Discrete Mathematics and Theoretical Computer Science **5**, 2002, 227–262

A Unified Framework to Compute over Tree Synchronized Grammars and Primal Grammars

Frédéric Saubion and Igor Stéphan

LERIA, Université d'Angers, 2, Bd Lavoisier
49045 Angers Cedex 01, France
{`Frederic.Saubion,Igor.Stephan`}`@univ-angers.fr`

received Mar 27, 2001, *accepted Nov 17, 2002*.

Tree languages are powerful tools for the representation and schematization of infinite sets of terms for various purposes (unification theory, verification and specification ...). In order to extend the regular tree language framework, more complex formalisms have been developed. In this paper, we focus on Tree Synchronized Grammars and Primal Grammars which introduce specific control structures to represent non regular sets of terms. We propose a common unified framework in order to achieve the membership test for these particular languages. Thanks to a proof system, we provide a full operational framework, that allows us to transform tree grammars into Prolog programs (as it already exists for word grammars with DCG) whose goal is to recognize terms of the corresponding language.

Keywords: Tree Grammars, Proof Systems, Prolog Implementation

1 Introduction

Tree languages [3, 6] have been extensively studied and have many applications in various areas such as term rewriting, term schematization, specification and verification ... Languages can be handled either from the generation point of view (i.e. grammars) or from the recognition point of view (i.e. automata).
Regular tree languages [3, 6] are very close to regular word grammar with tree (ie terms[†]) as basic objects. They have nice closure properties (concerning complementation, union and intersection) and the membership test and the emptiness test are decidable. Moreover they can be defined using a notion of finite automata which can be easily used to test the membership of a term to a given language. Therefore, from a practical point of view, tree automata appear to be easily implementable tools for recognition. But, these notions are not sufficient to describe more complex sets of terms (non regular tree languages).
In this paper, we focus on two particular non regular classes of tree languages defined by their associated notion of tree grammars: Tree Synchronized Grammars and Primal Grammars. The purpose of this paper is to define a uniform and operational framework to describe and implement a membership test procedure for these languages.

[†] In the following and without any loss of generality, we consider always terms as the basic elements of tree languages

Tree Synchronized Grammars (*TSG*) [13] and Primal Grammars (*PG*) [11] are issued from the E-unification framework. E-unification [17] is known to be undecidable in general, but, as explained in [14], some decidable classes can be characterized by using such tree languages. Due to their specific definitions introducing control in the derivation process, a notion of automata can not be easily derived for these two classes of tree languages. Their expressive power can be highlighted by the two following simple examples.

Example 1 (*TSG*) We want to define the language corresponding to $\{f(s^n(z), s^{2n}(z)) \mid n \in \mathbb{N}\}$ which is obviously not a regular language.
The language description problem lies in the fact that, informally, each time a function symbol s is produced in the first argument of f one has to produce two in the second argument. In fact, the growth of the first argument has to be synchronized with the second. Tree Synchronized Grammars take this control into account by allowing packs of synchronized production rules instead of usual single grammar productions rules.

Example 2 (*PG*) In this second example, we want to describe the language of all the integer lists $[n, \ldots, 0]$. Integer are represented with z (for the zero integer) and successor function s and lists are built thanks to the binary constructor $*$ and the empty list $\perp$ (i.e. the list $[2,1,0]$ is obtained as $*(s(s(z)), *(s(z), *(z, \perp)))$. This language is also not regular.

The main problem here is the relation between the different arguments of the list you are constructing. Once you have generated an element $s^n(z)$, next element of the list must be $s^{n+1}(z)$. It appears that you have to count the depth of successive argument. By introducing counters variables in the production rules, Primal Grammars are able to handle this language.
One has to remark that the language of the first example cannot be generated using *PG* while the language of the second example cannot be described by a *TSG*. Due to these two specific control mechanisms (synchronization and counter variables), the standard notion of automata does not appear clearly.
Inspired by the implementation of Definite Clause Grammars [2] that allows to write word grammars as Horn clauses in a Prolog environment, our aim is to propose a similar approach for these particular tree languages. The key idea is to define a proof framework to describe the behavior of tree grammars and therefore to provide a proof theoretical semantics for grammar derivations. In this context, we change from syntactic generation of trees to logical deduction (i.e. productions rules are translated into logical formulas and grammar derivations into logical inferences). Then, the membership test, for a language described by a grammar, just consists in proving a particular formula in a sequent calculus using a proof system. This approach provides a uniform framework for the definition and use of both type of grammars. The method can be briefly described as follows;
Production rules are translated into logical formulas built over linear logic syntax [8] to handle the problem of synchronization (this idea was already investigated in [10]. Then, a sequent calculus proof system, inspired by the proof system of D. Miller [16], allows us to translate derivations into proof searches. The same job can be adapted for Primal Grammars.
The equivalence between the notion of grammar computation and our notion of proof is established by a correctness and completeness result. The last part of our work consists in refining this initial proof system

in order to introduce a notion of strategy in the proof search and thus to get a goal directed procedure. At this step, both transformation function and logical inference system can be easily translated into Prolog Horn clauses. Of course, it should be noticed that our system also works for Regular Tree Languages, since they are included in *TSG*. Note that an alternative algorithm has been proposed for the membership test of a TTSG in [9].

This paper is an extension of the previous work [18] and is organized as follows. Section 2 presents basic definitions of Tree Synchronized Grammars, Primal Grammars and proof systems. Section 3 described how the different formalism can be translated into linear logic formulas. The proof system is defined and refined in Section 4. Section 5 is the implementation issue of our work and we conclude in Section 6.

2 Preliminaries

In this paper, we recall here some notions about tree grammars and linear logic proof systems. We refer the reader to [3, 6, 7, 8, 13, 16] for more details.

We first introduce the two specific grammar formalisms we deal with in this paper. The original formalisms have been adapted to unify and simplify the presentation.

2.1 Tree Synchronized Grammars

Let $\mathcal{L}$ be a finite set of symbols (a signature), $T(\mathcal{L})$ denotes the first-order algebra of ground terms over $\mathcal{L}$. $\mathcal{L}$ is partitioned in two parts: the set $\mathcal{F}$ of terminal symbols, and the set $\mathcal{N}$ of non terminal symbols. Upper-case letters denote elements of $\mathcal{N}$. $t|u$ denotes the subterm of t at occurrence u and $t[u \leftarrow v]$ denotes the term obtained by replacing in t the subterm $t|u$ by v. $t[v]$ denotes a term containing a subterm v at a particular occurrence and $O(t)$ denotes the set of occurrences in t. $C(X_1,...,X_n)$ denotes a term with occurrences: $\{1...n\}$ such that $C(X_1,...,X_n)|i = X_i$. C will be called a context. We first define the notion of productions for *TSG*. We require that $\mathcal{F} \cap \mathcal{N} = \emptyset$ and that each element of $\mathcal{F} \cup \mathcal{N}$ has a fixed arity. We refer the reader to [4] for more details on these basic notions.

Definition 1 (Productions) *A production is a rule of the form* $X \Rightarrow t$ *where* $X \in \mathcal{N}$ *and* $t \in T(\mathcal{L})$ *. A pack of productions is a set of productions denoted* $\{X_1 \Rightarrow t_1, \ldots, X_n \Rightarrow t_n\}$.

- *When the pack is a singleton of the form* $\{X_1 \Rightarrow C(Y_1, \ldots, Y_n)\}$ *where* C *is a context of terminal symbols and* $Y_1, \ldots, Y_n$ *non terminals. The production is said free, and is written more simply* $X_1 \Rightarrow C(Y_1, \ldots, Y_n)$.
- *When the pack is of the form* $\{X_1 \Rightarrow Y_1, \ldots, X_n \Rightarrow Y_n\}$ *where* $Y_1, \ldots, Y_n$ *are terminals or non terminals. The productions of the pack are said synchronized.*

We can then define the notion of *TSG*.

Definition 2 (*TSG*) *A TSG is defined by a 4-tuple* $(\mathcal{F}, \mathcal{N}, PP, TI)$ *where*

- $\mathcal{F}$ *is the set of terminals,*
- $\mathcal{N}$ *is the finite set of non terminals,*
- *PP is a finite set of packs of productions,*

- *TI is the axiom of the TSG denoted $(I,\#)$ where I is a non terminal and # a new symbol added to the signature.*

Note that, the axiom is a pair due to the control of synchronizations. Basically, counters are associated to non terminal symbols to ensure the integrity of the application of synchronized productions.

Back to example 1, we wanted to define the language corresponding to $\{f(s^n(z),s^{2n}(z)) \mid n \in \mathbb{N}\}$. The corresponding *TSG* consists of the following production rules:

$$\begin{array}{l} R_0 : I \Rightarrow f(X,Y) \\ R_1 : X \Rightarrow z \\ R_2 : Y \Rightarrow z \\ P_1 : \{R_3 : X \Rightarrow s(X), R_4 : Y \Rightarrow s(s(Y))\} \end{array}$$

where X,Y are non terminal symbols and f,s,z terminal symbols. $P_1 : \{R_3,R_4\}$ denotes that, using the pack of production P_1, R_3 and R_4 have to be applied at the same time (synchronization).

Definition 3 (Computations of a *TSG*) *The set of computations of a TSG $Gr = (\mathcal{F},\mathcal{N},PP,TI)$, denoted $Comp(Gr)$, is the smallest set defined by:*

- *TI is in $Comp(Gr)$,*
- *if t is in $Comp(Gr)$, $t|_u = (X,c)$ and the free production $X \Rightarrow C(Y_1,\ldots,Y_n)$ is in PP then $t[u \leftarrow C((Y_1,c),\ldots,(Y_n,c))]$ is in $Comp(Gr)$,*
- *if t is in $Comp(Gr)$ and there exists n pairwise different occurrences $u_1,\ldots,u_n$ of t such that $\forall i \in [1,n]$ $t|_{u_i} = (X_i,c_i)$ and $c_i = a$ and the pack of productions $\{X_1 \Rightarrow Y_1,\ldots,X_n \Rightarrow Y_n\} \in PP$, then $t[u_1 \leftarrow (Y_1,b)]\ldots[u_n \leftarrow (Y_n,b)]$ (where b is a new symbol) is in $Comp(Gr)$.*

The symbol $\Rightarrow$ denoting also the above two deduction steps, a derivation of Gr is a sequence of computations $TI \Rightarrow t_1 \Rightarrow \ldots \Rightarrow t_n$.

Always following Example 1, one possible derivation of this grammar is thus (where a is a new symbol):

$$(I,\#) \Rightarrow_{R_0} f((X,\#),(Y,\#)) \Rightarrow_{P_1} f(s((X,a)),s(s((Y,a))))$$
$$\Rightarrow_{R_1} f(s(z),s(s(Y,a))) \Rightarrow_{R_2} f(s(z),s(s(z)))$$

As mentioned in the introduction, counters are introduced to control the application of the synchronized production rules. The previous definition imposes that only non terminals having the same control symbol can be used in a synchronized production. It should be noticed that *TSG* were originally defined using tuple of counters. Here, as we already did, concerning tuples of terms in the definition of the axiom, we consider a single counter to control synchronization in order to simplify the presentation of the basic concepts. The second step of this derivation clearly illustrates the notion of synchronized production rules.

Definition 4 (Recognized Language) *The language recognized by a TSG Gr, denoted $Rec(Gr)$, is the set of trees composed of terminal symbols $Comp(Gr) \cap T(\mathcal{F})$.*

2.2 Primal Grammars

In this section, we recall definitions related to primal grammars which have been introduced by M. Hermann and R. Galbavý in [11]. To avoid too long developments, some notations have been modified to fit ours. More details can be found in [11].
The control mechanism introduced by Hermann and Galbavý is based on the notion of counter variables which are variables having integer values.

Definition 5 *A counter expression is an expression built over* 0*, successor function* s *and a set of counter variables* $\mathcal{C}nt$. $s(c)$ *means* $1+c$ *if* c *is a counter expression. The ground counter expression* $s^n(0)$ *is interpreted as the integer* n.

Now, we recall the notion of primal term.

Definition 6 (Primal Algebra) *The algebra of primal terms is defined over a set of functions* $\mathcal{D}$ *(whose elements* $\tilde{f}$ *are called defined functions and have a counter arity* $\mathrm{car}(\tilde{f})$ *and a standard arity* $\mathrm{ar}(\tilde{f})$*), a set of constructor symbols* $\mathcal{F}$*, a set of ordinary variables* $\mathcal{X}$ *and a set of counter variables* $\mathcal{C}nt$*. This is the smallest set such that*

- *each ordinary variable of* $\mathcal{X}$ *is a primal term,*
- *if* $c_1,\dots,c_k$ *are counter expressions,* $t_1,\dots,t_n$ *are primal terms and* $\tilde{f} \in \mathcal{D}$ *such that* $car(\tilde{f}) = k$ *and* $ar(\tilde{f}) = k+n$*, then* $\tilde{f}(c_1,\dots,c_k;t_1,\dots,t_n)$ *is a primal term,*
- *if* $t_1,\dots,t_n$ *are primal terms and* c *is a constructor with the arity* $ar(c) = n$ *the* $c(t_1,\dots,t_n)$ *is a primal term.*

The schematization is achieve by introducing a particular type of rewrite system.

Definition 7 (Presburger rewrite system) *A* Presburger rewrite system *contains for each defined functions* $\tilde{f}$ *the following pair of Presburger rewrite rules:*
• *basic rule*

$$\tilde{f}(0,\overline{c};\overline{x}) \rightarrow_{Pbg} t_1$$

• *and the inductive rule which have one of the following forms:*

$$\tilde{f}(n+1,\overline{c};\overline{x}) \rightarrow_{Pbg} t_2[u \leftarrow \tilde{f}(n,\overline{c};\overline{x})]$$

$$\tilde{f}(n+1,\overline{c};\overline{x}) \rightarrow_{Pbg} t_2[u \leftarrow \tilde{f}(n,c_1,...,c_{i-1},c_i+1,c_{i+1},...,c_n;\overline{x})]$$

where $\overline{c}$ *is a vector of counter variables,* $\overline{x}$ *a vector of ordinary variables and* $u \in O(t_2)$*,*

To simplify the presentation, we omit here some additional conditions that are needed to ensure decidability properties of *PG*. We refer the reader to [11] for more details.
From the previous definitions, we can now define a primal grammar and its associated language.

