Discrete Mathematics and Theoretical Computer Science **4**, 2001, 193–234

The First-Order Theory of Ordering Constraints over Feature Trees

Martin Müller[1] and Joachim Niehren[1] and Ralf Treinen[2]

[1]*Programming Systems Lab, Universität des Saarlandes, Saarbrücken, Germany.*
`http://www.ps.uni-sb.de/~mmueller`, `http://www.ps.uni-sb.de/~niehren`
[2]*Laboratoire de Recherche en Informatique, Université Paris-Sud, Orsay, France.*
`http://www.lri.fr/~treinen`

received April 19, 1999, revised February 2001, accepted Aug 15, 2001.

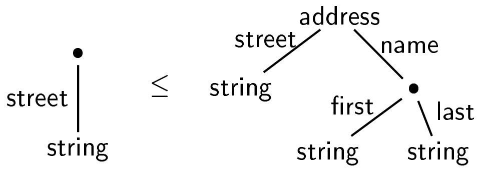
The system $FT_\leq$ of ordering constraints over feature trees has been introduced as an extension of the system FT of equality constraints over feature trees. We investigate the first-order theory of $FT_\leq$ and its fragments in detail, both over finite trees and over possibly infinite trees. We prove that the first-order theory of $FT_\leq$ is undecidable, in contrast to the first-order theory of FT which is well-known to be decidable. We show that the entailment problem of $FT_\leq$ with existential quantification is PSPACE-complete. So far, this problem has been shown decidable, coNP-hard in case of finite trees, PSPACE-hard in case of arbitrary trees, and cubic time when restricted to quantifier-free entailment judgments. To show PSPACE-completeness, we show that the entailment problem of $FT_\leq$ with existential quantification is equivalent to the inclusion problem of non-deterministic finite automata.

Keywords: feature constraints, logic of trees, automata

1 Introduction

Feature constraints have been used for describing records in constraint programming [1, 30, 31, 36] and record-like structures in computational linguistics [14, 12, 23, 26]. Feature constraints also occur naturally in type inference for programming languages with object types or record types [22, 5, 24].

Following [2, 4, 3], we consider feature constraints as predicate logic formulas interpreted in the structure of feature trees. A feature tree is a tree with unordered edges labeled by features and with possibly labeled nodes. Features are functional in that the features labeling the edges departing from the same node must be pairwise different. The structure of feature trees gives rise to an ordering in a very natural way which is called *weak subsumption ordering* in [7]. Consider the following example where an unlabeled node is indicated as • :



Here, the left tree τ_1 is said to *weakly subsume* the right tree τ_2 since τ_1 has fewer edges and node labels than τ_2. In other words, every *positive* assertion about the presence of labels or features that holds for τ_1 also holds for τ_2. In general, a tree τ_1 *weakly subsumes* a tree τ_2, written $\tau_1 \leq \tau_2$, if

- every word of features in the tree domain of τ_1 belongs to the tree domain of τ_2
- and the (partial) labeling function of τ_1 is contained in the labeling function of τ_2.

We consider the system $FT_\leq$ of ordering constraints over feature trees [18, 19, 17]. Its constraints φ are given by the following abstract syntax

$$\varphi \ ::= \ x \leq x' \ \mid \ x[f]x' \ \mid \ a(x) \ \mid \ \varphi \wedge \varphi'$$

where f denotes a *feature symbol* and a a *label symbol*. The constraints of $FT_\leq$ are interpreted in the structure of feature trees with the weak subsumption ordering. We distinguish two cases, the structure of finite feature trees and the structure of possibly infinite feature trees. A constraint $x \leq x'$ holds if the denotation of x weakly subsumes the denotation of x', $x[f]x'$ is valid if the denotation of x has an edge at the root that is labeled with the feature f and leads to the denotation of x', and $a(x)$ means that the root of the denotation of x is labeled with a.

The constraint system $FT_\leq$ is an extension of the well-investigated constraint system FT [2, 4], which provides for equality constraints $x{=}y$ rather than more general ordering constraints $x \leq y$. The system FT can be seen as a sub-system of $FT_\leq$ since $x = y$ can be expressed as $x \leq y \wedge y \leq x$ thanks to anti-symmetry of the weak subsumption order.

The full first-order theory of FT is decidable [4] and has non-elementary complexity [37]. The decidability question for the first-order theory of $FT_\leq$ has been raised in [17]. There, two indications in favour of decidability have been formulated: its analogy to FT and its relationship to second-order monadic logic. However, we show in this paper that the *the first-order theory of* $FT_\leq$ *is undecidable*. Our result holds in the structure of possibly infinite feature trees and, more surprisingly, even in the structure of finite feature trees. Our proof is based on an encoding of the Post Correspondence Problem using a technique of [33].

Once the undecidability of the first-order theory of $FT_\leq$ is settled, it remains to distinguish decidable fragments and their complexity. It is well-known that the satisfiability

	$FT_{\leq}$	$FT_{\leq}^{fin}$
Satisfiability of positive constraints	n^3 [18]	n^5 [7] n^3 [18]
Entailment w/o quantifiers	n^3 [18]	n^3 [18]
Entailment with quantifiers	Co-NP hard [17] PSPACE complete [here]	PSPACE hard [17] PSPACE complete [here]
Full theory	undecidable [here]	undecidable [here]

Fig. 1: Fragments of the first-order theories of $FT_{\leq}$ and $FT_{\leq}^{fin}$

problem of FT, its entailment problem $\varphi \models \varphi'$, and its entailment problem with existential quantifiers $\varphi \models \exists x_1 \ldots \exists x_n \varphi'$ can be solved in quasi-linear time [31]. The investigation of ordering constraints was initiated by Dörre [7] who gave an $O(n^5)$-algorithm for deciding satisfiability of $FT_{\leq}$-constraints. This result was improved to $O(n^3)$ in [18], where also the entailment problem of $FT_{\leq}$ concerning *quantifier-free* judgments $\varphi \models \varphi'$ was shown decidable in cubic time. The next step towards larger fragments of the theory of $FT_{\leq}$ was to consider entailment judgments with existential quantification $\varphi \models \exists x_1 \ldots \exists x_n \varphi'$ which are equivalent to unsatisfiability judgments $\varphi \wedge \neg\exists x_1 \ldots \exists x_n \varphi'$ with quantification below negation. As shown in [17], this problem is decidable, coNP-hard in case of finite trees, and PSPACE-hard in case of arbitrary trees. Decidability is proved by reduction to (weak) second order monadic logic (W)S2S. In a first reduction step, it is shown how to substitute the structure of feature trees by the related structure of so-called *sufficiently labeled* feature trees. We note that this step cannot be generalized to arbitrary first-order formulas beyond entailment with existential quantifiers. Since the full first-order theory of ordering constraints over sufficiently labeled (finite) feature trees can easily be encoded in (weak) second order monadic logic, decidability of entailment of $FT_{\leq}$ with existential quantifiers follows from the classical results on (W)S2S [32, 25].

This paper contributes the exact complexity of the entailment problem of $FT_{\leq}$ with existential quantification. We prove PSPACE-completeness, both in the structure of finite trees and in the structure of possibly infinite trees. This result is obtained by reducing the entailment problem of $FT_{\leq}$ with existential quantifiers to the inclusion problem of non-deterministic finite automata (NFA), and vice versa. Our reduction of entailment is based on the following idea: Given an existential formula $\exists\bar{x}\varphi$ we construct an automaton that accepts all its consequences in form of so called path constraints. An inverse reduction in the case of possibly infinite trees was first presented in [17]. In this paper, we present another inverse reduction which also applies for finite trees.

Applications and Related Work. The application domains of ordering constraints over feature trees are quite diverse. They have been used to describe so-called coordination phenomena in natural language [7] but also for the analysis of concurrent constraint programming languages [20]. The less general equality constraints over feature trees are central to constraint based grammars, and they provide record constraints for logic programming [31] or concurrent constraint programming [27, 15]. In concurrent constraint programming, entailment with existential quantification is needed for deciding the satisfaction of conditional guards. As mentioned above, our results are also relevant for constraint-based inference of record types and object types. In this context, the entailment test has recently received some attention as a justification for constraint simplification and as a means to check type interfaces [24, 5, 35, 16, 10, 11].

Originally, weak subsumption has been introduced as a weakening of subsumption. The subsumption ordering between feature structures [13, 28, 6] is omnipresent in linguistic theories like HPSG (head-driven phrase structure grammar) [23]. According to the more general view of [29, 7], the subsumption ordering and the weak subsumption ordering are definable between elements of an arbitrary feature algebra (which include the structure of feature trees and all feature structures). Following [8], ordering constraints interpreted with respect to the subsumption (resp. weak subsumption) ordering of arbitrary feature algebras are called subsumption (resp. weak subsumption) constraints. Syntactically, subsumption constraints, weak subsumption constraints, and $FT_{\leq}$ constraints coincide but semantically they differ. As proved in [8], the satisfiability problem of subsumption constraints is undecidable. The satisfiability problem of weak subsumption constraints is equivalent to the satisfiability problem of $FT_{\leq}$ constraints [7, 18] and hence decidable in cubic time.

Structure of the Paper. Section 2 reviews the definitions of feature trees and weak subsumption constraints. We demonstrate the expressivity of the constraint language in Section 3 and introduce some formulas used in later sections. Undecidability of the first-order theory of weak subsumption constraints is shown in Section 4. Finally, we show the entailment problem of existentially quantified constraints to be PSPACE-complete in Section 5. We prove the correctness of our algorithm in Section 6 and its completeness in Section 7.

A short version of this paper has been published as [21].

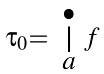
2 Ordering Constraints

The constraint system $FT_{\leq}$ is defined by a set of constraints, the structure of feature trees, and an interpretation of constraints over feature trees. We assume an infinite set $\mathcal{V}$ of *variables* ranged over by x, y, z, a set $\mathcal{F}$ of at least two *features* ranged over by f, g and a set $\mathcal{L}$ of *labels* ranged over by a, b.

2.1 Feature Trees

A *path* π is a word of features. The *empty path* is denoted by ε and the free-monoid concatenation of paths π and π' as $\pi\pi'$. We have $\varepsilon\pi = \pi\varepsilon = \pi$. A path π' is called a *prefix of* π if $\pi = \pi'\pi''$ for some path π''. A *tree domain* is a non-empty prefix closed set of paths.

A *feature tree* τ is a pair (D, L) consisting of a tree domain D and a partial function $L : D \rightharpoonup \mathcal{L}$ that we call *labeling function* of τ. Given a feature tree τ, we write D_τ for its tree domain and L_τ for its labeling function. For instance, $\tau_0 = (\{\varepsilon, f\}, \{(f, a)\})$ is a feature tree with domain $D_{\tau_0} = \{\varepsilon, f\}$ and $L_{\tau_0} = \{(f, a)\}$. A feature tree is *finite* if its tree domain is finite, and *infinite* otherwise. A *node of* τ is an element of D_τ. A node π of τ is *labeled with* a if $(\pi, a) \in L_\tau$. A node of τ is unlabeled if it is not labeled with any a. The *root* of τ is the node ε. The *root label* of τ is $L_\tau(\varepsilon)$, and $f \in \mathcal{F}$ is a *root feature* of τ if $f \in D_\tau$. A feature tree τ is *fully labeled* if L_τ is a total function with domain D_τ.



Given a tree τ with $\pi \in D_\tau$, we write as $\tau[\pi]$ the subtree of τ at path π; formally $D_{\tau[\pi]} = \{\pi' \mid \pi\pi' \in D_\tau\}$ and $L_{\tau[\pi]} = \{(\pi', a) \mid (\pi\pi', a) \in L_\tau\}$.

2.2 Syntax and Semantics

An $FT_{\leq}$ *constraint* φ is defined by the abstract syntax

$$\varphi \quad ::= \quad x \leq y \;\mid\; a(x) \;\mid\; x[f]y \;\mid\; \varphi_1 \wedge \varphi_2$$

where $a \in \mathcal{L}$ and $f \in \mathcal{F}$. In other words, an $FT_{\leq}$ constraint is a conjunction of *basic constraints* which are either *ordering constraints* $x \leq y$, *labeling constraints* $a(x)$, or *selection constraints* $x[f]y$.

We define the structure $FT_{\leq}$ over feature trees in which we interpret $FT_{\leq}$ constraints. Its universe consists of the set of all feature trees. The constraints are interpreted as follows:

$$\begin{array}{rll} \tau_1 \leq \tau_2 & \text{iff} & D_{\tau_1} \subseteq D_{\tau_2} \text{ and } L_{\tau_1} \subseteq L_{\tau_2} \\ \tau_1[f]\tau_2 & \text{iff} & D_{\tau_2} = \{\pi \mid f\pi \in D_{\tau_1}\} \text{ and } L_{\tau_2} = \{(\pi, a) \mid (f\pi, a) \in L_{\tau_1}\} \\ a(\tau) & \text{iff} & (\varepsilon, a) \in L_\tau \end{array}$$

The substructure of $FT_{\leq}$ whose universe contains only the finite trees is denoted by $FT_{\leq}^{fin}$. We will often use the following *decomposition* property without further mention:

Proposition 2.1 *If* $\tau_1 \leq \tau_2$ *and* $\tau_1[f]\tau_1'$ *and* $\tau_2[f]\tau_2'$ *then* $\tau_1' \leq \tau_2'$.

2.3 First-Order Formulas

If not specified otherwise, a formula is said to be valid (satisfiable) if it is valid (satisfiable) both in $FT_{\leq}$ and $FT_{\leq}^{fin}$. Our intention here is to treat both cases simultaneously and to note a distinction when needed only. Let Φ and Φ' be first-order formulas built from $FT_{\leq}$ constraints with the usual first-order connectives and quantifiers. We say that Φ *entails* Φ', written $\Phi \models \Phi'$, if $\Phi \rightarrow \Phi'$ is valid, and that Φ is *equivalent* to Φ' if $\Phi \leftrightarrow \Phi'$ is valid. We denote with $\mathcal{V}(\Phi)$ the set of variables occurring free in Φ, and with $\mathcal{F}(\Phi)$ and $\mathcal{L}(\Phi)$ the set of features and labels occurring in Φ.

3 Expressiveness of the First-Order Theory

In this section we introduce some abbreviations of formulas needed in Section 4. We use the usual abbreviations for ordering constraints, for instance we write $x \neq y$ for $\neg x = y$, $x<y$ for $x \leq y \wedge x \neq y$, $x \geq y$ for $y \leq x$ and $x \leq y \leq z$ for $x \leq y \wedge y \leq z$.

3.1 Minimal and Maximal Values

We can construct, for any formula φ, formulas $\mu x\, \varphi$ and $\nu x\, \varphi$ expressing that *x is minimal (maximal) with the property* φ:

$$\begin{array}{rcl} \mu x\, \varphi & := & \varphi \wedge \neg \exists y\, (\varphi[y/x] \wedge y<x) \\ \nu x\, \varphi & := & \varphi \wedge \neg \exists y\, (\varphi[y/x] \wedge y>x) \end{array}$$

Here, y is a fresh variable not occurring in φ, and $\varphi[y/x]$ denotes the formula where every free occurrence of x is replaced by y. Typically, x occurs free in φ but this is not required. Note that, in contrast to $\forall x$ and $\exists x$, μx and νx are *no* variable binders that restrict the scope of the variable x; hence x is free in $\mu x\, \varphi$ and in $\nu x\, \varphi$ if it is free in φ.

The formula $\mu x\, \varphi$ expresses that x denotes a minimal tree satisfying φ, which is *not* necassarily a smallest tree with this property. In analogy, $\nu x\, \varphi$ expresses that x denotes a maximal but not necessary greatest tree satisfying φ.

Example 1 *The sentence* $\exists x\, (\mu x\, true)$ *is valid in* $FT_{\leq}$ *and in* $FT_{\leq}^{fin}$ *(there even exists a smallest tree, namely* $(\{\varepsilon\}, \{\})$*). The formula* $\nu x\, true$ *is not satisfiable in* $FT_{\leq}^{fin}$ *but is satisfied in* $FT_{\leq}$ *by any fully labeled tree with domain* $\mathcal{F}^*$.

The difference between smallest and minimal trees is important for the formula $\mathsf{atom}(x)$ which expresses that x denotes an atom in the lattice-theoretic sense, i.e. that it is a tree strictly greater than the smallest tree $(\{\varepsilon\}, \{\})$ but with minimal distance (one feature or one label more):

$$\begin{array}{rcl} \mathsf{one\text{-}dist}(x,y) & := & \mu y\, x<y \\ \mathsf{atom}(y) & := & \exists x\, ((\mu x\, true) \wedge \mathsf{one\text{-}dist}(x,y)) \end{array}$$

Example 2 *The formula* $\mu x\,(x[0]x \wedge x[1]x)$ *is satisfied in* $FT_{\leq}$ *by* $(\{0,1\}^*, \{\})$, *that is the full binary and everywhere unlabeled tree, and is not satisfiable in* $FT^{fin}_{\leq}$ *since* $FT^{fin}_{\leq}$ *contains no infinite trees.*

3.2 Label Restrictions

The formula $x{\sim}y$ reads *x and y are consistent*, that is whenever $(\pi, a) \in L_\tau$ and $(\pi', a') \in L_{\tau'}$ then $a = a'$:

$$x{\sim}y := \exists z\,(x{\leq}z \wedge y{\leq}z)$$

For any label $a \in \mathcal{L}$ we write $x{\sim}a$ to express that the root of x is either unlabeled or labeled with a:

$$x{\sim}a := \exists y\,(x{\leq}y \wedge a(y))$$

The following formula expresses that the root of a tree is unlabeled:

$$\textsf{not-root-labeled}(x) := x{\sim}a \wedge x{\sim}b$$

where a and b are two arbitrary different label symbols. We obtain a first-class status of labels by encoding a label a as the feature tree $(\{\varepsilon\}, \{(\varepsilon, a)\})$.

$$\textsf{label-atom}(x) := \textsf{atom}(x) \wedge \neg\textsf{not-root-labeled}(x)$$

We can now express that x and y either have the same root label or are both unlabeled at the root by:

$$\textsf{same-root-label}(x,y) := \forall z\,(\textsf{label-atom}(z) \rightarrow (x{\sim}z \leftrightarrow y{\sim}z))$$

3.3 Arity Restrictions

We can simulate a first-class status of feature symbols by encoding a feature f by the tree $(\{\varepsilon, f\}, \emptyset)$.

$$\textsf{feature-atom}(x) := \textsf{atom}(x) \wedge \textsf{not-root-labeled}(x)$$

We can express that y has at least all the root features of x by

$$\forall z\,(\textsf{feature-atom}(z) \wedge z{\leq}x \rightarrow z{\leq}y)$$

The following formula expresses that x has exactly the root features $f_1, \ldots, f_n$:

$$\begin{array}{rcl} x\{f_1,\ldots,f_n\} & := & \exists x_1,\ldots,x_n\,(x[f_1]x_1 \wedge \ldots x[f_n]x_n \\ & & \wedge \forall y\,(y[f_1]x_1 \wedge \ldots \wedge y[f_n]x_n \wedge \textsf{same-root-label}(x,y) \rightarrow x{\leq}y)) \end{array}$$

These so-called *arity constraints* have been introduced in [31]. A decidable feature logic where feature symbols have first class status has been investigated in [34].

3.4 Inductive Properties

We start this section by a demonstration of the expressivity of $FT_{\leq}$ and show that we can express in $FT_{\leq}$ "inductive properties" of trees, that is properties that require an inductive construction (for instance an automaton) to define. We conclude the section by the definition of the predicate $\textsf{string-c}(x)$ that we will need in the undecidability proof of Section 4.

In the case of infinite trees it is in fact quite simple to express "inductive properties" of a tree. For instance, we can express that the domain of x contains the set $\{0,1\}^*$ by

$$\exists y\,(y[0]y \wedge y[1]y \wedge y{\leq}x)$$

The following formula says that the tree denoted by x has domain $\{0,1\}^*$ and that exactely one of its nodes is labeled with $\mathtt{a}$ whereas all its remaining nodes are unlabeled:

$$\begin{aligned}\textsf{a-singleton}(x) \quad := \quad & \exists y,z\,(\mu y\,(y[\mathtt{0}]y \wedge y[\mathtt{1}]y \wedge \textsf{not-root-labeled}(y)) \wedge \\ & \mu z\,(z[\mathtt{0}]z \wedge z[\mathtt{1}]z \wedge \mathtt{a}(z)) \wedge \\ & y{<}x{<}z \wedge \textsf{one-dist}(y,x))\end{aligned}$$

If $\textsf{a-singleton}(x)$ is satisfied then y denotes the complete binary, everywhere unlabeled tree (with domain $\{0,1\}^*$), and z denotes the complete binary, everywhere $\mathtt{a}$-labeled tree. The formula $\textsf{b-singleton}$ is defined analogously. We can now express that x denotes a tree with domain $\{0,1\}^*$ and that all its nodes are labeled with either $\mathtt{a}$ or $\mathtt{b}$ by:

$$\mu x\left(\forall y,z\,(\textsf{a-singleton}(y) \wedge \textsf{b-singleton}(z) \wedge y{\not\sim}z \rightarrow (y{\leq}x \vee z{\leq}x))\right)$$

The idea behind this formula is the following: an $\mathtt{a}$-singleton and a $\mathtt{b}$-singleton are inconsistent iff they have their label at the same position. Hence, the formula says that $\{0,1\}^* \subseteq D_x$ and every node of x which is reachable via a $\{0,1\}^*$-path is either labeled with $\mathtt{a}$ or with $\mathtt{b}$. The minimality of x yields $D_x \subseteq \{0,1\}^*$.

