Learnability of Pregroup Grammars

Denis Béchet[1], Annie Foret[2], and Isabelle Tellier[3]

[1] LIPN - UMR CNRS 7030, Villetaneuse
Denis.Bechet@lipn.univ-paris13.fr
[2] IRISA - IFSIC, Campus de Beaulieu, Rennes
Annie.Foret@irisa.fr
[3] MOSTRARE project RU Futurs INRIA**
GRAppA & Inria Futurs, Lille
tellier@univ-lille3.fr

Abstract. This paper investigates the learnability of Pregroup Grammars, a context-free grammar formalism recently defined in the field of computational linguistics. In a first theoretical approach, we provide learnability and non-learnability results in the sense of Gold for subclasses of Pregroup Grammars. In a second more practical approach, we propose an acquisition algorithm from a special kind of input called Feature-tagged Examples, that is based on sets of constraints.
Key-words. Learning from positive examples, Pregroup grammars, Computational linguistics, Categorial Grammars, Context-Free grammars.

1 Introduction

Pregroup Grammars [1] (PGs in short) is a context-free grammar formalism used in the field of computational linguistics. This recently-defined formalism for syntax allies expressivity (in this respect it is close to Lambek Grammars) and computational efficiency. Subtle linguistic phenomena have already been treated in this framework [2, 3]. PGs share many features with Categorial Grammars of which they are inheritors, especially their lexicalized nature.

Since the seminal works of Kanazawa[4], a lot of learnability results in Gold's model [5] have been obtained for various classes of Categorial Grammars and various input data. But the learnability of PGs has yet received very little attention except a negative result in [6]. In the first part of this paper, we prove several results of learnability or of non-learnability for classes of PGs. But these results are mainly theoretical and are not associated with learning algorithms.

In the second part of the paper, we define an acquisition algorithm to specify a set of PGs compatible with input data. The input data considered, called Feature-tagged Examples, are richer than strings but chosen to be language-independent (inspired by [7–9]). The originality of the process is that it allows to reconsider the learning problem as a constraints resolution problem.

** This research was partially supported by: "CPER 2000-2006, Contrat de Plan état - région Nord/Pas-de-Calais: axe TACT, projet TIC"; fonds européens FEDER "TIC - Fouille Intelligente de données - Traitement Intelligent des Connaissances" OBJ 2-phasing out - 2001/3 - 4.1 - n 3. And by "ACI masse de données ACIMDD"

2 Pregroup Grammars

2.1 Background

Definition 1 (Pregroup). *A* pregroup *is a structure* $(P, \leq, \cdot, l, r, 1)$ *such that* $(P, \leq, \cdot, 1)$ *is a partially ordered monoid*[4] *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*[5] *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 (Free Pregroup). *Let* $(P, \leq)$ *be a partially ordered set of primitive categories,* $P^{(\mathbb{Z})} = \{p^{(i)} \mid p \in P, i \in \mathbb{Z}\}$ *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 Buskowski, defines a pregroup that extends* $\leq$ *on* P *to* $\mathcal{C}at_{(P,\leq)}$.

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

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

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

Fig. 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.*

Definition 3 (Pregroup Grammars). $(P, \leq)$ *is a finite partially ordered set. A* free pregroup grammar *based on* $(P, \leq)$ *is a lexicalized*[6] *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.

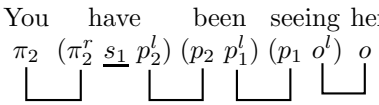
Width. We define the width of a category $C = p_1^{u_1} \ldots p_n^{u_n}$ as $wd(C) = n$ (the number of atomic categories).

[4] 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 $a \cdot c \leq b \cdot c$.

[5] We use this notation in technical parts

[6] 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.

Example 1. Our first example is taken from [10] with the basic categories: $\pi_2 =$ 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 $\pi_2\pi_2^r \leq 1$, $p_2^l p_2 \leq 1$, $p_1^l p_1 \leq 1$, $o^l o \leq 1$:



2.2 Parsing

Pregroup languages are context-free languages and their parsing is polynomial. We present in this section a parsing algorithm working directly on lists of words. For that, we first extend the notion of inference to lists of categories, so as to reflect the separations between the words of the initial string. 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)**: $\Gamma, Xp^{(n)}q^{(n+1)}Y, \Delta \vdash_I \Gamma, XY, \Delta$, if $q \leq p$ and n is even or if $p \leq q$ and n is odd.

