Une boîte à outils pour développer et utiliser les grammaires de prégroupe

Denis Béchet
LINA – UMR 6241 – Université de Nantes
2, rue de la Houssinière – BP 92208
44322 Nantes Cedex 03 – France
Denis.Bechet@univ-nantes.fr

Annie Foret
IRISA – Université de Rennes 1
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex – France
Annie.Foret@irisa.fr

Résumé

Les grammaires de prégroupes sont un formalisme dans l'esprit des grammaires catégorielles et du calcul de Lambek mais contrairement à ces dernières, elles sont analysables en temps polynomial. Nous présentons dans cet article une boîte à outils contenant un analyseur et un ensemble de programmes permettant de développer et d'utiliser des grammaires en particulier pour le français.

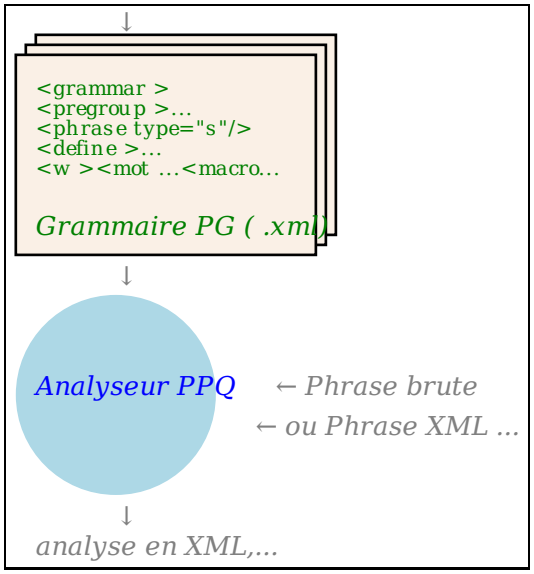
1 Introduction

Les grammaires de prégroupes (PG) (**?**) ont été introduites comme une simplification du calcul de Lambek (**?**). Elles ont été utilisées pour modéliser des fragments de la syntaxe de plusieurs langages naturels : anglais (**?**), italien (**?**), français (**?**), allemand (**?**; **?**), japonais (**?**), perse (**?**), etc.

Elles appartiennent à la classes des grammaires catégorielles et sont fortement lexicalisées : les grammaires catégorielles ont des liens privilégiés avec l'interprétation sémantique tandis que l'aspect lexicalisé apporte des avantages pour la construction des grammaires et pour l'analyse syntaxique.

Un autre interêt des PG réside dans leur possibilité de définir un ordre sur les types primitifs, ce qui aide à la conception des grammaires en introduisant des types compacts et moins nombreux (comparées à d'autres types de grammaires catégorielles par exemple). Ce point permet aussi de combiner les systèmes de manière hiérarchisée (**?**). De plus, contrairement à des variantes de grammaires catégorielles comme le calcul de Lambek, l'analyse syntaxique des PG reste polynomial.

Nous avons développé une boîte à outils pour le formalisme des PG et certaines extensions : cet ensemble de modules contient un analyseur syntaxique pour les prégroupes, basé sur la composition majoritaire (contrairement à (**?**; **?**)) ainsi qu'un ensemble de programmes et de ressources permettant la contruction de grammaires en particulier pour le français. Les formats utilisés pour les données sont des formats XML (avec DTD) pour faciliter l'interconnection avec d'autres outils.



Une version web de l'analyseur est aussi fournie avec un analyseur pouvant traiter du texte brut, du texte partiellement parenthésé ou du texte analysé (comme les arbres syntaxiques).

Cet article présente les caractéristiques de cette boîte à outils.

2 Prégroupe

2.1 Grammaires de prégroupes

Prégroupe. Un *prégroupe* est une structure $(P, \leq, \cdot, l, r, 1)$ telle que $(P, \leq, \cdot, 1)$ est un monoïde partiellement ordonné et l, r sont deux opérations unaires sur P vérifiant pour tout élément : $x \in P$, $x^l x \leq 1 \leq x x^l$ et $x x^r \leq 1 \leq x^r x$.

