Learnability of Pregroup Grammars

Denis Béchet, Annie Foret, Isabelle Tellier

D. Béchet
A. Foret
I. Tellier

Learnability of Pregroup Grammars

Abstract. This paper investigates the learnability by positive examples in the sense of Gold of Pregroup Grammars. In a first part, Pregroup Grammars are presented and a new parsing strategy is proposed. Then, theoretical learnability and non-learnability results for subclasses of Pregroup Grammars are proved. In the last two parts, we focus on learning Pregroup Grammars from a special kind of input called feature-tagged examples. A learning algorithm based on the parsing strategy presented in the first part is given. Its validity is proved and its properties are examplified.

Keywords: Learning from positive examples, Pregroup grammars, Computational linguistics, parsing, Categorial Grammars, constraints.

1. Introduction

Pregroup Grammars [18] (PGs in short) is a context-free grammar formalism recently appeared in the field of computational linguistics. This formalism allies expressivity (in this respect it is close to Lambek Grammars) and computational efficiency. Subtle linguistic phenomena have already been treated in this framework [1, 9]. PGs share many features with Categorial Grammars of which they are inheritors, especially their lexicalized nature.

Since the seminal works of Kanazawa [17] (using unification procedures from [8]), a lot of learnability results in Gold's model [16] have been obtained for various classes of Categorial Grammars and various input data. The learnability of PGs has yet received much less attention, except a negative result in [3]. This paper is an extended version of an article presented at ICGI in 2004 [4] on this subject. In Section 3, we prove several results of learnability or of non-learnability for classes of PGs. These first results are mainly theoretical and are not associated with learning algorithms.

Section 4 defines a new learning algorithm to specify a set of PGs compatible with some input data[1]. The input data considered, called feature-tagged examples, are richer than strings but chosen to be language-independent (inspired by [12, 13, 11]). The algorithm, whose properties are detailed in

Presented by **Name of Editor**; *Received* December 1, 2002

[1]a PG is said compatible with some examples when it generates at least these examples.

Studia Logica **68**: 1–28, 2001.

Section 5, implements a *specialization strategy*. One of its originalities is to reconsider the learning problem as a constraints resolution problem.

2. Pregroup grammars

2.1. Background

DEFINITION 2.1 (Pregroup). A *pregroup* is a structure $(P, \leq, \cdot, l, r, 1)$ such that $(P, \leq, \cdot, 1)$ is a partially ordered monoid[2] and l, r are two unary operations on P that satisfy: $\forall a \in P$: $a^l a \leq 1 \leq a a^l$ and $a a^r \leq 1 \leq a^r a$. The following equations follow from this definition: $\forall a, b \in P$, we have $a^{rl} = a = a^{lr}$, $1^r = 1 = 1^l$, $(a \cdot b)^r = b^r \cdot a^r$, $(a \cdot b)^l = b^l \cdot a^l$. Iterated adjoints[3] are defined for $i \in \mathbb{Z}$: $a^{(0)} = a$, for $i \leq 0 : a^{(i-1)} = \left(a^{(i)}\right)^l$, for $i \geq 0 : a^{(i+1)} = \left(a^{(i)}\right)^r$

DEFINITION 2.2 (Free Pregroup). Let $(P, \leq)$ be a partially ordered set of primitive categories[4], $P^{(\mathbb{Z})} = \{p^{(i)} \mid p \in P, i \in \mathbb{Z}\}$ is the set of atomic categories and $\mathcal{C}at_{(P,\leq)} = \left(P^{(\mathbb{Z})}\right)^* = \{p_1^{(i_1)} \cdots p_n^{(i_n)} \mid 1 \leq k \leq n, p_k \in P, i_k \in \mathbb{Z}\}$ is the set of categories. For $X, Y \in \mathcal{C}at_{(P,\leq)}$, $X \leq Y$ iff this relation is deducible in the system in Fig. 1 where $p, q \in P$, $n, k \in \mathbb{Z}$ and $X, Y, Z \in \mathcal{C}at_{(P,\leq)}$. This construction, proposed by Buszkowski [7], defines a pregroup that extends $\leq$ on P to $\mathcal{C}at_{(P,\leq)}$.

$$X \leq X \quad (Id) \qquad \frac{XY \leq Z}{Xp^{(n)}p^{(n+1)}Y \leq Z}\,(A_L) \qquad \frac{Xp^{(k)}Y \leq Z}{Xq^{(k)}Y \leq Z}\,(IND_L)$$

$$\frac{X \leq Y \quad Y \leq Z}{X \leq Z}\,(Cut) \qquad \frac{X \leq YZ}{X \leq Yp^{(n+1)}p^{(n)}Z}\,(A_R) \qquad \frac{X \leq Yq^{(k)}Z}{X \leq Yp^{(k)}Z}\,(IND_R)$$

$q \leq p$ if k is even
or $p \leq q$ if k is odd

Figure 1. System for Pregroup Grammars

Cut elimination. Every derivable inequality has a cut-free derivation.

Simple free pregroup. A *simple free pregroup* is a free pregroup where the order on primitive categories is equality.

---

[2] a *monoid* is a structure $< M, \cdot, 1 >$, such that $\cdot$ is associative and has a neutral element 1 ($\forall x \in M : 1 \cdot x = x \cdot 1 = x$). A partially ordered monoid is a monoid $(M, \cdot, 1)$ with a partial order $\leq$ that satisfies $\forall a, b, c$: $a \leq b \Rightarrow c \cdot a \leq c \cdot b$ and $a \cdot c \leq b \cdot c$.

[3] we use this notation in technical parts.

[4] we use the word "category" when one usually uses "type" to distinguish syntactical properties from semantical properties.

Definition 2.3 (Pregroup Grammars). $(P, \leq)$ is a finite partially ordered set. A *pregroup grammar* based on $(P, \leq)$ is a lexicalized[5] grammar $G = (\Sigma, I, s)$ such that $s \in P$; G assigns a category X to a string $v_1 \cdots v_n$ of Σ^* iff for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ such that $X_1 \cdots X_n \leq X$ in the free pregroup based on $(P, \leq)$. *The language* $\mathcal{L}(G)$ is the set of strings in Σ^* that are assigned s by G.

Rigid and k-valued grammars. Grammars that assign at most k categories to each symbol in the alphabet are called *k-valued grammars*; 1-valued grammars are also called *rigid* grammars.

Width. We define the width of a category $C = p_1^{u_1} \ldots p_n^{u_n}$ as $wd(C) = n$ (the number of atomic categories). The width of a grammar G is the maximum width of the categories assigned by G.

Example 2.4. *Our first example is taken from [19] with the basic categories: π_2 = second person, s_1 = statement in present tense, p_1 = present participle, p_2 = past participle, o = object. The sentence "You have been seeing her" gets category s_1 ($s_1 \leq s$), with successive reductions on $\pi_2 \pi_2^r \leq 1$, $p_2^l p_2 \leq 1$, $p_1^l p_1 \leq 1$, $o^l o \leq 1$:*

$$\begin{array}{ccccc} \textit{You} & \textit{have} & \textit{been} & \textit{seeing} & \textit{her} \\ \pi_2 & (\pi_2^r \, \underline{s_1} \, p_2^l) & (p_2 \, p_1^l) & (p_1 \, o^l) & o \end{array}$$

2.2. Parsing

Pregroup languages are context-free languages and their parsing is polynomial [6, 22]. We use here a parsing algorithm that works directly on lists of words [2]. The relations noted $\Gamma \vdash_{\mathcal{R}} \Delta$ where $\mathcal{R}$ consists in one or several rules are defined on lists of categories (p, q are atomic, X, Y range over categories and Γ, Δ over lists of categories):

- **M (merge)**: $\Gamma, X, Y, \Delta \vdash_M \Gamma, XY, \Delta$.
- **I (internal)**: $\Gamma, Xq^{(n)}p^{(n+1)}Y, \Delta \vdash_I \Gamma, XY, \Delta$, if $q \leq p$ and n is even or if $p \leq q$ and n is odd.
- **E (external)**: $\Gamma, Xq^{(n)}, p^{(n+1)}Y, \Delta \vdash_E \Gamma, X, Y, \Delta$, if $q \leq p$ and n is even or if $p \leq q$ and n is odd.

$\vdash_{\mathcal{R}}^*$ is the reflexive-transitive closure of $\vdash_{\mathcal{R}}$. This system is equivalent with the deduction system when the final right element is a primitive category. As a consequence, parsing can be done using $\vdash_{MIE}^*$.

[5] a lexicalized grammar is a triple (Σ, I, s): Σ is a finite alphabet, I assigns a finite set of categories to each $c \in \Sigma$, s is a category associated to correct sentences.

Lemma 2.5. *For* $X_1, \ldots, X_n \in \mathcal{C}at_{(P,\leq)}$ *and* $s \in P$*:*
$X_1 \cdots X_n \leq s$ *iff* $\exists p \in P$ *such that* $X_1, \ldots, X_n \vdash^*_{MIE} p$ *and* $p \leq s$.

Corollary 2.6. $G = (\Sigma, I, s)$ *generates a string* $v_1 \cdots v_n$ *iff for* $1 \leq i \leq N$, $\exists X_i \in I(v_i)$ *and* $\exists p \in P$ *such that* $X_1, \cdots, X_n \vdash^*_{MIE} p$ *and* $p \leq s$.