In case of finite trees we have to use another trick (which works also in case of infinite trees). The next formula is crucial for our undecidability proof. A tree τ satisfies this formula iff $\{\varepsilon,\mathtt{c}\} \subseteq D_\tau \subseteq \{\mathtt{c}\}^*$ and all its nodes are unlabeled:

$$\textsf{string-c}(x) := x\{\mathtt{c}\} \wedge \textsf{not-root-labeled}(x) \wedge \exists y\,(x[\mathtt{c}]y \wedge y{\leq}x)$$

The correctness of this definition of $\textsf{string-c}(x)$ with respect to the above stated semantics follows from the following lemma where we write c^n for the word $c\cdots c$ consisting of n letters c.

Lemma 3.1 *The formula $\exists y\,(x[c]y \wedge y{\leq}x)$ is satisfied by τ iff $c \in D_\tau$ and for all $k,m \geq 0$, whenever $c^{m+k} \in D_\tau$ then*

$$\tau[c^{m+k}]{\leq}\tau[c^m]$$

Proof. Let $\tau[\mathtt{c}]\tau'$ and $\tau'{\leq}\tau$. Obviously, $c \in D_\tau$. The inequality follows by induction: For any m, if $c^m \in D_\tau$ then $\tau[\mathtt{c}^m]{\leq}\tau[\mathtt{c}^m]$. Furthermore, for any k with $c^{m+k+1} \in D_\tau$ and $\tau[\mathtt{c}^{m+k}]{\leq}\tau[\mathtt{c}^m]$ we have that

$$\tau[\mathtt{c}^{m+k+1}] = \tau[\mathtt{c}][\mathtt{c}^{m+k}] = \tau'[\mathtt{c}^{m+k}]{\leq}\tau[\mathtt{c}^{m+k}]{\leq}\tau[\mathtt{c}^m]$$

For the other direction, since $\mathtt{c} \in D_\tau$ there is a τ' such that $\tau[\mathtt{c}]\tau'$. From the above inequality we get by setting $m = 0$ and $k = 1$ that

$$\tau' = \tau[\mathtt{c}^1]{\leq}\tau[\mathtt{c}^0] = \tau[\varepsilon] = \tau$$

□

4 Undecidability Results

Theorem 4.1 *The first-order theories of $FT^{fin}_{\leq}$ and of $FT_{\leq}$ are undecidable.*

The result holds for arbitrary (even empty) $\mathcal{L}$ and for $\mathcal{F}$ of cardinality ≥ 2. For the sake of clarity we use in the proof distinct label symbols $\mathtt{a},\mathtt{b},\mathtt{e}$ and pairwise distinct feature symbols $\mathtt{s},\mathtt{c},\mathtt{p},\mathtt{l},\mathtt{r}$. We prove Theorem 4.1 by reduction of the Post Correspondence Problem (PCP). The choice of PCP is motivated by the fact that our proof works by simulation of an iterative construction, and that PCP uses a technically very simple iteration. This is different in nature to the technique in [8] for the proof of undecidability of the satisfiability of strong subsumption constraints. There, Thue-systems could be used by exploiting a correspondence between word equations and the algebraic properties of feature structures. See [33] for a discussion of the proof technique employed in this chapter.

An instance of PCP is a finite sequence $P = ((p_i, q_i))_{i=1,\dots,m}$ of pairs of words from $\{\mathtt{a},\mathtt{b}\}^*$. Such an instance is *solvable* if there is a nonempty sequence $(i_1,\dots,i_n)$, $1 \le i_j \le m$, such that $p_{i_1}\cdots p_{i_n} = q_{i_1}\cdots q_{i_n}$. According to a classical result due to Post, it is undecidable whether an instance of the PCP is solvable.

In the following, let $P = ((p_i, q_i))_{i=1,\dots,m}$ be a fixed instance of PCP. We say that a pair (v,w) is *P-constructed from* a pair of words (v',w') if, for some j, $v = p_j v'$ and $w = q_j w'$. We say that a set X of pairs of words is *P-constructed* if every pair in X is either $(\varepsilon,\varepsilon)$ or is P-constructed from some other pair in X. To encode solvability of P into the theory of $FT^{fin}_{\le}$, resp. $FT_{\le}$, we employ the following equivalent definition of solvability:

Proposition 4.2 *P is solvable iff there is a P-constructed set X of pairs of words containing a pair (w,w) with $w \neq \varepsilon$.*

4.1 Words and Trees

Given a word $w \in \{\mathtt{a},\mathtt{b}\}^*$ over labels $a,b \in \mathcal{L}$ fixed above we denote its length by $|w|$ and for a natural number $1 \le j \le |w|$ we write $w.j$ for the j'th letter of w. There is an obvious one-to-one encoding function γ from words $w \in \{\mathtt{a},\mathtt{b}\}^*$ to feature trees for which we use the feature symbol $\mathtt{s}$ and label $\mathtt{e}$ that we also fixed above: $\gamma(w) = (D_w, L_w)$ where $D_w = \{\varepsilon, \mathtt{s}, \dots, \mathtt{s}^{|w|}\}$, $L_w(\mathtt{s}^j) = w.j$ for $0 \le j \le |w|-1$, and $L_w(\mathtt{s}^{|w|}) = \mathtt{e}$ (see Figure 2(a)).

We define a left-inverse function $\bar\gamma$, that is $\bar\gamma(\gamma(w)) = w$, from feature trees to (possibly infinite) words in $\{\mathtt{a},\mathtt{b}\}^\omega$ as follows: If τ does not have root feature $\mathtt{s}$, or if its root is unlabeled or has label different from $\mathtt{a}$ and from $\mathtt{b}$ then $\bar\gamma(\tau) = \varepsilon$. Otherwise let τ' be such that $\tau[\mathtt{s}]\tau'$. We define $\bar\gamma(\tau) = \mathtt{a}\cdot\bar\gamma(\tau')$ if τ has root label $\mathtt{a}$, and $\bar\gamma(\tau) = \mathtt{b}\cdot\bar\gamma(\tau')$ if τ has root label $\mathtt{b}$.

To express that y denotes the fixed word π appended with the denotation of x, we define for any $\pi \in \{\mathtt{a},\mathtt{b}\}^*$ a formula $\mathsf{app}_\pi(x,y)$, such that

1. if $\mathsf{app}_\pi[\tau,\tau']$ then $\pi\bar\gamma(\tau) = \bar\gamma(\tau')$
2. $\mathsf{app}_\pi[\gamma(w),\gamma(\pi w)]$ is valid

for all words w and feature trees τ,τ', by induction on π:

$$\begin{aligned}
\mathsf{app}_\varepsilon(x,y) &:= x = y\\
\mathsf{app}_{\mathtt{a}\pi}(x,y) &:= \mathtt{a}(y) \wedge \exists z\,(y[\mathtt{s}]z \wedge \mathsf{app}_\pi(x,z))\\
\mathsf{app}_{\mathtt{b}\pi}(x,y) &:= \mathtt{b}(y) \wedge \exists z\,(y[\mathtt{s}]z \wedge \mathsf{app}_\pi(x,z))
\end{aligned}$$

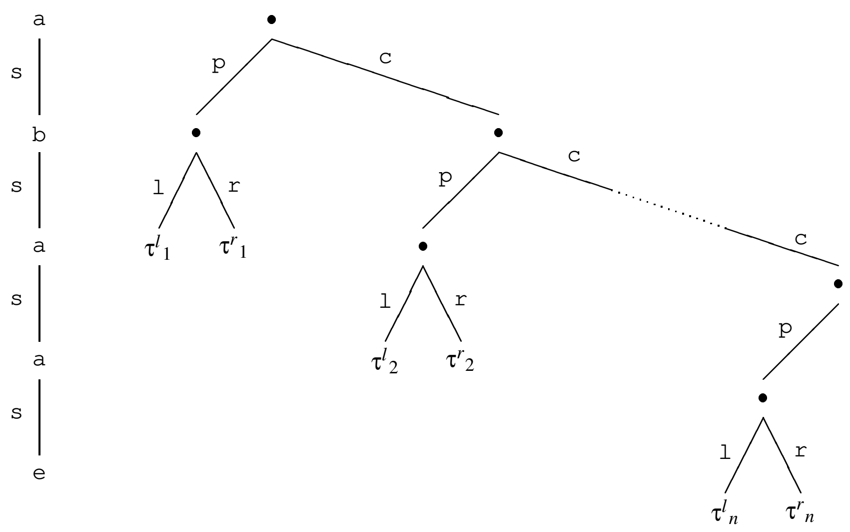
Furthermore, we define $\mathsf{eps}(x)$, expressing that x denotes a tree τ with $\bar\gamma(\tau) = \varepsilon$, by

$$\mathsf{eps}(x) := \neg\exists y\, x[\mathtt{s}]y \vee \neg(\mathtt{a}(x) \vee \mathtt{b}(x))$$

Finally, the following formula expresses that x denotes a finite string:

$$\mathsf{finite}(x) := \neg\exists y\,(y[\mathtt{s}]y \wedge y \le x)$$

In case of $FT^{fin}_{\le}$ this formula is, of course, equivalent to *true*.



(a) The string *abaa*.

(b) The solution sequence $(v_i, w_i)_i$.

Fig. 2: Representation of strings and of solution sequences.

4.2 *P*-Constructions

Provided an appropriate encoding of sets of pairs of words and a predicate $\mathsf{in}(x_l, x_r, s)$, expressing that the pair (x_l, x_r) is member of the set s, we can express that s is a P-constructed set of pairs of words and that P is solvable:

$$\begin{aligned}
\mathsf{construction}_P(s) \quad &:= \quad \forall y, y' \,(\mathsf{in}(y, y', s) \rightarrow ((\mathsf{eps}(y) \wedge \mathsf{eps}(y')) \\
&\qquad \vee \exists z, z' \,(\mathsf{in}(z, z', s) \wedge \bigvee_{j=1\dots m} (\mathsf{app}_{p_j}(z, y) \wedge \mathsf{app}_{q_j}(z', y'))))) \\
\mathsf{solvable}_P \quad &:= \quad \exists s \,(\mathsf{construction}_P(s) \wedge \exists x \,(\mathsf{in}(x, x, s) \wedge \neg \mathsf{eps}(x) \wedge \mathsf{finite}(x)))
\end{aligned}$$

Lemma 4.3 *For any predicate* $\mathsf{in}(x, y, z)$*, if* $\mathsf{solvable}_P$ *is valid then the instance P of the Post Correspondence Problem is solvable.*

Proof. Let σ be a fixed value for s such that $\mathsf{solvable}_P$ holds. In particular, $\mathsf{construction}_P(\sigma)$ holds. We can show for all finite trees τ, τ' satisfying $\mathsf{in}(\tau, \tau', \sigma)$ that there exists a set containing $(\bar{\gamma}(\tau), \bar{\gamma}(\tau'))$ which is P-constructed from $(\varepsilon, \varepsilon)$. The proof is by induction on $|\bar{\gamma}(\tau)| + |\bar{\gamma}(\tau')|$. □

Lemma 4.4 *There is a predicate* $\mathsf{in}(x, y, z)$ *such that if the instance P of the Post Correspondence Problem is solvable then* $\mathsf{solvable}_P$ *is valid.*

Proof. The crux of the proof is to define

1. for any sequence of pairs of words $\sigma = ((v_i, w_i))_{i=1,\dots,n}$ a feature tree $\rho(\sigma)$
2. a predicate $\mathsf{in}(y_l, y_r, x)$

such that $\mathsf{in}(\tau_l, \tau_r, \rho(\sigma))$ holds iff $\tau_l = \gamma(v_i)$ and $\tau_r = \gamma(w_i)$ for some $1 \leq i \leq n$. There is, however, no need to define a formula expressing that a feature tree is the encoding of a sequence of words.

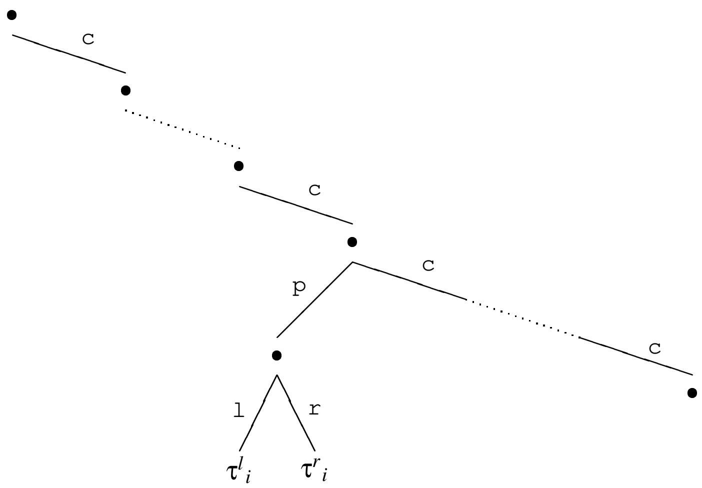


Fig. 3: A possible value for x' such that $\textsf{one-branch}(x,x')$, where x is as in Figure 2(b).

Since we know already how to encode words as trees, we now have to define an appropriate encoding of an arbitrary set of pairs of trees as a feature tree, together with a corresponding formula in. The representation of a sequence $((\tau^l{}_i, \tau^r{}_i))_{i=1,\ldots,n}$ is given in Figure 2(b).

We define, for any formula φ, a formula $\mu!x\,\varphi$ expressing that x denotes the *smallest* element satisfying φ. This formula is stronger than $\mu x\,\varphi$ in that it requires the existence of a smallest tree satisfying φ in addition:

$$\mu!x\,\varphi := \varphi \wedge \forall y\,(\varphi[y/x] \rightarrow x \leq y)$$

If x denotes a tree as given in Figure 2(b), then the formula $\textsf{one-branch}(x,x')$ given below expresses that x' denotes a tree as given in Figure 3.

$$\begin{array}{rcl} \textsf{one-branch}(x,x') & := & \exists x_c\,(\nu x_c\,(\textsf{string-c}(x_c) \wedge x_c \leq x) \\ & & \wedge x_c < x' \leq x \\ & & \wedge \nu x'\,(\exists z\,(\mu! z\,(x_c < z \leq x')))) \end{array}$$

In this formula, x' is smaller than x but is strictly greater than the $\mathtt{c}$-spine x_c of x. The tree x' can have only one of the $\mathtt{p}$-edges of x since the set of trees between x_c and x' must have a smallest element. By the maximality of x', the tree x' contains x_c plus exactly one of the subtrees of x starting with a $\mathtt{p}$-edge (see Figure 3).

The following formula $\textsf{select}(\tau^l, \tau^r, \sigma)$, where σ is as in Figure 3, expresses that τ^l is the tree $\tau^l{}_i$ and τ^r is the tree $\tau^r{}_i$:

$$\begin{array}{rcl} \textsf{select}(y_l, y_r, x') & := & \exists x''\,(\mu x''(x' \leq x'' \wedge \exists x'''\,(x''[c]x''' \wedge x''' \leq x'')) \\ & & \wedge \exists z\,(x''[\mathtt{p}]z \wedge z[\mathtt{l}]y_l \wedge z[\mathtt{r}]y_r)) \end{array}$$

From a tree σ' as given in Figure 3, we get the tree σ'' (denoted by x'') containing at all nodes $\mathtt{c}^j$ with $j \leq i$ a pair $(\tau^{l'}{}_j, \tau^{r'}{}_j)$ such that $\tau^l{}_i \leq \tau^{l'}{}_j$ and $\tau^r{}_i \leq \tau^{r'}{}_j$ (by Lemma 3.1). By the minimality of σ'' we get that $\tau^l{}_i = \tau^{l'}{}_j$ and $\tau^r{}_i = \tau^{r'}{}_j$ for all $j \leq i$, hence in particular for $j = 0$ (see Figure 4). Combination of the two formulas yields

$$\textsf{in}(y_l, y_r, x) \quad := \quad \exists x'(\textsf{one-branch}(x,x') \wedge \textsf{select}(y_l, y_r, x'))$$

Now, it is easy to verify the conditions announced at the beginning of the proof. $\square$

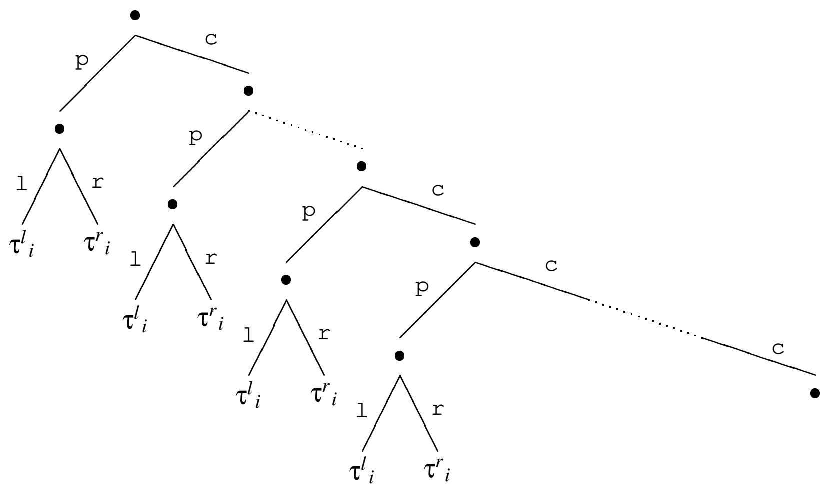


Fig. 4: The value of x' in the formula $\mathsf{select}(y_l, y_r, x)$ where x is as in Figure 3.

Note that this proof did not make use of the fact that the feature trees considered here are partial. The proof of Theorem 4.1 transfers immediately to the structures of *completely labeled* trees (both in the case of finite and of arbitrary trees), where a tree (D,L) is called *completely labeled* if L is a total function with domain D. In this case, the trees depicted in Figures 2(b), 3 and 4 have to be completed by giving some label to the nodes $\bullet$.

5 Entailment with Existential Quantifiers

In [17] it is shown that the entailment problem of $FT_{\leq}$ with existential quantifiers $\varphi \models \exists\bar{x}\varphi'$ is decidable, PSPACE-hard in the case of infinite trees and coNP-hard in the case of finite trees. We settle the precise complexity of this entailment problem in both cases.

Theorem 5.1 *Entailment of $FT_{\leq}$ with existential quantification $\varphi \models \exists\bar{x}\varphi'$ is PSPACE-complete for both structures $FT_{\leq}$ and $FT^{fin}_{\leq}$.*

In Section 5.3 we modify the PSPACE-hardness proof given in [17] for the case of infinite trees such that it proves PSPACE-hardness for both cases (Theorem 5.2). In particular, we show that we can encode the Kleene-star operator without need for infinite trees. Containment in PSPACE is shown (Theorem 5.9) by reducing in polynomial time the entailment problem to an inclusion problem between the languages accepted by nondeterministic finite state automata (NFA). Language equivalence for NFA (and hence inclusion, since $A \subseteq B \leftrightarrow B = A \cup B$) is known to be PSPACE-complete if the alphabet contains at least two distinct symbols [9].