Nous notons $p^{(0)} = p$, et $p^{(n)}$ pour $(p^{(n-1)})^r$ si $n > 0$ et $p^{(n)}$ pour $(p^{(n+1)})^l$ si $n < 0$; ces

types sont appelés des *types simples*. Désormais, P désignera un prégroupe libre avec un ordre partiel sur des types de base (Pr,$\leq_{Pr}$).

Grammaire de prégroupe. Une *grammaire de prégroupe* G est $G \subset \Sigma \times P$ (Σ ensemble de mots, G fini). Son langage $L(G) \subseteq \Sigma^+$ est l'ensemble des séquences de mots $w_1..w_n$ dont une concaténation de types $t_1...t_n$ (où t_i est un type de w_i) a pour conséquence un type distingué s (dans le calcul logique des prégroupes).

L'analyse syntaxique peut être basée sur des règles de réécriture telles que :

$$Xp^{(n)}q^{(n+1)}Y \overset{(GCON)}{\longrightarrow} XY$$

où $p \leq q$ si n est pair, et $q \leq p$ si n est impair

Une autre approche consiste à utiliser ce système de déduction logique :

$X \leq X \ \ (Id)$	$\dfrac{X \leq Y \ \ Y \leq Z}{X \leq Z}\,(Cut)$
$\dfrac{XY \leq Z}{Xp^{(n)}p^{(n+1)}Y \leq Z}\,(A_L)$	$\dfrac{X \leq YZ}{X \leq Yp^{(n+1)}p^{(n)}Z}\,(A_R)$
$\dfrac{Xp^{(k)}Y \leq Z}{Xq^{(k)}Y \leq Z}\,(IND_L)$	$\dfrac{X \leq Yq^{(k)}Z}{X \leq Yp^{(k)}Z}\,(IND_R)$
$q \leq p$ si k est pair, et $p \leq q$ si k est impair	

Analyses avec composition partielle et majoritaire. Les règles ci-dessous procèdent par couples de mots (leurs types sont séparés par une virgule) ; ainsi l'analyse syntaxique produit aussi un arbre binaire de mots.

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

  $$\Gamma, \mathbf{X}\mathbf{X}', \mathbf{Y}'\mathbf{Y}, \Delta = \Gamma, \mathbf{X}p_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)}\mathbf{Y}, \Delta \overset{C}{\longrightarrow} \Gamma, \mathbf{X}\mathbf{Y}, \Delta$$

  si ($p_i \leq_{Pr} q_i$ et n_i est pair) ou si ($q_i \leq_{Pr} p_i$ et n_i est impair), pour $1 \leq i \leq k$.

- [$\overset{@}{\longrightarrow}$] **(composition partielle majoritaire)** : si (de plus)

  largeur du résultat $\leq$ largeur maximum des arguments (de XX' ou $Y'Y$)

où la *largeur* d'un type $p_1^{u_1}...p_n^{u_n}$ vaut n

2.2 Prégroupe étendu avec des types itérés

Le formalisme de base des prégroupes ne permet pas de définir naturellement des types correspondant aux arguments optionnels ou itérés, tels que les compléments optionnels ou les modifieurs adverbiaux des verbes. (**?**) étend pour cela les prégroupes avec deux constructions : les types optionnels et les types itérés.

Pour des types itérés p^*, l'analyseur est aussi basé sur des règles de composition partielle :

- [C] **(composition partielle)** :
  pour $\mathbf{X}'\mathbf{Y}' \leq \mathbf{Z}'$,
  avec Z' vide (comme 1) ou $a^{*(2k+1)}$:

  $$\boxed{\Gamma, X\mathbf{X}', \mathbf{Y}'Y, \Delta \overset{C}{\longrightarrow} \Gamma, X\mathbf{Z}'Y, \Delta}$$

- [$\overset{@}{\longrightarrow}$] **(composition partielle majoritaire)** : si (de plus)

  largeur du résultat $\leq$ largeur maximum des arguments

Example 1 *Voici un extrait d'une phrase extraite de "Un amour de Swann" de M. Proust:* Quand il l'avait ramenée chez elle, il fallait qu'il entrât. *La figure 1 montre une "preuve" de la correction de l'affectation des types pour ce fragment. Les types primitifs utilisés ici sont :* π_3 *et* $\overline{\pi_3}$ *= troisième personne (sujet) avec* $\pi_3 \leq \overline{\pi_3}$, p_2 *= participe passé,* ω *= complément d'objet direct,* s *= phrase,* s_5 *= subordonnée, avec* $s_5 \leq s$, σ *= phrase subordonnée complète,* τ *= groupe adverbial. La grammaire assigne le type* s *à la phrase comme le montre la figure* **??**.

