ is defined} \\ T & \text{if } x = S \text{ or } f \text{ is not defined at } x \end{cases}$$

It is clear that if $f \neq g$, then $F(f) \neq F(g)$.

10. On the one hand, if $\mathscr{A}$ is discrete, for any function $F_0 : \mathscr{A}_0 \to \mathscr{B}_0$ there is a unique functor $F : \mathscr{A} \to \mathscr{B}$ whose value at the identity of some object A of $\mathscr{A}$ is the identity of $F_0(A)$. To go the other way, suppose $\mathscr{A}$ is not discrete. Suppose first that there is an arrow $f : A \to B$ in $\mathscr{A}$ with $A \neq B$. Let $\mathscr{B}$ be the category with the same objects as $\mathscr{A}$, but with no nonidentity arrows. Then the identity function $F_0 : \mathscr{A}_0 \to \mathscr{B}_0$ cannot be extended to a functor since there is nowhere to send f. Now suppose that all arrows of $\mathscr{A}$ are endoarrows. Let $\mathscr{B}$ have the same objects as $\mathscr{A}$ and let

$$\mathrm{Hom}_{\mathscr{B}}(A, A) = \mathrm{Hom}_{\mathscr{A}}(A, A) \times \mathrm{Hom}_{\mathscr{A}}(A, A)$$

for an object A of $\mathscr{A}$. The identity function $F_0 : \mathscr{A}_0 \to \mathscr{B}_0$ can be extended in at least two ways to a functor. The first way is to take $F(f) = (f, \mathrm{id}_A)$ for $f : A \to A$ in $\mathscr{A}$ and the second is $F'(f) = (\mathrm{id}_A, f)$.

11. We claim that a category is indiscrete if and only if there is exactly one arrow between any two objects. It is obvious that such a category is indiscrete. To go the other way, first suppose that $\mathscr{A}$ has two objects A and B with no arrows between them, then $\mathscr{A}$ cannot be indiscrete. For if $\mathbf{2}$ denotes the category with two objects 0 and 1 and one arrow between them, then the object function that takes 0 to A and 1 to B cannot be extended. If there is more than one arrow from A to B, then the same object function on $\mathbf{2}$ has more than one extension to a functor.

Section 3.2

1. If M acts on S, let $\phi : M \to FT(S)$ be defined by $\phi(a)(x) = ax$ for $a \in M$ and $x \in S$. We have $\phi(1)(a) = 1a = a = \mathrm{id}(a)$ so that $\phi(1) = \mathrm{id}$. Also, for $a, b \in M$,

$$\phi(ab)(x) = (ab)x = a(bx) = \phi(a)(\phi(b)(a))$$

so that $\phi(ab) = \phi(a) \circ \phi(b)$. Conversely, if $\phi : M \to FT(S)$, then let M act on S by letting $ax = \phi(a)(x)$. The computations above can be reversed to show that since ϕ is a monoid homomorphism, the M-set identities are satisfied. It is clear that these processes are inverse to each other.

Section 3.3

1. a. Since a functor is faithful if it is injective between hom sets and a monoid has only one hom set, such a functor is faithful if and only if it is injective.

b. Since hom sets are either singleton or empty and a function on such a set is always injective, such a functor is always faithful.

2. A functor is full if it is surjective on hom sets so a functor between monoids is full if and only if it is surjective. As for posets, a functor $f : P \to Q$ between posets is full if and only if whenever $f(x) \leq f(y)$, then $x \leq y$.

3. No. For example let $(\mathbf{N}, +)$ and $(\mathbf{N}, *)$ denote the monoids of integers with the operations of addition and multiplication, respectively. The function $f : (\mathbf{N}, +) \to (\mathbf{N}, *)$ that is constantly 0 is a semigroup homomorphism that is not a monoid homomorphism since it does not preserve the identity.

4. It is faithful, but not full. Certainly, if $f \neq g : S \to T$, then $F(f) \neq F(g) : S^* \to T^*$ since on strings of length 1, $F(f)$ is essentially the same as f. On the other hand, there are infinitely many homomorphisms from $\mathbf{N} = F(1)$ to itself (take the generating element to any power of itself), but only one function from $\{1\}$ to itself.

5. It is faithful because f can be recovered from $\mathscr{P}(f)$ by its actions on singletons. On the other hand, it cannot be full since, for example, there is only one function from 1 to 1 and four from $\mathscr{P}(1)$ to $\mathscr{P}(1)$.

6. The isomorphism is the functor from $\mathbf{Rel}^{\,\mathrm{op}}$ to **Rel** that takes every object to itself and every relation $R \subseteq A \times B$ from A to B in **Rel** (hence from B to A in $\mathbf{Rel}^{\,\mathrm{op}}$) to the relation $R^{\mathrm{op}} = \{(b, a) \mid (a, b \in R\}$. The inverse functor is this same functor considered as going from **Rel** to $\mathbf{Rel}^{\,\mathrm{op}}$.

7. Let $\mathscr{G}$ be a groupoid. The isomorphism $F : \mathscr{G}^{\mathrm{op}} \to \mathscr{G}$ is the function that takes objects to themselves and an arrow $f : X \to Y$ in $\mathscr{G}$ (hence $f : Y \to X$ in $\mathscr{G}^{\mathrm{op}}$) to $f^{-1} : Y \to X$ in $\mathscr{G}$. F preserves composition by the Shoe-Sock Theorem (Exercise 1, page 45.) Since $F(F(f)) = (f^{-1})^{-1} = f$ (see 2.7.3), it follows that $F : \mathscr{G}^{\mathrm{op}} \to \mathscr{G}$ is the inverse of $F : \mathscr{G} \to \mathscr{G}^{\mathrm{op}}$, so that F is an isomorphism.

8. A functor from a monoid to a category takes the single object of the monoid to some object of the category and is a monoid homomorphism from the monoid to the monoid of endoarrows of that object. In other words, its arrow part is a monoid homomorphism. Therefore, if $i : \mathbf{N} \to \mathbf{Z}$ is the inclusion, the nonexistence of monoid homomorphisms $g \neq h$ such that $g \circ i = h \circ i$ implies that there also cannot exist functors with that property.

9. Suppose $f : A \to B$ is a split mono in a category. Then there is a $g : B \to A$ with $g \circ f = \mathrm{id}_A$. For any functor F, $F(g) \circ F(f) = F(g \circ f) = F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$, since functors preserve composition and identities. The argument for split epis is dual.

10. a. There is exactly one semigroup structure on the empty set and also on the one point set and these are the initial, respectively terminal semigroups. Thus the underlying functor preserves and reflects both initial and terminal objects.

b. There is exactly one category structure on the empty graph and that is the initial category, so the underlying functor preserves and reflects the initial object.

The terminal category has one object and one arrow, the identity. This is the unique category structure on the graph with one object and one arrow and that is the terminal graph. Thus the terminal object is also preserved and reflected.

11. T is a terminal object if and only if there is exactly one arrow from any object to T. In particular, $\mathrm{Hom}(A,T)$ is a singleton set, which is a terminal object in **Set**.

12. Let $F : \mathscr{C}/B \to (\mathscr{C}/A)/f$ send $u : C \to B$ to $u : f \circ u \to f$ as suggested. If $v : D \to B$ is another object of $\mathscr{C}/B$ and $g : u \to v$ is an arrow (which is to say that $v \circ g = u$), then $f \circ v \circ g = f \circ u$ so that we can simply let $F(g : u \to v) = g : F(u) \to F(v)$. Let $G : (\mathscr{C}/A)/f \to \mathscr{C}/B$ take an object $w : (g : C \to A) \to f$ to $w : C \to B$. Suppose $w' : g' \to f$ is also an arrow of $\mathscr{C}/A$ (hence an object of $(\mathscr{C}/A)/f$) and $x : w' \to w$ is an arrow of $(\mathscr{C}/A)/f$. Then by definition of arrow in a slice category, $w \circ x = w' : g' \to f$ in $\mathscr{C}/A$, hence in $\mathscr{C}$. Thus we can set $G(x : w' \to w) = x : \mathrm{dom}(w') \to \mathrm{dom}(w)$. It is straightforward to check that F and G are functors. It is clear that they are inverse to each other.

Section 3.4

1. Let $F : \mathscr{C} \to \mathscr{D}$ be an isomorphism with inverse G. Then G is a pseudo-inverse for F. We define the arrows u_C required by E–2 to be id_C for each object C, and similarly $v_C = \mathrm{id}_C$. Since $G(F(f)) = f$ and $F(G(g)) = g$ for all arrows f of $\mathscr{C}$ and g of $\mathscr{D}$, requirements E–2 and E–3 are satisfied.

2. Define $F : \mathbf{Pfn} \to \mathbf{Pts}$ by $F(S) = (S \cup \{S\}, S)$. S is chosen as the additional element to guarantee that it is not already an element of S. If $f : S \to T$ is a partial function, let $F(f)$ be defined by

$$F(f)(x) = \begin{cases} f(x) & \text{if } x \in S \text{ and } f \text{ is defined at } x \\ T & \text{otherwise} \end{cases}$$

It is obvious that F preserves identities. As for composition, if $g : T \to R$ is a partial function, then $g(f(x))$ is defined if and only if f is defined at x and g is defined at $f(x)$, which is exactly when $g \circ f$ is defined at x. From this, it is immediate that F preserves composition.

We define a pseudo-inverse $G : \mathbf{Pts} \to \mathbf{Pfn}$ by $G(S,s) = S - \{s\}$ and if $f : (S,s) \to (T,t)$ is an arrow, then

$$G(f)(x) = \begin{cases} f(x) & \text{if } f(x) \neq t \\ \text{undefined} & \text{if } f(x) = t \end{cases}$$

It is easy to show that G is a functor. Now if S is a set, it is clear that $G(F(S)) = S$ and $(S,s) \cong F(G(S,s))$ by an isomorphism v_S which is the identity on S and takes s to S (the latter being the added element of $F(S - \{s\})$). What has to be shown is that for any partial function $f : S \to T$ and any function $g : (S,s) \to (T,t)$, E–2 and E–3 hold. These are a simple matter of considering cases and we omit them.

3. Let $\mathscr{P}$ be the category of preordered sets and $\mathscr{Q}$ the category of small categories as described. We let $F : \mathscr{P} \to \mathscr{Q}$ take a preordered set $(P, \leq)$ to the category $C(P, \leq)$ as described in 2.3.1. If $f : P \to P'$ is a monotone function, then $F(f)$ agrees with f on objects and when $a \leq b$, then we let $F(f)(b,a) = (f(b), f(a))$. In

the other direction, let Q be a category in $\mathscr{Q}$ and let $G(Q)$ be the preordered set whose elements are the objects of Q with the preorder that $a \leq b$ if there is an arrow $a \longrightarrow b$. The composition law in the category makes this relation transitive and the identity makes it reflexive. If $f : Q \longrightarrow Q'$ is a functor in $\mathscr{Q}$, let $G(f)$ be the object function of f. $G(f)$ is monotone because if $x \leq y$ in Q then f must take the corresponding arrow to an arrow in Q'. It is clear that $G \circ F$ is the identity, so that E–2 is satisfied with $u_C = \mathrm{id}_C$. $F \circ G$ is the identity on the objects and that there will be an arrow $a \longrightarrow b$ in Q if and only if $a \leq b$ in $G(Q)$ if and only if there is an arrow $a \longrightarrow b$ in $F(G(Q))$. Thus there is an isomorphism $v_C : Q \longrightarrow F(G(Q))$ which is the identity on objects and which takes an arrow $a \longrightarrow b$ in Q to (the only) arrow $a \longrightarrow b$ in $F(G(Q))$. These isomorphisms must satisfy E–3 because for any arrow g there is only one arrow that $v_{D'} \circ g \circ v_D^{-1}$ can be.

4. There is an functor $F : \mathscr{M} \longrightarrow \mathscr{L}$ that takes n to the space of n-rowed column vectors and a matrix $A : m \longrightarrow n$ to the linear transformation of multiplying on the left by A. In case one of the numbers is 0, there is only the 0 linear transformation between them. It is well known that this is a functor (matrix multiplication corresponds to composition of linear transformations) which is full and faithful and that every finite dimensional vector space is isomorphic to a space of column vectors. A pseudo-inverse G is found by choosing, for each space V, a basis B and then letting $G(V)$ be the number of elements of B. If V' is another space with chosen basis B' and $T : V \longrightarrow V'$ is a linear transformation, then $G(T)$ is the matrix of T with respect to B and B'. Although G is uniquely defined on objects, its value on arrows depends completely on the choice, for each vector space V, of a basis for that space. In this case u_m is the $m \times m$ identity matrix, so E–2 is satisfied, and E–3 follows from the definition of the linear transformation determined by a matrix, given a basis.

5. a. Let $F : \mathbf{Mon} \longrightarrow \mathbf{Ooc}$ take the monoid M to the category with one object and with M as its set of endoarrows. If $f : M \longrightarrow N$ is a monoid homomorphism, then $F(f)(M) = N$ on the object and $F(f)(a) = f(a)$ for $a \in M$. It is immediate that F is a functor. Let $G : \mathbf{Ooc} \longrightarrow \mathbf{Mon}$ take the one-object category C to the object of endomorphisms of that single object. It is clear that $G \circ F$ is the identity, so E–2 holds. $F \circ G$ is the identity on the arrows of the category and is evidently the only possible isomorphism on the singleton set of objects, so E–3 holds because there is no choice possible for the arrow.

b. Suppose $f, g : M \longrightarrow N$ are homomorphisms of monoids. Then since $F(f)$ agrees with f on the arrows of $F(M)$, $F(f) = F(g)$ implies $f = g$. This shows directly that F is faithful. To see it is full, let $h : F(M) \longrightarrow F(N)$ be a functor. Then the arrow function of h is a monoid homomorphism f from the monoid of endomorphisms of the object of $F(M)$ to the object of $F(N)$. $F(f)$ agrees with h on arrows and certainly does on objects since there is no choice.

Section 3.5

1. Suppose that both CR–1 and CR–2 of 3.5.1 are satisfied. Then special cases result from letting h or k be an identity. Using these special cases, we have that

$$g_1 \circ f_1 \sim g_2 \circ f_1 \sim g_2 \circ f_2$$

On the other hand, special cases of the diagram here result by setting $f_1 = f_2$ or $g_1 = g_2$. Using them, we have that $f \sim g$ implies that $k \circ f \sim k \circ g$ and that $f \circ g \sim f \circ h$.

2. Suppose that $\sim_1$ and $\sim_2$ are congruences. The intersection of two equivalence relations is an equivalence relation on any set. Also if $f_1 \sim_1 f_2$ and $g_1 \sim_1 g_2$ implies that $f_1 \circ g_1 \sim_1 f_2 \circ g_2$ and if $f_1 \sim_2 f_2$ and $g_1 \sim_2 g_2$ implies that $f_1 \circ g_1 \sim_2 f_2 \circ g_2$, then for $\sim = \sim_1 \cap \sim_2$, $f_1 \sim f_2$ and $g_1 \sim g_2$ imply that $f_1 \circ g_1 \sim f_2 \circ g_2$.

3. A functor F is full if every arrow in $\mathrm{Hom}(F(A), F(B))$ has the form $F(f)$ for some $f : A \longrightarrow B$. A quotient as described here is surjective on arrows, so the condition of fullness is certainly satisfied.

4. a. It is an equivalence because $F(f) = F(f)$, $F(f) = F(g)$ implies $F(g) = F(f)$ and $F(f) = F(g)$ and $F(g) = F(h)$ implies $F(f) = F(h)$. To see that it is a congruence, suppose $F(f_1) = F(f_2)$ and $F(g_1) = F(g_2)$. Then

$$F(f_1 \circ g_1) = F(f_1) \circ F(g_1) = F(f_2) \circ F(g_2) = F(f_2 \circ g_2)$$

The result follows from Exercise 1.

b. Suppose $[f], [g] : A \longrightarrow B$ are arrows of $\mathscr{C}/\!\sim$ such that $F_0([f]) = F_0([g])$. Then $F(f) = F(g)$, whence $f \sim g$ and $[f] = [g]$.

c. $F = F_0 \circ Q$ and we just seen that F_0 is faithful and that Q is full (previous exercise).

5. It follows from 2.2.5(ii) that $\neg\,\mathrm{true} = \mathrm{false}$ and $\neg\,\mathrm{false} = \mathrm{true}$. Thus

$$\widehat{F}(\neg\,\mathrm{true}) = \widehat{F}(\mathrm{false}) \qquad \text{and} \qquad \widehat{F}(\neg\,\mathrm{false}) = \widehat{F}(\mathrm{true})$$

Similarly, $\widehat{F}(\texttt{chr} \circ \texttt{ord}) = \mathrm{id}$. Thus $f \sim g$ implies $\widehat{F}(f) = \widehat{F}(g)$ for the generators of the congruence. The set of pairs $\{(f, g)\}$ for which $\widehat{F}(f) = \widehat{F}(g)$ thus includes the generators and is closed under composition since functors preserve composition. Thus it includes all pairs for which $f \sim g$.

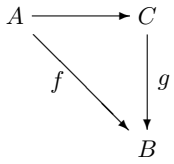
6. a. Such a relation satisfies Definition 3.5.1(a) automatically, since all elements of M are arrows of $C(M)$ with the same domain and codomain. The definition forces it to satisfy (b).

b. Suppose K is a submonoid and $n \sim n'$. Then (m, m) and (n, n') are both in K and $(m, m)(n, n') = (mn, mn')$ so $mn \sim mn'$. Similarly, $nm \sim n'm$. Conversely, suppose $\sim$ is a congruence on M. Let (m, m') and (n, n') be elements of K. Then $m \sim m'$ and $n \sim n'$, so by Exercise 1, $(mn, m'n') \in K$. Finally, $(1, 1) \in K$ because the relation is reflexive. Thus K is a submonoid of $M \times M$.

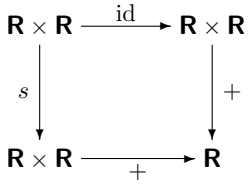
Solutions for Chapter 4

Section 4.1

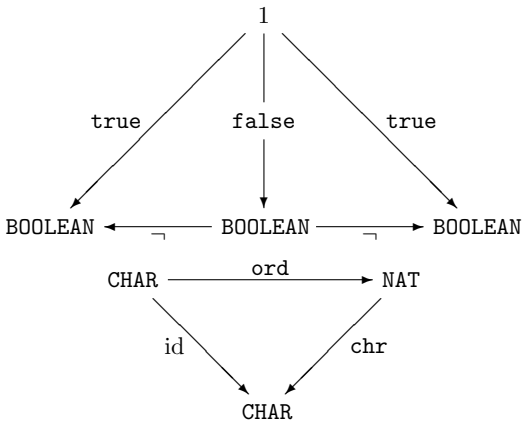
1.



2. Let s be the function $(x, y) \mapsto (y, x)$. Then the diagram is

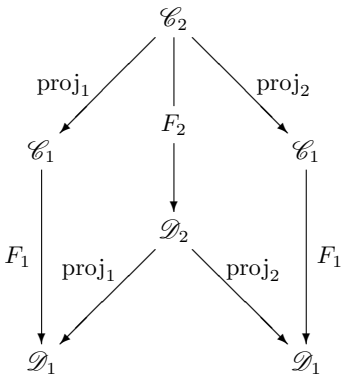


3.

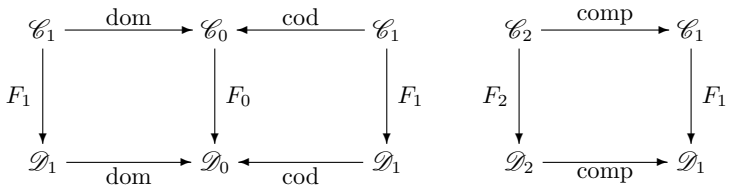


4. Let $\mathscr{C}$ and $\mathscr{D}$ be categories with sets of objects $\mathscr{C}_0$ and $\mathscr{D}_0$, sets of arrows $\mathscr{C}_1$ and $\mathscr{D}_1$, and sets of composable pairs of arrows $\mathscr{C}_2$ and $\mathscr{D}_2$, respectively. A functor $F : \mathscr{C} \to \mathscr{D}$ consists of functions $F_0 : \mathscr{C}_0 \to \mathscr{D}_0$, $F_1 : \mathscr{C}_1 \to \mathscr{D}_1$ along with the

uniquely determined function $F_2 : \mathscr{C}_2 \to \mathscr{D}_2$ such that



commutes. In addition, the following diagrams must commute:



Section 4.2

1. It is the model M defined by $M(\mathsf{n}) = \{\mathsf{n}, \mathsf{a}\}$, $M(\mathsf{a}) = \{\mathsf{source}, \mathsf{target}\}$,

$$M(\mathsf{source})(\mathsf{source}) = M(\mathsf{source})(\mathsf{target}) = \mathsf{a}$$

and

$$M(\mathsf{target})(\mathsf{source}) = M(\mathsf{target})(\mathsf{target}) = \mathsf{n}$$

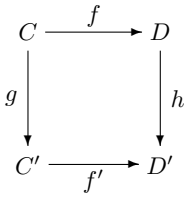
2. Define an isomorphism ϕ : **u-Struc** $\to$ **Mod**$(\mathscr{U}, \mathbf{Set})$ as follows (these are the constructions in 4.2.15). If (S, f) is a u-structure, $\phi(S, f)$ is the model of $\mathscr{U}$ that takes u_0 to S and e to f. If $h : (S, f) \to (T, g)$ is a homomorphism, then $\phi(h)$ is the natural transformation whose only component (at u_0) is h. The inverse of ϕ takes a model M to $M(u_0)$ and a natural transformation γ to its only component.

An isomorphism ψ : **u-Struc** $\to$ $\mathbf{N}$-**Act** can be defined this way: If (S, f) is a u-structure, the action $\alpha : \mathbf{N} \times S \to S$ is defined by $\alpha(k, x) = f^k(x)$, where f^k denotes $f \circ f \circ \ldots \circ f$ (k occurrences of f) as in 2.3.5. This is indeed an action, since $\alpha(0, x) = f^0(x) = x$ and

$$\alpha(k + k', x) = f^{k+k'}(x) = f^k(f^{k'}(x)) = \alpha(k, \alpha(k', x))$$

The inverse to ψ takes an action $\alpha : \mathbf{N} \times S \to S$ to the u-set (S, f), where $f(x) = \alpha(1, x)$.

3. The arrow category has as objects arrows $f : C \to D$ and an arrow from $f : C \to D$ to $f' : C' \to D'$ is a pair of arrows (g, h) where $g : C \to C'$ and $h : D \to D'$ are such that



commutes. The slice $\mathscr{C}/B$ is the subcategory that consists of those objects $f : C \to D$ for which $D = B$ and those arrows (g, h) for which $h = \mathrm{id}_B$. Since it does not include all arrows between objects of the arrow category, it is not full (nor, since it does not include all the objects, is it wide).

4. An object of $\mathbf{Mod}(\mathscr{G}, \mathscr{C})$ is given by a graph homomorphism $\mathscr{G} \to \mathscr{C}$. Since $\mathscr{G}$ has two nodes such a homomorphism is given by a pair of objects of $\mathscr{C}$. Since there are no arrows in $\mathscr{G}$, such a pair of arrows is exactly a graph homomorphism so the objects of $\mathbf{Mod}(\mathscr{G}, \mathscr{C})$ are the pairs. An arrow from one pair to another is a pair of arrows as in $\mathscr{C} \times \mathscr{C}$, which are generally subject to commutativity conditions corresponding to arrows of $\mathscr{G}$. Since $\mathscr{G}$ has no arrows, there are no conditions in this case.

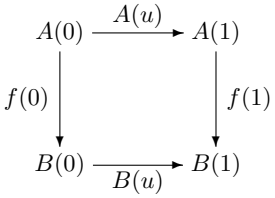
5. If $(h, k) : f \to g$ is an isomorphism with inverse (h', k'), then we have $(h \circ h', k \circ k') = (h, k) \circ (h', k') = \mathrm{id}_g = (\mathrm{id}_C, \mathrm{id}_D)$ so that $h \circ h' = \mathrm{id}_C$ and $k \circ k' = \mathrm{id}_D$. Similarly, $h' \circ h = \mathrm{id}_A$ and $k' \circ k =^{-1} \mathrm{id}_B$. Conversely, suppose h and k are invertible. Then we must show that (h^{-1}, k^{-1}) is an arrow from $g \to f$. We have

$$f \circ h^{-1} = k^{-1} \circ k \circ f \circ h^{-1} = k^{-1} \circ g \circ h \circ h^{-1} = k^{-1} \circ g$$

It is evident that $(h^{-1}, k^{-1}) = (h, k)^{-1}$.

6. Let $f : D \to D'$ be an arrow of $\mathscr{D}$. There are objects C and C' of $\mathscr{C}$ and isomorphisms $h : D \to F(C)$ and $k : D' \to F(C')$. The arrow $k \circ f \circ h^{-1} : F(C) \to F(C')$ is $F(g)$ for a unique $g : C \to C'$ because an equivalence is full and faithful. Then $k \circ f = F(g) \circ h$, which means that (h, k) is an arrow from f to $F(g)$. Since h and k are isomorphisms, so is (h, k) (see previous exercise).

7. An object of $\mathscr{C}^{\to}$ is an arrow $A(u) : A(0) \to A(1)$ of $\mathscr{C}$ and an arrow $(f(0), f(1)) : A(u) \to B(u)$ is a commutative square



A functor $A : \mathscr{G} \to \mathscr{C}^{\to}$ is thus required to produce, for each node a of $\mathscr{G}$ an arrow $A(a,u) : A(a,0) \to A(a,1)$ and to each arrow $s : a \to b$ a commutative diagram

$$\begin{array}{ccc} A(a,0) & \xrightarrow{A(a,u)} & A(a,1) \\ {\scriptstyle A(s,0)}\downarrow & & \downarrow{\scriptstyle A(s,1)} \\ A(b,0) & \xrightarrow[A(b,u)]{} & A(b,1) \end{array}$$

If we let $E(a) = A(a,0)$, $E(s) = A(s,0)$, one sees immediately that $E : \mathscr{G} \to \mathscr{C}$ is a model. Similarly, if we let $F(a) = A(a,1)$ and $F(s) = A(s,1)$ it is also a model. Finally, define $\alpha : E \to F$ by $\alpha(a) = A(a,u)$. The commutativity above implies that α is a natural transformation. Conversely, we can start with a natural transformation $\alpha : E \to F$ between two models and use the equations above to define A. The details are trivial.

8. We have to show that the naturality condition engendered by an arrow $f : a \to b$ in $\mathscr{G}$ is satisfied by β, given that it is satisfied by α. We have

$$D(f) \circ \beta a = \beta b \circ \alpha b \circ D(f) \circ \beta a = \beta b \circ E(f) \circ \alpha a \circ \beta a = \beta b \circ E(f)$$

(Compare this argument with that of Exercise 5 above.)

Section 4.3

1. Recall that if $\mathscr{G}$ is graph, $\eta\mathscr{G}$ is the identity on objects and is defined on arrows by $\eta\mathscr{G}[u] = (u)$, where we use square brackets to denote application to distinguish it from the parentheses used for lists. If $f : \mathscr{G} \to \mathscr{H}$ is a graph homomorphism, we have to show that

$$\begin{array}{ccc} \mathscr{G} & \xrightarrow{\eta\mathscr{G}} & U(F(\mathscr{G})) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle U(F(f))} \\ \mathscr{H} & \xrightarrow[\eta\mathscr{H}]{} & U(F(\mathscr{H})) \end{array}$$

commutes. Applied to objects, we get $\eta\mathscr{H} \circ f[a] = f[a]$, while

$$U(F(f)) \circ \eta\mathscr{G}[a] = U(F(f))[a] = f[a]$$

If $u : a \to b$ is an arrow, $\eta\mathscr{H} \circ f[u] = (f[u])$, while

$$U(F(f)) \circ \eta\mathscr{G}[u] = U(F(f))[(u)] = (f[u])$$

2. We must show that if $f : \mathscr{G} \to \mathscr{H}$ is a graph homomorphism,

$$\begin{array}{ccc} \mathscr{G} & \xrightarrow{\beta\mathscr{G}} & W(\mathscr{G}) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle W(f)} \\ \mathscr{H} & \xrightarrow[\beta\mathscr{H}]{} & W(\mathscr{H}) \end{array}$$

commutes. But for a node a of $\mathscr{G}$, $W(f)(\beta\mathscr{G}(a))$ is calculated as the component of $\mathscr{H}$ containing $f(a)$, which is exactly what $\beta\mathscr{H}(f(a))$ is. In other words, the naturality is the very definition of $W(f)$.

3. We must show that for each arrow $f : C \to C'$ of $\mathscr{C}$,

$$\begin{array}{ccc} G(C) & \xrightarrow{i_C} & F(C) \\ {\scriptstyle G(f)}\downarrow & & \downarrow{\scriptstyle F(f)} \\ G(C') & \xrightarrow[i_{C'}]{} & F(C') \end{array}$$

commutes. We have, for $x \in G(C)$,

$$F(f)((i_C)(x)) = F(f)(x) = G(f)(x) = i_{C'}(G(f)(x))$$

4. We must show that for any graph homomorphism $f : \mathscr{G} \to \mathscr{H}$, the square

$$\begin{array}{ccc} A(\mathscr{G}) & \xrightarrow{\text{source}\,\mathscr{G}} & N(\mathscr{G}) \\ {\scriptstyle A(f)}\downarrow & & \downarrow{\scriptstyle N(f)} \\ A(\mathscr{H}) & \xrightarrow[\text{source}\,\mathscr{H}]{} & N(\mathscr{H}) \end{array}$$

commutes. We have for $u : a \to b$ in $\mathscr{G}$, $N(f)(\text{source}\,\mathscr{G}(u)) = N(f)(a) = f(a)$, while $\text{source}\,\mathscr{H}(A(f)(u)) = \text{source}(f(u)) = f(a)$. The argument for target is similar.

5. A functor $F : \mathscr{C} \to \mathbf{Set}$ is determined uniquely by giving, for each $C \in \mathscr{C}_0$, a set $F(C)$. The disjoint union of these sets can be modeled as $W(F) = \bigcup\{(x, C) \mid x \in F(C)\}$. The function $w(F) : W(F) \to \mathscr{C}_0$ defined by $w(F)(x, C) = C$ makes $w(F)$ an object of $\mathbf{Set}/\mathscr{C}_0$. Given a natural transformation $\alpha : F \to G$, define $W(\alpha) : W(F) \to W(G)$ by $W(\alpha)(x, C) = (\alpha C(x), C)$. This makes $W : \mathbf{Func}(\mathscr{C}, \mathbf{Set}) \to \mathbf{Set}/\mathscr{C}_0$ a functor.

Conversely, given an object $f : S \to \mathscr{C}_0$ of $\mathbf{Set}/\mathscr{C}_0$, define a functor $V(f) : \mathscr{C} \to \mathbf{Set}$ by $V(f)(C) = \{x \in S \mid f(x) = C\}$ for an object C of $\mathscr{C}$. If $u : f \to g : S' \to \mathscr{C}_0$ is an arrow of $\mathbf{Set}/\mathscr{C}_0$, define a natural transformation $V(u) : V(f) \to V(g)$ by $(V(u)C)(x) = u(x)$ for $x \in S'$. Then $V : \mathbf{Set}/\mathscr{C}_0 \to \mathbf{Func}(\mathscr{C}, \mathbf{Set})$ is a functor.

We have that $V(W(F))(C) = \{(x, C) \mid x \in F(C)\}$. Let $\beta C(x, C) = x$ for $x \in F(C)$. Then β is a natural isomorphism from $V \circ W$ to the identity functor on $\mathbf{Func}(\mathscr{C}, \mathbf{Set})$. In the other direction, for $f : S \to \mathscr{C}_0$,

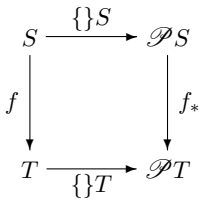
$$W(V(f)) = \bigcup\{(x, C) \mid x \in V(f)(C)\} = \{(x, f(x)) \mid x \in S\}$$

which is isomorphic to S. The diagram

$$\begin{array}{ccc} & S & \\ {\scriptstyle \cong}\swarrow & & \searrow{\scriptstyle f} \\ W(V(f)) & \xrightarrow[w(V(f))]{} & \mathscr{C}_0 \end{array}$$

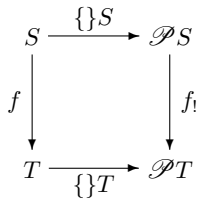
commutes and the naturality of the isomorphism is straightforward to verify.

6. a. We must show that if $f : S \to T$ is a function, then



commutes. But each path applied to an $x \in S$ produces $\{f(x)\}$.

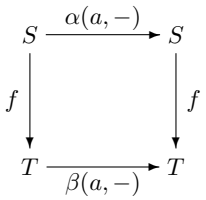
b. The diagram that would have to commute is



for an $f : S \to T$. The reader may want to verify that it does if and only if f is injective. When, for example, $f : 2 \to 1$ is the unique function, going around counter-clockwise gives, at 0, the element $\{0\}$. Going the other way, we get $f_!(\{0\})$ which is the set of $x \in 1$ whose *every* inverse image is included in $\{0\}$. Since no element of 1 has this property, $f_!\{0\} = \emptyset$.

c. It makes no sense to ask for a natural transformation between a contravariant functor and a covariant functor.

7. A natural transformation from $F \to G$ is a function f from $S = F(M)$ to $T = G(M)$ and it must satisfy the naturality condition that for any $a \in M$,



commutes. This is the definition of an equivariant function.

8. This refers to the answer to Exercise 7 of Section 4.2. In addition to that, in describing a model $A : \mathscr{D} \to \mathscr{C}^{\to}$ we must add, whenever we have arrows $s : a \to b$ and $t : b \to c$ of $\mathscr{D}$, the equations $A(t,0) \circ A(s,0) = A(t \circ s, 0)$ and $A(t,1) \circ A(s,1) = A(t \circ s, 1)$. Similar equations have to be added for E and F and will satisfied if and only if they are for A.

9. If $F, G : \mathscr{E} \to \mathscr{G}$ is such that $\alpha(F) = \alpha(G)$, then the sole vertex of $\alpha(F)$ is the same as that of $\alpha(G)$. But that is all there is to a homomorphism on $\mathscr{E}$. Thus α is injective. Similarly, every node of $\mathscr{G}$ does give graph homomorphism on $\mathscr{E}$ so α is surjective.

10. a. If $f : \mathscr{G} \to \mathscr{H}$ is a graph homomorphism, we define

$$P_k(f)((u_n, u_{n-1}, \ldots, u_1)) = (f(u_n), f(u_{n-1}), \ldots, f(u_1))$$

This makes sense since f preserves source and target. The functoriality is clear.

b. A path of length 0 goes from a node n to itself (if you don't go anywhere, you stay in the same place!). The natural isomorphism takes a path of length 0 to the node n that it determines. The naturality condition is trivial.

c. Since a path of length 1 is an arrow, the object functions of P_1 and A are the same. The arrow functions are the same by definition. Thus the components of the required natural isomorphism are identity functions.

Section 4.4

1. By Definition 4.4.2, the component of $(G_1\alpha)E$ at an object A of $\mathscr{A}$ is the arrow $(G_1\alpha)E(A)$. This is $G_1(\alpha E(A))$ by Definition 4.4.3. By Definition 4.4.3, the component of $G_1(\alpha E)$ at A is $G_1((\alpha E)A)$. This is $G_1(\alpha E(A))$ by Definition 4.4.2.