5.1 Path Constraints

We characterize existential $FT_{\leq}$ formulas $\exists\bar{x}\varphi$ by equivalent sets of path constraints (where sets are interpreted as conjunctions). Feature path constraints for FT have been introduced in [29] and have been used in [4] for a quantifier elimination procedure for FT. The abstract syntax of *path constraints* ψ is defined as follows, where $\pi, \pi' \in \mathcal{F}^*$ and $a \in \mathcal{L}$:

$$\psi ::= \quad x[\pi]{\downarrow} \mid a(x[\pi]) \mid x?[\pi]{\sim}a \mid x?[\pi]{\leq}y?[\pi'] \mid x?[\pi]{\sim}y?[\pi']$$

$$
\begin{array}{l}
\exists y \exists y' \exists z \exists z' \\
x \leq y \\
\;\; f \,\big| \\
\;\; y' \leq \;\; z \\
\qquad\; g \,\big| \\
\qquad\;\; a(z')
\end{array}
\quad \models\!\mid \quad x?[fg]{\sim}a \quad \models\!\mid \quad
\begin{array}{l}
\exists y \exists y' \exists z \exists z' \\
\;\; x \;\; \leq y \\
\qquad\; f \,\big| \\
\;\; z \;\; \leq y' \\
g \,\big| \\
a(z')
\end{array}
$$

Fig. 5: Graphical Presentation of Example 4

The semantics of path constraints is given by extension of the structure $FT_{\leq}$ through the following predicates, which are defined on basis of the subtree selection function $\tau[\pi]$ introduced above.

$$
\begin{array}{lll}
\tau[\pi]{\downarrow} & \text{iff} & \pi \in D_\tau \\
a(\tau[\pi]) & \text{iff} & (\pi, a) \in L_\tau \\
\tau?[\pi]{\sim}a & \text{iff} & \pi \in D_\tau \text{ implies } \tau[\pi]{\sim}a \\
\tau?[\pi]{\leq}\tau'?[\pi'] & \text{iff} & \pi \in D_\tau \text{ and } \pi' \in D_{\tau'} \text{ imply } \tau[\pi] \leq \tau'[\pi'] \\
\tau?[\pi]{\sim}\tau'?[\pi'] & \text{iff} & \pi \in D_\tau \text{ and } \pi' \in D_{\tau'} \text{ imply } \tau[\pi]{\sim}\tau'[\pi']
\end{array}
$$

In the Section 5.2, we use path constraints for presenting typical examples of entailment judgements. Path constraints are also helpful for proving PSPACE-hardness in Section 5.3. In Section 5.5 we will construct a finite automaton that accepts all path constraints ψ entailed by $\exists\bar{x}\varphi$ and thereby reduce the entailment problem with existential quantification to the inclusion problem of finite automata.

5.2 Examples

A major difficulty in testing entailment with existential quantifiers is that there exist many equivalent $FT_{\leq}$ constraints of quite distinct syntactic shape. This makes it very difficult (if not impossible) to apply a standard technique for deciding entailment, which performs a comparison of constraints in some syntactic normal form [2, 31, 18]. In this section we present some examples showing the difficulties of deciding entailment statement. We will come back to some of these examples in Section 5.5 to illustrate our solution.

We start with a rather simple case:

Example 3 *The formula* $\exists y(x{\leq}y \wedge a(y))$ *is equivalent to* $x?[\varepsilon]{\sim}a$ *which is equivalent to* $\exists y \exists z(x{\leq}y \wedge z{\leq}y \wedge a(z))$.

The next example of equivalent constraints with distinct syntactic shape is more complex.

Example 4 *(see Figure 5) Both of the following formulas are equivalent to* $x?[fg]{\sim}a$ *and hence equivalent to each other:*

$$
\begin{array}{rl}
& \exists y \exists y' \exists z \exists z' \, (x{\leq}y \wedge y[f]y' \wedge y'{\leq}z \wedge z[g]z' \wedge a(z')) \\
\models\!\mid & \exists y \exists y' \exists z \exists z' \, (x{\leq}y \wedge y[f]y' \wedge z{\leq}y' \wedge z[g]z' \wedge a(z'))
\end{array}
$$

In the next example, a constraint is given that entails $x?[fg]{\sim}a$ for all a. Note that this constraint thus also entails the constraints given in the previous example.

Example 5 *(see Figure 6) If* $b \neq c$ *then for all* a *the judgement*

$$
\left.
\begin{array}{l}
x[f]x' \wedge x'{\leq}x'' \wedge x''[g]x''' \wedge b(x''') \wedge \\
x{\leq}y \wedge y[f]y' \wedge y'{\leq}u \wedge u{\geq}z' \wedge z'[g]z'' \wedge c(z'')
\end{array}
\right\} \models x?[fg]{\sim}a
$$

$$\begin{array}{ccccccc} x & & \leq & & y & & \\ f\,\big| & & & & f\,\big| & & \\ x' \leq & x'' & & & y' \sim & z' & \\ & g\,\big| & & & & g\,\big| & \\ & b(x''') & & & & c(z'') & \end{array} \qquad \models \quad x?[fg]{\sim}a$$

Fig. 6: Graphical Presentation of Example 5

$$\begin{array}{ccccccccc} & & u & \leq & x & \leq & v & & \\ & & f\,\big| & & & & f\,\big| & & \\ y & \leq & u' & & & & v' & \leq & y \end{array} \qquad \models \quad \begin{array}{c} x \\ f\,\big| \\ y \end{array}$$

Fig. 7: Graphical Representation of Example 6

holds. In other words, if α *is a solution of the constraint displayed on the left hand side and if* $fg \in D_{\alpha(x)}$ *then the subtree of* $\alpha(x)$ *at* fg *is compatible with any label a, and hence is unlabeled.*

Example 6 *(see Figure 7) The following situation illustrates a non-trivial example for entailment of selection constraints without existential quantifiers.*

$$(y \leq u' \wedge u[f]u' \wedge u \leq x) \wedge (x \leq v \wedge v[f]v' \wedge v' \leq y) \quad \models \quad x[f]y$$

The right-hand side $x[f]y$ *is equivalent to the conjunction* $(y?[\varepsilon]{\leq}x?[f] \wedge x[f]{\downarrow}) \wedge (x?[f]{\leq}y?[\varepsilon])$ *of path constraints which are entailed by the first and second part of the left-hand side, respectively.*

5.3 Entailment is PSPACE-hard

In this section we show how the PSPACE completeness proof of [17] can be modified such that it applies to the structure of finite feature trees as well. The formulas used in the earlier proof require the existence $x[\pi]{\downarrow}$ of all paths π in some regular language R; every solution of the formula for an infinite language R has to map x to an infinite tree. Compared to this earlier proof, the trick is here to use conditional path constraints which may constrain infinitely many paths without requiring their existence.

Theorem 5.2 *The entailment problem for existentially quantified* $FT_{\leq}$*-constraints is PSPACE-hard in both the finite and the infinite tree case.*

This follows from Proposition 5.6 (see below), which claims a polynomial reduction of the inclusion problem between regular languages over the alphabet $\mathcal{F}$ to an entailment problem between two existential $FT_{\leq}$ formulas. Notice that we have assumed $\mathcal{F}$ to contain at least two features.

Our PSPACE-hardness proof is based on the fact that a satisfiable ordering constraint φ may entail an infinite conjunction of path constraints, even in case of finite trees:

Example 7

1. *for all* $n : x[f]y \wedge y{\leq}x \wedge a(x) \models x?[f^n]{\sim}a$.

2. *for all* $n,m : x[f]y \wedge y \leq x \models x?[f^{m+n}] \leq x?[f^n]$.
3. *for all* $\pi \in \{f,g\}^* : x \leq x' \wedge x[f]x' \wedge x[g]x' \models x?[\pi] \leq x?[\varepsilon]$.

For this reason the entailment problem for $FT_{\leq}^{fin}$ does not necessarily reduce to an inclusion problem between finite regular languages (which is decidable in coNP [9]). We fix a finite subset $F \subseteq \mathcal{F}$ of features and consider regular expressions of the following form:

$$R \quad ::= \quad \varepsilon \;\mid\; f \;\mid\; R^* \;\mid\; R_1 \cup R_2 \;\mid\; R_1 R_2 \qquad (\text{where } f \in F)$$

For encoding a regular expression R the main idea is to define an existential formula $\Theta(x,R,y)$ for fresh variables x,y such that $\Theta(x,R,y)$ is equivalent to $\bigwedge_{\pi \in \mathcal{L}(R)} x?[\pi] \leq y?[\varepsilon]$. Once this is done, it will follow immediately that $\mathcal{L}(R') \subseteq L(R)$ iff $\Theta(x,R,y) \models \Theta(x,R',y)$. It is not obvious, however, how to define such a formula. The reader might notice, that a naive definition of $\Theta(x,R,y)$ yields some unintended compatibility relations to be entailed too. Hence, we have to refine our main idea.

We define the formula $\mathsf{com}_{F^*}(x)$ expressing that all subtrees of x reachable via an F-path are compatible with each other, i.e. they have a common upper bound:

$$\mathsf{com}_{F^*}(x) := \exists y (x \leq y \wedge \bigwedge_{f \in F} \exists y' (y[f]y' \wedge y' \leq y))$$

Lemma 5.3 (Comon upper bound) $\mathsf{com}_{F^*}(x) \models\!\mid \exists y \forall \pi \in F^* \; x?[\pi] \leq y?[\varepsilon]$.

For encoding a regular expression R, a refined idea is to define an existential formula $\Theta(x,R,y)$ such that $\Theta(x,R,y)$ is equivalent to $\mathsf{com}_{F^*}(x) \wedge \bigwedge_{\pi \in \mathcal{L}(R)} x?[\pi] \leq y?[\varepsilon]$. We define for all regular expressions R over F and variables x and y, the existential formulas $\Theta(x,R,y)$ and $\Theta'(x,R,y)$ recursively as follows.

$$\begin{array}{lcl}
\Theta(x,R,y) & = & \mathsf{com}_{F^*}(x) \wedge \Theta'(x,R,y) \\
\Theta'(x,\varepsilon,y) & = & x \leq y \\
\Theta'(x,f,y) & = & \exists z (x \leq z \wedge z[f]y) \\
\Theta'(x,R_1 \cup R_2,y) & = & \Theta'(x,R_1,y) \wedge \Theta'(x,R_2,y) \\
\Theta'(x,R_1 R_2,y) & = & \exists z (\Theta'(x,R_1,z) \wedge \Theta(z,R_2,y)) \\
\Theta'(x,R^*,y) & = & \exists z (x \leq z \wedge \Theta'(z,R,z) \wedge z \leq y)
\end{array}$$

Apparently, $\Theta(x,R,y)$ has size linear in the size of R.

Lemma 5.4 *For all regular expressions R*

$$\mathsf{com}_{F^*}(x) \quad \models \quad \Theta'(x,R,y) \leftrightarrow \bigwedge_{\pi \in \mathcal{L}(R)} x?[\pi] \leq y?[\varepsilon]$$

Proof. We proceed by induction on R.

ε: $\Theta'(x,\varepsilon,y) = x \leq y \leftrightarrow x?[\varepsilon] \leq y?[\varepsilon] = \bigwedge_{\pi \in \mathcal{L}(\varepsilon)} x?[\pi] \leq y?[\varepsilon]$.

f: $\Theta'(x,f,y) = \exists z (x \leq z \wedge z[f]y) \leftrightarrow x?[f] \leq y?[\varepsilon] = \bigwedge_{\pi \in \mathcal{L}(f)} x?[\pi] \leq y?[\varepsilon]$.

$R_1 \cup R_2$: By induction hypothesis $\mathsf{com}_{F^*}(x)$ entails the equivalences $\Theta'(x,R_1,y) \leftrightarrow \bigwedge_{\pi \in \mathcal{L}(R_1)} x?[\pi] \leq y?[\varepsilon]$ and $\Theta'(x,R_2,y) \leftrightarrow \bigwedge_{\pi \in \mathcal{L}(R_2)} x?[\pi] \leq y?[\varepsilon]$. Hence, $\mathsf{com}_{F^*}(x)$ entails $\Theta'(x,R_1 \cup R_2,y) \leftrightarrow \bigwedge_{\pi \in \mathcal{L}(R_1 \cup R_2)} x?[\pi] \leq y?[\varepsilon]$ also.

$R_1 R_2$: By definition $\Theta'(x,R_1 R_2,y) = \exists z (\Theta'(x,R_1,z) \wedge \mathsf{com}_{F^*}(z) \wedge \Theta'(z,R_2,y))$. By induction hypothesis, $\mathsf{com}_{F^*}(x)$ entails $\Theta'(x,R_1,z) \leftrightarrow \bigwedge_{\pi_1 \in \mathcal{L}(R_1)} x?[\pi_1] \leq z?[\varepsilon]$ and

$\mathsf{com}_{F^*}(z)$ entails $\Theta'(z,R_2,y) \leftrightarrow \bigwedge_{\pi_2\in\mathcal{L}(R_2)} z?[\pi_2]\leq y?[\varepsilon]$. Hence, $\mathsf{com}_{F^*}(x)$ entails that $\Theta'(x,R_1R_2,y)$ is equivalent to (1):

$$\exists z \left(\bigwedge_{\pi_1\in\mathcal{L}(R_1)} x?[\pi_1]\leq z?[\varepsilon] \quad \wedge \mathsf{com}_{F^*}(z) \wedge \bigwedge_{\pi_2\in\mathcal{L}(R_2)} z?[\pi_2]\leq y?[\varepsilon] \right) \tag{1}$$

It remains to show that $\mathsf{com}_{F^*}(x)$ entails the equivalence between (1) and (2):

$$\bigwedge_{\pi\in\mathcal{L}(R_1R_2)} x?[\pi]\leq y?[\varepsilon] \tag{2}$$

Since (1) obviously entails (2), it is sufficient to prove the validity of $\mathsf{com}_{F^*}(x) \models (2) \rightarrow (1)$. Let α be an $FT_{\leq}$-valuation which satisfies both $\mathsf{com}_{F^*}(x)$ and (2). We define a tree τ such that $\alpha, z \mapsto \tau$ satisfies the matrix of (1). For this definition we use a least upper bound operator on feature trees denoted by $\sqcup$:

$$\tau \quad = \quad \bigsqcup_{\pi_1\in\mathcal{L}(R_1)\cap D_{\alpha(x)}} \alpha(x)[\pi_1]$$

Since α solves $\mathsf{com}_{F^*}(x)$ there exists an upper bound of $\{\alpha(x)[\pi] \mid \pi \in F^*\}$ as stated by Lemma 5.3 and thus the least upper bound τ exists. We next demonstrate that $\alpha, z \mapsto \tau$ satisfies the matrix of (1). The definition of τ yields $\alpha(x)[\pi_1]\leq\tau$ for all $\pi_1 \in \mathcal{L}(R_1)\cap D_{\alpha(x)}$, i.e. the variable assignment $\alpha, z\mapsto\tau$ satisfies the first conjunction in (1). From $\mathsf{com}_{F^*}(\alpha(x))$ it follows that $\mathsf{com}_{F^*}(\tau)$ holds, i.e. $\alpha, z\mapsto\tau$ satisfies $\mathsf{com}_{F^*}(z)$. Furthermore, all $\pi_2 \in D_\tau$ satisfy: $\tau[\pi_2] = \bigsqcup_{\pi_1\in\mathcal{L}(R_1)\cap D_{\alpha(x)}} \alpha(x)[\pi_1\pi_2]$. Since α is a solution of (2), $\alpha(x)[\pi_1\pi_2]\leq\alpha(y)$ is satisfied by all $\pi_2 \in \mathcal{L}(R_2)$. Thus $\tau[\pi_2]\leq\alpha(y)$ is valid for all $\pi_2 \in \mathcal{L}(R_2)$, i.e. $\alpha, z\mapsto\tau$ satisfies $\bigwedge_{\pi_2\in\mathcal{L}(R_2)} z?[\pi_2]\leq y?[\varepsilon]$, the remaining conjunct in (1).

R^*: By definition $\Theta'(x,R^*,y) = \exists z(\mathsf{com}_{F^*}(z)\wedge x\leq z\wedge\Theta'(z,R,z)\wedge z\leq y)$. The induction assumption yields that $\mathsf{com}_{F^*}(z)$ entails $\Theta'(z,R,z) \leftrightarrow \bigwedge_{\pi\in\mathcal{L}(R)} z?[\pi]\leq z?[\varepsilon]$. Hence, $\mathsf{com}_{F^*}(x)$ entails that $\Theta'(x,R^*,y)$ is equivalent to (3):

$$\exists z \left(\mathsf{com}_{F^*}(z)\wedge x\leq z\wedge \bigwedge_{\pi\in\mathcal{L}(R)} z?[\pi]\leq z?[\varepsilon] \qquad \wedge z\leq y \right) \tag{3}$$

It remains to show that $\mathsf{com}_{F^*}(z)$ entails the equivalence between (3) and (4):

$$\bigwedge_{\pi\in\mathcal{L}(R^*)} x?[\pi]\leq y?[\varepsilon] \tag{4}$$

In order to show the non-trivial implication, we assume an $FT_{\leq}$-valuation α which satisfies both $\mathsf{com}_{F^*}(x)$ and (4). We define a tree τ such that $\alpha, z\mapsto\tau$ satisfies the matrix of (3) as follows:

$$\tau \quad = \quad \bigsqcup_{\pi\in\mathcal{L}(R^*)\cap D_{\alpha(x)}} \alpha(x)[\pi]$$

Note that τ is well-defined for the same reason as in the preceeding case. Our assumptions on the choice of α yields: $\mathsf{com}_{F^*}(\tau)$, $\alpha(x)\leq\tau$ (since $\varepsilon\in\mathcal{L}(R^*)$) and $\tau\leq\alpha(y)$. In order to show that $\alpha, z\mapsto\tau$ is a solution of (3) it remains to prove $\tau[\pi']\leq\tau$ for all $\pi' \in \mathcal{L}(R^*)\cap D_\tau$:

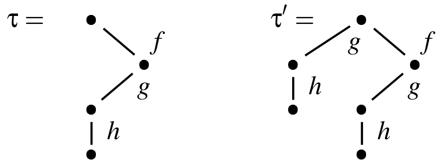
$$\tau[\pi'] \quad = \bigsqcup_{\pi\in\mathcal{L}(R^*)\cap D_{\alpha(x)}} \alpha(x)[\pi][\pi'] \quad \leq \bigsqcup_{\pi''\in\mathcal{L}(R^*)\cap D_{\alpha(x)}} \alpha(x)[\pi''] \quad = \quad \tau$$

□

Lemma 5.5 *For all regular expressions R_1 and R_2*

$$\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2) \quad \textit{iff} \quad \mathsf{com}_{F^*}(x) \models \underbrace{\bigwedge_{\pi\in\mathcal{L}(R_2)} x?[\pi]{\leq}y?[\varepsilon]}_{(*)} \rightarrow \underbrace{\bigwedge_{\pi\in\mathcal{L}(R_1)} x?[\pi]{\leq}y?[\varepsilon]}_{(**)}$$

Proof. The implication from the left to the right is trivial since $(**)$ is a sub-conjunction of $(*)$ if $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$. For the other direction, we assume $\mathcal{L}(R_1) \not\subseteq \mathcal{L}(R_2)$ and show how to contradict the entailment judgment to the right. We fix a word $\pi \in F^*$ from $\mathcal{L}(R_1) - \mathcal{L}(R_2)$ and a new feature $h \in \mathcal{F} - F$ (which exists since F is finite whereas $\mathcal{F}$ is not). We construct values τ for x and τ' for y such that $(*)$ is satisfied but $(**)$ is not. Both trees are completely unlabeled; hence $\mathsf{com}_{F^*}(\tau)$ holds. We define the domain D_τ to be the prefix closure of the word πh and the domain $D_{\tau'}$ to be the suffix closure of D_τ with the exception of the word h. For illustration, we display the trees τ and τ' for the word $\pi = fg$ below:



It is easy to check that $[x \mapsto \tau, y \mapsto \tau']$ satisfies $(*)$ but not $(**)$ since $\pi \in \mathcal{L}(R_1) - \mathcal{L}(R_2)$ and $h \in D_{\tau[\pi]}$ but $h \notin D_{\tau'}$. □

Proposition 5.6 *For all variables x,y and for every pair of regular expressions R_1 and R_2: $\Theta(x,R_1,y) \models \Theta(x,R_2,y)$ is equivalent to $\mathcal{L}(R_2) \subseteq \mathcal{L}(R_1)$.*

Proof. This follows from Lemmas 5.4 and 5.5. □

5.4 Satisfiability Test

In this section we recall the satisfiability test for $FT_{\leq}$ introduced in [18], which we will also need as a preprocessing step in our entailment test in Section 5.5. Clearly, satisfiability (and hence entailment) depends on the choice of finite or infinite trees. For instance, $x[f]x$ is unsatisfiable in $FT^{fin}_{\leq}$ but satisfiable in $FT_{\leq}$.

Let an *extended constraint* be a conjunction of constraints φ and (atomic) compatibility constraints $x{\sim}y$. From now on, we will only deal with extended constraints and freely call them constraints for simplicity.

In the case of infinite trees, we say that an (extended) constraint φ is F-*closed* if it satisfies the following properties for all x,y,z,x',y',f,a,b.