E **(external)**: $\Gamma, Xp^{(n)}, q^{(n+1)}Y, \Delta \vdash_E \Gamma, X, Y, \Delta$, 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 $\vdash_{MIE}^*$.

Lemma 1. *For $X \in \mathcal{C}at_{(P,\leq)}$ and $p \in P$, $X \leq p$ iff $\exists q \in P$ such that $X \vdash_{MIE}^* q$ and $q \leq p$.*

Corollary 1. *$G = (\Sigma, I, s)$ 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 $p \leq s$.*

All $\vdash_I^*$ can be performed before $\vdash_M^*$ and $\vdash_E^*$ as the next lemma shows.

Lemma 2. *(easy) $\Gamma_1 \vdash_{MIE}^* \Gamma_2$ iff $\exists \Delta$ such that $\Gamma_1 \vdash_I^* \Delta$ and $\Delta \vdash_{ME}^* \Gamma_2$*

The external reductions corresponding to the same couple and a merge reduction can be joined together such that, at each step, the number of categories decreases.

E$^+$ **(external+merge)**: For $k \in \mathbb{N}$,
$\Gamma, Xp_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)}Y, \Delta \vdash_{E^+} \Gamma, XY, \Delta$, if $q_i \leq p_i$ and n_i is even or if $p_i \leq q_i$ and n_i is odd, for $1 \leq i \leq k$.

Lemma 3. *For a list of categories Γ and $p \in P$, $\Gamma \vdash_{ME}^* p$ iff $\Gamma \vdash_{E^+}^* p$.*

To define a polynomial algorithm, we finally constraint the application of $\vdash_{E^+}^*$ such that the width of the resulting category is never greater than the maximal width of the two initial categories: one category plays the role of an argument and the other plays the role of a functor even if the application is partial. The rule is thus called Functional. In fact, there is a small difference between left and right functional reductions (see the two different conditions $wd(X) \leq k$ or $wd(Y) < k$) to avoid some redundancies. The last condition $wd(Y) = 0$ is necessary when Δ is empty and $k = 0$ to mimic a (degenerated) merge reduction.

F (functional): For $k \in \mathbb{N}$,
$\Gamma, Xp_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)} Y, \Delta \vdash_F \Gamma, XY, \Delta$, if $q_i \leq p_i$ and n_i is even or if $p_i \leq q_i$ and n_i is odd, for $1 \leq i \leq k$ and if $wd(X) \leq k$ or $wd(Y) < k$ or $wd(Y) = 0$.

Lemma 4. *For a list of categories Γ and $p \in P$, $\Gamma \vdash^*_{E^+} p$ iff $\Gamma \vdash^*_F p$.*

Proof. The proof is based on the fact that for any planar graph where the vertice are put on a line and where the edges are only on one side of this line, there always exists at least one vertex that is connected only to one of its neighbours or to both of them but not to any other vertex. This vertex is then associated to its neighbour if it is connected to only one neighbour. If it is connected to its two neighbours, we choose the one that interacts the most with the vertex.

The parsing of a string with n words consists in the following steps:

1. Search for the categories associated to the n words through the lexicon.
2. Add the categories deduced with $\vdash^*_I$.
3. Compute recursively the possible categories associated to a contiguous segment of words of the string with $\vdash_F$.

The third step uses a function that takes the positions of the first and last words in the segment as parameters. The result is a set of categories with a bounded width (i.e. by the maximum width of the categories in the lexicon).

Property 1 *For a given grammar, this algorithm is polynomial (wrt. the number of words of input strings).*

Example 2. Parsing of "whom have you seen ?". The categories are as follows in the lexicon ($q' \leq s$):

whom	have	you	seen
$q'o^{ll}q^l$	$qp_2^l\pi_2^l$	π_2	p_2o^l

	...whom	...have	...you	...seen
seen...				$\{p_2o^l\}$
you...			$\{\pi_2\}$	$\emptyset$
have...		$\{qp_2^l\pi_2^l\}$	$\{qp_2^l\}$	$\{qo^l\}$
whom...	$\{q'o^{ll}q^l\}$	$\emptyset$	$\emptyset$	$\{q'\}$

The cell of line i (numbered from the bottom) and column j contains the category computed for the fragment starting at the i^{th} word and ending at the j^{th} word.