Avec plus de détails, cette grammaire assigne σ *à "qu'il entrât", car :*

$$\dfrac{\dfrac{\dfrac{\sigma = \sigma^{(0)} \leq \sigma^{(0)}}{\sigma s_5^l s_5 = \sigma^{(0)} s_5^{(-1)} s_5^{(0)} \leq \sigma^{(0)}}\,(A_L)}{\sigma s_5^l \pi_3 \pi_3{}^r s_5 = \sigma^{(0)} s_5^{(-1)} \pi_3{}^{(0)} \pi_3{}^{(1)} s_5^{(0)} \leq \sigma^{(0)}}\,(A_L)}{\sigma s_5^l \pi_3 \overline{\pi_3}^r s_5 = \sigma^{(0)} s_5^{(-1)} \pi_3{}^{(0)} \overline{\pi_3}^{(1)} s_5^{(0)} \leq \sigma^{(0)}}\,(IND_L)$$

qui est affiché sur la figure comme :

qu' *il* *entrât*

$\sigma\, s_5^l \quad \pi_3 \quad \overline{\pi_3}^r s_5$

3 Analyse avec la composition majoritaire

Un algorithme à la Cocke-Younger-Kasami (CYK) a été développé pour les prégroupes. La

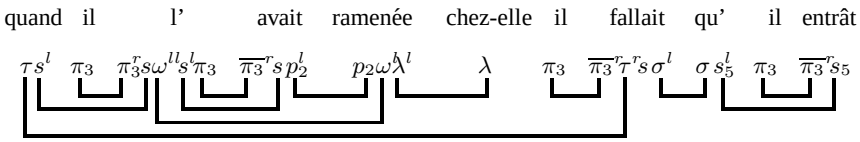
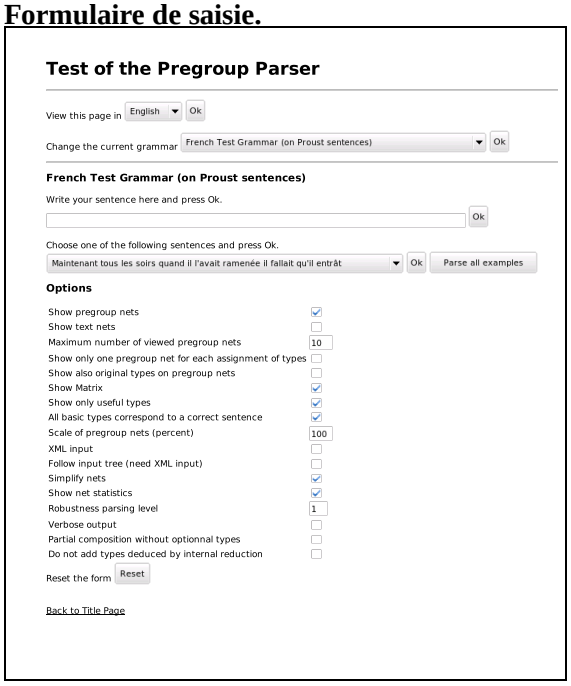


Figure 1: Un exemple d'analyse

granularité de cet algorithme est le mot (ou les entrées lexicales lorsque des expressions figées sont repérées comme dans *pomme de terre*). L'analyseur utilise une approche tabulaire et la programmation dynamique. Actuellement, cet analyseur peut fournir toutes les analyses compatibles avec la théorie des prégroupes (avec types simples itérés et optionnels). Toutefois, avec des grammaires à large couverture, même si la complexité de la partie utilisant la composition partielle est polynomiale par rapport à l'entrée, le nombre total d'analyses est exponentiel. Aussi, le parseur peut se limiter à ne produire qu'un nombre limité de résultats et dans ce cas, la complexité totale du parseur est polynomial. L'analyseur comporte plusieurs phases dont les caractéristiques sont présentées ci-dessous.