F1.1	$x{\leq}x \in \varphi$	if	$x \in \mathcal{V}(\varphi)$
F1.2	$x{\leq}z \in \varphi$	if	$x{\leq}y \in \varphi$ and $y{\leq}z \in \varphi$
F2	$x'{\leq}y' \in \varphi$	if	$x[f]x' \in \varphi,\ x{\leq}y \in \varphi,\ y[f]y' \in \varphi$
F3.1	$x{\sim}y \in \varphi$	if	$x{\leq}y \in \varphi$
F3.2	$x{\sim}z \in \varphi$	if	$x{\leq}y \in \varphi$ and $y{\sim}z \in \varphi$
F3.3	$x{\sim}y \in \varphi$	if	$y{\sim}x \in \varphi$
F4	$x'{\sim}y' \in \varphi$	if	$x[f]x' \in \varphi,\ x{\sim}y \in \varphi,\ y[f]y' \in \varphi$
F5	$a = b$	if	$a(x) \in \varphi,\ x{\sim}y \in \varphi,\ b(y) \in \varphi$

The rules of F1 and F2 require that φ is closed with respect to reflexivity, transitivity, and decomposition of $\leq$. The rules in F3 and F4 require that φ contains all compatibility constraints that it entails (this is proved in [18]), and F5 requires φ to be clash-free.

In the case of finite trees, we say that a constraint φ is F-*closed* if it satisfies F1-F5 and the additional *occurs check property* F6 for all $n \geq 1, x_1, \ldots, x_{n+1}, y_1, \ldots y_n, f_1, \ldots, f_n$:

$$\text{F6} \quad x_1 \leq x_{n+1} \notin \varphi \quad \text{if} \quad x_i[f_i]y_i \wedge x_{i+1} \leq y_i \in \varphi \quad \text{for all } 1 \leq i \leq n$$

The following result is proved in [18] (Theorem 1 and Proposition 4). It holds in both cases, for finite trees and for possibly infinite trees, but with the respective notion of F-closedness.

Proposition 5.7 *There exists a cubic time algorithm that, given a constraint φ, computes an* F-*closed constraint containing φ or proves its unsatisfiability. Every* F-*closed constraint is satisfiable.*

5.5 An Automaton for Path Constraints

In this section we show that for every F-closed constraint φ there is a non-deterministic automaton $\mathcal{A}_\varphi$ of size polynominal in the size of φ which accepts the set of all path constraints which are entailed by φ and which mentions only symbols from a fixed set of variables, labels, and features. Note that F-closedness is a necessary assumption for our automaton construction. Note also that the automaton does not differ in the case of finite and infinite trees, only the assumed version of F-closedness differs.

The algorithm of Dörre [7] can be seen in this perspective. There, the non-satisfiability of a (in some sense normalised) weak subsumption constraint φ was equivalent to the fact that two labeling path constraints $a(x[\pi])$ and $b(x[\pi])$ for different label symbols a and b are entailed by φ, which could be checked by inspection of the automaton that describes all the labeling path constraints entailed by φ.

5.5.1 Path Constraints as Words

The automaton accepts words $\langle\psi\rangle$ associated with a path constraint ψ over some finite subalphabet of $\mathcal{F} \cup \mathcal{L} \cup \mathcal{V} \cup \{\leq, \sim, \downarrow, ?, [,], (,)\}$. In first approximation, let $\langle\psi\rangle$ be the *concrete syntax* of ψ. There is however a serious problem with recognizing the concrete syntax of entailed path constraints:

Example 8 *1. The set of words representing a path constraint entailed by $x \leq x$ is not regular (when restricted to the variables in $x \leq x$):*

$$\{\langle\psi\rangle \mid x \leq x \models \psi\} = \{x?[\pi] \sim x?[\pi] \mid \pi \in \mathcal{F}^*\} \cup \{x?[\pi] \leq x?[\pi] \mid \pi \in \mathcal{F}^*\}$$

2. The set of words representing a path ordering constraint entailed by $\exists y(x[f]y \wedge x \leq y)$ is not regular:

$$\begin{array}{rcl} \{\langle\psi\rangle \mid \exists y(x[f]y \wedge x \leq y) \models \psi\} & = & \{x?[f^m] \leq x?[f^n] \mid 0 \leq m \leq n\} \\ & \cup & \{x[f^n]\downarrow \mid n \geq 0\} \\ & \cup & \{x?[f^m] \sim x?[f^n] \mid m, n \geq 0\} \end{array}$$

We therefore have to alter the definition of $\langle\varphi\rangle$ slightly but fundamentally. The trick is to "factor out" the maximal common suffix of the two paths in a path constraint of the form $x?[\pi_1] \sim y?[\pi_2]$. More exactly, we add the symbol # to the alphabet and alter the definition of $\langle\psi\rangle$ such that:

$$\begin{array}{rcl} \langle x?[\pi_1] \sim y?[\pi_2]\rangle & = & x?[\pi] \sim y?[\pi']\#\pi'' \\ \langle x?[\pi_1] \leq y?[\pi_2]\rangle & = & x?[\pi] \leq y?[\pi']\#\pi'' \end{array}$$

where π'' is the longest common suffix of π_1 and π_2 such that $\pi_1 = \pi\pi''$ and $\pi_2 = \pi'\pi''$. Hence, either one of π or π' is the empty path, or π and π' end with distinct feature symbols. This solves the regularity problem of Example 8, *i.e.*, the following sets are regular:

$$\begin{array}{rcl}
\{\langle\psi\rangle \mid x{\leq}x \models \psi\} & = & \{x?[\varepsilon]{\sim}x?[\varepsilon]\#\pi \mid \pi \in \mathcal{F}^*\} \\
& \cup & \{x?[\varepsilon]{\leq}x?[\varepsilon]\#\pi \mid \pi \in \mathcal{F}^*\} \\
\{\langle\psi\rangle \mid \exists y(x[f]y \wedge x{\leq}y) \models \psi\} & = & \{x?[\varepsilon]{\leq}x?[f^n]\#f^m \mid n,m \geq 0\} \\
& \cup & \{x[f^n]{\downarrow} \mid n \geq 0\} \\
& \cup & \{x?[f^n]{\sim}x?[\varepsilon]\#f^m \mid n,m \geq 0\} \\
& \cup & \{x?[\varepsilon]{\sim}x?[f^n]\#f^m \mid n,m \geq 0\}
\end{array}$$

The definition of $\langle\psi\rangle$ also adjusts some simple but tedious regularity problems raised by the validity of the following entailment judgement:

$$x?[\pi]{\sim}y?[\pi'] \models x?[\pi\pi'']{\sim}y?[\pi'\pi'']$$

Example 9 *The set* $\{\langle\psi\rangle \mid x?[gf]{\sim}y?[ff] \models \psi\}$ *restricted to words with features* f,g *and variables* x,y *is regular:*

$$\begin{array}{rl}
& \{x?[g]{\sim}y?[f]\#f\pi \mid \pi \in \{f,g\}^*\} \\
\cup & \{z?[\varepsilon]{\sim}z?[\varepsilon]\#\pi \mid z \in \{x,y\}, \pi \in \{f,g\}^*\} \\
\cup & \{z?[\varepsilon]{\leq}z?[\varepsilon]\#\pi \mid z \in \{x,y\}, \pi \in \{f,g\}^*\}
\end{array}$$

5.5.2 The Alphabet of the Automaton

For each constraint φ we will define a non-deterministic finite automaton $\mathcal{A}_\varphi$ whose alphabet is the set:

$$\mathcal{F}(\varphi) \cup \mathcal{L}(\varphi) \cup \mathcal{V}(\varphi) \cup \{\leq, \sim, ?, [,], (,), \#\}.$$

Given a sequence of variables $\bar{x}$, we will also define another automaton $\mathcal{A}_\varphi^{\bar{x}}$ for the existential formula $\exists\bar{x}\varphi$, which is obtained from $\mathcal{A}_\varphi$ by removing the local variables in $\bar{x}$ from the alphabet, i.e. by removing all transitions labeled with a symbol from $\bar{x}$. Note that the local variables in $\bar{x}$ matter for the definition of the states (but not the alphabet) of $\mathcal{A}_\varphi^{\bar{x}}$ if they occur in $\mathcal{V}(\varphi)$.

To solve an entailment problem of the form $\varphi \models \exists\bar{x}\varphi'$ we construct the automata $\mathcal{A}_\varphi$ and $\mathcal{A}_{\varphi'}^{\bar{x}}$ and test for language inclusion. In order to avoid that $\mathcal{A}_{\varphi'}^{\bar{x}}$ accepts tautological constraints not accepted by $\mathcal{A}_\varphi$ we will require in Proposition 5.10 that $\mathcal{F}(\varphi') \subseteq \mathcal{F}(\varphi)$ and $\mathcal{V}(\exists\bar{x}\varphi') \subseteq \mathcal{V}(\varphi)$, which can be imposed w.l.o.g. Furthermore, we assume throughout the paper that bound variables are renamed apart, *i.e.* when considering an entailment problem $\varphi \models \exists\bar{x}\varphi'$ we assume $\{\bar{x}\} \cap \mathcal{V}(\varphi) = \emptyset$.

Every automaton A_φ (and thereby $A_\varphi^{\bar{x}}$) falls into five parts (sharing only the initial state q_s and the accepting state q_f), corresponding to the five kinds of path constraints.

The construction of the automaton $\mathcal{A}_\varphi$ is given in Figures 8, 9, 10 and 11. It is completely spelled out except for one additional symmetry (rule 6) which can be expressed through a dozen of further transitions. In the rest of this section we explain this construction.

5.5.3 Constraints as Graphs

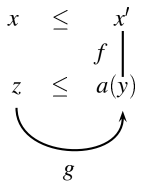
Our construction of the automaton is motivated by considering constraints as graphs. For instance, the constraint

$$x{\leq}x' \wedge x'[f]y \wedge a(y) \wedge z{\leq}y \wedge z[g]y$$

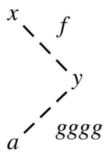
1.1	q_s	$\xrightarrow{x[}$	$/x$	
1.2	$/x$	$\xrightarrow{\varepsilon}$	$/y$	$x{\geq}y{\in}\varphi$
1.3	$/x$	$\xrightarrow{f}$	$/y$	$x[f]y{\in}\varphi$
1.4	$/x$	$\xrightarrow{]\downarrow}$	q_f	
2.1	q_s	$\xrightarrow{a(x[}$	$/x{:}a$	
2.2	$/x{:}a$	$\xrightarrow{\varepsilon}$	$/y{:}a$	$x{\geq}y{\in}\varphi$
2.3	$/x{:}a$	$\xrightarrow{f}$	$/y{:}a$	$x[f]y{\in}\varphi$
2.4	$/x{:}a$	$\xrightarrow{])}$	q_f	$a(x){\in}\varphi$

Fig. 8: The sections of the automaton $\mathcal{A}_\varphi$ for path constraints $x[\pi]{\downarrow}$ and $a(x[\pi])$.

can be depicted as the following graph, where variables are represented as nodes.



Intuitively, when the automaton $\mathcal{A}_\varphi$ accepts a word $\langle\psi\rangle$ it traverses the constraint graph associated with φ where ψ is associated a certain traversal pattern. We will depict such traversal patterns graphically; for instance, the above constraint entails $x?[fgggg]{\sim}a$ and its associated graph allows for the following traversal:



In these pictures, the horizontal dimension corresponds to the ordering $\leq$ (left to right) and the vertical one corresponds to feature selection (top to bottom).

Path Existence and Labeling Constraints (Fig 8). The subautomaton comprising rules 1.1–1.4 recognizes all the path existence constraints $x[\pi]{\downarrow}$ entailed by φ. Analogously, the rules 2.1–2.4 serve to recognize the path labeling constraints $a(x[\pi])$ entailed by φ. The associated patterns look as follows.

Rules 2.1–2.4 differ from rules 1.1–1.4 in that its states of the form $/y{:}a$ memorize the label a read at the beginning of some input word $\langle a(x[\pi])\rangle$ (rule 2.1) in order to check it against a labeling constraint in φ (rule 2.4).

3.1	q_s	$\xrightarrow{x?[}$	$\searrow x{:}\varepsilon$	
3.2	$\searrow x{:}h$	$\xrightarrow{\varepsilon}$	$\searrow y{:}h$	$x\leq y\in\varphi$
3.3	$\searrow x{:}h$	$\xrightarrow{f}$	$\searrow y{:}f$	$x[f]y\in\varphi$
3.4	$\searrow x{:}h$	$\xrightarrow{]\leq y?[}$	$\swarrow y{:}\searrow x, h, \varepsilon$	
3.5	$\swarrow x{:}\searrow y, h, g$	$\xrightarrow{\varepsilon}$	$\swarrow x'{:}\searrow y, h, g$	$x\geq x'\in\varphi$
3.6	$\swarrow x{:}\searrow y, h, g$	$\xrightarrow{f}$	$\swarrow x'{:}\searrow y, h, f$	$x[f]x'\in\varphi$
3.7	$\swarrow x{:}\searrow y, h, g$	$\xrightarrow{]\#}$	$\searrow y \swarrow x$	$h\neq g \vee h=g=\varepsilon$
3.8	$\searrow x \swarrow y$	$\xrightarrow{\varepsilon}$	$\searrow x' \swarrow y'$	$x\leq x', y\geq y'\in\varphi,$
3.9	$\searrow x \swarrow y$	$\xrightarrow{f}$	$\searrow x' \swarrow y'$	$x[f]x', y[f]y'\in\varphi$
3.10	$\searrow x \swarrow x$	$\xrightarrow{\mathcal{F}(\varphi)^*}$	q_f	

Fig. 9: The section of the automaton $\mathcal{A}_\varphi$ for path constraints $x?[\pi_1]\leq y?[\pi_2]\#\pi_3$ which is concrete syntax for $x?[\pi_1\pi_3]\leq y?[\pi_2\pi_3]$.

Example 10 *The constraint*

$$x[f]y \wedge y\geq y' \wedge y'[g]z \wedge a(z)$$

entails $a(x[fg])$. *This constraint is accepted by the following transitions:*

$$q_s \xrightarrow{a(x[} \swarrow x{:}a \xrightarrow{f} \swarrow y{:}a \xrightarrow{\varepsilon} \swarrow y'{:}a \xrightarrow{g} \swarrow z{:}a \xrightarrow{])} q_f$$

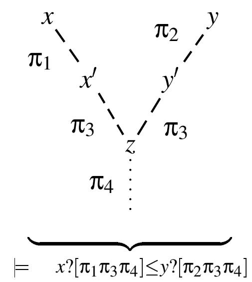
Ordering Path Constraints (Fig. 9) The next group of rules 3.1–3.10 serves to recognize constraints of the form $x?[\pi]\leq y?[\pi']$. Note that $\varphi \models x?[\pi]\leq y?[\pi']$ iff $\pi = \pi_1\pi_3\pi_4$ and $\pi' = \pi_2\pi_3\pi_4$ for some π_1, π_2, π_3, and π_4, and there exists x', y', z such that

$$\begin{aligned} \varphi &\models x?[\pi_1]\leq x'?[\varepsilon] && (5)\\ \varphi &\models x'?[\pi_3]\leq z?[\varepsilon] && (6)\\ \varphi &\models z?[\varepsilon]\leq y'?[\pi_3] && (7)\\ \varphi &\models y'?[\varepsilon]\leq y?[\pi_2] && (8)\\ & && (9) \end{aligned}$$

where we may assume that

$$\pi_1 \text{ and } \pi_2 \text{ have no common suffix except } \varepsilon \tag{10}$$

The associated graph pattern is as follows, where a dashed line indicates a paths of the constraint graph and a dotted an arbitrary path.



4.1	q_s	$\xrightarrow{x?[}$	$\searrow x \searrow x$	
4.2	$\searrow x \searrow x'$	$\xrightarrow{\varepsilon}$	$\searrow y \searrow y'$	$x \leq y, x' \leq y' \in \varphi$
4.3	$\searrow x \searrow x'$	$\xrightarrow{f}$	$\searrow y \searrow y'$	$x[f]y, x'[f]y' \in \varphi$
4.4	$\searrow x \searrow x'$	$\xrightarrow{\varepsilon}$	$\rangle y \searrow x'$	$x \sim y \in \varphi$
4.5	$\rangle x \searrow x'$	$\xrightarrow{\varepsilon}$	$\rangle y \searrow y'$	$x \geq y, x' \leq y' \in \varphi$
4.6	$\rangle x \searrow x'$	$\xrightarrow{f}$	$\rangle y \searrow y'$	$x[f]y, x'[f]y' \in \varphi$
4.7	$\rangle x \searrow x'$	$\xrightarrow{\varepsilon}$	$\rangle x \rangle y'$	$x' \sim y' \in \varphi$
4.8	$\rangle x \rangle x'$	$\xrightarrow{\varepsilon}$	$\rangle y \rangle y'$	$x \geq y, x' \geq y' \in \varphi$
4.9	$\rangle x \rangle x'$	$\xrightarrow{f}$	$\rangle y \rangle y'$	$x[f]y, x'[f]y' \in \varphi$
4.10	$\rangle x \rangle x'$	$\xrightarrow{]\sim a}$	q_f	$a(x) \in \varphi$
4.11	$\rangle x \rangle x'$	$\xrightarrow{]\sim c}$	q_f	$a(x), b(x') \in \varphi, a \neq b$

Fig. 10: The section of the automaton $\mathcal{A}_\varphi$ for path constraints $x?[\pi]{\sim}a$.

Note that $\pi_3\pi_4$ is the maximal common suffix of $\pi_1\pi_3\pi_4$ and $\pi_2\pi_3\pi_4$. Consequently, the concrete syntax of the constraint $x?[\pi_1\pi_3\pi_4] \leq y?[\pi_2\pi_3\pi_4]$ as checked by the automaton is

$$x?[\pi_1] \leq y?[\pi_2] \# \pi_3\pi_4$$

Rule 3.1 starts reading $\langle x?[\pi] \leq y?[\pi'] \rangle$ which is continued by rules 3.2 and 3.3 verifing condition (5). Rule 3.4 switches to the verification of condition (8) by rules 3.5 and 3.6. Rule 3.7 switches to the verification of conditions (6) and (7) which is done jointly by rules 3.8 and 3.9. The respective last symbols of π_1 and π_2 are memorized in the state (the symbols h and g in the state $\swarrow x{:}\searrow y, h, g$), allowing rule 3.7 to verify condition 10. In order to allow for π_1 and π_2 to be ε, the automaton also memorizes whether or not a feature symbol has been consumed (rules 3.1 and 3.4). Slightly abusing notation, we allow for h and g in these rules to denote either a feature symbol or ε.