2. The horizontal composites $\beta * \mathrm{id}_F$ and $\mathrm{id}_H * \alpha$ are defined to be the two equal composites in each of these special cases of Diagram (4.30):

$$\begin{array}{ccc}
(H \circ F)A & \xrightarrow{H(\mathrm{id}_F)A} & (H \circ F)A \\
{\scriptstyle (\beta F)A}\downarrow & & \downarrow{\scriptstyle (\beta F)A} \\
(K \circ F)A & \xrightarrow[K(\mathrm{id}_F)A]{} & (K \circ F)A
\end{array}$$

$$\begin{array}{ccc}
(H \circ F)A & \xrightarrow{(H\alpha)A} & (H \circ G)A \\
{\scriptstyle (\mathrm{id}_H)FA}\downarrow & & \downarrow{\scriptstyle (\mathrm{id}_H)GA} \\
(H \circ F)A & \xrightarrow[(H\alpha)A]{} & (H \circ G)A
\end{array}$$

Since

$$H(\mathrm{id}_F)A = H(\mathrm{id}_{FA}) = \mathrm{id}_{(H \circ F)A}$$

(the first equality by the definition of identity natural transformation in 4.2.13), we have that

$$(\beta * \mathrm{id}_F)A = (\beta F)A \circ \mathrm{id}_{H \circ FA} = (\beta F)A$$

as required. Similarly $(\mathrm{id}_H)GA = \mathrm{id}_{(H \circ G)A}$, so that

$$(\mathrm{id}_H * \alpha)A = (\mathrm{id}_H)GA \circ (H\alpha)A = (H\alpha)A$$

3. a. The first equation in Godement's fifth rule follows from this calculation, using the Interchange Law and Exercise 2.

$$\begin{aligned}
(\gamma F_2) \circ (G_1\alpha) &= (\gamma * \mathrm{id}_{F_2}) \circ (\mathrm{id}_{G_1} * \alpha) \\
&= (\gamma \circ \mathrm{id}_{G_1}) * (\mathrm{id}_{F_2} \circ \alpha) = \gamma * \alpha
\end{aligned}$$

A similar argument shows that $\gamma\alpha = (G_2\alpha) \circ (\gamma F_1)$

b. This is shown by the following calculation, using (in order) Exercise 2, the Interchange Law, the definition of the (vertical) composite of natural transformations and Exercise 2 again:

$$\begin{aligned}(G_1\beta E) \circ (G_1\alpha E) &= (\mathrm{id}_{G_1} * \beta E) \circ (\mathrm{id}_{G_1} * \alpha E) \\ &= (\mathrm{id}_{G_1} \circ \mathrm{id}_{G_1}) * (\beta E \circ \alpha E) \\ &= \mathrm{id}_{G_1} * (\beta \circ \alpha)E = G_1(\beta \circ \alpha)E\end{aligned}$$

Section 4.5

1. Let $F : \mathscr{C} \to \mathbf{Set}$ be representable. Then if 1 is a terminal object in $\mathscr{C}$, then $F(1)$ is isomorphic to $\mathrm{Hom}(C, 1)$, so has exactly one element, so is a terminal object in **Set**.

On the other hand, $\mathrm{Hom}_{\mathbf{Set}}(\emptyset, \emptyset)$ contains exactly one element (the identity function on the empty set), so is not the initial object of **Set**, which is the empty set.

2. Let $H = \mathrm{Hom}_{\mathbf{Set}}(1, -)$ and I the identity functor. Let $\alpha : H \to I$ assign to each function from 1 to I the element which is the image of that function. This is clearly bijective. If $f : S \to T$ is a function, we have to show that the diagram

$$\begin{array}{ccc} H(S) & \xrightarrow{\alpha S} & I(S) \\ {\scriptstyle H(f)}\downarrow & & \downarrow{\scriptstyle I(f)} \\ H(T) & \xrightarrow[\alpha T]{} & I(T) \end{array}$$

commutes. If we take a function $g : 1 \to S$ whose value is $x \in S$, then

$$I(f)(\alpha S(g)) = I(f)(x) = f(x) = \alpha T(f \circ g) = \alpha T(H(f)(g))$$

3. A graph homomorphism $\mathbf{2} \to \mathscr{G}$ takes e to an arrow u of $\mathscr{G}$ and takes 0 to the source of u and 1 to the target of u. In other words, it is completely determined by u. Thus $A(\mathscr{G}) \cong \mathrm{Hom}(\mathbf{2}, \mathscr{G})$. It remains only to show that the isomorphism is natural in $\mathscr{G}$. If $f : \mathscr{G} \to \mathscr{H}$, we must show that

$$\begin{array}{ccc} A(\mathscr{G}) & \xrightarrow{\cong} & \mathrm{Hom}(\mathbf{2}, \mathscr{G}) \\ {\scriptstyle A(f)}\downarrow & & \downarrow{\scriptstyle \mathrm{Hom}(\mathbf{2}, f)} \\ A(\mathscr{H}) & \xrightarrow[\cong]{} & \mathrm{Hom}(\mathbf{2}, \mathscr{H}) \end{array}$$

commutes. Then for an arrow u of $\mathscr{G}$, let $h : \mathbf{1} \to \mathscr{G}$ be defined by $h(e) = u$. Then the upper route in the diagram takes u to h and then to $f \circ h$, whereas the lower route takes u to $f(u)$ and then to the homomorphism h' defined by $h'(e) = f(u)$. Since $(f \circ h)(e) = f(h(e)) = f(u)$, the two are the same.

4. Global elements are arrows from the terminal object. In the case of categories, the terminal object is the category $\mathbf{1}$ with one object and one arrow, the identity of that object. A functor from that object is determined by where it sends that object, which can be to any arbitrary object. The identity is sent to the identity of that chosen object. Thus $\mathrm{Hom}_{\mathbf{Cat}}(\mathbf{1}, \mathscr{C})$ is isomorphic to the set of objects of $\mathscr{C}$. The set of objects is a functor $\mathscr{O} : \mathbf{Cat} \to \mathbf{Set}$: if $F : \mathscr{C} \to \mathscr{D}$ is a functor then $\mathscr{O}(F)$ is F_0, the object part of the functor. To verify naturality amounts to showing that if $G : \mathbf{1} \to \mathscr{C}$ is a functor, then $\mathrm{Hom}(\mathbf{1}, F)(G(*)) = F(G(*))$ which is immediate from the definition. (This says that the global elements of a category are its objects. Note that the global elements of a graph are its *loops*.)

5. Yes and yes. The representing object is the category $\mathbf{2}$ which is the graph $\mathbf{2}$ with the addition of two identity arrows. The argument is virtually identical to the argument for graphs.

6. Set $F^*(C) = \{(x, C) \mid x \in F(C)\}$ and for $f : C \to D$, set $F^*(f)(x, C) = (F(f)(x), D)$. The function $\beta C = x \mapsto (x, C)$ is clearly an isomorphism. To be natural requires that $F^*(f)(\beta C(x))$ be the same as $\beta D(F(f)(x))$, which is immediate from the definition.

7. D is represented by $\{1, 2\}$ (or any other two-element set) and a universal object is $(1, 2)$ (or any other pair with distinct coordinates.) The natural isomorphism from $\mathrm{Hom}(\{1, 2\}, A)$ to $A \times A$ takes a function $f : \{1, 2\} \to A$ to the pair $(f(1), f(2))$. The inverse takes a pair (a_1, a_2) to the function f for which $f(i) = a_i$.

8. Let $\mathscr{D} = \mathscr{C}^{\mathrm{op}}$. Then the ordinary Yoneda embedding

$$\mathscr{D}^{\mathrm{op}} \to \mathbf{Func}(\mathscr{D}, \mathbf{Set})$$

is full and faithful. But $\mathscr{C}^{\mathrm{op}} = \mathscr{D}$ means $\mathscr{D}^{\mathrm{op}} = \mathscr{C}$ so this is just 4.5.5.

9. For each $c \in F(C)$, define a natural transformation $\beta(C, F)(c) : \mathrm{Hom}(C, -) \to F$ as follows: for each $k : C \to A$, let $\beta(C, F)(c)(k) = F(f)(c)$. (This is the natural transformation defined by 4.5.6.) This is the value at c of a function we call $\beta(C, F) : F(C) \to \mathrm{NT}(\mathrm{Hom}(C, -), F)$ where $\mathrm{NT}(G, F)$ stands for the set of natural transformations between functors G and F. It is easy to see that $\mathrm{NT}(G, F)$ is contravariant in G and covariant in F. It is, in fact, the hom functor in the category $\mathbf{Func}(C, \mathbf{Set})$. Then the formulation of naturality is that if $f : C \to C'$ is an arrow and $\alpha : F \to F'$ is a natural transformation, the square

$$\begin{array}{ccc} F(C) & \xrightarrow{\beta(C,F)} & \mathrm{NT}(\mathrm{Hom}(C,-),F) \\ {\scriptstyle \alpha f}\downarrow & & \downarrow{\scriptstyle \mathrm{NT}(\mathrm{Hom}(f,-),\alpha)} \\ F'(C') & \xrightarrow[\beta(C',F')]{} & \mathrm{NT}(\mathrm{Hom}(C',-),F') \end{array}$$

commutes. Here αf is defined to be $F'f \circ \alpha C = \alpha C' \circ Ff$, equal by naturality. Note the double contravariance. From $f : C \to C'$, we get

$$\mathrm{Hom}(f, -) : \mathrm{Hom}(C', -) \to \mathrm{Hom}(C, -)$$

and then

$$\mathrm{NT}(\mathrm{Hom}(C,-),F) \to \mathrm{NT}(\mathrm{Hom}(C',-),F)$$

Now to prove this, we must apply it to a $c \in F(C)$. Going around clockwise gives us the natural transformation from $\mathrm{Hom}(C',-) \to F'$ whose value at an object A takes an arrow $g : C' \to A$ to $\alpha A(F(g \circ f)(c))$. Going the other way we get the natural transformation whose value at an object A is $F'g(\alpha C'(Ff(c)))$. From the functoriality of F and naturality of α, we have

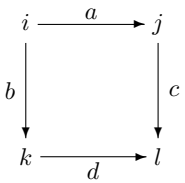
$$\alpha A(F(g \circ f)(c)) = \alpha A(Fg(Ff(c))) = F'g(\alpha C'(Ff(c)))$$

10. If $c \in F(C)$ is a universal element of F, then there is a unique $f : C \to C'$ such that $Ff(c) = c'$. Symmetrically, there is a unique $g : C' \to C$ such that $Fg(c') = c$. Then $Fg(Ff(c)) = Fg(c') = c$. But the universality of c says that there is a unique arrow $h : C \to C$ such that $Fh(c) = c$. Clearly, $h = \mathrm{id}_C$ is one such; it follows that $g \circ f = \mathrm{id}_C$. Symmetrically, $f \circ g = \mathrm{id}_{C'}$.

Section 4.6

1. Let $\mathscr{S}$ be the sketch whose graph has one node $*$ and no arrows, and (necessarily) no diagrams. If M is a model of $\mathscr{S}$, M determines and is determined by $M(*)$, which is a set. If N is a model and $\alpha : M \to N$ a natural transformation, then α has only one component, namely αM, and the naturality condition of Diagram (4.20) is vacuous since the sketch has no arrows. The isomorphism takes M to $M(*)$ and α to αM.

2. The sketch has one node, call it s, and two arrows $u, v : s \to s$. There is one diagram $D : \mathscr{I} \to \mathscr{S}$ based on the shape



defined by $D(i) = D(j) = D(k) = D(l) = s$, $D(a) = D(d) = u$ and $D(b) = D(c) = v$.

3. Since the sketch underlying a category has the same objects as the category and since the theory of a linear sketch also has the same objects as the sketch, the objects are same. For each arrow $f : A \to B$, there is an arrow in the sketch. Whenever $(f_1, f_2, \ldots, f_n)$ is a path in the graph underlying $\mathscr{C}$ there is a diagram $(f_1, f_2, \ldots, f_n) = (f_1 \circ f_2 \circ \cdots \circ f_n)$ so that in the theory category every path is equal to a single arrow and the obvious functor is full. It is also faithful since there is no relation among paths in the theory that does not come from a commutative diagram in $\mathscr{C}$.

4. In general a homomorphism must commute in that way with every arrow in the sketch. However, an arrow that commutes in this way with an invertible arrow also commutes with the inverse:

$$\begin{aligned} f \circ M(v) &= N(v) \circ N(u) \circ f \circ M(v) \\ &= N(v) \circ f \circ M(u) \circ M(v) = N(v) \circ f \end{aligned}$$

5. The theory has nodes 0 and 1. The arrows are given by

(i) $\mathrm{Hom}(0,0) = \{\mathrm{id}_0, u \circ v\}$.

(ii) $\mathrm{Hom}(1,1) = \{\mathrm{id}_1, v \circ u\}$.

(iii) $\mathrm{Hom}(1,0) = \{v, v \circ u \circ v\}$.

(iv) $\mathrm{Hom}(0,1) = \{u\}$.

The nontrivial composites are given by $u \circ v \circ u = u$, $v \circ u \circ v \circ u = v \circ u$, and $u \circ v \circ u \circ v = u \circ v$.

Section 4.7

1. $I(0) = \{[x], [vux], [vy], [vuvy]\}$ and $I(1) = \{[y], [uvy], [ux]\}$. For any element $[z]$ of $I(0)$, $I(u)([z]) = [uz]$ and for any element $[w]$ of $I(1)$, $I(v)([w]) = [vw]$, all subject to the equation $uvu = u$.

2. If two terms are forced to be equal by the equivalence relation, then they are certainly equal in every model since the relations are valid in every model. On the other hand, the theory is a model and if the two terms are not forced to be equal by the equivalence relation, they are not equal in the theory.

3. Define the model $T : \mathscr{S} \to \mathbf{Set}$ this way: for any node a of the sketch, $T(a) = \{*\}$ (any one element set will do). For any arrow $f : a \to b$, $T(f)$ is the only possible function. If x is a constant of type a, then set $T(x) = *$ (the only possibility). If $\mathscr{S}$ has diagrams, T automatically takes them to commutative diagrams. If M is any model of $\mathscr{S}$, there is just one natural transformation from M to T whose value at a node a is the only possible map $M(a) \to T(a) = \{*\}$. The requisite naturality diagram commutes because there is only one possible map to $\{*\}$ from any set.

Section 4.8

1. Suppose the composites $\beta * \alpha$, $\delta * \gamma$, $\gamma \circ \alpha$ and $\delta \circ \beta$ are all defined. We must show that $\mathrm{cod}^v(\beta * \alpha) = \mathrm{dom}^v(\delta * \gamma)$.

$$\begin{aligned} \mathrm{id}^v \mathrm{cod}^v(\beta * \alpha) &= \mathrm{id}^v \mathrm{cod}^v \beta * \mathrm{id}^v \mathrm{cod}^v \alpha && \text{TC–5} \\ &= \mathrm{id}^v \mathrm{dom}^v \delta * \mathrm{id}^v \mathrm{dom}^v \gamma \\ &= \mathrm{id}^v \mathrm{dom}^v(\delta * \gamma) && \text{TC–5} \end{aligned}$$

where the second equality is because $\delta \circ \beta$ and $\gamma \circ \alpha$ are both defined.

2. We must show that for all posets A, B and C (with the ordering written $\leq$ in all of them), $\mathrm{comp} : \mathrm{Hom}(B,C) \times \mathrm{Hom}(A,B) \to \mathrm{Hom}(A,C)$ is monotone. Suppose $f \leq g : A \to B$ and $h \leq k : B \to C$. We must show that $h \circ f \leq k \circ g$. For any $x \in A$, $h(f(x)) \leq h(g(x)$ because $f \leq g$ and h is monotone, and $h(g(x)) \leq k(g(x))$ because $h \leq k$. The result follows from transitivity.

3. Let α and α' be relations from A to B, γ and γ' relations from B to C, and $\alpha \subseteq \alpha'$ and $\gamma \subseteq \gamma'$. We must show that $\gamma \circ \alpha \subseteq \gamma' \circ \alpha'$. Suppose $(x,z) \in \gamma \circ \alpha$. Then there is $y \in B$ such that $(x,y) \in \alpha$ and $(y,z) \in \gamma$. But then $(x,y) \in \alpha'$ and $(y,z) \in \gamma'$, so that $(x,z) \in \gamma' \circ \alpha'$ as required.

4. Let $f, f' : A \to B$ be partial functions with f defined on A_0 and f' defined on A'_0, and $g, g' : B \to C$ partial functions with g defined on B_0 and g' defined on B'_0. We must show that if $f \leq f'$ and $g \leq g'$, then domain of definition of $g \circ f$ is included in the domain of definition of $g' \circ f'$ and for x such that $g(f(x))$ is defined, $g'(f'(x)) = g(f(x))$. So suppose $g(f(x))$ is defined for some $x \in A$. Then by definition of composition in 2.1.13, $x \in A_0$ and $f(x) \in B_0$. By the assumption that $f \leq f'$, $x \in A'_0$ and $f'(x) = f(x)$. Hence $f'(x) \in B_0$ and $g(f'(x))$ is defined and equal to $g(f(x))$. Now the assumption that $g \leq g'$ means that $f'(x) \in B'_0$ and $g(f'(x)) = g'(f'(x))$. Hence $g(f(x)) = g'(f'(x))$ as required.

Solutions for Chapter 5

Section 5.1

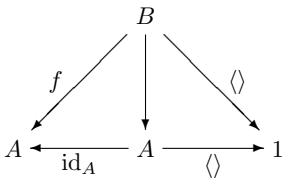
1. Let $\mathscr{C}$, $\mathscr{D}$ and $\mathscr{E}$ be categories. The category $\mathscr{C} \times \mathscr{D}$ has functors $P_1 : \mathscr{C} \times \mathscr{D} \to \mathscr{C}$ and $P_2 : \mathscr{C} \times \mathscr{D} \to \mathscr{D}$ defined by $P_1(C,D) = C$, $P_2(C,D) = D$, $P_1(f,g) = f$ and $P_2(f,g) = g$ for C and D objects and f and g arrows of $\mathscr{C}$ and $\mathscr{D}$ respectively. That these are functors follows immediately from the fact that source, target and composition in the product category are defined coordinatewise (see 2.6.6). Now let $F : \mathscr{E} \to \mathscr{C}$ and $G : \mathscr{E} \to \mathscr{D}$ be functors. Define $\langle F, G\rangle : \mathscr{E} \to \mathscr{C} \times \mathscr{D}$ by $\langle F, G\rangle(E) = (F(E), G(E))$ and $\langle F, G\rangle(h) = (F(h), G(h))$ for E an object and h an arrow of $\mathscr{E}$. The proof that this is a functor is immediate. Then $P_1 \circ \langle F, G\rangle(E) = P_1(F(E), G(E)) = F(E)$ and similarly for arrows. And similarly, $P_2 \circ \langle F, G\rangle = G$. If $H : \mathscr{E} \to \mathscr{C} \times \mathscr{D}$ is any functor with $P_1 \circ H = F$ and $P_2 \circ H = G$, let $H(E) = (H_1(E), H_2(E))$. Then $F(E) = P_1 \circ H(E) = H_1(E)$ and similarly for arrows. Thus $F = H_1$ and similarly $G = H_2$, which proves uniqueness.

2. Let M and N be monoids. The product is the product of the underlying sets with multiplication $(m_1, n_1)(m_2, n_2) = (m_1 m_2, n_1 n_2)$. The identity element is $(1, 1)$.

3. If P and Q are posets, their product is the product of the underlying sets with $(p_1, q_1) \leq (p_2, q_2)$ if and only if $p_1 \leq p_2$ and $q_1 \leq q_2$.

4. This is essentially the same as for categories.

5. A cone over A and 1 has to have this form, where $f : B \to A$ is any arrow.



Clearly the only possible arrow in the middle is f.

6. In the category of sets the product of any set A with the empty set is the empty set. If A is nonempty, the projection onto A is not surjective, hence not an epimorphism.

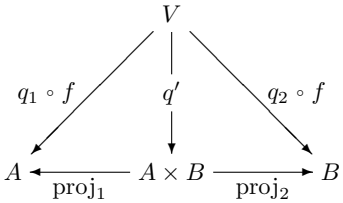
Section 5.2

1. The isomorphism is given by

$$f(x) = \begin{cases} (2,1) & \text{if } x = 1 \\ (1,1) & \text{if } x = 2 \\ (3,1) & \text{if } x = 3 \\ (2,2) & \text{if } x = 4 \\ (1,2) & \text{if } x = 5 \\ (3,2) & \text{if } x = 6 \end{cases}$$

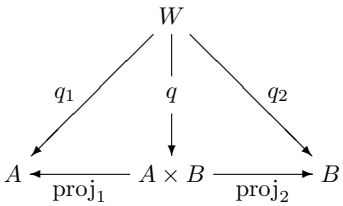
2. The isomorphism of the preceding exercise can be composed with any of the $6! = 720$ permutations of 6 to give another one.

3. By 5.2.13, the left vertical arrow in (5.12) takes the pair of arrows (q_1, q_2), where $q_1 : W \to A$ and $q_2 : W \to B$, to $(q_1 \circ f, q_2 \circ f)$, and the right vertical arrow takes $q : W \to A \times B$ to $q \circ f : V \to A \times B$. Therefore, by Definition 5.2.7, if you start at lower left with (q_1, q_2) and go north and then east, you get the unique arrow q' which makes



(∗)

commute. If you go east and north, you get $q \circ f$, where q is the unique arrow which makes



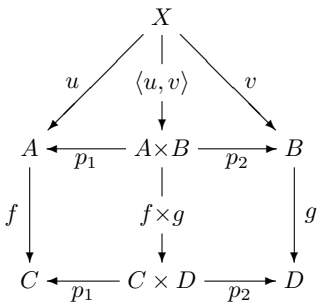
commute. Because $\mathrm{proj}_i \circ (q \circ f) = (\mathrm{proj}_i \circ q) \circ f = q_i \circ f$ for $i = 1, 2$, it follows that $q \circ f = q'$. This is a consequence of the fact that q' is the *unique* arrow making $(*)$ commute. Hence (5.12) commutes.

4. Let $f : E \to A$ and $g : E \to B$. Then $i^{-1} \circ \langle f, g\rangle : C \to V$ is an arrow such that

$$\mathrm{proj}_1 \circ i \circ i^{-1} \circ \langle f, g\rangle = \mathrm{proj}_1 \circ \mathrm{id}_C \circ \langle f, g\rangle = \mathrm{proj}_1 \circ \langle f, g\rangle = f$$

and similarly $\mathrm{proj}_2 \circ i \circ i^{-1} \circ \langle f, g\rangle = g$. Moreover if $h : E \to V$ is such that $\mathrm{proj}_1 \circ i \circ h = f$ and $\mathrm{proj}_2 \circ i \circ h = g$, then we have $p_1 \circ i \circ h = p_1 \circ i \circ i^{-1} \circ \langle f, g\rangle$ and $p_2 \circ i \circ h = p_2 \circ i \circ i^{-1} \circ \langle f, g\rangle$. But since arrows to C are uniquely determined by their projections to A and B, we conclude that $i \circ h = i \circ i^{-1} \circ \langle f, g\rangle$ from which the isomorphism i can be cancelled to give $h = i^{-1} \circ \langle f, g\rangle$.

5. Chase the diagram



to show that

$$p_1 \circ (f \times g) \circ \langle u, v\rangle = f \circ p_1 \circ \langle u, v\rangle = f \circ u$$

and similarly $p_2 \circ (f \times g) \circ \langle u, v\rangle = g \circ v$, so that $(f \times g) \circ \langle u, v\rangle$ satisfies the condition that determines the map $\langle f \circ u, g \circ v\rangle$ uniquely.

Section 5.3

1. a. Define $q_1 = \mathrm{proj}_1 : A \times (B \times C) \to A$, $q_2 = \mathrm{proj}_1 \circ \mathrm{proj}_2 : A \times (B \times C) \to B$ and $q_3 = \mathrm{proj}_2 \circ \mathrm{proj}_2 : A \times (B \times C) \to C$. The meaning of the last, for example, is the second projection to $B \times C$, followed by the second projection from the latter to C. Now suppose D is an object and $f_1 : D \to A$, $f_2 : D \to B$ and $f_3 : D \to C$ are given. Then there is a unique arrow $\langle f_2, f_3\rangle : D \to B \times C$ such that $\mathrm{proj}_1 \circ \langle f_2, f_3\rangle = f_2$ and $\mathrm{proj}_2 \circ \langle f_2, f_3\rangle = f_3$. It follows that there is a unique arrow $\langle f_1, \langle f_2, f_3\rangle\rangle : D \to A \times (B \times C)$ such that $\mathrm{proj}_1 \circ \langle f_1, \langle f_2, f_3\rangle\rangle = f_1$ and $\mathrm{proj}_2 \circ \langle f_1, \langle f_2, f_3\rangle\rangle = \langle f_2, f_3\rangle$ from which the required identities follow immediately. If $g : D \to A \times (B \times C)$ is another arrow with $q_i \circ g = f_i$ for $i = 1, 2, 3$, then $\mathrm{proj}_1 \circ \mathrm{proj}_2 \circ g = f_2$ and $\mathrm{proj}_2 \circ \mathrm{proj}_2 \circ g = f_3$, from which it follows from the uniqueness of arrows into a product, that $\mathrm{proj}_2 \circ g = \langle f_2, f_3\rangle$. Also, $p_1 \circ g = f_1$ so that $g = \langle f_1, \langle f_2, f_3\rangle\rangle$.

b. If $p : B \to A$ is a unary product diagram, then by definition there is for each object X a bijection

$$f \mapsto \langle f\rangle : \mathrm{Hom}(X, A) \to \mathrm{Hom}(X, B)$$

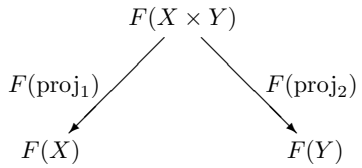
for which $p \circ \langle f \rangle = f$. This bijection is a natural isomorphism from $\mathrm{Hom}(-, A)$ to $\mathrm{Hom}(-, B)$: if $u : Y \to X$, then naturality follows from the fact that $p \circ \langle f \rangle \circ u = f \circ u$, so that $\langle f \rangle \circ u = \langle f \circ u \rangle$. It follows from Corollary 4.5.4 that p is an isomorphism.

c. The preceding part shows that every category has unary products, so such a category has n-ary products for $n = 0$, 1 and 2. The first part shows the same for $n = 3$ and also gives for that case the essential step for an obvious induction on n.

2. Given q_1 and q_2 as in (ii), we form $\langle q_1, q_2 \rangle$. From (iii), we see that $p_1 \circ \langle q_1, q_2 \rangle = q_1$ and $p_2 \circ \langle q_1, q_2 \rangle = q_2$. If h is another arrow satisfying the same identities, then (iv) tells us that $h = \langle p_1 \circ h, p_2 \circ h \rangle = \langle q_1, q_2 \rangle$ so that we have the uniqueness required by 5.1.3.

3. We saw in Exercises 1 and 4 of Section 5.1 that the product in each category took as objects of the product category the product of the objects and as arrows of the product the product of the arrows. Thus the product is constructed in the same way in both categories.

4. Let $\mathscr{C} = \mathscr{D}$ be the category of countably infinite sets and all functions between them. Let $F : \mathscr{C} \to \mathscr{D}$ be defined by $F(X) = \{X\} \cup X$ for such a set X (the idea is that $F(X)$ is X with a new element adjoined; the choice of $\{X\}$ to be the new element is convenient but not the only possible one) and if $f : X \to Y$, $F(f) = 1 + f$ is the function whose restriction to X is f and which takes the added element $\{X\}$ of $F(X)$ to the added element $\{Y\}$ of Y. Then although $F(X \times Y) \cong F(X) \times F(Y)$ since any two countable sets are isomorphic, the cone

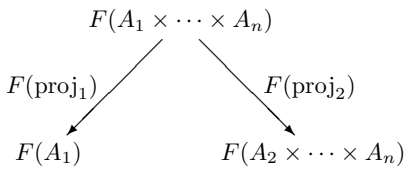


is not a product cone. In fact, $F(X \times Y)$ contains no point u with the property that $F(\mathrm{proj}_1)(u) = \{X\}$ but $F(\mathrm{proj}_2)(u) \neq \{Y\}$ or vice versa.

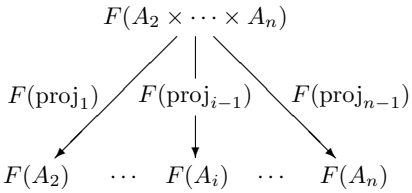
5. a. Since a unary product diagram is an isomorphism and every functor preserves isomorphisms, every functor preserves unary products. Now for $n \geq 3$, n-ary products are defined by induction. Assuming that a functor F preserves $n - 1$-ary products, then the product

$$A_1 \times A_2 \times \cdots \times A_n = A_1 \times (A_2 \times \cdots \times A_n)$$

Then



is a product cone. By the inductive assumption so is

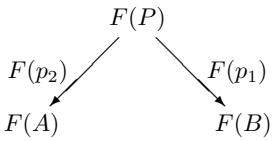


from which the conclusion follows.

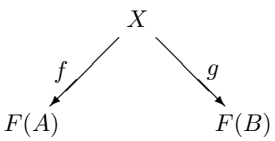
b. Let $\mathscr{C} = \mathbf{1}$, the category with one object called 0 and its identity arrow. Let $\mathscr{D} = \mathbf{2}$, the category with two objects, called 0 and 1, their identities and one arrow $0 \to 1$. Then 0 is the terminal object of $\mathscr{C}$ and 1 is the terminal object of $\mathscr{D}$. Both categories have finite products, with $0^n = 0$ in both and $1^n = 1$ and $0 \times 1 = 0$ in $\mathscr{D}$. The products are canonical since there is only one possible choice. Then the functor $F : \mathscr{C} \to \mathscr{D}$ given by $F(0) = 0$ preserves n-ary products for $n \geq 1$, but not nullary products.

6. A terminal object in a category is an object that every other object has an arrow to (unique, of course). In a poset, that is an element that every element is less than or equal to, that is a top element. There is no largest integer so $\mathbf{N}$ has no terminal element. We saw in 5.1.8 that products in posets are just meets. Since the meet of two nonnegative integers is the smaller of the two, $\mathbf{N}$ has binary products.

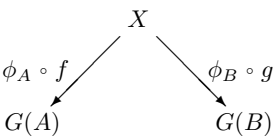
7. Suppose



is a product diagram and $\phi : F \to G$ is a natural isomorphism. Suppose



is given. Then



induces a unique map $u : X \to G(P)$ for which $G(p_1) \circ u = \phi_A \circ f$ and $G(p_2) \circ u = \phi_B \circ g$. The required map from X to $F(P)$ is $\phi_P^{-1} \circ u$. We have $F(p_1) \circ \phi_P^{-1} \circ u = \phi_A^{-1} \circ G(p_1) \circ u$ by naturality, but the right side is $\phi_A^{-1} \circ \phi_A \circ f = f$ by definition of u. Similarly $F(p_2) \circ \phi_P^{-1} \circ u = g$ as required. Uniqueness follows from the fact that u is unique and ϕ_P is an isomorphism

Section 5.4

1. Using the version of the disjoint sum of 5.4.5, define $(f+g)(s,0) = (f(s),0)$ and $(f+g)(t,1) = (g(t),1)$.

2. Let P be a poset and $x, y \in P$. The sum $x+y$ is characterized by the fact that there is an arrow $x+y \longrightarrow z$ corresponding to every pair consisting of an arrow $x \longrightarrow z$ and an arrow $y \longrightarrow z$. In a poset, there is an arrow $x \longrightarrow z$ and only one if and only if $x \leq z$ and similarly for y. Thus $x+y \leq z$ if and only if $x \leq z$ and $y \leq z$. But this property characterizes the join $x \vee y$.

3. Let P and Q be two posets and $P+Q$ denote the disjoint sum of the sets P and Q as described in 5.4.5. Define $(x,i) \leq (y,i)$, $i = 0, 1$ if and only if $x \leq y$, while $(x,0) \not\leq (y,1)$ and $(x,1) \not\leq (y,0)$ for all $x, y \in P+Q$. The proof that this is the sum is essentially the same as the sum for the category of sets, augmented by the observation that an arrow from $P+Q \longrightarrow R$ preserves the partial order just defined if and only if its restrictions to P and Q do.

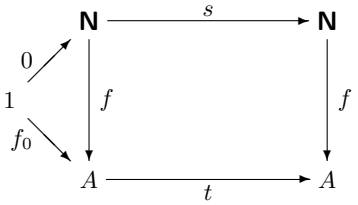
4. As noted in 5.4.7, we can take the canonical injections to be the same as in **Set**: $i_1(x) = (x,0)$ for $x \in S$ and $i_2(x) = (x,1)$ for $x \in T$. Then given $f : S \longrightarrow X$, $g : T \longrightarrow X$, define $\langle f|g\rangle : S+T \longrightarrow X$ by $\langle f|g\rangle(s,0) = f(s)$ if and only if $f(s)$ is defined and $\langle f|g\rangle(t,1) = g(t)$ if and only if $g(t)$ is defined. Then, because i_1 is defined for all $x \in S$, $\langle f|g\rangle(i_1(x)) = \langle f|g\rangle(x,0) = f(x)$ if and only if $f(x)$ is defined, so that $\langle f|g\rangle \circ i_1 = f$, and similarly for g. It is clear that $\langle f|g\rangle$ is the only arrow such that $\langle f|g\rangle \circ i_1 = f$ and $\langle f|g\rangle \circ i_2 = g$.

5. Use the example given in the answer to Exercise 6 of Section 5.1 in the dual category.

Section 5.5

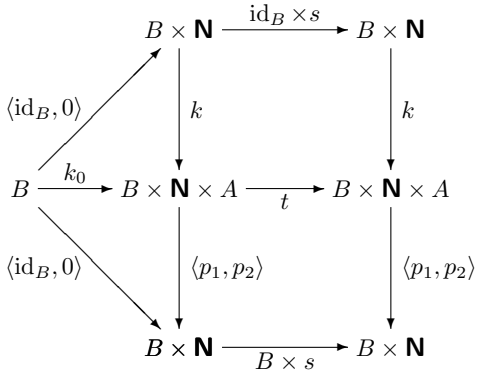
1. Given sets A and B, a function $f_0 : B \longrightarrow A$ and $t : A \longrightarrow A$, define a function $f : B \times \mathbf{N} \longrightarrow A$ by letting $f(b,0) = f_0(b)$ and having defined $f(b,i)$ for $i \leq n$, define $f(b,n+1) = t(f(b,n))$.

2. To be a model M of that sketch is to be an object $A = M(n)$ together with arrows $f_0 = M(\text{zero}) : 1 \longrightarrow A$ and $t = M(\text{succ}) : A \longrightarrow A$, so that $(\mathbf{N}, 0, s)$ is certainly a model. If (A, f_0, t) is another model, then there is a unique arrow $f : \mathbf{N} \longrightarrow A$ such that

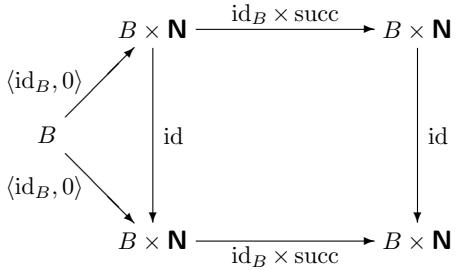


commutes. But the commutation of the two parts of that diagram express the fact that f is an arrow in the category of models of the sketch. Since f is unique, this shows that there is exactly one such arrow from $(\mathbf{N}, 0, s)$ to any other model of the sketch, whence that is the initial model.

3. Define $t : B \times \mathbf{N} \times A \to B \times \mathbf{N} \times A$ by $t(b, n, a) = h(b, n, f(b, n))$. Define $k_0 : B \to B \times \mathbf{N} \times A$ by $k_0(b) = (b, 0, g(b))$. Then the induction property gives an arrow $k : B \times \mathbf{N} \to B \times \mathbf{N} \times A$ such that $k(b, 0) = k_0(b) = (b, 0, g(b))$ and $k(b, s(n)) = t(k(b, n))$. If we let $k_i = \text{proj}_i \circ k$, $i = 1, 2, 3$, then $k(b, n) = (k_1(b, n), k_2(b, n), k_3(b, n))$. Next we claim that $k_1(b, n) = b$ and $k_2(b, n) = n$. For consider the diagram (in which we have abbreviated proj as p)



Compare this to the diagram



and the uniqueness of recursively defined arrows implies that

$$\langle p_1, p_2 \rangle \circ k = \langle \text{id}_B, \text{id}_{\mathbf{N}} \rangle$$

Then $f = k_3$ has the required properties.

Section 5.6

1. There is, by CC–1 and CC–2, a proof of $A \Rightarrow$ true and only one for each object A of $\mathscr{C}$ so that true is the terminal object. If $q_1 : X \to A$ and $q_2 : X \to B$ are arrows in $\mathscr{C}$, then from CC–3, there is an arrow $\langle q_1, q_2 \rangle : X \to A \wedge B$. According to CC–4, $p_1 \circ \langle q_1, q_2 \rangle = q_1$ and $p_2 \circ \langle q_1, q_2 \rangle = q_2$ and by CC–5, the arrow $\langle q_1, q_2 \rangle$ is unique with this property, so that $A \wedge B$ together with p_1 and p_2 is a product of A and B in $\mathscr{C}$. The proof follows from Proposition 5.3.10.