Formulaire de saisie.



Ce formulaire permet de sélectionner une grammaire et une phrase ou un fichier (la phrase peut être choisie dans une liste d'exemples associée à la grammaire). Ce formulaire permet aussi de voir la grammaire et le fichier d'exemples et éventuellement de les modifier à la main en utilisant l'éditeur intégré. Le formulaire de saisie permet aussi de choisir certaines options comme le format de sortie, le nombre maximal d'analyses calculées, si l'entrée est une chaîne brute ou bien un arbre XML et, dans ce cas, si l'analyseur doit suivre cet arbre, etc.

Chargement du lexique.

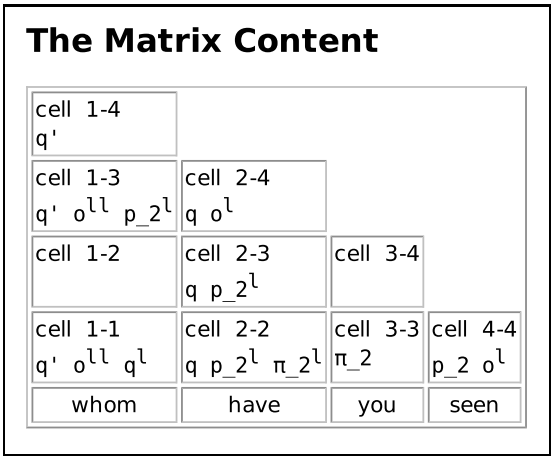
```
<?xml version="1.0" encoding="UTF-8"?>
<grammar>
 <pregroup>
  <order inf="n" sup="n-bar"/>
...
 </pregroup>
 <sentence type="s"/>
 <lexicon>
  <w><word>whom</word>
     <type><simple atom="q'"/>
           <simple atom="o" exponent="-2"/>
           <simple atom="q" exponent="-1"/>
     </type>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

Les grammaires sont décrites par des fichiers XML. Une grammaire contient un ordre partiel sur les types de base, un ensemble de types de base qui sont considérés comme type des phrases correctes et un lexique qui associe à chaque entrée (usuellement un mot ou une suite de mots) un ensemble de types. Il est aussi possible de décrire certaines entrées par des expressions rationnelles ce qui est très utile pour introduire une classe pour les nombres, pour les noms propres (certains noms propres sont inconnus du lexique mais commencent par une majuscule) ou pour les dates. Pour augmenter l'efficacité de cette étape qui peut être très longue si le lexique est très gros (Lefff 2.5.5 (**?**) comporte 534753 entrées – le lexique XML basé sur Lefff est un fichier de 31 367 146 octets), un format compressé du lexique est utilisable ainsi qu'un version utilisant une base de donnée SQLite. Avec une table indexée, même un très gros lexique est accessible très rapidement (moins d'une seconde plutôt que plusieurs dizaines de secondes). En fait, l'analyseur ne charge pas tout le lexique mais uniquement la partie qui correspond à la chaîne en entrée.

Affectation des types aux mots avec réductions internes. Cette étape assigne à chacune des unités en entrée (ou des groupes contigus d'unités) un ensemble de types du prégroupe de la grammaire courante. En fait, comme l'analyseur

utilise la composition majoritaire, une phase de complétion des types utilisant le principe des réductions internes est nécessaire si la grammaire n'a pas été, au préalable, complétée (voir (**?**)). En général cette étape ne fait rien car les types présents dans les grammaires sont rarement réductibles.

Composition majoritaire des séquences d'entrée.