3. Learning

3.1. Background

We now recall some useful definitions and known properties on learning in the limit [16]. Let $\mathcal{G}$ be a class of grammars, that we wish to learn from positive examples. Formally, let $\mathcal{L}(G)$ denote the language associated with a grammar G, and let V be a given alphabet, a learning algorithm is a computable function ϕ from finite sets of words in V^* to $\mathcal{G}$, such that $\forall G \in \mathcal{G}$, $\forall (e_i)_{i\in\mathbb{N}}$ such that $\mathcal{L}(G) = \{e_i \mid i \in \mathbb{N}\}$ $\exists G' \in \mathcal{G}$ and $\exists n_0 \in \mathbb{N}$ such that $\forall n > n_0$ $\phi(e_1, \ldots, e_n) = G' \in \mathcal{G}$ and $\mathcal{L}(G') = \mathcal{L}(G)$.

Limit points. A class $\mathcal{CL}$ of languages has *a limit point* iff there exists an infinite sequence $(L_n)_{n\in\mathbb{N}}$ of languages in $\mathcal{CL}$ and a language $L \in \mathcal{CL}$ such that: $L_0 \subset L_1 \ldots L_i \subset L_{i+1} \subset \ldots$ and $L = \bigcup_{n\in\mathbb{N}} L_n$ (L is *a limit point* of $\mathcal{CL}$). If the languages of the grammars in a class $\mathcal{G}$ have a limit point then the class $\mathcal{G}$ is *unlearnable* in Gold's model.

Elasticity. A class $\mathcal{CL}$ of languages has *infinite elasticity* iff there exists $(e_i)_{i\in\mathbb{N}}$ a sequence of sentences and $(L_i)_{i\in\mathbb{N}}$ a sequence of languages in $\mathcal{CL}$ such that : $\forall i \in \mathbb{N}$: $e_i \notin L_i$ and $\{e_1, \ldots, e_i\} \subseteq L_{i+1}$. It has *finite elasticity* in the opposite case. If $\mathcal{CL}$ has *finite elasticity* then the corresponding class of grammars is learnable in Gold's model[6].

3.2. Non-learnability from strings – a review

The class of rigid (also k-valued for any k) PGs has been shown not learnable from strings in [14] using [15]. So, no learning algorithm is possible. This has also been shown for subclasses of rigid PGs as summarized below (from [3]).

Pregroups of order n and of order $n + 1/2$. A PG on $(P, \leq)$ is of order $n \in \mathbb{N}$ when its primitive categories are in $\{a^{(i)} | a \in P\ ,\ -n \leq i \leq n\}$; it is of order $n + 1/2, n \in \mathbb{N}$ when its primitive categories are in $\{a^{(i)} | a \in P\ ,\ -n-1 \leq i \leq n\}$.

[6] we also need to assume that (1) the class of grammars is recursively enumerable and (2) the universal membership problem for this class is decidable.

Construction of rigid limit points. We have proved [3] that the smallest such class (except order 0) has a limit point. Let $P = \{p, q, r, s\}$ and $\Sigma = \{a, b, c, d, e\}$. We consider grammars on $(P, =)$:

$G_n = (\Sigma, I_n, s)$	$G_* = (\Sigma, I_*, s)$
$a \mapsto (p^l)^n q^l$	$a \mapsto q^l$
$b \mapsto qpq^l$	$b \mapsto qp^l q^l$
$c \mapsto qr^l$	$c \mapsto qr^l$
$d \mapsto rp^l r^l$	$d \mapsto rpr^l$
$e \mapsto rp^n s$	$e \mapsto rs$

THEOREM 3.1. *The language of G_* is a limit point for the languages of grammars G_n on $(P, =)$ in the class of languages of rigid simple PGs of order 1/2 : for $n \geq 0$, $\mathcal{L}(G_n) = \{ab^k cd^k e \mid 0 \leq k \leq n\}$ and $\mathcal{L}(G_*) = \{ab^k cd^k e \mid k \geq 0\}$.*

For $P = \{p, q, r, s\}$, we write $P^l = \{p^l, q^l, r^l, s^l\}$ and we define $\mathcal{R}$, $\mathcal{R}^*$ by:
$\mathcal{R} : Xz^l zY \xrightarrow{\mathcal{R}} XY$, for $z \in P$ and $X, Y \in (P \cup P^l)^*$
$\mathcal{R}^*$: the reflexive and transitive closure of $\mathcal{R}$, written $\xrightarrow{\mathcal{R}^*}$.

LEMMA 3.2. *For $X \in (P \cup P^l)^*$ and $u \in P$, $X \leq u$ if and only if $X \xrightarrow{\mathcal{R}^*} u$*

PROOF. This lemma is shown by considering the (cut-free) system for simple free pregroups: the only possible rules on $(P \cup P^l)^*$ are the identity (Id) and (A_L) with specific exponents that correspond to $\mathcal{R}$ ∎

LEMMA 3.3. *For $X, Y \in (P \cup P^l)^*$ and $z, u \in P$, if $Xz^l uY \xrightarrow{\mathcal{R}} s$ then $z = u$*

PROOF. In a string on $(P \cup P^l)^*$, each x^l of P^l will disappear with an x of P on its right; symetrically, each y of P will disappear with an y^l of P^l on its left; hence, a pair of two consecutive z^l and u cannot disappear by application of rule $\xrightarrow{\mathcal{R}}$ unless $z = u$. ∎

We now prove theorem 3.1 (language descriptions).

PROOF. Lemmas 3.2 and 3.3 indicate that only members of ab^*cd^*e can belong to $\mathcal{L}(G_n)$ and similarly for $\mathcal{L}(G_*)$.

For $i \geq 0$ and $j \geq 0$, we consider $ab^i cd^j e \in ab^*cd^*e$:
-if $i = j$ and $i < n$, then $ab^i cd^j e \in ab^*cd^*e$ belongs to $\mathcal{L}(G_n)$, since:
$(p^l)^n q^l (qpq^l)^i qr^l (rp^l r^l)^j rp^n s \xrightarrow{\mathcal{R}*} s$
-if $i = j$ then $ab^i cd^j e \in ab^*cd^*e$ belongs to $\mathcal{L}(G_*)$, since:

$q^l(qp^lq^l)^i qr^l(rpr^l)^j rs \overset{\mathcal{R}*}{\to} s$
-if $ab^icd^je \in \mathcal{L}(G_n)$, we get, by replacing q and r by 1 :
$(p^l)^n(p)^i(p^l)^jp^ns \overset{\mathcal{R}*}{\to} s$ that necessitates $i = j$ and $i \leq n$
-if $ab^icd^je \in \mathcal{L}(G_*)$, we get, by replacing p, q and r by 1 :
$(p^l)^i(p)^js \overset{\mathcal{R}*}{\to} s$ that necessitates $i = j$ ∎

Remark. This construction can be simplified, taking $p = s$ (similar).

COROLLARY 3.4. *The classes* $\mathcal{CG}_n^k$ *of* k*-valued simple PGs of order* $n, n > 0$ *and* $\mathcal{CG}_{n+1/2}^k$ *of* k*-valued simple PGs of order* $n+1/2$, $n \geq 0$ *are not learnable from strings.*

3.3. Learnability for restricted categories

We consider three cases of restricted categories. Case (ii) is used in next section.

Width and order bounded categories. It is first to be noted that when we bind the width and the order (the length) of categories, as well as the number of categories (k-valued), the class is learnable from strings (since we have a finite number of grammars - up to renaming).

Width bounded categories. We will establish that when we bind the width of categories, as well as the number of categories (k-valued), the class is also learnable from strings (however this class of grammars is infinite since there is no bound on the exponents, for example $\pi_3^{(1)}s_1o^{(-1)}$ and $\pi_3^{(5)}s_1o^{(-1)}$ are both of width 3).

Category patterns. In what follows, we use known relationships between categorial formalisms, to obtain a case of learnability from strings for pregroup grammars. We refer to [5, 20] for definitions and details.

3.3.1. From Lambek calculus L to pregroup

We have a translation $A \to [A]$ on formulas and sequents from L to the simple free pregroup, that translates a valid sequent in a valid inequality :[7]

$$\begin{array}{lcl} [A] & = & A \quad \text{when } A \text{ is primitive} \\ [A \setminus B] & = & [A]^r[B] \\ [B \,/\, A] & = & [B][A]^l \\ [A_1, \ldots, A_n \vdash B] & = & [A_1]\cdots[A_n] \leq [B] \end{array}$$

[7] the converse is not true : $[(a \cdot b) \,/\, c] = abc^l = [a \cdot (b \,/\, c)]$ but $(a \cdot b) \,/\, c \not\vdash a \cdot (b \,/\, c)$ and $[(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p] = pp^{ll}p^{ll}p^lp^l \leq [p]$ but $(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p \not\vdash p$

$L_\emptyset$ will denote the variant of L allowing empty left-hand side in sequents.