2. a. Either projection is a proof.
b. $\langle \text{id}_A, \text{id}_A \rangle$ is a proof.
c. $\langle \text{proj}_2, \text{proj}_1 \rangle$ is a proof.
d. $\langle \text{proj}_1 \circ \text{proj}_1, \langle \text{proj}_2 \circ \text{proj}_1, \text{proj}_2 \rangle \rangle$ is a proof.

Section 5.7

1. The reason for both categories is the same: they fail Proposition 5.7.3 (ii). In **Rel** the product of 0 and S is the disjoint union of the empty set and S, that is, it is S. Similarly in **Mon**, the initial object is isomorphic to the terminal object whose product with any monoid M is M.

2. We have, for any objects X, Y and Z, that $X + Y + X \cong (X + Y) + Z$ by the dual of the argument in 5.3.3. Then $D \times (A + B + C) \cong D \times ((A + B) + C) \cong D \times (A + B) + D \times C \cong (D \times A + D \times B) + D \times C \cong D \times A + D \times B + D \times C$.

3. The arrow h is the composite

$$\begin{aligned} C \times T &\xrightarrow{\mathrm{id}_C \times c} C \times (1 + 1 + 1) \xrightarrow{\cong} C \times 1 + C \times 1 + C \times 1 \\ &\xrightarrow{p_1 + p_1 + p_1} C + C + C \xrightarrow{f + g + h} D + D + D \\ &\xrightarrow{\langle \mathrm{id}_D \,|\, \mathrm{id}_D \,|\, \mathrm{id}_D \rangle} D \end{aligned}$$

Solutions for Chapter 6

Section 6.1

1. Define $\mathrm{eval} : [\mathscr{C} \to \mathscr{D}] \times \mathscr{C} \to \mathscr{D}$ on objects as follows. Let $F : \mathscr{C} \to \mathscr{D}$ and let C be an object of $\mathscr{C}$. Then $\mathrm{eval}(F, C) = F(C)$. On arrows, suppose $\alpha : F \to F'$ is a natural transformation and $f : C \to C'$ is an arrow of $\mathscr{C}$. Then set $\mathrm{eval}(\alpha, f) = F'f \circ \alpha C = \alpha C' \circ Ff : F(C) \to F'(C')$ (they are the same by naturality).

Since (α, f) is an arrow from (F, C) to (F', C'), eval preserves source and target because $\mathrm{eval}(F, C) = F(C)$ and $\mathrm{eval}(F', C') = F'(C')$. Preservation of identities is easy to verify. As for composition, let $\beta : F' \to F''$ and $g : C' \to C''$. Then

$$\begin{aligned} \mathrm{eval}((\beta, g) \circ (\alpha, f)) &= \mathrm{eval}(\beta \circ \alpha, g \circ f) \\ &= F''(g \circ f) \circ (\beta \circ \alpha)C \\ &= F''(g) \circ F''(f) \circ \beta C \circ \alpha C \\ &= F''(g) \circ \beta C' \circ F'(f) \circ \alpha C \\ &= \mathrm{eval}(\beta, g) \circ \mathrm{eval}(\alpha, f) \end{aligned}$$

The third line follows from the functoriality of F'' and the definition of composition of natural transformations; the fourth line is by naturality of β.

For a functor $F : \mathscr{B} \times \mathscr{C} \to \mathscr{D}$ and an object (B, C) of $\mathscr{B} \times \mathscr{C}$,

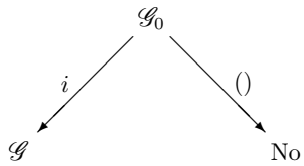
$$\begin{aligned} \mathrm{eval} \circ (\lambda F \times \mathscr{C})(B, C) &= \mathrm{eval} \circ (\lambda F(B), C) \\ &= \lambda F(B)(C) = F(B, C) \end{aligned}$$

Thus eval $\circ (\lambda F \times \mathscr{C}) = F$, as required. Similarly, for an arrow $(f, g) : (B, C) \to (B', C')$ of $\mathscr{B} \times \mathscr{C}$,

$$\begin{aligned} \text{eval} \circ (\lambda F \times \mathscr{C})(f, g) &= \text{eval} \circ (\lambda F(f), g) \\ &= \lambda F(B')(g) \circ \lambda F(f)(C) \\ &= F(B', g) \circ F(f, C) = F(f, g) \end{aligned}$$

as required.

2. Let $\mathscr{G}_0$ be the set of nodes of $\mathscr{G}$ regarded as a graph with no arrows. We must show that



is a product diagram, where () is the unique function and $i : \mathscr{G}_0 \to \mathscr{G}$ is the identity function on nodes and (necessarily) the empty function on arrows. Let $\mathscr{T}$ be a graph and $f : \mathscr{T} \to \mathscr{G}$, $g : \mathscr{T} \to$ No be graph homomorphisms. We must find $u : \mathscr{T} \to \mathscr{G}_0$ for which $i \circ u = f$ and $() \circ u = ()$. The second equation is automatic. The first equation requires that u be the same as f on nodes. The fact that there is a homomorphism from $\mathscr{T}$ to No (which has no arrows) means that $\mathscr{T}$ has no arrows. Thus u has to be the empty function on arrows; since f must also be the empty function on arrows it follows that $i \circ u = f$.

A node of $\mathscr{G} \times$ No as constructed in Exercise 4 of Section 5.1 is a pair (g, n) where g is a node of $\mathscr{G}$ and n is the unique node of No. (The construction in that exercise shows that $\mathscr{G} \times$ No has no arrows since No has none.) Thus the function $(g, n) \mapsto g$ is clearly a bijection between $\mathscr{G} \times$ No and the set of nodes of $\mathscr{G}$. What we have constructed here is more than that: it is a natural isomorphism in the category of graphs.

3. Given $f : \mathscr{C} \times \mathscr{G} \to \mathscr{H}$, we must show that eval $\circ (\lambda f \times \mathscr{G}) = f$. First, suppose that (c, n) is a node of $\mathscr{C} \times \mathscr{G}$. Then

$$\begin{aligned} \text{eval}((\lambda f \times \mathscr{G})(c, n)) &= \text{eval}(\lambda f(c), n) \\ &= \lambda f(c)(n) = f(c, n) \end{aligned}$$

For an arrow $(a : c \to d, w)$ of $\mathscr{C} \times \mathscr{G}$,

$$\begin{aligned} \text{eval}((\lambda f \times \mathscr{G})(a, w)) &= \text{eval}(\lambda f(a), w) \\ &= f_a(w) = f(a, w) \end{aligned}$$

as required.

4. Corollary 4.5.13 says that two representing objects for the same functor are isomorphic by a unique isomorphism that preserves the universal element. Let $\phi(A, B) : [A \to B]' \to [A \to B]$ be the isomorphism. In this case the functor is $\text{Hom}(- \times A, B)$ and preserving the universal element means that

$$\text{Hom}(\phi(A, B) \times A, B)(\text{eval}) = \text{eval} \circ (\phi(A, B) \times A) = \text{eval}'$$

which is the left diagram of the proposition. The right diagram follows from this calculation: eval $\circ (\phi(A, B) \times A) \circ \lambda' f = \text{eval}' \circ \lambda' f = f$, so by the uniqueness requirement of CCC–3, $(\phi(A, B) \times A) \circ \lambda' f = \lambda f$.

Section 6.2

1. By CCC–3, $\text{eval} \circ ((\lambda(\text{eval})) \times A) = \text{eval} : [A \to B] \times A \to B$. By the uniqueness requirement of CCC-3, $(\lambda(\text{eval})) \times A = [A \to B] \times A$ (the identity arrow) so the first component of $(\lambda(\text{eval}) \times A)$ must be the identity.

2. By CCC–3, $\text{eval} \circ (\lambda f \times A) \circ (g \times A) = f \circ (g \times A)$. But by 5.2.19, $\text{eval} \circ (\lambda f \times A) \circ (g \times A) = \text{eval} \circ ((\lambda f \circ g) \times A)$. By the uniqueness property of eval, it follows that $\lambda(f \circ (g \times A)) = (\lambda f) \circ g$.

3. Since $f \mapsto \lambda f$ is a bijection on each hom set by CCC-3, we only need to show that the following diagrams is commutative, where $g : B \to B'$.

$$\begin{array}{ccc} \text{Hom}(C \times A, B) & \xrightarrow{f \mapsto \lambda f} & \text{Hom}(C, [A \to B]) \\ \Big\downarrow{\scriptstyle \text{Hom}(C \times A, g)} & & \Big\downarrow{\scriptstyle \text{Hom}(C, [A \to g])} \\ \text{Hom}(C \times A, B') & \xrightarrow[f \mapsto \lambda f]{} & \text{Hom}(C, [A \to B']) \end{array}$$

This requires that $\lambda(g \circ f) = [A \to g] \circ \lambda f$. This follows from CCC–3 and Exercise 2 by the following calculation:

$$\begin{aligned} [A \to g] \circ \lambda f &= \lambda(g \circ \text{eval}) \circ \lambda f \\ &= \lambda(g \circ \text{eval} \circ (\lambda f \times A)) \\ &= \lambda(g \circ f) \end{aligned}$$

4. Let $h : 1 \to [A \to B]$ be a global element. Let $u : A \to 1 \times A$ be an isomorphism (see Section 5.3). The bijection from $\text{Hom}(1, [A \to B])$ to $\text{Hom}(A, B)$ takes h to

$$\text{eval} \circ (h \times A) \circ u : A \to 1 \times A \to [A \to B] \times A \to B$$

Its inverse takes $f : A \to B$ to $\lambda(f \circ u^{-1}) : 1 \to [A \to B]$, which fits since $f \circ u^{-1} : 1 \times A \to A \to B$. That this is indeed the inverse follows from CCC–3. (In categorical writing, these manipulations with u are nearly always suppressed by assuming that $A = A \times 1$ and $u = \text{id}_A$.)

5. In this answer, we regard λ as binding more tightly than composition or product, so that for example $\lambda f \circ g$ means $(\lambda f) \circ g$ and $\lambda f \times g$ means $(\lambda f) \times g$.

a. Define $\Gamma B : \text{Hom}(C, [A \to B]) \to \text{Hom}(C \times A, B)$ by $\Gamma B(g) = \text{eval} \circ (g \times A)$. Then by CCC–3, $\Gamma B \circ \Lambda B(f) = \text{eval} \circ (\lambda f \times A) = f$ for $f : C \times A \to B$, and by the uniqueness requirement for eval,

$$\Lambda B \circ \Gamma B(g) = \lambda(\text{eval} \circ (g \times A)) = g$$

for $g : C \to [A \to B]$, so ΓB is the inverse of ΛB which is therefore bijective.

b. Let $g : C' \to C$ and $h : B \to B'$. Naturality requires that this diagram commute:

$$\begin{array}{ccc} \text{Hom}(C, [A \to B]) & \xrightarrow{\widehat{h}C} & \text{Hom}(C, [A \to B']) \\ \Big\downarrow{\scriptstyle \text{Hom}(g, [A \to B])} & & \Big\downarrow{\scriptstyle \text{Hom}(g, [A \to B'])} \\ \text{Hom}(C', [A \to B]) & \xrightarrow[\widehat{h}C']{} & \text{Hom}(C', [A \to B']) \end{array}$$

Because ΛB is a bijection, an arbitrary arrow of $\mathrm{Hom}(C, [A \to B])$ can be taken to be λf for some $f : C \times A \to B$. The upper route around the diagram takes λf to

$$\lambda(h \circ \mathrm{eval} \circ (\lambda f \times A)) \circ g = \lambda(h \circ f) \circ g$$

whereas the lower route takes it to

$$\begin{array}{rcl} \lambda(h \circ \mathrm{eval} \circ ((\lambda f \circ g) \times A)) & = & \lambda(h \circ \mathrm{eval} \circ (\lambda f \times A) \circ (g \times A)) \\ & = & \lambda(h \circ f \circ (g \times A)) \end{array}$$

which is the same thing by Exercise 2.

c. Let $h : B \to B'$. The required diagram for naturality, namely

$$\begin{array}{ccc} \mathrm{Hom}(C \times A, B) & \xrightarrow{\mathrm{Hom}(C \times A, h)} & \mathrm{Hom}(C \times A, B') \\ \Big\downarrow{\scriptstyle \Lambda B} & & \Big\downarrow{\scriptstyle \Lambda B'} \\ \mathrm{Hom}(C, [A \to B]) & \xrightarrow[\mathrm{Hom}(C, [A \to h])]{} & \mathrm{Hom}(C, [A \to B']) \end{array}$$

commutes by definition of $[A \to h]$. It is a natural isomorphism because each component is a bijection.

d. Let $C = [A \to B]$ in the preceding diagram and start with eval in the upper right corner. The upper route gives $\lambda(h \circ \mathrm{eval})$ and the lower route gives $[A \to h] \circ \lambda(\mathrm{eval})$, which is $[A \to h]$ by Exercise 1.

6. We use the formulation of $[\mathscr{G} \to \mathscr{H}]$ in Exercise 3. Let $h : 1 \to [\mathscr{G} \to \mathscr{H}]$ be a global element. Remember that in the category of graphs and graph homomorphisms, 1 is the graph with one node n and one arrow e. The node n goes to a node of $[\mathscr{G} \to \mathscr{H}]$, which is a function $f_0 : G_0 \to H_0$. The arrow e goes to an arrow of $[\mathscr{G} \to \mathscr{H}]$, which is an ordered triple $(f_1, f_2, f_3) : f_1 \to f_2$ as described in Exercise 3. The fact that 1 has only one node means that, since h is a graph homomorphism, necessarily $f_1 = f_2 = f_0$. Then the conditions in the description in Exercise 3 say that for any arrow g of $\mathscr{G}$, $\mathrm{source}(f_3(g)) = f_0(\mathrm{source}(g))$ and $\mathrm{target}(f_3(g)) = f_0(\mathrm{target}(g))$. Thus f_0 must be the node map and f_3 the arrow map of a graph homomorphism from $\mathscr{G}$ to $\mathscr{H}$. It follows from Exercise 4 that the *loops* of $[\mathscr{G} \to \mathscr{H}]$ are in one to one correspondence with the graph homomorphisms from $\mathscr{G}$ to $\mathscr{H}$: to recover the homomorphism from the loop $(f_1, f_1, f_3) : f_1 \to f_1$, take the node map to be f_1 and the arrow map to be f_3.

Section 6.3

1. For $i = 1, 2$,

$$\begin{array}{rcll} \mathrm{proj}_i(c) & =_X & \lambda_{x \in A \times B}(\mathrm{proj}_i(x))`c & (\mathrm{TL}\text{–}17) \\ & =_X & \lambda_{x \in A \times B}(\mathrm{proj}_i(x))`c' & (\mathrm{TL}\text{–}12) \\ & =_X & \mathrm{proj}_i(c') & (\mathrm{TL}\text{–}17) \end{array}$$

where in the applications of TL–17, we judiciously choose a variable x of type $A \times B$ that does not occur freely in c or c'.

2.

$$\begin{array}{lll} (a,b) & =_X & \lambda_{x\in A}(x,b)\text{`}a \quad \text{(TL–17)} \\ & =_X & \lambda_{x\in A}(x,b)\text{`}a' \quad \text{(TL–12)} \\ & =_X & (a',b) \quad \text{(TL–17)} \end{array}$$

and similarly $(a',b) =_X (a',b')$. The result follows from TL–9.

Section 6.4

1. We must show that for $f : C \times A \to B$, $\text{eval} \circ (\lambda f \times A) = f$. First note that $\lambda f \times A = \langle \lambda f \circ p_1, p_2\rangle : C \times A \to [A \to B] \times A$. (See 5.2.17.) Now let z be a variable of type C, y a variable of type A which is not in X, and suppose f is determined by a term $\phi(z,y)$ of type B. Then $\lambda f \times A$ is represented by $(\lambda_y \phi(z,y) \circ z, y) =_X (\lambda_y \phi(z,y), y)$ by definition of composition in $C(\mathscr{L})$. Then $\text{eval} \circ (\lambda f \times A)$ is

$$(p_1(\lambda_y \phi(z,y), y))\text{`}p_2(\lambda_y \phi(z,y), y) =_X \lambda_y \phi(z,y)\text{`}y$$

by TL–15 and Exercise 2 of Section 6.3. Since $y \notin X$, this is $\lambda_y \phi(z,y)$ by TL–18.

2.

$$\begin{array}{rcl} \Gamma(\Lambda(\phi(u))) & = & \Gamma(\lambda_x \phi((z,x))) \\ & = & \lambda_x \phi(\text{proj}_1\, u, x)\text{`}\,\text{proj}_2\, u \\ & =_X & \phi((\text{proj}_1\, u, \text{proj}_2\, u)) \\ & =_X & \phi(u) \end{array}$$

Section 6.5

1. a. $\mathbf{N} \times \mathbf{N} \xrightarrow{\langle p_1, p_1, p_2, p_2\rangle} \mathbf{N} \times \mathbf{N} \times \mathbf{N} \times \mathbf{N} \xrightarrow{* \times *} \mathbf{N} \times \mathbf{N} \xrightarrow{+} \mathbf{N}$

b. $\mathbf{N} \times \mathbf{N} \times \mathbf{N} \xrightarrow{\langle p_1, p_1, p_3, p_2\rangle} \mathbf{N} \times \mathbf{N} \times \mathbf{N} \times \mathbf{N} \xrightarrow{* \times *} \mathbf{N} \times \mathbf{N} \xrightarrow{+} \mathbf{N}$

c. $\mathbf{N} \times \mathbf{N} \xrightarrow{()} 1 \xrightarrow{5} \mathbf{N}$

Solutions for Chapter 7

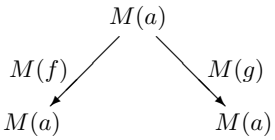
Section 7.1

1. Let us temporarily denote the usual addition by $\oplus$. The fact that $0 + m = m$ implies that $k + m = k \oplus m$ when $k = 0$. Assuming that equation for some k, we have that

$$\mathsf{succ}(k) + m = \mathsf{succ}(k + m) = \mathsf{succ}(k \oplus m) = \mathsf{succ}(k) \oplus m$$

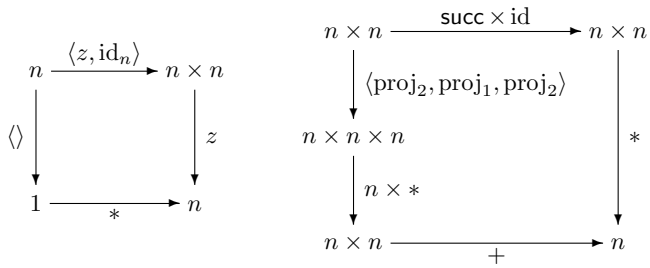
2. The cone requires that $M(a) = M(a) \times M(a)$. If $M(a)$ is finite then the number of elements of $M(a) \times M(a)$ is the square of the number of elements of $M(a)$. Hence $M(a)$ has either no elements or one element or an infinite number.

To see that $\mathscr{S}$ has an infinite model, define $M(a) = \mathbf{N}$, $M(f)(n) = r$ and $M(g)(n) = s$, where (r, s) is the unique pair of nonnegative integers for which $n-1 = 2^r(2s+1)$. It follows from the unique factorization of integers that $n \mapsto (r, s)$ is a bijection from $\mathbf{N}$ to $\mathbf{N} \times \mathbf{N}$. By Proposition 5.2.3, this means that



is a product diagram.

3. What we must do is to add an operation and equations to implement the standard inductive definition of multiplication: $0 * m = 0$ and $\mathsf{succ}(k) * m = m + k * m$. We do this by adding one operation $* : n \times n \to n$ and diagrams

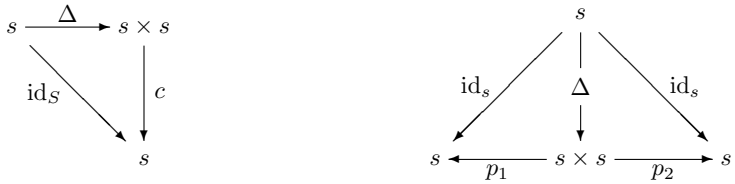


In order to make these diagrams, we also have to add arrows $n \to n \times n$, $n \times n \to n \times n$, $n \times n \to n \times n \times n$ and $n \times n \times n \to n \times n$ and, following the pattern of similar constructions in 7.1.7, diagrams forcing them to be $\langle z, \mathrm{id}_n \rangle$, $\mathsf{succ} \times \mathrm{id}$, $\langle \mathrm{proj}_2, \mathrm{proj}_1, \mathrm{proj}_2 \rangle$ and $n \times *$ respectively.

Section 7.2

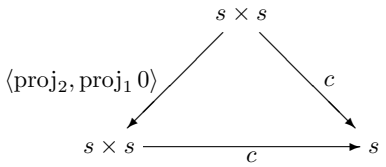
1. If we write xy for $M(c)(x, y)$ for $x, y \in M(s)$, then the diagram requires that $xy = x$ for all $x, y \in M(s)$. Then $x(yz) = x$ and $(xy)z = xz = x$. Of course, most semigroups do not have such a multiplication, so this sketch is not a sketch for semigroups.

2. One arrow $\Delta : s \to s \times s$ is needed, with the following diagrams:

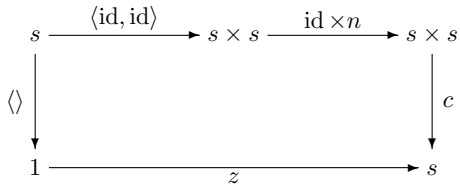


3. We give the answer for the case of real vector spaces; the other is similar. We assume the real number field $\mathbf{R}$ as a given structure. We suppose, for each $r \in R$, a

unary operation we will denote $r^* : s \to s$. We require a unit element $z : 1 \to s$ for the operation c and a diagram similar to the previous exercise to say that z is the unit element. The following diagram says that c is commutative:



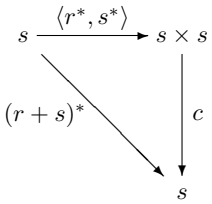
We have to add a unary negation operator $n : s \to s$ together with a diagram to say it is the negation operator:



In addition, we need diagrams that express the following identities:

$$\begin{aligned} 0^*(x) &= z \\ r^*(c(x,y)) &= c(r^*(x), r^*(y)) \\ (r+s)^*(x) &= c(r^*(x), s^*(x)) \\ 1^*(x) &= x \\ r^*(s^*(x)) &= (rs)^*(x) \end{aligned}$$

We give, for example, the diagram required to express the third of the equations above:



Section 7.4

1. We define a functor $F : \mathbf{Mod}(\mathscr{S}, \mathbf{Set}) \to \mathbf{Set}$: Given a model M of $\mathscr{S}$, let $F(M) = M(s)$. Given a homomorphism $\alpha : M \to M'$ of models, let $F(\alpha) = \alpha s : M(s) \to M(s)$. Define the inverse $G : \mathbf{Set} \to \mathbf{Mod}(\mathscr{S}, \mathbf{Set})$ this way: For a set A, $G(A)$ is the model M with $M(s) = A$ and multiplication $M(c) : A \times A \to A$ defined by $M(c)(a, a') = a$. $M(p)$ and $M(q)$ are necessarily the projections and clearly $M(c) = M(p)$ as required by the lone diagram of $\mathscr{S}$. If $\phi : A \to B$ is a set function, define the homomorphism $G(\phi) : G(A) \to G(B)$ by $G(\phi)s = \phi$ and $G(\phi)(s \times s) = \phi \times \phi$. It is easy to see that this is a homomorphism of models and that F and G are inverse functors.

2. Let the functor $F : \mathbf{Mod}(\mathscr{S}, \mathscr{C}) \to \mathbf{Mod}(\mathscr{T}, \mathscr{C})$ take a model M of $\mathscr{S}$ to the model $F(M)$ of $\mathscr{T}$ that is the same as M on the nodes and arrows with the same labels, and such that $F(M)(c) = M(a) = F(M)(a)$, $F(M)(h) = F(M)(j) = \mathrm{id}_{F(M)(a)}$ and $F(M)(k) = M(f)$. Given a natural transformation $\alpha : M \to M'$, let $F(\alpha)$ have the same components as α on the nodes of $\mathscr{S}$, and set $F(\alpha)c = \alpha a$.

Define $G : \mathbf{Mod}(\mathscr{S}, \mathscr{C}) \to \mathbf{Mod}(\mathscr{T}, \mathscr{C})$ so that $G(K)$ is K restricted to the nodes and arrows of $\mathscr{S}$. For $\beta : K \to K'$ define $G(\beta)$ to have the same component as β on a and b.

Then F is an equivalence of categories with pseudoinverse G. $G \circ F$ is actually the identity functor on $\mathscr{S}$. The required natural isomorphism η from $F \circ G$ to the identity functor on $\mathscr{T}$ has $\eta a = \mathrm{id}_a$, $\eta_b = \mathrm{id}_b$ and $\eta c = h$.

Section 7.6

1. It is the model I with $I(d) = \{a, b\}$ and $I(l) = \emptyset$.

2. Let I be the initial model. $I(l)$ contains u, so it contains elements obtained by repeatedly applying tail to u. Let us write tu for $\mathsf{tail}(u)$, t^2u for $\mathsf{tail}(\mathsf{tail}(u))$, and so on. The operation head can be applied to each of these, producing a countably infinite list of elements ht^nu of $I(d)$, one for each $n \in \mathbf{N}$. By the product property, we then get all possible pairs (ht^mu, t^nu) for $m, n \in \mathbf{N}$. These pairs include the elements t^nu because $t^nu = (ht^nu, t^{n+1}u)$. Again by the product property we get elements $(ht^{m_1}u, (ht^{m_2}u, t^nu))$, $(ht^{m_1}u, (ht^{m_2}u, (ht^{m_3}u, t^nu)))$ and so on.

This suggests simplifying the notation further as follows: We define $I(l)$ to be the set of equivalence classes of finite sequences $s = (s_1, s_2, s_3, \ldots, s_m)$ of all possible nonzero lengths m, with all $s_i \in \mathbf{N}$, using the equivalence relation $\sim$ generated by requiring that $s \sim s'$, where s' is obtained from s by adjoining $s_m + 1$ as the $m + 1$st entry. (Thus $(3, 1) \sim (3, 1, 2) \sim (3, 1, 2, 3)$, etc.) We denote the equivalence class of s by $[s]$. Note that all elements of an equivalence class have the same first entry and that every equivalence class contains entries of length greater than 1. Now define $I(d) = \mathbf{N}$, $\mathsf{head}([s]) = s_1$ and $\mathsf{tail}([s])$ the the equivalence class of the sequence obtained by dropping the first entry of a representative of $[s]$ of length greater than 1.

The only problematical thing to verify is that $I(C)$ is a product cone. Suppose $f : X \to \mathbf{N}$ and $g : X \to I(l)$ are given. By definition of head and tail there is only one possible map $h : X \to I(l)$ that makes $\mathsf{head} \circ h = f$ and $\mathsf{tail} \circ h = g$, and that is for $h(x)$ to be the equivalence class of the infinite sequence s with $s_1 = f(x)$ and $s_k = g(x)_{k-1}$ for $i > 1$, and that definition does indeed work.

To see that I is an initial model, let M be any model. Define a natural transformation $\alpha : I \to M$ as follows. Let $I(u) = M(u)$. If $s \in [s]$ has length greater than 1, let s' denote the sequence obtained by deleting the first entry of s, and define

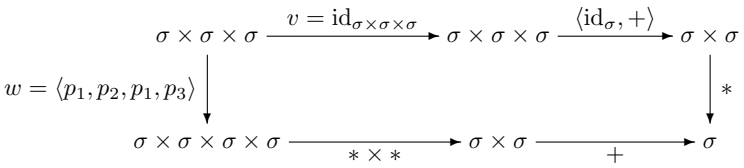
$$\alpha l([s]) = (M(\mathsf{head})(M(\mathsf{tail})^{s_1}(M(u))), \alpha l([s']))$$

This is the only possible definition for a natural transformation from I to M and it is easy to prove inductively that it is well-defined and a natural transformation.

3. Let the elements of X be x and y. Form the set $\mathbf{N}_x = \{(n, x) \mid n \in \mathbf{N}\}$ and similarly $\mathbf{N}_y = \{(n, y) \mid n \in \mathbf{N}\}$. Then $S = \mathbf{N} \vee \mathbf{N}_x \vee \mathbf{N}_y$ is certainly the disjoint union of three copies of $\mathbf{N}$. We can identify x with $(0, x)$ and y with $(0, y)$ so that, up to isomorphism, $x \in S$ and $y \in S$. We define succ on S by $\mathrm{succ}(n) = n + 1$, $\mathrm{succ}(n, x) = (n + 1, x)$ and $\mathrm{succ}(n, y) = (n + 1, y)$. With these definitions and no relations, this is the initial term model.

Section 7.7

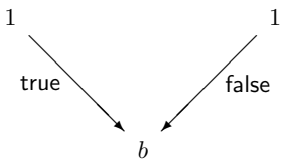
1. Let the sort on which the operations are defined be σ. Thus $* : \sigma \times \sigma \longrightarrow \sigma$ and $+ : \sigma \times \sigma \longrightarrow \sigma$. The diagram is then



Solutions for Chapter 8

Section 8.1

1. This should have two sorts 1 and b. There should be an operation $\mathsf{true} : 1 \longrightarrow b$, an operation $\mathsf{false} : 1 \longrightarrow b$ and a cocone

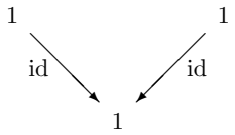


We need a cone with empty base to express that 1 is terminal. We should have operations $\vee : b \times b \longrightarrow b$, $\wedge : b \times b \longrightarrow b$ and $\neg : b \longrightarrow b$. We will need diagrams to express equations like

$$\begin{aligned} \neg\,\mathsf{true} &= \mathsf{false} \\ \neg\,\mathsf{false} &= \mathsf{true} \\ \neg \wedge (x, y) &= \vee(x, y) \\ \vee(0, 0) &= 0 \\ \vee(0, 1) &= 1 \\ \vee(1, 0) &= 1 \\ \vee(1, 1) &= 1 \end{aligned}$$

There are other equations, but they all follow from these. In particular, we do not, in this case, have to express the distributive laws since they follow, there being so few elements.

2. If we have one sort 1, one empty cone with vertex 1 and one cocone



then there is no model in sets because the model must take 1 to the one-element set and that must satisfy that $1 = 1 + 1$, which is impossible. It follows from Exercise 4 of Section 9.6 that this sketch does have a model in the category of monoids.

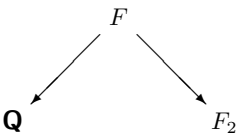
Section 8.2

1. This is most readily done using elements, although it can all be done with diagrams. If $x * x^{-1} = 1$ and $y * y^{-1} = 1$, then

$$\begin{aligned}(x * y) * (y^{-1} * x^{-1}) &= x * (y * (y *^{-1} *x^{-1})) = x * ((y * y^{-1}) * x^{-1}) \\ &= x * (1 * x^{-1}) = x * x^{-1} = 1\end{aligned}$$

If $2 \neq 0$ in a field, then 2^{-1} exists, whence $(2 * 2)^{-1} = 4^{-1}$ also exists. Hence $4 \neq 0$.

2. Let $\mathbf{Q}$ and F_2 denote the fields of rational numbers and the two-element field respectively. In order to have a product of $\mathbf{Q}$ and F_2, we have to have a cone

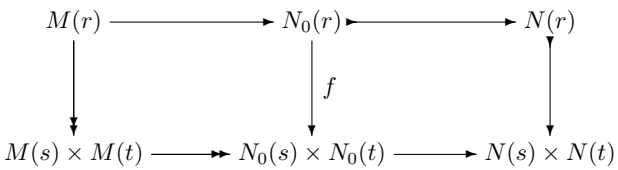


which is exactly what we do not have in the category of fields.

Section 8.3

1. a. Several things have to be shown. First that the image $N_0 \subseteq N$ is closed under the operations (that is if all the arguments of the operation are in the image, so does the value); second that the diagrams continue to commute; third that the cones remain products; and fourth that the cocones remain sums.

Let $u : s_1 \times s_2 \times \cdots \times s_n \longrightarrow s$ be an operation. For $i = 1, \ldots, n$ let $x_i \in M(s_i)$ and $x = u(x_1, \ldots, x_n)$. Then $f(x) = u(f(x_1), \ldots, f(x_n))$, which demonstrates the first point. Since the operations in N_0 are the restrictions of those in N any two paths that agree on N do so on N_0 (actually, whether or not they do on M). For the cones, we illustrate on binary cones. Suppose $s \leftarrow r \longrightarrow t$ is a cone in the sketch. Then in the diagram



certain functions have been labeled as being surjective or injective (although others are and some are bijective). These use the facts that the product of two injective functions is injective and the product of two surjective functions is surjective. That f is surjective follows from the fact that the composite of two surjective functions is surjective and if the composite of two functions is surjective, so is the second one. Dually, the fact that f is injective uses the facts that the composite of two injective functions is injective and that if the composite of two functions is injective, the first one is.

The dual argument, using the fact that a sum of injectives is injective and a sum of surjectives is surjective, gives the corresponding result for discrete cocones.

It is worth noting that these arguments fail if either the cones or cocones fail to be discrete or if the category in which the models are taken is other than the category of sets. The reason is that even in the category of sets, the arrow induced between equalizers of epis is not necessarily epic and between coequalizers of monos will not be monic. (Equalizers and coequalizers are defined in chapter 9.) In categories other than sets, even the sum of epics will not generally be epic, nor the sums of monics monic.

b. There are two ways of doing this. One is to show that the intersection of submodels is a submodel. Then the intersection of all submodels is clearly the smallest submodel. Another is to take the image of the initial term model in the component of that model. Let M be the initial term model, N is the given model and N_0 this submodel. If $N_1 \subseteq N$ is any other submodel, there is an initial term model M' in the component of N_1 that has an arrow $M' \to N_1$. But $N_1 \subseteq N$ and so is in the same component as N. Thus $M' = M$ and the map $M \to N_1 \subseteq N$ is the original map $M \to N$. Since that factors through N_1, it follows that $N_0 \subseteq N_1$.

Solutions for Chapter 9

Section 9.1

1. Let us write 1 for an identity arrow. We claim that f is the equalizer of 1 and $f \circ g$. In fact, $f \circ g \circ f = f = 1 \circ f$ so $f \circ g$ and 1 equalize f. If $h : C \to B$ is an arrow such that $f \circ g \circ h = 1 \circ h = h$, then $g \circ h : C \to A$ satisfies $f \circ g \circ h = h$. Since f is monic, this arrow is unique.

2. Suppose by induction that this is true for every list of $n-1$ parallel arrows and $f_1, \ldots, f_n : A \to B$ is a list of n parallel arrows. Let $j : E \to A$ be an equalizer of $f_1, \ldots, f_{n-1}$. We may clearly suppose that $n \geq 3$. Then the parallel pair $f_1 \circ j, f_n \circ j : E \to B$ has an equalizer $k : F \to E$. I claim that $j \circ k : F \to A$ is an equalizer of the list. In fact, for $i < n$, $f_i \circ j \circ k = f_1 \circ j \circ k$, because j simultaneously equalizes all those arrows, while $f_n \circ j \circ k = f_1 \circ j \circ k$ because k equalizes $f_1 \circ j$ and $f_n \circ j$. If $h : C \to A$ is simultaneously equalized by $f_1, \cdots, f_n$, then there is a unique $m : C \to E$ such that $j \circ m = h$. Since $f_1 \circ j \circ m = f_1 \circ h = f_n \circ h = f_n \circ j \circ m$, there is a unique $g : C \to F$ such that $k \circ g = m$. Then $j \circ k \circ g = j \circ m = h$. If $j \circ k \circ g' = h$, then $g' = g$ because $j \circ k$ is monic; hence g has the required uniqueness property.