Cette étape est le coeur de l'analyseur. Elle calcule la matrice de l'analyse en utilisant le principe de la composition majoritaire (plutôt que les règles de production des formes normales de Chomsky des grammaires algébriques avec l'algorithme CYK). Bien sûr, comme nous voulons aussi savoir pourquoi une phrase est correcte, la matrice forme plutôt un graphe acyclique orienté complexe. Cette matrice peut être affichée en cas de nécessité.

Calcul des réseaux. La présentation d'une analyse repose sur la notion de réseau qui, à la manière des grammaires de dépendances, montre les liaisons entre les mots plutôt que la hiérarchie des constituants. La figure **??** présente un réseau correspondant à une variante de la phrase de M. Proust. Les types associés aux mots sont présentés sous les mots. Les liens qui unissent les types simples et qui représentent des contractions généralisées dans la théorie des prégroupes sont représentés sous les types. L'introduction de types simples itérés (**?**) autorise qu'un même type simple soit connecté à plusieurs autres types simples contrairement à la théorie de base des prégroupes comme le montre l'exemple de la figure **??**.

Simplification des réseaux. Cette étape simplifie les réseaux en supprimant les types itérés et optionnels qui ne sont pas utilisés et supprime les coupures (voir la section suivante) présentes dans les types ce qui permet l'introduction d'une forme limitée de liens non-projectifs.

Affichage des résultats. Finalement, les résultats sont présentés en utilisant plusieurs formats. Il existe actuellement trois formats : un format HTML utilisable par la version web de l'analyseur avec un affichage des réseaux sous la forme d'une image ou bien d'un texte, un format purement textuel pour l'utilisation sur une console et, finalement, un format XML permettant l'interconnexion de l'analyseur avec d'autres composants logiciels.

4 Introduction de liens non-projectifs dans l'analyse

Les grammaires de prégroupes même avec les types simples optionnels et itérés sont algébriques. En principe, les liens entres les types simples sont projectifs (ils ne se croisent pas). Il est parfois nécessaire d'utiliser des constructions sortant de ce cadre restrictif pour permettre qu'un type simple puisse se lier avec un autre type simple en coupant d'autres liens. Ce type de construction est possible avec cet analyseur. Il utilise la notion de *coupure* qui est une forme restreinte d'interprétation sémantique un peu dans l'esprit des grammaires catégorielles abstraites (**?**). Sur les types comportant une coupure, deux liens vont pouvoir être fusionnés pour donner un lien distant pouvant croiser d'autres liens. Par exemple, les clitiques du français sont placés avant le verbe : dans "il la mange", "la" se trouve entre "il" et "mange". Dans ce cas, il n'y a plus de lien direct entre "il" et "mange" car "la" est un modifieur du verbe dans les prégroupes. Pour retrouver ce lien entre le pronom sujet et le verbe, une coupure sur le type de "la" va permettre de fusionner les deux liens (correspondant à la fonction du sujet) qui vont de "il" à "la" et de "la" à "mange". Cela se voit très bien sur l'exemple de la phrase de Proust dont nous voyons le réseau initial (figure **??