PPQ : a pregroup parser using majority composition

Denis Béchet[1] and Annie Foret[2]

[1] LINA CNRS – UMR 6241 – Université de Nantes
2, rue de la Houssiniére – BP 92208
44322 Nantes Cedex 03 – France
Denis.Bechet@univ-nantes.fr
[2] IRISA – Université de Rennes 1
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex – France
Annie.Foret@irisa.fr

Abstract. Pregroup grammars are a mathematical formalism in the spirit of categorial grammars. They are close to logical formalism like Lambek calculus but have a polynomial parsing algorithm. The paper presents a parser based on pregroup grammar that uses a tabular approach based on *majority partial composition*.
Keywords: parser, pregroups, Lambek categorial grammars, parsing software, XML data.

1 Introduction

Pregroup grammars (PG) [1] have been introduced as a simplification of Lambek calculus [2]. They have been used to model fragments of syntax of several natural languages. They belong to categorial and lexicalized grammatical frameworks : categorial grammars have nice links to semantical interpretation while lexicalism has many advantages for constructing grammars and for parsing.

Another interest of PG is their order on basic types, that helps grammar design with natural and compact types (less types) ; this point can also be generalized to combine calculi, both formally and in software [3]. In contrast to some other categorial variants, PG parsing is polynomial ($O(n^3)$).

Based on the PG formalism, and some extensions of it, we have programmed a pregroup toolbox, including a specific parser based on partial composition (in contrast to [4, 5]), and a grammar definition tool. Data are stored in XML format (with DTD), to allow better interconnections with other tools. A web version is also provided for parsing with a grammar, either from raw text, from (partially) parenthesized text or from analyzed text (as in XML treebanks). This article explains the tool characteristics in connection with the underlying formalism, and gives an overview of the toolbox.

2 Pregroups

2.1 Pregroup Grammars

A *pregroup* is a structure $(P, \leq, \cdot, l, r, 1)$ such that $(P, \leq, \cdot, 1)$ is a partially ordered monoid and l, r are two unary operations on P that satisfy for every element $x \in P$, $x^l x \leq 1 \leq x x^l$ and $x x^r \leq 1 \leq x^r x$. We write $p^{(0)} = p$, and $p^{(n)}$ for $(p^{(n-1)})^r$ if $n > 0$ and $p^{(n)}$ for $(p^{(n+1)})^l$ if $n < 0$; these are called simple types. From now on, let P denote a free pregroup based on a poset of basic types written $(Pr, \leq_{Pr})$.

A *Pregoup Grammar* G is $G \subset \Sigma \times P$ (Σ words, G finite). Its language $L(G) \subseteq \Sigma^+$ is the set of sequence of words such that the concatenation of types entails ($\leq$) the distinguished type s.

Parsing can be based on rewrite rules such as: $Xp^{(n)}q^{(n+1)}Y \overset{(GCON)}{\longrightarrow} XY$
if $p \leq_{Pr} q$ and n is even or if $q \leq_{Pr} p$ and n is odd

Parsing using partial and majority composition Rules below proceed by pairs of words (their types are separated by a comma) ; thus parsing also provides a binary tree on words.

- [**C**] **(partial composition)** : for $k \in \mathbb{N}$, $X' = p_1^{(n_1)} \cdots p_k^{(n_k)}, Y' = q_k^{(n_k+1)} \cdots q_1^{(n_1+1)}$

$$\Gamma, Xp_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)} Y, \Delta \overset{C}{\longrightarrow} \Gamma, XY, \Delta$$

  if $p_i \leq_{Pr} q_i$ and n_i is even or if $q_i \leq_{Pr} p_i$ and n_i is odd , for $1 \leq i \leq k$.
- [$\overset{@}{\longrightarrow}$] **(majority composition)** : if (moreover) the result's width (of $|XY|$) is not greater than the maximum argument width (of XX' and $Y'Y$)

2.2 Pregroup extended with iteration types

For iteration types p^*, the parser is also based on partial composition rules:

- [**C**] **(partial composition)** : for $X'Y' \leq Z'$, with Z' empty (as 1) or $a^{*(2k+1)}$:

$$\Gamma, XX', Y'Y, \Delta \overset{C}{\longrightarrow} \Gamma, XZ'Y, \Delta$$

- [$\overset{@}{\longrightarrow}$] **(majority composition)** : if (moreover) at most a half of the argument is in the result ($|X'| \geq |X|$ or $|Y'| \geq |Y|$).