Definition 8 (Primal term grammar) *A* primal term grammar *(or* primal grammar *for short) is a quadruple $G = (\mathcal{F}, \mathcal{D}, \mathcal{R}, t)$ where $\mathcal{C}$ is a set of constructors, $\mathcal{D}$ the set of defined functions, $\mathcal{R}$ is a Presburger rewrite system and t a primal term called* axiom.
The language generated by such a primal term grammar is the set of terms $L(G) = \{\sigma t \downarrow_{\mathcal{R}} \mid \sigma$ affects a ground integer value to each counter variable of t $\}$. Note that, as usual, $\sigma t \downarrow_{\mathcal{R}}$ represents the normal form of σt w.r.t. the system $\mathcal{R}$ (see [5] for details on rewriting).

Intuitively, a *PG* generates terms thanks to a rewrite system R which is controlled by counter variables. Elements of the schematized language are then obtained by completely instantiating the terms with ground substitutions if needed.

Back to the introducing Example 2, let G be a primal grammar defined by $\mathcal{F} = \{suc, z, *\}$, $\mathcal{D} = \{\tilde{f}, \tilde{g}\}$, $\mathcal{R}$ is composed by the 4 Presburger rewrite rules

$$\begin{gathered}\tilde{g}(0) \rightarrow_{Pbg} z,\\ \tilde{g}(s(n)) \rightarrow_{Pbg} s(\tilde{g}(n)),\\ \tilde{f}(0) \rightarrow_{Pbg} z,\\ \tilde{f}(s(n)) \rightarrow_{Pbg} \tilde{g}(s(n)) * \tilde{f}(n)\end{gathered}$$

and the axiom $\tilde{f}(c)$.

To simplify the notation, $(,)$ are omitted in the use of binary function $*$ and moreover, the list reduced to one element $*(z, \perp)$ is simply denoted as z. Using the precedence $\tilde{f} \succ \tilde{g}$, our Presburger rules fit Definition 7.

$$\begin{gathered}\tilde{f}(1) \rightarrow_{Pbg} \tilde{g}(1) * \tilde{f}(0) \rightarrow_{Pbg} \tilde{g}(1) * z \rightarrow_{Pbg} s(\tilde{g}(0)) * z \rightarrow s(z) * z.\\ \text{So } L(G) = \{z, s(z) * z, \ldots, suc^n(z) * suc^{n-1}(z) * \cdots * z, \ldots\}\end{gathered}$$

2.3 Linear Logic and Sequent Calculus

We recall here some basic notations and notions related to first order and linear logics [8, 7], and proof systems in sequent calculus [16].
Let us consider a first order logic signature Σ with V a countable set of variables, $\Sigma_{\mathcal{F}}$ a finite set of function symbols with fixed arity and Σ_N a finite set of predicate symbols. $T(\Sigma_{\mathcal{F}}, V)$ denotes the first order term algebra built over $\Sigma_{\mathcal{F}}$ and V and $\mathcal{T}(\Sigma_{\mathcal{F}})$ denotes the set of ground terms. Atoms are, as usual, built over predicates symbols and terms. A substitution is a mapping from V to $T(\Sigma_{\mathcal{F}}, V)$ which is extended to $T(\Sigma_{\mathcal{F}}, V)$. A substitution assigning t to a variable x will be denoted $\{x \leftarrow t\}$. We introduce some linear logic[‡] notations: $\multimapinv$ denotes the linear implication and $\otimes$ denotes the multiplicative conjunction (see [8, 7] for the precise definitions of these connectives).
A formula $A \multimapinv B_1 \otimes \ldots \otimes B_n$ will be called a clause. The set $\Sigma_{formula}$ is the set of Σ-formulas built using atoms and the logic connectives.
We recall basic notations for sequent calculus.

[‡] Intuitively, the key idea of linear logic we use here is that, when performing logical inferences, some hypothesis can be consumed. This means that, along a proof, some formulas are persistent (as in usual first order logic) but some formulas can only be used once.

Definition 9 (Sequent) *A sequent will be written as $\Sigma : \Delta \to G$ where Σ is the signature, Δ a multiset of Σ-formulas and G a Σ-formula.*

From these sequents we can built proof systems.

Definition 10 (Proof System) *A proof system consists of a set of inference rules between sequents. An inference rule is presented here as:*

$$\frac{\Sigma : \Delta \to G}{\Sigma' : \Delta' \to G'}$$

We use the classical notion of proof (i.e. there is a proof of a sequent in a proof system if this sequent is the root of a proof tree constructed using inference rules with empty leaves).

3 From *TSG* and *PG* Computation to Proof Search

In this section, we define the translation of grammar production rules into linear logic formulas. Then, designing a particular proof system, we show that usual grammar computations as defined in Definitions 3 and 8 can be reduced to proof searches using this system.

3.1 Transformation of TSG into Linear Logic Formulas

Given a *TSG* $(\mathcal{F}, \mathcal{N}, PP, TI)$, the set PP of production rules is partitioned into PP_{free} the set of free production rules and PP_{sync} the set of packs of synchronized production rules. We define now a transformation function Ψ which is decomposed into two different mappings (corresponding respectively to the transformation of free and synchronized production rules). A predicate symbol and a variable will be associated to each non terminal.

Definition 11 (Transformation Function Ψ_{TSG})

Let $\sigma_{\mathcal{N}} : \mathcal{N} \to \Sigma_{\mathcal{N}}$ be the mapping that translates every non terminal symbol into a predicate symbol (to simplify, we will write $\sigma_{\mathcal{N}}(N) = n$). Let $\sigma_{\mathcal{V}} : \mathcal{N} \to \mathcal{V}$ be the mapping that transforms every non terminal symbol into a logical variable (we will write $\sigma_{\mathcal{V}}(N) = N$). For the sake of readability, universal quantifications have been omitted when clear.
Let $\sigma_{PP_{free}} : PP_{free} \to \Sigma_{formula}$ be the mapping that translates every free production rule into a Σ-formula and $\sigma_{PP_{sync}} : PP_{sync} \to \Sigma_{formula}$ the mapping that translates every pack of synchronized production rules into a Σ-formula.
Free productions
Let $N \to g(N_1, \ldots, N_p)$ and $N \to t$ in PP_{free}.

$$\begin{aligned}&\sigma_{PP_{free}}(N \to g(N_1, \ldots, N_p))\\&= n(g(N_1, \ldots, N_p), c) \multimapinv n_1(N_1, c) \otimes \ldots \otimes n_p(N_p, c)\end{aligned}$$

$$\sigma_{PP_{free}}(N \to t) \quad = n(t, c) \multimapinv \boldsymbol{1}$$

Synchronized Productions
Let $\mathcal{S} = \{R \mid P \in PP_{sync} \wedge R \in P\}$ and $\{R_1; \ldots; R_n\} \in PP_{sync}$:

$\sigma_{PP_{sync}}(\{R_1,\ldots,R_n\}) = \forall c \exists nc(\sigma_{\mathcal{S}}(R_1,c,nc) \otimes \ldots \otimes \sigma_{\mathcal{S}}(R_n,c,nc))$

where $\sigma_{\mathcal{S}} : \mathcal{S} \times C \times C \to \Sigma_{formula}$ *is defined*[§]*, for* $N \to g(N_1,\ldots,N_p)$ *and* $N \to t$ *in* $\mathcal{S}$*, as:*

$$\begin{aligned}
&\sigma_{\mathcal{S}}(N \to g(N_1,\ldots,N_p),c,nc)\\
&= n(g(N_1,\ldots,N_p),c) \multimapinv n_1(N_1,nc) \otimes \ldots \otimes n_p(N_p,nc)\\[1em]
&\sigma_{\mathcal{S}}(N \to t,c,nc) \quad = n(t,c) \multimapinv \boldsymbol{1}
\end{aligned}$$

Let $\Sigma_\Psi = \Sigma_{\mathcal{F}} \cup \Sigma_{\mathcal{N}}$ and $\Psi((\mathcal{F},\mathcal{N},PP,TI)) = \sigma_{PP_{free}}(PP_{free}) \cup \sigma_{PP_{sync}}(PP_{sync})$ (in the following, Ψ will denote a set of formulas obtained from a *TSG* and will be called a *TSG* program). At this step, a particular sequent calculus defines the semantics of Ψ.

A similar transformation function can be defined for primal grammars.

3.2 Transformation of PG into Linear Logic Formulas

We suppose in the following that Presburger rewrite systems are such that each definite symbol not appearing in the left hand side of a rule must also appear linearly in the right hand side. This is not in fact a restriction since multiple occurrences of a same symbol can be replaced by a new symbol with the same definition. This transformation does not modify the primal grammar definition.

Given a primal grammar $(\mathcal{F},\mathcal{D},\mathcal{R},A)$, the function$\sigma_{\mathcal{D}} : \mathcal{D} \to \Sigma_{\mathcal{D}}$ maps each defined symbol N to a predicate symbol (we will use the notation $\sigma_{\mathcal{D}}(N) = n$). $\sigma_{\mathcal{V}} : \mathcal{D} \to \mathcal{V}$ is the function that maps each defined symbol N to a logical variable ($\sigma_{\mathcal{V}}(N) = N$). We use the notation $\vec{c} = (c^1,\ldots,c^{ar_c(N)-1})$, $\vec{c'} = (c^1,\ldots,c^{i-1},s(c^i),c^{i+1},\ldots,c^{ar_c(N)-1})$ and $\forall i \in [1..p], \vec{c}_i \sqsubseteq \vec{c}$.

$\sigma_R : \mathcal{R} \to \Sigma_{formule}$ is the function that maps to each Presburger rewrite rule a formula:

[§] The set C denotes a set of counters (i.e. new symbols which are not in the current signature).

$$\forall(N(0) \to t) \in \mathcal{R},$$
$$\sigma_{\mathcal{R}}(N(0) \to t)$$
$$= n(0,t) \multimapinv \mathbf{1}$$

$$\forall(N(0,\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p}))) \in \mathcal{R},$$
$$\sigma_{\mathcal{R}}(N(0,\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p})))$$
$$= \quad n(g(N_1,\ldots,N_p,0,\vec{c})) \multimapinv n_1(N_1,\vec{c_1}) \otimes \ldots \otimes n_p(N_p,\vec{c_p})$$

$$\forall(N(s(c^0),\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p}))[N(c^0,\vec{c})]) \in \mathcal{R},$$
$$\sigma_{\mathcal{R}}(N(s(c^0),\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p}))[N(c^0,\vec{c})])$$
$$= \quad n(g(N_1,\ldots,N_p)[N],c^0,\vec{c}) \multimapinv n_1(N_1,\vec{c_1}) \otimes \ldots \otimes n_p(N_p,\vec{c_p}) \otimes n(N,c^0,\vec{c})$$

$$\forall(N(s(c^0),\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p}))[N(c^0,\vec{c'})]) \in \mathcal{R},$$
$$\sigma_{\mathcal{R}}(N(s(c^0),\vec{c}) \to g(N_1(\vec{c_1}),\ldots,N_p(\vec{c_p}))[N(c^0,\vec{c'})])$$
$$= n(g(N_1,\ldots,N_p)[N],s(c^0),\vec{c}) \multimapinv \quad n_1(N_1,\vec{c_1}) \otimes \ldots \otimes n_p(N_p,\vec{c_p}) \otimes n(N,c^0,\vec{c'})$$

As previously defined for *TSG*, we get the notion of *PG* program Ψ from the above definition. After these syntactic transformations, there is no more need to distinguish the two different types of grammar definition. We propose a uniform proof system which allows us to handle both formalisms at once.

3.3 The Proof System FG

In this section, we define a sequent calculus proof system inspired by the system Forum of D. Miller [16]. Our proof system *FG* (**F**orum for **G**rammars) is defined by the following inference rules and defines the basic proof theoretical semantics of a *TSG* (resp. *PG*) program Ψ. In order to simplify the presentation, we have omitted Ψ in the sequents since, of course, it does not change.

Definition 12 (*FG* system)

[**1**]

$$\overline{\Sigma :\to \mathbf{1}}$$

[*Sync*]

$$\frac{\Sigma, \alpha : C_1\theta,\ldots,C_r\theta,\Delta \to G}{\Sigma : \Delta \to G}$$

where $\forall c \exists nc(C_1 \otimes \ldots \otimes C_r) \in \Psi$, $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$, $\beta \in \Sigma$, $\alpha \notin \Sigma$.

[*Back* !]

$$\frac{\Sigma : \Delta_1 \rightarrow A_1\sigma \quad \ldots \quad \Sigma : \Delta_p \rightarrow A_p\sigma}{\Sigma : \Delta_1, \ldots, \Delta_p \rightarrow G}$$

where $C = (H \circ\!\!- A_1 \otimes \ldots \otimes A_p) \in \Psi$ *and* $H\sigma = G$.

[*Back* ?]

$$\frac{\Sigma : \Delta_1 \rightarrow A_1\sigma \quad \ldots \quad \Sigma : \Delta_p \rightarrow A_p\sigma}{\Sigma : (H \circ\!\!- A_1 \otimes \ldots \otimes A_p), \Delta_1, \ldots, \Delta_p \rightarrow G}$$

where $H\sigma = G$.

Comments: It should be noticed that we distinguish free production rules and synchronized production rules (see the definition of the transformation function Ψ in Section 3.1). Clearly free productions appear as clauses $H \circ\!\!- A_1 \otimes \ldots \otimes A_n$ and packs of synchronized productions appear as formulas $\forall c \exists nc(C_1 \otimes \ldots \otimes C_n)$ where the C_i's are $H \circ\!\!- A_1 \otimes \ldots \otimes A_n$ and are called linear clauses (in the following linear clause will always refer to a clause generated by synchronization rules). Clauses corresponding to free productions in Ψ are persistent along the proof and are used in the inference [*Back* !] (this corresponds to a step of the grammar derivation using this free production). The treatment of a pack of synchronization is performed thanks to the rules [*Sync*] and [*Back* ?]. The first step consists in generating the formula corresponding to this pack in Δ (rule [*Sync*]). The control is ensured by the instantiation of the counter variables with new symbols added to the signature Σ in rule [*Sync*]. Since Δ is the linear logic part of our system, the linear clauses will be consumed when used by rule [*Back* ?] (in the philosophy of linear logic). This ensures the integrity of the synchronization and the control of the simultaneous application of the different productions of this pack. [*Back*?] and [*Back*!] lead to branching in the search tree.

As a consequence, our system *FG* is a subset of Forum (i.e. our rules can be expressed in Forum).

