

# La chaîne d'analyse syntaxique de LEOPAR

**Guy Perrier**                      **Bruno Guillaume**                      **Jonathan Marchand**  
LORIA Nancy Université   LORIA INRIA Nancy Grand-Est   LORIA Nancy Université  
Guy.Perrier@loria.fr                      Bruno.Guillaume@loria.fr                      Jonathan.Marchand@loria.fr

## Résumé

LEOPAR est un analyseur syntaxique fondé sur le formalisme des grammaires d'interaction, intégré dans une chaîne d'outils visant avant tout à la précision et la couverture linguistique. À l'aide de XMG, il est possible de construire des grammaires de façon semi-automatique à partir de connaissances linguistiques. Ces grammaires sont ensuite ancrées à l'aide de lexiques indépendants d'un quelconque formalisme. Les grammaires ancrées sont enfin utilisées par LEOPAR pour analyser des corpus bruts et produire des sorties sous forme d'arbres syntagmatiques ou de structures de dépendances.

## 1 Introduction

Nous nous plaçons du point de vue de chercheurs visant à développer des grammaires formelles pour des systèmes de TAL à partir de connaissances linguistiques. Comme chacun le sait, la construction de telles grammaires est une tâche extrêmement complexe et qui prend énormément de temps.

Lorsque l'on cherche à formaliser des connaissances linguistiques, la première question qui se pose est celle du choix du cadre formel. Actuellement, il n'y a aucun consensus pour un cadre qui prévaudrait sur tous les autres. Parmi les plus courants (TAG, HPSG, LFG, CCG ...), chacun a ses points forts et ses points faibles. De notre point de vue, un bon formalisme doit avoir trois propriétés difficiles à concilier : il doit être suffisamment expressif pour représenter les généralisations linguistiques, facilement lisible par des linguistes et traitable automatiquement. Guidés par ces principes, nous plaçons pour un formalisme introduit récemment, les Grammaires d'Interaction (GI) (Perrier, 2000; Perrier, 2003; Guillaume

and Perrier, 2009). Les GI visent à synthétiser deux idées exprimées jusqu'à maintenant par deux types de formalismes différents : la composition syntaxique vue comme un mécanisme de saturation de ressources, idée centrale des Grammaires Catégorielles (GC) (Retoré, 2000), et les grammaires vues comme des systèmes de contraintes, idée à la source des grammaires d'unification comme LFG (Bresnan, 2001) ou HPSG (Pollard and Sag, 1994).

Les chercheurs qui développent des lexiques et des grammaires à large couverture à partir de connaissances linguistiques sont confrontés à la contradiction entre la nécessité de choisir un cadre formel particulier et le coût requis par le développement des ressources. Prenons par exemple un des systèmes les plus avancés dédiés à cette tâche : LKB (Copestake, 2001). LKB permet de développer des grammaires et des lexiques pour différentes langues mais seulement dans le cadre de HPSG, ou au mieux de grammaires fondées sur des structures de traits typées. C'est pourquoi, les ressources utilisées sont difficilement réutilisables pour d'autres cadres formels. Dans la conception de notre chaîne de traitement, nous avons ce souci en tête et nous avons essayé de rendre réutilisable tout ce qui pouvait l'être. C'est le sens notamment de la déconnection totale du lexique du formalisme grammatical. C'est aussi le sens d'une organisation modulaire de la grammaire en une hiérarchie de classes.

La chaîne de traitement de LEOPAR est décrite figure 1.

- Pour construire des grammaires, nous utilisons XMG (Section 3.1), qui compile une *grammaire source*, écrite par un humain, en une *grammaire objet* utilisable par un système de TAL.
- Les grammaires développées sont lexicalisées, si bien que les grammaires objets produites à l'étape précédente doivent encore