3. Let A and B be monoids and $f, g : A \to B$ be monoid homomorphisms. Let $E = \{x \in A \mid f(a) = g(a)\}$. Then $f(1) = 1 = g(1)$ so that $1 \in E$. Also if $x, y \in E$, then $f(xy) = f(x)f(y) = g(x)g(y) = g(xy)$ so that $xy \in E$ and E is a submonoid of A. Let $j : E \to A$ be the inclusion homomorphism. Now let $h : C \to A$ be a monoid homomorphism with $f \circ h = g \circ h$. Then for all $x \in C$, $f(h(x)) = g(h(x))$ so that $h(x) \in E$. Thus $h(C) \subseteq E$ so there is a function $k : C \to E$ with $j \circ k = h$. It has to be shown that k is a homomorphism, but $k(x) = h(x)$ for all $x \in C$, so that is trivial. The fact that j is injective makes the uniqueness of k evident.

4. a. For x and y to have an equalizer, we would need, at least, an element z with $xz = yz$ and this never happens in a free monoid.

b. Suppose the element x is the equalizer of y and z. Then x is monic, which means, according to the cited exercise, that x is invertible. But then $yx = zx$ implies that $y = yxx^{-1} = zxx^{-1} = z$.

5. A monomorphism in **Set** is an injective function (see Theorem 2.8.3), so let $f : A \to B$ be an injective function. Let C be the set of all pairs

$$\{(b, i) \mid b \in B,\ i = 0, 1\}$$

and impose an equivalence relation on these pairs forcing $(b, 0) = (b, 1)$ if and only if there is an $a \in A$ with $f(a) = b$ (and not forcing $(b, i) = (c, j)$ if b and c are distinct). Since f is injective, if such an a exists, there is only one. Let $g : B \to C$ by $g(b) = (b, 0)$ and $h : B \to C$ by $h(b) = (b, 1)$. Then clearly $g(b) = h(b)$ if and only if there is an $a \in A$ with $f(a) = b$. Now let $k : D \to B$ with $g \circ k = h \circ k$. It must be that for all $x \in D$, there is an $a \in A$, and only one, such that $k(x) = f(a)$. If we let $l(x) = a$, then $l : D \to A$ is the unique arrow with $f \circ l = k$.

6. The reason v and w are inverse to each other is that there is no other arrow for $v \circ u$ to be but id_C and similarly $u \circ v = \mathrm{id}_D$. Let $F : \mathscr{A} \to \mathscr{B}$ be the inclusion. For any functors $G, H : \mathscr{B} \to \mathscr{C}$, if $G \circ F = H \circ F$, then $G(u) = H(u)$. But then

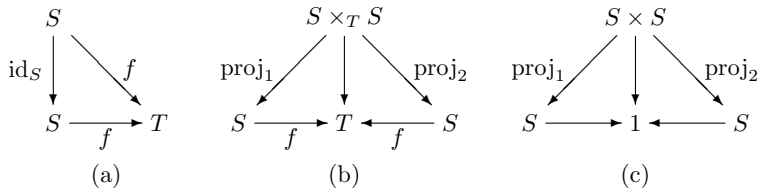
$$G(v) = G(u^{-1}) = G(u)^{-1} = H(u)^{-1} = H(u^{-1}) = H(v)$$

Since the objects of $\mathscr{B}$ and the other arrows are in the image of F, it follows that $G = H$. Since F is not an isomorphism, it cannot be a regular monomorphism by Proposition 9.1.8.

Section 9.2

1. If D is the base of Diagram (9.1), then $\mathrm{cone}(E, D)$ is the set of all commutative cones of that shape. The equalizer of f and g is the universal element of the functor F of the proof of Proposition 9.1.4. These two functors are naturally isomorphic by an isomorphism $\phi : F \to \mathrm{cone}(-, D)$: if X is any object and $u : X \to A$ is in $F(X)$ (so $f \circ u = g \circ u$), then $\phi A(u)$ is the cone with components $u : X \to A$ and $f \circ u : X \to B$. By its very definition this is a commutative cone, so an element of $\mathrm{cone}(X, D)$. Since u determines $f \circ u$ uniquely, this function ϕA is a bijection, so ϕ is a natural isomorphism. Since the two functors are isomorphic, so are the objects that represent them, by the uniqueness of universal elements.

2. The limits are as indicated:



where $S \times_T S$ is the subset $\{(s, s') \mid f(s) = f(s')\}$ of $S \times S$. (This is standard notation to be introduced in Section 9.3.)

3. a. An arrow $f : C \to \prod_{i \in \mathscr{I}} Di$ is simply a collection of arrows $f_i : C \to Di$, one for each object of $\mathscr{I}$. In order that it be a commutative cone on the diagram D it is necessary and sufficient that it satisfy the additional condition that $Da \circ f_j = f_k$. This is equivalent to

$$\text{proj}_k \circ f = f_k = Da \circ f_j = Da \circ \text{proj}_j \circ f$$

which is a necessary and sufficient condition that f factor through $E \to \prod Di$. Thus a cone over D is equivalent to an arrow to E, which means that E is a limit of D.

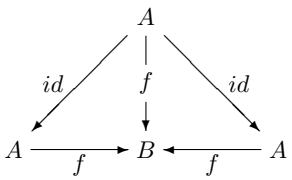
b. An arrow $f : C \to \prod Di$ is a collection of arrows $f_i : C \to Di$. In order that it be a commutative cone on D it must simultaneously satisfy the conditions $Da \circ f_j = f_k$ and $Db \circ f_l = f_m$. This is equivalent to $\langle f_i, f_l \rangle : C \to Di \times Dl$ and $\langle f_k, f_m \rangle : C \to Dk \times Dm$ satisfying $(Da \times Db) \circ \langle f_j, f_l \rangle = \langle f_k, f_m \rangle$, in turn equivalent to $r \circ f = s \circ f$, that is to factoring through $E \to \prod Di$.

c. Again an arrow $f : C \to A$ is a family of arrows $f_i : C \to Di$. In order to be a commutative cone on D it must satisfy $Da \circ f_{\text{source}(a)} = f_{\text{target}(a)}$ for every arrow $a \in \mathscr{I}$. This is exactly the condition that $r \circ f = s \circ f$, which means that f is a cone over D if and only if it factors through $E \to A$. Hence a cone over D is equivalent to an arrow to E, which means that E is the limit.

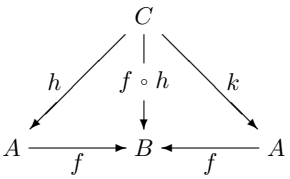
Section 9.3

1. Let $f : R \to S$ and $k : R \to A$ be functions such that $g \circ f = i \circ k$. Then for each $x \in R$, $g(f(x)) \in A$ which means that $f(x) \in g^{-1}(A)$. Thus f factors through $g^{-1}(A)$ by the corestriction $m : R \to g^{-1}(A)$ of f to $g^{-1}(A)$, so $f = j \circ m$. Also $i \circ h \circ m = i \circ k$, so $h \circ m = k$ because i is monic.

2. Suppose that $f : A \rightarrowtail B$ is monic. Then the only way you can have $g, h : C \to A$ such that $f \circ g = f \circ h$ is if $g = h$. In that case $g : C \to A$ is the unique arrow such that $\text{id} \circ g = g$ and $\text{id} \circ g = h$ so that



satisfies the condition of being a pullback cone. Thus (a) implies (b). It is obvious that (b) implies (c). Now suppose that the diagram in (c) is a pullback. Let $h, k : C \to A$ be arrows such that $f \circ h = f \circ k$. Then we have a cone



so that there is an arrow $l : C \to P$ such that $h = g \circ l = k$. Thus (c) implies (a).

3. Suppose $h, k : D \to P$ with $p_2 \circ h = p_2 \circ k$. We have

$$f \circ p_1 \circ h = g \circ p_2 \circ h = g \circ p_2 \circ k = f \circ p_1 \circ k$$

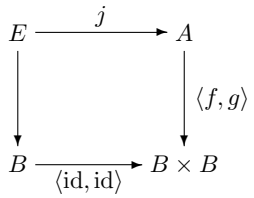
and f is monic by assumption so that $p_1 \circ h = p_1 \circ k$. Thus $x = h$ and $x = k$ are solutions to the equation $p_1 \circ x = p_1 \circ h$ and $p_2 \circ x = p_2 \circ h$. But the definition of pullback requires that solution to be unique so that $h = k$.

4. Let $q_1 : S \times U \to S$ and $q_2 : S \times U \to U$ be the product projections. Suppose there are arrows $u, v : X \to P$ such that $\langle p_1, p_2 \rangle \circ u = \langle p_1, p_2 \rangle \circ v$. Then

$$\begin{aligned} f \circ p_1 \circ u &= f \circ q_1 \circ \langle p_1, p_2 \rangle \circ u \\ &= f \circ q_1 \circ \langle p_1, p_2 \rangle \circ v \\ &= f \circ p_1 \circ v \end{aligned}$$

and similarly $g \circ p_2 \circ u = g \circ p_2 \circ v$ so $u = v$ by the uniqueness part of the universal property of pullbacks.

5. If $f, g : A \to B$, then an equalizer is the arrow $j : E \to A$ in a pullback



The proof of this fact comes down to showing that any pullback of this square and an equalizer of f and g represent equivalent functors: the functor G representing a pullback of the square has value

$$\begin{aligned} G(X) &= \{(u : X \to A, v : X \to B) \mid \langle \mathrm{id}, \mathrm{id} \rangle \circ v = \langle f, g \rangle \circ u\} \\ &= \{(u, v) \mid \langle v, v \rangle = \langle f \circ u, g \circ u \rangle\} \\ &= \{(u, v) \mid f \circ u = v = g \circ u\} \\ &\cong \{u \mid f \circ u = g \circ u\} \end{aligned}$$

which is the value of the functor representing an equalizer of f and g. (Compare the proof of Exercise 1 of Section 9.2.)

6. A pullback of a diagram $A \to 1 \leftarrow B$ is just a product $A \times B$. We know from the preceding exercise that a category with products and pullbacks has equalizers. Hence such a category has finite products and equalizers. The conclusion now follows from Corollary 9.2.11.

7. In **Set**, a pullback is

$$P = \{(a, b) \mid f(a) = g(b)\}$$

Since f is surjective, for any $b \in B$, there is a $a \in A$ such that $f(a) = g(b)$. Then $p_2(a, b) = b$. Thus p_2 is surjective. The condition now follows from Proposition 2.9.2.

8. In the category of monoids, we have seen in 2.9.3 that the inclusion of the monoid $\mathbf{N}$ of nonnegative integers into the monoid $\mathbf{Z}$ of integers is an epimorphism. Let $A \subseteq \mathbf{Z}$ denote the submonoid of nonpositive integers. Both inclusions are epic, but their pullback, which is the intersection, is the submonoid $\{0\}$ and the maps to both $\mathbf{N}$ and A are not epic. In fact all pairs of arrows on $\mathbf{N}$ (or on A) agree on $\{0\}$.

9. Suppose the arrow $g : A \to C$ in Diagram (9.2) is a split epi, so that there is an arrow $h : C \to A$ so that $g \circ h = \mathrm{id}_C$. Then we have $\mathrm{id} : B \to B$ and $h \circ f : B \to A$ satisfy $f \circ \mathrm{id} = g \circ h \circ f$ so there is an arrow $k : B \to P$ such that $p_1 \circ k = h \circ f$ and $p_2 \circ k = \mathrm{id}_B$. This latter identity is what we want.

10. Suppose we have arrows $A \to B$ and $A \to C$ giving a commutative cone. By composing the latter with $C \to D$, we get a pair of arrows $A \to B$ and $A \to D$ giving a commutative cone and this leads to a unique arrow $A \to Q$. We now have $A \to C$ and $A \to Q$ giving a commutative cone and this leads to a unique arrow $A \to P$ making the left hand cone commute. Clearly, the outer rectangle commutes as well.

It says that the weakest precondition of the composite of two procedures can be calculated as the weakest precondition under the second procedure of the weakest precondition under the first.

Section 9.4

1. In Exercise 9.1.5 of Section 9.1, we showed that any equalizer is a monomorphism. Interpreted in the dual category, it is the result of this exercise.

2. By Proposition 2.9.2, every epimorphism in **Set** is surjective. We claim that every surjective function is a regular epimorphism. In fact, let $f : S \to T$ be surjective and suppose $E = \{(x, y) \in S \times S \mid f(x) = f(y)\}$. Let $p, q : E \to S$ be the first and second projections. We claim that f is the coequalizer of p and q. Clearly $f \circ p = f \circ q$. Let $h : S \to R$ such that $h \circ p = h \circ q$. Define $k : T \to R$ as follows. For $t \in T$, there is an $s \in S$ with $f(s) = t$. We would like to define $k(t) = h(s)$. If s' is another element of S with $f(s') = t$, then $(s, s') \in E$ and so $h(s) = (h \circ p)(s, s') = (h \circ q)(s, s') = h(s')$. Thus k is well defined and clearly $k \circ f = h$. The uniqueness of k follows from the fact that f is epic.

3. In Proposition 9.1.8, it is shown that a regular mono that is an epimorphism is an isomorphism. The dual says that a monomorphism that is a regular epimorphism is an isomorphism. In 2.9.3 the inclusion was shown to be epic. Since the inclusion is evidently a monomorphism and not an isomorphism, it cannot be a regular epimorphism.

4. We use the terminology of Exercise 2, just assuming that S and T are monoids and f a monoid homomorphism. First we have to say that if f is not surjective, then it still may be an epimorphism, as the inclusion of $\mathbf{N}$ into $\mathbf{Z}$ shows, but it cannot be regular. The reason is the easily verified fact that the image of a monoid homomorphism $f : S \to T$ is a submonoid $T_0 \subseteq T$ and if it is not all of T, the properties of regular epimorphism and the fact that the inclusion is a monomorphism combine to provide an arrow $T \to T_0$ such that the composite $T \to T_0 \to T$ is the identity of T. This is possible if and only if $T_0 = T$.

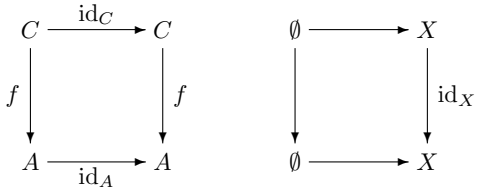
To go the other way, we need add to the construction of Exercise 2 only the facts that $E \subseteq S \times S$ is a submonoid and that all the arrows constructed are monoid homomorphism. Of all of these, only the fact that k is a monoid homomorphism is interesting. In fact, if t and t' are elements of T and if s, $s' \in S$ are such that $f(s) = t$ and $f(s') = t'$, then since f is a monoid homomorphism, $f(ss') = f(s)f(s') = tt'$ so that $k(tt') = h(ss') = h(s)h(s') = k(t)k(t')$. Similarly, since f is a monoid homomorphism, $f(1) = 1$, so $k(1) = h(1) = 1$.

5. A kernel pair of f is characterized by the following mapping property: $f \circ d^0 = f \circ d^1$ and if e^0, $e^1 : C \to A$ satisfies $f \circ e^0 = f \circ e^1$, then there is a unique $g : C \to K$ such that $d^i \circ g = e^i$, $i = 0, 1$. Now if f is the coequalizer of e^0 and e^1 and the kernel pair exists, let $g : C \to K$ as described. Suppose $h : A \to D$ is an arrow such that $h \circ d^0 = h \circ d^1$. Then $h \circ e^0 = h \circ d^0 \circ g = h \circ d^1 \circ g = h \circ e^1$ so that there is a unique $k : B \to D$ with $k \circ f = h$. Thus f is the coequalizer of d^0 and d^1.

Section 9.5

1. Suppose E is some object and $\langle l|l'\rangle : C + C' \to E$ and $\langle m|m'\rangle : B + B' \to E$ satisfy $\langle l|l'\rangle \circ (g + g') = \langle m|m'\rangle \circ (f + f')$. If i and j represent the inclusions of the components, then by definition, $\langle l|l'\rangle \circ (g+g') \circ i = \langle l|l'\rangle \circ i \circ g = l \circ g$ and similarly $\langle m|m'\rangle \circ (f + f') \circ i = m \circ f$ so these equations imply that $l \circ g = m \circ f$. By using j we similarly conclude that $l' \circ g' = m' \circ f'$. The mapping properties of the pushout imply the existence of $n : D \to E$ and $n' : D' \to E$ such that $n \circ k = l$, $n \circ h = m$ $n' \circ k' = l'$ and $n' \circ h' = m'$. Then $\langle n|n'\rangle$ is the required arrow. Uniqueness follows from the uniqueness of the components, together with the uniqueness of an arrow from a sum, given its components.

2. a. Since e is injective, $e(C) \cong C$. Up to isomorphism, the diagram is the sum of the following two:



and the preceding exercise completes the argument.

b. e is an arbitrary monomorphism and the function i is injective.

3. a. As suggested by the hint, we take pairs of natural numbers with coordinatewise addition and subject to the relation that for any $a \in \mathbf{N}$, $(b, c) = (a + b, a + c)$. This relation is not an equivalence relation (it is not symmetric because a cannot be negative), but the symmetric closure is an equivalence relation. We will show that the quotient is isomorphic to $\mathbf{Z}$. To do this we define a function $f : \mathbf{N} \times \mathbf{N} \to \mathbf{Z}$ by $f(b, c) = b - c$. Clearly, $f(a + b, a + c) = f(b, c)$ so that $f \circ g = f \circ h$. On the other hand, if $f(b, c) = f(b', c')$, then $c - b = c' - b'$ or $c - c' = b - b'$. If $c \leq c'$, then $(b, c) = g(c' - c, b, c)$ while $(b', c') = h(c' - c, b, c)$. If $c' \leq c$, their roles are reversed. In either case, f is the quotient by the generated equivalence relation. The function f is also surjective, since every $n \geq 0$ is $f(n, 0)$, while every $n < 0$ is $f(0, -n)$. Since $f(0, 0) = 0$ and $f(b+b', c+c') = b+b'-(c+c') = b-c+b'-c' = f(b, c)+f(b', c')$ so that f is a monoid homomorphism and the coequalizer is the coequalizer in the category of monoids.

b. Let us denote this subset by A. The function $f : A \to \mathbf{Z}$ defined by

$$f(n, m) = \begin{cases} n & \text{if } m = 0 \\ -m & \text{if } n = 0 \end{cases}$$

is obviously bijective. It is a matter of consideration of cases to see that it is additive if addition is defined in A as suggested above.

4. Define $h : \mathbf{Z} \times \mathbf{N}^+ \to \mathbf{Q}$ by $h(b, c) = b/c$. Since $b/c = (ab)/(ac)$, $h \circ f = h \circ g$. Moreover, if $h(b, c) = h(b', c')$, then $bc' = b'c$. We have the equations

$$\begin{aligned} f(c', b, c) &= (b, c) \\ g(c', b, c) &= (c'b, c'c) \\ f(c, b', c') &= (b', c') \\ g(c, b', c') &= (cb', cc') \end{aligned}$$

Thus the coequalizer of f and g must render (b, c) and (b', c') equal. Thus h is injective. It is clearly surjective.

5. Let X be a set, $r : X \to B + B'$ and $s : X \to C + C'$ functions such that $(h + h') \circ r = (k + k') \circ s = v$. If we let $Y = v^{-1}(D)$ and $Y' = v^{-1}(D')$, then $X = Y + Y'$. Moreover, it is clear that $r(Y) \subseteq B$, $r(Y') \subseteq B'$, $s(Y) \subseteq C$ and $s(Y') \subseteq C'$. Let $t : Y \to B$ and $t' : Y' \to B'$ be the restrictions of r to Y and Y', respectively. Then $r = \langle t|t' \rangle$. Similarly, we have $s = \langle u|u' \rangle$ for $u : Y \to C$ and $u' : Y' \to C'$. Moreover, $(h + h') \circ r = (k + k') \circ s$ is equivalent to $h \circ t = k \circ u$ and $h' \circ t' = k' \circ u'$. Since the original two squares were pullbacks, it follows that there are arrows $w : Y \to A$ and $w' : Y' \to A'$ such that $f \circ w = t$, $g \circ w = u$, $f' \circ w' = t'$ and $g' \circ w' = u'$. This implies that $(f + f') \circ \langle w|w' \rangle = \langle t|t' \rangle = r$ and $(g + g') \circ \langle w|w' \rangle = \langle u|u' \rangle = s$. We also have to show uniqueness, but the arguments are similar.

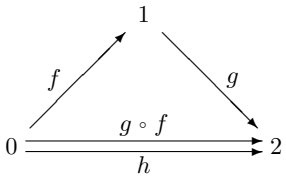
6. a. $e^0 : \mathbf{1} \to \mathbf{2}$ take the single object of $\mathbf{1}$ to one object of $\mathbf{2}$ and let e^1 take it to the other. Then to say of a functor $u : \mathbf{2} \to \mathscr{A}$ that $u \circ e^0 = u \circ e^1$ is simply to say that u takes the two objects to the same one. If a is the single arrow in $\mathbf{2}$, then source$(u(a))$ = target$(u(a))$ so that not only will there be $u(a)$, but also $u(a) \circ u(a)$ and $u(a) \circ u(a) \circ u(a)$ and so on. Now let $\mathbf{N}$ also denote the category with one object and the natural numbers as arrows as defined in the exercise. Then $q(a) \circ q(a) = 2$, $q(a) \circ q(a) \circ q(a) = 3$ and so on. Given a functor u as above, we can

define a functor $v : \mathbf{N} \to \mathscr{A}$ by $v(0) = \mathrm{id}_{\mathrm{source}(u(a))}$, $v(1) = u(a)$, $v(2) = u(a) \circ u(a)$ and so on. Clearly v is unique such that $v \circ q = u$. Thus q is the coequalizer of u and v.

b. In the diagram, the arrow from $\mathbf{1} + \mathbf{1}$ to $\mathbf{2}$ is inclusion and the arrow from $\mathbf{1} + \mathbf{1}$ to $\mathbf{N}$ is the only possible one. Let $v : \mathscr{A} \to \mathbf{2}$ and $w : \mathscr{A} \to \mathbf{N}$ be such that $q \circ v = t \circ w$. If f is an arrow in $\mathscr{A}$ then $q(v(f)) = t(w(f))$. But the only integer that is in the image of $q \circ v$ and $t \circ w$ is 0. Thus $q(v(f)) = 0$, which means that $v(f)$ is an identity. Thus v takes every arrow to an identity arrow, that is it factors through $\mathbf{1} + \mathbf{1}$, and the factorization is clearly unique.

The arrow $s : \mathbf{1} + \mathbf{1} \to \mathbf{N}$ is not even epic, let alone regular. In fact, $t \circ s = s = \mathrm{id}_{\mathbf{N}} \circ s$ without $t = \mathrm{id}_N$.

c. Let $\mathscr{D}$ be the category whose nonidentity arrows can be pictured as:



There is a functor $G : \mathscr{C} \to \mathscr{D}$ that is like F except that $G(h) = h$. This functor cannot factor through F because under any $H : \mathbf{3} \to \mathscr{D}$, $H(F(h)) = H(g \circ f) = H(g) \circ H(f) \neq h$.

d. A complete answer is too long, but we give enough details that the reader should have no difficulty filling in the rest. The first thing to observe is that being surjective on composable pairs of arrows includes, as special cases, being surjective on objects and on arrows. Suppose $T : \mathscr{A} \to \mathscr{B}$ is a functor which fails to be surjective on composable pairs, but is surjective on arrows and objects. Let f_1 and g_1 be a composable pair of arrows such that whenever $T(f_2) = f_1$ and $T(g_2) = g_1$, then f_2 and g_2 are not composable. Then let $S : \mathbf{3} \to \mathscr{B}$ be the unique functor such that $S(f) = f_1$ and $S(g) = g_1$. The pullback of S along T will be a category that includes subcategories like $\mathscr{C}$ of part (c) of this problem and other pieces. It will have a functor to the category $\mathscr{D}$ of the preceding part that takes every arrow lying above f to f, every arrow lying above g to g and every arrow lying above $g \circ f$ to h, which is not the composite $g \circ f$. Then just as in the preceding part, this functor makes all the identifications made by the functor to $\mathbf{3}$, but does not factor through that functor. Thus that functor is not a regular epi. If T fails to be epi on arrows or on objects, even easier arguments suffice.

For the other direction, in fact a functor that is surjective on composable pairs of arrows is a stable regular epi (which is more than the problem asked for). It is immediate that the condition of being surjective on composable pairs is stable under pullback, so it is sufficient to show that such an arrow is a regular epi. The argument is similar to that for monoids and we omit it.

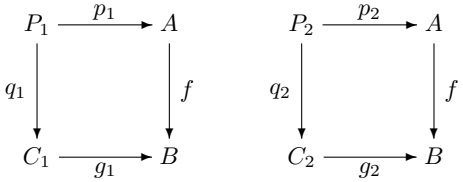
Section 9.6

1. The functor that assigns to each object the set of commutative cocones to that object on the base $A \leftarrow 0 \rightarrow B$ is naturally isomorphic to $\mathrm{Hom}(A, -) \times \mathrm{Hom}(B, -)$, which is the functor which assigns the set of cocones on the discrete base consisting

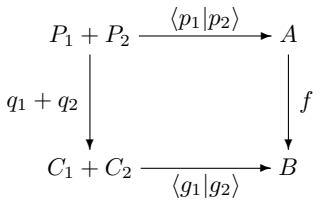
of A and B. The isomorphism forgets the arrow $0 \longrightarrow A + B$ and its inverse puts back the only arrow that can go from 0 to $A + B$. Thus the universal elements are isomorphic.

2. Use exactly the same argument as for the preceding exercise.

3. We have to show that if

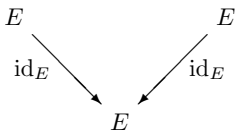


are both pullbacks, then so is



Let $i_1 : C_1 \longrightarrow C_1 + C_2$, $i_2 : C_2 \longrightarrow C_1 + C_2$, $j_1 : P_1 \longrightarrow P_1 + P_2$ and $j_2 : P_2 \longrightarrow P_1 + P_2$ be the coproduct injections. If $a \in A$ and $c \in C_1 + C_2$ such that $f(a) = \langle g_1 | g_2 \rangle(c)$, then either $c = i_1(c_1)$ for some $c_1 \in C_1$ and $f(a) = g_1(c)$ or $c = i_2(c_2)$ for some $c_2 \in C_2$ and $f(a) = g_2(c)$. In the first case, there is a unique $x_1 \in P_1$ such that $p_1(x_1) = a$ and $q_1(x_1) = c_1$. Then also $(q_1 + q_2)(j_1(x_1)) = j_1(q_1(x_1)) = j_1(c_1) = c$ and $\langle p_1 | p_2 \rangle(j_1(x_1)) = p_1(x_1) = a$. We get a similar conclusion if $c = j_2(c_2)$. This shows that $P_1 + P_2$ satisfies the existence condition of pullback. The uniqueness condition is similar.

4. Let us deal first with the monoid case. If M is a monoid and $f, g : E \longrightarrow M$ are monoid homomorphisms, then $f = g$ is the function taking the unique element of $e \in E$ to the identity of M. Hence $f : E \longrightarrow M$ is the unique monoid homomorphism such that $f \circ \mathrm{id}_E = f$ and $f \circ \mathrm{id}_E = g$. Thus the cocone



is a colimit.

In the category of semigroups, the situation is quite different. If $f : E \longrightarrow S$ is a semigroup homomorphism, $f(e)$ need not be an identity arrow, if any exists. Since $e^2 = e$, it must be that $f(e)^2 = f(e)$ (such an element of S is called an **idempotent**). Other than that, there is no restriction on what $f(e)$ can be. Now let S be the semigroup of endofunctions on $\mathbf{N}$. The elements u and v defined in the problem are idempotents such that for all n, $(u \circ v)^n$ are distinct. In fact, $(u \circ v)^n$

adds $2n$ to odd integers and $2n-2$ to even ones (and $(v \circ u)^n$ does the opposite). Now let $i : E \to F$ and $j : E \to F$ give a sum $F = E + E$. Let $i(e) = a$ and $j(e) = b$. Since there are arrows $f, g : E \to S$ defined by $f(e) = u$ and $g(e) = v$, there is an $h : F \to S$ such that $h \circ i = f$ and $h \circ j = g$. In F there are all the elements $(ab)^n$ and since $h((ab)^n) = (u \circ v)^n$ in order to be a semigroup homomorphism, all the $(ab)^n$ must also be distinct. Thus F is infinite.

5. This follows because they are each isomorphic to E in the previous example.

6. a. The recursive functions are defined as the smallest set of functions including successor and projections and closed under certain operations, of which the simplest is composition. (Details are in [Lewis and Papadimitriou, 1981], where recursive functions are called μ-recursive.) Since the identity is also recursive (it is, for one thing, the projection from the unary product), it follows that this defines a category.

b. The first thing we do is to choose a bijective pairing with the additional property that it reflects. This means we choose a pair of functions $l, r : \mathbf{N} \to \mathbf{N}$ such that the function $\langle l, r\rangle : \mathbf{N} \to \mathbf{N} \times \mathbf{N}$ is bijective and such that if $l(m) \leq l(m')$ and $r(m) \leq r(m')$, then $m \leq m'$. It follows that if $l(m) = a$, then the number of $m' \leq m$ for which $l(m') = a$ is $r(m)$. A pairing that works is given by $l(m) = \exp_2(m)$, the largest integer l for which 2^l divides m, and

$$r(m) = \frac{m + 1 - 2^l}{2^{l+1}}$$

where $l = l(m)$. Let M_i be the ith Turing machine in some enumeration of them. Let M be the process that at step n runs $M_{l(n)}$ one step. Because of our assumption on the bijective pairing, this is actually the $r(n)$th step that this machine has executed. Define a recursive function $f : \mathbf{N} \to \mathbf{N}$ by letting $f(n) = r(n)$ if machine $M_{l(n)}$ halts at the nth step and 0 otherwise. This is not injective (takes the value 0 more than once) but the composite $\langle r, l\rangle^{-1} \circ \langle \mathrm{id}, f\rangle$ is and defines an injective recursive function $\mathbf{N} \to \mathbf{N}$ whose image is not decidable (else the halting problem would be). But then there cannot be any recursive function whose image is the complement. Thus the subobject defined by that injection has no complement in the category.

7. Let N denote the sum of countably many copies of 1 and $i_n : 1 \to N$ be the nth injection to the sum. Let $z = n_0 : 1 \to N$ and let $s : N \to N$ be defined by $s \circ i_n = i_{n+1}$. Given an object A, an arrow $f_0 : 1 \to A$ and an arrow $t : A \to A$, we use ordinary induction to define a sequence of arrows $f_n : 1 \to A$, for $n \geq 1$ by $f_{n+1} = t \circ f_n$. Then the arrow $f = \langle f_0, f_1, \ldots,\rangle : N \to A$ clearly satisfies the recursion. The uniqueness is shown similarly. If the countable sums are stable under pullbacks, then this construction is clearly stable.

Section 9.7

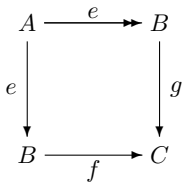
1. The definition of epimorphism implies that for any $h : B \to X$, the condition $h \circ f = h \circ g$ is equivalent to the condition $h \circ f \circ e = h \circ g \circ e$ so that the condition to be satisfied by a coequalizer of f and g is identical to that to be satisfied by a coequalizer of $f \circ e$ and $g \circ e$. Thus the cocone functors are equivalent and are therefore the universal elements.

2. If $h = \langle h_1|h_2\rangle : B_1 + B_2 \to D$, then the condition $\langle h_1|h_2\rangle \circ (f_1 + f_2) = \langle h_1|h_2\rangle \circ (g_1 + g_2)$ is equivalent to the two conditions $h_1 \circ f_1 = h_1 \circ g_1$ and $h_2 \circ f_2 = h_2 \circ g_2$. There results unique arrows $k_1 : C_1 \to D$ and $k_2 : C_2 \to D$ such that $k_1 \circ c_1 = h_1$ and $k_2 \circ c_2 = h_2$. This gives $\langle k_1|k_2\rangle \circ (c_1 + c_2) = \langle h_1|h_2\rangle$.

3. By definition of coequalizer, $e \circ d^0 \circ u_1 = e \circ d^1 \circ u_1$, so $c \circ e \circ d^0 \circ u_1 = c \circ e \circ d^1 \circ u_1$, and also $c \circ e \circ d^0 \circ u_2 = c \circ e \circ d^1 \circ u_2$, so by the universal property of sums, $c \circ e \circ d^0 = c \circ e \circ d^1$. Suppose $f \circ d^0 = f \circ d^1$. Then $f \circ d^0 \circ u_1 = f \circ d^1 \circ u_1$, so there is a unique arrow $x : C \to X$ for which $x \circ e = f$. Since $x \circ e \circ d^0 \circ u_2 = f \circ d^0 \circ u_2 = f \circ d^1 \circ u_2 = x \circ e \circ d^1 \circ u_2$, there is a unique arrow $y : D \to X$ for which $y \circ c = x$, hence $y \circ c \circ e = f$. Thus $c \circ e$ satisfies the existence part of the definition of coequalizer of d^0 and d^1. If $y' \circ c \circ e = f$, then $y \circ c = y' \circ c$ since e is a coequalizer, and so $y = y'$ because c is a coequalizer. Thus $c \circ e$ also satisfies the uniqueness requirement.

Section 9.8

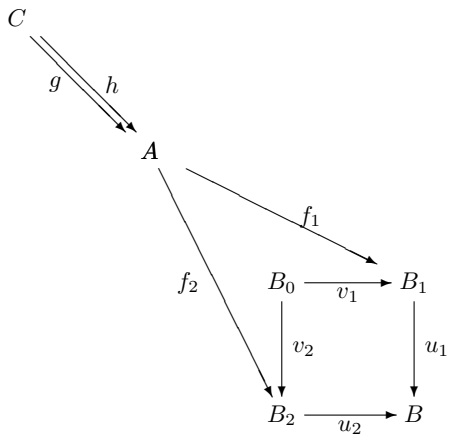
1. Suppose $e : A \to B$ in $\mathscr{E}$ is not an epimorphism. Then the cokernel pair $B \underset{g}{\overset{f}{\rightrightarrows}} C$ have a common left inverse $h : C \to B$ and $h \circ f = \mathrm{id} \in \mathscr{M}$. If f were in $\mathscr{M}$, the diagonal fill-in in



would provide a map $k : B \to B$ such that $f \circ k = l$. But then $k = h \circ f \circ k = h \circ g = \mathrm{id}$ so that $f = g$ which contradicts the assumption that e is not epic.

2. a. Suppose $\{C_i\}$ is a collection of subobjects of an object C and C_0 is an intersection. Suppose $u_i : C_i \to C$ is the inclusion and that $v_i : C_0 \to C_i$ is the element of the limit. Then I claim that each v_i is monic. If not, there is some object B and two arrows $B \underset{g}{\overset{f}{\rightrightarrows}} C_0$ with $v_i \circ f = v_i \circ g$. Then for each $j \in I$, we have $u_j \circ v_j \circ f = u_i \circ v_i \circ f = u_i \circ v_i \circ g = v_j \circ u_j \circ g$ from which the monomorphism v_j can be cancelled to give that $u_j \circ f = u_j \circ g$ for all $j \in I$ and then g and h are two arrows to C_0 that give the same cone over the diagram D whose limit is C_0, which contradicts the uniqueness of the arrow to the limit. This shows that any of the equal composites $C_0 \to C_i \to C$ is monic. If $C' \subseteq C$ is any subobject of C for which $C' \subseteq C_i$ for all i, then the universal mapping property of C_0 shows that there is an arrow $C' \to C_0$ such that $C' \to C \to C_i$ is the inclusion into C_i. If the composite of two arrows is monic, so is the first, so that $C' \to C_0$ is also monic. Thus C_0 is the meet in the poset of subobjects of C.

b. Consider a category with six objects and eight non-identity arrows as shown

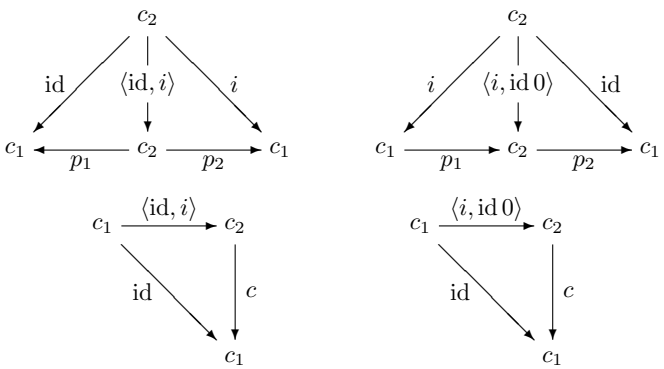


subobject to $u_1 \circ v_1 = u_2 \circ v_2$, $u_1 \circ f_1 = u_2 \circ f_2$, $f_1 \circ g = f_1 \circ h$ and $f_2 \circ g = f_2 \circ h$. Then the subobjects of B are the four element lattice consisting of B_0, B_1, B_2 and B and B_0 is the meet of B_1 and B_2. But the lack of an arrow $A \longrightarrow B_0$ implies that B_0 is not their intersection. The reason for g and h is so that A is not a subobject of B.