Example 11 *The constraint from Example 6 entails $x[f]y$. This selection constraint is equivalent to the conjunction of the three path constraints $x[f]\downarrow$, $x?[f] \leq y?[\varepsilon]$, and $y?[\varepsilon] \leq x?[f]$. The words corresponding to these constraints are accepted by the following transitions of the automaton (for the constraint in Example 6):*

$$q_s \xrightarrow{x[} \swarrow x \xrightarrow{\varepsilon} \swarrow u \xrightarrow{f} \swarrow u' \xrightarrow{]\downarrow} q_f$$
$$q_s \xrightarrow{x?[} \searrow x{:}\varepsilon \xrightarrow{\varepsilon} \searrow v{:}\varepsilon \xrightarrow{f} \searrow v'{:}f \xrightarrow{\varepsilon} \searrow y{:}f \xrightarrow{]\leq y?[} \swarrow y{:}\searrow y, f, \varepsilon \xrightarrow{]\#} \searrow y \swarrow y \xrightarrow{\varepsilon} q_f$$
$$q_s \xrightarrow{y?[} \searrow y{:}\varepsilon \xrightarrow{\varepsilon} \searrow u'{:}\varepsilon \xrightarrow{]\leq x?[} \swarrow x{:}\searrow u', \varepsilon, \varepsilon \xrightarrow{\varepsilon} \swarrow u{:}\searrow u', \varepsilon, \varepsilon \xrightarrow{f} \swarrow u'{:}\searrow u', \varepsilon, f \xrightarrow{]\#} \searrow u' \swarrow u' \xrightarrow{\varepsilon} q_f$$

Label Compatibility (Fig. 10). Rules 4.1–4.11 check constraints of the kind $x?[\pi]{\sim}a$. Note that $\varphi \models x?[\pi]{\sim}a$ iff there are $y, y', z, z', v, v', b, c$ and $\pi_1, \pi_1', \pi_2, \pi_2'$ with $\pi = \pi_1\pi_2 = \pi_1'\pi_2'$ such that:

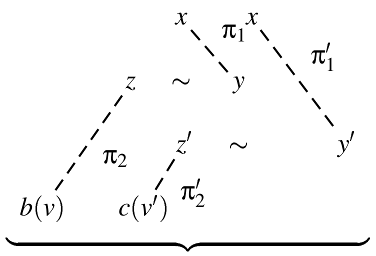
$$\varphi \models x?[\pi_1] \leq y?[\varepsilon] \wedge y{\sim}z \quad (11)$$
$$\varphi \models x?[\pi_1'] \leq y'?[\varepsilon] \wedge y'{\sim}z' \quad (12)$$
$$\varphi \models v?[\varepsilon] \leq z?[\pi_2], \quad (13)$$
$$\varphi \models v'?[\varepsilon] \leq z'?[\pi_2'] \quad (14)$$
$$\varphi \models b(v) \wedge c(v') \text{ and } (b \neq c \text{ or } a = b = c) \quad (15)$$

5.1	$\backslash x{:}h$	$\xrightarrow{]\sim y?[}$	$\backslash y{:}\backslash x,h,\varepsilon$	
5.2	$\backslash x{:}\backslash z,h,g$	$\xrightarrow{\varepsilon}$	$\backslash y{:}\backslash z,h,g$	$x{\leq}y{\in}\varphi$
5.3	$\backslash x{:}\backslash z,h,g$	$\xrightarrow{f}$	$\backslash y{:}\backslash z,h,f$	$x[f]y{\in}\varphi$
5.4	$\backslash x{:}\backslash z,h,g$	$\xrightarrow{]\#}$	$\backslash z\backslash x{:}\#$	$h{\neq}g \vee h{=}g{=}\varepsilon$
5.5	$\backslash x{:}\backslash z,h,g$	$\xrightarrow{\varepsilon}$	$\rangle y{:}\backslash z,h,g$	$x\sim y{\in}\varphi$
5.6	$\rangle x{:}\backslash z,h,g$	$\xrightarrow{\varepsilon}$	$\rangle y{:}\backslash z,h,g$	$x{\geq}y{\in}\varphi$
5.7	$\rangle x{:}\backslash z,h,g$	$\xrightarrow{f}$	$\rangle y{:}\backslash z,h,f$	$x[f]y{\in}\varphi$
5.8	$\rangle x{:}\backslash z,h,g$	$\xrightarrow{]\#}$	$\backslash z\rangle x$	$h{\neq}g \vee h{=}g{=}\varepsilon$
5.9	$\backslash x\backslash x'{:}\#$	$\xrightarrow{\varepsilon}$	$\backslash y\backslash y'{:}\#$	$x{\leq}y, x'{\leq}y'{\in}\varphi$
5.10	$\backslash x\backslash x'{:}\#$	$\xrightarrow{f}$	$\backslash y\backslash y'{:}\#$	$x[f]y, x'[f]y'{\in}\varphi$
5.11	$\backslash x\backslash x'{:}\#$	$\xrightarrow{\varepsilon}$	$\backslash x\rangle y'$	$x'\sim y'{\in}\varphi$
5.12	$\backslash x\rangle x'$	$\xrightarrow{\varepsilon}$	$\backslash y\rangle y'$	$x{\leq}y, x'{\geq}y'{\in}\varphi$
5.13	$\backslash x\rangle x'$	$\xrightarrow{f}$	$\backslash y\rangle y'$	$x[f]y, x'[f]y'{\in}\varphi$
5.14	$\backslash x\rangle x$	$\xrightarrow{\mathcal{F}(\varphi)^*}$	q_f	

6	$\dfrac{x?[\pi]{\sim}y?[\pi']\#\pi''{\in}\mathcal{L}(\mathcal{A}_\varphi)}{y?[\pi']{\sim}x?[\pi]\#\pi''{\in}\mathcal{L}(\mathcal{A}_\varphi)}$

Fig. 11: The section of the automaton $\mathcal{A}_\varphi$ for path constraints $x?[\pi_1]{\sim}y?[\pi_2]\#\pi_3$ which is concrete syntax for $x_1?[\pi_1\pi_3]{\sim}x_2?[\pi_2\pi_3]$.

The associated pattern looks as follows.



$$\models \quad x?[\pi_1\pi_2]{\sim}a$$
$$\text{if } (b \neq c \text{ or } a = b = c) \text{ and } \pi_1\pi_2 = \pi'_1\pi'_2$$

We check the conditions (11) and (12) as well as (13) and (14) in parallel where we assume, by symmetry, that π_1 is a prefix of π'_1. With the names used above, the automaton consumes π_1 by rules 4.2–4.3, switches y to z with rule 4.4, then consumes π_2 minus its suffix π'_2 (which is identical to π'_1 minus its prefix π_1) by rules 4.5–4.6, switches from y' to z' in rule 4.7 and consumes π'_2 in rules 4.8–4.9. Finally rules 4.10–4.11 check the label constraints (15).

Example 12 *In the case of Example 5 we obtain*

$$q_s \xrightarrow{x?[} \backslash x\backslash x \xrightarrow{\varepsilon} \backslash x\backslash y \xrightarrow{f} \backslash x'\backslash y' \xrightarrow{\varepsilon} \rangle x''\backslash y' \xrightarrow{\varepsilon} \rangle x''\rangle z' \xrightarrow{g} \rangle x'''\rangle z'' \xrightarrow{]\sim a} q_f$$

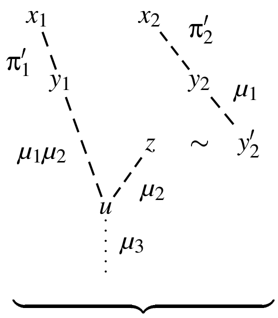
Path Compatibility (Fig. 11). Rules 5.1–5.14, in conjunction with rules 3.1–3.3, check for constraints $x_1?[\pi_1]{\sim}x_2?[\pi_2]$. One possible justification for $\varphi{\models}x_1?[\pi_1]{\sim}x_2?[\pi_2]$ is that there are variables y_1, y_2, y_2', z, u and paths $\pi_1', \pi_2', \mu_1, \mu_2, \mu_3$ such that $\pi_1 = \pi_1'\mu_1\mu_2\mu_3$, $\pi_2 = \pi_2'\mu_1\mu_2\mu_3$, and

$$\varphi \models x_1?[\pi_1'] \leq y_1?[\varepsilon] \tag{16}$$
$$\varphi \models y_1?[\mu_1\mu_2] \leq u?[\varepsilon] \tag{17}$$
$$\varphi \models x_2?[\pi_2'] \leq y_2?[\varepsilon] \tag{18}$$
$$\varphi \models y_2?[\mu_1] \leq y_2'?[\varepsilon] \wedge y_2' \sim z \tag{19}$$
$$\varphi \models u?[\varepsilon] \leq z?[\mu_2] \tag{20}$$

where we may assume that

$$\pi_1' \text{ and } \pi_2' \text{ have no common suffix except } \varepsilon \tag{21}$$

Note that there is no assumption on μ_3, *i.e.* μ_3 is arbitrary. This situation corresponds to the following pattern, where the arbitrary path μ_3 is indicated by a dotted line.



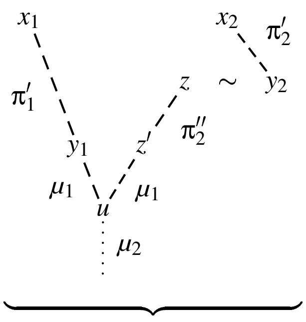
$$\models \quad x_1?[\pi_1'\mu_1\mu_2\mu_3] \sim x_2?[\pi_2'\mu_1\mu_2\mu_3]$$

The rules 3.1–3.3, 5.1–5.4 and 5.9–5.14 deal with this situation: Rules 3.2–3.3 consume π_1' and rules 5.2–5.3 consume π_2'; rules 5.9–5.10 and 5.12–5.13 consume μ_1 and μ_2, respectively, *i.e.*, the part of the common suffix $\mu_1\mu_2$ that is explicit in the constraint graph, and rule 5.14 consumes the rest of the common suffix μ_3 which is arbitrary and does not explicitly occur in the constraint graph. Condition 21 is checked in rule 5.4 in the same way as it has been done for rule 3.7.

The second justification is similar but contains the switch through the compatibility constraint $\sim$ before the common suffix of π_1 and π_2 instead of within it; *i.e.*, there are variables y_1, y_2, z, z', u and paths $\pi_1', \pi_2', \pi_2'', \mu_1, \mu_2$ such that $\pi_1 = \pi_1'\mu_1\mu_2$, $\pi_2 = \pi'\pi_2''\mu_1\mu_2$, and

$$\varphi \models x_1?[\pi_1'] \leq y_1?[\varepsilon] \tag{22}$$
$$\varphi \models y_1?[\mu_1] \leq u?[\varepsilon] \tag{23}$$
$$\varphi \models x_2?[\pi_2'] \leq y_2?[\varepsilon] \wedge y_2 \sim z \tag{24}$$
$$\varphi \models z'?[\varepsilon] \leq z?[\pi_2''] \tag{25}$$
$$\varphi \models u?[\varepsilon] \leq z'?[\mu_1] \tag{26}$$

The associated pattern is:



$$\models \quad x_1?[\pi'_1\mu_1\mu_2]{\sim}x_2?[\pi'_2\pi''_2\mu_1\mu_2]$$

For the traversal of this pattern we need the additional rules 5.5–5.8 (instead of rules 5.9–5.11).

For both situations there also is the symmetric one (rule 6) which contains the switch through the compatibility constraint $\sim$ in the branch for x_1. We do not detail the automaton checking for these possibilities since its definition is completely symmetric to the rules 3.1–3.3 and 5.1–5.14.

Proposition 5.8 (Correctness of the Automaton) *If* $\langle\psi\rangle \in \mathcal{L}(\mathcal{A}^{\bar{x}}_{\varphi})$ *then* $\exists\bar{x}\varphi \models \psi$.

Proof. By induction over the paths mentioned in ψ. □

5.6 Deciding Entailment in PSPACE

Theorem 5.9 *The entailment problem for existentially quantified* $FT_{\leq}$*-constraints is in PSPACE (and thus PSPACE-complete) in both the finite and the infinite tree case.*

In order to decide $\varphi \models \exists\bar{x}\varphi'$, we test satisfiability of φ and $\varphi \wedge \exists\bar{x}\varphi'$. By Proposition 5.7, this can be done in time $O(n^3)$ where n is the size of the entailment problem. If one of the tests fails, entailment is trivial. Otherwise, we compute the F-closures of φ and of φ' and construct the associated automata $\mathcal{A}_{\varphi}$ and $\mathcal{A}^{\bar{x}}_{\varphi'}$ in time $O(n^4)$. By Proposition 5.10, $\varphi \models \exists\bar{x}\varphi'$ if and only if $\mathcal{L}(\mathcal{A}^{\bar{x}}_{\varphi'}) \subseteq \mathcal{L}(\mathcal{A}_{\varphi})$. This inclusion is decidable in PSPACE [9].

Proposition 5.10 (Correctness and Completeness of the Entailment Test) *Let* φ *and* φ' *be closed* $FT_{\leq}$ *constraints and* $\bar{x}$ *a sequence of variables such that all free variables and features in* $\exists\bar{x}\varphi'$ *occur in* φ. *Further assume that* $\varphi \wedge \exists\bar{x}\varphi'$ *is satisfiable. Then*

$$\varphi \models \exists\bar{x}\varphi' \qquad \textit{if and only if} \qquad \mathcal{L}(\mathcal{A}^{\bar{x}}_{\varphi'}) \subseteq \mathcal{L}(\mathcal{A}_{\varphi})\,.$$

Proof. The proof is subject of Sections 6 and 7. The plan is as follows:

1. Correctness - the direction from right to left - will follow from a characterization of formulas with (or without) existential quantifiers in terms of regular languages of path constraints. For all sequences of variables $\bar{y}$ and constraints φ_0 the formula $\exists\bar{y}\varphi_0$ is equivalent to the conjunction of path constraints recognized by the automaton $\mathcal{A}^{\bar{y}}_{\varphi_0}$ (see Proposition 6.6):

$$\exists\bar{y}\varphi_0 \models\!\mid \bigwedge\{\psi \mid \langle\psi\rangle \in \mathcal{L}(\mathcal{A}^{\bar{y}}_{\varphi_0})\}$$

2. Completeness is the the direction from left to right. We assume $\varphi \models \exists\bar{x}\varphi'$. Proposition 7.3 asserts that for all ψ with $\mathcal{V}(\psi) \subseteq \mathcal{V}(\varphi)$ and $\mathcal{F}(\psi) \subseteq \mathcal{F}(\varphi)$ it holds that

$\langle\psi\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$ implies $\varphi \not\models \langle\psi\rangle$. So, assume that $\mathcal{L}(\mathcal{A}^{\bar{x}}_{\varphi'}) \not\subseteq \mathcal{L}(\mathcal{A}_\varphi)$, that is that there is a $\langle\psi\rangle \in \mathcal{L}(\mathcal{A}^{\bar{x}}_{\varphi'}) - \mathcal{L}(\mathcal{A}_\varphi)$. By construction of the automaton, $\mathcal{V}(\psi) \subseteq \mathcal{V}(\exists\bar{x}\varphi') \subseteq \mathcal{V}(\varphi)$. By Proposition 5.8, $\exists\bar{x}\varphi' \models \langle\psi\rangle$, and by Proposition 7.3, $\varphi \not\models \langle\psi\rangle$, which contradicts the assumption $\varphi \models \exists\bar{x}\varphi'$.

□

6 Correctness of the Entailment Test

The correctness part in the proof of Proposition 5.10 bases on a characterization of existential formulas in terms of regular languages of path constraints that are recognized by the constructed automata (Proposition 6.6).

6.1 Properties of $\mathcal{A}_\varphi$

Clearly, the states of the automaton $\mathcal{A}_\varphi$ carry a lot of cumbersome control information (for testing two properties simultaneously, or for recognizing greatest common suffixes). We first formulate three Lemmas 6.2, 6.3, and 6.4 that allow us to safely ignore the control information. Based on these, we show the key Lemma 6.5 for correctness, which states a closure property for the automaton $\mathcal{A}_\varphi$.

In the following we note the fact that the automaton A_φ allows a sequence of transitions from state q_1 to state q_2 by reading the word π by

$$A_\varphi \quad \vdash \quad q_1 \xrightarrow{\pi} q_2$$

Definition 6.1 (Shortcuts)

1. *We write* $\mathcal{A}_\varphi \vdash \searrow x \xrightarrow{\pi} \searrow y$ *if* $\mathcal{A}_\varphi \vdash \searrow x{:}g \xrightarrow{\pi} \searrow y{:}h$ *for some* $g,h \in \mathcal{L} \cup \{\varepsilon\}$.
2. *We write* $\mathcal{A}_\varphi \vdash \searrow x \xrightarrow{\pi} \rightsquigarrow y$ *if* $\mathcal{A}_\varphi \vdash \searrow x{:}\searrow z,h,g \xrightarrow{\pi} \rightsquigarrow y{:}\searrow z,h,f$ *for some* z,f,g,h.

Lemma 6.2 *For all* x,y,π*, there exists a transition of the form* $\mathcal{A}_\varphi \vdash \searrow x \xrightarrow{\pi} \rightsquigarrow y$ *if and only if there are* z,z' *and a decomposition of* π*, say* $\pi = \pi'\pi''$ *for some* π',π''*, such that:*

$$\mathcal{A}_\varphi \vdash \searrow x \xrightarrow{\pi'} \searrow z, \quad z{\sim}z' \in \varphi, \quad \mathcal{A}_\varphi \vdash \swarrow z' \xrightarrow{\pi''} \swarrow y$$

Proof. Follows from the construction of the automaton by some straightforward inductions that we omit as they don't contribute further insights. □

Lemma 6.3 (Using Shortcuts) *For all* φ,x,y,μ,ν,a *the following equivalences hold:*

1. $\langle x?[\mu]{\le}y?[\nu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ *if and only if there exist a (not necessary longest) common suffix* $\pi \in \mathcal{F}(\varphi)^*$ *of* μ *and* ν *and two transitions of the following forms for some* μ',ν',z *with* $\mu = \mu'\pi$ *and* $\nu = \nu'\pi$*:*

$$\mathcal{A}_\varphi \vdash \searrow x \xrightarrow{\mu'} \searrow z \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash \swarrow y \xrightarrow{\nu'} \swarrow z$$

2. $\langle x?[\mu]{\sim}y?[\nu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ *if and only if there exists* u *and a common suffix* $\pi \in \mathcal{F}(\varphi)^*$ *of* μ *and* ν*, i.e. there are* μ' *and* ν' *with* $\mu = \mu'\pi$ *and* $\nu = \nu'\pi$*, such that one of the following symmetric properties holds:*

(a) $\mathcal{A}_\varphi \vdash \backslash x \xrightarrow{\mu'} \backslash u$ *and* $\mathcal{A}_\varphi \vdash \backslash y \xrightarrow{\nu'} \rangle u$

(b) *or* $\mathcal{A}_\varphi \vdash \backslash x \xrightarrow{\mu'} \rangle u$ *and* $\mathcal{A}_\varphi \vdash \backslash y \xrightarrow{\nu'} \backslash u$

3. $\langle x?[\pi]{\sim}a\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ *if and only if there exist variables* x_1, x_2 *and labels* $a_1 = a_2 = a$ *or* $a_1 \neq a_2$ *such that* $\mathcal{A}_\varphi \vdash \backslash x \xrightarrow{\mu} \rangle x_i$ *and* $a_i(x_i) \in \varphi$ *for* $i = 1$ *and* $i = 2$.

Proof. In all three cases it follows immediately from the definition of the automaton that if the respective path constraint is in $\mathcal{L}(\mathcal{A}_\varphi)$, then there exist paths such that the claimed transitions can be performed. The problem is to show the inverse direction for case 1 and 2 since μ' and ν' may have a common non-trivial suffix.

For every $h \in \mathcal{F} \cup \{\varepsilon\}$, we define the function $last_h : \mathcal{F}^* \to (\mathcal{F} \cup \{\varepsilon\})$ as follows:

$$\mathrm{last}_h(\pi) = \begin{cases} h & \text{if } \pi = \varepsilon \\ f & \text{if } \pi = \pi' f \text{ for some } \pi' \end{cases}$$

For the first claim, let $\mu' = \mu''\pi'$ and $\nu' = \nu''\pi'$ such that μ'' and ν'' have no non-trivial common suffix. We show that $x?[\mu'']{\le}y?[\nu'']\#\pi'\pi = \langle x?[\mu]{\le}y?[\nu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.

$$\begin{array}{lll}
q_s \xrightarrow{x?[} & \backslash x{:}\varepsilon & \text{rule 3.1} \\
\xrightarrow{\mu''} & \backslash x_1{:}\mathrm{last}_\varepsilon(\mu'') & \text{rule 3.2, 3.3} \\
\xrightarrow{]\le y?[} & /y{:}\backslash x_1, \mathrm{last}_\varepsilon(\mu''), \varepsilon & \text{rule 3.4} \\
\xrightarrow{\nu''} & /y_1{:}\backslash x_1, \mathrm{last}_\varepsilon(\mu''), \mathrm{last}_\varepsilon(\nu'') & \text{rule 3.5, 3.6} \\
\xrightarrow{]\#} & \backslash x_1/y_1 & \text{rule 3.7} \\
\xrightarrow{\pi'} & \backslash z/z & \text{rule 3.8, 3.9} \\
\xrightarrow{\pi} & q_f & \text{rule 3.10 and } \pi \in \mathcal{F}(\varphi)^*
\end{array}$$

The second claim is proven analogously. The proof of the third claim is simpler since no common suffix has to be factored out. □

Lemma 6.4 (Compatibility) *Let* φ *be* F*-closed and assume variables* x, z_1, z_2 *and a path* π. *If there exist transitions* $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi} /z_1$ *and* $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi} /z_2$ *then* $z_1{\sim}z_2 \in \varphi$.

Proof. We slightly strengthen the statement of the lemma to the following claim:

C1 For all x_1, x_2, π, z_1, z_2 if $x_1{\sim}x_2 \in \varphi$, $\mathcal{A}_\varphi \vdash /x_1 \xrightarrow{\pi} /z_1$ and $\mathcal{A}_\varphi \vdash /x_2 \xrightarrow{\pi} /z_2$ then $z_1{\sim}z_2 \in \varphi$.

The lemma follows from claim C1 when choosing $x = x_1 = x_2$. In this case, $x \in \mathcal{V}(\varphi)$ and F1.1-closeness of φ yields $x{\le}x \in \varphi$ such that F3.1-closeness of φ guarantees $x{\sim}x \in \varphi$. We next prove C1 by induction on π:

1. Case $\pi = \varepsilon$: There exist two sequences of variables $x_1 = u_1, \ldots, u_n = z_1$ and $x_2 = v_1, \ldots, v_m = z_2$ with the following transitions for $1 \le i < n$ and $1 \le j < m$:

$$\mathcal{A}_\varphi \vdash /u_i \xrightarrow{\varepsilon} /u_{i+1} \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash /v_j \xrightarrow{\varepsilon} /v_{j+1}$$

The application condition of rule 1.2 implies for $1 \le i < n$ and $1 \le j < m$:

$$u_i{\ge}u_{i+1} \in \varphi \qquad \text{and} \qquad v_j{\ge}v_{j+1} \in \varphi$$

Since φ is closed with respect to transitivity due to F1.2 we obtain $z_1{\le}x_1 \in \varphi$ and $z_2{\le}x_2 \in \varphi$, and by F3.2-closeness we get $z_1{\sim}x_2 \in \varphi$. Hence $x_2{\sim}z_1 \in \varphi$ since $\sim$ is symmetric due to F3.3. Thus F3.2-closeness again yields $z_2{\sim}z_1 \in \varphi$. Finally, symmetry again implies $z_1{\sim}z_2 \in \varphi$.