Proof: The rules of FORUM used in this proof can be found in [16]. We introduce the program Ψ in the proof to keep the initial FORUM presentation.

Proof of the correctness of 1 w.r.t. FORUM
Let $\mathbf{1} \equiv \perp \circ\!\!- \perp$ in

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Sigma : \Psi; \xrightarrow{\perp}}[\perp L]}{\Sigma : \Psi; \perp \rightarrow}[decide_1]}{\Sigma : \Psi; \perp \rightarrow \perp}[\perp R]}{\Sigma : \Psi; \rightarrow \mathbf{1}}[\circ\!\!- R]$$

Proof of the correctness of $[Sync]$ **w.r.t. FORUM**
Let $C \otimes C' \equiv (\bot \multimapinv ((\bot \multimapinv C) \parr (\bot \multimapinv C')))$ and $[\otimes L]$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Sigma : \Psi; C, C', \Delta \longrightarrow G}{\Sigma : \Psi; C, C', \Delta \longrightarrow \bot, G}[\bot R]}{\Sigma : \Psi; C, \Delta \longrightarrow \bot \multimapinv C', G}[\multimapinv R]}{\Sigma : \Psi; C, \Delta \longrightarrow \bot, \bot \multimapinv C', G}[\bot R]}{\Sigma : \Psi; \Delta \longrightarrow \bot \multimapinv C, \bot \multimapinv C', G}[\multimapinv R]}{\Sigma : \Psi; \Delta \longrightarrow (\bot \multimapinv C) \parr (\bot \multimapinv C'), G}[\parr R] \qquad \dfrac{}{\Sigma : \Psi; \overset{\bot}{\rightarrow}}[\bot L]}{\Sigma : \Psi; \Delta \overset{C \otimes C'}{\longrightarrow} G}[\multimapinv L]$$

and $\exists x B \equiv (\forall x B^{\bot})^{\bot} \equiv \bot \multimapinv (\forall x (\bot \multimapinv B))$ and $[\exists L]$ ($y \notin \Sigma$):

$$\dfrac{\dfrac{\dfrac{\dfrac{y, \Sigma : \Psi; \Delta, B\{x \leftarrow y\} \rightarrow G}{y, \Sigma : \Psi; \Delta, B\{x \leftarrow y\} \rightarrow \bot, G}[\bot R]}{y, \Sigma : \Psi; \Delta \rightarrow \bot \multimapinv B\{x \leftarrow y\}, G}[\multimapinv R]}{\Sigma : \Psi, \exists x B; \Delta \rightarrow \forall x (\bot \multimapinv B), G}[\forall R] \qquad \dfrac{}{\Sigma : \Psi; \overset{\bot}{\rightarrow}}[\bot L]}{\Sigma : \Psi; \Delta \overset{\exists x B}{\rightarrow} G}[\multimapinv L]$$

in

$$\dfrac{\dfrac{\dfrac{\dfrac{\Sigma, \beta, \alpha : \Psi; C_1\{c \leftarrow \beta, nc \leftarrow \alpha\}, \ldots, C_n\{c \leftarrow \beta, nc \leftarrow \alpha\}, \Delta \rightarrow G}{\Sigma, \beta, \alpha : \Psi; \Delta \overset{C_1\{c \leftarrow \beta, nc \leftarrow \alpha\} \otimes \ldots \otimes C_n\{c \leftarrow \beta, nc \leftarrow \alpha\}}{\longrightarrow} G}[\otimes L]^{n-1}}{\Sigma, \beta : \Psi; \Delta \overset{\exists nc (C_1\{c \leftarrow \beta\} \otimes \ldots \otimes C_n\{c \leftarrow \beta\})}{\longrightarrow} G}[\exists L]}{\Sigma, \beta : \Psi; \Delta \overset{\forall c \exists nc (C_1 \otimes \ldots \otimes C_n)}{\longrightarrow} G}[\forall L]}{\Sigma, \beta : \forall c \exists nc (C_1 \otimes \ldots \otimes C_n) \in \Psi; \Delta \rightarrow G}[decide_2]$$

with $\alpha (\neq \beta) \notin \Sigma$.

Proof of the correctness of $[Back\ !]$ **and** $[Back\ ?]$ **w.r.t. FORUM**
Let $t_1, \ldots, t_n \in T(\Sigma_{\mathcal{F}})$, $\rho = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ with $H\rho = G$
and $\delta_{\otimes} =$

$$\dfrac{\Sigma : \Psi; \Delta_1 \rightarrow A_1 \rho \qquad \dfrac{\dfrac{\Sigma : \Psi; \Delta_{p-1} \rightarrow A_{p-1} \rho \qquad \Sigma : \Psi; \Delta_p \rightarrow A_p \rho}{\vdots}}{\Sigma : \Psi; \Delta_2, \ldots, \Delta_p \rightarrow A_2 \rho \otimes \ldots \otimes A_p \rho}}{\Sigma : \Psi; \Delta_1, \ldots, \Delta_p \rightarrow A_1 \rho \otimes \ldots \otimes A_p \rho}$$

and $C \otimes C' \equiv (\bot \multimapinv ((\bot \multimapinv C) \parr (\bot \multimapinv C')))$ and

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\Sigma:\Psi;\Delta\to C \qquad \cfrac{}{\Sigma:\Psi;\overset{\perp}{\to}}[\perp L]}{\Sigma:\Psi;\Delta\overset{(\perp\multimapinv C)}{\longrightarrow}}[\multimapinv L] \qquad \cfrac{\Sigma:\Psi;\Delta'\to C' \qquad \cfrac{}{\Sigma:\Psi;\overset{\perp}{\to}}[\perp L]}{\Sigma:\Psi;\Delta'\overset{(\perp\multimapinv C')}{\longrightarrow}}[\multimapinv L]}{\Sigma:\Psi;\Delta,\Delta'\overset{(\perp\multimapinv C)\parr(\perp\multimapinv C')}{\longrightarrow}}[\parr L]}{\Sigma:\Psi;\Delta,\Delta',((\perp\multimapinv C)\parr(\perp\multimapinv C'))\to}[decide_1]}{\Sigma:\Psi;\Delta,\Delta',((\perp\multimapinv C)\parr(\perp\multimapinv C'))\to\perp}[\perp R]}{\Sigma:\Psi;\Delta,\Delta'\to C\otimes C'}[\multimapinv R]$$

in

$$\cfrac{\cfrac{\cfrac{\cfrac{\overset{\delta_\otimes}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\to A_1\rho\otimes\ldots\otimes A_p\rho} \qquad \cfrac{}{\Sigma:\Psi;\overset{H\rho}{\longrightarrow}G}[initial]}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\overset{H\rho\multimapinv A_1\rho\otimes\ldots\otimes A_p\rho}{\longrightarrow}G}[\multimapinv L]}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\overset{\forall x_1\ldots x_n(H\multimapinv A_1\otimes\ldots\otimes A_p)}{\longrightarrow}G}[\forall L]^n}{\Sigma:\Psi;(\forall x_1\ldots x_n(H\multimapinv A_1\otimes\ldots\otimes A_p)),\Delta_1,\ldots,\Delta_p\to G}[decide_2]}{}$$

and

$$\cfrac{\cfrac{\cfrac{\overset{\delta_\otimes}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\to A_1\rho\otimes\ldots\otimes A_p\rho} \qquad \cfrac{}{\Sigma:\Psi;\overset{H\rho}{\longrightarrow}G}[initial]}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\overset{H\rho\multimapinv A_1\rho\otimes\ldots\otimes A_p\rho}{\longrightarrow}G}[\multimapinv L]}{\Sigma:\Psi;\Delta_1,\ldots,\Delta_p\overset{\forall x_1\ldots x_n(H\multimapinv A_1\otimes\ldots\otimes A_p)}{\longrightarrow}G}[\forall L]^n}{\Sigma:\Psi;(\forall x_1\ldots x_n(H\multimapinv A_1\otimes\ldots\otimes A_p)),\Delta_1,\ldots,\Delta_p\to G}[decide_2]$$

□

3.4 Correctness and Completeness of FG

The correctness and completeness of the system *FG* is ensured by two theorems: one w.r.t. the *TSG* and the other w.r.t. *PG*.

3.4.1 Correctness and Completeness of FG w.r.t. TSG

We have to prove the following theorem.

Theorem 1 (Correctness and Completeness of *FG* w.r.t. *TSG*)
Given a TSG $(\mathcal{F},\mathcal{N},PP,(I,\#))$*, the corresponding TSG program* Ψ *and* $t\in T(\Sigma_{\mathcal{F}})$*,* $(\Sigma_\Psi,\#:\to\sigma_{\mathcal{N}}(I)(t,\#))$ *has a proof in FG w.r.t. the TSG program* Ψ *if and only if* $((I,\#)\overset{*}{\Rightarrow}t)$*.*

Prologue to the proof of Correctness and completeness of *FG* w.r.t. *TSG*.
The set of instantiated productions of a computation of a *TSG* $Gr=(\mathcal{F},\mathcal{N},PP,TI)$ denoted $IP(Gr)$ is the smallest set defined by:

- if t is in $Comp(Gr)$, $t|u = (X,\beta)$, and $R = (X \to g(Y_1,\ldots,Y_n))$ is in PP_{free} then $R(\beta) = ((X,\beta) \Rightarrow g((Y_1,\beta),\ldots,(Y_n,\beta)))$ is in $IP(Gr)$,
- if t is in $Comp(Gr)$ and there exists n pairwise different occurrences $u_1,\ldots,u_n$ of t such that $\forall i \in [1,n], t|u_i = (X_i,c_i)$ and $c_i = \beta$ and the pack of productions $P = \{R_1 = X_1 \to g_1(Y_1^1,\ldots,Y_{n_1}^1),\ldots,R_r = X_r \to g_r(Y_1^r,\ldots,Y_{n_r}^r)\} \in PP$ (where α is a new symbol) then $P(\alpha,\beta) = \{(X_1,\beta) \Rightarrow g_1((Y_1^1,\alpha),\ldots,(Y_{n_1}^1,\alpha)),\ldots,(X_r,\beta) \Rightarrow g_n((Y_1^r,\alpha),\ldots,(Y_{n_r}^r,\alpha))\}$ $= \{R_1(\alpha,\beta),\ldots,R_r(\alpha,\beta)\}$ is included in $IP(Gr)$.

Let $\sigma_I : IP(Gr) \to \Sigma_{formula}$ be the mapping that translates every instantiated productions into a $\Sigma_{formula}$:

$$\begin{aligned}&\sigma_I((X,\beta) \Rightarrow g((Y_1,\alpha),\ldots,(Y_n,\alpha)))\\ =\ & x(g(Y_1,\ldots,Y_n),\beta) \multimapinv y_1(Y_1,\alpha) \otimes \ldots \otimes y_n(Y_n,\alpha)\end{aligned}$$

A presynchronised production is a production of the form $(X,\beta) \to C((Y_1,\alpha),\ldots,(Y_n,\alpha))$ where C is a context of terminal symbols, β, α two new symbols and $Y_1,\ldots,Y_n$ non terminals.
The set of computations of a *TSG Gr* enlarged by presynchronised productions Φ is $Comp(Gr)$ plus

- $t[u \leftarrow C((Y_1,\alpha),\ldots,(Y_n,\alpha))]$ if t is in the computation of *Gr* enlarged with Φ, $t|u = (X,\beta)$ and $((X,\beta) \to g((Y_1,\alpha),\ldots,(Y_n,\alpha)))$ is in Φ.

If s is the number of instances of packs of productions used during a *TSG* derivation ($t \overset{*}{\Rightarrow} t'$):

- $\Sigma_\Psi \cup \{\#\}$ plus the set of counters used during the derivation is denoted $\Sigma(t \overset{*}{\Rightarrow} t')$ and
- the set of instances of synchronized productions is denoted $\Phi(t \overset{*}{\Rightarrow} t')$.

If s is the number of instances of packs of productions used during a *TSG* derivation $((I,\#) \overset{*}{\Rightarrow} t)$:

- the h^{th} pack of productions used during this *TSG* derivation is denoted $P_h(\beta_h,\alpha_h)$, $1 \le h \le s$,
- the family of sets $\{\Sigma_h\}_{0 \le h \le s}$ is defined by induction: $\Sigma_0 = \Sigma_\Psi \cup \{\#\}$ and $\forall h, 0 \le h < s$, $\Sigma_{h+1} = \Sigma_h \cup \{\alpha_{h+1}\}$,
- the family of sets $\{\Delta^h\}_{0 \le h \le s}$ is defined by induction: $\Delta^0 = \emptyset$ and $\forall h, 0 \le h < s$, $\Delta^{h+1} = \Delta^h \cup \{R_1,\ldots,R_p/\{R_1,\ldots,R_p\} = \sigma_I(P_{h+1}(\beta_{h+1},\alpha_{h+1}))\}$ (Remark: $\Delta^s = \sigma_I(\Phi((I,\#) \overset{*}{\Rightarrow} t))$).

Proof of *FG* $\Rightarrow$ *TSG*.
First, we prove by structural induction that if there exists a *FG* proof (with only instances of the [*Back* !], [*Back* ?] and [**1**] rules) for $(\Sigma_s : \Delta^s \to n(t,\beta))$ then there is a derivation $((N,\beta) \overset{*}{\Rightarrow} t)$ for the *TSG* enlarged by the set of presynchronised productions Φ^s with $\sigma_I(\Phi^s) = \Delta^s$.

Base Cases: Only the case $[Back\ ?] + [\mathbf{1}]$ is treated, the case $[Back\ !] + [\mathbf{1}]$ is similar.

$$\dfrac{\dfrac{}{\Sigma_s : \to \mathbf{1}}[\mathbf{1}]}{\Sigma_s : (n(t,\beta) \multimapinv \mathbf{1}) \to n(t,\beta)}[Back\ ?]$$

with the linear clause $(n(t,\beta) \multimapinv \mathbf{1})$ from a presynchronised production $((N,\beta) \to t)$. Then $(N,\beta) \Rightarrow t$ for the *TSG* enlarged with $\{((N,\beta) \to t)\}$ and $\sigma_I(\{(N,\beta) \Rightarrow t)\}) = \{(n(t,\beta) \multimapinv \mathbf{1})\}$.