Solutions for Chapter 10

Section 10.1

1. Add a new unary operation $i : c_1 \longrightarrow c_1$ and the following diagrams:



2. Let (C_0, C_1, s, t, u, c) and (D_0, D_1, s, t, u, c) be two models of the sketch for categories. Note that we have used the common convention of using the same letter to stand for the arrow in the sketch and in the model (in every model, in fact). A

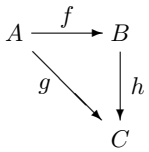
homomorphism consists of an arrow $F_0 : C_0 \to D_0$ and an arrow $F_1 : C_1 \to D_1$ that satisfy some conditions forced by the fact that a homomorphism is a natural transformation. First off, we have $s \circ F_1 = F_0 \circ s$, $t \circ F_1 = F_0 \circ t$ and $u \circ F_0 = F_1 \circ u$. These mean that the homomorphism preserves source, target and identity arrows. These identities induce a unique arrow $F_2 : C_2 \to D_2$ such that $p_1 \circ F_2 = F_1 \circ p_1$ and $p_2 \circ F_2 = F_1 \circ p_2$. We further suppose that $c \circ F_2 = F_1 \circ c$. These conditions mean that the homomorphism preserves composition; thus the homomorphism is a functor.

Section 10.2

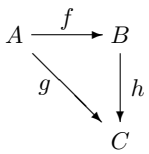
1. Referring to Diagram (10.1), we see that incl is the arrow opposite not $\circ$ $\mathsf{true} : 1 \to b$ in a cone, which means that in a model it is the arrow opposite an arrow from 1. But every arrow from 1 is monic (see Section 2.8, Exercise 7). Hence incl is monic.

Section 10.3

1. It is the set of all diagrams (not necessarily commutative) of the form



2. It is the set of all commutative diagrams of the form



3. For each object C of $\mathscr{C}$, let $M'(c) = \{(x, c) \mid x \in M(c)\}$. If $t : c \to d$ is an arrow in $\mathscr{S}$, let $t(x, c) = (t(x), d)$. Then the first projection is an isomorphism $M' \to M$ and it is clear that $M(c) \cap M(d) = \emptyset$ for $c \neq d$.

Solutions for Chapter 11

Section 11.1

1. A model of $\mathscr{E}$ is a sketch homomorphisms $\mathscr{E} \to \mathscr{C}$. This assigns to the single node of $\mathscr{E}$ an object E and that is all. If we denote the single object of $\mathscr{E}$ by w, then the functor $M \mapsto M(e)$ is evidently an isomorphism between the objects of $\mathscr{C}$ and those of $\mathbf{Mod}(\mathscr{E}, \mathscr{C})$. If $\alpha : M \to N$ is a natural transformation, the $\alpha e : M(e) \to N(e)$ is an arrow of $\mathscr{C}$ and is subject to no conditions. Thus means that $\mathbf{Nat}(M, N) = \mathrm{Hom}(M(e), N(e))$ and so the functor is an isomorphism.

Section 11.2

1. **Nat** has one such node, denoted 1, as does **Stack**. The disjoint union has two such nodes, but when the coequalizer is formed they are identified to a single node in $\mathscr{S}$. This node, and only this, must become a singleton in any model.

Section 11.3

1. $U \times U$ is induced by the sketch homomorphism from $\mathscr{E}$ to the sketch for monoids (given in 7.2.1 and 7.3.2) that takes e to $s \times s$.

2. For any $u : s_1 \to s_2$ in $\mathscr{S}$, we have $F(u) : F(s_1) \to F(s_2)$ which gives a commutative square

$$\begin{array}{ccc} M(F(s_1)) & \xrightarrow{M(F(u))} & M(F(s_2)) \\ {\scriptstyle \alpha F(s_1)}\downarrow & & \downarrow{\scriptstyle \alpha F(s_2)} \\ N(F(s_1)) & \xrightarrow[N(F(u))]{} & N(F(s_2)) \end{array}$$

which is the same as

$$\begin{array}{ccc} F^*(M)(s_1) & \xrightarrow{F^*(M)u} & F^*(M)(s_2) \\ {\scriptstyle F^*(\alpha)s_2}\downarrow & & \downarrow{\scriptstyle F^*(N)u} \\ F^*(N)(s_1) & \xrightarrow[F^*(\alpha)s_2]{} & F^*(N)(s_2) \end{array}$$

which is naturality of $F^*(M)$.

3. We have for $\mathrm{id}_{\mathscr{S}} : \mathscr{S} \to \mathscr{S}$ that $(\mathrm{id}_{\mathscr{S}})^*(M) = M \circ \mathrm{id}_{\mathscr{S}} = M$ and for $\alpha : M \to N$, $(\mathrm{id}_{\mathscr{S}})^*(\alpha) = \alpha\,\mathrm{id}_{\mathscr{S}} = \alpha$. Thus $(\mathrm{id}_{\mathscr{S}})^*$ is the identity functor. If $G : \mathcal{R} \to \mathscr{S}$ is another homomorphism of sketches, then $(F \circ G)^*(M) = M \circ F \circ G = G^*(M \circ F) = G^*(F^*(M)) = (G^* \circ F^*)(M)$ and similarly for $\alpha : M \to N$. Thus $(F \circ G)^* = G^* \circ F^*$ which shows that $\mathbf{Mod}_{\mathscr{C}}(-)$ is a contravariant functor.

4. Suppose, say, that σ is a diagram. The other two possibilities are similar. Suppose σ says that

$$s_1 \circ \cdots \circ s_n = t_1 \circ \cdots \circ t_m$$

Then σ is satisfied in $F^*(M)$ if and only if

$$F^*(M)(s_1) \circ \cdots \circ F^*(M)(s_n) = F^*(M)(t_1) \circ \cdots \circ F^*(M)(t_m)$$

which is the same as

$$M(F(s_1)) \circ \cdots \circ M(F(s_n)) = M(F(t_1)) \circ \cdots \circ M(F(t_m))$$

which is the same as the condition that M satisfy $F(\sigma)$.

Solutions for Chapter 12

Section 12.1

1. Let $\mathscr{E}_C$ be the fiber over an object C. If X is an object of $\mathscr{E}_C$, then $P(X) = C$ and $P(\mathrm{id}_X) = \mathrm{id}_C$ by definition of functor It follows that id_X is an arrow of $\mathscr{E}_C$. If $f : X \to Y$ and $g : Y \to Z$ are arrows of $\mathscr{E}_C$, then $P(f) = P(g) = \mathrm{id}_C$, so $P(g \circ f) = P(g) \circ P(f) = \mathrm{id}_C \circ \mathrm{id}_C = \mathrm{id}_C$, so $g \circ f$ is an arrow of $\mathscr{E}_C$. Hence $\mathscr{E}_C$ is a subcategory.

2. Let $P : \mathscr{E} \to \mathscr{C}$ be a fibration with cleavage γ. To see that Ff is a functor, note first that $\gamma(f, Y) \circ \mathrm{id}_Y = \mathrm{id}_Y \circ \gamma(f, Y)$ so that $Ff(\mathrm{id}_Y) = \mathrm{id}_{Ff_Y}$. Suppose $u' : Y' \to Y''$. Then

$$\gamma(f, Y'') \circ Ff(u') \circ Ff(u) = u' \circ \gamma(f, Y') \circ Ff(u) = u' \circ u \circ \gamma(f, Y)$$

so by uniqueness $Ff(u' \circ u) = Ff(u') \circ Ff(u)$.

Now we show that F is a contravariant functor. If we show that $F(\mathrm{id}_C)$ is the identity on arrows it will have to be the identity on objects because functors preserve source and target. Let C be an object of $\mathscr{C}$ and u an arrow of $F(C)$. Then $F(\mathrm{id}_C)(u)$ is the unique arrow for which $\gamma(\mathrm{id}_C, Y') \circ F(\mathrm{id}_C)(u) = u \circ \gamma(\mathrm{id}_C, Y)$. By SC–1, this requirement becomes $\mathrm{id}_{Y'} \circ F(\mathrm{id}_C)(u) = u \circ \mathrm{id}_Y$. Thus $F(\mathrm{id}_C)(u) = u$ as required.

Now suppose $f : C \to D$ and $g : D \to E$ in $\mathscr{C}$. Let $u : Y \to Y'$ in $F(C)$. Then $F(g \circ f)(u)$ must be the unique arrow for which

$$\gamma(g \circ f, Y') \circ F(g \circ f)(u) = u \circ \gamma(g \circ f, Y)$$

But

$$\begin{aligned} u \circ \gamma(g \circ f, Y) &= u \circ \gamma(g, Y) \circ \gamma(f, Ff(Y)) \\ &= \gamma(g, Y') \circ Fg(u) \circ \gamma(f, Ff(Y)) \\ &= \gamma(g, Y') \circ \gamma(f, Ff(Y')) \circ Ff(Fg(u)) \end{aligned}$$

so by CA–2 and SC–2, $F(g \circ f) = F(f) \circ F(g)$.

3. a. We will also denote the functor by ϕ. It is a fibration if for every element x of $\mathbf{Z}_2$ there is an element u of $\mathbf{Z}_4$ such that $\phi(u) = x$ (that is CA–1) and for every $v \in \mathbf{Z}_4$ and $h \in \mathbf{Z}_2$ such that $x + h = \phi(v) \pmod 2$, there is a unique $w \in \mathbf{Z}_4$ for which $\phi(w) = h$ and $u + w = v \pmod 4$ (which is CA–2). This amounts to saying that ϕ is surjective and every equation $m + y = n \pmod 4$ can be solved uniquely for y, which requires simple case checking (or knowing that $\mathbf{Z}_2$ and $\mathbf{Z}_4$ are groups). An analogous proof works for opfibration.

b. A splitting would be a monoid homomorphism (this follows immediately from SC–1 and SC–2) which would make ϕ a split epimorphism, which it is not by Exercise 2 of Section 2.9.

4. a. P preserves source and target by definition. Let $(h, k) : f \to f'$ and $(h', k') : f' \to f''$. Then

$$P((h', k') \circ (h, k)) = P(h' \circ h, k' \circ k) = k' \circ k = P(h', k') \circ P(h, k)$$

Thus P preserves composition. The identity on an object (A, B) is $(\mathrm{id}_A, \mathrm{id}_B)$, so it follows immediately that P preserves identities.

b. Let $f : C \to D$ in $\mathscr{C}$ and let $k : B \to D$ be an object of $\mathscr{A}$ lying over D. Let $\gamma(f, k)$ be the arrow (u, f) of $\mathscr{A}$ defined in 12.1.5. $P(\gamma(f, k)) = f$ so CA–1 is satisfied. As for CA–2, let $(v, v') : z \to k$ in $\mathscr{A}$ and let h be an arrow of $\mathscr{C}$ such that $f \circ h = v'$. Let w be the unique arrow given by the pullback property for which $u \circ w = v$ and $u' \circ w = h \circ z$. The last equation says that (w, h) is an arrow of $\mathscr{A}$, and $(u, f) \circ (w, h) = (u \circ w, f \circ h) = (v, v')$ as required. The uniqueness property of the pullback means that (w, h) is the only such arrow.

Section 12.2

1. The identity arrow for (x, C) is (x, id_C). Let $(x, f) : (x, C) \to (x', C')$, $(x', f') : (x', C') \to (x'', C'')$ and $(x'', f'') : (x'', C'') \to (x''', C''')$ be arrows of $\mathbf{G}_0(\mathscr{C}, F)$. Then $((x'', f'') \circ (x', f')) \circ (x, f) = (x'', f'' \circ f') \circ (x, f) = (x'', (f'' \circ f') \circ f) = (x'', f'' \circ (f' \circ f)) = (x'', f'') \circ ((x', f') \circ (x, f))$ so composition is associative.

2. Let $f : C \to C'$ in $\mathscr{C}$. Let (x, C) be an object of $\mathbf{G}_0(\mathscr{C}, F)$ lying over C. Define $\kappa(f, (x, C))$ to be $(x, f) : (x, C) \to (x', C')$ where $x' = Ff(x)$. Now suppose $(x, g) : (x, C) \to (y, C'')$ and suppose $k : C' \to C''$ has the property that $k \circ f = g$. The arrow required by OA–2 is $(x', k) : (x', C') \to (y, C'')$. This is well defined since $Fk(x') = Fk(Ff(x)) = F(k \circ f)(x) = Fg(x)$ which must be y since (y, C'') is the given target of (x, g). Moreover it satisfies OA–2 since $(x', k) \circ (x, f) = (x, k \circ f) = (x, g)$. An arrow satisfying OA–2 must lie over k and have source (x', C') and target (y, C'') so that it can compose with f to give g, so (x', k) is the only possible such arrow. To see that κ is a splitting, suppose that $(x', f') : (x', C') \to (x'', C'')$. Then $\kappa(f', (x', C')) = (x', f')$ and $(x', f') \circ (x, f) = (x, f' \circ f) = \kappa(f' \circ f, (x, C))$ as required. The verification for identities is even easier.

3. The identity arrow is $(\mathrm{id}_x, \mathrm{id}_C) : (x, C) \to (x, C)$. It is well defined since $F(\mathrm{id}_C)(x) = x$, so $\mathrm{id}_x : F(\mathrm{id}_C)(x) \to x$. Suppose that (u, f) is an arrow from (x, C) to (x', C'), so that $u : Ff(x) \to x'$. Similarly, let $(u', f') : (x', C') \to (x'', C'')$ and $(u'', f'') : (x'', C'') \to (x''', C''')$. Then

$$\begin{aligned} ((u'', f'') \circ (u', f')) \circ (u, f) &= (u'', Ff''(u'), f'' \circ f') \circ (u, f) \\ &= (u'' \circ Ff''(u') \circ F(f'' \circ f')(u), (f'' \circ f') \circ f) \end{aligned}$$

and

$$\begin{aligned} (u'', f'') \circ ((u', f') \circ (u, f)) &= (u'', f'') \circ (u' \circ Ff(u), f' \circ f) \\ &= (u'' \circ Ff''(u' \circ Ff'(u)), f'' \circ (f' \circ f)) \end{aligned}$$

The result follow from the facts that F and Ff'' are functors and composition in $\mathscr{C}$ is associative.

4. Let $f : C \to C'$ and let (x, C) lie over C. Define $\kappa(f, (x, C))$ to be

$$(\mathrm{id}_{Ff(x)}, f) : (x, C) \to (Ff(x), C')$$

Let $(u, g) : (x, C) \longrightarrow (y, C'')$ so that $u : Fg(x) \longrightarrow y$. Let $k : C' \longrightarrow C''$ satisfy $k \circ f = g$. Then $(u, k) : (Ff(x), C') \longrightarrow (y, C'')$ because the domain of u is $Fg(x) = F(k \circ f)(x) = F(k)(Ff(x))$. Furthermore,

$$(u, k) \circ (\mathrm{id}_{Ff(x)}, f) = (u \circ F(\mathrm{id}_{Ff(x)}), k \circ f) = (u, g)$$

as required by OA–2. It follows as in the answer to the second problem that (u, k) is the only possible arrow with this property.

5.

$$\begin{aligned} ((t, m)(t', m'))(t'', m'') &= (t\alpha(m, t'), mm')(t'', m'') \\ &= (t\alpha(m, t')\alpha(mm', t''), mm'm'') \end{aligned}$$

and

$$\begin{aligned} (t, m)((t', m')(t'', m'')) &= (t, m)(t, \alpha(m', t''), m'm'') \\ &= (t\alpha(m, t'\alpha(m', t'')), mm'm'') \end{aligned}$$

However, by MA–4,

$$\alpha(m, t')\alpha(mm', t'') = \alpha(m, t')\alpha(m, \alpha(m', t''))$$

and that is $\alpha(m, t'\alpha(m', t''))$ by MA—2.

6. Let β take an object x of $F(C)$ to (x, C) and an arrow u to (u, id_C). By GC–3, if $v : y \longrightarrow z$ then

$$(v, \mathrm{id}_C) \circ (u, \mathrm{id}_C) = (v \circ F(\mathrm{id}_C)(u), \mathrm{id}_C \circ \mathrm{id}_C) = (v \circ u, \mathrm{id}_C)$$

so β preserves composition. It is clearly a bijection and preserves identities.

Section 12.3

1. Suppose that $\alpha : F \longrightarrow G$ is a natural transformation. $\mathbf{G}\alpha$ preserves identities because $\mathbf{G}\alpha(\mathrm{id}_X, \mathrm{id}_C) = \alpha C(\mathrm{id}_X, \mathrm{id}_C) = (\mathrm{id}_{\alpha C(x)}, \mathrm{id}_C)$. Suppose that $(u, f) : (x, C) \longrightarrow (x', C')$ and $(u', f') : (x', C') \longrightarrow (x'', C'')$. Then

$$(u', f') \circ (u, f) = (u' \circ Ff'(u), f' \circ f)$$

by GC–3 (Section 12.2). On the other hand,

$$\begin{aligned} \mathbf{G}\alpha(u', f') \circ \mathbf{G}\alpha(u, f) &= (\alpha C''u', f') \circ (\alpha C'u, f) \\ &= (\alpha C''u' \circ Gf'(\alpha C'u), f' \circ f) \\ &= (\alpha C''u' \circ \alpha C''(Ff'(u)), f' \circ f) \\ &= (\alpha C''(u' \circ Ff'(u)), f' \circ f) \\ &= \mathbf{G}\alpha(u' \circ Ff'(u), f' \circ f) \end{aligned}$$

where the third equality uses the naturality of α and the fourth uses the fact that $\alpha C''$ is a functor. Thus $\mathbf{G}\alpha$ preserves composition.

2. GR–2 implies that $\mathbf{G}$ preserves identity natural transformations, since any component of an identity transformation is an identity arrow. Let $\alpha : F \to F'$ and $\beta : F' \to F''$ be natural transformations, where F, F' and F'' are functors from $\mathscr{C}$ to **Cat**. Then for $(u, f) : (x, C) \to (x, C')$,

$$\mathbf{G}(\beta \circ \alpha)(u, f) = ((\beta \circ \alpha)C'u, f) = (\beta C'(\alpha C'u), f)$$

and

$$(\mathbf{G}\beta(\mathbf{G}\alpha))(u, f) = \mathbf{G}\beta(\alpha C'u, f) = (\beta C'(\alpha C'u), f)$$

so $\mathbf{G}$ preserves composition.

Section 12.4

1. If $\mathscr{A}$ and $\mathscr{B}$ are monoids, then each has only one object. By WP–1, an object of $\mathscr{A}\,\mathrm{wr}^G\,\mathscr{B}$ is a pair (A, P) where A is the only object of $\mathscr{A}$ and $P : G(A) \to \mathscr{B}$ is a functor. But if $\mathscr{B}$ has only one object and $G(A)$ is discrete, there is only one functor from $G(A)$ to $\mathscr{B}$ – it must take all the objects of $G(A)$ to the only object of $\mathscr{B}$. Hence $\mathscr{A}\,\mathrm{wr}^G\,\mathscr{B}$ has only one object, so is a monoid.

2. Let $\mathscr{A}$ and $\mathscr{B}$ be groups. Since they have only one object each, we can simplify the notation in WP–1 through WP–3 and omit mention of the objects A, A' and A''. The value of G at the only object of $\mathscr{A}$ is a category we will call $\mathscr{G}$. If f is an element of the group $\mathscr{A}$, then Gf is an automorphism of the category $\mathscr{G}$. An object of $\mathscr{A}\,\mathrm{wr}^G\,\mathscr{B}$ is a functor $P : \mathscr{G} \to \mathscr{B}$. An arrow $(f, \lambda) : P \to P'$ consists of an element f of the group $\mathscr{A}$ and a natural transformation $\lambda : P \to P' \circ Gf$. Each component of λ is an element of the group $\mathscr{B}$ so has an inverse; thus λ is an invertible natural transformation. Since f is also a group element, it has an inverse f^{-1}. Let μ be the natural transformation whose component at an object X of $\mathscr{G}$ is $(\lambda(Gf)^{-1}(X))^{-1}$. Then the inverse of the arrow (f, λ) is (F^{-1}, μ). To verify this, we calculate

$$(F^{-1}, \mu) \circ (f, \lambda) = (f^{-1} \circ f, \mu Gf \circ \lambda)$$

Now $F^{-1} \circ f$ is the identity of $\mathscr{A}$ and for an object X of $\mathscr{G}$,

$$\begin{aligned}(\mu Gf \circ \lambda)X &= \mu(Gf(X)) \circ \lambda X \\ &= \lambda((Gf)^{-1}(Gf(X)))^{-1} \circ \lambda X \\ &= (\lambda X)^{-1} \circ \lambda X = \mathrm{id}_X\end{aligned}$$

so (f, λ) is invertible.

Solutions for Chapter 13

Section 13.1

1. By definition, g takes the identity of $F(X)$ to that of M. If $a = (x_1, \ldots, x_n)$ and $b = (y_1, \ldots, y_m)$, then $ab = (x_1, \ldots, x_n, y_1, \ldots, y_m)$ and

$$\begin{aligned} g(ab) &= g(x_1, \ldots, x_n, y_1, \ldots, y_m) \\ &= u(x_1) \cdot \cdots \cdot u(x_n) \cdot u(y_1) \cdot \cdots \cdot u(y_m) \\ &= g(x_1, \ldots, x_n) \cdot g(y_1, \ldots, y_m) \\ &= g(a) \cdot g(b) \end{aligned}$$

Section 13.2

1. If $S_0 \subseteq f^{-1}(T_0)$, let $y \in f(S_0)$. Then there is some $x \in S_0$ such that $f(x) = y$ and since $x \in S_0 \subseteq f^{-1}(T_0)$, $x \in f^{-1}(T_0)$ so that $y = f(x) \in T_0$. Therefore $f_*(S_0) \subseteq T_0$. Conversely, suppose $f_*(S_0) \subseteq T_0$. For $x \in S_0$, $f(x) \in F(S_0)$ so that $f(x) \in T_0$, whence $x \in f^{-1}(T_0)$. Therefore $S_0 \subseteq f^{-1}(T_0)$.

2. Let $\Sigma \dashv \Delta$. Then for every object C,

$$\begin{aligned} \mathrm{Hom}_{\mathscr{C}}(\Sigma(A,B), C) &\cong \mathrm{Hom}_{\mathscr{C}\times\mathscr{C}}((A,B), \Delta(C)) \\ &\cong \mathrm{Hom}_{\mathscr{C}\times\mathscr{C}}((A,B), (C,C)) \\ &\cong \mathrm{Hom}_{\mathscr{C}}(A,C) \times \mathrm{Hom}_{\mathscr{C}}(B,C) \end{aligned}$$

which is the mapping condition that determines $A + B$ uniquely.

3. The unit of the adjunction is the map from $\Delta\Pi(A,B) \to (A,B)$ that corresponds under the adjunction to the identity arrow $\Pi(A,B) \to \Pi(A,B)$. Since $\Pi(A,B) = A \times B$, we are looking for an arrow $(A \times B, A \times B) \to (A,B)$. The arrow is $(\mathrm{proj}_1, \mathrm{proj}_2)$ since that is the pair that gives the universal mapping property.

4. If n is an integer, let us (temporarily) denote by $\mathrm{coe}(n)$ (for coercion), the real number n. If there is a left adjoint, say ci, then it is characterized by the property that for all real $n \in \mathbf{N}$ and $r \in \mathbf{R}$, $\mathrm{ci}(r) \leq n$ if and only if $r \leq \mathrm{coe}(n)$, the property that obviously characterizes the ceiling function. In a similar way the right adjoint fl is characterized by the property that $n \leq \mathrm{fl}(r)$ if and only if $\mathrm{coe}(n) \leq r$, which is the defining property of the floor function.

5. There is a sketch for monoids gotten by augmenting that of semigroups given in Section 7.2 by adding a constant of type s and diagrams that force it to be a left and right identity. Call this sketch $\mathscr{M}$. For a set X, let $\mathscr{M}(X)$ denote the sketch gotten from $\mathscr{M}$ by adding to $\mathscr{M}$ the set X of constants. Then the initial algebra for $\mathscr{M}(X)$ is just the free monoid $F(X)$. The reason is that a model of $\mathscr{M}(X)$ is just a model M of $\mathscr{M}$ together with a chosen element $u(x) \in M$ for each $x \in X$, which is just a function $u : X \to U(M)$. Thus an initial model of the sketch is the free monoid generated by X. In 4.7.17, exactly the same construction of adding a set of constants to a sketch and forming the initial model yields the free model on the original sketch generated by that set.

6. This is the content of Proposition 3.1.15.

7. Exercise 10 of Section 3.1 says that the discrete category has the unique map lifting property, so the functor that takes a set to the discrete category with that set of objects is the left adjoint. Similarly by Exercise 11 of Section 3.1, the construction of the indiscrete category gives the right adjoint.

8. Given a set S, the value of the left adjoint is the category with object set $S \times \{s, t\}$ and arrow set $S \times \{1, 2, 3\}$. If $f \in S$ then $(f, 1) : (f, s) \to (f, t)$, $(f, 2) : (f, s) \to (f, s)$ and $(f, 3) : (f, t) \to (f, t)$, with $(f, 2)$ and $(f, 3)$ acting as identity arrows. There are no other compositions.

There is no right adjoint by Theorem 13.3.7, since by Exercise 6 of Section 9.5, the underlying arrow functor does not preserve coequalizers.

9. If the functor defined in 13.2.5 has a left adjoint F, then by definition of left adjoint there is for any set X an arrow $\eta X : X \to FX \times A$ with the property that for any function $f : X \to Y \times A$ there is a unique function $g : FX \to Y$ for which $(g \times A) \circ \eta X = f$. Now take $Y = 1$, the terminal object (any one element set). There is only one function $g : FX \to 1$, so there can be only one function $f : X \to 1 \times A \cong A$. If A has more than one element and X is non-empty, this is a contradiction.

10. Since it is the arrow that corresponds under adjunction to the identity, its value at B is the unique arrow $\eta B : B \to R_A(B \times A)$ such that the composite

$$B \times A \xrightarrow{\eta B \times \mathrm{id}_A} R_A(B \times A) \times A \xrightarrow{e} B \times A$$

is the identity. In **Set**, $R_A(B \times A)$ is the set of functions from A to $A \times B$, and ηB takes $b \in B$ to the function $a \mapsto (b, a)$.

Section 13.3

1. We use Theorem 13.3.2. We have

$$\mathrm{Hom}_{\mathscr{A}}(LTA, A') \cong \mathrm{Hom}_{\mathscr{B}}(TA, TA') \cong \mathrm{Hom}_{\mathscr{A}}(A, RTA')$$

and

$$\mathrm{Hom}_{\mathscr{B}}(TLB, B') \cong \mathrm{Hom}_{\mathscr{A}}(LB, RB') \cong \mathrm{Hom}_{\mathscr{B}}(B, TRB')$$

2. a. A limit of the inclusion $X \subseteq R$ is a real number r such that $r \leq x$ for all $x \in X$ (that is the cone with vertex r) and that if $r' \leq x$ for all $x \in X$, then $r' \leq r$. This is precisely the definition of the infimum. The second half is dual.

b. Since the floor function is a right adjoint, it preserves all limits that exist, in particular, all infimums and dually the ceiling function preserves colimits, which includes supremums. On the other hand, if we take, say a sequence that converges downwards on an integer, say the sequence $X = 1, 1/2, \ldots, 1/n, \ldots$ the infimum of the ceilings is 1, while the infimum is 0, whose ceiling is 0. Similarly, floor does not necessarily preserve supremum.

3. This translates to showing for any $g : A' \to A$ and $h : B \to B'$, that

$$\begin{array}{ccc} \mathrm{Hom}(FA, B) & \xrightarrow{\beta(A,B)} & \mathrm{Hom}(A, UB) \\ \Big\downarrow{\scriptstyle \mathrm{Hom}(Fg,h)} & & \Big\downarrow{\scriptstyle \mathrm{Hom}(g,Uh)} \\ \mathrm{Hom}(FA', B') & \xrightarrow[\beta(A',B')]{} & \mathrm{Hom}(A', UB') \end{array}$$

Applied to an $f \in \mathrm{Hom}(FA, B)$, this requires that we show that

$$U(h \circ f \circ Fg) \circ \eta A' = Uh \circ (Uf \circ \eta A) \circ g$$

But using the functoriality of U and naturality of η, we have

$$U(h \circ f\ ) \circ \eta A' = Uh \circ Uf \circ UFg \circ \eta A' = Uh \circ Uf \circ \eta A \circ g$$

4. The adjointness $F \dashv U$, for $F : \mathscr{C} \to \mathscr{D}$ and $U : \mathscr{D} \to \mathscr{C}$ is equivalent to the natural isomorphism

$$\mathrm{Hom}_{\mathscr{C}}(-, U-) \cong \mathrm{Hom}_{\mathscr{D}}(F-, -) : \mathscr{C}^{\mathrm{op}} \times \mathscr{D} \to \mathbf{Set}$$

But that is the same as the natural isomorphism

$$\mathrm{Hom}_{\mathscr{C}^{\mathrm{op}}}(U^{\mathrm{op}}-, -) \cong \mathrm{Hom}_{\mathscr{D}^{\mathrm{op}}}(-, F^{\mathrm{op}}-) : \mathscr{D} \times \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$$

and since $(\mathscr{D}^{\mathrm{op}})^{\mathrm{op}} = \mathscr{D}$, the conclusion follows.

5. a. Here and below, we use Hom for $\mathrm{Hom}_{\mathscr{C}}$ and Hom_A for $\mathrm{Hom}_{\mathscr{C}/A}$. If $f : B \to A$ is an object of $\mathscr{C}/A$ and C is an object of $\mathscr{C}$, an arrow $g : f \to P_A(C) = p_2 : C \times A \to A$ has two coordinates, say $g_1 = p_1 \circ g : B \to C$ and $g_2 = p_2 \circ g : B \to A$. It is an arrow in $\mathscr{C}/A$ if and only if $g_2 = f$. Thus there are no conditions on g_1, so $\mathrm{Hom}_A(f, P_A(C)) \cong \mathrm{Hom}(B, C) = \mathrm{Hom}(L_A(f), C)$.

b. This is an application of Exercise 1.

c.

$$\begin{aligned} &\mathrm{Hom}(B, \Phi(P_A(C))) \cong \mathrm{Hom}(B, [A \to C]) \cong \mathrm{Hom}(B \times A, C) \\ &\cong \mathrm{Hom}_A(B \times A \to A, C \times A \to A) \cong \mathrm{Hom}_A(P_A(B), P_A(C)) \end{aligned}$$

d. The hint suggests that we find a natural transformation from $\mathrm{Hom}(-, \Phi(C))$ to $\mathrm{Hom}(-, \Phi(D))$. It is simple to prove that the isomorphism of part (c) is natural in B. Thus we have for any B, the arrow

$$\begin{aligned} &\mathrm{Hom}(B, \Phi(P_A(C))) \cong \mathrm{Hom}(P_A(B), P_A(C)) \\ &\xrightarrow{\mathrm{Hom}(P_A(B), f)} \mathrm{Hom}(P_A(B), P_A(D)) \cong \mathrm{Hom}(B, \Phi(P_A(D))) \end{aligned}$$

The fact that these isomorphisms are natural in B implies the existence of the required arrow $\Phi(f)$. The desired commutation is essentially the definition. This part justifies the use of the notation $\Phi(P_A(C))$ for it implies that if $P_A(C) \cong P_A(D)$, then $\Phi(P_A(C)) \cong \Phi(P_A(D))$.

e. The required diagram is given by letting $d = \langle \mathrm{id}_C, f\rangle : C \to C \times A$, $d^0 = \langle \mathrm{proj}_1, \mathrm{proj}_2, \mathrm{proj}_2\rangle$ and $d^1 = \langle \mathrm{proj}_1, f \circ \mathrm{proj}_1, \mathrm{proj}_2\rangle : C \times A \to C \times A \times A$. These are readily seen to be arrows in $\mathscr{C}/A$. The equalizer of d^0 and d^1 is

$$\begin{aligned}&\{(c,a) \in C \times A \mid (c,a,a) = (c, f(c), a)\}\\ &= \{(c,a) \in C \times A \mid a = f(c)\} = \{(c, f(c)) \mid c \in C\}\end{aligned}$$

which is the image of d.

f. In the diagram

$$\begin{array}{ccccc}
\mathrm{Hom}(B, \Phi(f)) & \longrightarrow & \mathrm{Hom}(B, \Phi(P_A(C))) & \rightrightarrows & \mathrm{Hom}(B, \Phi(P_A L_A P_A(C)))\\
\downarrow & & \downarrow \cong & & \downarrow \cong\\
\mathrm{Hom}_A(P_A(B), f) & \longrightarrow & \mathrm{Hom}_A(P_A(B), P_A(C)) & \rightrightarrows & \mathrm{Hom}_A(P_A(B), P_A L_A P_A(C))
\end{array}$$

both lines are equalizers since both $\mathrm{Hom}(B, -)$ and $\mathrm{Hom}_A(P_A(B), -)$ preserve equalizers. Moreover the middle and right hand vertical arrows are isomorphisms by part (c) and by definition. Hence by uniqueness of equalizers, the left hand vertical arrow is an isomorphism.

g. This is an immediate consequence of the Pointwise Adjointness Theorem 13.3.5.

Solutions for Chapter 14

Section 14.1

1. In order that h be a homomorphism of algebras, it is required that the diagram

$$\begin{array}{ccc}
1 + \mathbf{N} & \xrightarrow{\langle 0; s\rangle} & \mathbf{N}\\
{\scriptstyle 1+h}\downarrow & & \downarrow{\scriptstyle h}\\
1 + S & \xrightarrow[f]{} & S
\end{array}$$

commute. Applied to the element $* \in 1$, this means that $h(0) = f(*)$ and applied to $h(n)$ that $f(h(n)) = h(\mathrm{succ}(n))$. This shows the uniqueness of h and since this is the only condition to be satisfied, it shows that h works.

2. The category associated with S has the elements of S as objects and pairs (x, y) as the unique arrow $y \to x$ when $y \leq x$. Then f takes elements to elements, that is objects to objects and we define f on arrows by $f(x, y) = (f(x), f(y))$ which is well defined since $y \leq x$ implies that $f(y) \leq f(x)$. Since (x, x) is the identity and $f(x, x) = (f(x), f(x))$, we see that f preserves identities. Since $(x, y) \circ (y, z) = (x, z)$ is the composite, it is immediate that f preserves composition as well.

Now an object of $(f : S)$ is an x together with an arrow $f(x) \to x$. Such an arrow exists if and only if $f(x) \leq x$. So the objects of $(f : S)$ can be identified as those elements. If also $f(y) \leq y$, then $y \leq x$ implies that $f(y) \leq f(x)$ and that

$$\begin{array}{ccc} f(y) & \longrightarrow & y \\ \downarrow & & \downarrow \\ f(x) & \longrightarrow & x \end{array}$$

commutes. Thus $(f : S)$ can be identified as the set of all x with $f(x) \leq x$ with the restricted order. If x_0 is initial in $(f : S)$, then $f(y) \leq y$ implies that $x_0 \leq y$. But then $f(x_0) \leq x_0$ implies $f(f(x_0)) \leq f(x_0)$ so that $y = f(x_0)$ is such an element and so $x_0 \leq f(x_0)$. Thus x_0 is fixed and it is clearly the least fixed point.

3. a. Since $S \neq \emptyset$, let s_0 be an arbitrary element of S. Define

$$g(t) = \begin{cases} s & \text{if } f(s) = t \\ s_0 & \text{otherwise} \end{cases}$$

b. Choose $g : T \to S$ as above. From $F(g) \circ F(f) = \mathrm{id}_{F(S)}$ it follows easily that $F(f)$ is monic.

c. If $S = \emptyset$, there is nothing to prove since in that case the inclusions $S_0 \to S$ and $S_1 \to S$ are isomorphisms. Otherwise let $S_2 = S_0 \cup S_1$ unless $S_0 = S_1 = \emptyset$ and in that case let S_2 be any singleton set. Then if $i_2 : S_2 \to S$ is inclusion, $R(i_2)$ is monic. Hence from $i_2 \circ j_0 = i_0$, $i_2 \circ j_1 = i_1$, $Ri_0(x_0) = Ri_1(x_1)$ we see that $Ri_2 \circ Rj_0(x_0) = Ri_2 \circ Rj_1(x_1)$ from which Ri_2 can be canceled to give $Rj_0(x_0) = Rj_1(x_1)$.