2. Case $\pi = f\pi'$ for some f, π': There exist u_1, v_1, u_2, v_2 such that the following transitions exist:

$$\mathcal{A}_\varphi \vdash /x_1 \xrightarrow{\varepsilon} /u_1 \xrightarrow{f} /v_1 \xrightarrow{\pi'} /z_1$$
$$\mathcal{A}_\varphi \vdash /x_2 \xrightarrow{\varepsilon} /u_2 \xrightarrow{f} /v_2 \xrightarrow{\pi'} /z_2$$

As proved in the case $\pi = \varepsilon$, this implies $u_1{\sim}u_2 \in \varphi$. The application condition of rule 1.3 yields $u_1[f]v_1 \in \varphi$ and $u_2[f]v_2 \in \varphi$. Thus, the closeness under the decomposition axiom F4 implies $v_1{\sim}v_2 \in \varphi$. Finally, $z_1{\sim}z_2$ follows from the induction hypothesis.

□

Lemma 6.5 (Key Lemma) *For all paths $\mu_1, \mu_2, \pi_1, \pi_2$, variables x, y_1, y_2, and* F*-closed* φ*:*

1. *If $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\langle a(x[\pi_1\pi_2])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ then $\langle y_1?[\mu_1\pi_2]{\sim}a\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$*
2. *If $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\langle y_2?[\mu_2]{\le}x?[\pi_1\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ then $\langle y_1?[\mu_1\pi_2]{\sim}y_2?[\mu_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.*

Proof.

1. Let $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\langle a(x[\pi_1\pi_2])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. According to Lemma 6.3(1), the first assumption $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ is equivalent to the existence of ν_1, μ_1', π_1' with $\mu_1 = \mu_1'\nu_1$ and $\pi_1 = \pi_1'\nu_1$ and of transitions of the following forms for some variable z_1:

$$\mathcal{A}_\varphi \vdash \backslash y_1 \xrightarrow{\mu_1'} \backslash z_1 \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash /x \xrightarrow{\pi_1'} /z_1$$

The assumption $\langle a(x[\pi_1\pi_2])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ yields the existence of u_2, v_2 with the following transitions of $\mathcal{A}_\varphi$ based on rules 2.1–2.4:

$$q_s \xrightarrow{a([}_{2.1} /x{:}a \xrightarrow{\pi_1'}_{2.2,2.3} /u_2{:}a \xrightarrow{\nu_1\pi_2}_{2.2,2.3} /v_2{:}a \xrightarrow{])}_{2.4} q_f$$

Thus, there are two transitions $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi_1'} /u_2$ and $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi_1'} /z_1$ such that Lemma 6.4 and the F-closeness of φ yield $z_1{\sim}u_2 \in \varphi$. We now construct a transition proving $\langle y_1?[\mu_1\pi_2]{\sim}a\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$:

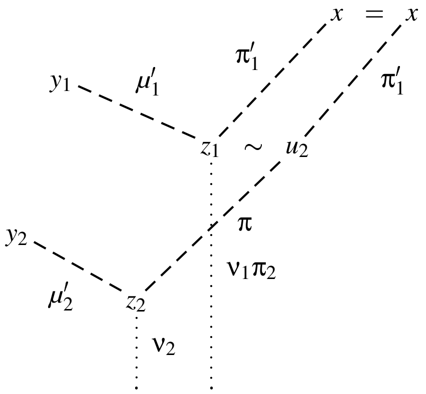
$$\begin{array}{llll}
q_s & \xrightarrow{y_1?[}_{4.1} & \backslash y_1 \backslash y_1 & \\
 & \xrightarrow{\mu_1'}_{4.2,4.3} & \backslash z_1 \backslash z_1 & \mathcal{A}_\varphi \vdash \backslash y_1 \xrightarrow{\mu_1'} \backslash z_1 \\
 & \xrightarrow{\varepsilon}_{4.4,4.7} & \rangle u_2 \rangle u_2 & z_1{\sim}u_2 \in \varphi \\
 & \xrightarrow{\nu_1\pi_2}_{4.8,4.9} & \rangle v_2 \rangle v_2 & \mathcal{A}_\varphi \vdash /u_2 \xrightarrow{\nu_1\pi_2} /v_2 \\
 & \xrightarrow{]\sim a}_{4.10} & q_f & a(v_2) \in \varphi
\end{array}$$

2. We now assume $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\langle y_2?[\mu_2]{\le}x?[\pi_1\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. According to Lemma 6.3(1) the first assumption $\langle y_1?[\mu_1]{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ is equivalent to the existence of ν_1, μ_1', π_1' with $\mu_1 = \mu_1'\nu_1$ and $\pi_1 = \pi_1'\nu_1$ and of transitions of the following forms for some variable z_1:

$$\mathcal{A}_\varphi \vdash \backslash y_1 \xrightarrow{\mu_1'} \backslash z_1 \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash /x \xrightarrow{\pi_1'} /z_1$$

The second assumption $\langle y_2?[\mu_2]{\le}x?[\pi_1\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ yields the existence of ν_2, μ_2', π_2' such that $\mu_2 = \mu_2'\nu_2$ and $\pi_1\pi_2 = \pi_2'\nu_2$ and of the following transition for some variable z_2:

$$\mathcal{A}_\varphi \vdash \backslash y_2 \xrightarrow{\mu_2'} \backslash z_2 \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash /x \xrightarrow{\pi_2'} /z_2$$



$$\begin{aligned} &\mu_1 = \mu'_1 \nu_1 \\ &\pi_1 = \pi'_1 \nu_1 \\ &\mu_2 = \mu'_2 \nu_2 \\ &\pi_1\pi_2 = \pi'_2 \nu_2 \\ &\pi'_2 = \pi'_1 \pi \qquad \text{(case 2.(a))} \end{aligned}$$

Fig. 12: The paths in the proof of claim 2.(a) of the Key Lemma.

We distinguish two cases depending on whether π'_1 is a prefix of π'_2 or vice versa. This case distinction is complete since $\pi'_1\nu_1\pi_2 = \pi'_2\nu_2$ such that the paths π'_1 and π'_2 can not diverge (see Figure 12).

(a) π'_1 is a prefix of π'_2: There exist π with $\pi'_1\pi = \pi'_2$ and u_2 such that:

$$\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_1} /u_2 \xrightarrow{\pi} /z_2$$

Hence, $\pi\nu_2 = \nu_1\pi_2$, and there are two transitions $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_1} /u_2$ and $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_1} /z_1$ such that Lemma 6.4 and the F-closeness of φ yield $z_1{\sim}u_2 \in \varphi$. By combining our intermediate results

$$\begin{aligned} &\mathcal{A}_\varphi \vdash \backslash y_1 \xrightarrow{\mu'_1} \backslash z_1, \quad z_1{\sim}u_2 \in \varphi, \quad \mathcal{A}_\varphi \vdash /u_2 \xrightarrow{\pi} /z_2 \\ &\mathcal{A}_\varphi \vdash \backslash y_2 \xrightarrow{\mu'_2} \backslash z_2 \end{aligned}$$

with Lemma 6.3(2), we obtain $\langle y_1?[\mu'_1\pi\nu_2]{\sim}y_2?[\mu'_2\nu_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. The claim follows since $\mu'_2\nu_2 = \mu_2$, and $\mu'_1\pi\nu_2 = \mu'_1\nu_1\pi_2 = \mu_1\pi_2$.

(b) π'_2 is a prefix of π'_1: There exist π with $\pi'_2\pi = \pi'_1$ and u_1 such that:

$$\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_2} /u_1 \xrightarrow{\pi} /z_1$$

Hence, $\pi\nu_1\pi_2 = \nu_2$, and there are two transitions $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_2} /u_1$ and $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi'_2} /z_2$ such that Lemma 6.4 and the F-closeness of φ yield $z_2{\sim}u_1 \in \varphi$. By combining our intermediate results

$$\begin{aligned} &\mathcal{A}_\varphi \vdash \backslash y_1 \xrightarrow{\mu'_1} \backslash z_1 \\ &\mathcal{A}_\varphi \vdash \backslash y_2 \xrightarrow{\mu'_2} \backslash z_2, \quad z_2{\sim}u_1 \in \varphi, \quad \mathcal{A}_\varphi \vdash /u_1 \xrightarrow{\pi} /z_1 \end{aligned}$$

with Lemma 6.3(2), we obtain $\langle y_1?[\mu'_1\nu_1\pi_2]{\sim}y_2?[\mu'_2\pi\nu_1\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. The claim follows since $\mu'_1\nu_1\pi_2 = \mu_1\pi_2$, and since $\mu'_2\pi\nu_1\pi_2 = \mu'_2\nu_2 = \mu_2$.

$\square$

6.2 Characterization of Existential Formulas

In the following we will slightly abuse notation and allow in writing path constraints their concrete syntax. This allows us to write $\bigwedge \mathcal{L}(\mathcal{A}_\varphi)$ instead of $\bigwedge\{\psi \mid \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_\varphi)\}$ and similarly for $\bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\{\bar{x}\}})$. With this notation in mind, the characterization proposition can be written as:

Proposition 6.6 (Characterization of existential formulas by path constraints) *If* φ *is an* F*-closed* $FT_\leq$ *constraint and* $\bar{x}$ *a sequence of variables then*

$$\exists\bar{x}\varphi \quad \models\!\mid \quad \bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$$

Proof. The implication from left to right follows form the correctness of the automata construction (Proposition 5.8). For the inverse direction, we assume a solution α of $\bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$. We define an extension α' of α by setting, for all $x \in \mathcal{V}(\exists\bar{x}\varphi)$: $\alpha'(x) = \alpha(x)$, and for all $x \in \{\bar{x}\}$:

$$\begin{array}{rcl} D_{\alpha'(x)} & = & \{\pi \mid \langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)\} \cup \\ & & \{\pi\pi'' \mid z \in \mathcal{V}(\exists\bar{x}\varphi),\ \langle z?[\pi'] \leq x?[\pi]\rangle \in \mathcal{L}(\mathcal{A}_\varphi), \pi'\pi'' \in D_{\alpha(z)}\} \\ L_{\alpha'(x)} & = & \{(\pi, a) \mid \langle a(x[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)\} \cup \\ & & \{(\pi\pi'', a) \mid z \in \mathcal{V}(\exists\bar{x}\varphi),\ \langle z?[\pi'] \leq x?[\pi]\rangle \in \mathcal{L}(\mathcal{A}_\varphi), (\pi'\pi'', a) \in L_{\alpha(z)}\} \end{array}$$

To complete the proof we have to show

1. that $\alpha'(x)$ is a feature tree. This statement is not completely obvious for $x \in \{\bar{x}\}$:
   (a) $\mathcal{D}_{\alpha'(x)}$ is non-empty since x belongs to the input alphabet of the automaton $\mathcal{A}_\varphi$ which therefore accepts $\langle x[\varepsilon]\downarrow\rangle$.
   (b) $D_{\alpha'(x)}$ is prefix closed, as shown in Lemma 6.7.
   (c) $L_{\alpha'(x)}$ is a partial function as shown in Lemma 6.8 which mainly relies on the Key Lemma 6.5.
   (d) In the case of finite trees, we have to show that $D_{\alpha'(x)}$ is finite. This is done in Lemma 6.9.
2. that α' is indeed a solution of φ. This in done in Lemma 6.10.

□

Lemma 6.7 $D_{\alpha'(x)}$ *is prefix closed.*

Proof. The only interesting case is $x \in \{\bar{x}\}$. The proof relies essentially on the following claim which relies directely on the definition of the automaton $\mathcal{A}_\varphi$:

C2 For all x, π, $\langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ iff there exists y such $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi} /y$.

We show the prefix closeness of $D_{\alpha'(x)}$ as follows. Suppose $\pi f \in D_{\alpha'(x)}$. There are two cases according to the definition of $D_{\alpha'(x)}$:

1. Case $\langle x[\pi f]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$: Claim C2 yields the existence of y such that $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi f} /y$. Hence there also exists z with $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi} /z$ such that Claim C2 yields $\langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, i.e. $\pi \in D_{\alpha'(x)}$.
2. Case exists $\mu, \mu', \mu'', z \in \mathcal{V}(\exists\bar{x}\varphi)$ with $\pi f = \mu\mu''$, $\langle z?[\mu'] \leq x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\mu'\mu'' \in D_{\alpha(z)}$: We distinguish two sub-cases:

(a) If $\mu'' = \varepsilon$ then $\langle z?[\mu'] \leq x?[\pi f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Hence, by Lemma 6.3 there are μ_1, μ_2, μ_3 with $\mu' = \mu_1\mu_3$ and $\pi f = \mu_2\mu_3$ and a variable y such that $\mathcal{A}_\varphi \vdash \backslash z \xrightarrow{\mu_1} \backslash y$ and $\mathcal{A}_\varphi \vdash /x \xrightarrow{\mu_2} /y$. We distinguish again two cases:

   i. $\mu_3 = \varepsilon$, hence $\mu_2 = \pi f$. In this case, there is a variable y' such that $\mathcal{A}_\varphi \vdash /x \xrightarrow{\pi} /y'$, hence $\langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\pi \in D_{\alpha'(x)}$.

   ii. $\mu_3 = \mu_3' f$. In this case, we obtain from Lemma 6.3 that $\langle z?[\mu_1] \leq x?[\mu_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Since $\mu_1\mu_3' f \in D_{\alpha(z)}$, $\mu_1\mu_3' \in D_{\alpha(z)}$ by prefix-closeness of the domain of $\alpha(z)$, hence $\mu_2\mu_3' = \pi \in D_{\alpha(x)}$.

(b) Case $\mu'' = \tilde{\mu}'' f$ for some $\tilde{\mu}''$: Since $D_{\alpha(z)}$ is prefix closed, we know $\mu'\tilde{\mu}'' \in D_{\alpha(z)}$. Hence, $\pi = \mu\tilde{\mu}'' \in D_{\alpha'(x)}$.

□

Lemma 6.8 *$L_{\alpha'(x)}$ is a partial function on $D_{\alpha'(x)}$.*

Proof. It is again sufficient to assume $x \in \{\bar{x}\}$.

1. We first show that the definition domain of $L_{\alpha'(x)}$ is a subset of $D_{\alpha'(x)}$, i.e. we prove for all π, a that if $(\pi, a) \in L_{\alpha'(x)}$ then $\pi \in D_{\alpha'(x)}$. There are two cases to be considered according to the definition of $L_{\alpha'(x)}$.

   (a) Case $\langle a(x[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$: From the definition of the automaton it is easy to see that $\langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, hence $\pi \in D_{\alpha'(x)}$.

   (b) Case $\pi = \mu\mu''$ for some μ, μ'', $z \in V$, $\langle z?[\mu'] \leq x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, and $(\mu'\mu'', a) \in L_{\alpha(z)}$: Hence, $\mu'\mu'' \in D_{\alpha(z)}$ which implies $\pi = \mu\mu'' \in D_{\alpha'(x)}$.

2. We show that the relation $L_{\alpha'(x)}$ is functional, *i.e.* for all π, a, b if $(\pi, a) \in L_{\alpha'(x)}$ and $(\pi, b) \in L_{\alpha'(x)}$ then $a = b$.

   (a) Suppose that (π, a) and (π, b) are both contributed to $L_{\alpha'(x)}$ by the first clause of its definition. Then, by Proposition 5.8,

   $$\varphi \models \bigwedge \mathcal{L}(\mathcal{A}_\varphi) \models a(x[\pi]) \wedge b(x[\pi])$$

   such that the satisfiability of φ (which follows from F-closeness of φ and Proposition 5.7) implies $a = b$.

   (b) Suppose that both pairs have been contributed to $L_{\alpha'(x)}$ by the second clause of its definition. There exist paths $\mu, \mu', \mu'', \nu, \nu', \nu''$ and variables $y, z \in V$ such that $\pi = \mu\mu'' = \nu\nu''$ and

      i. $\langle y?[\mu'] \leq x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, $(\mu'\mu'', a) \in D_{\alpha(y)}$

      ii. $\langle z?[\nu'] \leq x?[\nu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, $(\nu'\nu'', b) \in D_{\alpha(z)}$

   Since $\mu\mu'' = \nu\nu''$ either μ is a prefix of ν or vice versa. Without loss of generality, we can assume that ν is a prefix of μ, i.e. $\mu = \nu\pi_1$ for some π_1. The Key Lemma 6.5 implies $\langle y?[\mu'] \sim z?[\nu'\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. The assumption $y, z \in \mathcal{V}(\exists\bar{x}\varphi)$ and the Correctness Proposition 5.8 yield:

   $$\varphi \quad \models \quad \bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}}) \quad \models \quad y?[\mu'] \sim z?[\nu'\pi_1] \quad \models \quad y?[\mu'\mu''] \sim z?[\nu'\pi_1\mu'']$$

   It remains to show that $\pi_1\mu'' = \nu''$ which then implies $a = b$ since $(\mu'\mu'', a) \in D_{\alpha(y)}$ and $(\nu'\nu'', b) \in D_{\alpha(z)}$. This can be seen as follows. Since $\mu = \nu\pi_1$ we know $\mu\mu'' = \nu\pi_1\mu''$. In combination with $\mu\mu'' = \nu\nu''$ this implies $\nu\nu'' = \nu\pi_1\mu''$ and hence $\nu'' = \pi_1\mu''$ as required.

(c) Suppose that (π, a) is contributed by the first clause of the definition of $\mathcal{L}_{\alpha'(x)}$ and (π, b) by its second clause. There exist paths μ,μ',μ'' and a variables $y \in \mathcal{V}(\exists\bar{x}\varphi)$ such that $\pi = \mu\mu''$ and

i. $\langle y?[\mu']{\le}x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, $(\mu'\mu'', b) \in D_{\alpha(y)}$

ii. $\langle a(x[\mu\mu''])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$

Part 1 of the Key Lemma 6.5 implies $\langle y?[\mu'\mu'']{\sim}a\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Since $y \in \mathcal{V}(\exists\bar{x}\varphi)$ the Correctness Proposition 5.8 yields:

$$\varphi \quad\models\quad \bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}}) \quad\models\quad y?[\mu'\mu'']{\sim}a$$

Since α is a solution of $\bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ and $(\mu'\mu'', b) \in L_{\alpha(y)}$ we conclude $b = a$.

□

Lemma 6.9 *If we consider the model of finite trees $FT_\le^{fin}$ then $\alpha'(x)$ is finite for all x.*

Proof. Let $V = \mathcal{V}(\exists\bar{x}\varphi)$, and let $D_{\alpha(z)}$ be finite for all $z \in V$. We have to show that $D_{\alpha'(x)}$ is finite for all $x \in \{\bar{x}\}$.

In the case of finite feature trees, the following axiom is required by F-closeness of φ:

$$\mathsf{F6} \quad x_1{\le}x_{n+1} \notin \varphi \quad \text{if} \quad x_i[f_i]y_i \wedge x_{i+1}{\le}y_i \in \varphi \qquad \text{for all } 1 \le i \le n$$

Let n be the number of variables of φ and d be the maximal depth of any tree $\alpha(z)$ for $z \in V$. Note that d is finite since V is finite and all trees $\alpha(z)$ for $z \in V$ have finite depth.

We show that for all $x \in \{\bar{x}\}$ the length of the paths in $D_{\alpha'(x)}$ is bounded by $n + d$. Let $\pi = \pi_1 \cdots \pi_p \in D_{\alpha'(x)}$.

1. Case $\langle x[\pi]{\downarrow}\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Then we have

$$\mathcal{A}_\varphi \vdash /x \xrightarrow{\varepsilon} /x_1 \xrightarrow{\pi_1} /y_1 \xrightarrow{\varepsilon} /x_2 \xrightarrow{\pi_2} /y_2 \cdots /x_p \xrightarrow{\pi_p} /y_p \xrightarrow{\varepsilon} /x_{p+1}$$

and hence, with an argument as in the proof of Lemma 6.4, that $x_{i+1}{\le}y_i \in \varphi$ for all $1 \le i \le p$. By F6-closeness, all variables x_i have to be different, hence $p \le n \le n+d$.

2. Case $\pi = \pi_1\pi''$, $\langle z?[\pi']{\le}x?[\pi_1]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and $\pi'\pi'' \in D_{\alpha(z)}$. In this case $|\pi''| \le d$ by assumption and $|\pi_1| \le n$ as in the first case, hence $p = |\pi_1| + |\pi''| \le d + n$.