Induction cases: Only the case $[Back\ ?]$ is treated, the case $[Back\ !]$ is similar.

$$\dfrac{\dfrac{\nabla_1}{\Sigma_s : \Delta_1 \rightarrow n_1(t_1,\alpha)} \quad \cdots \quad \dfrac{\nabla_p}{\Sigma_s : \Delta_p \rightarrow n_p(t_p,\alpha)}}{\Sigma_s : \Delta_1,\ldots,\Delta_p, n(g(N_1,\ldots,N_p),\beta) \multimap n_1(N_1,\alpha)\otimes\ldots\otimes n_p(N_p,\alpha) \rightarrow n(t,\beta)}[Back\ ?]$$

with $\alpha \notin \Sigma_s$.

By induction hypothesis $\forall i, 1 \leq i \leq p$, there exists a derivation $(N_i,\beta) \stackrel{*}{\Rightarrow} t_i$ for the *TSG* enlarged with the set of presynchronised productions Φ_i and $\Delta_i = \sigma_I(\Phi_i)$. Then there exists a derivation $(N,\beta) \stackrel{*}{\Rightarrow} t$ for the *TSG* enlarged with the presynchronised productions

$$\bigcup_{i=1}^{p} \Phi_i \cup \{(N,\beta) \rightarrow g((N_1,\alpha),\ldots,(N_p,\alpha))\}$$

and

$$\Delta^s = \bigcup_{i=1}^{p} \sigma_I(\Phi_i) \cup \{\sigma_I((N,\beta) \rightarrow g((N_1,\alpha),\ldots,(N_p,\alpha)))\}.$$

□

Synchronization Rule Introduction: We begin with a definition.

Definition 13 (Restricted *FG* proof) *A FG proof is in* restricted *form if all the instances of the* $[Sync]$ *rule are at the root of the proof.*

First, we prove by induction on k if there exists a restricted *FG* proof for$(\Sigma_\Psi, \# : \rightarrow i(t,\#))$:

$$\dfrac{\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}[Sync]^s \quad \vdots}{\Sigma_\Psi, \# : \rightarrow i(t,\#)}[Sync]^1$$

then there exists a restricted *FG* proof for $(\Sigma_k : \Delta^k \rightarrow i(t,\#))$:

$$\dfrac{\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}[Sync]^s \quad \vdots}{\Sigma_k : \Delta^k \rightarrow i(t,\#)}[Sync]^{k+1}$$

The case $k = 0$ is trivial. Now we assume $0 < k \leq s$. By induction hypothesis, there exists a *FG* proof for $(\Sigma_{k-1} : \Delta^{k-1} \rightarrow i(t,\#))$ with $s-k+1$ instances of the $[Sync]$ rule, then :

$$\begin{array}{c}
\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}{\scriptstyle [Sync]^s} \\
\vdots \\
\dfrac{\overline{\Sigma_{k-1}, \alpha : \Delta^{k-1}, \sigma_{\mathcal{S}}(R_1,c,nc)\theta, \ldots, \sigma_{\mathcal{S}}(R_p,c,nc)\theta \rightarrow i(t,\#)}{\scriptstyle [Sync]^{k+1}}}{\Sigma_{k-1} : \Delta^{k-1} \rightarrow i(t,\#)}{\scriptstyle [Sync]^k}
\end{array}$$

with $P_k = \{R_1, \ldots, R_p\}, p > 1$ and $\beta \in \Sigma_{k-1}$, $\alpha \notin \Sigma_{k-1}$ and $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$ and a derivation $(I,\#) \Rightarrow t)$ for the *TSG* enlarged with Φ^{k-1}, $\sigma_I(\Phi^{k-1}) = \Delta^{k-1}$.

But $\Delta^k = \Delta^{k-1} \cup \{\sigma_{\mathcal{S}}(R_1,c,nc)\theta, \ldots, \sigma_{\mathcal{S}}(R_p,c,nc)\theta\}$ and $\Sigma_k = \Sigma_{k-1} \cup \{\alpha\}$ then

$$\begin{array}{c}
\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}{\scriptstyle [Sync]^s} \\
\vdots \\
\overline{\Sigma_k : \Delta^k \rightarrow i(t,\#)}{\scriptstyle [Sync]^{k+1}}
\end{array}$$

Finally, we prove by induction on k that if there exists a restricted *FG* proof for $(\Sigma_\Psi, \# : \rightarrow i(t,\#))$:

$$\begin{array}{c}
\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}{\scriptstyle [Sync]^s} \\
\vdots \\
\overline{\Sigma_\Psi, \# : \rightarrow i(t,\#)}{\scriptstyle [Sync]^1}
\end{array}$$

then there exists a derivation $((I,\#) \overset{*}{\Rightarrow} t)$ for the *TSG* enlarged with Φ^k, $\sigma_I(\Phi^k) = \Delta^k$.

The case $k = s$ is a consequence of the previous result. Now we assume that $0 \leq k < s$. By induction hypothesis on the proof (with $P_{k+1} = \{R_1, \ldots, R_p\}, p > 1$):

$$\begin{array}{c}
\dfrac{\nabla}{\Sigma_s : \Delta^s \rightarrow i(t,\#)}{\scriptstyle [Sync]^s} \\
\vdots \\
\dfrac{\overline{\Sigma_k, \alpha : \Delta^{k+1} \rightarrow i(t,\#)}}{\Sigma_k : \Delta^k \rightarrow i(t,\#)}{\scriptstyle [Sync]^{k+1}} \\
\vdots \\
\overline{\Sigma_\Psi, \# : \rightarrow i(t,\#)}{\scriptstyle [Sync]^1}
\end{array}$$

there exists a derivation $((I,\#) \overset{*}{\Rightarrow} t)$ for the *TSG* enlarged with Φ^{k+1} and $\sigma_I(\Phi^{k+1}) = \Delta^{k+1}$. But $\Delta^{k+1} = \Delta^k \uplus \{\sigma_{\mathcal{S}}(R_1,c,nc)\theta, \ldots, \sigma_{\mathcal{S}}(R_p,c,nc)\theta\}$ $(\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\})$ so in the derivation, the presynchronised productions $\{R_i(\beta,\alpha)\}_{1 \leq i \leq p}$ can be replaced by the instance $P_{k+1}(\beta,\alpha)$ of the pack P_{k+1}. $\square$

Proof of *TSG* $\Rightarrow$ *FG*.
First, we prove by induction on the structure of derivation: if there exists a derivation

$$g((N_1,\beta),\ldots,(N_l,\beta)) \overset{*}{\Rightarrow} g(t_1,\ldots,t_l) \tag{1}$$

for a *TSG* then there exists for every k, $1 \leq k \leq l$, a *FG* proof for

$$(\Sigma((N_k,\beta) \overset{*}{\Rightarrow} t_k) : \sigma_I(\Phi((N_k,\beta) \overset{*}{\Rightarrow} t_k)) \rightarrow n_k(t_k,\beta)) \tag{2}$$

Base cases: Only the case with a pack of productions is treated, the case with a free production is similar. Let $\{N_1 \rightarrow t_1,\ldots,N_r \rightarrow t_r\} \in PP_{sync}$ and the derivation $(g((N_1,\beta),\ldots,(N_r,\beta)) \Rightarrow g(t_1,\ldots,t_r))$. Then for every $k, 1 \leq k \leq r$ there exists a *FG* proof:

$$\frac{\overline{\Sigma_\Psi,\beta :\rightarrow \mathbf{1}}}{\Sigma_\Psi,\beta : (n_k(t_k,\beta) \circ\!\!- \mathbf{1}) \rightarrow n_k(t_k,\beta)}$$

with

$$\begin{array}{rcl}
\Phi((N_k,\beta) \Rightarrow t_k) & = & \{(N_k,\beta) \Rightarrow t_k\} \\
\sigma_I(\Phi((N_k,\beta) \Rightarrow t_k)) & = & \{n_k(t_k,\beta) \circ\!\!- \mathbf{1}\} \\
\Phi(g((N_1,\beta),\ldots,(N_r,\beta)) \Rightarrow g(t_1,\ldots,t_r)) & = & \{(n_1(t_1,\beta) \circ\!\!- \mathbf{1}),\ldots,(n_r(t_r,\beta) \circ\!\!- \mathbf{1})\}.
\end{array}$$

Induction cases: Only the case with a pack of productions is treated, the case with a free production is similar. Let

$$\{N_1 \rightarrow g_1(N_1^1,\ldots,N_1^{p_1}),\ldots,N_r \rightarrow g_r(N_r^1,\ldots,N_r^{p_r})\} \in PP_{sync} \tag{3}$$

and a derivation

$$\begin{array}{l}
(g((N_1,\beta),\ldots,(N_r,\beta)) \\
\quad \Rightarrow g(g_1((N_1^1,\alpha),\ldots,(N_1^{p_1},\alpha)),\ldots,g_r((N_r^1,\alpha),\ldots,(N_1^{p_r},\alpha)))) \\
\quad \overset{*}{\Rightarrow} g(g_1(t_1^1,\ldots,t_1^{p_1}),\ldots,g_r(t_r^1,\ldots,t_1^{p_r})))
\end{array}$$

Then by induction hypothesis, for every $k, 1 \leq k \leq r$ and $i_k, 1 \leq i_k \leq p_k$ there exists a *FG* proof:

$$\delta_k^{i_k} = \frac{\nabla_k^{i_k}}{\Sigma((N_k^{i_k},\alpha) \Rightarrow t_k^{i_k}) : \sigma_I(\Phi((N_k^{i_k},\alpha) \Rightarrow t_k^{i_k})) \rightarrow n_k^{i_k}(t_k^{i_k},\alpha)}$$

For every $k, 1 \leq k \leq r$,

$$\begin{array}{rl}
\multicolumn{2}{l}{\sigma_I(\Phi((N_k,\beta) \overset{*}{\Rightarrow} g_k(t_k^1,\ldots,t_k^{p_k})))} \\
= & \sigma_I(\bigcup_{i=1}^{p_k} \Phi((N_k^{i_k},\alpha) \overset{*}{\Rightarrow} t_k^{i_k}) \cup \{(N_k,\beta) \Rightarrow g_k((N_k^1,\alpha),\ldots,(N_k^{p_k},\alpha))\}) \\
= & \bigcup_{i=1}^{p_k} \sigma_I(\Phi((N_k^{i_k},\alpha) \overset{*}{\Rightarrow} t_k^{i_k})) \cup \{\sigma_I((N_k,\beta) \Rightarrow g_k((N_k^1,\alpha),\ldots,(N_k^{p_k},\alpha)))\} \\
= & \bigcup_{i=1}^{p_k} \sigma_I(\Phi((N_k^{i_k},\alpha) \overset{*}{\Rightarrow} t_k^{i_k})) \cup \{n_k(g_k(N_k^1,\ldots,N_k^{p_k}),\beta) \circ\!\!- n_k^1(N_k^1,\alpha),\ldots,n_k^{p_k}(N_k^{p_k},\alpha)\}
\end{array}$$

Then for every $k, 1 \leq k \leq r$ there exits a *FG* proof (the signatures of $\delta_k^{i_k}$, $1 \leq i_k \leq p_k$, are augmented with β):

$$\frac{\delta_k^1 \quad \dots \quad \delta_k^{p_k}}{\Sigma((N_k,\beta) \stackrel{*}{\Rightarrow} g_k(t_k^1,\dots,t_k^{p_k})) : \sigma_I(\Phi((N_k,\beta) \stackrel{*}{\Rightarrow} g_k(t_k^1,\dots,t_k^{p_k}))) \to n_k(g_k(t_k^1,\dots,t_k^{p_k}),\beta)}$$

$$\Sigma((N_k,\beta) \stackrel{*}{\Rightarrow} g_k(t_k^1,\dots,t_k^{p_k})) = \bigcup_{i_k=1}^{p_k} \Sigma((N_k^{i_k},\alpha) \stackrel{*}{\Rightarrow} t_k^{i_k}) \cup \{\beta\}$$

and

$$\Phi(g((N_1,\beta),\dots,(N_r,\beta)) \stackrel{*}{\Rightarrow} g(g_1(t_1^1,\dots,t_1^{p_1}),\dots,g_r(t_r^1,\dots,t_1^{p_r}))) = \bigcup_{k=1}^{r} \Phi((N_k,\beta) \stackrel{*}{\Rightarrow} g_k(t_k^1,\dots,t_k^{p_k}))).$$

By the result above, we have the following corollary:

Corollary 1 *If there exists a derivation* $((I,\#) \stackrel{*}{\Rightarrow} t)$ *for a TSG then there exists a FG proof for* $(\Sigma_s : \Delta^s \to i(t,\#))$.

Now we prove by induction on h: if there exists a *FG* proof for $(\Sigma_s : \Delta^s \to i(t,\#))$ then there exists an *FG* proof for $(\Sigma_h : \Delta^h \to i(t,\#))$.
The base case $h = 0$ is trivial, now we assume that $0 < h \leq s$. $\Sigma_{h+1} = \Sigma_h \cup \{\alpha_{h+1}\}$ and $\Delta^{h+1} = \Delta^h \cup \{R_1^{h+1},\dots,R_p^{h+1} / \{R_1^{h+1},\dots,R_p^{h+1}\} = \sigma_I(P_{h+1}(\beta_{h+1},\alpha_{h+1}))\}$. By induction hypothesis, there exists a *FG* proof for:

$$\frac{\nabla}{\Sigma_h,\alpha_{h+1} : \Delta^h, R_1^{h+1},\dots,R_p^{h+1} \to i(t,\#)}$$

so there exists a *FG* proof:

$$\cfrac{\cfrac{\nabla}{\Sigma_h,\alpha_{h+1} : \Delta^h, R_1^{h+1},\dots,R_p^{h+1} \to i(t,\#)}}{\Sigma_h : \Delta^h \to i(t,\#)}\,[Sync]$$

□

This achieves the first part of the correctness and completeness of our transformation. We now have to provide a similar theorem for Primal Grammars.

3.4.2 Correctness and Completeness of FG w.r.t. PG

Theorem 2 (Correctness and Completeness of *FG* w.r.t. *PG*)
Given a PG $(\mathcal{F},\mathcal{D},\mathcal{R},A)$*, the corresponding PG program* Ψ *and* $\vec{c} \in \mathbb{N}^{ar_c(A)}$ *and* $t \in T(\mathcal{F})$. $(\Sigma_\Psi : \Psi; \to \sigma_{\mathcal{D}}(A)(t,\vec{c}))$ *has a proof in FG if and only if* $(A(\vec{c}) \stackrel{*}{\Rightarrow} t)$.