3 Learning

3.1 Background

We now recall some useful definitions and known properties on learning in the limit [5]. 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 function ϕ from finite sets of words in V^* to $\mathcal{G}$, such that $\forall G \in \mathcal{G}$, $\forall (e_i)_{i\in N}$ such that $\mathcal{L}(G) = (e_i)_{i\in N}$ $\exists G' \in \mathcal{G}$ 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 $< L_n >_{n\in N}$ of languages in $\mathcal{CL}$ and a language $L \in \mathcal{CL}$ such that : $L_0 \subsetneq L_1 \ldots \subsetneq L_n \subsetneq \ldots$ and $L = \bigcup_{n\in N} L_n$ (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 : $\forall i \in N \ : \ e_i \notin L_i$ 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.

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 [11] using [12]. So, no learning algorithm is possible. This has also been shown for subclasses of rigid PGs as summarized below (from [6]).

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 $\{a^{(i)} | a \in P \ , \ -n-1 \leq i \leq n\}$.

Construction of rigid limit points. We have proved [6] 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 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 free 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 $\mathcal{L}(G_*) = \{ab^k cd^k e \mid k \geq 0\}$.*

Corollary 2. *The classes $\mathcal{CG}^k_{n/2}$ of k-valued simple free pregroups of order $n/2, n \geq 0$ 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.

(i) *Width and order* bounds. Here, by the *order* of a category $C = p_1^{u_1} \ldots p_n^{u_n}$ we mean its integer order : $max\{ |u_i| \ / \ 1 \leq i \leq n\}$.

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

(ii) *Width* bounded categories. By normalizing with translations, we get:

Theorem 2 *The class of rigid (also k-valued for each k) PGs with categories of width less than N is learnable from strings for each N.*

Proof. Let G denote a rigid PG on Σ and n be the maximum width of G, we can show that G is equivalent (same language) to a similar PG of order $\leq 2n|\Sigma|$. This PG is a normalized version of G obtained by repeating translations as follows: consider possible iterations of r; if two consecutive exponents never appear in any iterated adjoints (a hole), decrease all above exponents; proceed similarly for iterations of l. Therefore, a bounded width induces a bounded order for rigid PGs; we then apply the learnability result for a class with a width bound and an order bound. In the k-valued case, we proceed similarly with an order $\leq 2n|\Sigma|k$.

(iii) *Pattern* of Category. We infer from known relationships between categorial formalisms, a case of learnability from strings for PGs. We refer to [13, 14] for definitions and details on formalisms.

From Lambek calculus to pregroup. We have a transformation $A \rightarrow [A]$ on formulas and sequents from $L_\emptyset$ (Lambek calculus allowing empty sequents) to the simple free pregroup, that translates a valid sequent into a valid inequality[7]:

$[A] = A$ when A is primitive

$[A \setminus B] = [A]^r[B]$; $[B \,/\, A] = [B][A]^l$

$[A_1, \ldots, A_n \vdash B] = [A_1] \cdots [A_n] \leq [B]$

The *order of a category* $o(A)$ for Categorial Grammars is:

$o(A)=0$ when A is primitive; $o(A \setminus B)=o(B \,/\, A)=max(o(A)+1, o(B))$

Lemma 5. [15] *If B is primitive and $o(A_i) \leq 1$ for $1 \leq i \leq n$ then:*

$A_1, \ldots, A_n \vdash_{AB} B$ *(Classical or AB Categorial Grammars)*

iff $A_1, \ldots, A_n \vdash_{L_\emptyset} B$ *(Lambek)*

iff $[A_1] \cdots [A_n] \leq B$ *(simple free pregroup)*

We infer the following result :

Theorem 3 *The class $\mathcal{C}_L^k$ of k-valued (simple free) PGs with categories of the following pattern (P_1) : $g_n^r \ldots g_1^r p d_1^l \ldots d_m^l$, where $n \geq 0$ and $m \geq 0$ is learnable from strings in Gold's model.*

Proof. The class of k-valued AB grammars is learnable from strings [4]. Lemma 5 shows that the class of PGs that are images of k-valued Lambek Grammars of order 1 (also k-valued AB-grammars with the same language) is also learnable. And when $o(A) \leq 1$, then $[A]$ must be written as: $g_n^r \ldots g_1^r p d_1^l \ldots d_m^l$.