** – sans interprétation des coupures) et le réseau sans les coupures (figure **??**). Sur cette modélisation (dans l'esprit des grammaires de dépendances de Mel'chuk), le sujet et les circonstants se rattachent à l'auxiliaire et les autres arguments (principalement les compléments d'objets directs, indirects ou seconds) au participe passé. Actuellement, les coupures sont placées sur les types associés

aux mots comme des annotations qui fonctionnent par paire de types simples. Sur l'extrait de la phrase de Proust, le type original du clitique "l'" est $\pi_3^r s o^{ll} s^l \pi_3$. Deux coupures sont introduites sur ce type que nous représentons chacune par des "$[\cdots]$". Cela donne $[\pi_3^r [s o^{ll} s^l] \pi_3]$. La coupure interne $[s \cdots s^l]$ demande de transformer l'axiome qui sort de s et l'axiome qui sort de s^l par un axiome non-projectif (cela correspond bien à l'élimination d'une coupure logique entre deux axiomes en logique classique surtout sur l'on pense aux réseaux de preuve de la logique linéaire). La coupure externe $[\pi_3^r \cdots \pi_3]$ fait la même chose pour les axiomes sortant de π_3^r et π_3.

Comme ce traitement a lieu sur les réseaux et n'est pas inclus au parseur tabulaire, l'expressivité de ces grammaires (les langages algébriques sans mot vide) n'est pas changée.

5 Outils pour la construction de grammaire

Une partie des modules de la chaîne de traitement concerne la construction des grammaires XML, pour indiquer quels types de prégroupe sont associés à quels mots. On obtient en effet un analyseur en fournissant l'une de ces grammaires à l'analyseur générique de prégroupe.

Nous avons développé pour cela des programmes en xslt (un langage qui permet notamment de transformer des documents XML en d'autres documents XML) ; à la suite d'expériences antérieures orientées d'avantage vers l'apprentissage, (**?**) à partir du Corpus arboré de Paris7, un mode spécifique (en xslt) a été élaboré dans la chaîne de traitement pour le Corpus arboré de Paris7. Ce mode ne couvre cependant actuellement qu'un fragment. Cette approche avec corpus arboré devrait donner des moyens de construction de grammaire par apprentissage et des moyens de validation.

Pour une large couverture du Français, nous avons utilisé plus récemment une autre ressource : Lefff (**?**) ; nous proposons un lien entre ce lexique et des types en prégroupes, à travers ce que nous appelons des "macro-types" qui regroupent des types de prégroupes en classes (voir le diagramme suivant).

Ces macro-types (milieu-haut et partie droite des diagrammes) permettent une classification pouvant rester proche d'un étiquetage d'origine (comme Lefff). Cela permet aussi de maintenir et tester plusieurs systèmes d'affectation de types catégoriels : nous avons ainsi écrit plusieurs modes d'affectation (variantes d' éléments `<define>`, pour les mêmes macro-types) menant à plusieurs maquettes de grammaires de prégroupes (au centre du diagramme) : un mode dans le style applicatif des grammaires de Lambek, et un autre mode proche des grammaires de dépendance, incluant ou non les types itérés.