Proof of $PG \Longrightarrow FG$.
We prove by induction on the length of a PG derivation if there exists a PG derivation $(\tilde{f}(m^0,\vec{m}) \Rightarrow^*_{Pbg} t)$, with $\tilde{f} \in \mathcal{D}$ and $m^0 \in \mathbb{N}$ and $\vec{m} \in \mathbb{N}^{ar(\tilde{f})-1}$ then there exists a FG proof for $(\Sigma_\Psi : \Psi; \rightarrow f(t,m^0,\vec{m}))$
The right hand-side of the sequent $(\Sigma_\Psi : \Psi;)$ does not vary and is omitted.

Case $l = 1$**:** $\tilde{f}(m^0,\vec{m}) \Rightarrow_{Pbg} t$ then $m^0 = 0$ and there exists a basic rule $(\tilde{f}(0,\vec{c}) \rightarrow t) \in \mathcal{R}$ and a linear clause $(\forall \vec{c}(f(t,0,\vec{c}) \circ\!- \mathbf{1})) \in \Psi$ and a FG proof:

$$\dfrac{\dfrac{}{\rightarrow \mathbf{1}}[\mathbf{1}]}{\rightarrow f(t,0,\vec{m})}[Back\,!]$$

Case $l > 1$**:** Only the case with the last applied rule as an instance of the second inductive rule[¶] is treated, the case for the first inductive rule is similar. So $m^0 = s(m^{-1})$ and there exists a context C with $t = C(t_1,\ldots,t_p)[t']$ and an inductive rule $(\tilde{f}(s(c^0),\vec{c}) \rightarrow_{Pbg} C(\tilde{f}_1(\vec{c}_1),\ldots,\tilde{f}_p(\vec{c}_p))[\tilde{f}(c^0,\vec{c^+})]) \in \mathcal{R}$.

So $\forall i \in [1..p], \tilde{f}_i(\vec{m}_i) \overset{k_i \leq l}{\Rightarrow}_{Pbg} t_i$, $\tilde{f}(m^{-1},\vec{m^+}) \overset{k' \leq l}{\Rightarrow}_{Pbg} t'$ and $\Sigma_{i=1}^{p} k_i + k' = l-1$. Then by induction hypothesis, there exist FG proofs for $\forall i \in [1..p], (\rightarrow f_i(t_i,\vec{m}_i))$:

$$\dfrac{\delta_i}{\rightarrow f_i(t_i,\vec{m}_i)}$$

and there exists a FG proof for $(\rightarrow f(t',m^{-1},\vec{m^+}))$:

$$\dfrac{\delta'}{\rightarrow f(t',m^{-1},\vec{m^+})}$$

Moreover,

$$\begin{array}{l}\sigma_{\mathcal{R}}(\tilde{f}(s(c^0),\vec{c}) \rightarrow_{Pbg} C(\tilde{f}_1(\vec{c}_1),\ldots,\tilde{f}_p(\vec{c}_p))[\tilde{f}(c^0,\vec{c^+})]) = \\ \forall c^0 \forall \vec{c} \forall F_1 \ldots \forall F_p \forall F(f(C(F_1,\ldots,F_p)[F],s(c^0),\vec{c}) \\ \qquad \circ\!- f_1(F_1,\vec{c}_1) \otimes \ldots \otimes f_p(F_p,\vec{c}_p) \otimes f(F,c^0,\vec{c^+}))\end{array}$$

Then there exists a FG proof for $(\rightarrow f(C(t_1,\ldots,t_p)[t'],s(m^{-1}),\vec{m}))$:

$$\dfrac{\dfrac{\delta_1}{\rightarrow f_1(t_1,\vec{m}_1)} \quad \ldots \quad \dfrac{\delta_p}{\rightarrow f_p(t_p,\vec{m}_p)} \quad \dfrac{\delta'}{\rightarrow f(t',m^{-1},\vec{m^+})}}{\rightarrow f(C(t_1,\ldots,t_p)[t'],s(m^{-1}),\vec{m})}[Back\,!]$$

[¶] We write $k = ar(\tilde{f})$ and $\vec{c} = (c^1,\ldots,c^k)$ and $\vec{m} = (m^1,\ldots,m^k)$ and $\vec{c^+} = (c^1,\ldots,c^{i-1},s(c^i),c^{i+1},\ldots,c^k)$ and $\vec{m^+} = (m^1,\ldots,m^{i-1},s(m^i),m^{i+1},\ldots,m^k)$ and $\forall i \in [1..p], \vec{c}_i$ is a subsequence of $\vec{c}$ and $\vec{m}_i$ is a subsequence of $\vec{m}$ on the same positions.

□

Proof of $PG \Leftarrow FG$**.**
We prove by induction on the structure of a *FG* proof if there exists a *FG* proof for

$$(\Sigma_\Psi : \Psi; \rightarrow f(t, m^0, \vec{m})), \tag{4}$$

with $\tilde{f} \in \mathcal{D}$ and $m^0 \in \mathbb{N}$ and $\vec{m} \in \mathbb{N}^{ar(\tilde{f})-1}$ then there exists a *PG* derivation $(\tilde{f}(m^0, \vec{m}) \Rightarrow^*_{Pbg} t)$
The right hand-side of the sequent $(\Sigma_\Psi : \Psi;)$ does not vary and is omitted.

Base case: There exists a proof of $\frac{\rightarrow f(t,m^0,\vec{m})}{\mathbf{1}}$ so $m^0 = 0$ and there exists a linear clause $\forall \vec{c}(f(t, 0, \vec{c}) \multimapinv \mathbf{1}) \in \Psi$ and a basic rule $\tilde{f}(0, \vec{c}) \rightarrow t \in \mathcal{R}$). Then $\tilde{f}(0, \vec{m}) \Rightarrow_{Pbg} t$.

Induction cases: There exists a *FG* proof for $(\rightarrow f(t, m^0, \vec{m}))$. The only possible inference rule applied at the root of the proof is an instance of the $[Back\ !]$ rule. Only the case for a linear clause obtained from the second inductive rule[‖] is treated, the cases for the basic rule or the first inductive rule are similar.

We have a *FG* proof of shape

$$\frac{\delta_1 \quad \dots \quad \delta_p \quad \delta'}{\rightarrow f(t, s(m^{-1}), \vec{m})}\ [Back\ !]$$

with the linear clause

$$\forall c^0 \forall \vec{c} \forall F_1 \dots \forall F_p \forall F (f(C(F_1, \dots, F_p)[F], s(c^0), \vec{c}) \multimapinv$$
$$f_1(F_1, \vec{c_1}) \otimes \dots \otimes f_p(F_p, \vec{c_p}) \otimes f(F, c^0, \vec{c^+}) \in \Psi$$

(obtained from the rule $\tilde{f}(s(c^0), \vec{c}) \rightarrow C(\tilde{f}_1(\vec{c_1}), \dots, \tilde{f}_p(\vec{c_p}))[\tilde{f}(c^0, \vec{c^+})] \in \mathcal{R}$) with $t = C(t_1, \dots, t_p)[t']$ and $\forall i \in [1..p]\ \delta_i$ a proof with the root $(\rightarrow f_i(t_i, \vec{m_i}))$ and δ' a proof with the root $(\rightarrow n(t', m^{-1}, \vec{m^+}))$.

For all proofs δ_i, $i \in [1..p]$, by induction hypothesis, there exist *PG* derivations $\tilde{f}_i(\vec{m_i}) \Rightarrow_{Pbg} t_i$ and for the proof δ', by induction hypothesis, there exists a *PG* derivation $\tilde{f}(m^{-1}, \vec{m^+}) \Rightarrow_{Pbg} t'$. Then by the inductive rule $\tilde{f}(s(c^0), \vec{c}) \rightarrow_{Pbg} C(\tilde{f}_1(\vec{c_1}), \dots, \tilde{f}_p(\vec{c_p}))[N(c^0, \vec{c^+})] \in \mathcal{R}$, $\tilde{f}(m^0, \vec{m}) \Rightarrow_{Pbg} t$.

□

3.4.3 Application

We address here Example 1 on *TSG* to illustrate how the transformation and proof system *FG* work. We build the derivation tree for the *TSG* program Ψ. Σ_Ψ is omitted in the left hand side of the sequents of the following proof in order to simplify the notation, we only mention the new symbols added to the signature along the proof and universal quantifiers are omitted in the sequents:

[‖] We denote $k = ar(\tilde{f})$ and $\vec{c} = (c^1, \dots, c^k)$ and $\vec{m} = (m^1, \dots, m^k)$ and $\vec{c^+} = (c^1, \dots, c^{i-1}, s(c^i), c^{i+1}, \dots, c^k)$ and $\vec{m^+} = (m^1, \dots, m^{i-1}, s(m^i), m^{i+1}, \dots, m^k)$ and $\forall i \in [1..p], \vec{c_i}$ is a subsequence of $\vec{c}$ and $\vec{m_i}$ a subsequence of $\vec{m}$ on the same positions.

$$\begin{array}{l}\{\ F_0 : \forall c(\forall X \forall Y(i(f(X,Y),c) \multimapinv x(X,c) \otimes y(Y,c))),\\ \quad F_1 : \forall c(x(z,c) \multimapinv 1),\\ \quad F_2 : \forall c(y(z,c) \multimapinv 1),\\ \quad F_3 : \forall c \exists nc\,((\forall X(x(s(X),c) \multimapinv x(X,nc))) \otimes (\forall Y(y(s(s(Y)),c) \multimapinv y(Y,nc))))\ \}\end{array}$$

$$\dfrac{\dfrac{\delta_1 \qquad\qquad\qquad\qquad \delta_2}{\#,\beta : x(s(X),\#) \multimapinv x(X,\beta), y(s(s(Y)),\#) \multimapinv y(Y,\beta) \to i(f(s(z),s(s(z))),\#)}[Back\ !]\ F_0}{\# :\to i(f(s(z),s(s(z))),\#)}[Sync]\ F_3$$

where

$$\delta_1 : \left\{ \dfrac{\dfrac{\mathbf{1}}{\#,\beta :\to x(z,\beta)}[Back\ !]\ F_1}{\#,\beta : x(s(X),\#) \multimapinv x(X,\beta) \to x(s(z),\#)}[Back\ ?] \right.$$

and

$$\delta_2 : \left\{ \dfrac{\dfrac{\mathbf{1}}{\#,\beta :\to y(z,\beta)}[Back\ !]\ F_2}{\#,\beta : y(s(s(Y)),\#) \multimapinv y(Y,\beta) \to y(s(s(z)),\#)}[Back\ ?] \right.$$

4 From Linear Logic to Prolog

From now on, there is no more need to distinguish *TSG* and PG since they have been embedded into the same formalism (i.e. linear logic formulas). From an operational point of view, it appears clearly that the previous system *FG* does not provide any strategy for a proof search (especially concerning the use of the rule [*Sync*]). Furthermore, a refinement will help us to get the implementation of the system. Therefore, we define a goal directed proof system FG^{dir} with the following inference rules.
We define the set operator $\uplus$ as $\uplus_{1\leq i\leq n} A_i = \cup_{1\leq i\leq n} A_i$ such that $\forall 1 \leq i,j \leq n, i \neq j, A_i \cap A_j = \emptyset$.

Definition 14 (FG^{dir} system)

$[Back\ !^{dir}]$

$$\frac{\begin{array}{l}\Sigma: \quad ((\Delta_1 \to G_1),\ldots,(\Delta_{i-1} \to G_{i-1}),\\ \qquad (\Delta'_1 \to A_1\sigma),\ldots,(\Delta'_p \to A_p\sigma),\\ \qquad (\Delta_{i+1} \to G_{i+1}),\ldots,(\Delta_n \to G_n))\end{array}}{\Sigma : ((\Delta_1 \to G_1),\ldots,(\Delta_n \to G_n))}$$

if $H \multimap A_1 \otimes \ldots \otimes A_p \in \Psi$, *there exists a substitution* σ *such that* $H\sigma = G_i$ *and* $\Delta_i = \uplus_{1\leq k\leq p}\Delta'_k$.

$[Back\ ?^{dir}]$

$$\frac{\begin{array}{l}\Sigma :((\Delta_1 \to G_1),\ldots,(\Delta_{i-1} \to G_{i-1}),\\ \qquad (\Delta'_1 \to A_1\sigma),\ldots,(\Delta'_p \to A_p\sigma),\\ \qquad (\Delta_{i+1} \to G_{i+1}),\ldots,(\Delta_n \to G_n))\end{array}}{\Sigma : ((\Delta_1 \to G_1),\ldots,(\Delta_n \to G_n))}$$

if $C = (H \multimap A_1 \otimes \ldots \otimes A_p) \in \Delta_i$, *there exists a substitution* σ *such that* $H\sigma = G_i$ *and* $\Delta_i \setminus \{C\} = \uplus_{1\leq k\leq p}\Delta'_k$.