Section 14.2

1. In effect we claim that if A^* is the set of lists of elements of A (including the empty list ()), then $\mathrm{rec}(A, B) = A^* \times B$. Let $\mathrm{cons} : A \times A^* \to A^*$ denote the function that adjoins an element of A to the head of a list. Then $r_0(A, B) : B \to A^* \times B$ takes $b \in B$ to $(\langle\rangle, b)$ and $r(A, B) : A \times A^* \to B$ is just $\mathrm{cons} \times \mathrm{id}_B$. Now let $t_0 : B \to X$ and $t : A \times X \to X$ be given. In order that $f : A^* \times B \to X$ make the diagram from 14.2.5 commute, it is necessary that $f(\langle\rangle, b) = t_0(b)$ and that $f(\mathrm{cons}(a, l), b) = t(a, f(l, b))$ for $a \in A$, $l \in A^*$ and $b \in B$. But those conditions define, by induction on the length of a word in A^*, a unique function $f : A^* \times B \to X$ that makes the necessary diagram commute.

2. From the definition of recursive in 14.2.5, we see that when $A = B = 1$, we get $r_0(1, 1) : 1 \to \mathrm{rec}(1, 1)$ and $r(1, 1) : \mathrm{rec}(1, 1) \to \mathrm{rec}(1, 1)$ (taking $\mathrm{rec}(1, 1)$ as the product $1 \times \mathrm{rec}(1, 1)$) and the diagram of that section describes exactly the universal mapping property of a natural numbers object.

Section 14.3

1. Since T must be a functor, it is a monotone function so that for any x and y with $x \leq y$, $T(x) \leq T(y)$. Then the existence of the natural transformations $\eta : \mathrm{id} \to T$ and $\mu : T^2 \to T$ imply that for any x, $x \leq T(x)$ and $T(T(x)) \leq T(x)$. Since $x \leq T(x)$, it follows from these inequalities and monotonicity that $T(x) \leq T(T(x)) \leq T(x)$, so that $T(x) = T(T(x))$.

2. We freely use the Godement rules of Section 4.4 together with the identities satisfied by an adjoint pair given in Section 13.2 to compute

$$\mu \circ T\mu = U\epsilon F \circ UF\eta = U(\epsilon F \circ F\eta) = U(\mathrm{id}_F) = \mathrm{id}_{UF} = \mathrm{id}_T$$

$$\mu \circ \eta T = U\epsilon F \circ \eta UF = (U\epsilon \circ \eta U)F = \mathrm{id}_U F = \mathrm{id}_{UF} = \mathrm{id}_T$$

Now the naturality of ϵ implies that for any $\alpha : R \to S$,

$$\alpha \circ \epsilon R = \epsilon S \circ FU\alpha$$

Applied when $R = FU$, $S = I$ (the identity functor) and $\alpha = \epsilon$, this gives that $\epsilon \circ FU\epsilon = \epsilon \circ \epsilon FU$. We then compute

$$\begin{aligned} \mu \circ \mu T &= U\epsilon F \circ U\epsilon FUF = U(\epsilon \circ \epsilon FU)F = U(\epsilon \circ FU\epsilon)F \\ &= U\epsilon F \circ UFU\epsilon F = \mu \circ T\mu \end{aligned}$$

3. If we can identify the η and μ with those determined by the triple, then the conclusion follows from the preceding exercise. Let us temporarily call the natural transformations determined by the triple $\widehat{\eta}$ and $\widehat{\mu}$. Then $\widehat{\eta}A$ is defined as the unique $A \to A^*$ such that the extension to a monoid homomorphism $F(A) \to F(A)$ is the identity. But $\eta(a) = (a)$ clearly extends to the identity on $F(A)$ since the extension takes $(a_1, \ldots, a_n)$ to $(a_1) \cdots (a_n) = (a_1, \ldots, a_n)$ by definition. Hence $\widehat{\eta} = \eta$.

As for $\widehat{\mu} = U\epsilon F$, that is the function underlying the unique homomorphism from $FUF(A)$ to $F(A)$ that is gotten by extending the identity function from $UF(A)$ to $UF(A)$. This function takes $(a_1, \ldots, a_n)$ to itself and the extension to a homomorphism takes, for example, the two letter string

$$((a_1, \ldots, a_n), (a'_1, \ldots, a'_{n'})) \in FUF(A)$$

to the product (concatenate)

$$(a_1, \ldots, a_n, a'_1, \ldots, a'_{n'}) \in FU(A)$$

Here we see explicitly that μ and $\widehat{\mu}$ agree on two letter strings and it is clear they agree on strings of arbitrary length.

Section 14.4

1. Let us write $f : A \Rightarrow B$ for an arrow in the Kleisli category and $f \bullet g$ for the Kleisli composition. Then for $f : A \Rightarrow B$, we have $f \bullet \eta A = \mu B \circ Tf \circ \eta A = \mu B \circ \eta TB \circ f = \mathrm{id}_B \circ f = f \circ$ For $g : C \Rightarrow A$, we have $\eta A \bullet g = \mu A \circ T\eta A \circ g = \mathrm{id}_A \circ g = g$. In this exercise and many of the later ones we use naturality without comment. Here, for example, the fact that $Tf \circ \eta A = \eta TB \circ f$ is a consequence of the naturality of η.

2. Using the same notation as in the previous problem, we have, for $f : A \Rightarrow B$, $g : B \Rightarrow C$ and $h : C \Rightarrow D$ that $(h \bullet g) \bullet f = \mu D \circ T(\mu C \circ Th \circ g) \circ f = \mu D \circ T\mu D \circ T^2 h \circ Tg \circ f = \mu D \circ \mu TD \circ T^2 h \circ Tg \circ f = \mu D \circ Th \circ \mu C \circ Tg \circ f = h \bullet (g \bullet f)$.

3. a. Let us take the last point first. Given a monoid structure $\cdot$ and 1 on A, define $\alpha : A^* \to A$ by $\alpha() = 1$, $\alpha(a) = a$ and $\alpha(a_1, a_2, \cdots, a_n) = a_1 \cdot a_2 \cdot \cdots \cdot a_n$. Because of the associativity, it is not necessary to parenthesize that expression. Since $\alpha(a) = a$, the identity $\alpha \circ \eta A = \mathrm{id}_A$ is satisfied. As for the identity $\alpha \circ T\alpha = \alpha \circ \mu A$, let us do this for a list of length two of A^*; the general case follows by an easy induction. So let $l = (a_1, a_2, \cdots, a_n)$ and $l' = (a'_1, a'_2, \cdots a'_{n'})$. If $n = 0$, we have

$$\begin{aligned} \alpha \circ T\alpha(()l') &= \alpha(1, a'_1 \cdot a'_2 \cdot \cdots \cdot a'_{n'}) \\ &= 1 \cdot a'_1 \cdot a'_2 \cdot \cdots \cdot a'_{n'} \\ &= \alpha(l') = \alpha \circ \mu A(()l') \end{aligned}$$

and similarly if $n' = 0$. For $n > 0$ and $n' > 0$,

$$\begin{aligned} \alpha \circ T\alpha(ll') &= \alpha(a_1 \cdot a_2 \cdot \cdots \cdot a_n, a'_1 \cdot a'_2 \cdot \cdots \cdot a'_{n'}) \\ &= a_1 \cdot a_2 \cdot \cdots \cdot a_n \cdot a'_1 \cdot a'_2 \cdot \cdots \cdot a'_{n'} \\ &= \alpha(ll') = \alpha \circ \mu A(ll') \end{aligned}$$

This shows that each monoid gives an algebra structure. An algebra structure on A assigns to each list $(a_1 a_2 \cdots a_n) \in A^*$ an element $\alpha(a_1 a_2 \cdots a_n) \in A$. In particular, there is a multiplication given by $a \cdot b = \alpha(ab)$. Also let 1 denote $\alpha()$. We must show that $\cdot$ and 1 constitute a monoid structure and that the associated algebra structure is the one we started with. It is already evident that if α is the algebra structure just constructed, then this monoid structure is the original one. We have $\alpha \circ T\alpha(()(a)) = \alpha(1a) = 1 \cdot a$ and $\alpha \circ \mu A(()(a)) = \alpha(a) = a$ so that $1 \cdot a = a$ and similarly $a \cdot 1 = a$. Next $\alpha \circ T\alpha((a), (b, c)) = a \cdot (b \cdot c)$ while $\alpha \circ \mu A((a), (b, c)) = \alpha(abc)$. Similarly, using $((a, b), (c))$ we can show that $(a \cdot b) \cdot c = \alpha(abc)$ and so $\cdot$ is associative. The proof that $a \cdot b \cdot c = \alpha(abc)$ extends by an obvious induction to show that $\alpha(a_1, a_2, \cdots, a_n) = a_1 \cdot a_2 \cdot \cdots \cdot a_n$, which means that the monoid structure determines uniquely the algebra structure.

b. Let (A, α) and (B, β) be algebra structures with corresponding monoid structures that we denote $\cdot$. Let $f : A \to B$. We show that f is a homomorphism of monoids if and only if it is a homomorphism of algebra structures. Suppose f is a homomorphism of algebra structures. Then from $\beta \circ Tf() = \beta() = 1$ and $f \circ \alpha() = f(1)$, we conclude that $f(1) = 1$. From $\beta \circ Tf(a, a') = \beta(f(a)f(a')) = f(a) \cdot f(a')$ and $f \circ \alpha(a, a') = f(a \cdot a')$ we see that f is a monoid homomorphism.

If f is a monoid homomorphism, then

$$\beta \circ Tf() = \beta() = 1 = f(1) = f \circ \alpha()$$

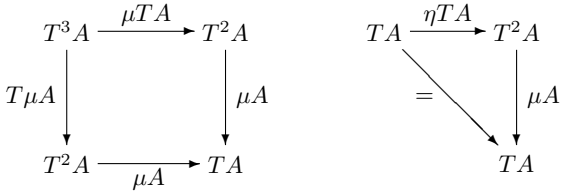
and

$$\beta \circ Tf(a) = \beta(f(a)) = f(\alpha(a)) = f \circ \alpha(a)$$

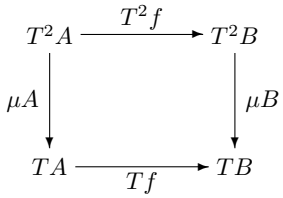
If $l = (a_1, a_2, \cdots, a_n)$, then

$$\begin{aligned} \beta \circ Tf(a_1 a_2 \cdots a_l) &= \beta(f(a_1)f(a_2)\cdots f(a_n)) \\ &= f(a_1) \cdot f(a_2) \cdot \cdots \cdot f(a_n) \\ &= f(a_1 \cdot a_2 \cdot \cdots \cdot a_n) \\ &= f \circ \alpha(a_1, a_2, \cdots, a_n) \end{aligned}$$

4. The first thing to be checked is that $(TA, \mu A)$ is actually a $\mathbf{T}$-algebra. The relevant diagrams that have to be checked to be commutative are



which are two of the three commutative diagrams required for a triple. Next we have to show that for $f : A \to B$, $Ff : FA \to FB$ is an homomorphism in the category of $\mathbf{T}$-algebras. To do this we must show that the diagram



commutes. But the commutation of this diagram for all f is exactly what is meant by the statement that μ is a natural transformation. The fact that F is a functor is left to the end.

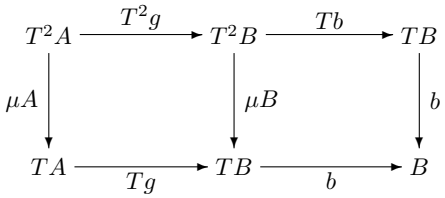
Next we use the result of Theorem 13.3.5 to show that F is left adjoint to U. To do so we must give an isomorphism

$$\mathrm{Hom}(FA, (B, b)) \to \mathrm{Hom}(A, B)$$

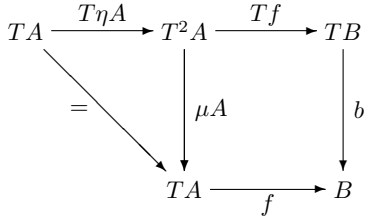
which is natural in B. The function associates to each arrow

$$f : (TA, \mu A) \to (B, b)$$

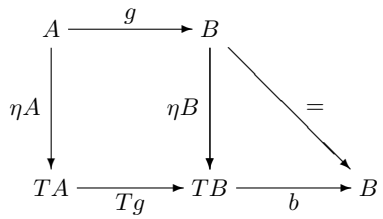
the arrow $f \circ \eta A : A \to B$. The naturality of this function follows from that of η. It must be shown to be an isomorphism. To do this, we define a function in the other direction that sends $g : A \to B$ to $b \circ Tg : TA \to B$. First we claim that that arrow is an arrow of $\mathbf{T}$-algebras. To see that, we must show that the diagram



commutes. But the left hand square does by the naturality of μ and the right hand one does because the commutativity of that square is one of the hypotheses on b. Next we observe that if $f : TA \to B$ is an algebra homomorphism, then the square in the diagram



commutes and the triangle does by one of the identities that define a triple. Thus the whole square commutes which shows that one of the composites is the identity. As for the other, that follows from the commutativity of the diagram



whose square commutes by naturality of η and triangle by one of the hypotheses on b.

Finally we show that F and U are functors, that $F \dashv U$ and that the triple gotten from the adjoint pair is $\mathbf{T}$. For an object A, $U\eta A = \mu A \circ T\eta A = \mathrm{id}_{TA} = \mathrm{id}_{UA}$ so U preserves identities. If $f : A \Rightarrow B$ and $g : B \Rightarrow C$ are arrows, then

$$\begin{aligned} Ug \circ Uf &= \mu C \circ Tg \circ \mu B \circ Tf = \mu C \circ \mu TC \circ T^2 g \circ Tf \\ &= \mu C \circ T\mu C \circ T^2 g \circ Tf = \mu C \circ T(\mu C \circ Tg \circ f) \\ &= \mu C \circ T(g \bullet f) = U(g \bullet f) \end{aligned}$$

so that U preserves composition. Thus U is a functor.

As for F, $F\,\mathrm{id}_A = \eta A \circ \mathrm{id}_A = \eta_A$ so F preserves identities. If $f : A \to B$ and $g : B \to C$, then

$$\begin{aligned} Fg \bullet Ff &= \mu C \circ TFg \circ Ff = \mu C \circ T(\eta C \circ g) \circ \eta B \circ f \\ &= \mu C \circ T\eta C \circ Tg \circ \eta B \circ f = \eta C \circ g \circ f = F(g \circ f) \end{aligned}$$

and so F is also a functor.

It is evident that $T = UF$. Also, $\eta A : A \to UFA = TA$ is natural and we let $\epsilon A = \mathrm{id}_{TA} : UFA \Rightarrow A$ in $\mathscr{K}(\mathbf{T})$. Then

$$\epsilon FA \bullet F\eta A = \mu A \circ T(\mathrm{id}_{TA}) \circ \eta TA \circ \eta A = \mu A \circ \eta TA \circ \eta A$$

which is the identity of FA. For the other adjointness, we have for $B = FA$ in $\mathscr{K}(\mathbf{T})$,

$$U\epsilon FA \circ \eta UFA = \mu A \circ T(\mathrm{id}_A) \circ \eta TA = \mu A \circ \eta TA = \mathrm{id}_{TA}$$

This completes the proof.

Section 14.5

1. If A and B are ω-CPOs, let $[A \to B]$ denote the set of monotone functions from A to B that preserve joins of countable chains. Make it a poset by the pointwise ordering, that is $f \leq g$ if $f(a) \leq g(a)$ for all $a \in A$. We claim it is an ω-CPO. In fact, if $f_0 \leq f_1 \leq \cdots$ is a countable increasing chain of such functions, then for each $a \in A$, let $f(a) = \bigvee f_i(a)$. Let us show that f is countably chain complete (it will obviously be the join in that case). If $a_0 \leq a_1 \leq \cdots$ is a countable increasing sequence with join a we have a double sequence in which every row and every column except perhaps the bottom row consists of a countable increasing sequence and its join.

$$\begin{array}{ccccc} f_0(a_0) & f_0(a_1) & f_0(a_2) & \cdots & f_0(a) \\ f_1(a_0) & f_1(a_1) & f_1(a_2) & \cdots & f_1(a) \\ f_2(a_0) & f_2(a_1) & f_2(a_2) & \cdots & f_2(a) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(a_0) & f(a_1) & f(a_2) & \cdots & f(a) \end{array}$$

For each i, $f(a_i) = \bigvee_j f_j(a_i) \leq \bigvee_j f_j(a) = f(a)$ so $f(a)$ is an upper bound for the bottom row. If $b \in B$ were another bound, we would have for each i, $f_i(a) = \bigvee_j f_i(a_j) \leq \bigvee_j f(a_j) \leq b$ so that $f(a) \leq b$. Thus $f(a) = \bigvee f(a_i)$.

Now we must show that if C is any other ω-CPO,

$$\mathrm{Hom}(C, [A \to B]) \cong \mathrm{Hom}(C \times A, B)$$

and this isomorphism is natural in B. If $f : C \times A \to B$ is a function, let $\phi(f) : C \to [A \to B]$ be the function defined by $\phi(f)(c)(a) = f(c, a)$. So far this is just the cartesian closed structure on sets. Now we claim that f is countably chain complete if and only if for all $c \in C$, $\phi(f)(c)$ is countably chain complete *and* $\phi(f)$ is countably chain complete.

Suppose f is countably chain complete. This means that if $c = \bigvee c_i$ and $a = \bigvee a_i$ are sups along increasing chains, then $f(c, a) = \bigvee f(c_i, a_i)$. This can be specialized to the case that all $c_i = c$ to conclude that $f(c, a) = \bigvee f(c, a_i)$ so that $\phi(f)(c)$ is countably chain complete. Similarly, if $c = \bigvee c_i$ is the sup of a countable increasing chain, then the fact that $f(c, a) = \bigvee f(c_i, a)$ implies that $\phi(f)(c) = \bigvee \phi(f)(c_i)$. This shows one direction. By reversing all these implications, one easily shows that if $\phi(f)(c)$ preserves joins along countable chains and $\phi(f)$ also preserves them, then f preserves joins in its variables *separately*. To prove that it preserves them jointly, let $c = \bigvee c_i$ and $a = \bigvee a_i$ and consider the double sequence

$$\begin{array}{ccccc} f(c_0, a_0) & f(c_0, a_1) & f(c_0, a_2) & \cdots & f(c_0, a) \\ f(c_1, a_0) & f(c_1, a_1) & f(c_1, a_2) & \cdots & f(c_1, a) \\ f(c_2, a_0) & f(c_2, a_1) & f(c_2, a_2) & \cdots & f(c_2, a) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(c, a_0) & f(c, a_1) & f(c, a_2) & \cdots & f(c, a) \end{array}$$

in which all rows and all columns are joins of increasing sequences. Clearly, $f(c_i, a_i) \leq f(c, a)$ for all i and if b is an upper bound for the $f(c_i, a_i)$, then for all $j \geq i$, $f(c_i, a_j) \leq f(c_j, a_j) \leq b$, whence $f(c_i, a) = \bigvee_j (c_i, a_j) \leq b$. Since this is true for all i, it follows that $f(c, a) \leq b$. Hence f preserves joins of countable increasing chains.

2. This is clearly a poset. If $c^0 \leq c^1 \leq c^2 \leq \cdots$ is an increasing chain in $\widehat{P}$, let c^n be the chain $c^n_0 \leq c^n_1 \leq c^n_2 \leq \cdots$. Construct a chain d as follows. Let $d_0 = c^0_0$ and having chosen $d_0 \leq d_1 \leq \cdots \leq d_n$ such that $d_i = c^i_{j(i)}$, choose an integer $j(n+1) > j(n)$ so that $c^n_{j(n)} \leq c^{n+1}_{j(n+1)}$. This is always possible since $c^n \leq c^{n+1}$. Let $d_{n+1} = c^{n+1}_{j(n+1)}$. An obvious induction shows that $j(n) \geq n$ and so $c^i_n \leq d_n$ for $n \geq i$. For a fixed i, the finite set of $n \leq i$ does not matter and so $c^i \leq d$ for all i. Thus d is an upper bound on the c^i. Suppose $e = e_0 \leq e_1 \leq e_2 \leq \cdots$ is an upper bound on the c^i. Then since $d_n = c^n_{j(n)}$, and $c_n \leq e$, it follows that d_n is less than or equal to some term of e, so that $d \leq e$, whence $d = e$. Thus $\widehat{P}$ is an ω-CPO. Now suppose that A is an ω-CPO and $f : P \to A$ is an order-preserving map. Define $\widehat{f} : \widehat{P} \to A$ by $\widehat{f}(c) = \bigvee f(c_i)$ for $c = c_0 \leq c_1 \leq c_2 \leq \cdots$. It is easy to see that this preserves the order and therefore the equality. Since the join of a constant sequence is itself, this extends f. Finally, if d is the join of the c^i as constructed above, it easily follows from the fact that the terms of d are a selection of those of the c^i that $\widehat{f}(d) = \bigvee \widehat{f}(c^i)$.

3. Let $\mathscr{C}$ be the category of ω-CPOs and functions that preserve joins along countable increasing chains and let $D : \mathscr{I} \to \mathscr{C}$ be a diagram. Let A be the set that is the limit of the diagram in sets. Then A comes equipped with projections $p_I : A \to DI$ for I an object of $\mathscr{I}$. Say that $a \leq a'$ in A if for every object I of $\mathscr{I}$, $p_I(a) \leq p_I(a')$. Then A is a poset and the projections preserve order. If $a_0 \leq a_1 \leq a_2 \leq \cdots$ is an increasing chain in A, then for each I, $p_I(a_0) \leq p_I(a_1) \leq p_I(a_2) \leq \cdots$ is an countable increasing chain in DI and has a join a_I. If $\alpha : I \to I'$ is an arrow in $\mathscr{I}$, then

$$D(\alpha)(a_I) = D(\alpha)\left(\bigvee_n (p_I(a_n))\right) = \bigvee_n D(\alpha)(p_I(a_n)) = \bigvee_n p_{I'}(a_n) = a_{I'}$$

so there is a unique $a \in A$ such that $p_I(a) = a_I$. Evidently $a_n \leq a$ for each integer n. Also if $a' \in A$ is another upper bound for the $\{a_n\}$ then for each I, $p_I(a_n) \leq p_I(a')$ so that $a_I = p_I(a) \leq p_I(a')$ and then $a \leq a'$. This shows that A is an ω-CPO and it is clear that the projections preserve the joins along countable chains. If B is any poset and $\{f_I\} : B \to D$ is a natural transformation from the constant diagram to D, let $f : B \to A$ be the induced function. Let $b_0 \leq b_1 \leq b_2 \leq \cdots$ be an increasing chain in B with join b. Then for each I, $f_I(b) = \bigvee_n f_I(b_n)$ so that $p_I(f(b)) = \bigvee_n p_I(f(b_n))$. An argument similar to the above shows that this implies $f(b) = \bigvee_n f(b_n)$. The uniqueness of f follows because of the uniqueness of functions into a limit. This completes the proof for limits.

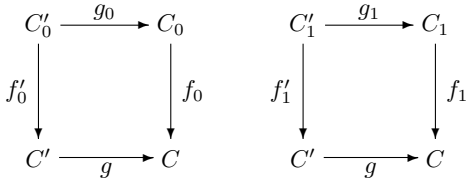
Assuming that the category of posets has colimits, one way to get colimits in the category of ω-CPOs is to form the colimit in the category of posets and apply the completion process of the preceding problem. If A is the colimit and $\widehat{A}$ its completion then any map from A to an ω-CPO extends to a unique map on $\widehat{A}$ that preserves joins along ω-chains. Putting the two universal mapping conditions together gives the result.

4. Since a chain is a directed set, one direction is immediate. For the other direction, let P be an ω-CPO and $D \subseteq P$ be a countable directed set in P. Since D is countable, we can name the elements of D as d_0, d_1, d_2, ... Let $c_0 = d_0$ and having chosen $c_0 \leq c_1 \leq c_2 \leq \cdots \leq c_n$ an increasing sequence of elements of D, let $c_{n+1} \in D$ be a common upper bound of d_{n+1} and c_n. A join of the $\{c_n\}$ is clearly an upper bound of D.

Solutions for Chapter 15

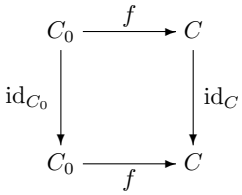
Section 15.1

1. Let the pullbacks be



Let $u : C_0 \to C_1$ and $v : C_1 \to C_0$ be the inverse isomorphisms such that $f_0 \circ v = f_1$ and $f_1 \circ u = f_0$. Then $f_1 \circ u \circ g_0 = f_0 \circ g_0 = g \circ f_0'$ so by the universal mapping property of the right hand pullback, there is a unique $u' : C_0' \to C_1'$ such that $g_1 \circ u' = f_1 \circ u$ and $f_1' \circ u' = f_0'$. Similarly, there is a unique $v' : C_1' \to C_0'$ such that $g_0 \circ v' = f_0 \circ v$ and $f_0' \circ v' = f_1'$. Thus f_0' and f_1' belong to the same subobject.

2. For $f : C_0 \rightarrowtail C$, the square



is a pullback. Thus the function induced by the identity of C assigns to each subobject of C the subobject itself (or an equivalent one).

3. If S and T are finite, so is the set of functions between them. In fact if we let $\#S$ denote the number of elements, then $\#[T \to S] = \#(S)^{\#(T)}$. For if $T = \emptyset$, then there is exactly one function from T to S no matter what S is. If $T \neq \emptyset$ and $S = \emptyset$, then there are no functions from T to S. If both are nonempty, we suppose by induction that the conclusion is true for sets with fewer elements than T, let $t \in T$ and $T_0 = T - \{t\}$. A function from T to S is determined by a function from T_0 to S plus an element of S for t to go to. Thus the number of such functions is

$$\#(S)^{\#(T_0)}\#(S) = \#(S)^{\#(T)-1}\#(S) = \#(S)^{\#(T)}$$

which is still finite. Thus the category of finite sets is cartesian closed. The two-element set is also finite so the category of finite sets has a subset classifier.

4. It is not quite sufficient to point out that when S is infinite, the set 2^S of subsets of S is not countable. The point to be made is that $\mathrm{Hom}(1, 2^S) \cong \mathrm{Hom}(S, 2)$ (where here 2 is the set $\{0, 1\}$) and the latter set of functions is not countable, while there is no countable or finite set T for which $\mathrm{Hom}(1, T)$ is not countable or finite. Thus there is no finite or countable set that has the universal mapping property required to be the powerset 2^S.

Section 15.2

1. In the category of sets, $\Omega = 2$ so we must verify that

$$\mathrm{Hom}(B, 2^A) \cong \mathrm{Hom}(A \times B, 2) \cong \mathrm{Sub}(A \times B)$$

If $f : B \to 2^A$ is a function, let $\phi(f) : A \times B \to 2$ by $\phi(f)(a,b) = \phi(b)(a)$. If $g : A \times B \to 2$ is a function, let $\psi(g) = \{(a,b) \mid g(a,b) = 1\} \subseteq A \times B$. If $U \subseteq A \times B$ is a subset, then let

$$\rho(U)(b)(a) = \begin{cases} 1 & \text{if } (a,b) \in U \\ 0 & \text{if } (a,b) \notin U \end{cases}$$

It is immediate that $\rho \circ \psi \circ \phi = \mathrm{id}_{\mathrm{Hom}(B,2^A)}$ and similarly for the other two serial composites, from which it is immediate that each of them is invertible, with inverse the composite of the other two.

2. For $x \in S$, let $[x]$ denote the equivalence class containing it. Then $[x] = [y]$ if and only if $(x,y) \in E$. Let $d^0, d^1 : E \to S$ be the two arrows mentioned above and let $d : S \to S/E$ denote the arrow $x \mapsto [x]$. Then for $(x,y) \in E$, $d \circ d^0(x,y) = d(x) = [x] = [y] = d(y) = d \circ d^1(x,y)$ so that d coequalizes d^0 and d^1.

Now suppose that $e^0, e^1 : T \to S$ such that $d \circ e^0 = d \circ e^1$. Then for all $x \in T$, $[e^0(x)] = [e^1(x)]$; equivalently $(e^0(x), e^1(x)) \in E$. Thus we can let $f : T \to E$ by defining $f(x) = (e^0(x), e^1(x))$. Then $d^0 \circ f(x) = d^0(e^0(x), e^1(x)) = e^0(x)$ and similarly $d^1 \circ f(x) = e^1(x)$. Finally if $g : T \to E$ is such that $d^0 \circ g(x) = d^0(e^0(x), e^1(x)) = e^0(x)$ and $d^1 \circ g(x) = e^1(x)$, then $g(x) = (e^0(x), e^1(x)) = f(x)$.

3. a. Suppose that d, e form an equivalence relation. This means that for any monoid N, the pair

$$\mathrm{Hom}(N,d), \mathrm{Hom}(N,e) : \mathrm{Hom}(N,E) \to \mathrm{Hom}(N,M)$$

gives an equivalence relation on the latter. This is another way of saying that

$$\langle \mathrm{Hom}(N,d), \mathrm{Hom}(N,e) \rangle : \mathrm{Hom}(N,E) \to \mathrm{Hom}(N,M) \times \mathrm{Hom}(N,M)$$

is an equivalence relation. Since

$$\mathrm{Hom}(N,M) \times \mathrm{Hom}(N,M) \cong \mathrm{Hom}(N, M \times M)$$

we have that $\langle d, e \rangle : E \to M \times M$ is monic. Since the image of a monoid homomorphism is a submonoid, the image is a submonoid. By letting $N = \mathbf{N}$, whence $\mathrm{Hom}(N,E)$ and $\mathrm{Hom}(N,M)$ are the underlying sets of E and M respectively, we conclude that the image of $\langle d, e \rangle$ is also an equivalence relation on the set underlying M. Thus it is a congruence.

Conversely, let $d, e : E \to M$ have the given property. Then up to isomorphism, we may suppose that $E \subseteq M \times M$ is both a submonoid and an equivalence relation and that d and e are the inclusion followed by the product projections. Then for any monoid N, $\mathrm{Hom}(N,E)$ consists of those pairs of arrows $f, g : N \to E$ such that for all $x \in N$, $(f(x), g(x)) \in E$. Since E is an equivalence relation, it is immediate that this makes $\mathrm{Hom}(N,E)$ into an equivalence relation on $\mathrm{Hom}(N,M)$.

b. Let us suppose that $E \subseteq M \times M$ is a submonoid and simultaneously an equivalence relation. For an $x \in M$, let $[x]$ denote the equivalence class containing x. If $x' \in [x]$ and $y' \in [y]$, then $(x,x') \in E$ and $(y,y') \in E$, whence so is

$(x,x')(y,y') = (xy, x'y')$. Thus $x'y' \in [xy]$ which means there an unambiguous multiplication defined on the set M/E of equivalence classes by $[x][y] = [xy]$. The associativity is obvious as is $[1][x] = [x][1] = [x]$. Thus M/E is a monoid and evidently the arrow $p : M \to M/E$, $x \mapsto [x]$ is a monoid homomorphism. $p(x) = p(x')$ if and only if $x' \in [x]$ if and only if $(x,x') \in E$ so E is the kernel pair of p.

4. Let f be the coequalizer of the two arrows $e^0, e^1 : E \to C$ and let the kernel pair be $d^0, d^1 : K \to C$. Since $f \circ e^0 = f \circ e^1$, it follows from the universal mapping property of kernel pairs that there is a unique arrow $e : E \to K$ such that $d^0 \circ e = e^0$ and $d^1 \circ e = e^1$. Now $d^0 \circ f = d^1 \circ f$ is one of the defining properties of kernel pairs. If $g : C \to B$ is an arrow such that $g \circ d^0 = g \circ d^1$, then $g \circ e^0 = g \circ d^0 \circ e = g \circ d^1 \circ e = g \circ e^1$. Since f is the coequalizer of e^0 and e^1, there is a unique $h : D \to B$ such that $h \circ f = g$.

5. Let $S = \{a\}$, $T = \{a, b\}$ and $f : S \to T$ be the inclusion. The kernel pair d^0, d^1 of f is the equality relation on S, that is $d^0 = d^1$, but f is not the coequalizer. For any set U with more than one element and any function $g : S \to U$ there are many functions $h : T \to U$ such that $h \circ f = g$, since $g(b)$ can be any element of U.

6. Let T be a set, $T_0 \subseteq T$ a subset and $f : T_0 \to S$ be an arbitrary function. Define $\widehat{f} : T \to S \cup \{*\}$ by

$$\widehat{f}(x) = \begin{cases} f(x) & \text{if } x \in T_0 \\ * & \text{otherwise} \end{cases}$$

Then $T_0 = \{x \in T \mid \widehat{f}(x) \in S\}$, which is another way of saying that

$$\begin{array}{ccc} T_0 & \xrightarrow{f} & S \\ \downarrow & & \downarrow \\ T & \xrightarrow[\widehat{f}]{} & S \cup \{*\} \end{array}$$

is a pullback. If g is a function with the same property, then $g(x) \in S$ if and only if $\widehat{f}(x)$ is, which means that $g(x) = *$ if and only if $f(x) = *$. Also, if $g(x) \neq *$, then $x \in T_0$ so that $g(x) = f(x) = \widehat{f}(x)$. Thus $g = \widehat{f}$ in all cases.

7. Since each object has a unique arrow to 1, there is a one to one correspondence between subobjects of an object A and partial arrows of A to 1. But partial arrows of A to 1 are in one to one correspondence with arrows $A \to \widetilde{1}$. Thus $\mathrm{Hom}(A, \widetilde{1}) \cong \mathrm{Sub}(A)$ which is the defining property of Ω.

Section 15.4

1. We are thinking of $\mathscr{C} = 0 \rightrightarrows 1$ as a category and 0 and 1 as two objects. The category of graphs is the category of contravariant functors $\mathscr{C} \to \mathbf{Set}$. In particular, the objects 0 and 1 represent functors and the question is which ones. Let us denote the two nonidentity arrows of $\mathscr{C}$ by s and t (mnemonic for 'source' and 'target'). A functor $F : \mathscr{C} \to \mathbf{Set}$ determines and is determined by the two sets $F(1)$ and $F(0)$ and the two arrows $F(s) : F(1) \to F(0)$ and $f(t) : F(1) \to F(0)$. In the case that

$F = \mathrm{Hom}(-, 0)$, the two sets are $\mathrm{Hom}(0, 0)$ and $\mathrm{Hom}(1, 0)$ which are a one-element set and the empty set, respectively, and the functions are the unique functions that exists in that case. Thus the contravariant functor represented by 0 is the graph **1** which we have also called No.

For take the functor represented by 1, the two sets are $\mathrm{Hom}(1, 1) = \{\mathrm{id}_1\}$ and $\mathrm{Hom}(0, 1) = \{s, t\}$. The two arrows take the element of $\mathrm{Hom}(1, 1)$ to s and t respectively. The graph **2** that we have also called No has just one arrow and two nodes which are the source and target of the arrow, respectively, and so is the graph represented by 1.

2. There are two ways of dealing with this question and similar ones. We already have an explicit description of the subobject classifier in this category and we could simply examine the set of nodes and set of arrows and see that their subfunctors have the desired structure. However, there is an easier way, once we know there is a subobject classifier. For the set of subobjects of the graph No is $\mathrm{Hom}(\mathrm{No}, \Omega)$ which, in turn, is isomorphic by the preceding problem, to the set of nodes of Ω. Similarly, the set of subobjects of the graph Ar is $\mathrm{Hom}(\mathrm{Ar}, \Omega)$, which is the set of arrows of Ω.