Example 1. Let us see the following sentence taken from "Un amour de Swann" by M. Proust: *Maintenant, tous les soirs, quand il l'avait ramenée chez elle, il fallait qu'il entrât.* [3] In Fig. 1 we show a proof of correctness of assignment of types to its fragment. The primitive types used in this proof are: π_3 and $\overline{\pi_3}$ = third person (subject) with $\pi_3 \leq \overline{\pi_3}$, p_2 = past participle, ω = object, s = sentence, s_5 = subjunctive clause, with $s_5 \leq s$, σ = complete subjunctive clause, τ = adverbial phrase. This grammar assigns s to the following sentence:

quand il l' avait ramenée chez-elle il fallait qu' il entrât

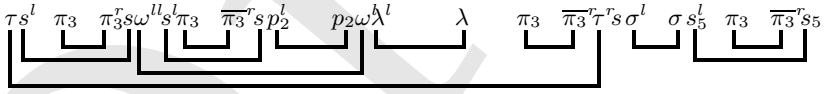
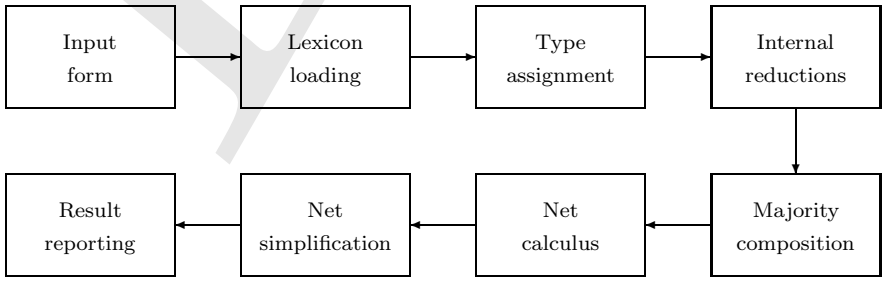


Figure 1

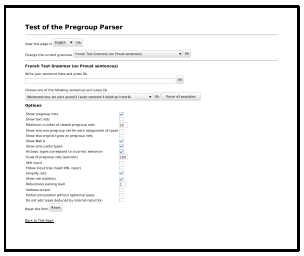
3 Parsing using majority partial composition

A Cocke-Younger-Kasami (CYK) algorithm for pregroup grammar can be developed. The granularity of this algorithm is words (or entries if the lexicon assigns also types to a sequence of words like "pomme de terre" (potato in French)). This method presented in [6] has been implemented into a tabular parser together with other components:



[3] [FR: *Now, every evening when he took back her to her home, he ought to enter*]

Input form.



This form selects one of the grammar, asks for the input string (alternatively, one may choose one or all samples that are associated to the selected lexicon). Several options are also offered. For instance, the parser can show only a limited number of analyses (ten by default). The user can enter a different limit. An option selects if the input is a string or an XML tree. In this case, the parser can also follow the XML structure when computing majority partial composition.

Lexicon loading.