Relevance of pattern P_1. We have observed that many linguistic examples follow the pattern P_1 or are images of these by some *increasing function*,

[7] The converse is not true :

$[(a \cdot b) \,/\, c] = abc^l = [a \cdot (b \,/\, c)]$ but $(a \cdot b) \,/\, c \not\vdash a \cdot (b \,/\, c)$

$[(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p] = pp^{ll}p^{ll}p^lp^l \leq [p]$ but $(p \,/\, ((p \,/\, p) \,/\, p)) \,/\, p \not\vdash p$

i.e. a function h such that $X \leq h(X)$ (for example type-raised introduction $h_{raise}(X) = ss^l X$); moreover if G assigns $h_i(t_i)$ to c_i, where all h_i are increasing and all t_i have the pattern P_1, we consider G_{P_1} assigning t_i to c_i and get : $L(G) \subseteq L(G_{P_1})$ and the class of G_{P_1} is learnable from strings.

3.4 Learning Pregroup Grammars from Feature-Tagged Examples

Previous learnability results lead to non tractable algorithms. But an idea from Categorial Grammars 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. Under some conditions specifying the link between categories and types, interesting subclasses of AB-Categorial Grammars and of Lambek Grammars have been proved learnable from Typed Examples, i.e. from sentences where each word is associated with its semantic type [8, 9].

To adapt this idea to PGs, the first problem is that the link between PGs and semantics is not clearly stated. So, the notion of semantic types has no obvious relevance in this context and our first task is to identify what can play the role of language-independent features in 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. Finally, we present an algorithm whose purpose is to identify every possible PG of this class compatible with a set of Feature-tagged Examples. An original point is that this set will be specified by a set of constraints. We provide examples showing that this set can be exponentially smaller than the set of grammars it specifies.

Specification of Input Data. Let us consider how the various 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 various PGs (Figure 2): The common points between every possible analysis

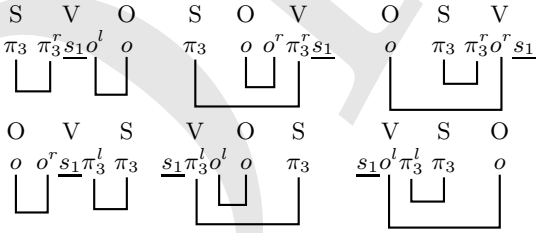


Fig. 2. Pregroup Grammars and possible word orders

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. 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.

Definition 4. *For any PG $G = (\Sigma, I, s)$, the Feature-tagged Language of G, noted $FT(G)$, is defined by: $FT(G) = \{\langle v_1, T_1\rangle ... \langle v_n, T_n\rangle | \forall i \in \{1, ..., n\} \exists X_i \in I(v_i)$ such that $X_1 ... X_n \leq s$ and $T_i = f_P(X_i)\}$*

Example 3. Let $P = \{\pi_3, o, s, s_1\}$ with $s_1 \leq s$, $\Sigma = \{he, loves, her\}$ and let $G = (\Sigma, I, s)$ with $I(he) = \{\pi_3\}$, $I(loves) = \{\pi_3^r s_1 o^l\}$, $I(her) = \{o\}$. We have: $\langle he, \{\pi_3\}\rangle\langle loves, \{s_1, \pi_3, o\}\rangle\langle her, \{o\}\rangle \in FT(G)$. An element of $FT(G)$ is a Feature-tagged Example. We study how PGs can be learned from such examples.

Definition 5. *For any sets Σ and P, we call $\mathcal{G}_f$ the set of PGs $G = (\Sigma, I, s)$ satisfying: $\forall v \in \Sigma$, $\forall X_1, X_2 \in I(v)$: $f_P(X_1) = f_P(X_2) \Longrightarrow X_1 = X_2$*

Theorem 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 $V = \Sigma \times \mathcal{M}(\mathcal{P})$).*
Proof. The theorem is a corollary of Theorem 2, where k and N can be computed from any sequence of Feature-tagged Examples that enumerates $FT(G)$:
- the condition satisfied by a PG for being an element of $\mathcal{G}_f$ implies that the number of distinct multisets associated with the same word in Feature-tagged Examples is the same as the number of distinct categories associated to it by function I. So k can be easily obtained.
- the width of a category is exactly the number of elements in the corresponding multiset, so N can also be easily obtained.