Un autre aspect dans la construction de grammaire est aussi proposé (dans les modules XML2CTX, LIS2XML) : il s'agit d'une interface avec Camelis/Glis (http://www.irisa.fr/LIS/ferre/camelis/index.html) une implementation des systèmes d'Information logique (LIS (**?**)). L'intégration dans un tel logiciel permet la navigation et l'interrogation dans un lexique, selon des requêtes écrites dans une certaine logique de propriétés des objets : ici les objets sont des mots ou expressions et leurs propriétés sont notamment les types catégoriels (de prégroupe), en outre ceci apporte un meilleur contrôle de l'écriture des types. Sur les diagrammes, cette interface est proposée principalement pour les lexiques, en format source ou cible (voir partie gauche), mais aussi pour les macro-types (à droite). En pratique, une grammaire XML est transformée (module XML2CTX) en contexte LIS ou inversement un contexte LIS sur de tels objets est transformé en XML (module LIS2XML), dans le format d'entrée du méta-analyseur.

Camelis/Glis a ainsi été utilisé pour plusieurs prototypes de langues (anglais, français, breton, bambara), soit pour la définition, soit pour la mise à jour de parties de grammaires, ou encore pour la visualisation et le contrôle.

6 Conclusion

L'analyseur générique des prégroupes implémente le principe de composition partielle majoritaire. Ce programme, qui peut être utilisé à travers un serveur Web PHP ou en ligne de commande, utilise des fichiers XML décrivant une grammaire de prégroupe. En option, une base de données indexée peut être utilisée pour accélérer la recherche dans le lexique. Lorsqu'une phrase est fournie à l'analyseur générique appliqué à une grammaire, le résultat est soit une page HTML, soit une page de texte représentant l'analyse syntaxique sous

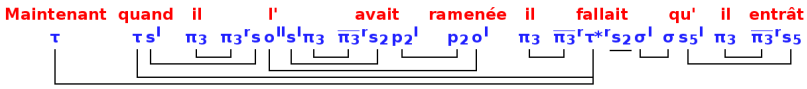


Figure 2: Réseau initial sans interprétation des coupures

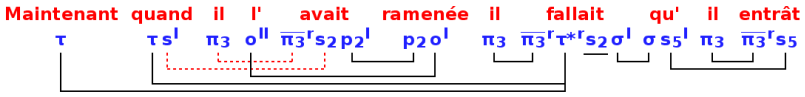
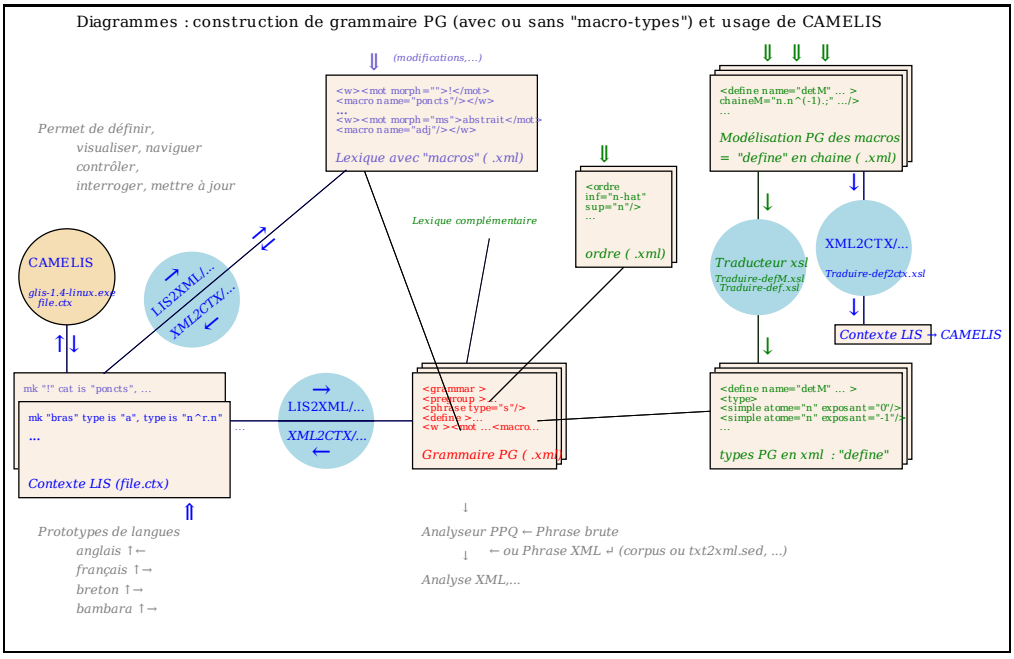
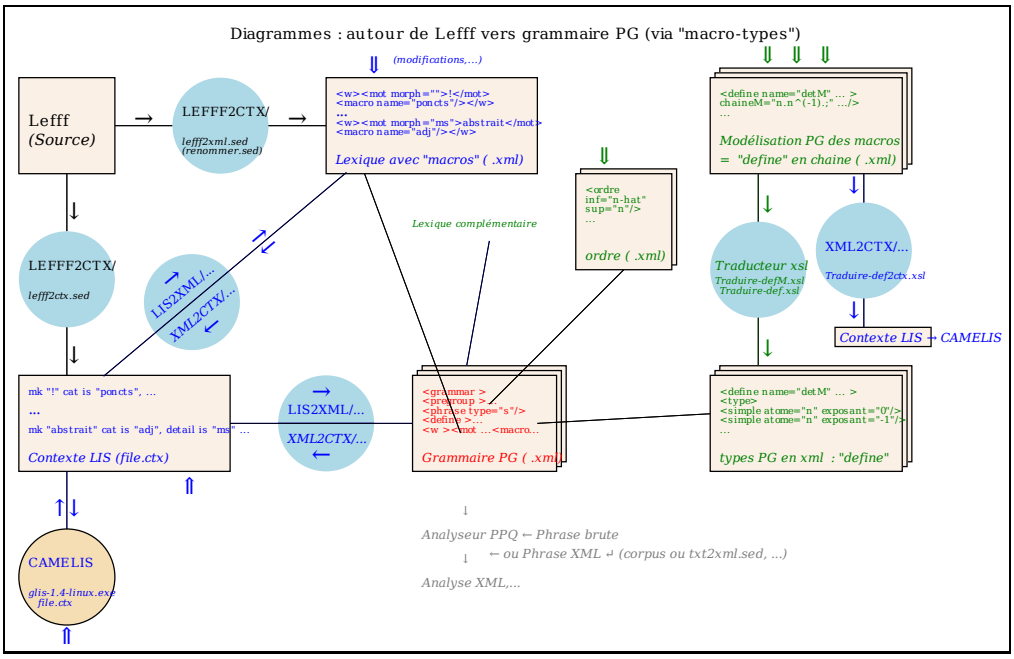


Figure 3: Réseau avec interprétation des coupures





forme de réseau de prégroupe, dans un format approprié à la lecture. En ligne de commande, le programme peut aussi produire un résultat en XML, en format approprié pour une chaine de traitement utilisant les analyses produites. Ce modèle et cet outil permettent aussi actuellement, une forme d'interprétation sémantique, bien que limitée, à l'aide des réductions des "coupures".

Un autre groupe de modules concerne la construction des grammaires XML de prégroupe et une interface avec Camelis/Glis (une grammaire lexicalisée étant vue comme un système d'information logique) facilitant la création et la mise à jour de telles grammaires, leur visualisation et leur contrôle selon certaines propriétés. Ces modules ont été utilisés pour plusieurs prototypes de grammaires (voir schéma), une partie est proposée aussi en lien avec Lefff (voir schéma correspondant).