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar>
 <pregroup>
  <order inf="n" sup="n-bar"/>
...
 </pregroup>
 <sentence type="s"/>
 <lexicon>
  <w><mot>whom</mot>
     <type><simple atom="q'"/>
           <simple atom="o" exponent="-2"/>
           <simple atom="q" exponent="-1"/>
     </type>
  </w>
...
 </lexicon>
</grammar>
```

Grammars are described in XML files. A grammar defines a partial order on basic types, a set of basic types that are considered to form the correct sentences and a lexicon that associated to an entry (a list of tokens) a set of types. It is also possible to describe special entries with a regular expression that is useful for instance for the class of numerical number or the class of proper noun (that starts with an upper case letter). To improve the efficiency of this step that may be very long if the lexicon is big (Lefff 2.5.5[7] has 534753 entries – The PPQ XML corresponding lexicon is a file whose size is 31,367,146 bytes), a compressed text format or a SQLite database can be used.

With an indexed table, even a big lexicon is accessed very quickly (less than a second rather than several tens of seconds). In fact, the parser does not load all the lexicon. It selects the entries that correspond to the input string.

Type assignment to words/entries. This step assigns to each list of tokens of the input string a set a pregroup types. The input string is split into tokens using spaces and several regular expressions For instance *l'homme* (the man in French) is segmented into two tokens: *l'* and *homme*. If the string is split in n tokens, there are $n \times (n+1)/2$ possible entries. Each one is searched in the lexicon and defines the initial value of the parsing matrix that computes the types associated with each segment of tokens of the input string.

Internal reduction of types. Because the main step of the parser is based on majority partial composition, a completion using internal reduction must be performed on the types computed during the previous step (see [6]). Usually this step is not useful because the types in the grammar cannot be reduced or the grammar already includes all the derived types.

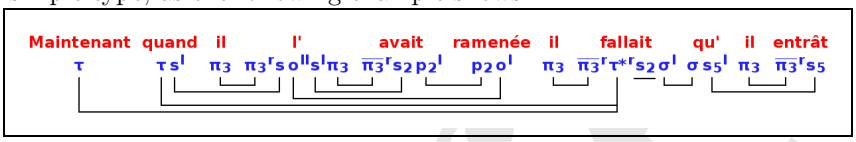
Majority partial composition of sequence of entry.

The Matrix Content

cell 1-4 q'			
cell 1-3 q' o^{ll} p_2^{l}	cell 2-4 q o^{l}		
cell 1-2	cell 2-3 q p_2^{l}	cell 3-4	
cell 1-1 q' o^{ll} q^{l}	cell 2-2 q p_2^{l} π_2^{l}	cell 3-3 π_2	cell 4-4 p_2 o^{l}
whom	have	you	seen

This step computes the parsing matrix with the result of majority partial composition (rather than using production rules of the Chomsky normal form of a context-free grammar for CYK algorithm). Of course, because we also want to describe the resulting analyses as pregroup nets, the matrix is in fact a complex directed acyclic graph. This matrix may be displayed by the parser at the end of the report. The matrix of "whom have you seen" as input string for Test grammar is displayed on the left.

Net calculus. This step computes representation of the analyses of the parser. They are called pregroup nets. In a net, each entry is associated to a pregroup type and the link represents the different axioms that associate two by two the simple types. This representation is close to a dependency tree except that the structure is a graph rather than a tree. Moreover, with the introduction of iterative simple types[8], a simple type can be connected to more than one other simple type, as the following example shows.



Net simplifications. This step simplifies nets by suppressing the iterated simple types that are not used and by taking into account cut annotations (see Section 4).

Result reporting. This step puts together all the results and presents it using different formats. Actually, there are three possible output formats: an HTML format useful for a web server, a text output that is suitable for a terminal and an XML format that may be used if the output needs to be processed by another program.

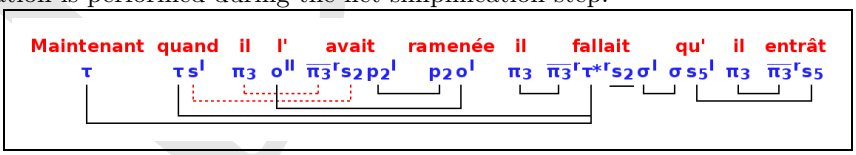
4 Parsing with added cuts