3. Let $\mathrm{NT}(F, G)$ denote the set of natural transformations between functors F and G. The Yoneda lemma implies that if $[M \to N] = F$, then

$$\begin{aligned} F(X) &\cong \mathrm{NT}(\mathrm{Hom}(-, X), F) \cong \mathrm{NT}(\mathrm{Hom}(-, X), [M \to N]) \\ &\cong \mathrm{NT}(\mathrm{Hom}(-, X) \times M, N) \end{aligned}$$

4. We have that

$$\mathrm{Sub}(\mathrm{Hom}(-, X)) \cong \mathrm{NT}(\mathrm{Hom}(-, X), \Omega) \cong \Omega(X)$$

the latter by the Yoneda Lemma.

Section 15.5

1. a. Let $C = \{c \in \mathscr{H} \mid a \wedge c \leq b\}$. Then $a \Rightarrow b = \bigvee_{c \in C} c$. Then

$$a \wedge (a \Rightarrow b) = a \wedge \bigvee_{c \in C} c = \bigvee_{c \in C} (a \wedge c) \leq b$$

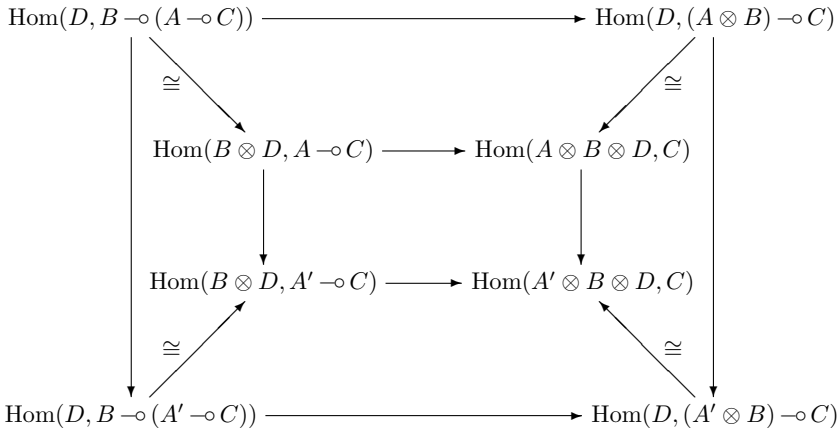
since a join of elements less than b is also less than b. Therefore, if $c \leq a \Rightarrow b$ then $a \wedge c \leq b$. On the other hand, if $a \wedge c \leq b$, then $c \in C$ so that $c \leq a \Rightarrow b$.

b. The fact that $a \leq b \wedge c$ if and only if $a \leq b$ and $a \leq c$ means that when the Heyting algebra is considered as a category, $\mathrm{Hom}(a, b \wedge c) = \mathrm{Hom}(a, b) \times \mathrm{Hom}(a, c)$. This means that $b \wedge c$ is the categorical product of b and c. But then $a \wedge c \leq b$ if and only if $c \leq a \Rightarrow b$ means that $\mathrm{Hom}(a \times c, b) = \mathrm{Hom}(c, a \Rightarrow b)$, which means that $a \Rightarrow b$ is the internal hom.

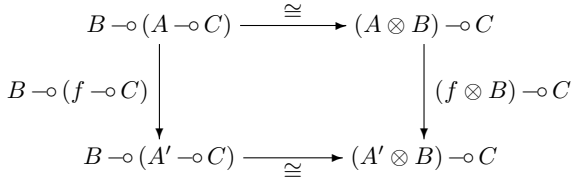
Solutions for Chapter 16

Section 16.2

1. Suppose $f : A' \to A$ is given. In the diagram below, the upper and lower trapezoids commute by the naturality of the isomorphism, the left and right hand trapezoids commute from the definitions of $B \multimap (f \multimap C)$ and $(f \otimes B) \multimap C$, resp. and the inner square commutes by definition of $f \multimap C$. Therefore the outer square commutes.



According to the Yoneda Lemma, it follows that



commutes. This is just the internalized version (that is, with Hom sets replaced by $\multimap$) of the main diagram used to prove the naturality in the preceding exercise. By replacing Hom sets everywhere in that argument with these 'internal hom objects', the above proof can be adapted to give the solution to this exercise.

2. We begin by showing that M is the unit. We must give, for each M-action S an isomorphism $l = lS : M \otimes_M S \to S$. Since $S \otimes M$ is defined only up to isomorphism as a coequalizer, one way of doing this is to show how S itself can be made into the relevant coequalizer, that is that there is a coequalizer diagram of the form

$$M \times M \times S \underset{d^1}{\overset{d^0}{\rightrightarrows}} M \times S \xrightarrow{d} S$$

where $d^0(m, n, s) = (mn, s)$ and $d^1(m, n, s) = (m, ns)$. Define $d(m, s) = ms$. The fact that $d \circ d^0 = d \circ d^1$ is just condition A–2 of 3.3.2.1. Now suppose that $f : M \times S$

$\to T$ is an equivariant map of M-actions such that $f \circ d^0 = f \circ d^1$. Define $g : S \to T$ by $g(s) = f(1, s)$. Then $(g \circ d)(m, s) = g(ms) = f(1, ms) = (f \circ d^1)(1, m, s) = (f \circ d^0)(1, m, s) = f(m, s)$ so that $f = g \circ d$. If $h : S \to T$ is another map such that $h \circ d = f$, then for all $s \in S$, $h(s) = (h \circ d)(1, s) = f(1, s) = g(s)$. This shows that S is the coequalizer, so that $M \otimes S$ can be taken to be S.

To show associativity, suppose that R, S and T are M-actions. Define $a = a(R, S, T) : R \otimes (S \otimes T) \to (R \otimes S) \otimes T$ by $a(r \otimes (s \otimes t)) = (r \otimes s) \otimes t$. We have that

$$a(mr \otimes (s \otimes t)) = (mr \otimes s) \otimes t = (r \otimes ms) \otimes t$$

$$= a(r \otimes (ms \otimes t)) = a(r \otimes m(s \otimes t))$$

so that a is well defined on the tensor product. It is just as easy to show it is M-equivariant and the similarly constructed inverse map shows it is an isomorphism. We similarly define $c = c(S, T) : S \otimes T \to T \otimes S$ by $c(s \otimes t) = t \otimes s$ which can be similarly shown to be well-defined and equivariant.

Now suppose $f : R \otimes S \to T$ is an equivariant map. Define $\phi(f) : S \to R \multimap T$ by $\phi(f)(s) \in S \multimap T$ is the map for which $\phi(f)(s)(r) = f(r, s)$. We have to show for each f and s, that $\phi(f)(s)$ is equivariant; for each f, that $\phi(f)$ is equivariant and, finally, that ϕ is equivariant. Each of the three is a different assertion. For the first, we have that

$$\phi(f)(s)(mr) = f(mr \otimes s) = f(m(r \otimes s)) = mf(r, s) = m\phi(f)(s)(r)$$

The second follows from

$$\phi(f)(ms)(r) = f(r \otimes ms) = f(m(r \otimes s)) = mf(r \otimes s)$$

$$= m\phi(f)(s)(r) = (m\phi(f)(s))(r)$$

The second computation looks quite similar to the first, but really is different since it depends on the fact that the action of M on a map $g : R \to T$ is by $(mg)(r) = mg(r)$. As for the third, we have that

$$\phi(mf)(s)(r) = (mf)(r, s) = mf(r, s) = m\phi(f)(s)(r)$$

$$= (m\phi(f)(s))(r) = (m\phi(f))(s)(r)$$

so that $\phi(mf) = m\phi(f)$ where we have twice used the definition of the action of M on a homomorphism. To go the other way, given an equivariant map $g : S \to R \multimap T$, define $\psi(g) : R \otimes S \to T$ by $\psi(g)(r, s) = g(s)(r)$. The crucial fact, that $\psi(g)(mr, s) = \psi(g)(r, ms)$, is shown by

$$\psi(g)(mr, s) = g(s)(mr) = m(g(s)(r))$$

$$= (mg(s))(r) = g(ms)(r) = \psi(g)(r, ms)$$

The rest, such as that ψ is equivariant, is similar. It is easily seen that $\psi = \phi^{-1}$.

3. a. If F is a functor, then let $F(A, -)$ be defined on arrows $g : B \to B'$ by $F(A, g) = F(\mathrm{id}_A, g)$ and $F(-, B)$ similarly defined on arrows $f : A \to A'$ by $F(f, B) =$

$F(f, \mathrm{id}_B)$. The commutation follows from the equations $(f,g) = (f, \mathrm{id}_{B'}) \circ (\mathrm{id}_A, g) = (\mathrm{id}_{A'}, g) \circ (f, \mathrm{id}_B)$.

Conversely, if we have the condition satisfied, define

$$F(f,g) = F(A,g) \circ F(f,B) = F(f,b) \circ F(A,g)$$

Since $\mathrm{id}_{(A,B)} = (\mathrm{id}_A, \mathrm{id}_B)$, we have

$$\begin{aligned} F(\mathrm{id}_{(A,B)}) &= F(\mathrm{id}_A, \mathrm{id}_B) = F(\mathrm{id}_A, B) \circ F(A, \mathrm{id}_B) \\ &= \mathrm{id}_{F(A,B)} \circ \mathrm{id}_{F(A,B)} = \mathrm{id}_{F(A,B)} \end{aligned}$$

If $(f,g) : (A,B) \to (C,D)$ and $(f',g') : (A',B') \to (A'',B'')$ then

$$\begin{aligned} F(f',g') \circ F(f,g) &= F(f',B'') \circ F(A',g') \circ F(A',g) \circ F(f,B) \\ &= F(f',B'') \circ F(A',g' \circ g) \circ F(f,B) \\ &= F(f',B'') \circ F(f,B'') \circ F(A,g \circ g') \\ &= F(f' \circ f, B'') \circ F(A, g \circ g') = F(f' \circ f, g' \circ g) \end{aligned}$$

b. The definition of naturality of $F(f,-)$ is that for all $g : B \to B'$, the square

$$\begin{array}{ccc} F(A,B) & \xrightarrow{F(A,g)} & F(A,B') \\ {\scriptstyle F(f,B)}\downarrow & & \downarrow{\scriptstyle F(f,B')} \\ F(A',B) & \xrightarrow[F(A',g)]{} & F(A',B') \end{array}$$

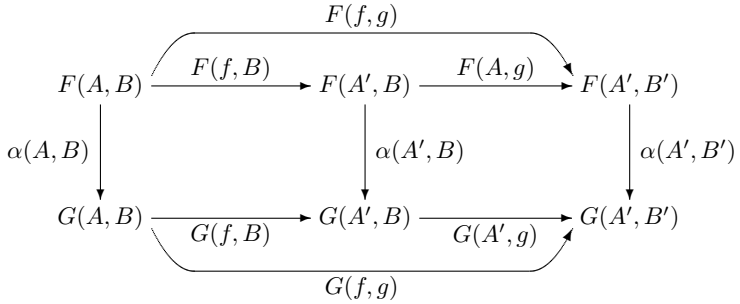
commutes, which is exactly the third condition and also exactly the naturality of $F(-,g)$.

c. Naturality of α means that for each arrow $(f,g) : (A,B) \to (A',B')$ the diagram

$$\begin{array}{ccc} F(A,B) & \xrightarrow{F(f,g)} & F(A',B') \\ {\scriptstyle \alpha(A,B)}\downarrow & & \downarrow{\scriptstyle \alpha(A',B')} \\ G(A,B) & \xrightarrow[G(f,g)]{} & G(A',B') \end{array}$$

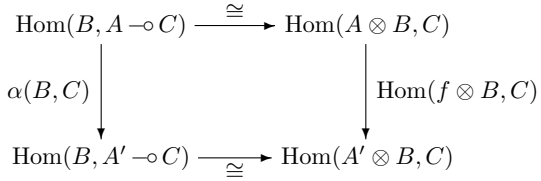
Specializing to the case that $f : A \to A$ is the identity, we see that if α is natural, then so is $\alpha(A,-)$ and similarly for $\alpha(-,B)$. Conversely, suppose that all $\alpha(A,-)$ and $\alpha(-,B)$ are natural. First we observe that $(f,g) = (\mathrm{id}, g) \circ (f, \mathrm{id})$ (this is the

crucial observation, actually) and hence $F(f,g) = F(\mathrm{id},g) \circ F(f,\mathrm{id})$ and analogously for G. This gives us the following commutative diagram

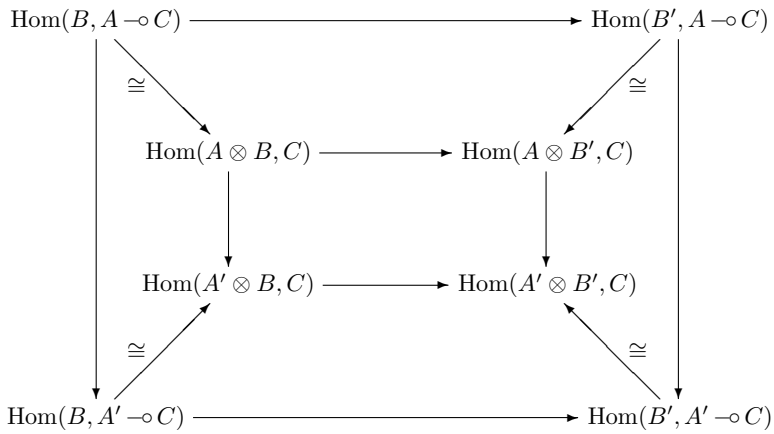


Since the two squares and the two triangles commute, so does the outer diagram, which is just the naturality of α.

d. Suppose we have $f : A' \to A$. Since the bottom map in the diagram below is an isomorphism, there is, for each B and C a unique arrow $\alpha(B,C) : \mathrm{Hom}(B, A \multimap C) \to \mathrm{Hom}(B, A' \multimap C)$ such that



commutes. I claim that this is natural in B. In fact, if $B' \to B$ is a map, we have a diagram



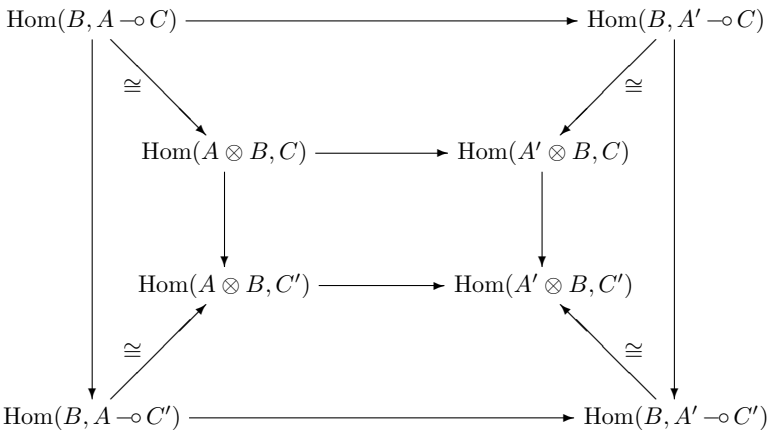
The left and right hand trapezoids commute by definition of α; the top and bottom ones because of the naturality of isomorphisms in the adjunction. The middle square commutes because $\otimes$ is a functor. It follows from a simple diagram chase that the

outer square commutes, which is just naturality of $\alpha(B,C)$ with respect to B. Thus we have, for each C, a natural transformation $\mathrm{Hom}(-, A \multimap C) \to \mathrm{Hom}(-, A' \otimes C)$. The Yoneda lemma implies that any such natural transformation is induced by a unique map we call $f \multimap C : A \multimap C \to A' \multimap C$. If f is the identity, then the identity map of $A \multimap C$ has all the properties required of $f \multimap C$. It then follows from uniqueness that $\mathrm{id} \multimap C$ is the identity. If $f' : A'' \to A'$ is a morphism, then one easily sees that $(f \multimap C) \circ (f' \multimap C)$ has all the properties required of $(f \circ f') \multimap C$ and the uniqueness implies they are equal. Thus the object function $A \mapsto A \multimap C$ extends to a functor.

If $h : C \to C'$ is given, then the same argument with appropriate changes gives maps $A \multimap h : A \multimap C \to A \multimap C'$ for each A. For example, for each A and B we have a unique function $\gamma(A,B) : \mathrm{Hom}(B, A \multimap C) \to \mathrm{Hom}(B, A \multimap C')$ for which

$$\begin{array}{ccc} \mathrm{Hom}(B, A \multimap C) & \xrightarrow{\cong} & \mathrm{Hom}(A \otimes B, C) \\ \downarrow{\scriptstyle \gamma(A,B)} & & \downarrow{\scriptstyle \mathrm{Hom}(A \otimes B, h)} \\ \mathrm{Hom}(B, A \multimap C') & \xrightarrow[\cong]{} & \mathrm{Hom}(A \otimes B, C') \end{array}$$

commutes. This is natural in B and thus there is a unique map $A \multimap h : A \multimap C \to A \multimap C'$ as above. To complete the argument, we must show that for $f : A' \to A$ and $h : C \to C'$, $(A' \multimap h) \circ (f \circ C) = (f \circ C') \circ (A \multimap h)$. This follows from the commutativity of



The four trapezoids commute from the definitions of $f \multimap -$ and $- \multimap h$, while the inner square does because $\otimes$ is a functor and hence the outer square does too. One easily sees that for three variables it is sufficient that all the restrictions to two variables be functorial, since all the necessary commutations can be performed by permuting two variables at a time.

Section 16.3

1. a. Since there is are only finitely many elements, there is a sup of all them, clearly the greatest element. Given any two elements x and y the set of elements z such that $z \leq x$ and $z \leq y$ is finite and therefore has a sup, which is evidently the inf of x and y.

b. This is actually an instance of the fact that right adjoints preserve limits (Theorem 13.3.7), since if L and M are treated as categories, f and g are functors and f is left adjoint to g, since from the definition of g, $x \leq g(y)$ if and only if $f(x) \leq y$. In particular, $f(x) \leq y$ and $f(x) \leq y'$ if and only if $f(x) \leq y \wedge y'$. Thus $x \leq g(y)$ and $x \leq g(y')$ if an only if $x \leq g(y \wedge y')$ from which it follows that $x \leq g(y) \wedge g(y')$ if and only if $x \leq g(y \wedge y')$ which implies that $g(y \wedge y') = g(y) \wedge g(y')$.

c. The previous part shows that when $f : L \to M$ is sup preserving, then so is $f^* = g : M^* \to L^*$. The uniqueness of adjoints implies that $f^{**} = f$ since f^{**} is another left adjoint to f (note that although $g : M \to L$ is a right adjoint to f, $g : M^{\mathrm{op}} \to L^{\mathrm{op}}$ is left adjoint.) Thus * looks like a duality. We must also show that $L \multimap M \cong M^* \multimap L^*$. But in fact the correspondence $f \mapsto f^*$ is a one-one correspondence between the two hom sets. If $f \leq g : L \to M$ are sup preserving, then I claim that $f^* \leq g^* : M^* \to L^*$, which is equivalent to the statement that $g^* \leq f^* : M \to L$. In fact, this is a general fact about adjoints. If F, $G : \mathscr{A} \to \mathscr{B}$ are functors with right adjoints F^*, G^*, resp., then any natural transformation $\alpha : F \to G$ induces a natural transformation $G^* \to F^*$ by the composite $G^* \to G^*FF^* \to G^*GF^* \to F^*$, where the first and third map are induced by the adjunctions and the middle one by α. Between posets, there is one natural transformation $f \to g$ if and only if $f \leq g$ and none otherwise. Thus $f \leq g$ if and only if $g^* \leq f^*$, just what is required for this to be a duality in the closed monoidal category.

Bibliography

At the end of each entry, the pages on which that entry is cited are listed in parentheses.

Adámek, J. and J. Rosický (1994). *Locally Presentable and Accessible Categories*. Cambridge University Press. (xv, 102, 139, 219, 299, 300, 312)

Arbib, M. and E. Manes (1975). *Arrows, Structures and Functors: The Categorical Imperative*. Academic Press. (372)

Arbib, M. and E. Manes (1980). 'Machines in a category'. *Journal of Pure and Applied Algebra*, volume **19**, pages 9–20. (372)

Asperti, A. and G. Longo (1991). *Categories, Types and Structures*. The MIT Press. (xv, 417, 422)

Asperti, A. and S. Martini (1992). 'Categorical models of polymorphism'. *Information and Computation*, volume **99**, pages 1–79. (332)

Backhouse, R., M. Bijsterveld, R. van Geldrop, and J. van der Woude (1995). 'Categorical fixed point calculus'. In Pitt et al. [1995], pages 159–179. (215)

Backus, J. (1981). 'The algebra of functional programs: Function level reasoning, linear equations, and extended definitions'. In *Formalization of Programming Concepts*, J. Diaz and I. Ramos, editors, volume 107 of *Lecture Notes in Computer Science*. Springer-Verlag. (20, 34, 215)

Backus, J. (1981). 'Is computer science based on the wrong fundamental concept of program?'. In *Algorithmic Languages*, J. W. deBakker and J. C. van Vliet, editors. North-Holland. (20)

Baez, J. C. (1997). 'An introduction to n-categories'. In Moggi and Rosolini [1997], pages 1–33. (140)

Bagchi, A. and C. Wells (1997). 'Graph-based logic and sketches I: The general framework'. Available by web browser from `http://www.cwru.edu/artsci/math/wells/pub/wellspub.html`. (311, 312)

Bagchi, A. and C. Wells (1997). 'Graph-based logic and sketches II: Equational deduction'. Available by web browser from `http://www.cwru.edu/artsci/math/wells/pub/wellspub.html`. (249)

Barendregt, H. (1984). *The Lambda Calculus – its Syntax and Semantics*. North-Holland. (29, 208)

Barr, M. (1971). 'Exact categories'. In *Exact Categories and Categories of Sheaves*, volume 236 of *Lecture Notes in Mathematics*. Springer-Verlag. (367)

Barr, M. (1979). **-Autonomous Categories*, volume 752 of *Lecture Notes in Mathematics*. Springer-Verlag. (422, 519)

Barr, M. (1986). 'Fuzzy sets and topos theory'. *Canadian Math. Bull.*, volume **24**, pages 501–508. (400)

Barr, M. (1986). 'Models of sketches'. *Cahiers de Topologie et Géométrie Différentielle Catégorique*, volume **27**, pages 93–107. (240, 244, 304)

Barr, M. (1989). 'Models of Horn theories'. *Contemporary Mathematics*, volume **92**, pages 1–7. (299)

Barr, M. (1990). 'Fixed points in cartesian closed categories'. *Theoretical Computer Science*, volume **70**, pages 65–72. (218)

Barr, M. (1991). '$*$-autonomous categories and linear logic'. *Mathematical Structures in Computer Science*, volume **1**, pages 159–178. (422)

Barr, M. (1995). 'Non-symmetric *-autonomous categories'. *Theoretical Computer Science*, volume **139**, pages 115–130. (422, 424)

Barr, M. (1996). '$*$-autonomous categories, revisited'. *Journal of Pure and Applied Algebra*, volume **111**, pages 1–20. (422)

Barr, M. (1996). 'The Chu construction'. *Theory and Applications of Categories*, volume **2**, pages 17–35. (424)

Barr, M. (1996). 'Separability of tensor in Chu categories of vector spaces'. *Mathematical Structures in Computer Science*, volume **6**, pages 213–217. (429)

Barr, M., C. McLarty, and C. Wells (1985). 'Variable set theory'. Technical report, McGill University. (77, 400)

Barr, M. and C. Wells (1985). *Toposes, Triples and Theories*, volume 278 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, New York. A list of corrections and additions is maintained in [Barr and Wells, 1993]. (xv, 139, 237, 279, 303, 307, 309, 311, 312, 347, 357, 358, 375, 384, 386, 387, 388, 389, 394, 396, 518)

Barr, M. and C. Wells (1992). 'On the limitations of sketches'. *Canadian Mathematical Bulletin*, volume **35**, pages 287–294. (312)

Barr, M. and C. Wells (1993). 'Corrections to Toposes, Triples and Theories'. Available by anonymous FTP from `triples.math.mcgill.ca` in directory `pub/barr` and by web browser from `http://www.cwru.edu/artsci/math/wells/pub/wellspub.html`. Corrections and additions to [Barr and Wells, 1985]. (518)

Bartha, M. (1992). 'An algebraic model of synchronous systems'. *Information and Computation*, volume **97**, pages 1–31. (417)

Bastiani, A. and C. Ehresmann (1972). 'Categories of sketched structures'. *Cahiers de Topologie et Géométrie Différentielle*, volume **13**, pages 104–213. (219, 307, 311)

Bell, J. L. (1988). *Toposes and Local Set Theories: An Introduction*. Oxford Logic Guides. Oxford University Press. (xv, 187, 384)

Bergman, C. and J. Berman (1998). 'Algorithms for categorical equivalence'. *Mathematical Structures in Computer Science*, volume **8**, pages 1–16. (87)

Bird, R. S. (1986). 'An introduction to the theory of lists'. In *Logic of Programming and Calculi of Discrete Designs*, M. Broy, editor, pages 5–42. Springer-Verlag. (112)

Bloom, S. and E. Wagner (1985). 'Many-sorted theories and their algebras with some applications to data types'. In *Algebraic Methods in Semantics*, M. Nivat and J. C. Reynolds, editors. Cambridge University Press. (250)

Boileau, A. and A. Joyal (1981). 'La logique des topos'. *Symbolic Logic*, volume **46**, pages 6–16. (384)

Borceux, F. (1994). *Handbook of Categorical Algebra I, II and III*. Cambridge University Press. (xv)

Carboni, A., P. Freyd, and A. Scedrov (1988). 'A categorical approach to realizability and polymorphic types'. In *Mathematical Foundations of Programming Language Semantics*, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 298 of *Lecture Notes in Computer Science*, pages 23–42. Springer-Verlag. (408)

Cartmell, J. (1986). 'Generalized algebraic theories and contextual categories'. *Annals Pure Applied Logic*, volume **32**, pages 209–243. (300)

Chen, H. G. and J. R. B. Cockett (1989). 'Categorical combinators'. Technical report, University of Calgary. (347)

Chu, P.-H. (1978). 'Constructing *-autonomous categories'. Master's thesis, McGill University. (424)

Chu, P.-H. (1979). 'Constructing *-autonomous categories'. Appendix to [Barr, 1979]. (424)

Cockett, J. R. B. (1989). 'On the decidability of objects in a locos'. In *Categories in Computer Science and Logic*, J. W. Gray and A. Scedrov, editors, volume 92 of *Contemporary Mathematics*. American Mathematical Society. (368, 371)

Cockett, J. R. B. (1990). 'List-arithmetic distributive categories: Locoi'. *Journal of Pure and Applied Algebra*, volume **66**, pages 1–29. (368, 369)

Cockett, J. R. B. (1993). 'Introduction to distributive categories'. *Mathematical Structures in Computer Science*, volume **3**, pages 277–307. (188, 189)

Cockett, J. R. B. and D. Spencer (1992). 'Strong categorical datatypes i'. In *Category Theory 1991*, R. Seely, editor. American Mathematical Society. (330, 376)

Coppey, L. and C. Lair (1984). 'Leçons de théorie des esquisses'. *Diagrammes*, volume **12**. (219)

Coppey, L. and C. Lair (1988). 'Leçons de théorie des esquisses, partie II'. *Diagrammes*, volume **19**. (219)

Coquand, T. (1988). 'Categories of embeddings'. In *Proceedings of the Third Annual Symposium on Logic in Computer Science, Edinburgh, 1988*, pages 256–263. Computer Society Press and IEEE. (332)

Coquand, T. and G. Huet (1988). 'The calculus of constructions'. *Information and Computation*, volume **76**, pages 95–120. (332)

Corradini, A. and A. Asperti (1993). 'A categorical model for logic programs: Indexed monoidal categories'. In *Semantics: Foundations and Applications (Beekbergen, 1992)*, volume 666 of *Lecture Notes in Computer Science*, pages 110–137. Springer-Verlag. (417)

Corradini, A. and F. Gadducci (1997). 'A 2-categorical presentation of term graph rewriting'. In Moggi and Rosolini [1997], pages 87–105. (141)

Coste, M. (1976). 'Une approche logique des théories définissables par limites projectives finies'. In *Séminaire Bénabou*. Université Paris-Nord. (300)

Cousineau, G., P.-L. Curien, and M. Mauny (1985). 'The categorical abstract machine'. In *Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*. Springer-Verlag. (195)

Crole, R. L. (1994). *Categories for Types*. Cambridge University Press. (xv)

Curien, P.-L. (1986). *Categorical Combinators, Sequential Algorithms and Functional Programming*. Wiley. (195)

Davey, B. A. and H. A. Priestley (1990). *Introduction to Lattices and Order*. Cambridge University Press. (180)

Devlin, K. (1993). *The Joy of Sets: Fundamentals of Contemporary Set Theory, Second Edition*. Undergraduate Texts in Mathematics. Springer-Verlag. (2)

Diers, Y. (1980). 'Catégories localement multiprésentables'. *Arch. Math.*, volume **34**, pages 344–356. (257, 347)

Diers, Y. (1980). 'Quelques constructions de catégories localement multiprésentables'. *Ann. Sci. Math. Québec*, volume **4**, pages 79–101. (257, 347)

Duval, D. and J.-C. Reynaud (1994). 'Sketches and computation (1)'. *Mathematical Structures in Computer Science*, volume **4**. (260)

Duval, D. and J.-C. Reynaud (1994). 'Sketches and computation (2)'. *Mathematical Structures in Computer Science*, volume **4**. (260)

Dybjer, P. (1986). 'Category theory and programming language semantics: an overview'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 165–181. Springer-Verlag. (201)

Dybkjaer, H. and A. Melton (1993). 'Comparing Hagino's categorical programming language and typed lambda-calculi'. *Theoretical Computer Science*, volume **111**, pages 145–189. (195)

Edalat, A. and M. B. Smyth (1993). 'I-categories as a framework for solving domain equations'. *Theoretical Computer Science*, volume **115**, pages 77–106. (147)

Ehresmann, C. (1968). 'Esquisses et types des structures algébriques'. *Bul. Inst. Polit. Iaşi*, volume **XIV**. (307, 311)

Ehrhard, T. (1988). 'A categorical semantics of constructions'. In *Proceedings of the Third Annual Symposium on Logic in Computer Science, Edinburgh, 1988*. Computer Society Press and IEEE. (332)

Ehrich, H. (1986). 'Key extensions of abstract data types, final algebras, and database semantics'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*. Springer-Verlag. (250)

Ehrig, H., H. Herrlich, H. J. Kreowski, and G. Preuss, editors (1988). *Categorical Methods in Computer Science*, volume 393 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Ehrig, H., H. J. Kreowski, J. Thatcher, E. Wagner, and J. Wright (1984). 'Parameter passing in algebraic specification languages'. *Theoretical Computer Science*, volume **28**, pages 45–81. (319)

Ehrig, H. and B. Mahr (1985). *Fundamentals of Algebraic Specifications I*. Springer-Verlag. (249)

Eilenberg, S. (1976). *Automata, Languages and Machines, Vol. B*. Academic Press. (76, 344)

Eilenberg, S. and J. C. Moore (1965). 'Adjoint functors and triples'. *Illinois J. Math.*, volume **9**, pages 381–398. (374)

Fokkinga, M. M. (1992). 'Calculate categorically!'. *Formal Aspects of Computing*, volume **4**. (xv)

Fourman, M. (1977). 'The logic of topoi'. In *Handbook of Mathematical Logic*, J. Barwise, editor. North-Holland. (384)

Fourman, M., P. Johnstone, and A. Pitts (1992). *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press. (xv)

Fourman, M. and S. Vickers (1986). 'Theories as categories'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 434–448. Springer-Verlag. (384)

Freyd, P. and A. Scedrov (1990). *Categories, Allegories*, volume 39 of *North-Holland Mathematical Library*. North-Holland. (xv, 87, 358, 384)

Gabriel, P. and F. Ulmer (1971). *Lokal Präsentierbare Kategorien*, volume 221 of *Lecture Notes in Mathematics*. Springer-Verlag. (300)

Girard, J.-Y. (1987). 'Linear logic'. *Theoretical Computer Science*, volume **50**, pages 69–108. (247)

Girard, J.-Y., P. Taylor, and Y. Lafont (1989). *Proofs and Types*. Cambridge University Press. (247, 422)

Godement, R. (1958). *Théorie des Faisceaux*. Hermann. (117)

Goguen, J. A. (1974). 'Concept representation in natural and artificial languages: Axioms, extensions and applications for fuzzy sets'. *Int. J. Man-Machine Studies*, volume **6**, pages 513–564. (401)

Goguen, J. A. (1978). 'Abstract errors for abstract data types'. In *Formal Description of Programming Concepts*, E. J. Neuhold, editor, pages 491–526. North-Holland. (222)

Goguen, J. A. (1988). 'What is unification? A categorical view of substitution, equation and solution'. Technical Report CSLI–88–124, Center for the Study of Language and Information. (293)

Goguen, J. A. (1991). 'A categorical manifesto'. *Mathematical Structures in Computer Science*, volume **1**, pages 49–68. (xv)

Goguen, J. A. and R. M. Burstall (1986). 'A study in the foundations of programming methodology: Specifications, institutions, charters and parchments'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 313–333. Springer-Verlag. (324)

Goguen, J. A., J.-P. Jouannaud, and J. Meseguer (1985). 'Operational semantics for order-sorted algebra'. In *Proc. 12th Int. Colloq. on Automata, Languages and Programming, Napflion, Greece*, volume 194 of *Lecture Notes in Computer Science*, pages 221–231. Springer-Verlag. (245)

Goguen, J. A., J. W. Thatcher, and E. G. Wagner (1978). 'An initial algebra approach to the specification, correctness and implementation of abstract data types'. In *Current Trends in Programming Methodology IV*, R. Yeh, editor, pages 80–149. Prentice-Hall. (245)

Goguen, J. A., J. W. Thatcher, E. G. Wagner, and J. B. Wright (1977). 'Initial algebra semantics and continuous algebras'. *J. ACM*, volume **24**, pages 68–95. (245)

Gray, J. W. (1966). 'Fibred and cofibred categories'. In *Proceedings of the Conference on Categorical Algebra, La Jolla 1965*. Springer-Verlag. (340)

Gray, J. W. (1974). *Formal Category Theory: Adjointness for 2-Categories*, volume 391 of *Lecture Notes in Mathematics*. Springer-Verlag. (140)

Gray, J. W. (1987). 'Categorical aspects of data type constructors'. *Theoretical Computer Science*, volume **50**, pages 103–135. (245, 250)

Gray, J. W. (1988). 'A categorical treatment of polymorphic operations'. In *Mathematical Foundations of Programming Language Semantics*, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 298 of *Lecture Notes in Computer Science*, pages 2–22. Springer-Verlag. (140)

Gray, J. W. (1989). 'The category of sketches as a model for algebraic semantics'. In *Categories in Computer Science and Logic*, J. W. Gray and A. Scedrov, editors, volume 92 of *Contemporary Mathematics*, pages 109–135. American Mathematical Society. (313, 318)

Gray, J. W. (1990). 'Executable specifications for data-type constructors'. *Diagrammes*, volume **24**, pages 7–31. (250)

Gray, J. W. (1991). 'Products in PER: an elementary treatment of the semantics of the polymorphic lambda calculus'. In *Category Theory at Work (Bremen, 1990)*, volume 18 of *Res. Exp. Math.*, pages 325–340. Heldermann. (408)

Gray, J. W. and A. Scedrov (1989). *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*. American Mathematical Society. (xv)

Guitart, R. and C. Lair (1980). 'Calcul syntaxique des modèles et calcul des formules internes'. *Diagrammes*, volume **4**. (219)

Guitart, R. and C. Lair (1981). 'Existence de diagrammes localement libres 1'. *Diagrammes*, volume **6**. (309)

Guitart, R. and C. Lair (1982). 'Existence de diagrammes localement libres 2'. *Diagrammes*, volume **7**. (309)

Gunter, C. (1992). *Semantics of Programming Languages*. The MIT Press. (xv, 195, 201)

Hagino, T. (1987). *A categorical programming language*. PhD thesis, University of Edinburgh. (347)

Hagino, T. (1987). 'A typed lambda calculus with categorical type constructors'. In *Category Theory and Computer Science*, D. Pitt, A. Poigné, and D. Rydeheard, editors, volume 283 of *Lecture Notes in Computer Science*, pages 140–157. Springer-Verlag. (195, 347)

Halmos, P. (1960). *Naive Set Theory*. Van Nostrand. (7, 160)

Hindley, J. R. and J. P. Seldin (1986). *Introduction to Combinators and λ-Calculus*. Cambridge University Press. (208)