Categorial order. In categorial grammars with operators $\backslash$, $/$ the order of a category $o(A)$ is :

$o(A)=0$ when A is primitive

$o(A \backslash B)=o(B / A)=max(o(A)+1, o(B))$

LEMMA 3.5. **[Buszkowski [6]]** *If B is primitive and $o(A_i)\leq 1$ for $1\leq i\leq n$:*

$A_1, \ldots, A_n \vdash_{AB} B$ *(AB denotes classical categorial grammars)*

iff $A_1, \ldots, A_n \vdash_{L_\emptyset B}$

iff $[A_1]\cdots[A_n] \leq B$ *valid in the simple free pregroup.*

In fact, this lemma leads to a learnabilty result:

COROLLARY 3.6. *The class $\mathcal{C}_L^k$ of k-valued PGs with categories of this form (hereafter called pattern P_1) : $g_n^r \ldots g_1^r p d_1^l \ldots d_m^l$, for $n \geq 0$ et $m \geq 0$ is learnable in Gold's model.*

PROOF. Lemma 3.5 shows that the class of pregroup grammars that are images of k-valued Lambek grammars of order 1 is learnable (in Gold's sense). And when $o(A) \leq 1$, then $[A]$ must be written as: $g_n^r \ldots g_1^r p d_1^l \ldots d_m^l$. ■

Use of patterns P_1. We have observed that many examples [18, 1, 9] follow pattern P_1 (as in example 4.2) or may be approximated by these in the following sense : by an increasing category function we mean h such that $x \leq h(x)$ (type-raise introduction $h_{<raise,y>}(x) = yy^l x$ as in $I(\text{“}every'') = \{ss^l \pi_3 n^l\}$ is an example) ; if $G = \langle \Sigma, I, s\rangle$ assigns $I(c_i) = \{h_i(t_i)\}$, where all h_i are increasing and all t_i have the pattern P_1, we get $\mathcal{L}(G) \subseteq \mathcal{L}(G_P)$ where $G_P = \langle \Sigma, I_P, s\rangle$ assigns $I_P(c_i) = \{t_i\}$ and all G_P belong to a class learnable from strings.

3.3.2. Width bounded categories

We define a normalization of grammars [4] which is very similar to the concept of centered dictionaries introduced in [24] and that leads to another learnable class of grammar:

THEOREM 3.7. *The class of rigid (or k-valued for each k) pregroup grammars with categories of width less than N is learnable from strings for each N.*

PROOF. Let G denote a rigid grammar on Σ, let r be the size of the alphabet Σ, and n be the width of G. We can show that G is equivalent (same language) to a similar grammar of order not greater than $2 \times n \times r$; this grammar can be seen as the normalized version of G obtained by *repeating shifts* (i) and (ii) as follows:

(i) consider possible iterations of r, if two consecutive integer exponents (u), $(u+1)$ with $u \geq 0$ never appear in the iterated adjoints $p_i^{(u_i)}$ in categories assigned by G, (this is a gap of length at least two in the succession of exponents occurring in categories assigned by G) ;
then decrement all above exponents in G: in each $C = p_1^{(u_1)} \dots p_n^{(u_m)}$ assigned by G replace each $p_i^{(u_i)}$ such that $u_i{>}u{+}1$ with $p_i^{(u_i-1)}$

(ii) proceed similarly for iterations of l: consider $(u-1)$, (u) with $u \leq 0$ corresponding to a gap and increment by 1 exponents lower than $(u-1)$.

Therefore, a width bound induces an order bound for rigid grammars ; we then apply the learnability result for a class with a width and an order bounds. In the k-valued case, we proceed similarly with an order $\leq 2 \times n \times r \times k$ ■

Note that a bound on the width does not give a bound on the categories: the class of grammars is still infinite since there is no bound on the exponents. In fact, the above proof shows how to transform any grammar to an equivalent one using (repeated) exponent shifts ; it also shows that each class of languages generated by k-valued grammars of width less than N is finite and thus has finite elasticity.

EXAMPLE 3.8. *Let* $P = \{\pi_3, o, s, s_1, n\}$ *with* $s_1 \leq s$, $n \leq o, \pi_3$, *let* $\Sigma = \{he, loves, cats\}$, *grammar* $G = (\Sigma, I, s)$ *with* $I(he) = \{\pi_3^{(4)}\}$, $I(cat) = \{n\}$, $I(loves) = \{\pi_3^{(5)} s_1 o^{(-1)}\}$ *will become* $I(he) = \{\pi_3^{(2)}\}$, $I(loves) = \{\pi_3^{(3)} s_1 o^{(-1)}\}$

4. Learning pregroup grammars from feature-tagged examples

Previous learnability results do not lead to any tractable algorithm. But an idea from Categorial Grammar learning is worth being applied to PGs: the learnability from typed examples. Types are to be understood here in the sense Montague's logic gave them.

In this context, semantic types are derived from syntactic categories by a homomorphism. Under some conditions concerning this homomorphism, interesting subclasses of AB-Categorial Grammars and of Lambek Grammars have been proved learnable from typed examples, i.e. from sentences

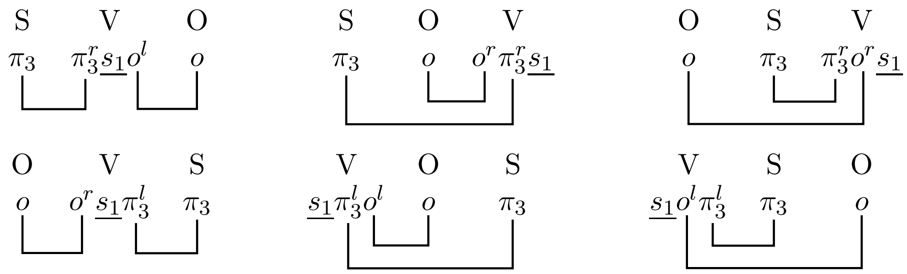

Figure 2. Pregroup Grammars and possible word orders

where each word is associated with its semantic type [13, 11]. In this case, learning algorithms implement a *specialization strategy.* We want to adapt this strategy to PGs.

4.1. Conditions of learning

The first problem met is that the link between PGs and semantics is not clearly stated, although the problem has been recently addressed in [23]. So, the notion of semantic types has no obvious relevance in our context and our first task is to identify what can play the role of language-independent features for PGs. We call feature-tagged examples the resulting input data. We then define a subclass of PGs learnable from feature-tagged examples in the sense of Gold.

4.1.1. Specification of input data

Let us consider how the six possible word orders for a basic sentence expressing a statement at the present tense, with a third person subject S, a transitive verb V and a direct object O would be treated by PGs (Figure 2). The common points between every possible analysis are the primitive categories associated with S and O. The category of V is always a concatenation (in various orders) of the elements of the set $\{s_1, \pi_3^u, o^v\}$ where u and v are either r or l: this set simply expresses that V expects a subject and an object. But the nature of the exponent (r or l or a combination of them) and their relative positions in the category associated with V are language-specific.

This comparison suggests that multisets of primitive categories play the role of language-independent features in PGs.

Definition 4.1 (Multisets $\mathcal{M}(P)$, mapping f_P, feature-tagged language).

- For any set $(P, \leq)$, we call $\mathcal{M}(P)$ the set of multisets of elements of P and f_P the mapping from $\mathcal{C}at_{(P,\leq)}$ to $\mathcal{M}(P)$ that transforms any category into the multiset of its primitive categories.
- For any PG $G = (\Sigma, I, s)$, the *feature-tagged language* of G, noted $FT(G)$, is defined by:

  $FT(G) = \{\langle v_1, T_1\rangle ... \langle v_n, T_n\rangle \mid \forall i \in \{1, ..., n\}, \exists X_i \in I(v_i)$ such that $X_1...X_n \leq s$ and $T_i = f_P(X_i)\}$.
- A pair $\langle a, M\rangle$ is called a *feature-tagged symbol* and an element of $FT(G)$ is called a *feature-tagged example*.

EXAMPLE 4.2. *Let $P = \{\pi_3, o, s, s_1, n\}$ with $s_1 \leq s$, $n \leq o, \pi_3$, let $\Sigma = \{he, loves, small, cats\}$ and let $G = (\Sigma, I, s)$ with $I(he) = \{\pi_3\}$, $I(loves) = \{\pi_3^r s_1 o^l\}$, $I(small) = \{nn^l\}$, $I(cat) = \{n\}$. Because $\pi_3 \pi_3^r s_1 o^l n n^l n \leq s$, $f_P(\pi_3) = \{\pi_3\}$, $f_P(\pi_3^r s_1 o^l) = \{s_1, \pi_3, o\}$, $f_P(nn^l) = \{n, n\}$ and $f_P(n) = \{n\}$, we have: $\langle he, \{\pi_3\}\rangle\langle loves, \{s_1, \pi_3, o\}\rangle\langle small, \{n, n\}\rangle\langle cats, \{n\}\rangle \in FT(G)$.*

We want to study how PGs can be learned from feature-tagged examples. For this, we need to have a condition on the mapping f_P that transforms a category into a multiset.

DEFINITION 4.3. For any sets Σ and P, we call $\mathcal{G}_f$ the set of PGs $G = (\Sigma, I, s)$ satisfying: $\forall v \in \Sigma$, $\forall X_1, X_2 \in I(v)$: $f_P(X_1) = f_P(X_2) \Longrightarrow X_1 = X_2$

The mapping f_P plays for PGs the role of the homomorphism transforming syntactic categories into semantic types for Categorial Grammars. So, the previous condition corresponds to the one used in the context of Categorial Grammars [13, 11]. It means that for each word in the vocabulary of a PG, each initial assignment of a category corresponds to a distinct multiset. This would for example forbid that, in our previous example, the verb "loves" be assigned $I(loves) = \{\pi_3^r s_1 o^l, o^r s_1 \pi_3^l\}$ because both categories give rise to the same multiset: $f_P(\pi_3^r s_1 o^l) = f_P(o^r s_1 \pi_3^l) = \{s_1, \pi_3, o\}$.