Grammars based on free pregroups even with iterative simple types are context free. Thus, for several complex syntactical constructions some types must include a way to cross part of the environment. This is particularly the case for non projective dependencies. For instance, the French clitics are placed between the verb and the subject: In "il la mange" (he is eating it) "la" is between "il" and "mange". The previous example shows such a construction. The clitic "l'" is assigned $\pi_3^r s o^{ll} s^l \pi_3$. The main simple type is o^{ll}. The rest enables the crossing of two axioms (one corresponding to π_3^r and π_3, the other to s and s^l).

To solve this problem, PPQ uses *cut* annotations on the types assigned to special words like clitics or adverb. These annotations that can be seen as a limited form of semantical interpretation replace two normal axiom links by a long distance axiom link.

The previous French example has such annotations for the type associated to the clitic *l'*. Here two cuts have been added, one between π_3^r and π_3 and one between s and s^l. Thus on the net, the axiom between π_3 of *il* and π_3^r of *l'* and the axiom between π_3 of *l'* and π_3^r of *avait* are replaced by a single long distance axiom between π_3 of *il* and π_3^r of *avait*. Another long distance axiom is created for the other cut that links s^l of *quand* and s^2 of *avait*.

On the final picture, these special long distance axioms are shown as dashed red lines. Such lines can cross other axioms. The cut simple types are also erased from the picture. This interpretation is performed during the net simplification step.



5 Grammar construction

Other packages concern the construction of XML pregroup grammars : xslt programs have been developed for this task, including a specific mode for the French Paris7 Treebank. Another set of programs (XML2CTX, LIS2XML) provides an interface with Camelis/Glis (http://www.irisa.fr/LIS/ferre/camelis/index.html) an implementation of Logical Information Systems (LIS), allowing navigation. A user can define a lexicon with Glis, then save it

as a LIS context, where objects are words ; this context is then transfered to the pregroup XML format (using LIS2XML) conversely, a pregroup grammar in XML format, can be transfered to a LIS context (using XML2CTX). This mode has been used for several prototype languages.

6 Conclusion

The pregroup parser PPQ implements majority partial composition. This program, that can be used inline through a PHP webserver or as a command line program, uses XML files for describing a pregroup grammar. An optional indexed database can speed up the lookup in the lexicon. The result is a HTML or text page with pregroup nets as syntactical analysis that is convenient for human reading. The command line program can also produce a XML output if the result must be used by another program. This model also enables a form of semantical interpretation limited to the reduction of annotated "cuts".

This program which is rather a test platform than a finished software has at present a large cover of the French language (however with a rough pregroup type system) and several toy lexicons for English, Breton (a Celtic language) and Bambara (an African language).

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. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly **65** (1958)
3. Foret, A.: Pregroup calculus as a logical functor. In: Proceedings of WOLLIC 2007. Volume LNCS 4576., Springer (2007)
4. Degeilh, S., Preller, A.: Efficiency of pregroup and the french noun phrase. Journal of Language, Logic and Information **14**(4) (2005) 423–444
5. Oehrle, R.: A parsing algorithm for pregroup grammars. In Moortgat, M., Prince, V., eds.: Proc. of Intern. Conf. on Categorial Grammars, Montpellier (2004)
6. Béchet, D.: Parsing pregroup grammars and Lambek calculus using partial composition. Studia logica **87**(2/3) (2007)
7. Sagot, B., Clément, L., de la Clergerie, E.V., Boullier, P.: The lefff 2 syntactic lexicon for french: architecture, acquisition. In: LREC'06. (2006)
8. Béchet, D., Dikovsky, A., Foret, A., Garel, E.: Optional and iterated types for pregroup grammars. In: Proceedings of the 2nd International Conference on Language and Automata Theory and Applications (LATA 2008), March 2008, Tarragona, Spain. Lecture Notes in Computer Science (LNCS), Springer (2008) 88–100
9. Došen, K.: Cut Elimination in Categories. Kluwer Academic publishers (1999)
10. Buszkowski, W.: Cut elimination for the lambek calculus of adjoints. In Abrusci, V., Casadio, C., eds.: New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop, Bulzoni Editore (2001)