Huet, G. (1986). 'Cartesian closed categories and lambda-calculus'. In *Combinators and Functional Programming Languages*, G. Cousineau, P.-L. Curien, and B. Robinet, editors, volume 242 of *Lecture Notes in Computer Science*. Springer-Verlag. (195)

Huwig, H. and A. Poigné (1990). 'A note on inconsistencies caused by fixpoints in a cartesian closed category'. *Theoretical Computer Science*, volume **73**, pages 101–112. (215)

Hyland, J. M. E. (1982). 'The effective topos'. In *The L. E. J. Brouwer Centenary Symposium*, pages 165–216. North-Holland. (384, 409)

Hyland, J. M. E. and A. Pitts (1989). 'The theory of constructions: Categorical semantics and topos-theoretic models'. In *Categories in Computer Science and Logic*, J. W. Gray and A. Scedrov, editors, volume 92 of *Contemporary Mathematics*, pages 137–199. American Mathematical Society. (332, 384)

Jacobson, N. (1974). *Basic Algebra I*. W. H. Freeman. (18)

Ji-Feng, H. and C. A. R. Hoare (1990). 'Categorical semantics for programming languages'. In *Mathematical Foundations of Programming Semantics*, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 442 of *Lecture Notes in Computer Science*, pages 402–417. Springer-Verlag. (140, 147)

Johnson, M. (1989). 'The combinatorics of n-categorical pasting'. *Journal of Pure and Applied Algebra*, volume **62**, pages 211–225. (151)

Johnstone, P. T. (1977). *Topos Theory*. Academic Press. (384, 387, 388, 390, 396)

Johnstone, P. T. and R. Paré (1978). *Indexed Categories and their Applications*. Lecture Notes in Mathematics. Springer-Verlag. (330)

Kasangian, S. and S. Vigna (1991). 'Trees in distributive categories'. In *Category Theory (Como, 1990)*, volume 1488 of *Lecture Notes in Mathematics*, pages 237–248. Springer-Verlag. (188)

Kelly, G. M. (1974). 'On clubs and doctrines'. In *Proceedings Sydney Category Theory Seminar 1972/1973*, G. M. Kelly, editor, volume 420 of *Lecture Notes in Mathematics*. Springer-Verlag. (343, 344)

Kelly, G. M. (1982). *Basic Concepts of Enriched Category Theory*. Cambridge University Press. (140, 145, 377)

Kelly, G. M. (1982). 'On the essentially-algebraic theory generated by a sketch'. *Bulletin of the Australian Mathematical Society*, volume **26**, pages 45–56. (311)

Kelly, G. M. and R. Street (1973). 'Review of the elements of 2-categories'. In *Proceedings Sydney Category Theory Seminar 1972/1973*, volume 420 of *Lecture Notes in Mathematics*. Springer-Verlag. (140)

Kleisli, H. (1965). 'Every standard construction is induced by a pair of adjoint functors'. *Proc. Amer. Math. Soc.*, volume **16**, pages 544–546. (374)

Kock, A. (1972). 'Strong functors and monoidal monads'. *Archiv der Mathematik*, volume **20**, pages 113–120. (376)

Lair, C. (1987). 'Trames et sémantiques catégoriques des systèmes de trames'. *Diagrammes*, volume **18**. (312)

Lallement, G. (1979). *Semigroups and Combinatorial Applications*. Wiley. (76)

Lambek, J. (1970). 'Subequalizers'. *Canad. Math. Bull.*, volume **13**, pages 337–349. (365)

Lambek, J. (1986). 'Cartesian closed categories and typed lambda-calculi'. In *Combinators and Functional Programming Languages*, G. Cousineau, P.-L. Curien, and B. Robinet, editors, volume 242 of *Lecture Notes in Computer Science*, pages 136–175. Springer-Verlag. (208)

Lambek, J. and P. Scott (1984). 'Aspects of higher order categorical logic'. In *Mathematical Applications of Category Theory*, J. W. Gray, editor, volume 30 of *Contemporary Mathematics*, pages 145–174. American Mathematical Society. (208)

Lambek, J. and P. Scott (1986). *Introduction to Higher Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press. (xv, 102, 187, 195, 208, 211, 212, 384)

Lawvere, F. W. (1963). *Functorial Semantics of Algebraic Theories*. PhD thesis, Columbia University. (56)

Lawvere, F. W. (1966). 'The category of categories as a foundation for mathematics'. In *Proceedings of the Conference on Categorical Algebra, La Jolla 1965*. Springer-Verlag. (56, 303)

Lawvere, F. W. (1989). 'Qualitative distinctions between some toposes of generalized graphs'. In *Categories in Computer Science and Logic*, volume 92. American Mathematical Society. (11)

Lellahi, S. K. (1989). 'Categorical abstract data types'. *Diagrammes*, volume **21**, pages 1–23. (245)

Lewis, H. R. and C. H. Papadimitriou (1981). *Elements of the Theory of Computation*. Prentice-Hall. (76, 148, 185, 254, 286, 383, 486)

Linton, F. E. J. (1969). 'Applied functorial semantics'. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 53–74. Springer-Verlag. (116)

Linton, F. E. J. (1969). 'An outline of functorial semantics'. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 7–52. Springer-Verlag. (116)

Lüth, C. and N. Ghani (1997). 'Monads and modular term rewriting'. In Moggi and Rosolini [1997], pages 69–86. (376)

Mac Lane, S. (1971). *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag. (xv, 126, 347, 357, 358, 415, 417)

Mac Lane, S. and I. Moerdijk (1992). *Sheaves in Geometry and Logic*. Universitext. Springer-Verlag. (200, 384, 395, 396)

Main, M., A. Melton, M. Mislove, and D. Schmidt, editors (1988). *Mathematical Foundations of Programming Language Semantics*, volume 298 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Makkai, M. (1997). 'Generalized sketches as a framework for completeness theorems, I, II, III'. *Journal of Pure and Applied Algebra*, volume **115**, pages 49–79, 179–212, 241–274. (312)

Makkai, M. and R. Paré (1990). *Accessible Categories: the Foundations of Categorical Model Theory*, volume 104 of *Contemporary Mathematics*. American Mathematical Society. (102, 312)

Makkai, M. and G. Reyes (1977). *First Order Categorical Logic*, volume 611 of *Lecture Notes in Mathematics*. Springer-Verlag. (xv, 102, 187, 288, 384)

Manes, E. G. (1986). 'Weakest preconditions: Categorial insights'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 182–197. Springer-Verlag. (276)

Manes, E. G. and M. Arbib (1986). *Algebraic Approaches to Program Semantics*. Springer-Verlag. (xv)

Marti Oliet, N. and J. Meseguer (1991). 'From Petri nets to linear logic through categories: a survey'. *International Journal of Foundations of Computer Science*, volume **2**, pages 297–399. (417)

Martin, C. E., C. A. R. Hoare, and J. He (1991). 'Pre-adjunctions in order enriched categories'. *Mathematical Structures in Computer Science*, volume **1**, pages 141–158. (147)

McLarty, C. (1986). 'Left exact logic'. *Journal of Pure and Applied Algebra*, volume **41**, pages 63–66. (300, 324)

McLarty, C. (1989). 'Notes toward a new philosophy of logic'. Technical report, Case Western Reserve University. (56)

McLarty, C. (1992). *Elementary Categories, Elementary Toposes*, volume 21 of *Oxford Logic Guides*. Clarendon Press. (xv, 384, 387, 388, 390, 396)

Meseguer, J. and J. A. Goguen (1985). 'Initiality, induction and computability'. In *Algebraic Methods in Semantics*, M. Nivat and J. C. Reynolds, editors. Cambridge University Press. (245)

Mikkelson, C. J. (1976). *Lattice Theoretic and Logical Aspects of Elementary Topoi*, volume 25 of *Aarhus University Various Publications Series*. Aarhus University. (389)

Mitchell, J. C. and P. J. Scott (1989). 'Typed lambda calculus and cartesian closed categories'. In *Categories in Computer Science and Logic*, J. W. Gray and A. Scedrov, editors, volume 92 of *Contemporary Mathematics*, pages 301–316. American Mathematical Society. (195)

Moggi, E. (1989). 'Computational lambda-calculus and monads'. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 14–23. IEEE. (376)

Moggi, E. (1991). 'A category-theoretic account of program modules'. *Mathematical Structures in Computer Science*, volume **1**, pages 103–139. (376)

Moggi, E. (1991). 'Notions of computation and monads'. *Information and Control*, volume **93**, pages 155–92. (376)

Moggi, E. and G. Rosolini, editors (1997). *Category Theory and Computer Science, 7th International Conference*. Springer-Verlag. (xv, 517, 519, 524, 525)

Mulry, P. S. (1992). 'Strong monads, algebras and fixed points'. In *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press. (376)

Nico, W. (1983). 'Wreath products and extensions'. *Houston J. Math.*, volume **9**, pages 71–99. (340, 344)

Otto, J. R. (1995). *Complexity Doctrines*. PhD thesis, McGill University. (185)

Pavlović, D. and S. Abramsky (1997). 'Specifying interaction categories'. In Moggi and Rosolini [1997], pages 147–158. (340, 422)

Peake, E. J. and G. R. Peters (1972). 'Extension of algebraic theories'. *Proceedings of the American Mathematical Society*, volume **32**, pages 358–362. (311)

Permvall, O. (1991). 'A bibliography on sketches from the computer science point of view'. *Bulletin of the European Association for Theoretical Computer Science*, volume **45**, pages 238–255. (219)

Pierce, B. (1991). *Basic Category Theory for Computer Scientists*. The MIT Press. (xv)

Pitt, D. (1986). 'Categories'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 6–15. Springer-Verlag. (22)

Pitt, D., S. Abramsky, A. Poigné, and D. Rydeheard, editors (1986). *Category Theory and Computer Programming*, volume 240 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Pitt, D., P.-L. Curien, S. Abramsky, A. M. Pitts, A. Poigné, and D. E. Rydeheard, editors (1991). *Category Theory and Computer Science (Paris, 1991)*, volume 530 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Pitt, D., A. Poigné, and D. Rydeheard, editors (1987). *Category Theory and Computer Science*, volume 283 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Pitt, D., D. Rydeheard, P. Dybjer, A. Pitts, and A. Poigné, editors (1989). *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*. Springer-Verlag. (xv)

Pitt, D., D. E. Rydeheard, and P. Johnstone, editors (1995). *Category Theory and Computer Science, 6th International Conference*. Springer-Verlag. (xv, 517)

Pitts, A. M. (1982). 'Fuzzy sets do not form a topos'. *Fuzzy Sets and Systems*, volume **8**, pages 101–104. (400)

Poigné, A. (1986). 'Elements of categorical reasoning: Products and coproducts and some other (co-)limits'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 16–42. Springer-Verlag. (174)

Power, A. J. (1990). 'A 2-categorical pasting theorem'. *Journal of Algebra*, volume **129**, pages 439–445. (151)

Power, A. J. (1990). 'An algebraic formulation for data refinement'. In *Mathematical Foundations of Programming Semantics*, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 442 of *Lecture Notes in Computer Science*, pages 402–417. Springer-Verlag. (141, 376)

Power, A. J. (1991). 'An n-categorical pasting theorem'. In *Category Theory (Como, 1990)*, volume 1488 of *Lecture Notes in Mathematics*, pages 326–358. Springer-Verlag. (151)

Power, A. J. and C. Wells (1992). 'A formalism for the specification of essentially algebraic structures in 2-categories'. *Mathematical Structures in Computer Science*, volume **2**, pages 1–28. (140, 141, 312)

Reichel, H. (1987). *Initial Computability, Algebraic Specifications and Partial Algebras*. Clarendon Press. (300)

Reynolds, J. C. (1980). 'Using category theory to design implicit conversions and generic operators'. In *Proceedings of the Aarhus Workshop on Semantics-Directed Compiler Generation*, N. D. Jones, editor, volume 94 of *Lecture Notes in Computer Science*. Springer-Verlag. (51)

Rhodes, J. and B. Tilson (1989). 'The kernel of monoid homomorphisms'. *Journal of Pure and Applied Algebra*, volume **62**, pages 227–268. (343, 344, 345)

Rhodes, J. and P. Weil (1989). 'Decomposition techniques for finite semigroups i, ii'. *Journal of Pure and Applied Algebra*, volume **62**, pages 269–284. (344)

Rosebrugh, R. and R. Wood (1992). 'Relational databases and indexed categories'. In *Category Theory 1991*, R. Seely, editor. American Mathematical Society. (330)

Rosolini, G. (1987). 'Categories and effective computation'. In *Category Theory and Computer Science*, volume 283 of *Lecture Notes in Computer Science*. Springer-Verlag. (409)

Rosolini, G. (1990). 'About modest sets'. *International Journal of Foundations of Computer Science*, volume **3**, pages 341–353. (408)

Rutten, J. J. M. M. (1996). 'Universal coalgebra: a theory of systems'. Available by FTP from `ftp://ftp.cwi.nl/pub/CWIreports/AP/CS-R9652.ps.Z`. (254)

Rutten, J. J. M. M. (1998). 'Automata and coinduction (an exercise in coalgebra)'. Available by FTP from `ftp://ftp.cwi.nl/pub/CWIreports/SEN/SEN-R9803.ps.Z`. (254)

Rydeheard, D. E. and R. M. Burstall (1985). 'Monads and theories: A survey for computation'. In *Algebraic Methods in Semantics*, M. Nivat and J. C. Reynolds, editors. Cambridge University Press. (292, 376)

Rydeheard, D. E. and R. M. Burstall (1986). 'A categorical unification algorithm'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 493–505. Springer-Verlag. (293)

Rydeheard, D. E. and R. M. Burstall (1988). *Computational Category Theory*. International Series in Computer Science. Prentice Hall. (xv)

Scott, D. (1972). 'Continuous lattices'. In *Toposes, Algebraic Geometry and Logic*, F. W. Lawvere, editor, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer-Verlag. (201, 376)

Scott, D. (1982). 'Domains for denotational semantics'. In *Automata, Languages and Programming*, M. Nielson and E. M. Schmidt, editors, volume 140 of *Lecture Notes in Computer Science*, pages 577–613. Springer-Verlag. (201, 376)

Sedgewick, R. (1983). *Algorithms*. Addison-Wesley. (278)

Seely, R. (1984). 'Locally cartesian closed categories and type theory'. *Proc. Cambridge Philos. Soc.*, volume **95**, pages 33–48. (362)

Seely, R. (1987). 'Categorical semantics for higher order polymorphic lambda calculus'. *J. Symbolic Logic*, volume **52**, pages 969–989. (362)

Seely, R. (1987). 'Modelling computations: a 2-categorical framework'. In *Proceedings of the Conference on Logic in Computer Science*. IEEE. (140)

Seely, R., editor (1992). *Category Theory 1991*, volume 13 of *Canadian Mathematical Society Conference Proceedings*. American Mathematical Society. (xv)

Smyth, M. B. (1983). 'The largest cartesian closed category of domains'. *Theoretical Computer Science*, volume **27**. (201)

Smyth, M. B. and G. D. Plotkin (1983). 'The category-theoretic solution of recursive domain equations'. *SIAM J. Computing*, volume **11**, pages 761–783. (376)

Spivey, M. (1989). 'A categorical approach to the theory of lists'. In *Mathematics of Program Construction*, J. L. A. Van de Snepscheut, editor, volume 375 of *Lecture Notes in Computer Science*, pages 399–408. Springer-Verlag. (112)

Street, R. and R. Walters (1973). 'The comprehensive factorization of a functor'. *Bull. Amer. Math. Soc.*, volume **79**, pages 936–941. (345)

Tarlecki, A., R. M. Burstall, and J. A. Goguen (1991). 'Some fundamental algebraic tools for the semantics of computation III: Indexed categories'. *Theoretical Computer Science*, volume **91**, pages 239–264. (330)

Taylor, P. (1989). 'Quantitative domains, groupoids and linear logic'. In *Category Theory and Computer Science*, D. Pitt, D. Rydeheard, P. Dybjer, A. Pitts, and A. Poigné, editors, volume 389 of *Lecture Notes in Computer Science*. Springer-Verlag. (393)

Tennent, R. D. (1986). 'Functor category semantics of programming languages and logics'. In *Category Theory and Computer Programming (Guildford, 1985)*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 206–224. Springer-Verlag. (109)

Thatcher, J. W., E. G. Wagner, and J. B. Wright (1982). 'Data type specification: Parametrization and the power of specification techniques'. *ACM Transactions of Programming Languages and Systems*, volume **4**, pages 711–732. (319)

Vickers, S. (1992). 'Geometric theories and databases'. In *Applications of Categories in Computer Science (Durham, 1991)*, volume 177 of *London Mathematical Society Lecture Notes Series*, pages 288–314. Cambridge University Press. (384)

Volger, H. (1987). 'On theories which admit initial structures'. Technical report, Universität Passau. (304)

Volger, H. (1988). 'Model theory of deductive databases'. In *CSL '87. First Workshop on Computer Science Logic. Proceedings*, volume 329 of *Lecture Notes in Computer Science*. Springer-Verlag. (304)

Wadler, P. (1989). 'Theorems for free!'. In *Fourth ACM Symposium on Functional Programming Languages and Computer Architecture*, pages 347–359. Association for Computing Machinery. (376)

Wadler, P. (1992). 'Comprehending monads'. *Mathematical Structures in Computer Science*, volume **2**, pages 461–493. (376)

Wagner, E. G. (1986). 'Algebraic theories, data types and control constructs'. *Fundamenta Informatica*, volume **9**, pages 343–370. (23, 181)

Wagner, E. G. (1986). 'Categories, data types and imperative languages'. In *Category Theory and Computer Programming*, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of *Lecture Notes in Computer Science*, pages 143–162. Springer-Verlag. (181, 318, 319)

Wagner, E. G. (1987). 'A categorical treatment of pre- and post conditions'. *Theoretical Computer Science*, volume **53**. (276)

Wagner, E. G., S. Bloom, and J. W. Thatcher (1985). 'Why algebraic theories?'. In *Algebraic Methods in Semantics*, M. Nivat and J. C. Reynolds, editors. Cambridge University Press. (249)

Walters, R. F. C. (1991). *Categories and Computer Science*. Cambridge University Press. (xv, 188, 191)

Walters, R. F. C. (1992). 'An imperative language based on distributive categories'. *Mathematical Structures in Computer Science*, volume **2**, pages 249–256. (188)

Wells, C. (1976). 'Some applications of the wreath product construction'. *Amer. Math. Monthly*, volume **83**, pages 317–338. (344)

Wells, C. (1980). 'A Krohn-Rhodes theorem for categories'. *J. Algebra*, volume **64**, pages 37–45. (344)

Wells, C. (1988). 'Wreath product decomposition of categories I'. *Acta Sci. Math. Szeged*, volume **52**, pages 307–319. (344)

Wells, C. (1988). 'Wreath product decomposition of categories II'. *Acta Sci. Math. Szeged*, volume **52**, pages 321–324. (344)

Wells, C. (1990). 'A generalization of the concept of sketch'. *Theoretical Computer Science*, volume **70**, pages 159–178. (311, 312)

Wells, C. (1993). 'Sketches: An outline'. Technical report, Case Western Reserve University. Available by web browser from `http://www.cwru.edu/artsci/math/wells/pub/wellspub.html`. (219)

Wells, C. (1995). 'Communicating mathematics: Useful ideas from computer science'. *American Mathematical Monthly*, volume **102**, pages 397–408. (7)

Wells, C. and M. Barr (1988). 'The formal description of data types using sketches'. In *Mathematical Foundations of Programming Language Semantics*, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 298 of *Lecture Notes in Computer Science*. Springer-Verlag. (222, 243, 245)

Williams, J. H. (1982). 'Notes on the fp style of functional programming'. In *Functional Programming and its Applications*, P. H. Darlington, J. and D. Turner, editors. Cambridge University Press. (20)

Zilles, S. N., P. Lucas, and J. W. Thatcher (1982). 'A look at algebraic specifications'. Technical Report 41985, IBM T. J. Watson Research Center. (245)

Index

A-action, 360
abuse of notation, 24
acceptor states, 73
accessible category, 304
action, 71, 73, 326
adjoint, 342, 364
adjunction, 343
algebra, 215
algebraic structure, 31, 274
all colimits, 274
all finite colimits, 274
all finite limits, 265
all limits, 265
alphabet, 72, 234
amalgamated sum, 275
antisymmetric, 24
application, 13
apply, 109
arity, 239
arrow between fuzzy sets, 392
arrow category, 103, 321
arrow functor, 114, 123, 278
arrow of a category, 16
arrow of a graph, 8
assertion, 269
associative, 18, 97
associative identity (of a triple), 364
associative law, 220
atom, 84
automorphism, 41
$*$-autonomous category, 411

barred arrow notation, 3
base (of a cocone), 274
base (of a cone), 150, 214
base category (of a 2-category), 138
base category (of a fibration), 320
base category (of an opfibration), 326
bicategory, 56
bijective, 5
binary discrete cocone, 174
binary operation, 156
binary product functor, 344
binary products, 164, 167
binary relation, 24
binary tree, 297, 309
Boolean algebra, 84, 186, 193, 248, 280
bound (variable), 202

canonical binary products, 167
canonical finite products, 168
canonical injection, 174
carrier, 399
cartesian arrow, 319
cartesian category, 168
cartesian closed, 377
cartesian closed category, 190, 345, 349, 352, 406
cartesian product, 2, 5, 149
cartesian product of arrows, 161
cartesian square, 156
Cat, 65
category, 16, 293
category determined by a monoid, 26
category determined by a poset, 24, 43
category object, 293, 395
category of R-dynamics, 363
category of finite sets, 18, 84
category of fuzzy sets, 392
category of graphs, 27
category of models, 102, 135, 312
category of monoids, 32, 53, 168

category of posets, 27, 147
category of semigroups, 32, 43, 78
category of sets, 18, 41, 43, 46, 49, 56, 156, 272, 273, 279, 378
category of sets and partial functions, 19, 147, 176
category of sets and relations, 19, 147, 176, 406, 412
characteristic arrow, 378
chase a diagram, 96
Chu construction, 413
cleavage, 320
closed (term), 202
closed (under a binary operation), 26
closed monoidal category, 405, 406
closure operator, 365
cocomplete, 274
cocone, 245, 274
codomain, 3, 8
coequalizer, 271, 283
coherent category, 282
coherent functor, 282
colimit, 274, 349, 381
commutative cocone, 274
commutative cone, 263
commutative diagram, 93
commutative semigroup, 24
commutes (of a diagram), 93
compact $*$-autonomous category, 412
compatible family, 296
complement, 279, 362
complemented subobject, 384
complemented subobjects, 279
complete, 194, 265
complete Heyting algebra, 387
component (of a cone), 262
component (of a natural transformation), 82, 101
composable pair, 16
composite, 16, 101
composite function, 6
composition, 16, 142
computable, 180, 280
cone, 150, 164, 214, 262
confusion, 132, 133, 234, 253
congruence relation, 86, 133
conjunction calculus, 182
connected, 110
connected component, 111
constant, 44, 388
context free grammar, 144, 235
context free rewrite system, 144
continuous, 29
contravariant functor, 68
contravariant hom functor, 69
coordinate, 5
coordinate projections, 149
coproduct, 174
corestriction, 6
coretract, 370
cotriple, 365, 409
counit, 343
covariant, 68
covariant hom functor, 68, 170
CPO, 195
crisp, 394
crossed product, 328
curry, 190, 247

dæmon, 251
decidable, 362
deduction system, 181, 195
derivation tree, 144
derived category, 337
diagonal, 19
diagonal relation, 2
diagram, 92
direct image, 68, 346
directed poset, 373
directly derive, 144
discrete category, 45, 71
discrete cocone, 274
discrete graph, 9
discrete opfibration, 326
discrete wreath product, 334
disjoint sum, 279, 381
disjoint union, 185, 325
distributive category, 183, 184, 198
division, 336
domain, 3, 8
domain of a partial arrow, 382
Doolittle diagrams, 275
dual concept, 37
dual notion, 37
dual of a category, 36, 68
duality functor, 411
dualizing object, 412

dynamorphism, 363

effective equivalence relation, 380
effective topos, 400
element-free, 55
empty list, 25
empty path, 15
empty semigroup, 25, 274
empty set, 1, 392
endoarrow, 8
endofunctor, 355
endomorphism, 8, 41
enriched category, 137, 142
epimorphism, 52, 57, 261, 273
equalize, 260
equalizer, 260
equation, 21, 240
equational variety, 240
equivalence of categories, 82, 232, 330
equivalence relation, 272, 380
equivariant map, 72
essentially, 104
evaluation map, 409
excluded middle, 362
existential image, 68
exponential object, 191, 351
extension, 1
external (pair of arrows), 160
external functor, 395
extremal epimorphism, 56, 288

factorization system, 57, 414
factorization systems, 288
factors through, 48
faithful functor, 77, 87
family of sets, 38
FD sketch, 245
fiber, 322
fiber product, 267
fibration, 320
fibration as generalized product, 322
field, 248, 254
filtered poset, 373
Fin, 18, 35, 37, 84
final algebra, 248
final states, 73
finitary functor, 358
finite cone, 291
finite discrete cone, 214
finite discrete sketch, 245
finite graph, 9
finite limit sketch, 291
finite product, 168, 184
finite product sketch, 214, 225
finite state machine, 71, 72, 75, 76, 247, 363
finite sum, 184
finitely cocomplete, 274
finitely complete, 265
first projection, 64
fixed point, 29, 208, 355, 356, 368
FL sketch, 291
flat CPO, 211, 385
flatten, 109
FLS category, 282
fold, 109
forgetful functor, 65
formal, 217
formula, 181
FP sketch, 214, 225
FP theory, 231
free, 202, 343
free M-set, 365
free algebra, 237
free category, 39, 67, 75
free FP sketch, 284
free model, 136
free monoid, 25, 33, 75, 339, 363
free monoid functor, 66
full functor, 78
full subcategory, 35
full transformation monoid, 77
function, 3
functional programming language, 111, 169, 176, 195
functional property, 4
functor, 63, 72, 74
functor category, 128, 162, 330
functor induced by a sketch homomorphism, 312
functorial semantics, 87
fuzzy set, 392

G_0, 8
G_0-indexed set, 136
global element, 44, 46, 47

grammar, 144
graph, 8, 13, 27, 100, 385
graph of a function, 4
graph of sets and functions, 9
GRF, 27
Grf, 27
Grothendieck construction, 324, 396
Grothendieck topology, 387
group, 41
groupoid, 45, 295

hom function, 7
hom functor, 68, 69, 119, 170, 197
hom set, 17, 54, 368
homomorphism of graphs, 11, 27, 95
homomorphism of linear sketches, 127
homomorphism of models, 125, 301
homomorphism of models of a linear sketch with constants, 131
homomorphism of models of an FP sketch, 227
homomorphism of monoids, 31, 64, 108
homomorphism of relations, 31
homomorphism of semigroups, 31, 228
homomorphism of sketches, 305
homomorphism of split opfibrations, 330
homomorphism of sup semilattices, 175
homomorphism of u-structures, 99
horizontal composite, 116

idempotent, 45, 56, 225, 299
identity, 16
identity element, 25
identity function, 4, 32
identity homomorphism, 12
if-then, 280
image, 4, 305, 451
imposing the relation, 87
inclusion, 174
inclusion function, 4, 32
indexed function, 38
indexed set, 236, 322
indiscrete category, 71
induced natural transformation, 119
induction, 132
inf semilattice, 175
infimum, 152
initial algebra, 233, 250
initial algebra semantics, 124, 130, 238
initial model, 130, 233, 250
initial object, 43, 48, 49, 168, 184, 356
initial term algebra, 296
initial term model, 296
initial topos, 381
injective, 5, 7, 46
input process, 363
integer, 1, 26
interchange law, 117
internal, 160, 197, 375, 384, 395, 400, 401, 407
internal functor, 395
internal language, 205, 384
intersection, 288
inverse, 40
inverse function, 68
inverse image, 68, 268, 346
invertible, 41
isomorphism, 41, 232, 261
isomorphism of semigroups, 32

junk, 132, 133, 233, 253

kernel category, 336
kernel pair, 302, 380
Kleene closure, 25, 33, 109, 144, 340, 361
Kleisli category, 366, 367, 409
Krohn-Rhodes Theorem, 336

λ-calculus, 201, 368
language generated by a context free grammar, 144
large, 17
large graph , 9
lattice, 175
lazy evaluation, 224
LE sketches, 291
least fixed point, 356
least upper bound, 28, 175
left adjoint, 342
left cancellable, 46

left exact sketches, 291
left inverse, 53
lies over, 320
limit, 263, 349
linear logic, 409, 411
linear sketch, 125, 164
linear sketch with constants, 130
list, 25, 109, 245, 361
local recursion, 362
locally cartesian closed, 354, 379
locally presentable category, 292
locally recursive category, 362
locally small, 17
locos, 362
lower semilattice, 175

M-equivariant, 365
$\mathscr{M}$-extensional, 414
$\mathscr{M}$-separated, 414
M-set, 71
machine, 75
machine in $\mathscr{C}$, 363
map, 4
map lifting property, 67, 237
mapping, 4
mathematical induction, 132
Mod, 102
model of a graph, 100
model of a linear sketch, 125
model of a linear sketch with constants, 131
model of a signature, 239
model of a sketch, 301, 312
model of an FD sketch, 245
model of an FL sketch, 291
model of an FP sketch, 215, 229, 236
model of an FP theory, 231
modest set, 400
Mon, 32
monic, 46
mono, 46
monoid, 25, 31, 36, 45, 57, 71, 81, 108, 227, 229, 365, 367, 406
monoid action, 336
monoid congruence, 89
monoid homomorphism, 31, 108
monoidal category, 403
monomorphism, 46, 57, 261, 268
monotone function, 27, 147, 175
morphism of cones, 263
multiplication of a triple, 364

N, 1, 10, 12
n-ary product, 165
natural equivalence, 103
natural isomorphism, 82, 103
natural numbers, 1, 12, 26, 215, 246, 253, 308, 355
natural numbers object, 177, 199, 233
natural transformation, 101, 106, 119, 227
naturality condition, 101
nearly constant presheaf, 389
node, 8
nondeterminism, 75
nonterminal, 144
notational conventions for sketches, 225, 292
nullary product, 166

object of a category, 16
object of a graph, 8
objectwise products, 163
ω-chain, 28
ω-complete partial order, 28
ω-complete partial ordered object, 210
ω-CPO, 28
one to one, 5
one to one correspondence, 5
1-cell, 142
opcartesian, 320
operations, 125, 239
opfibration, 320, 327, 396
opposite of a category, 36, 68
order-enriched category, 143
ordered n-tuple, 2
ordered set, 24, 346

$\mathscr{P}$, 28, 29
P-valued set, 392
parallel pair, 260, 273, 300
parametrized natural numbers, 178
partial arrow, 382
partial arrows representable, 382
partial equivalence relation, 400

partial function, 19, 28, 29, 35, 147
partially ordered object, 210
partially ordered set, 42
paste diagrams, 97
path, 15
path category, 39
PER, 400
Pfn, 19, 147, 176
pointed set, 85
Pointwise Adjointness Theorem, 349
pointwise products, 163
polynomial functor, 362
poset, 24, 27, 32, 42, 44, 147, 152, 346
poset-enriched category, 143, 368
postcondition, 269
power object, 379
powerset, 28, 345, 379
powerset functor, 68
precondition, 269
preordered set, 24, 346
preserve a property, 78
preserve limits, 265
preserved by isomorphisms, 41
preserves canonical products, 170
preserves the product, 170
presheaf, 385
product, 36, 149, 150, 165, 176, 344
product category, 64, 70, 321
product cone, 150, 214
product diagram, 150
production, 144
programming language, 20, 28, 75, 87, 111, 149, 169, 176, 177, 180, 195, 243, 390
projection, 5, 149, 150
projection functor, 334
proof, 181
proper subobject, 49
pseudo-inverse, 82
pullback, 267, 281, 288, 352
pullback diagram, 267
pushout, 275

Q, 1
quotient category, 86

R, 1
R-algebra, 356
R-dynamic, 363
rational numbers, 1, 276
real numbers, 1
realizability set, 399
realizability topos, 400
recognized, 73
recognizer, 73
record type, 169, 310
recursion, 29, 30, 360
recursive category, 360
recursive function, 180, 381
recursive set, 383, 384
recursively enumerable, 280, 363, 383, 384
reduce, 109
reflect, 79
reflexive, 24, 127
regular category, 273, 381
regular cocone, 301
regular epimorphism, 273, 301, 381
regular functor, 273
regular monomorphism, 261
regular sketch, 302
Rel, 19, 147, 176, 406, 412
relation, 2, 4, 19, 147
relation induced by a functor, 86
remainder, 33
representable functor, 118, 171
representative functor, 83
representative subcategory, 43
represents, 118, 160
restriction, 6
restriction functions, 387
retract, 370
rewrite system, 144
right adjoint, 342
right inverse, 53
right regular representation, 335
rules of inference, 182
Russell's paradox, 2

$\mathscr{S}$-algebra, 215
S-indexed set, 38, 66
satisfy, 240, 315
Sem, 32
semantics, 28, 75, 87, 112, 130, 177, 180, 195, 376, 383, 384
semidirect product, 327

semigroup, 24, 31, 36, 43, 151, 218, 228, 299
semigroup homomorphism, 31
semilattice, 175, 406
sentence, 315
set, 1
Set, 18
set-valued functor, 71, 87, 118, 121, 385
setbuilder notation, 1
shape functor, 333
shape graph, 92
sheaf, 388
signature, 213, 239
simple graph, 10, 293
singleton set, 10
sketch, 213, 214, 245, 301, 305
sketch for categories, 293
sketch for graphs, 126
sketch for monoids, 227, 229
sketch for semigroups, 299
slice category, 37, 97, 322, 345, 352
small, 17
small category, 293
small graph, 9
Smyth–Plotkin category, 369
sort, 239
source, 8
specification, 149
split fibration, 321
split idempotent, 45
split monic, 169
split opfibration, 326, 396
splitting, 320
stable natural numbers object, 178, 199
stable regular epi, 278
stack, 307
standard wreath product, 335
$*$-autonomous category, 411
start state, 72
start symbol, 144
state machine, 247
state space, 71
state transition system, 71, 336
strict ω-CPO, 28
strict functor, 29
strict initial object, 168
strict partially ordered object, 210
strictly monotone, 27
string, 25, 144
structure type, 310
subcategory, 35, 65
subfunctor, 113
submonoid, 26
subobject, 49, 132, 376
subobject classifier, 378
subobject functor, 376, 378
substitutable, 202
substitution, 283
sum, 174, 176, 279
sum cocone, 174
sup semilattice, 175, 406
supremum, 28, 175
surjective, 5, 52
switch map, 159
symmetric monoidal category, 403

target, 8
term, 133, 239
term model, 131, 233
terminal (in a context free grammar), 144
terminal object, 43, 168, 184, 215, 248
ternary product diagram, 164
theory of a linear sketch, 128
theory of an FL sketch, 298
theory of an FP sketch, 231
topos, 377
total category, 320
total order, 295
transducer, 73
transitions, 71
transitive, 24
tree, 254
triangular action, 335
triple, 364
trivial Boolean algebra, 186
true (from 1 to Ω), 378
two-category, 117
2-category, 137
2-cell, 142
2-functor, 143
two-variable hom functor, 69
type, 38
type conversion, 50

typed λ-calculus, 202
typed finite state machine, 75
typed function, 38, 236
typed set, 38

u-structure, 99
unary operation, 99
unary product, 166
underlying functor, 65
underlying linear sketch, 125, 130
underlying set, 24, 99, 339
undirected path, 111
unification, 283
unit of a triple, 364
unit of an adjunction, 343
unitary identities (of a triple), 364
universal cone, 263
universal element, 122
universal image, 379
universal mapping property, 67, 151, 340, 343
universal model, 128, 231, 298
universal sum, 280, 381
upper bound, 28
upper semilattice, 175

value of a function, 3
variable element, 46, 55, 384
variable list, 239
variable set, 74
vertex of a cone, 214
vertical composite, 116

weakest precondition, 269
weakly reflexive graph, 303
wide, 36
wide intersection, 288
workspace, 63, 160, 329, 333, 384
wreath product, 333

Yoneda embedding, 118, 199
Yoneda functor, 118
Yoneda Lemma, 122

$\mathbf{Z}$, 1
$\mathbf{Z}_k$, 33
0-cell, 142