THEOREM 4.4. *The class $\mathcal{G}_f$ is learnable in Gold's model from feature-tagged examples (i.e. where, in Gold's model, FT plays the role of $\mathcal{L}$ and $V = \Sigma \times \mathcal{M}(P)$).*

PROOF. The theorem is a Corollary of Theorem 3.7 and the condition expressed in definition 4.3 that entails Fact (i) below:

- (i) For a grammar G in $\mathcal{G}_f$, for any v in Σ, the number of distinct multisets $f_p(X_i)$ for the set of X_i assigned to v is the number of distinct X_i assigned to v.

- We consider an algorithm $\phi_{<n,k>}$, that learns k-valued pregroup grammars with categories of width at most n (from Theorem 3.7).

- We define the following algorithm ϕ_f, that takes a finite list E_n of feature-tagged examples $e_i = \langle v_i, T_i \rangle$, (for $i < n$) and returns a pregroup grammar on Σ (or fails):

  (1) compute w_n = the maximum length of multisets T_i in E_n ;

  (2) let $k_{<i,n>}$ denote the number of distinct multisets T_j associated to the same v_i in E_n ; compute k_n = the maximum value of $k_{<i,n>}$, for $i < n$ (clearly if $n' > n$, then $k_{n'} \geq k_n$) ;

  (3) apply $\phi_{<w_n,k_n>}$ on E_n.

- For every G in $\mathcal{G}_f$, for every enumeration $e_i = \langle v_i, T_i \rangle$ of $L = FT(G)$ there exists some N, such that T_N (with $e_N = \langle v_N, T_N \rangle$) has the maximum length of multisets in L. For $n > N$, we thus apply $\phi_{<N,k_n>}$.

  Let k_* denote the maximum number of categories assigned to a symbol by G.

  There exists also some $N' > N$, such that $T_{N'}$ has the maximum value of k_n, written k, with $k \leq k_*$ from fact (i) ($k = k_*$ if there is no useless category[8]).

  For $n > N' > N$, we thus apply $\phi_{<N,k>}$ that converges to a grammar in G_f, having the same feature language as G.

Notice that this version does not require the knowledge of P. ∎

4.2. Learning algorithm

We now present an algorithm whose purpose is to identify every possible PG of the class $\mathcal{G}_f$ compatible with a set of feature-tagged examples. More precisely, our algorithm will provide a set of constraints on integer variables specifying a set of PGs, but the constraints resolution mechanism is not described here. This strategy allows to reconsider the learning problem as a two-steps process: the first step, detailed here, builds a logical formula where the propositions are constraints on a set of integer variables, the second one solves the constraints. This way, we hope to delay as much as possible the inevitable combinatorial explosion. We prove that the output specified by our algorithm is exactly, up to basic transformations, the set of every PG

[8]a category of G that is not necessary for the enumeration of L.

compatible with the input. Our algorithm itself has two main steps: first variables are introduced, then constraints are deduced on their values.

4.2.1. Variable introduction (first step of the algorithm)

Although feature-tagged examples provide a lot of information, two things remain to be learned: the value of the exponents of categories and their relative positions inside a concatenation. We introduce variables to code both problems. We consider two (infinite) sets of variables, $\mathcal{X}$ a set of position variables and $\mathcal{U}$ a set of exponent variables. Each variable is identified by (or associated to) a nuplet $\langle\langle a, M\rangle, p, k\rangle$ where $a \in \Sigma$, $M \in \mathcal{M}(P)$, $p \in M$ and $k \geq 1$[9].

Definition 4.5. Let $\langle a, M\rangle$ be a feature-tagged symbol ($a \in \Sigma$ and $M \in \mathcal{M}(P)$). For each occurrence of a primitive category p in M, we create a *position variable* $x \in \mathcal{X}$ and an *exponent variable* $u \in \mathcal{U}$ that are associated to the nuplet $(\langle a, M\rangle, p, k)$. k is the position of this occurrence of p among all the p in M. There is only one position variable in $\mathcal{X}$ and only one exponent variable in $\mathcal{U}$ for each nuplet $\langle\langle a, M\rangle, p, k\rangle$. For each $\langle a, M\rangle$ in a sentence, we associate the set $T = \{(x_1, u_1), \ldots, (x_l, u_l)\}$ of all the pairs of position and exponent variables corresponding to $\langle a, M\rangle$.

Example 4.6. *The feature-tagged example of Example 4.2 defines 7 pairs of variables distributed in 4 sets corresponding to the 4 feature-tagged symbols:*

$$\begin{array}{ll}
\langle he, \{\pi_3\}\rangle: & T_1 = \{(x_1, u_1)\} \\
\langle loves, \{s_1, \pi_3, o\}\rangle: & T_2 = \{(x_2, u_2), (x_3, u_3), (x_4, u_4)\} \\
\langle small, \{n, n\}\rangle: & T_3 = \{(x_5, u_5), (x_6, u_6)\} \\
\langle cats, \{n\}\rangle: & T_4 = \{(x_7, u_7)\}
\end{array}$$

The nuplet corresponding to each couple of variables is as follows:

$$\begin{array}{llll}
x_1, u_1: & (\langle he, \{\pi_3\}\rangle, \pi_3, 1) & x_2, u_2: & (\langle loves, \{s_1, \pi_3, o\}\rangle, s_1, 1) \\
x_3, u_3: & (\langle loves, \{s_1, \pi_3, o\}\rangle, \pi_3, 1) & x_4, u_4: & (\langle loves, \{s_1, \pi_3, o\}\rangle, o, 1) \\
x_5, u_5: & (\langle small, \{n, n\}\rangle, n, 1) & x_6, u_6: & (\langle small, \{n, n\}\rangle, n, 2) \\
x_7, u_7: & (\langle cats, \{n\}\rangle, n, 1) & &
\end{array}$$

For instance, x_6 and u_6 are associated to $(\langle small, \{n, n\}\rangle, n, 2)$: they correspond to the second occurrence (the right one) of an n in the category associated to 'small' with multiset elements $\{n, n\}$. x_3 and u_3 are associated to $(\langle loves, \{s_1, \pi_3, o\}\rangle, \pi_3, 1)$: they correspond to the first (and the only one)

[9] $\mathcal{X}$ and $\mathcal{U}$ may be seen as a four-dimensional array

occurrence of π_3 *in the category associated to 'loves' with multiset elements* $\{s_1, \pi_3, o\}$.

This coding allows to reformulate the acquisition problem into a *variable assignment problem.* Furthermore, as the feature-tagged examples provided as input are supposed to belong to the same $FT(G)$ for some G in $\mathcal{G}_f$, the *same variables* are used for every occurrence of the same feature-tagged symbol present in the set of feature-tagged examples.

4.2.2. Overview on constraints deductions and definitions

This step consists in deducing constraints applying on the variables. Each feature-tagged example is treated one after the other. For each one, the sequence of tags takes the form of a sequence of multisets. The deduction of constraints is performed by using rules that mimic rules **E**, **I** and **M** used for parsing PGs in section 2.2, with the addition of an initial rule **0** and a final rule **Z**. Constraints coming from the same syntactic analysis are linked by a conjunction, constraints from distinct alternative syntactic analyses are linked by a disjunction. For each sentence, after normalization, we thus obtain a disjunction of conjunctions of basic constraints (that we call *data constraint*) as follows:

Definition 4.7 (Basic Constraints). We consider several kinds of *basic constraints* on two sets $\mathcal{X}, \mathcal{U}$ of integer variables ranging respectively over $\mathbb{N} - \{0\}$ and $\mathbb{Z}$:

- *Position constraints* are of the form (strictly positive integer variables):

  $x < x' \qquad (x, x' \in \mathcal{X})$

- *Exponent constraints* are of four forms (positive or negative integer variables):

  - $u = u' + 1 \qquad (u, u' \in \mathcal{U})$
  - $odd(u) \qquad (u \in \mathcal{U})$
  - $\neg odd(u) \qquad (u \in \mathcal{U})$
  - $u = 0 \qquad (u \in \mathcal{U})$

- *Domain constraints* are of the form: $\{x_1, x_2, \ldots, x_n\} = \{1, 2, \ldots, n\}$

  corresponding to a multiset T_j of size n, with $x_i \in \mathcal{X}$, $u_i \in \mathcal{U}$ (for $1 \le i \le n$): $T_j = \{(x_1, u_1), (x_2, u_2), \ldots, (x_n, u_n)\}$

The set $\mathbb{B}C$ will denote *the set of basic constraints*.

DEFINITION 4.8 (Normal Forms of Constraints). *The set of data constraints* $\mathbb{D}C$ generated by *a set of basic constraints* $\mathbb{B}C$ is the set of disjunctions ($\vee$) of conjunctions ($\wedge$) of basic constraints in $\mathbb{B}C$ without repetition of basic constraints in each conjunction, and without repetition of conjunctions of basic constraints in disjunctions [10].

Finiteness. Notice that when the set $\mathcal{X}$ of position variables and the set $\mathcal{U}$ of variables are finite, the set of basic constraints $\mathbb{B}C$ expressible on them is then finite, and the set $\mathbb{D}C$ (relative to these sets) is also finite.