Acquisition Algorithm. Our algorithm takes as input a set of Feature-tagged Examples for some $G \in \mathcal{G}_f$ and provides a set of PGs. We conjecture (although we haven't proved yet) that the output is exactly, up to basic transformations, the set of every PGs compatible with the input. The algorithm has two steps: first variables are introduced, then constraints are deduced on their values.

First Step: Variable Introduction. Although Feature-tagged Examples provide a lot of information, two things remain to be learned: the nature of the potential exponents of categories and their relative positions inside a concatenation. We introduce variables to code both problems. Variables for the exponents take their value in $\mathbb{Z}$, those for the relative positions take their value in $\mathbb{N}\backslash\{0\}$.

Example 4. The Feature-tagged Example of Example 3 gives:
he: $T_1 = \{(\pi_3^u, x_{11})\}$
loves: $T_2 = \{(s_1^v, x_{21}), (\pi_3^{v'}, x_{22}), (o^{v''}, x_{23})\}$
her: $T_3 = \{(o^w, x_{31})\}$
with $u, v, v', v'', w \in \mathbb{Z}$. $\forall i, j$, $x_{ij} \in \mathbb{N}\backslash\{0\}$ is the position of the j^{th} primitive category of the i^{th} word. The following constraints and consequences are available:
$\{x_{11}\} = \{1\} \Longrightarrow x_{11} = 1$; $\{x_{21}, x_{22}, x_{23}\} = \{1, 2, 3\}$; $\{x_{31}\} = \{1\} \Longrightarrow x_{31} = 1$

This coding allows to reformulate the learning problem into a variable assignment problem. Furthermore, as the Feature-tagged Examples belong to the same $FT(G)$ for some G in $\mathcal{G}_f$, the *same variables* are used for every occurrence of the same couple $\langle word, multiset \rangle$ in the set of Feature-tagged Examples.

Second Step: Constraints Deduction. This step consists in deducing constraints applying on the variables. Each Example is treated one after the other. For a given Example, we call T_i the multiset associated with the i^{th} word. Each initial sentence of n words is then replaced by a sequence of n multisets. Constraint deduction takes the form of rules that mimic the rules I and F used for the parsing of PGs in section 2.2. 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, we thus obtain a disjunction of conjunctions of basic constraints (that we call *data constraint*) where each basic constraint consists in an exponent part and a position part.

Let $T_m = \{(p^u_{mi}, x_{mi})_{1 \leq i \leq k}\}$ and $T_{m'} = \{(p^{u'}_{m'j}, x_{m'j})_{1 \leq j \leq k'}\}$ be two consecutive sets (at the beginning: $m' = m+1$). If $\exists (p^u_{mi_0}, x_{mi_0}) \in T_m$ and $\exists (p^{u'}_{m'j_0}, x_{m'j_0}) \in T_{m'}$ such that $p_{mi_0} \leq p_{m'j_0}$ or $p_{m'j_0} \leq p_{mi_0}$ then:

- Position constraints:
  - $\forall i \neq i_0, \forall x_{mi} \in T_m$: $x_{mi_0} > x_{mi}$
  - $\forall j \neq j_0, \forall x_{m'j} \in T_{m'}$: $x_{m'j_0} < x_{m'j}$
- Exponent constraints:
  - (all cases) $u' = u + 1$
  - IF $p_{m'j_0} < p_{mi_0}$ THEN: u is odd
  - IF $p_{mi_0} < p_{m'j_0}$ THEN: u is even
- Next sets:
  - $T_m \longleftarrow T_m - (p^u_{mi_0}, x_{mi_0})$
  - $T_{m'} \longleftarrow T_{m'} - (p^{u'}_{m'j_0}, x_{m'j_0})$

For internal reductions, where $m = m'$, the Position constraint is replaced by: $\forall i \neq i_0, i \neq j_0$: $x_{mi} < x_{mi_0}$ or $x_{mj_0} < x_{mi}$

Whenever a set T_i becomes empty, drop it. The process ends when the list gets reduced to some $\{(p^u, x)\}$ where $p \leq s$ (the constraint $u = 0$ is deduced).

If a primitive category satisfying the precondition of the rules has several occurrences in a multiset, any of them can be chosen (they are interchangeable). By convention, take the one associated with the position variable of smallest index. Example 6 (further) illustrates this case. To efficiently implement these rules, an interesting strategy consists in following the parsing steps of section 2.2.