$[Sync+^{dir}]$

$$\frac{\begin{array}{l}\Sigma,\alpha :((\Delta_1 \cup \Delta''_1 \to G_1),\ldots,(\Delta_{i-1} \cup \Delta''_{i-1} \to G_{i-1}),\\ \qquad (\Delta'_1 \cup \Delta'''_1 \to A_1\sigma),\ldots,(\Delta'_p \cup \Delta'''_p \to A_p\sigma),\\ \qquad (\Delta_{i+1} \cup \Delta''_{i+1} \to G_{i+1}),\ldots,(\Delta_n \cup \Delta''_n \to G_n))\end{array}}{\Sigma :((\Delta_1 \to G_1),\ldots,(\Delta_n \to G_n))}$$

if

$$\begin{array}{c}\forall c \exists nc (C_1 \otimes \ldots \otimes C_r) \in \Psi, 1 \leq j \leq r,\\ C_j\theta = (H \multimap A_1 \otimes \ldots \otimes A_p),\\ H\sigma = G_i, \Delta_i = \uplus_{1\leq k\leq p}\Delta'_k,\\ \{C_1\theta,\ldots,C_{j-1}\theta, C_{j+1}\theta,\ldots,C_r\theta\} = (\uplus_{1\leq k(\neq i)\leq n}\Delta''_k) \uplus (\uplus_{1\leq k\leq p}\Delta'''_k),\\ \beta \in \Sigma, \alpha \notin \Sigma and \theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}.\end{array}$$

$[Sync\mathbf{1}^{dir}]$

$$\frac{\begin{array}{l}\Sigma,\alpha : \quad ((\Delta_1 \cup \Delta'_1 \to G_1),\ldots,(\Delta_{i-1} \cup \Delta'_{i-1} \to G_{i-1})\\ \qquad\qquad (\Delta_{i+1} \cup \Delta'_{i+1} \to G_{i+1}),\ldots,(\Delta_n \cup \Delta'_n \to G_n))\end{array}}{\Sigma : \qquad ((\Delta_1 \to G_1),\ldots,(\Delta_n \to G_n))}$$

if

$$\forall c \exists nc (C_1 \otimes \ldots \otimes C_r) \in \Psi, 1 \leq j \leq r, C_j\theta = (H \circ\!- \mathbf{1}),$$
$$H\sigma = G_i, \Delta_i = \emptyset,$$
$$C_1\theta, \ldots, C_{j-1}\theta, C_{j+1}\theta, \ldots, C_r\theta = (\uplus_{1 \leq k (\neq i) \leq n} \Delta'_k),$$
$$\beta \in \Sigma, \alpha \notin \Sigma and \theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}.$$

$[Axiom\ !^{dir}]$

$$\frac{\begin{array}{ll}\Sigma: & ((\Delta_1 \to G_1), \ldots, (\Delta_{i-1} \to G_{i-1}), \\ & (\Delta_{i+1} \to G_{i+1}), \ldots, (\Delta_n \to G_n))\end{array}}{\Sigma: \quad ((\Delta_1 \to G_1), \ldots, (\Delta_n \to G_n))}$$

if $H \circ\!- \mathbf{1} \in \Psi$, $\Delta_i = \emptyset$ *and there exists a substitution* σ *such that* $H\sigma = G_i$.

$[Axiom\ ?^{dir}]$

$$\frac{\begin{array}{ll}\Sigma: & (\Delta_1 \to G_1), \ldots, (\Delta_{i-1} \to G_{i-1}), \\ & (\Delta_{i+1} \to G_{i+1}), \ldots, (\Delta_n \to G_n)\end{array}}{\Sigma: \quad (\Delta_1 \to G_1), \ldots, (\Delta_n \to G_n)}$$

if $\Delta_i = \{H \circ\!- \mathbf{1}\}$, *there exists a substitution* σ *such that* $H\sigma = G_i$.

The main difference introduced by this system is that synchronizations and reduction are led by the goal to be solved. We now have to prove that this system is equivalent to our initial system *FG*. The equivalence of the two systems is ensured by the following theorem:

Theorem 3 (*FG^{dir}* $\Leftrightarrow$ *FG*) *There exists a proof of a sequent in the system FG if and only if there exists a proof of this sequent in the system* FG^{dir}.

Proof: This proof is inspired by [19].
The transformation of a *FG* proof tree into a FG^{dir} proof derivation is achieved by recombining the different branches of the proof into a single derivation and then collapsing the synchronization rule with its first corresponding linear clause. The intermediate system FG^{lin} is needed to achieve the first part of the proof.

Definition 15 (FG^{lin} System)

$[Back\ !^{lin}]$

$$\frac{\begin{array}{c}(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_{i-1} : \Delta_{i-1} \to G_{i-1}), \\ (\Sigma_i : \Delta'_1 \to A_1\sigma), \ldots, (\Sigma_i : \Delta'_p \to A_p\sigma), \\ (\Sigma_{i+1} : \Delta_{i+1} \to G_{i+1}), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

where $(H \circ\!- A_1 \otimes \ldots \otimes A_p) \in \Psi$, $H\sigma = G_i$ *and* $\Delta_i = \uplus_{1 \leq k \leq p} \Delta'_k$.

[*Back* ?^{lin}]

$$\frac{\begin{array}{c}(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_{i-1} : \Delta_{i-1} \to G_{i-1}),\\ (\Sigma_i : \Delta'_1 \to A_1\sigma), \ldots, (\Sigma_i : \Delta'_p \to A_p\sigma),\\ (\Sigma_{i+1} : \Delta_{i+1} \to G_{i+1}), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

where $C = (H \multimapinv A_1 \otimes \ldots \otimes A_p) \in \Delta_i$*,* $H\sigma = G_i$ *and* $\Delta_i \setminus \{C\} = \uplus_{1 \leq k \leq p} \Delta'_k$*.*

[*Sync*^{lin}]

$$\frac{\begin{array}{c}(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_{i-1} : \Delta_{i-1} \to G_{i-1}),\\ (\Sigma_i, \alpha : C_1\theta, \ldots, C_r\theta, \Delta_i \to G_i),\\ (\Sigma_{i+1} : \Delta_{i+1} \to G_{i+1}), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

where $\forall c \exists nc (C_1 \otimes \ldots \otimes C_r) \in \Psi$*,* $\alpha \notin \Sigma_i$*,* $\beta \in \Sigma_i$ *and* $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$*.*

[*Axiom* !^{lin}]

$$\frac{\begin{array}{c}(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_{i-1} : \Delta_{i-1} \to G_{i-1}),\\ (\Sigma_{i+1} : \Delta_{i+1} \to G_{i+1}), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

where $(H \multimapinv \boldsymbol{1}) \in \Psi$*,* $\Delta_i = \emptyset$ *and* $H\sigma = G_i$*.*

[*Axiom* ?^{lin}]

$$\frac{\begin{array}{c}(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_{i-1} : \Delta_{i-1} \to G_{i-1}),\\ (\Sigma_{i+1} : \Delta_{i+1} \to G_{i+1}), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

where $\Delta_i = \{(H \multimapinv \boldsymbol{1})\}$ *and* $H\sigma = G_i$*.*

Lemma 1 (*FG* ⇔ *FG*^{lin}) *There exists proofs of the sequents* $\forall i, 1 \leq i \leq n, (\Sigma_i : \Delta_i \to G_i)$ *in the system FG if and only if there exists a proof of the sequent* $(\Sigma_1 : \Delta_1 \to G_1) \ldots (\Sigma_n : \Delta_n \to G_n)$ *in the system FG*^{lin}*.*

Proof of *FG* => *FG*^{lin}.
We prove by induction on the structure of a *FG* proof if $\forall i$, $1 \leq i \leq n$ there exists a *FG* proof of $(\Sigma_i : \Delta_i \to G_i)$ then there exists a *FG*^{lin} proof of $(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)$.

Base case [*Back* ?] + [**1**]**:** The *FG* proof is as follows:

$$\frac{\overline{\Sigma :\to \mathbf{1}}}{\Sigma : H \multimapinv \mathbf{1} \to G} \qquad \text{with } H\sigma = G$$

Then the derivation:

$$\overline{(\Sigma :\to G)}^{[Axiom\ !^{lin}]}$$

is a *FG*^{lin} proof.

Base case $[Back\ !]+[\mathbf{1}]$**:** This case is similar to the $[Back\ ?]+[\mathbf{1}]$ base case.

Induction case $[Sync]$**:** The inference at the root of the proof is an instance of the $[Sync]$ rule:

$$\dfrac{\dfrac{\nabla}{\alpha,\Sigma : C_1\theta,\dots,C_r\theta,\Delta \to G}}{\Sigma : \Delta \to G}{[Sync]}$$

with $(\forall c \exists nc(C_1 \otimes \dots \otimes C_r)) \in \Psi$, $\beta \in \Sigma, \alpha \notin \Sigma$ and $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$.
By induction hypothesis, for the FG proof tree:

$$\dfrac{\nabla}{\alpha,\Sigma : C_1\theta,\dots,C_r\theta,\Delta \to G}$$

there exists a FG^{lin} proof:

$$\dfrac{\mathcal{D}}{(\alpha,\Sigma : C_1\theta,\dots,C_r\theta,\Delta \to G)}$$

Then the following derivation:

$$\dfrac{\dfrac{\mathcal{D}}{(\alpha,\Sigma : C_1\theta,\dots,C_r\theta,\Delta \to G)}}{(\Sigma : \Delta \to G)}$$

is a FG^{lin} proof.

Induction case $[Back\ ?]$**:** The inference at the root of the proof is an instance of the $[Back\ ?]$ rule:

$$\dfrac{\dfrac{\nabla_1}{\Sigma : \Delta_1 \to A_1\sigma} \quad \dots \quad \dfrac{\nabla_p}{\Sigma : \Delta_p \to A_p\sigma}}{\Sigma : H \circ\!\!- A_1 \otimes \dots \otimes A_p, \Delta_1, \dots, \Delta_p \to G} \qquad \text{with } H\sigma = G.$$

By induction hypothesis for every $1 \leq i \leq n$ there is a FG^{lin} proof:

$$\dfrac{\dfrac{\overline{S_{l_i}}}{\begin{array}{c}\vdots\\ \{S_j\}_{j \in I_i}\\ \vdots\end{array}}}{(\Sigma : \Delta_i \to A_i\sigma)}$$

Let give the following FG^{lin} derivation $\forall i, 1 \leq i \leq p$:

$$\dfrac{\mathcal{D}_i \left\{ \begin{array}{c} S_{l_i}, \{(\Sigma : \Delta_k \to A_k\sigma)\}_{i<k\leq p} \\ \vdots \\ \{S_j\}_{j \in I_i} \cup \{(\Sigma : \Delta_k \to A_k\sigma)\}_{i<k\leq p} \\ \vdots \end{array} \right.}{(\Sigma : \Delta_i \to A_i\sigma), \{(\Sigma : \Delta_k \to A_k\sigma)\}_{i<k\leq p}}$$

Then the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{S_{l_p}}}{\mathcal{D}_p}}{(\Sigma : \Delta_p \to A_p\sigma)}}{S_{l_{p-1}}, (\Sigma : \Delta_p \to A_p\sigma)}}{\begin{array}{c}\vdots\\ \dfrac{\dfrac{\mathcal{D}_{1<i\leq p}}{(\Sigma : \Delta_i \to A_i\sigma), \{(\Sigma : \Delta_k \to A_k\sigma)\}_{1<i\leq k\leq p}}}{S_{l_{i-1}}, \{(\Sigma : \Delta_k \to A_k\sigma)\}_{1<i<k\leq p}} \\ \vdots \\ \dfrac{\dfrac{\mathcal{D}_1}{(\Sigma : \Delta_1 \to A_1\rho), \{(\Sigma : \Delta_k \to A_k\sigma)\}_{1<k\leq p}}}{(\Sigma : \Delta_1, \ldots, \Delta_p \to G)}\end{array}}$$

is a FG^{lin} proof.

Induction case [*Back* !]**:** This case is similar to the [*Back* ?] case.

□

Proof of $FG^{lin} => FG$.
We prove by induction on the structure of a FG^{lin} proof if there exists a FG^{lin} proof of $(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)$ then $\forall i$, $1 \leq i \leq n$ there exist FG proofs for $(\Sigma_i : \Delta_i \to G_i)$.

Base case [*Axiom* $?^{lin}$]**:** The FG^{lin} proof is as follows:

$$\frac{}{(\Sigma : H \multimapinv \mathbf{1} \to G)}[Axiom\ ?^{lin}] \qquad \text{with } H\sigma = G.$$

The proof tree:

$$\dfrac{\overline{\Sigma :\to \mathbf{1}}[\mathbf{1}]}{\Sigma : H \multimapinv \mathbf{1} \to G}[Back\ ?]$$

is a FG proof.

Base case [*Axiom* $!^{lin}$]:]

This case is similar to the [*Axiom* ?] base case.

Induction case [*Back* $?^{lin}$]**:** The inference at the root of the proof is an instance of the [*Back* $?^{lin}$] rule (without loss of generality $i = 1$):

$$\dfrac{\mathcal{D}}{\dfrac{(\Sigma_1 : \Delta_1' \to A_1\sigma), \ldots, (\Sigma_1 : \Delta_p' \to A_p\sigma), (\Sigma_2 : \Delta_2 \to G_2), \ldots (\Sigma_n : \Delta_n \to G_n)}{(\Sigma_1 : \Delta_1, H \multimapinv A_1 \otimes \ldots \otimes A_p \to G_1), (\Sigma_2 : \Delta_2 \to G_2), \ldots (\Sigma_n : \Delta_n \to G_n)}}$$

with $H\sigma = G_1$. By induction hypothesis, $\forall j$, $1 \le j \le p$, there exists a FG proof:

$$\frac{\nabla'_j}{\Sigma_1 : \Delta'_j \to A_j\sigma}$$

and $\forall i$, $2 \le i \le n$, there exists a FG proof:

$$\frac{\nabla_i}{\Sigma_i : \Delta_i \to G_i}$$

Then the proof tree:

$$\frac{\dfrac{\nabla'_1}{\Sigma_1 : \Delta'_1 \to A_1\sigma} \quad \dots \quad \dfrac{\nabla'_p}{\Sigma_1 : \Delta'_p \to A_p\sigma}}{\Sigma_1 : \Delta'_1, \dots, \Delta'_p \to G_1}$$

is a FG proof.

Induction cases $[Axiom\ !^{lin}]$**,** $[Axiom\ ?^{lin}]$ **and** $[Back\ !^{lin}]$**:** The cases $[Axiom\ !^{lin}]$, $[Axiom\ ?^{lin}]$ and $[Back\ !^{lin}]$ are similar to the $[Back\ ?^{lin}]$ induction case.

Induction case $[Sync^{lin}]$**:** The inference at the root of the proof is an instance of the $[Sync^{lin}]$ rule (without loss of generality $i = 1$):

$$\frac{\dfrac{\mathcal{D}}{(\Sigma_1, \alpha : C_1\theta, \dots, C_r\theta, \Delta_1 \to G_1), (\Sigma_2 : \Delta_2 \to G_2), \dots, (\Sigma_n : \Delta_n \to G_n)}}{(\Sigma_1 : \Delta_1 \to G_1), \dots, (\Sigma_n : \Delta_n \to G_n)}$$

with $\forall c \exists nc(C_1 \otimes \dots \otimes C_r) \in \Psi$, $\beta \in \Sigma_1$, $\alpha \notin \Sigma_1$ and $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$. By induction hypothesis, there exist $\forall i$, $2 \le i \le n$, a FG proof:

$$\frac{\nabla_i}{\Sigma_i : \Delta_i \to G_i}$$

and a FG proof:

$$\frac{\nabla_1}{\Sigma_1, \alpha : C_1\theta, \dots, C_r\theta, \Delta_1 \to G_1}$$

Then the proof tree:

$$\frac{\dfrac{\nabla_1}{\Sigma_1, \alpha : C_1\theta, \dots, C_r\theta, \Delta_1 \to G_1}}{\Sigma_1, \alpha : \Delta_1 \to G_1}$$

is a FG proof.