4.2.3. Constraints deductions for a feature-tagged example

A constraint deduction takes the form of rules that mimic rules **E**, **I** and **M** used for parsing PGs in section 2.2 with the addition of an initial rule **0** and a final rule **Z**; we now detail these rules:

0 (initialization): Let n be the length of the sentence. $\forall m, 1 \leq m \leq n$, let $T_m = \{(x_i^m, u_i^m)_{1\leq i\leq l^m}\}$ be the set of pairs of position and exponent variables corresponding to the mth feature-tagged symbol of the sentence, each pair (x_i^m, u_i^m) being associated to $(\langle a^m, M^m\rangle, p_i^m, k_i^m)$.

- Domain constraints:
  · $\forall m :\ \{x_1^m, \ldots, x_{l^m}^m\} = \{1, \ldots, l^m\}$
  · $\forall m, i :\ u_i^m \in \mathbb{Z}$
- Position constraints[11]:
  · $\forall m, i, j$: IF $p_i^m = p_j^m$ and $k_j^m = k_i^m + 1$ THEN $x_i^m < x_j^m$
- Initial list of sets: $T_1, \ldots, T_n$

E (external): Let $T_m = \{(x_i, u_i)_{1\leq i\leq l}\}$ and $T_{m'} = \{(x'_j, u'_j)_{1\leq j\leq l'}\}$ be two consecutive sets (at the beginning: $m' = m + 1$) with (x_i, u_i) associated to $(\langle a_i, M_i\rangle, p_i, k_i)$ and (x'_j, u'_j) associated to $(\langle a'_j, M'_j\rangle, p'_j, k'_j)$. If $\exists i_0, j_0$ such that:

$$\begin{cases} p_{i_0} \leq p'_{j_0} \vee p'_{j_0} \leq p_{i_0} \\ k_{i_0} = \max\{k_i \mid 1 \leq i \leq l \wedge a_i = a_{i_0} \wedge M_i = M_{i_0} \wedge p_i = p_{i_0}\} \\ k'_{j_0} = \min\{k'_j \mid 1 \leq j \leq l' \wedge a'_j = a'_{j_0} \wedge M'_j = M'_{j_0} \wedge p'_j = p'_{j_0}\} \end{cases}$$

[10] we consider these constraints modulo the associativity-commutativity of $\vee$ and $\wedge$

[11] the occurrences of the same primitive category for a word must be ordered from left to right in the resulting category.

- Position constraints:
  · $\forall i \neq i_0, 1 \leq i \leq l$: $x_i < x_{i_0}$
  · $\forall j \neq j_0, 1 \leq j \leq l'$: $x'_{j_0} < x'_j$
- Exponent constraints:
  · (all cases) $u'_{j_0} = u_{i_0} + 1$
  · IF $p'_{j_0} < p_{i_0}$ THEN: $odd(u_{i_0})$
  · IF $p_{i_0} < p'_{j_0}$ THEN: $\neg odd(u_{i_0})$
- Next sets:
  · $T_m \longleftarrow T_m - \{(x_{i_0}, u_{i_0})\}$
  · $T_{m'} \longleftarrow T_{m'} - \{(x'_{j_0}, u'_{j_0})\}$

I **(internal)**: For internal reductions, where $m = m'$, the maximal condition on k_{i_0} and the minimal condition on k'_{j_0} is replaced by an adjacent condition but only when $a_{i_0} = a_{j_0} \wedge M_{i_0} = M_{j_0} \wedge p_{i_0} = p_{j_0}$:

$$\begin{cases} k_{i_0} < k_{j_0} \\ \forall i, a_i \neq a_{i_0} \vee M_i \neq M_{i_0} \vee p_i \neq p_{i_0} \vee k_i \leq k_{i_0} \vee k_{j_0} \leq k_i \end{cases}$$

- The position constraints are replaced by:
  · $x_{i_0} < x_{j_0}$
  · $\forall i \neq i_0, i \neq j_0$ $(x_i < x_{i_0}) \vee (x_{j_0} < x_i)$

M **(merge)**: Let $T_m = \{(x_i, u_i)_{1 \leq i \leq l}\}$ and $T_{m'} = \{(x'_j, u'_j)_{1 \leq j \leq l'}\}$ be two consecutive sets (in particular if one is empty).

- Next sets:
  · $T_m \longleftarrow T_m \cup T_{m'}$
  · Drop $T_{m'}$

Z **(final)**: The process ends when the list gets reduced to some $\{(x, u)\}$ associated to $(\langle a, M\rangle, p, k)$ where $p \leq s$ (the constraint $u = 0$ is deduced).

EXAMPLE 4.9. *Let us see what this algorithm gives on our basic Example 4.6 where the initial sequence of sets is:* T_1, T_2, T_3, T_4*:*

1. *Initial constraints: initial domain constraints and, for the two occurrences of* n *of* T_3*,* $x_5 < x_6$
2. *External reduction between* T_1 *and* T_2*:* $(x_1, u_1) \in T_1$ *and* $(x_3, u_3) \in T_2$ *satisfy the precondition. The position constraints obtained are:* $x_3 < x_2$ *and* $x_3 < x_4$*. The exponent constraint is:* $u_3 = u_1 + 1$*, and the*

remaining sets are the following: $T_1 = \emptyset$, $T_2 = \{(x_2, u_2), (x_4, u_4)\}$, $T_3 = \{(x_5, u_5), (x_6, u_6)\}$, $T_4 = \{(x_7, u_7)\}$

3. *Merge between* T_1 *and* T_2*: no new constraint.* T_2 *is dropped.*

4. *External reduction between* T_1 *and* T_3*:* $(x_4, u_4) \in T_1$ *and* $(x_5, u_5) \in T_3$ *satisfy the precondition. We deduce:* $x_2 < x_4$, $x_5 < x_6$ *(this constraint was already produced as an initial position constraint),* $u_5 = u_4 + 1$, u_4 *is odd and the remaining sets are:* $T_1 = \{(x_2, u_2)\}$, $T_3 = \{(x_6, u_6)\}$, $T_4 = \{(x_7, u_7)\}$.

5. *External reduction between* T_3 *and* T_4*:* $(x_6, u_6) \in T_3$ *and* $(x_7, u_7) \in T_4$ *satisfy the precondition. We deduce:* $u_7 = u_6 + 1$ *and the remaining sets are:* $T_1 = \{(x_2, u_2)\}$, $T_3 = \emptyset$, $T_4 = \emptyset$.

6. *Merges (twice) between* T_1, T_3 *and* T_4*: no new constraint.* T_3 *and* T_4 *are dropped.*

7. *As* $s_1 \leq s$*, we obtain* $u_2 = 0$ *and the algorithm stops.*

With initial domain constraints $\{x_1\} = \{x_7\} = \{1\}$, $\{x_2, x_3, x_4\} = \{1, 2, 3\}$ *and* $\{x_5, x_6\} = \{1, 2\}$*, from constraints* $x_3 < x_2$, $x_3 < x_2$, $x_2 < x_4$ *and* $x_5 < x_6$*, we easily deduce:* $x_2 = 2$, $x_3 = 1$, $x_4 = 3$, $x_5 = 1$ *and* $x_6 = 2$*. The PGs specified by these constraints are defined up to a shift on the exponents. The solution specified is thus an* equivalent class *of PGs. If we set* $u_1 = u_5 = u_7 = 0$ *(*u_4 *must be odd), then* $u_3 = 1$ *(or* $u_3 = r$*) and* $u_4 = u_6 = -1$ *(or* $u_4 = u_6 = l$*): the only remaining PG associates* $\pi_3^r s_1 o^l$ *with "loves" and* nn^l *with "small". But, in general, solution PGs are only implicitly specified by the set of constraints and a constraints resolution mechanism must be applied. We will see that the set of constraints can be exponentially smaller than the set of classes (up to shifts) of PGs it specifies.*

There is another possibility for this example that gives a different set of constraints:

1. *Initial constraints: initial domain constraints and, for the two occurrences of* n *in* T_3, $x_5 < x_6$

2. *Internal reduction inside* $T3$*:* $(x_5, u_5) \in T_3$ *and* $(x_6, u_6) \in T_3$ *satisfy the precondition (in this order). We deduce:* $x_5 < x_6$ *(this constraint was already produced as an initial position constraint) and* $u_6 = u_5 + 1$*, and the remaining sets are the following:* $T_1 = \{(x_1, u_1)\}$, $T_2 = \{(x_2, u_2), (x_3, u_3), (x_4, u_4)\}$, $T_3 = \emptyset$, $T_4 = \{(x_7, u_7)\}$.

3. *Merge between T_2 and T_3: no new constraint. T_3 is dropped.*

4. *External reduction between T_1 and T_2: $(x_1, u_1) \in T_1$ and $(x_3, u_3) \in T_2$ satisfy the precondition. The position constraints obtained are: $x_3 < x_2$ and $x_3 < x_4$. The exponent constraint is: $u_3 = u_1 + 1$, and the remaining sets are the following: $T_1 = \emptyset$, $T_2 = \{(x_2, u_2), (x_4, u4)\}$, $T_4 = \{(x_7, u_7)\}$*

5. *Merge between T_1 and T_2: no new constraint. T_2 is dropped.*

6. *External reduction between T_1 and T_4: $(x_4, u_4) \in T_1$ and $(x_7, u_7) \in T_4$ satisfy the precondition. We deduce: $u_7 = u_4 + 1$, u_4 is odd and the remaining sets are: $T_1 = \{(x_2, u_2)\}$, $T_4 = \emptyset$.*

7. *Merge between T_1 and T_4: no new constraint. T_4 is dropped.*

8. *As $s_1 \leq s$, we obtain $u_4 = 0$ and the algorithm stops.*

With the same idea as the previous resolution, we find nearly the same grammar except that the category of "small" is now nn^r. In fact, this grammar is not equivalent to the previous one: the language corresponding to the previous resolution is strictly included in the language for this resolution.