Enfin, cet ensemble de programmes est à considérer comme une plateforme de test plutôt qu'un logiciel terminé ; il permet actuellement de couvrir largement le Français, mais avec un typage de prégroupe non affiné. Des minigrammaires ont aussi été implémentées et utilisées avec l'analyseur générique pour l'anglais, le breton (celtique) et le bambara (africain).

Bibliographie

D. Bargelli and J. Lambek. 2001. An algebraic approach to french sentence structure. In 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.

Denis Béchet, Alexander Dikovsky, Annie Foret, and Emmanuelle Garel. 2008. 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), pages 88–100. Springer.

Denis Béchet. 2007. Parsing pregroup grammars and Lambek calculus using partial composition. *Studia logica*, 87(2/3).

Wojciech Buszkowski. 2001. Cut elimination for the lambek calculus of adjoints. In V.M. Abrusci and C. Casadio, editors, *New Perspectives in Logic and Formal Linguisitics, Proceedings Vth ROMA Workshop*. Bulzoni Editore.

K. Cardinal. 2002. An algebraic study of Japanese grammar. Master's thesis, McGill University, Montreal.

C. Casadio and J. Lambek. 2001. An algebraic analysis of clitic pronouns in italian. In 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.

Philippe de Groote. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155.

Sylvain Degeilh and Anne Preller. 2005. Efficiency of pregroup and the french noun phrase. *Journal of Language, Logic and Information*, 14(4):423–444.

Kosta Došen. 1999. *Cut Elimination in Categories*. Kluwer Academic publishers.

S. Ferré and O. Ridoux. 2004. An introduction to logical information systems. *Information Processing & Management*, 3(40):383–419.

A. Foret. 2007. Pregroup calculus as a logical functor. In *Proceedings of WOLLIC 2007*, volume LNCS 4576. Springer.

J. Lambek and A. Preller. 2003. An algebraic approach to the german noun phrase. *Linguistic Analysis*, 31:3–4.

Joachim Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65.

J. Lambek. 1999. 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.

J. Lambek. 2000. Type grammar meets german word order. *Theoretical Linguistics*, 26:19–30.

Richard Oehrle. 2004. A parsing algorithm for pregroup grammars. In M. Moortgat and V. Prince, editors, *Proc. of Intern. Conf. on Categorial Grammars*, Montpellier.

Eric Poupard, Denis Bechet, and Annie Foret. 2006. Categorial grammar acquisition from a french treebank. In *Actes de la Conférence d'APprentissage 2006 (CAP'06)*. (Poster).

Mehrnoosh Sadrzadeh. 2007. Pregroup analysis of persian sentences.

Benoit Sagot, Lionel Clément, Éric Villemonte de la Clergerie, and Pierre Boullier. 2006. The lefff 2 syntactic lexicon for french: architecture, acquisition. In *LREC'06*.