Example 5. Let us see what this algorithm gives on our basic Example 4 where the initial sequence of multisets is: $T_1 T_2 T_3$:

- $(\pi^u_3, x_{11}) \in T_1$ and $(\pi^{v'}_3, x_{22}) \in T_2$ satisfy the precondition. The position constraints obtained are: $x_{22} < x_{21}$ and $x_{22} < x_{23}$. The exponent constraint is: $v' = u + 1$, and the remaining sets are the following: $T_1 = \emptyset$, $T_2 = \{(s^v_1, x_{21}), (o^{v''}, x_{23})\}$, $T_3 = \{(o^w, x_{31})\}$

– then $(o^{v''}, x_{23}) \in T_2$ and $(o^w, x_{31}) \in T_3$ satisfy the precondition. We deduce: $x_{23} > x_{21}$, $w = v'' + 1$ and $T_2 = \{(s_1^v, x_{21})\}$, $T_3 = \emptyset$. As $s_1 \leq s$, we obtain $v = 0$ and the algorithm stops.

From the constraints: $\{x_{21}, x_{22}, x_{23}\} = \{1, 2, 3\}$, $x_{22} < x_{21}$, $x_{22} < x_{23}$ and $x_{23} > x_{21}$ we deduce: $x_{21} = 2$, $x_{22} = 1$ and $x_{23} = 3$. The PGs specified by these constraints are defined up to a translation on the exponents. If we set $u = 0 = w$, then $v' = 1$ (or $v' = r$) and $v'' = -1$ (or $v'' = l$): the only remaining PG associates $\pi_3^r s_1 o^l$ with "loves". But, in general, solution PGs are only specified by a set of constraints. We will see that this set can be exponentially smaller than the set of classes (up to translations) of PGs it specifies.

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 2 *Let $G \in \mathcal{G}_f$, and $FT(G) = \{e_i\}_{i \in \mathbb{N}}$, the data constraints $\mathcal{DC}_i$ obtained from the successive e_i converges*[8]*: $\exists n_0 \in \mathbb{N} \ \forall n \geq n_0 \ : \mathcal{DC}_{n+1} = \mathcal{DC}_n$*

Proof. At some stage N of the acquisition process, the set of primitive categories and the widths of the categories assigned to each word become known; after this N, we have only a finite number of possibilities for data constraints, that must therefore converge.

Even if our acquisition algorithm finds every possible PG compatible with a set of Feature-tagged Example, 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. 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 Example 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.

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 Example but this situation is of limited interest. Our algorithm can still sometimes lead to combinatorial explosion but seems more tractable than previous approaches, as shown by the following two examples.

Example 6 (First exponential gain). The first gain comes from the associativity of categories in PGs. Let a word "b" be associated with a category expecting $2n$ arguments of a category associated with a word "a", n of which are on its right and the other n on its left. The corresponding Feature-tagged Example is:

$$\underbrace{a \ldots a}_{n \text{ times}} \ \{s, \ \underbrace{e, \ldots, e}_{2n \text{ times}} \} \ \underbrace{a \ldots a}_{n \text{ times}}$$

[8] we consider constraints written in a format without repetition

This case is equivalent with the problem of learning AB or Lambek Categorial Grammars from the following Typed Example [7, 9]:

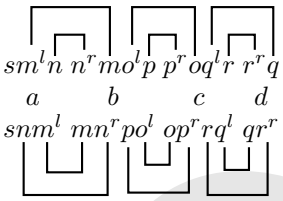
$$\underbrace{\text{a ... a}}_{n\ times}\ \underbrace{\text{b}}_{2n\ times}\ \underbrace{\text{a ... a}}_{n\ times}$$
$$\underbrace{e\ ...\ e}_{n\ times}\ \underbrace{\langle e, \langle e, ...\langle e}_{2n\ times}, t\rangle\rangle ...\rangle\ \underbrace{e\ ...\ e}_{n\ times}$$

There are $\binom{2n}{n}$ different Categorial 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 $n = 1$). The distinct categories assigned to "b"by each solution are deductible from one another in the Lambek calculus. Lambek Grammars are associative, but at the *rule level*, whereas PGs are associative at the *category level*. The only compatible PG (up to translations) is the one assigning $\underbrace{e^r ... e^r}_{n\ times}\ s\ \underbrace{e^l ... e^l}_{n\ times}$ to b.