□

The following definitions and lemmas are technical ones to establish the equivalence between FG^{lin} and FG^{dir} systems.

Lemma 2 (*FG* Restriction Lemma)
If there exists a FG proof then there exists a restricted FG proof.

Proof of the *FG* restriction.
By (a double) induction on the distance between the closer instance of the [*Sync*] rule to the root and the root, (and the number of instances of [*Sync*] rule which is straight forward), we prove that if there exists a *FG* derivation then there exists a restricted *FG* derivation with the same first and last resolvents.

Base case: An instance of the [*Sync*] rule is already at the root.

Induction case: The instance of the rule at the root of the derivation is either an instance of [*Back* !], [*Back* ?] or [**1**] rules. The [**1**] is trivial. The [*Back* ?] case is only treated, the [*Back* !] is similar. The derivation is of the form (without loss of generality $i = 1$):

$$\dfrac{\dfrac{\nabla}{\Sigma : \Delta_1 \to A_1\sigma \ldots \Sigma : \Delta_p \to A_p\sigma}}{\Sigma : \Delta \to G}$$

with

$$\begin{array}{rcl} C & = & (H \circ\!\!- A_1 \otimes \ldots \otimes A_p) \in \Delta, \\ H\sigma & = & G, \\ \Delta \setminus \{C\} & = & \uplus_{1 \leq k \leq p} \Delta_p. \end{array}$$

By induction hypothesis, there exists a *FG* derivation ∇' with an instance of the [*Sync*] rule at the root, the same number of instances of the [*Sync*] rule and the same first and last resolvents as ∇. A restricted *FG* derivation is obtained by permuting this instance of the [*Sync*] rule and the instance of the [*Back* ?] rule and inserting in the signature of every sequents of the descendants of $(\Sigma : \Delta_k \to A_k\sigma)$ $(\forall k, 1 \leq k \leq p)$ in the derivation ∇' the constant α inserted by the instance of the [*Sync*] rule.

□

At this stage, we need more intermediate definitions to continue the proof.

Definition 16 (Restricted FG^{lin} proof)
A FG^{lin} proof is in restricted *form if all the instances of the [$Sync^{lin}$] rule are at the root of the proof.*

Definition 17 (Initial instance)
The initial instance w.r.t. an instance of the [$Sync^{lin}$] rule is the instance of [Back $?^{lin}$] or [Axiom $?^{lin}$] rules which uses for the first time one of the linear clauses introduced by the [$Sync^{lin}$] rule.

Definition 18 (Ordered FG^{lin} proof)
A FG^{lin} proof is ordered if the order on the Eigenvariables induced by the instances of the [$Sync^{lin}$] rules is the same as the one induced by their initial instances.

Lemma 3 (FG^{lin} Restriction Lemma)
If there exists a FG^{lin} proof then there exists an restricted FG^{lin} proof.

Proof of the restriction lemma for FG^{lin}.
By (a double) induction on the distance between the closer instance of the $[Sync^{lin}]$ rule to the root and the root, (and the number of instances of $[Sync^{lin}]$ rule which is straight forward), we prove that if there exists a FG^{lin} derivation then there exists a restricted FG^{lin} derivation with the same first and last resolvents.

Base case: An instance of the $[Sync^{lin}]$ rule is already at the root.

Induction case: The instance of the rule at the root of the derivation is either an instance of $[Back\ !^{lin}]$, $[Axiom\ !^{lin}]$, $[Back\ ?^{lin}]$ or $[Axiom\ ?^{lin}]$ rules. The $[Back\ ?^{lin}]$ case is only treated, there others are similar. The derivation is of the form (without loss of generality $i = 1$):

$$\cfrac{\cfrac{\nabla}{\begin{array}{c}(\Sigma_1 : \Delta'_1 \to A_1\sigma), \ldots, (\Sigma_1 : \Delta'_p \to A_p\sigma),\\ (\Sigma_2 : \Delta_2 \to G_2), \ldots, (\Sigma_n : \Delta_n \to G_n)\end{array}}}{(\Sigma_1 : \Delta_1 \to G_1), \ldots, (\Sigma_n : \Delta_n \to G_n)}$$

with $C = (H \circ\!- A_1 \otimes \ldots \otimes A_p) \in \Delta_1$, $H\sigma = G_1$ and $\Delta_1 \setminus \{C\} = \uplus_{1 \leq k \leq p} \Delta'_p$. By induction hypothesis, there exists a FG^{lin} derivation ∇' with an instance of the $[Sync^{lin}]$ rule at the root, the same number of instances of the $[Sync^{lin}]$ rule and the same first and last resolvents as ∇. A restricted FG^{lin} derivation is obtained by permuting this instance of the $[Sync^{lin}]$ rule and the instance of the $[Back\ ?^{lin}]$ rule and inserting in the signature of every sequents of the descendants of $(\Sigma_1 : \Delta'_k \to A_k\sigma)$ $(\forall k, 1 \leq k \leq p)$ in the derivation ∇' the constant α inserted by the instance of the $[Sync^{lin}]$ rule.

□

Lemma 4 (Ordered Lemma)
If there exists a FG^{lin} proof then there exists a restricted, ordered FG^{lin} proof.

Proof of the order lemma for FG^{lin}.
By the restriction lemma, for any FG^{lin} proof there exists a restricted FG^{lin} proof. The existential introduction induces a partial order on the Eigenvariables. The order on the initial instances respects this partial order. Two instances of $[Sync^{lin}]$ rule can be permuted if this permutation respects the partial order. The order of the initial instances can be chosen as the order of introduction of the Eigenvariables. □

Lemma 5 ($FG^{lin} \Leftrightarrow FG^{dir}$)
There exists a proof of the sequent $(\Sigma_1 : \Delta_1 \to G_1) \ldots (\Sigma_n : \Delta_n \to G_n)$ in the system FG^{lin} if and only if there exists a proof of the sequent $(\Sigma_1 : \Delta_1 \to G_1) \ldots (\Sigma_n : \Delta_n \to G_n)$ in the system FG^{dir}.

Proof of $FG^{lin} => FG^{dir}$.
We prove by induction on the number of instances of the $[Sync^{lin}]$ rule if there exists a restricted, ordered FG^{lin} derivation of $(\# :\to G)$ then there exists a FG^{dir} derivation of $(\# :\to G)$ with the same last resolvent.

Case $l = 0$: The FG^{lin} derivation contains no instance of $[Sync^{lin}]$ rule so this derivation is also a FG^{dir} derivation.

Case $l > 0$: The restricted, ordered FG^{lin} derivation is composed of two parts:

- a derivation $\mathcal{R}$ containing only instances of the $[Back\ !^{lin}]$, $[Axiom\ !^{lin}]$, $[Back\ ?^{lin}]$ and $[Axiom\ ?^{lin}]$ rules:

$$\frac{\nabla}{(\Sigma, \alpha : \Delta, C_1\theta, \ldots, C_r\theta \to G)}$$

- and a derivation $\mathcal{S}$ containing only l instances of $[Sync^{lin}]$ rule with the last applied on

$$\forall c \exists nc(C_1 \otimes \ldots \otimes C_r) \in \Psi, \theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}, \beta \in \Sigma, \alpha \notin \Sigma : \tag{5}$$

$$\cfrac{\cfrac{(\Sigma, \alpha : \Delta, C_1\theta, \ldots, C_r\theta \to G)}{(\Sigma : \Delta \to G)}}{\cfrac{\vdots}{(\# : \to G)}}$$

By induction hypothesis $\mathcal{R}$ contains the initial instance corresponding to the instance of this $[Sync^{lin}]$. It can be an instance of the $[Back\ ?^{lin}]$ or $[Axiom\ ?^{lin}]$ rules. The $[Back\ ?^{lin}]$ case is only treated, the $[Axiom\ ?^{lin}]$ case is similar. The derivation $\mathcal{R}$ is of the form (without loss of generality $i = 1$):

$$\begin{array}{ll}
\mathcal{R}' & \left\{ \dfrac{\nabla}{\begin{array}{c}(\Sigma, \alpha : \Delta'_1 \cup \Delta'''_1 \to A_1\sigma), \ldots, (\Sigma, \alpha : \Delta'_p \cup \Delta'''_p \to A_p\sigma), \\ (\Sigma, \alpha : \Delta_2 \cup \Delta''_2 \to G_2), \ldots, (\Sigma, \alpha : \Delta_n \cup \Delta''_n \to G_n)\end{array}} \right. \\ \hline
\mathcal{D}' & \left\{ \begin{array}{c} \dfrac{\begin{array}{c}(\Sigma, \alpha : \Delta_1 \cup (\bigcup_{1 \le k \le p} \Delta'''_k) \cup \{H \multimapinv A_1 \otimes \ldots \otimes A_p\} \to G_1), \\ (\Sigma, \alpha : \Delta_2 \cup \Delta''_2 \to G_2), \ldots, (\Sigma, \alpha : \Delta_n \cup \Delta''_n \to G_n)\end{array}}{\vdots} \\ \hline (\Sigma, \alpha : \Delta, C_1\theta, \ldots, C_r\theta \to G) \end{array} \right.
\end{array}$$

with

$$\begin{aligned}
C_1\theta, \ldots, C_{j-1}\theta, C_{j+1}\theta, \ldots, C_r\theta &= (\uplus_{1 < k \le n} \Delta''_k) \uplus (\uplus_{1 \le k \le p} \Delta'''_k), \\
\uplus_{1 \le k \le p} \Delta'_k &= \Delta_1, \\
C_j\theta &= H \multimapinv A_1 \otimes \ldots \otimes A_p, \\
H\sigma &= G_1.
\end{aligned}$$

The derivation $\mathcal{R}$ is restricted and ordered then $\mathcal{R}'$ contains only instances of the $[Back\ !^{lin}]$, $[Axiom\ !^{lin}]$, $[Back\ ?^{lin}]$ and $[Axiom\ ?^{lin}]$ rules (on non initial instances) then the derivation $\mathcal{R}'$ is also a FG^{dir} derivation.

Let $\mathcal{D}''$ be the same derivation as $\mathcal{D}'$ without the instances $C_k\theta$, $1 \leq k \leq r$ and α. Since the derivation $\mathcal{R}$ is ordered the derivation $\mathcal{D}''$ is still a FG^{lin} derivation. So by induction hypothesis on the FG^{lin} derivation below formed from $\mathcal{S}$ without the last inference and $\mathcal{D}''$:

$$\cfrac{\cfrac{\cfrac{(\Sigma : \Delta_1 \rightarrow G_1), \ldots, (\Sigma : \Delta_n \rightarrow G_n)}{\vdots}}{(\Sigma : \Delta \rightarrow G)}}{\cfrac{\vdots}{(\# :\rightarrow G)}}$$

there is a FG^{dir} derivation $\mathcal{D}'''$:

$$\cfrac{\cfrac{(\Sigma : \Delta_1 \rightarrow G_1), \ldots, (\Sigma : \Delta_n \rightarrow G_n)}{\vdots}}{(\# :\rightarrow G)}$$

So the derivation:

$$\cfrac{\mathcal{R}' \left\{ \cfrac{\nabla}{\begin{array}{c}(\Sigma, \alpha : \Delta'_1 \cup \Delta'''_1 \rightarrow A_1\sigma), \ldots, (\Sigma, \alpha : \Delta'_p \cup \Delta'''_p \rightarrow A_p\sigma),\\ (\Sigma, \alpha : \Delta_2 \cup \Delta''_2 \rightarrow G_2), \ldots, (\Sigma, \alpha : \Delta_n \cup \Delta''_n \rightarrow G_n)\end{array}} \right.}{\mathcal{D}''' \left\{ \cfrac{\cfrac{(\Sigma : \Delta_1 \rightarrow G_1), \ldots, (\Sigma : \Delta_n \rightarrow G_n)}{\vdots}}{(\# :\rightarrow G)} \right.}$$

with $\{C_1\theta, \ldots, C_{j-1}\theta, C_{j+1}\theta, \ldots, C_r\theta\} = (\uplus_{1<k\leq n}\Delta''_k) \uplus (\uplus_{1\leq k\leq p}\Delta'''_k)$ and $\uplus_{1\leq k\leq p}\Delta'_k = \Delta_1$ in a FG^{dir} derivation.

$\square$

Proof of $\boldsymbol{FG^{dir}} => \boldsymbol{FG^{lin}}$.

We prove by induction on the number of instances of the $[Sync{+}^{dir}]$ or $[Sync\mathbf{1}^{dir}]$ rules if there exists a FG^{dir} derivation of $(\# :\rightarrow G)$ then there exists a FG^{lin} derivation of $(\# :\rightarrow G)$ with the same last resolvent.

Case $l = 0$: The FG^{dir} derivation contains no instance of $[Sync{+}^{dir}]$ nor $[Sync\mathbf{1}^{dir}]$ rule so this derivation is also a FG^{lin} derivation.

Case $l > 0$: The FG^{dir} derivation is composed of two parts:

- a derivation $\mathcal{R}$ containing only instances of the $[Back\ !^{dir}]$, $[Axiom\ !^{dir}]$, $[Back\ ?^{dir}]$ and $[Axiom\ ?^{dir}]$ rules;
- and a derivation $\mathcal{D}$ containing only l instances of $[Sync{+}^{dir}]$ or $[Sync\mathbf{1}^{dir}]$ rule;

and a link inference of $[Sync+^{dir}]$ or $[Sync\mathbf{1}^{dir}]$ between $\mathcal{R}$ and $\mathcal{D}$ applied on $\forall c \exists nc (C_1 \otimes \ldots \otimes C_r) \in \Psi$ with $\theta = \{c \leftarrow \beta, nc \leftarrow \alpha\}$, $\beta \in \Sigma$, $\alpha \notin \Sigma$.

The $[Sync+^{dir}]$ case is only treated, the $[Sync\mathbf{1}^{dir}]$ case is similar.