□

Lemma 6.10 *The variable assignment α' is a solution of φ (if α is a solution of $\bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ which we assume).*

Proof. Let $V = \mathcal{V}(\exists\bar{x}\varphi)$ be the set of global variables. We have to show that all basic constraints in φ are validated by α'. There are three kinds of basic constraints: $a(x)$, $x[f]y$ and $x{\le}y$, and we have to consider all combinations of x and y being in V or not. Hence there are 10 different cases.

1. Case $x[f]y \in \varphi$, $x, y \in V$ both global. In this case we have

$$\{\langle x[f]{\downarrow}\rangle, \langle x?[f]{\le}y?[\varepsilon]\rangle, \langle y?[\varepsilon]{\le}x?[f]\rangle\} \subseteq \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$$

These three path constraints are hence satisfied by α, and so is $x[f]y$ which is also satisfied by α' since $x, y \in V$ and thus $\alpha(x) = \alpha'(x)$ and $\alpha(y) = alpha'(y)$.

2. Case $x[f]y \in \varphi$, $x \in V$ global, $y \notin V$ local. For all π, a, we have to prove the following two equivalences: $\pi \in D_{\alpha'(y)}$ iff $f\pi \in D_{\alpha(x)}$ and $(\pi, a) \in L_{\alpha'(y)}$ iff $(f\pi, a) \in L_{\alpha(x)}$

   (a) We assume $\pi \in D_{\alpha'(y)}$ and show $f\pi \in D_{\alpha(x)}$:

      i. Case π is contributed to $D_{\alpha'(y)}$ by the first clause of its definition, i.e. $\langle y[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. The assumption $x[f]y \in \varphi$ and Claim C2 imply $\langle x[f\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and thus, since $x \in V$, $\langle x[f\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$. Since α is a solution $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ we conclude $f\pi \in D_{\alpha(x)}$.

      ii. Case π is contributed to $D_{\alpha'(y)}$ by the second clause of its definition, i.e. there exist μ, μ', μ'' and a variable $z \in V$ such that $\pi = \mu\mu''$ and:

      $$\langle z?[\mu'] \leq y?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi),\ \mu'\mu'' \in D_{\alpha(z)}$$

      The first condition $\langle z?[\mu'] \leq y?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and assumption $x[f]y \in \varphi$ yield $\langle z?[\mu'] \leq x?[f\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ due to Lemma 6.3 part 1. Since $z, x \in V$ we obtain $\langle z?[\mu'] \leq x?[f\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$, and since α is a solution of $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ and since $\mu'\mu'' \in D_{\alpha(z)}$ that $f\pi = f\mu\mu'' \in D_{\alpha(x)}$.

   (b) We assume $f\pi \in D_{\alpha(x)}$ and show $\pi \in D_{\alpha'(y)}$. Applied to our assumption $x[f]y \in \varphi$, Lemma 6.3 implies:

   $$\langle x?[f] \leq y?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

   In combination with $f\pi \in D_{\alpha(x)}$ and $x \in V$ the second clause of the definition of $D_{\alpha'(y)}$ yields $\pi \in D_{\alpha'(y)}$.

   (c) We assume $(\pi, a) \in L_{\alpha'(y)}$ and show $(f\pi, a) \in L_{\alpha(x)}$.

      i. Case (π, a) is contributed to $L_{\alpha'(y)}$ by the first clause of its definition. Thus $\langle a(y[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ such that $x[f]y$ implies $\langle a(x[f\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and thus, since $x \in V$, $\langle a(x[f\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$. Since α is a solution $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ we conclude $(f\pi, a) \in L_{\alpha(x)}$.

      ii. Case (π, a) is contributed to $L_{\alpha'(y)}$ by the second clause of its definition. There exist μ, μ', μ'' and a global variable $z \in V$ such that $\pi = \mu\mu''$ and:

      $$\langle z?[\mu'] \leq y?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi),\ (\mu'\mu'', a) \in L_{\alpha(z)}$$

      Due to Lemma 6.3 this implies $\langle z?[\mu'] \leq x?[f\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and hence, since $z, x \in V$, that $\langle z?[\mu'] \leq x?[f\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$. Since α is a solution of $\bigwedge \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ and $(\mu'\mu'', a) \in L_{\alpha(z)}$ we obtain that $(f\mu\mu'', a) = (f\pi, a) \in L_{\alpha(x)}$.

   (d) We assume $(f\pi, a) \in L_{\alpha(x)}$ and show $(\pi, a) \in L_{\alpha'(y)}$. The assumption $x[f]y \in \varphi$ and Lemma 6.3 imply

   $$\langle x?[f] \leq y?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

   In combination with $(f\pi, a) \in L_{\alpha(x)}$ the second clause of the definition of $L_{\alpha'(y)}$ yields $(\pi, a) \in L_{\alpha'(y)}$.

3. Case $x[f]y \in \varphi$, $x \notin V$ local, $y \in V$ global.

   From now on we only prove the assertions concerning the domains of the trees, the proofs for the labeling functions being analogous as we have seen in the case above. So, we have to prove the following equivalence for all π: $\pi \in D_{\alpha(y)}$ iff $f\pi \in D_{\alpha'(x)}$.

   (a) We assume $\pi \in D_{\alpha(y)}$ and show $f\pi \in D_{\alpha'(x)}$.
   Since $x[f]y \in \varphi$ we have that $\langle y?[\varepsilon] \leq x?[f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, hence $f\pi \in D_{\alpha'(x)}$ by the second clause of the definition of α'.

   (b) We assume $f\pi \in D_{\alpha'(x)}$ and show $\pi \in D_{\alpha(y)}$.

i. Suppose that $f\pi \in D_{\alpha'(x)}$ follows from the first clause of the definition of $D_{\alpha'(x)}$.

There exists y', u such that $\mathcal{A}_\varphi \vdash /x \xrightarrow{f} /y' \xrightarrow{\pi} u$. The closeness of φ under the decomposition rule F2 yields $y' \leq y \in \varphi$. Hence, $\mathcal{A}_\varphi \vdash /y \xrightarrow{\varepsilon} /y' \xrightarrow{\pi} u$, *i.e.* $\langle y[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, and hence $\pi \in \alpha(y)$.

ii. Suppose that $f\pi \in D_{\alpha'(x)}$ follows from the second clause of the definition of $D_{\alpha'(x)}$ Hence there exist $\mu, \mu'\mu''$ and $z \in V$ such that $f\pi = \mu\mu''$ and:

$$\langle z?[\mu'] \leq x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi), \mu'\mu'' \in D_{\alpha(z)}$$

According to Lemma 6.3 there exist a suffix $\tilde{\nu}$ of μ' and μ, say $\mu = \nu\tilde{\nu}$ and $\mu' = \nu'\tilde{\nu}$ and variables u, v such that:

$$\mathcal{A}_\varphi \vdash \backslash z \xrightarrow{\nu'} \backslash v \qquad \text{and} \qquad \mathcal{A}_\varphi \vdash /x \xrightarrow{\nu} /v \xrightarrow{\tilde{\nu}} /u$$

Since $f\pi = \nu\tilde{\nu}\mu''$ there are three cases, depending on whether the leading f in $f\pi$ belongs to ν, $\tilde{\nu}$ or to μ''.

A. Case $\nu = f\nu''$ for some ν''.
Since $f\pi = \nu\tilde{\nu}\mu''$ there is a y' such that

$$\mathcal{A}_\varphi \vdash /x \xrightarrow{f} /y' \xrightarrow{\nu''} /v$$

Due to the F2-closeness of φ and $x[f]y \in \varphi$ we obtain that $y' \leq y \in \varphi$. Hence, lemma 6.3 yields

$$\langle z?[\mu'] \leq y?[\nu''\tilde{\nu}]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

hence $\langle z?[\mu'] \leq y?[\nu''\tilde{\nu}]\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ due to $y, z \in V$. Since α is a solution of $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ and $\mu'\mu'' \in D_{\alpha(z)}$ we conclude $\pi = \nu''\tilde{\nu}\mu'' \in D_{\alpha(y)}$.

B. Case $\nu = \varepsilon$ and $\tilde{\nu} = f\tilde{\nu}'$ for some $\tilde{\mu}$.
Similar to the case above but with the decomposition rule F2 applied to $\tilde{\nu}$.

C. Case $\nu = \tilde{\nu} = \varepsilon$ and $\mu'' = f\tilde{\mu''}$.
In this case $\mu = \varepsilon$, hence $\langle z?[\mu'] \leq x?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and consequently by rule 3.10 of the automaton, $\langle z?[\mu' f] \leq x?[f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Hence, α is a solution of $\langle z?[\mu' f] \leq y?[\varepsilon]\rangle$, from which we conclude that $\pi \in D_{\alpha'(y)}$.

4. Case $x[f]y \in \varphi$, $x, y \notin V$ both local. We omit this case which is similar to the previous one.

5. Case $x \leq y \in \varphi$ and $x, y \in V$. This case is trivial since $\langle x?[\varepsilon] \leq y?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ and since α is a solution of $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$,

6. Case $x \leq y \in \varphi$ and $x \in V$, $y \notin V$. Let $\pi \in D_{\alpha'(y)}$. By construction of the automaton, $\langle x?[\varepsilon] \leq y?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, hence $\pi \in D_{\alpha'(y)}$ by the second clause of the definition of α'.

7. Case $x \leq y \in \varphi$ and $x \notin V$, $y \in V$. Let $\pi \in D_{\alpha'(x)}$ we have to show that $\pi \in D_{\alpha(y)}$.

   (a) If $\langle x[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ then, by construction of the automaton, $\langle y[\pi]\downarrow\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Hence, $\pi \in D_{\alpha(y)}$ since α is a solution of $\mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$.

   (b) If there is a $\langle z?[\mu'] \leq x?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ with $z \in V$, $\mu'\mu'' \in D_{\alpha(z)}$ and $\pi = \mu\mu''$, then we also have $\langle z?[\mu'] \leq y?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ by construction of the automaton, and hence $\langle z?[\mu'] \leq y?[\mu]\rangle \in \mathcal{L}(\mathcal{A}_\varphi^{\bar{x}})$ since $y, z \in V$. Since $\mu \in D_{\alpha'(x)}$ and since α' is a solution of $x?[\varepsilon] \leq y?[\varepsilon]$, we conclude that $\mu \in D_{\alpha'(y)} = D_{\alpha(y)}$, and hence $\pi \in D_{\alpha(y)}$.

8. Case $x{\leq}y \in \varphi$, $x \notin V$, $y \notin V$. Similar to the previous case.
9. Case $a(x) \in \varphi$, and $x \in V$. This is trivial since in this case $a(x) \in \mathcal{L}(\mathcal{A}_\varphi)$, and α is a solution of $\mathcal{L}(\mathcal{A}_\varphi)$ that coincides with α' on x since $x \in V$.
10. Case $a(x) \in \varphi$, and $x \notin V$. Omitted.

□

7 Completeness of the Entailment Test

We first recall some known results on simpler forms of entailment and then prove Proposition 7.3 from which the completeness of the entailment test follows.

7.1 Simpler Forms of Entailment

The following results on simpler forms of entailment from [18, 17] can be derived from the existence of least solutions for satisfiable constraints. These results will be used for proving the completeness of our entailment test as stated in Proposition 5.10.

Proposition 7.1 (Quantifier Free Entailment [18]) *If φ is* F*-closed then $\varphi \models x{\leq}y$ if and only if $x = y$ or $x{\leq}y \in \varphi$, and $\varphi \models x{\sim}y$ if and only if $x = y$ or $x{\sim}y \in \varphi$.*

Proof. Both properties are subsumed by Proposition 6 in [18]. □

Proposition 7.2 (Entailment of Simple Path Constraints [17]) *Let φ be satisfiable and* F*-closed. For every variable $x \in \mathcal{V}(\varphi)$ and all a, π the following two equivalences hold:*

1. $\varphi \models x[\pi]{\downarrow}$ *iff* $\langle x[\pi]{\downarrow}\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$
2. $\varphi \models a(x[\pi])$ *iff* $\langle a(x[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$

Proof. Modulo notation, this proposition is identical to Corollary 5.4 in [17]. Our notation $\langle x[\pi]{\downarrow}\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ used here is equivalent to the existence of z with $\varphi \vdash z?[\varepsilon]{\leq}x[\pi]$ in the notation of [17]. Similarly, $\langle a(x[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ is written $\varphi \vdash a(x[\pi])$ in the notation of [17]. □

7.2 The Completeness Proof

In this section we prove that our automaton construction and, as a consequence the entailment test, is complete.

Proposition 7.3 (Completeness) *For all constraints φ and path constraints ψ with $\mathcal{V}(\psi) \subseteq \mathcal{V}(\varphi)$ and $\mathcal{F}(\psi) \subseteq \mathcal{F}(\varphi)$:*

$$\textit{if } \varphi \models \psi \textit{ then } \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

The proof will proceed by induction on the length of the paths used in the path constraint ψ. In the induction step we will need to apply the induction hypothesis on path constraints with smaller paths. The idea is to remove a feature f at some position x of the path constraint ψ entailed by the constraint φ, thus yielding an extension φ^x_f of φ to be defined exactly in Definition 7.4. Then the induction roughly works as follows:

- From the hypothesis, conclude that the path constraint ψ' obtained by removing f at x from ψ is entailed by the extended constraint φ^x_f, thanks to Lemma 7.6.

- Apply the induction hypothesis to conclude that the path contraint with smaller paths ψ' is recognized by the automaton $\mathcal{A}_{\varphi_f^x}$ of the extended constraint.
- With a simple argument we will get that the original path constraint ψ is also recognized by the automaton $\mathcal{A}_{\varphi_f^x}$ of the extended constraint.
- Finally, we conclude that the original path constraint ψ is recognized by the automaton $\mathcal{A}_\varphi$ of the original constraint, thanks to Lemma 7.7.

Definition 7.4 (Constraint Extension) *For all constraints* φ *and features* f *fix a fresh variable* x_f *and define the* extension φ_f^x of φ at x and f *as follows:*

$$\begin{array}{rll}\varphi_f^x = \varphi \wedge x[f]x_f & \wedge & \bigwedge\{x_f{\leq}y \mid \exists x',y' : x \leq x' \wedge x'[f]y' \wedge y' \leq y \in \varphi\} \\ & \wedge & \bigwedge\{y{\leq}x_f \mid \exists x',y' : y \leq y' \wedge x'[f]y' \wedge x' \leq x \in \varphi\} \\ & \wedge & \bigwedge\{y{\sim}x_f, x_f{\sim}y \mid \exists x',y' : x \leq x' \wedge x'[f]y' \wedge y'{\sim}y \in \varphi\} \\ & \wedge & \bigwedge\{y{\sim}x_f, x_f{\sim}y \mid \exists x',y' : x{\sim}x' \wedge x'[f]y' \wedge y{\leq}y' \in \varphi\} \\ & \wedge & x_f \leq x_f \wedge x_f{\sim}x_f\end{array}$$

Lemma 7.5 *If* φ *is satisfiable and* F*-closed, then* φ_f^x *also is satisfiable and* F*-closed.*

Proof. Slightly tedious but straightforward. See Lemma 7 of [18] for the proof. □

Lemma 7.6 (Semantic Properties of Extensions) *For all* x, y, π, π', a, f*:*

1. *If* $\varphi \models x?[f\pi]{\sim}a$ *then* $\varphi_f^x \models x_f?[\pi]{\sim}a$
2. *If* $\varphi \models x?[f\pi]{\leq}y?[\pi']$ *then* $\varphi_f^x \models x_f?[\pi]{\leq}y?[\pi']$
3. *If* $\varphi \models x?[\pi]{\leq}y?[f\pi']$ *then* $\varphi_f^y \models x?[\pi]{\leq}y_f?[\pi']$
4. *If* $\varphi \models x?[f\pi]{\sim}y?[\pi']$ *then* $\varphi_f^x \models x_f?[\pi]{\sim}y?[\pi']$
5. *If* $\varphi \models x?[\pi]{\sim}y?[f\pi']$ *then* $\varphi_f^y \models x?[\pi]{\sim}y_f?[\pi']$

Proof. For illustration, we check only case 1.

$$\begin{array}{lll}\varphi \models x?[f\pi]{\sim}a & \text{implies} & \varphi \models \forall x_f(x[f]x_f \rightarrow x_f?[\pi]{\sim}a) \\ & \text{implies} & \varphi \wedge x[f]x_f \models x_f?[\pi]{\sim}a \\ & \text{implies} & \varphi_f^x \models x_f?[\pi]{\sim}a\end{array}$$

The other cases are proven analogously. □

Lemma 7.7 *For all* φ *and path constraints* ψ *of one of the forms* $u_1?[\pi_1]{\leq}u_2?[\pi_2]$, $u_1?[\pi_1]{\sim}u_2?[\pi_2]$*, or* $u?[\pi]{\sim}a$*, and with* $\mathcal{V}(\psi) \subseteq \mathcal{V}(\varphi)$ *and* $\mathcal{F}(\psi) \subseteq \mathcal{F}(\varphi)$ *:*

$$\textit{If } \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_{\varphi_f^x}) \textit{ then } \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

This statement is repeated as Lemma 7.9 in Section 7.3 and proved there.

Proof of Proposition 7.3 The proof is by case distinction over ψ.

1. $\psi = x[\pi]{\downarrow}$: If $\varphi \models x[\pi]{\downarrow}$ then $\langle x[\pi]{\downarrow}\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ by Proposition 7.2.(1).
2. $\psi = a(x[\pi])$: If $\varphi \models a(x[\pi])$ then $\langle a(x[\pi])\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ by Proposition 7.2.(2).

3. $\psi = x?[\pi]{\sim}a$: We assume $\langle x?[\pi]{\sim}a\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$ and show $\varphi \not\models x?[\pi]{\sim}a$ by induction on π.

   $\pi = \varepsilon$: We distinguish three cases, depending on the number n of distinct labels c such that $\langle x?[\varepsilon]{\sim}c\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.

   $n = 0$: Let b be an arbitrary label distinct from a. Then clearly $\varphi \wedge b(x)$ is F-closed and satisfiable and entails $\neg x?[\varepsilon]{\sim}a$. Hence $\varphi \not\models x?[\varepsilon]{\sim}a$.

   $n = 1$: Let b be the unique label such that $\langle x?[\varepsilon]{\sim}b\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. The assumption $\langle x?[\pi]{\sim}a\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$ implies $a \neq b$. Clearly, $\varphi \wedge b(x)$ is satisfiable and entails $\neg x?[\varepsilon]{\sim}a$. Hence $\varphi \not\models x?[\varepsilon]{\sim}a$.

   $n \geq 2$: This case is impossible under our assumption that $\langle x?[\varepsilon]{\sim}a\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$, since the automaton has the property that whenever $\langle x?[\varepsilon]{\sim}b\rangle, \langle x?[\varepsilon]{\sim}c\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ for $b \neq c$ then $\langle x?[\varepsilon]{\sim}a\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ for all labels a.

   $\pi = f\pi'$: By Lemma 7.7, $\langle x?[\pi]{\sim}a\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$, and hence $\langle x_f?[\pi']{\sim}a\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$. By induction assumption this implies $\varphi_f^x \not\models x_f?[\pi']{\sim}a$ and hence, by clause 1 of Lemma 7.6, $\varphi \not\models x?[f\pi']{\sim}a$, that is, $\varphi \not\models x?[\pi]{\sim}a$.

4. $\psi = x?[\pi]{\leq}y?[\pi']$: Assume that $\langle x?[\pi]{\leq}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$. We show $\varphi \not\models x?[\pi]{\leq}y?[\pi']$ by simultaneous induction over π and π'.

   $\pi = \pi' = \varepsilon$: Then $\psi = x?[\varepsilon]{\leq}y?[\varepsilon]$ is equivalent to the basic constraint $x{\leq}y$. Since $x, y \in \mathcal{V}(\varphi)$ and φ is F-closed, Proposition 7.1 implies that $\varphi \not\models x{\leq}y$ if and only if $x{\leq}y \notin \varphi$. However, $\langle x?[\varepsilon]{\leq}y?[\varepsilon]\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$ implies $x{\leq}y \notin \varphi$, again due to F-closedness, and hence $\varphi \not\models x{\leq}y$.

   $\pi = f\pi''$: By Lemma 7.7, $\langle x?[f\pi'']{\leq}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$ and hence $\langle x_f?[\pi'']{\leq}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$. By induction assumption this implies $\varphi_f^x \not\models x_f?[\pi'']{\leq}y?[\pi']$ and hence $\varphi \not\models x?[f\pi'']{\leq}y?[\pi']$ by Lemma 7.6, clause 2.

   $\pi' = f\pi''$: Symmetric, using clause 3 of Lemma 7.6.