Example 7 (Second exponential gain). 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 for readability:

a	b	c	d
$\{s, m, n\}$	$\{m, n, o, p\}$	$\{o, p, q, r\}$	$\{q, r\}$
$\{x_{11}, x_{12}, x_{13}\}$	$\{x_{21}, x_{22}, x_{23}, x_{24}\}$	$\{x_{31}, x_{32}, x_{33}, x_{34}\}$	$\{x_{41}, x_{42}\}$

There are several PGs compatible with this input, all of which sharing the same values for exponent variables, but differing in the way they embed the two reductions to be applied on each distinct category (one solution -up to translations- 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 first one is displayed on the right).



$$((x_{12} < x_{13}) \wedge (x_{22} < x_{21})) \vee ((x_{13} < x_{12}) \wedge (x_{21} < x_{22}))$$

4 Conclusion

Pregroup Grammars appear to be an interesting compromise between simplicity and expressivity. Their link with semantics is still an open question. 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 is provided. These data take the form of Feature-tagged sentences which, although very informative, are arguably language-independent. The interest of working with constraints is that the solution grammars are only implicitly defined by the output. The combinatorial explosion of solution grammars is then sometimes

delayed to the constraint resolution mechanism, as displayed in the examples. As many current learning algorithms are unification-based [4, 16], the use of constraints may also be seen as a natural generalization of such techniques. What remains to be done is to study further the properties of our algorithm, both from the point of view of tractability, and from the point of view of formal properties and to exploit further the good properties of bounded width grammars.

References

1. Lambek, J.: Type grammars revisited. In Lecomte, A., Lamarche, F., Perrier, G., eds.: Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers. Volume 1582., Springer-Verlag (1999)
2. Bargelli, D., Lambek, J.: An algebraic approach to french sentence structure. [17]
3. Casadio, C., Lambek, J.: An algebraic analysis of clitic pronouns in italian. [17]
4. Kanazawa, M.: Learnable classes of categorial grammars. Studies in Logic, Language and Information. FoLLI & CSLI (1998) distributed by Cambridge University Press.
5. Gold, E.: Language identification in the limit. Information and control **10** (1967) 447–474
6. Béchet, D., Foret, A.: Remarques et perspectives sur les langages de prgroupe d'ordre 1/2. In ATALA, ed.: Actes de la conférence internationale Traitement Automatique des Langues Naturelles (TALN'2003). (2003) (Poster in French).
7. Dudau-Sofronie, D., Tellier, I., Tommasi, M.: Learning categorial grammars from semantic types. In: Proccedings of the 13rd Amsterdam Colloquium. (2001) 79–84
8. Dudau-Sofronie, D., Tellier, I., Tommasi, M.: A learnable class of ccg from typed examples. In: Proccedings of the 8th conference on Formal Grammar (FGVienna). (2003) 77–88
9. Dudau-Sofronie, D., Tellier, I.: A study of learnability of lambek grammars from typed examples. In: Categorial Grammars 04. (2004) 133–147
10. Lambek, J.: Mathematics and the mind. In Abrusci, V., Casadio, C., eds.: New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop, Bulzoni Editore (2001)
11. Foret, A.: Some unlearnability results for lambek categorial and pregroup grammars (oral communication). In: ESSLII, Trento, Italy (2002)
12. Foret, A., Le Nir, Y.: Lambek rigid grammars are not learnable from strings. In: COLING'2002, 19th International Conference on Computational Linguistics, Taipei, Taiwan (2002)
13. Buszkowski, W.: Mathematical linguistics and proof theory. [18] chapter 12 683–736
14. Moortgat, M.: Categorial type logic. [18] chapter 2 93–177
15. Buszkowski, W.: Lambek grammars based on pregroups. [17]
16. Nicolas, J.: Grammatical inference as unification. Technical Report 3632, IRISA, Unit de Recherche INRIA Rennes (1999)
17. de Groote, P., Morill, G., Retoré, C., eds.: Logical aspects of computational linguistics: 4th International Conference, LACL 2001, Le Croisic, France, June 2001. Volume 2099., Springer-Verlag (2001)
18. van Benthem, J., ter Meulen, A., eds.: Handbook of Logic and Language. North-Holland Elsevier, Amsterdam (1997)