4.2.4. Global Acquisition Algorithm

Algorithm 1 which searches for the elements of $\mathcal{G}_f$ compatible with a set of feature-tagged examples $\langle e_1, ..., e_n \rangle$

Require: $\langle e_1, ..., e_n \rangle$ where $\forall i$, $e_i \in (\Sigma \times \mathcal{M}(P))^+$ is a feature-tagged example for some PG

introduce variables on every e_i
$\mathcal{SC}_0 = \emptyset$
for $(i = 1, i \leq n, i++)$ **do**
  treat the "variabilized" feature-tagged example e_i to obtain the constraint $\mathcal{C}_i$
  $DC_i =$ normal form of $\mathcal{C}_i$
  $\mathcal{SC}_i = \mathcal{SC}_{i-1} \cup DC_i$
end for

Ensure: $\mathcal{SC}_n$: a set of constraints specifying a set of PGs

The global algorithm we propose is Algorithm 1. For each sentence, following all the possible applications of rules **0**, **E**, **I**, **M** and **Z**, we obtain a complex logical formula of elementary exponent and position constraints (that we call *simple constraints*). This formula, the *data constraint* for this sentence, reflects its possible syntactic analyses. To formulate it, we can use the same schema as the parser presented in [2] : rather than using rules **E**, **I** and **M**, we can use rule **I** and a combinaison of **E** and **M** that mimics the functional-composition rule in [2]. Thus, if we set a maximal width for feature-tagged symbols (the maximal cardinality of T_i) and if the lexicon and the set of primitive categories are finite, we have a polynomial algorithm (w.r.t. the size of the feature-tagged example) for computing the data constraint of the sentence.

EXAMPLE 4.10. *Let us see what is the data constraint for our basic example. In fact, the logical formula corresponding to the different parsings is quite complex because the rules may be applied in several orders. But, all of them correspond to one of the two analyses that were presented. Thus, the resulting data constraint after normalization is (we omit initial domain constraints)*[12]*:*

$$\begin{array}{ll} & \left(\begin{array}{l}(x_3 < x_2) \wedge (x_3 < x_4) \wedge (x_2 < x_4) \wedge (x_5 < x_6) \wedge \\ (u_3 = u_1 + 1) \wedge (u_5 = u_4 + 1) \wedge (u_7 = u_6 + 1) \wedge (u_2 = 0) \wedge odd(u_4)\end{array}\right) \\ \vee & \left(\begin{array}{l}(x_3 < x_2) \wedge (x_3 < x_4) \wedge (x_2 < x_4) \wedge (x_5 < x_6) \wedge \\ (u_3 = u_1 + 1) \wedge (u_7 = u_4 + 1) \wedge (u_6 = u_5 + 1) \wedge (u_2 = 0) \wedge odd(u_4)\end{array}\right) \end{array}$$

5. Properties of the acquisition algorithm and data constraints

We formalize the links between grammars and data constraints as already sketched in previous examples

DEFINITION 5.1 (Mapping f_G and substitution σ_G). Given a grammar $G \in \mathcal{G}_f$, let us call f_G the mapping from the set of pairs on $\Sigma \times \mathcal{M}(P)$ to the set of categories $\mathcal{C}at_P$ defined by:

$$f_G(a, M) = T \text{ iff } G \text{ assigns } T \text{ to } a \text{ and } f_P(T) = M$$

This is indeed a mapping from the definition of $\mathcal{G}_f$. We then define a sub-

[12]the data constraint can also be written as follows:

$$\begin{array}{l}(x_3 < x_2) \wedge (x_3 < x_4) \wedge (x_2 < x_4) \wedge (x_5 < x_6) \wedge \\ (u_3 = u_1 + 1) \wedge (u_2 = 0) \wedge odd(u_4) \wedge \\ ((u_5 = u_4 + 1) \wedge (u_7 = u_6 + 1) \vee (u_7 = u_4 + 1) \wedge (u_6 = u_5 + 1))\end{array}$$

stitution σ_G on the variables $x \in \mathcal{X}$, $u \in \mathcal{U}$, onto $\mathbb{Z}$ as follows:

$$x \mapsto \begin{cases} \text{the position of the } k^{th} \text{ occurrence of the} \\ \text{primitive category } p \text{ in the category } f_G(a, M) \end{cases}$$
$$u \mapsto \begin{cases} \text{the exponent of the } k^{th} \text{ occurrence of the} \\ \text{primitive category } p \text{ in the category } f_G(a, M) \end{cases}$$

such that $(\langle a, M\rangle, p, k)$ is the nuplet associated to (x, u).

Definition 5.2 (Satisfaction of a set of constraints). Let $G \in \mathcal{G}_f$ denote a PG and let $\mathcal{SC} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ denote a set of constraints in $\mathbb{D}C$ defined on a set $\mathcal{X}$ of position variables and a set $\mathcal{U}$ of exponent variables.

G is said to satisfy $\mathcal{SC}$ whenever the substitution σ_G applied to $\mathcal{SC}$ is defined for all variables occurring in $\mathcal{SC}$ and yields only true statements on integers for each constraint $\mathcal{C}_i$ of the set of constraints $\mathcal{SC}$.

Example 5.3. *Grammar G given in Example 4.2 satisfies the data constraint (a singleton) described in Example 4.10. The similar grammar G' that assigns nn^r instead of nn^l to "small" also satisfies this constraint.*

5.1. Convergence property of the acquisition algorithm

In an acquisition process, each example gives rise to a new data constraints that is conjoined to the previous ones. We get a convergence property as follows:

Property 5.4. *Let $G \in \mathcal{G}_f$ and $FT(G) = \{e_i\}_{i \in \mathbb{N}}$, the sets of data constraints $\mathcal{DC}_i$ obtained from the successive sets $\{e_1, \ldots, e_i\}$ converges: $\exists n_0 \in \mathbb{N}$ such that $\forall n \geq n_0, \mathcal{SC}(\langle e_1, \ldots, e_{n+1}\rangle) = \mathcal{SC}(\langle e_1, \ldots, e_n\rangle)$. Moreover, the limit does not depend on the order of the enumeration of $FT(G)$. We write $\mathcal{SC}_*(FT(G))$ this limit.*

Proof. In fact $\mathcal{SC}(\langle e_1, \ldots, e_n\rangle) = \{DC_i \mid 1 \leq i \leq n\}$ where DC_i is the normal form of the constraints deduced from e_i. Because G has a finite lexicon, there exists a number N such that $\forall i \geq N$, each feature-tagged symbol of e_i already appears in at least one of the examples $\{e_1, \ldots, e_N\}$: for $\mathcal{SC}(\langle e_1, \ldots, e_n\rangle)$, only examples $e_1, \ldots, e_N$ introduce new position and exponent variables and the set of such variables is finite. Thus, the set of basic constraints and (normalized) data constraints is also finite. Since $\mathcal{SC}(\langle e_1, \ldots, e_n\rangle) = \mathcal{SC}(\langle e_1, \ldots, e_{n-1}\rangle) \cup \{DC_n\}$, Algorithm 1 must converge to a subset of the finite set of data constraints on position and exponent variables corresponding to the N first strings $e_1, \ldots, e_N$. Moreover, because $\mathcal{SC}(\langle e_1, \ldots, e_n\rangle) = \{DC_i \mid 1 \leq i \leq n\}$, the limit does not depend on the order of the enumeration of $FT(G)$. ∎

5.2. Correctness property of the acquisition algorithm

Let G be a grammar in the class $\mathcal{G}_f$, we show that G satisfies the data constraints computed from its feature-tagged language :

LEMMA 5.5. $\forall G \in \mathcal{G}_f$ *: G satisfies $\mathcal{SC}_*(FT(G))$*

PROOF. Let $DC \in \mathcal{SC}_*(FT(G))$. $\exists e = \langle\langle a_1, M_1\rangle \cdots \langle a_n, M_n\rangle\rangle$, a feature-tagged example of $FT(G)$ that generates DC in Algorithm 1. All the position and exponent variables that appear in DC correspond to a feature-tagged symbol of e. Because $e \in FT(G)$, there exists a parsing of e and each symbol of e must be associated to at least one category by G. Because $G \in \mathcal{G}_f$, there is at most one category corresponding to a feature-tagged symbol. As a consequence, σ_G is defined for all these position and exponent variables. Secondly, since $e \in FT(G)$, there exists a parsing of e using the parsing rules **E**, **I** and **M** of Section 2.2. By induction on this parsing, we can build a derivation using constraint deduction rules **0**, **E**, **I**, **M** and **Z** of Section 4.2.3. The list of categories that is produced at each step of the parsing is replaced by a list of sets of pairs of position and exponent variables, each pair corresponding to one occurrence of an atomic category of the list of categories. At each step, the conditions of application of the corresponding rule are verified and the substitution of generated constraints by σ_G is satisfied:

- Beginning of the parsing : we produce a **0** step. There is no condition of application of rule **0**. The domain constraints are obviously satisfied. The position constraints are also satisfied because in the definition of σ_G, the different position variables corresponding to the same primitive category are ordered from left to right. The list of sets of pairs of variables corresponds to the initial list of categories in the parsing.
- External parsing rule **E** : we produce a **E** step. We have the external parsing rule $\Gamma, Xp^{(n)}, q^{(n+1)}Y, \Delta \vdash_E \Gamma, X, Y, \Delta$. $q \leq p$ and n is even or $p \leq q$ and n is odd. By induction hypothesis, the list of categories $\Gamma, Xp^{(n)}, q^{(n+1)}Y, \Delta$ corresponds to a list of sets of pairs of variables $T_\Gamma, T_X \cup \{(x, u)\}, T_Y \cup \{(x', u')\}, T_\Delta$ where (x, u) corresponds to $p^{(n)}$ and (x', u') to $q^{(n+1)}$. $T_X \cup \{p^{(n)}\}$ and $T_Y \cup \{q^{(n+1)}\}$ are consecutive in the list. The pair of variables (x, u) is associated to $\langle a, M, p, k\rangle$ and the pair of variables (x', u') is associated to $\langle a', M', q, k'\rangle$. We have $p \leq q$ or $p \leq q$. Because in $Xp^{(n)}$, $p^{(n)}$ appears on the right, $\sigma_G(x)$ is maximal in $\{\sigma_G(x_i) \mid (x_i, u_i) \in T_{Xp^{(n)}}\}$. Similarly, $\sigma_G(x')$ is minimal. Thus the position constraints are satisfied by σ_G. Here, $\sigma_G(u) = n$ and $\sigma_G(u') = n + 1$ and the exponent constraints are also satisfied. Finally,

the next list of sets $T_\Gamma, T_X, T_Y, T_\Delta$ corresponds to the list of categories Γ, X, Y, Δ.

- Internal parsing rule **I** : we produce a **I** step. The reasonning is very similar to the previous case except that there is only one set of pairs. In fact, by induction hypothesis, each pair of variables corresponds to one occurrence of the categories in the parsing list of categories: this set of pairs can be split in four parts T_X, $\{(x, u)\}$, $\{(x', u')\}$, T_Y that correspond respectively to X, $p^{(n)}$, $q^{(n+1)}$ and Y. The rest of the proof is straightforward.
- Merge parsing rule **M** : we produce a **M** step. there is no particular difficulty here because no constraint is produced.
- Ending of the parsing : we produce a **Z** step. This step corresponds to the end of the parsing: the list of categories is reduced to an atomic category p such that $p \leq s$ where s is the primitive category to which G assigns to the strings of $\mathcal{L}(G)$. It corresponds to a unique pair of variables (x, u) associated to $\langle\langle a, M\rangle, p, k\rangle$. Here, we have $\sigma_G(u) = 0$.

Thus there exists a branch of DC that is satisfied by σ_G and G satisfies DC (the different branches of DC, at any level, are merged using a logical "or"). ■

Let G be a grammar in the class $\mathcal{G}_f$, we show that if G satisfies the data constraints computed from the feature-tagged language of a grammar G' in the class, then G generates these feature-tagged examples:

LEMMA 5.6. $\forall G, G' \in \mathcal{G}_f$: *if G satisfies $\mathcal{SC}_*(FT(G'))$ then $FT(G') \subseteq FT(G)$*

PROOF. We suppose that G satisfies $\mathcal{SC}_*(FT(G'))$. Let $e \in FT(G')$. Let DC be the data constraint generated by this string e in Algorithm 1. Because G satisfies $\mathcal{SC}_*(FT(G'))$, σ_G is defined for every variable that appears in DC. Thus, G assigns a (unique) category to each featured-tagged symbol of e. $\sigma_G(DC)$ is true and there exists a branch of DC where σ_G is satisfied. Because DC is generated by searching all the possible parsings of e using rules **0**, **E**, **I**, **M** and **Z** of Section 4.2.3, this branch corresponds to a derivation using these rules. Using this derivation, we construct a parsing of e in G using rules **E**, **I** and **M** of Section 2.2. At each step, we find a list of categories corresponding to an ordering of the list of sets of pairs of position and exponent variables given by the constraint deduction derivation:

- Rule **0** : The initial list of sets of pairs of variables corresponds to the list of feature-tagged symbol of e. G assigns a unique category to each feature-tagged symbol of e. The list of category is then well defined.

- Rule **E** : The parsing continues with a **E** step. For the constraint deduction, we have the list of sets of pairs $T_1, \ldots, T_m, T_{m'}, \ldots, T_n$ where $T_m = \{(x_i, u_i)_{1\leq i\leq l}\}$, (x_i, u_i) associated to $(\langle a_i, M_i\rangle, p_i, k_i)$ and $T_{m'} = \{(x'_j, u'_j)_{1\leq j\leq l'}\}$, (x'_j, u'_j) associated to $(\langle a'_j, M'_j\rangle, p'_j, k'_j)$. We know that $\exists i_0, j_0$ such that:

$$\begin{cases} p_{i_0} \leq p'_{j_0} \vee p'_{j_0} \leq p_{i_0} \\ k_{i_0} = \max\{k_i \mid 1 \leq i \leq l \wedge a_i = a_{i_0} \wedge M_i = M_{i_0} \wedge p_i = p_{i_0}\} \\ k'_{j_0} = \min\{k'_j \mid 1 \leq j \leq l' \wedge a'_j = a'_{j_0} \wedge M'_j = M'_{j_0} \wedge p'_j = p'_{j_0}\} \end{cases}$$

  By induction, $T_1 \ldots, T_m, T_{m'}, \ldots, T_n$ corresponds to a list of categories $X_1, \ldots, X_m, X_{m'}, \ldots, X_n$ for the parsing. G satisfies all the basic constraints of this case. Then $\forall i \neq i_0, 1 \leq i \leq l$: $\sigma_G(x_i) < \sigma_G(x_{i_0})$: in X_m, the position of the rightest p_{i_0} is on the right: $X_m = Y p_{i_0}^{\sigma_G(u_{i_0}))}$ for some Y. Similarly, $X_{m'} = {p'}_{j_0}^{\sigma_G(u'_{j_0}))} Y'$ for some Y'. Using exponent constraints, we also find that the conditions of application of parsing rule **E** are verified and ends with the list of categories $X_1, \ldots, Y, Y', \ldots, X_n$ that corresponds to the list of sets of pairs
  $T_1, \ldots, \{(x_i, u_i), 1 \leq i \leq l, i \neq i_0\}, \{(x'_j, u'_j), 1 \leq j \leq l', j \neq j_0\}, \ldots, T_n$.
- Rule **I** : quite similar to the previous case except that we must use the initial position constraints (rule **0**) that order the variables corresponding to the same primitive type from left to right in the corresponding category.
- Rule **M** : nothing very difficult here.
- Rule **Z** : the lists of sets of pairs of variables is reduced to a single set $\{(x, u)\}$. (x, u) is associated to $(\langle a, M\rangle, p, k)$ and $p \leq s$. We also have $\sigma_G(u) = 0$.

Thus we have a parsing of e in G and $e \in FT(G)$. ■

Let G, G' be grammars in the class $\mathcal{G}_f$, we show that if the feature-tagged language of G is a subset of the feature-tagged language of G', then data constraints computed from their respective examples are included in one another similarly :

Lemma 5.7 (Monotonicity). *$\forall G, G' \in \mathcal{G}_f$: if $FT(G) \subseteq FT(G')$ then $\mathcal{SC}_*(FT(G)) \subseteq \mathcal{SC}_*(FT(G'))$*

Proof. Esay by construction because each feature-tagged example generates a data constraint added to the set of constraints in Algorithm 1. ■

Let G, G' be grammars in the class $\mathcal{G}_f$, we conclude as stated in the next theorem that G is compatible with the feature-tagged examples generated by G' iff G satisfies the data constraints computed from the feature tagged examples of G' :

THEOREM 5.8 (Correctness). $\forall G, G' \in \mathcal{G}_f \ : \ G$ *satisfies* $\mathcal{SC}_*(FT(G'))$ *iff* $FT(G') \subseteq FT(G)$

PROOF. If G satisfies the data constraints computed from $FT(G')$ then $FT(G') \subseteq FT(G)$ from Lemma 5.6.

If $FT(G') \subseteq FT(G)$ then by Lemma 5.7 $\mathcal{SC}_*(FT(G')) \subseteq \mathcal{SC}_*(FT(G))$; also by Lemma 5.5, G satisfies $\mathcal{SC}_*(FT(G))$, then it also satisfies a subset $\mathcal{SC}_*(FT(G'))$ as desired. ■

Notice that two grammars may satisfy a common set of constraints while the inclusion of their FT languages can still be strict (see Example 5.3).

Our algorithm specifies the set of every PG compatible with a set of feature-tagged examples. But even if we are able to solve the constraints it produces, it is not enough to make it a learning algorithm in the sense of Gold. A remaining problem is to identify a unique PG in the limit. To avoid over-generalisation, inclusion tests between feature-tagged languages may be necessary for this purpose, and we do not even know if these tests are computable. They can nevertheless be performed for feature-tagged examples of bounded length (this is Kanazawa's strategy for learning k-valued AB-Categorial Grammars from strings) but, of course, make the algorithm intractable in practice.

5.3. Why pregroup grammars and constraints are efficient

The main weakness of known learning algorithms for Categorial Grammars is their algorithmic complexity. The only favourable case is when rigid AB-Grammars are to be learned from structural examples but this situation is of limited interest. Our algorithm seems more tractable than previous approaches. To see how, we provide in the following two relevant examples.