5. $\psi = x?[\pi]{\sim}y?[\pi']$: Assume that $\langle x?[\pi]{\sim}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$. We show $\varphi \not\models x?[\pi]{\sim}y?[\pi']$ by simultaneous induction over π and π'.

   $\pi = \pi' = \varepsilon$: Then $\psi = x?[\varepsilon]{\sim}y?[\varepsilon]$ is equivalent to the basic constraint $x{\sim}y$. Since $x, y \in \mathcal{V}(\varphi)$ and φ is F-closed, Proposition 7.1 implies that $\varphi \not\models x{\sim}y$ if and only if $x{\sim}y \notin \varphi$. However, $\langle x?[\varepsilon]{\sim}y?[\varepsilon]\rangle \notin \mathcal{L}(\mathcal{A}_\varphi)$ implies $x{\sim}y \notin \varphi$, hence $\varphi \not\models x{\sim}y$.

   $\pi = f\pi''$: By Lemma 7.7, $\langle x?[f\pi'']{\sim}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$, hence $\langle x_f?[\pi'']{\sim}y?[\pi']\rangle \notin \mathcal{L}(\mathcal{A}_{\varphi_f^x})$. By induction assumption this yields $\varphi_f^x \not\models x_f?[\pi'']{\sim}y?[\pi']$ and, by Lemma 7.6, clause 4, $\varphi \not\models x?[f\pi'']{\sim}y?[\pi']$.

   $\pi' = f\pi''$: Symmetric, using clause 5 of Lemma 7.6.

□

7.3 Details of the Completeness Proof

Lemma 7.8 *Let φ be an F-closed constraint, $u_0, u_n \in \mathcal{V}(\varphi)$, $x \in \mathcal{V}$ a variable, $f \in \mathcal{F}$ a feature, and $\pi_1, \pi_2, \pi_3 \in \mathcal{F}(\varphi)^*$ paths with features from φ.*

1. *If there exists a variable $u_{i_1} \in \mathcal{V}(\varphi_f^x)$ such that the extension φ_f^x satisfies*

$$\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1} \qquad \text{and} \qquad \mathcal{A}_{\varphi_f^x} \vdash /u_n \xrightarrow{\pi_2} /u_{i_1}$$

   then $\langle u_0?[\pi_1]{\leq}u_n?[\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ holds for the nonextended constraint φ.

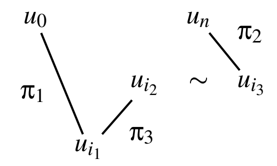


Fig. 13: The transitions used in the proof of Lemma 7.8.

2. *If there exists a variable $u_{i_1} \in \mathcal{V}(\varphi^x_f)$ such that the extension φ^x_f satisfies*

$$\mathcal{A}_{\varphi^x_f} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1} \qquad \textit{and} \qquad \mathcal{A}_{\varphi^x_f} \vdash \backslash u_n \xrightarrow{\pi_2\pi_3} \rangle u_{i_1}$$

then $\langle u_0?[\pi_1]{\sim}u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ holds for the nonextended constraint φ.

Proof. The proof of both parts are so similar that we can savely restrict ourselves to show the slightly more envolved part 2.

We can decompose the transitions into (see Figure 13)

$$\begin{array}{ll} \mathcal{A}_{\varphi^x_f} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}, & \mathcal{A}_{\varphi^x_f} \vdash \backslash u_n \xrightarrow{\pi_2} \backslash u_{i_3}, \\ \mathcal{A}_{\varphi^x_f} \vdash /u_{i_2} \xrightarrow{\pi_3} /u_{i_1}, & u_{i_2}{\sim}u_{i_3} \in \varphi^x_f. \end{array}$$

Let $\bar{q}$ be the sequence of states of the transition $\mathcal{A}_{\varphi^x_f} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$, followed by the states of the transition $\mathcal{A}_{\varphi^x_f} \vdash /u_{i_2} \xrightarrow{\pi_3} /u_{i_1}$ in reverse order, and without the last state $/u_{i_1}$, followed by the sequence of states of the transition $\mathcal{A}_{\varphi^x_f} \vdash \backslash u_n \xrightarrow{\pi_2} \backslash u_{i_3}$ in reverse order. Hence the first state of $\bar{q}$ is $\backslash u_0$, the last state is $\backslash u_n$, and the length of $\bar{q}$ is the length of the three transition sequences plus 2 (since we ignored one occurrence of $/u_1$).

Let $(u_0,\ldots,u_n)$ be the sequence of variables occuring as first variable-symbol in the states of $\bar{q}$. Thus, the length of this sequence, $n+1$, is equal to the length of $\bar{q}$, and u_0 and u_n are those of the lemma. Furthermore, let i_1, i_2 and i_3 the indices corresponding to the above decomposition of the transitions (see also Figure 13). This sequence of variables is exactly the sequence of variables that we encounter in the graph of Figure 13 when, starting from u_0, we go down π_1, then up π_3 to u_{i_2}, over to u_{i_3} and then up to u_n.

We prove $\langle u_0?[\pi_1]{\sim}u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ by well-founded induction on the lexicographic order on the triple of natural numbers (m_a,m_b,m_b) defined below.

m_a is the length of $(u_{i_3},\ldots,u_n)$, i.e. $n-i_3+1$.

m_b is the number of occurrences of x_f in $(u_0,\ldots,u_n)$.

m_b is the length of $(u_0,\ldots,u_n)$, i.e. $n+1$.

The idea of the proof is to remove either u_0 or else the left-most occurrence of x_f from the sequence $(u_0,\ldots,u_n)$. The induction step requires a number of case distinctions on the shape of the given transitions.

1. Case $i_2 = 0$.

   (a) Case $i_3 = n$. Thus $u_0{\sim}u_n \in \varphi^x_f$. Since $u_0, u_n \in \mathcal{V}(\varphi)$ it follows that $u_0{\sim}u_n \in \varphi$. The F-closedness of φ yields $\langle u_0?[\pi_1]{\sim}u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ since $\pi_1 = \pi_2 = \pi_3 = \varepsilon$ in this case.

(b) Case $i_3 < n$. Since $\pi_1 = \pi_3 = \varepsilon$, we can apply the induction hypothesis to the situation where u_0 and u_n are swapped whereby the measure m_a is strictly decreased. Hence, $\langle u_n?[\pi_2]{\sim}u_0?[\varepsilon]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ and by symmetry of the automaton construction $\langle u_0?[\varepsilon]{\sim}u_n?[\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.

2. Case $i_2 \neq 0$.

(a) Case $u_1 \in \mathcal{V}(\varphi)$. We can apply the induction hypothesis to $(u_1,\dots,u_n)$ since the measure m_b is decreased properly whereas m_a and m_b remain unchanged. For all constraints that might connect u_0 to u_1, the expected conclusion can be composed from the induction hypothesis and Lemma 6.3.

(b) Case $u_1 = x_f$. We can assume the transition sequence $\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$ to be non-empty since it is not possible to have $x_f[f]u_0$ (by construction of φ_f^x). Hence, should this transition be empty the relation between u_0 and u_1 is $u_0{\leq}u_1$ we can take this as part of$\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$. We distinguish two subcases: the transition $\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$ relates u_0 to u_1 either via an ordering or a feature selection constraint.

i. Case $u_0{\leq}x_f \in \varphi_f^x$. We further distinguish on which constraint connects x_f to u_2, either in $\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$ or in $\mathcal{A}_{\varphi_f^x} \vdash / u_{i_2} \xrightarrow{\pi_3} / u_{i_1}$.

A. Case $x_f[g]u_2 \in \varphi_f^x$ for some feature $g \in \mathcal{F}$. This is impossible since φ_f^x does not contain any selection constraints headed by x_f.

B. Case $x_f{\leq}u_2 \in \varphi_f^x$. Transitivity (F1.2-closure of φ) yields $u_0{\leq}u_2 \in \varphi$. Thus, we can cancel out u_1 from the sequence: the induction hypothesis applied to $(u_0,u_2,\dots,u_n)$ proves $\langle u_0?[\pi_1]{\sim}u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ as required.

C. Case $u_2[g]x_f \in \varphi_f^x$ for some $g \in \mathcal{F}$. Note that this situation may happen for $i_1 = 1$ and $i_2 > 1$. By construction of φ_f^x we know $u_2 = x$ and $g = f$. Since $u_0 \in \mathcal{V}(\varphi)$ and $u_0{\leq}x_f \in \varphi_f^x$, the construction of φ_f^x ensures the existence of variables $x_1,x_2 \in \mathcal{F}$ such that $x_1{\leq}x \wedge x_1[f]x_2 \wedge u_0{\leq}x_2 \in \varphi$. We can now cancel out the occurrence of x_f at u_1: the induction hypothesis applies to the sequence $(u_0,x_2,x_1,x,u_3,\dots,u_n)$ since m_a remains unchanged whereas m_b is decreased by 1. Thereby the lexicographic order on (m_a,m_b,m_b) is decreased properly even though m_b might be increased.

D. Case $x_f{\sim}u_2 \in \varphi_f^x$. This may happen but only if $i_1 = i_2 = 1$ and $i_3 = 2$. Since φ_f^x is F3.2-closed by Lemma 7.5, $u_0{\leq}x_f \wedge x_f{\sim}u_2 \in \varphi_f^x$ implies $u_0{\sim}u_2 \in \varphi_f^x$. Again, we can cancel out the occurrence of x_f at u_1.

ii. Case $u_0[g]x_f \in \varphi_f^x$. Thus, $u_0 = x$ and $g = f$ by construction of φ_f^x. We distinguish further depending on which constraint connects x_f to u_2 in either $\mathcal{A}_{\varphi_f^x} \vdash \backslash u_0 \xrightarrow{\pi_1} \backslash u_{i_1}$ or $\mathcal{A}_{\varphi_f^x} \vdash / u_{i_1} \xrightarrow{\pi_2} / u_{i_2}$.

A. The case $x_f[g]u_2 \in \varphi_f^x$ for some feature $g \in \mathcal{F}$ is impossible as φ_f^x does not contain any selection constraints headed by x_f.

B. Case $x_f{\leq}u_2 \in \varphi_f^x$. If $u_2 = x_f$ then we trivially cancel out the occurrence of x_f at u_2. Otherwise, $u_2 \in \mathcal{V}(\varphi)$. The construction of φ_f^x yields that there exist variables x_1,x_2 satisfying $x{\leq}x_2 \wedge x_1[f]x_2 \wedge x_1{\leq}u_2 \in \varphi$. Hence, we can apply the induction hypothesis to $(x,x_2,x_1,u_2,\dots,u_n)$.

C. Case $u_2[g]x_f \in \varphi_f^x$ for some $g \in \mathcal{F}$. Thus, $i_1 = 1$, $i_2 > 1$, $u_2 = x$, $f = g$, and $\pi_1 = f$ and there exists a path π_3' with $\pi_3 = \pi_3' f$. The induction hypothesis applied to $(u_0,u_2,\dots,u_n)$ yields $\langle x?[\varepsilon]{\sim}u_n?[\pi_2\pi_3']\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Due to the assumption $\pi_1 \in \mathcal{F}(\varphi)^*$ of the Lemma, it follows that $f \in$

$\mathcal{F}(\varphi)$ such that f belongs to the signature of the automaton $\mathcal{A}_\varphi$. Thus, $\langle x?[f]\sim u_n?[\pi_2\pi_3' f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$, i.e. $\langle u_1?[\pi_1]\sim u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.

D. Case $x_f\sim u_2 \in \varphi_f^x$. Now, we have $i_1 = i_2 = 1$, $i_3 = 2$, $\pi_1 = f$, and $\pi_3 = \varepsilon$. There are again two sub-cases which we consider below:

Subcases of 2.(b).ii.D:

1. Case $u_2 = x_f$. From $u_n \in \mathcal{V}(\varphi)$ it follows that $n \geq 3$. We distinguish further, according to the constraint which connects u_3 to x_f in $\mathcal{A}_{\varphi_f^x} \vdash \backslash u_n \xrightarrow{\pi_2} \backslash u_{i_3} (= \backslash u_2)$.

   (a) Case $u_3 \leq x_f \in \varphi_f^x$. The F3.2-closedness of φ_f^x entails $u_3 \leq x_f \in \varphi_f^x$ (Lemma 7.5). We cancel out the occurrence of x_f at u_2 by applying the induction hypothesis to the sequence $(u_0, u_1, u_3, \ldots, u_n)$.

   (b) Case $u_3[g]x_f \in \varphi_f^x$ for some $g \in \mathcal{F}$. Hence, $u_3 = x$, $g = f$, and there exists π_2' such that $\pi_2 = \pi_2' f$. Now, we can cancel out both occurrences of x_f at u_2 and at u_3: the induction hypothesis applied $(x, u_3, \ldots, u_n)$ yields $\langle x?[\varepsilon]\sim u_n?[\pi_2']\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$. Since $\pi_2 \in \mathcal{F}(\varphi)$ is assumed by the Lemma, we know that $f \in \mathcal{F}(\varphi)$. Thus $\langle x?[f]\sim u_n?[\pi_2' f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ from Lemma 6.3, i.e., $\langle x?[\pi_1]\sim u_n?[\pi_2\pi_3]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$.

2. Case $u_2 \neq x_f$. Thus, $u_2 \in \mathcal{V}(\varphi)$ and $x_f\sim u_2 \in \varphi_f^x$. We distinguish two possibilities for this to happen, according to the construction of φ_f^x.

   (a) Case $x \leq x_1 \wedge x_1[f]x_2 \wedge x_2\sim u_2 \in \varphi$ for some variables x_1, x_2. The induction hypothesis applied to the sequence $(x, x_1, x_2, u_3, \ldots, u_n)$ yields $\langle x?[f]\sim u_n?[\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ as required.

   (b) Case $x\sim x_1 \wedge x_1[f]x_2 \wedge u_2 \leq x_2 \in \varphi$ for some variables x_1, x_2. The induction hypothesis can be applied to the situation where u_0 and u_n are swapped (note that $\pi_3 = \varepsilon$) whereby m_a does not increase (it has been strictly positive before, and now is 1) and m_b decreases properly (since the occurrence of x_f at u_1 is eliminated). The induction hypothesis yields $\langle u_n?[\pi_2]\sim x?[f]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ such that symmetry implies $\langle x?[f]\sim u_n?[\pi_2]\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$ as required.

□

Lemma 7.9 *Let* ψ *be a path constraint of one of the forms* $u_1?[\pi_1] \leq u_2?[\pi_2]$, $u_1?[\pi_1]\sim u_2?[\pi_2]$, *or* $u?[\pi]\sim a$. *For all* φ *with* $\mathcal{V}(\psi) \subseteq \mathcal{V}(\varphi)$ *and* $\mathcal{F}(\psi) \subseteq \mathcal{F}(\varphi)$*:*

$$\textit{If } \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_{\varphi_f^x}) \textit{ then } \langle\psi\rangle \in \mathcal{L}(\mathcal{A}_\varphi)$$

Proof. From Lemmas 7.8 and 6.3. □

Acknowledgments. The research reported in this paper has been supported by the BMBF (FKZ ITW 9601), the Esprit Working Group CCL II (EP 22457), and the DFG (SFB 378). Special thanks are due to Denys Duchier for providing the initial zigzag macros. We thank the anonymous referees for their pertinent remarks.

References

[1] H. Aït-Kaci and A. Podelski. Towards a meaning of Life. *The Journal of Logic Programming*, 16(3 and 4):195–234, July, Aug. 1993.

[2] H. Aït-Kaci, A. Podelski, and G. Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1–2):263–283, Jan. 1994.

[3] R. Backofen. A complete axiomatization of a theory with feature and arity constraints. *The Journal of Logic Programming*, 24(1&2):37–71, 1995. Special Issue on Computational Linguistics and Logic Programming.

[4] R. Backofen and G. Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1–2):243–268, July 1995.

[5] F. Bourdoncle and S. Merz. Type checking higher-order polymorphic multi-methods. In *Proceedings24th ACM Symposium on Principles of Programming Languages*, pages 302–315, Paris, France, Jan. 1997. ACM Press, New York.

[6] B. Carpenter. *The Logic of Typed Feature Structures - with Applications to Unification Grammars, Logic Programs and Constraint Resolution*. Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, England, 1992.

[7] J. Dörre. Feature logics with weak subsumption constraints. In *Annual Meeting of the Association of Computational Linguistics*, pages 256–263, 1991.

[8] J. Dörre and W. C. Rounds. On subsumption and semiunification in feature algebras. In *5th IEEE Symposium on Logic in Computer Science*, pages 300–310. IEEE Computer Society Press, 1990.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[10] F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *Proceedings12th IEEE Symposium on Logic in Computer Science*, Warsaw, Poland, 1997. IEEE Computer Society Press.

[11] F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In K. Larsen, editor, *25th International Conference on Automata, Languages, and Programming*, Lecture Notes in Computer Science, Aalborg, Denmark, 1998. Springer-Verlag, Berlin, Germany.

[12] R. M. Kaplan and J. Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA, 1982.

[13] R. T. Kasper and W. C. Rounds. A logical semantics for feature structures. In *Annual Meeting of the Association of Computational Linguistics*, pages 257–265, 1986.

[14] M. Kay. Functional grammar. In C. Chiarello et al., editor, *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*, pages 142–158, 1979.

[15] M. J. Maher. Logic semantics for a class of committed-choice programs. In J.-L. Lassez, editor, *International Conference on Logic Programming*, pages 858–876. The MIT Press, Cambridge, MA, 1987.

[16] S. Marlow and P. Wadler. A practical subtyping system for Erlang. In *Proceedings2 nd ACM SIGPLAN International Conference on Functional Programming*, pages 136–149. ACM Press, New York, 1997.

[17] M. Müller and J. Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. *Information and Computation*, 159, May 5, 2000. Special Issue on RTA'98.

[18] M. Müller, J. Niehren, and A. Podelski. Ordering constraints over feature trees. *Constraints, an International Journal*, 5(1–2):7–42, Jan. 2000. Special Issue on CP'97.

[19] M. Müller. Ordering constraints over feature trees with ordered sorts. Technical report, Universität des Saarlandes, Programming Systems Lab, Saarbrücken, Germany, 1997.

[20] M. Müller. *Set-based Failure Diagnosis for Concurrent Constraint Programming*. PhD thesis, Universität des Saarlandes, Fachbereich Informatik, Saarbrücken, Germany, Jan. 1998.

[21] M. Müller, J. Niehren, and R. Treinen. The first-order theory of ordering constraints over feature trees. In V. Pratt, editor, *13$^{\text{th}}$ Annual IEEE Symposium on Logic in Computer Science*, pages 432–443, Indianapolis, IN, USA, June 1998. IEEE Computer Society Press.

[22] J. Palsberg. Efficient inference of object types. In *9th IEEE Symposium on Logic in Computer Science*, pages 186–185. IEEE Computer Society Press, 1994.

[23] C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Cambridge University Press, Cambridge, England, 1994.

[24] F. Pottier. Simplifying subtyping constraints. In *ACM SIGPLAN International Conference on Functional Programming*, pages 122–133. ACM Press, New York, May 1996.

[25] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[26] W. C. Rounds. Feature logics. In J. v. Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers B.V. (North Holland), 1997.

[27] V. A. Saraswat. *Concurrent Constraint Programming*. The MIT Press, Cambridge, MA, 1993.

[28] S. Shieber. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes No. 4. Center for the Study of Language and Information, 1986.

[29] G. Smolka. Feature constraint logics for unification grammars. *The Journal of Logic Programming*, 12:51–87, 1992.

[30] G. Smolka. The Oz Programming Model. In J. van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, Germany, 1995.

[31] G. Smolka and R. Treinen. Records for logic programming. *The Journal of Logic Programming*, 18(3):229–258, Apr. 1994.

[32] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, August 1968. Published by Springer-Verlag NY Inc.

[33] R. Treinen. A new method for undecidability proofs of first order theories. *Journal of Symbolic Computation*, 14:437–457, 1992.

[34] R. Treinen. Feature trees over arbitrary structures. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, chapter 7, pages 185–211. CSLI Publications and FoLLI, 1997.

[35] V. Trifonov and S. Smith. Subtyping constrained types. In R. Cousot and D. A. Schmidt, editors, *Proceedings3rd International Static Analysis Symposium*, volume 1145 of *Lecture Notes in Computer Science*, pages 349–365, Aachen, 1996. Springer-Verlag, Berlin, Germany.

[36] P. Van Roy, M. Mehl, and R. Scheidhauer. Integrating efficient records into concurrent constraint programming. In *International Symposium on Programming Language Implementation and Logic Programming*, Aachen, Germany, Sept. 1996. Springer-Verlag.

[37] S. Vorobyov. An improved lower bound for the elementary theories of trees. In *Procedings of the International Conference on Automated Deduction (CADE)*, volume 1104 of *Lecture Notes in Computer Science*, pages 275–287, Rutgers, NJ, 1996. Springer-Verlag, Berlin, Germany.