The derivation $\mathcal{D}$ is of the form (without loss of generality $i = 1$):

$$
\begin{array}{cc}
\mathcal{R} & \left\{ \begin{array}{c} \nabla \\ \hline (\Sigma,\alpha : \Delta'_1 \cup \Delta'''_1 \to A_1\sigma), \ldots, (\Sigma,\alpha : \Delta'_p \cup \Delta'''_p \to A_p\sigma), \\ (\Sigma,\alpha : \Delta_2 \cup \Delta''_2 \to G_2), \ldots, (\Sigma,\alpha : \Delta_n \cup \Delta''_n \to G_n) \end{array} \right. \\ \hline
\mathcal{D}' & \left\{ \begin{array}{c} (\Sigma : \Delta_1 \to G_1), \ldots, (\Sigma : \Delta_n \to G_n) \\ \hline \vdots \\ \hline (\# :\to G) \end{array} \right.
\end{array}
$$

with $\{C_1\theta, \ldots, C_{j-1}\theta, C_{j+1}\theta, \ldots, C_n\theta\} = (\uplus_{1<k\leq n}\Delta''_k) \uplus (\uplus_{1\leq k\leq p}\Delta'''_k)$, $\uplus_{1\leq k\leq p}\Delta'_k = \Delta_1$, $C_j\theta = H \multimap A_1 \otimes \ldots \otimes A_p$ and $H\sigma = G_1$. The derivation $\mathcal{R}$ contains only instances of $[Back\ !^{dir}]$, $[Axiom\ !^{dir}]$, $[Back\ ?^{dir}]$ and $[Axiom\ ?^{dir}]$ so it is also a FG^{lin} derivation. By induction hypothesis on the FG^{dir} derivation $\mathcal{D}'$ containing $l-1$ instances of $[Sync+^{dir}]$ or $[Sync\mathbf{1}^{dir}]$ rules there exists a FG^{lin} derivation:

$$
\begin{array}{cc}
\mathcal{D}'' & \left\{ \begin{array}{c} (\Sigma : \Delta_1 \to G_1), \ldots, (\Sigma : \Delta_n \to G_n) \\ \hline \vdots \\ \hline (\Sigma : \Delta \to G) \end{array} \right. \\ \hline
\mathcal{S} & \left\{ \begin{array}{c} \vdots \\ \hline (\# :\to G) \end{array} \right.
\end{array}
$$

Let $\mathcal{D}'''$ be the same derivation as $\mathcal{D}''$ but for all $(\Sigma' : \Delta' \to G')$ of $\mathcal{D}''$:

- if G' is an ancestor of G_k, $1 < k \leq n$, then Δ''_k is added to Δ';
- if G' is an ancestor of G_1 then $(\bigcup_{1\leq k\leq p}\Delta'''_k) \cup \{C_j\theta\}$ is added to Δ';
- α is added to Σ.

Then the derivation:

$$
\begin{array}{cc}
\mathcal{R} & \left\{ \begin{array}{c} \nabla \\ \hline (\Sigma,\alpha : \Delta'_1 \cup \Delta'''_1 \to A_1\sigma), \ldots, (\Sigma,\alpha : \Delta'_p \cup \Delta'''_p \to A_p\sigma), \\ (\Sigma,\alpha : \Delta_2 \cup \Delta''_2 \to G_2), \ldots, (\Sigma,\alpha : \Delta_n \cup \Delta''_n \to G_n) \end{array} \right. \\ \hline
\mathcal{D}''' & \left\{ \begin{array}{c} (\Sigma,\alpha : \Delta_1 \cup (\bigcup_{1\leq k\leq p}\Delta'''_k) \cup \{C_j\theta\} \to G_1), \\ (\Sigma,\alpha : \Delta_2 \cup \Delta''_2 \to G_2), \ldots, (\Sigma,\alpha : \Delta_n \cup \Delta''_n \to G_n) \\ \hline \vdots \\ \hline (\Sigma,\alpha : \Delta, C_1\theta, \ldots, C_r\theta \to G) \end{array} \right. \\
& \begin{array}{c} \hline (\Sigma : \Delta \to G) \end{array} \\
\mathcal{S} & \left\{ \begin{array}{c} \hline \vdots \\ \hline (\# :\to G) \end{array} \right.
\end{array}
$$

is a FG^{lin} derivation. $\square$

We have to mention here that this system provides a goal directed approach for the recognition of an element of a tree language. This kind of operation is not practically possible using the initial grammar definition which gives a method to produce elements of the language but does not give any strategy to recognize an element.
The introduction of unification and variable renaming in the system FG^{dir} would also ensure the generation of the element of the language. This has to be formally proved thanks to a lifting lemma (as it is done for the SLD resolution of Prolog [15]). As a consequence, the generation is available in the Prolog implementation of our system described in the next section.
Concerning the decidability of our approach, it is clear that problems encountered with DCG's appear here (mainly due to left recursions and empty transitions). Since Prolog is the underlying framework, the problems related to its depth left first strategy also occur. Thus, termination of our method depends on this search strategy.

5 Implementation Issue

This section briefly describes how the implementation can be achieved from the previous inference system. At this time, a library is able, taking a *TSG* (resp. a *PG*) as input, to provide, as output, a predicate `phrase_TSG` (resp. `phrase_PG`) which can recognize or generate a term for the defined language from the axiom. The implementation of the method described along this paper can be divided in two parts: the translation of the grammar into linear logic formulas (described in Section 3.1) and the implementation of the proof system FG^{dir} in Prolog. The following presentation of this implementation is related to Example 1.

5.1 Management of signature and linear context

The extension of the signature (i.e. new symbols introduced by existential quantifiers of synchronization) is inspired by the management of essentially universal quantifier (quantifier `pi` in the body of clauses) of λProlog [1]: only the cardinality of this extension is needed.

The management of linear clauses is meta-programmed by a linear program continuation LPC (i.e. a set containing the remaining linear clauses introduced by instances of [$Sync+^{dir}$] or [$Sync\mathbf{1}$] which have to be used later). This technique is inspired by the management of intuitionistic implication of λProlog [12, 1].

5.2 Transformation Function

Back to Example 1, the transformation is illustrated here by the implementation of the clause F_0 which corresponds to the free production R_0.

```
i(C, f(X, Y), SigmaIn, SigmaOut, DeltaIn, DeltaOut) :-
        x(C, X, SigmaIn, SigmaInter, DeltaIn, DeltaInter),
        y(C, Y, SigmaInter, SigmaOut, DeltaInter, DeltaOut).
```

`SigmaIn`, `SigmaInter` and `SigmaOut` represent the cardinality of the extension of the signature. `DeltaIn`, `DeltaInter` and `DeltaOut` represent the LPC.

5.3 Implementation of the Proof System FG^{dir} in Prolog

The inference rules [*Back* !] **and** [*Axiom* !] are handled by the reduction mechanism of Prolog.

The inference rules [*Back* ?] **and** [*Axiom* ?] are implemented by a meta-programming of Prolog by the predicate `linear` over the linear program continuation.

```
linear(PredicatSymbol, C, Term, SigmaIn, SigmaOut,
       [C_j|Cj_plus_1_Cm], C1_Cj_minus_1, DeltaOut) :-
   C_j = (Head :- Body),
   Head =.. [PredicatSymbol, C, Term, SigmaIn, SigmaOut,
             C1_Cm_but_Cj, DeltaOut],
   append(C1_Cj_minus_1, Cj_plus_1_Cm, C1_Cm_but_Cj),
   call(Body).
```

The linear clause `C_j` is meta-interpreted and then discarded from the linear program continuation: `C1_Cm_but_Cj` $=$ `C1_Cj_minus_1` $\uplus$ `Cj_plus_1_Cm` (see `append` predicate).
A recursive prolog clause is added for the program continuation traversal.

The inference rules [$Sync+^{dir}$] **and** [*Sync***1**] are implemented in a similar way as respectively [*Back* !] and [*Axiom* !] but with an increase of the linear program continuation by the linear clauses introduced by the synchronization.
The prolog clause below corresponds to the implementation of x^{**} for the translation of the pack of production $\{X \Rightarrow s(X), Y \Rightarrow s(s(Y))\}$:

$$(\forall c \exists nc((\forall X(x(s(X),c) \circ - x(X,nc))) \otimes (\forall Y(y(s(s(Y)),c) \circ - y(Y,nc))))).$$

```
x(C, s(X), SigmaIn, SigmaOut, DeltaIn, DeltaOut) :-
   NC is SigmaIn + 1,
   Cy = (y(C, s(s(Y)), SI, SO, DI, DO) :- y(NC, Y, SI, SO, DI, DO)),
   x(NC, X, NC, SigmaOut, [Cy|DeltaIn], DeltaOut).
```

The linear clause $\sigma_S(Y \Rightarrow s(s(Y)),0,1) = (\forall Y(y(s(s(Y)),0) \circ - y(Y,1)))$ is implemented by `Cy` added to the linear program continuation.

5.4 Reduction strategy

Due to the leftmost selection rule and the depth-first search strategy of Prolog the reduction strategy is a left-outermost strategy as shown on the trace below for the goal `phrase_TSG(f(s(z),s(s(z))),i)` (the clause `(y(0, s(s(Y)), SI, SO, DI, DO) :- y(1, Y, SI, SO, DI, DO))` is denoted *Cy*). The first column is the (simplified ††) prolog trace, the second one contains the linear program continuation and the third one the term already recognized.

** There is of course a symmetrical clause for *y*.

†† In the real system, variables are denoted as in Prolog (i.e. of the form _1234). For the sake of readability, explicit names have been introduced and meta-programming arguments have been discarded.

`phrase_TSG(f(s(z),s(s(z))),i)`	LPC	
`i(0,f(s(z),s(s(z))))`	$\{\}$	`(I,0)`
`x(0,s(z)),y(0,s(s(z)))`	$\{\}$	`f((X, 0), (Y, 0))`
`x(1,z),y(0,s(s(z)))`	$\{Cy\}$	`f(s((X, 1)), (Y, 0))`
`y(0,s(s(z)))`	$\{Cy\}$	`f(s(z), (Y, 0))`
`y(1,z)`	$\{\}$	`f(s(z), s(s((Y, 1))))`
	$\{\}$	`f(s(z), s(s(z)))`

5.5 Practical Use of the System

We can now illustrate the complete use of our system on PG's Example 2. The user has only to write as input the following prolog clause:

```
pg_with_trace :-
              name(FileName, "pg_with_trace.pl"),
              open(FileName,write,Stream),
              sigma_pg(pg(f,[f,g],[
                   [(f(0) -> z), (f(s(n)) -> g(s(n))*f(n))],
                   [(g(0) -> z),(g(s(n)) -> s(g(n)))]
                 ]),Stream),
              close(Stream).
```

in order to provide the description of the PG thanks to the predicate `sigma_pg`. The execution of this clause produces file `pg_with_trace.pl`, containing a predicate `phrase_pg`, which can be loaded and used for grammar recognition as:

```
| ?- phrase_pg(s(z)*z,f,[s(0)]).
    g*f
    s(g)*f
    s(z)*f
    s(z)*z

yes
| ?-
```

This execution shows that $s(z) * z$ belongs to the PG and how it can be generated from the PG.
The source code of this implementation is available at:

http://www.info.univ-angers.fr/pub/stephan/Research/Download.html.

6 Conclusion

In this paper, we describe an implementation scheme for particular types of tree language including specific control features. The main idea is to provide an uniform framework in order to compute over tree grammars. As it has been done for word grammars with DCG [2], we propose a transformation method that allows us to get a set of Prolog Horn clauses from a grammatical definition of a tree language. This method consists in translating grammar derivation into proof search using a sequent calculus proof system based on linear logic. By successive refinements, we get a goal directed procedure implemented in Prolog. Since such tree languages appear as powerful tools for the schematization of sets of terms to represent solutions of symbolic computation problems, it seemed necessary to define a method to use these representations. Moreover, this approach allows us to embed various formalism into a unique framework, were computations can include Regular Languages, *TSG* and PG.

Acknowledgements

We would like to thank Miki Hermann, Sébastien Limet and Piere Réty for fruitful discussions on this work.

References

[1] P. Brisset. *Compilation de λProlog*. PhD thesis, Thèse de doctorat de l'université de Rennes, 1992.

[2] J. Cohen and T.J. Hickey. Parsing and compiling using prolog. *ACM Transactions on Programming Languages and Systems*, 9(2):125–163, 1987.

[3] H. Comon, M. Dauchet, R. Gilleron, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. 1997.

[4] N. Dershowitz and J.P. Jouannaud. *Rewrite Systems*, volume B, chapter 6, pages 243–309. J. Van Leeuwen, 1990.

[5] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[6] F. Gecseg and M. Steinby. *Handbook of Formal Languages*, volume 3, chapter Tree Languages, pages 1–68. Springer-Verlag, 1997.

[7] J.-Y. Girard. Linear logic, its syntax and semantics. In Regnier Girard, Lafont, editor, *Advances in Linear Logic*, number 222 in London Mathematical Society Lecture Notes Series, pages 355–419. Cambridge University Press, 1993.

[8] Jean-Yves Girard. Linear Logic. *Theoretical Computer Science*, (50):1–102, 1987.

[9] V. Gouranton, P. Réty, and H. Seidl. Synchronized Tree Languages Revisited and New Applications. In *Proceedings of the 6th Conference on Foundations of Science and Computation Structures*, LNCS, Genova (Italy), 2001. Springer Verlag.

[10] P. De Groote and G. Perrier. A Note on Kobayashi's and Yonezawa's "Asynchronous Communication Model Based on Linear Logic". *Formal Aspects of Computing*, 10, 1998.

[11] M. Hermann and R. Galbavy. Unification of infinite sets of terms schematized by primal grammars. *Theoretical Computer Science*, 176, 1997.

[12] J. S. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. In *Proceedings of LICS'91*, pages 32–42, 1991.

[13] S. Limet and P. Réty. E-Unification by Means of Tree Tuple Synchronized Grammars. *Discrete Mathematics and Theoretical Computer Science*, 1:69–98, 1997.

[14] S. Limet and F. Saubion. Primal Grammars for *R*-unification. In *PLILP/ALP'98*, number 1490 in LNCS. Springer-Verlag, 1998.

[15] J.W. Lloyd. *Foundations of Logic Programming*. Symbolic Computation series. Springer Verlag, 1987.

[16] Dale Miller. A Multiple-Conclusion Meta-Logic. In *LICS 1994*, pages 272–281, 1994.

[17] G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[18] F. Saubion and I. Stephan. On Implementation of Tree Synchronized Languages. In *Proceedings of 10th Conference on Rewriting Techniques and Applications*, LNCS. Springer Verlag, 1999.

[19] I. Stéphan. *Nouvelles fondations pour la programmation en logique disjonctive*. PhD thesis, Thèse de doctorat de l'université de Rennes, 1995.