5.3.1. First exponential gain: inner associativity

The first gain comes from the "inner associativity" of categories in PGs. Let a symbol "b" be associated with a category requiring $2n$ combinations with a category associated with a symbol "a", n of which are on its right and the other n on its left. The corresponding feature-tagged example is:

$$\begin{array}{ccc} \text{a ... a} & \text{b} & \text{a ... a} \\ \underbrace{e \text{ ... } e}_{n\ times} & \{s, \underbrace{e, ..., e}_{2n\ times}\} & \underbrace{e \text{ ... } e}_{n\ times} \end{array}$$

This case is equivalent with the problem of learning AB or Lambek Categorial Grammars from the following typed example [12, 11]:

$$\begin{array}{ccc} \text{a ... a} & \text{b} & \text{a ... a} \\ \underbrace{e \text{ ... } e}_{n\ times} & \langle \underbrace{e, \langle e, ... \langle e}_{2n\ times}, t\rangle\rangle ...\rangle & \underbrace{e \text{ ... } e}_{n\ times} \end{array}$$

There are $\binom{2n}{n}$ different AB-Categorial Grammars (and the same number of Lambek Grammars) compatible with this input. This situation occurs with transitive verbs, whose category is $T\backslash(S/T)$ or $(T\backslash S)/T$ (both corresponding to the same type $\langle e, \langle e, t\rangle\rangle$, i.e. the example with $n = 1$). In the Lambek calculus, the distinct categories assigned to "b" by each solution are deductible from one another, so the language recognized by each grammar is in fact the same. Lambek Grammars are associative, but at the *rule level.* PGs are also associative, but at the *category level.* The result is that there is only one PG (up to shifts) compatible with the previous feature-tagged example: the one assigning $\underbrace{e^r ... e^r}_{n\ times}\ s\ \underbrace{e^l ... e^l}_{n\ times}$ to b. In PGs, the category corresponding both to $T\backslash(S/T)$ and $(T\backslash S)/T$ is unique : it is $T^r S T^l$.

5.3.2. Second exponential gain: efficiency of constraints

Another exponential gain can be earned from the reduction of the learning problem to a constraints resolution problem. In the following example, position variables are shown under the primitive category they are asoociated with, for readability:

$$\begin{array}{cccc} \text{a} & \text{b} & \text{c} & \text{d} \\ \{s, m, , n\} & \{m, n, o, p, \} & \{o, p, q, r\} & \{q, r\} \\ \{x_1, x_2, x_3\} & \{x_4, x_5, x_6, x_7\} & \{x_8, x_9, x_{10}, x_{11}\} & \{x_{12}, x_{13}\} \end{array}$$

There are several PGs compatible with this feature-tagged example, all of which sharing the same values for exponents (up to shifts), but differing in the way they embed the two combinations to be applied on every couple of consecutive multisets. In the following figure, one solution -up to shifts- is shown above, the other one is shown under. As each choice is independent, there are $2^3 = 8$ different PGs compatible with this example but defined by a conjunction of 3 constraints (the position part of the first one is displayed on the right).

$$\begin{array}{cccc} sm^l n & n^r m o^l p & p^r o q^l r & r^r q \\ a & b & c & d \\ snm^l & mn^r po^l & op^r rq^l & qr^r \end{array} \qquad \begin{array}{l} ((x_2 < x_3) \wedge (x_5 < x_4)) \\ \vee((x_3 < x_2) \wedge (x_4 < x_5))... \end{array}$$

This example illustrates that there can exist an exponentiel gap between the size of the set of constraints and the size of the set of PGs it specifies. By working with constraints, we delay the combinatorial explosion to the constraint resolution mechanism. The solution grammars are only implicitly defined by the set of constraints but this implicit definition may be enough for some applications.

6. Conclusion

Pregroup Grammars appear to be an interesting compromise between simplicity and expressivity. Their link with semantics is a difficult question that has not yet been completely solved. As far as learnability is concerned, very few was known till now. This paper provides theoretical as well as practical approaches to the problem. Theoretical results prove that learning PGs is difficult unless limitations are known. The practical approach shows that the limitations can be weakened when rich input data are provided. These data take the form of feature-tagged sentences which, although very informative, are arguably language-independent.

Many current learning algorithms are unification-based [17, 21]. These techniques implement a *generalization strategy*. By providing richer input data, we allow on the contrary to implement a *specialization strategy*, where the current hypotheses are only implicitly specified by a set of constraints. The interest of working with constraints is that the combinatorial explosion of solution grammars is sometimes delayed to the constraint resolution mechanism, as displayed in the examples.

What remains to be done is to study further the properties of our algorithm from the point of view of tractability and to exploit further the good properties of bounded width grammars.

References

[1] Daniela Bargelli and Joachim Lambek. An algebraic approach to French sentence structure. In de Groote et al. [10].

[2] Denis Béchet. Parsing pregroup grammars and Lambek calculus using partial composition. *Studia Logica*, 2007, this issue.

[3] Denis Béchet and Annie Foret. Remarques et perspectives sur les langages de prégroupe d'ordre 1/2. In ATALA, editor, *Actes de la conférence internationale Traitement Automatique des Langues Naturelles (TALN'2003)*, June 2003. (Poster in French).

[4] Denis Béchet, Annie Foret, and Isabelle Tellier. Learnability of pregroup grammars. In Georgios Paliouras and Yasubumi Sakakibara, editors, *Proceedings of the 7th international Colloquium on Grammatical Inference (ICGI-2004), Athens, Greece, October 2004*, volume LNAI 3264 of *Lecture Note in Artificial Intelligence*, pages 65–76. Springer-Verlag, October 2004.

[5] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [25], chapter 12, pages 683–736.

[6] Wojciech Buszkowski. Lambek grammars based on pregroups. In de Groote et al. [10].

[7] Wojciech Buszkowski. Sequent systems for compact bilinear logic. *Mathematical Logic Quarterly*, 49(5):467–474, 2003.

[8] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[9] Claudia Casadio and Joachim Lambek. An algebraic analysis of clitic pronouns in Italian. In de Groote et al. [10].

[10] Philippe de Groote, Glyn Morill, and Christian Retoré, editors. *Logical aspects of computational linguistics: 4th International Conference, LACL 2001, Le Croisic, France, June 2001*, volume 2099. Springer-Verlag, 2001.

[11] Daniela Dudau-Sofronie and Isabelle Tellier. A study of learnability of Lambek grammars from typed examples. In *Categorial Grammars 04*, pages 133–147, 2004.

[12] Daniela Dudau-Sofronie, Isabelle Tellier, and Marc Tommasi. Learning categorial grammars from semantic types. In *Proccedings of the 13rd Amsterdam Colloquium*, pages 79–84, December 2001.

[13] Daniela Dudau-Sofronie, Isabelle Tellier, and Marc Tommasi. A learnable class of ccg from typed examples. In *Proccedings of the 8th conference on Formal Grammar (FGVienna)*, pages 77–88, August 2003.

[14] Annie Foret. Some unlearnability results for Lambek categorial and pregroup grammars (oral communication). In *Gracq, ESSLII*, Trento, Italy, 2002.

[15] Annie Foret and Yannick Le Nir. Lambek rigid grammars are not learnable from strings. In *COLING'2002, 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[16] E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[17] Makoto Kanazawa. *Learnable Classes of Categorial Grammars*. Studies in Logic, Language and Information. Center for the Study of Language and Information (CSLI) and The European association for Logic, Language and Information (FOLLI), Stanford, California, 1998.

[18] Joachim Lambek. Type grammars revisited. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, *Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers*, volume 1582. Springer-Verlag, 1999.

[19] Joachim Lambek. Mathematics and the mind. In V.M. Abrusci and C. Casadio, editors, *New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop*. Bulzoni Editore, 2001.

[20] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [25], chapter 2, pages 93–177.

[21] Jacques Nicolas. Grammatical inference as unification. Technical Report 3632, IRISA, Unité de Recherche INRIA Rennes, March 1999.

[22] Richard T. Oehrle. A parsing algorithm for pregroup grammars. In M. Moortgat, editor, *Proceedings of the International Conference on Categorial Grammars (CG2004)*, pages 59–75, Montpellier, France, June 2004.

[23] Anne Preller. Category Theoretical Semantics for Pregroup Grammars. In Joan Busquets Philippe Blache, Edward Stabler and Richard Moot, editors, *Proceedings of the 5th International Conference on Logical Aspects of Computational Linguistics, Bordeaux, France, April 2005*, volume LNAI 3492 of *Lecture Notes in Artificial Intelligence*, pages 238–254. Springer-Verlag, April 2005.

[24] Anne Preller. Toward discourse representation via pregroup grammars. *JoLLI*, 16:173–193, 2007.

[25] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

Denis Béchet
LINA & CNRS FRE 2729
2, rue de la Houssinière
BP 92208
44322 Nantes Cedex 03
France
`Denis.Bechet@univ-nantes.fr`

Annie Foret
IRISA & Université Rennes 1
Campus de Beaulieu
35042 Rennes Cedex
France
`Annie.Foret@irisa.fr`

Isabelle Tellier
GRAppA & Inria Futurs
Université Lille 3
59653 Villeneuve d'Ascq
France
`Isabelle.Tellier@univ-lille3.fr`