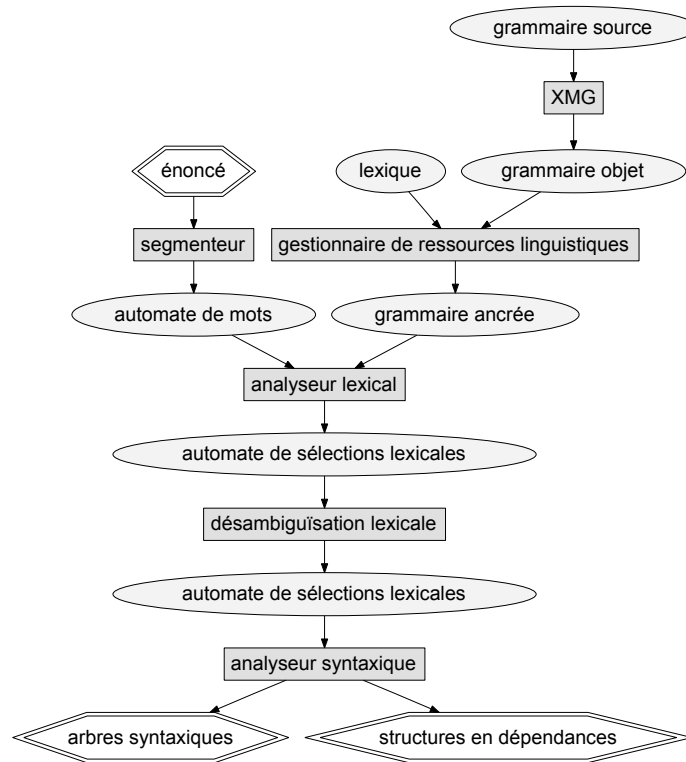


FIG. 1 – Architecture de la chaîne de traitement

être ancrées dans un lexique (Section 3.2) pour devenir des *grammaires ancrées*.

- Ensuite, l’analyse d’une phrase commence par une segmentation en mots, qui produit un automate de mots. Un analyseur lexical remplace dans l’automate les arcs portant des mots par des arcs portant les entrées correspondantes de la grammaire ancrée : un arc est remplacé par autant d’arcs qu’il y a d’entrées dans la grammaire pour le mot concerné. Puis un module de désambiguïsation lexicale (Section 3.3) produit un nouvel automate dans lequel certains choix lexicaux (dont on est sûr qu’ils ne participeront pas à une analyse qui réussit) sont supprimés.
- L’automate est fourni à l’analyseur LEOPAR (Section 3.4). Il produit un ensemble de solutions sous deux formes : des arbres syntagmatiques ou des structures de dépendances (Section 3.5).

## 2 Les Grammaires d’Interaction

Les GIs (Guillaume and Perrier, 2009) sont un formalisme grammatical fondé sur la notion de polarité. Cette notion permet d’exprimer la capa-

cité d’une structure syntaxique à interagir avec d’autres et de faire la distinction entre structure saturée et structure non saturée. La composition syntaxique est guidée par les polarités et vise à saturer les structures syntaxiques à travers leur superposition.

Plus précisément, une structure syntaxique est un arbre sous-spécifié décoré de polarités, présenté comme un ensemble de contraintes sous la forme d’une *description d’arbre polarisée*. L’opération élémentaire qui permet de superposer partiellement deux arbres syntaxiques sous-spécifiés est la fusion de deux nœuds visant à saturer deux de leurs polarités. Le processus s’achève avec succès lorsque l’on obtient un arbre syntaxique saturé et complètement spécifié.

Dans les GCs, les TAGs ou les grammaires de dépendances, la composition syntaxique peut aussi être vue comme un mécanisme de saturation de polarités, même si ce mécanisme n’est pas explicite. Mais elle est beaucoup moins expressive dans la mesure où elle est localisée à certains nœuds très spécifiques.

Dans les GIs, la superposition d’arbres est beaucoup plus souple et permet d’exprimer des

contraintes d'environnement sophistiquées. En cela, les GIs s'apparentent aux Grammaires d'Unification (HPSG, LFG ...), car la superposition d'arbres peut être vue comme une forme d'unification, avec une particularité essentielle : les polarités servent à contrôler cette unification.

### 3 Description de la chaîne de traitement

#### 3.1 Le compilateur de grammaires XMG

Le premier module dans la chaîne de traitement est XMG<sup>1</sup> (Duchier et al., 2004). Il s'agit d'un outil de développement de grammaires à large couverture, pour lesquelles le maintien de la cohérence est une tâche particulièrement difficile. XMG est fondé sur la distinction entre *grammaire source* et *grammaire objet*. La première est écrite par un humain alors que la seconde est utilisable dans un système de TAL.

XMG fournit un langage de haut niveau pour écrire des grammaires sources et un compilateur destiné à les traduire en grammaires objets. Dans le langage de haut niveau, les grammaires sources se présentent sous forme de modules qui sont combinés par conjonction ou disjonction. Ces modules peuvent être structurés en plusieurs dimensions (syntaxique et sémantique par exemple), et il est possible d'établir des liens entre les différentes dimensions.

La figure 2 montre comment quelques classes de la grammaire sont organisées en utilisant l'héritage multiple (conjonction de classes par le nœud AND) ou la disjonction (par le nœud OR).

XMG est particulièrement adapté au développement de grammaires lexicalisées. Les structures syntaxiques élémentaires de ces grammaires sont attachées à des mots et rassemblées dans un lexique. Bien entendu, il n'est pas question de construire ces items lexicaux un à un manuellement, d'autant plus qu'ils partagent plusieurs sous-structures. Par exemple, les items relatifs aux verbes possèdent une sous-structure exprimant l'accord du verbe avec son sujet. Si un linguiste veut introduire des changements dans cet accord verbe-sujet, il n'est pas question qu'il reprenne chaque item lexical séparément. L'approche modulaire qui est au centre de XMG permet d'isoler ce phénomène dans un module particulier de la grammaire source. Pour effectuer des changements dans l'accord verbe-sujet, il suffit de

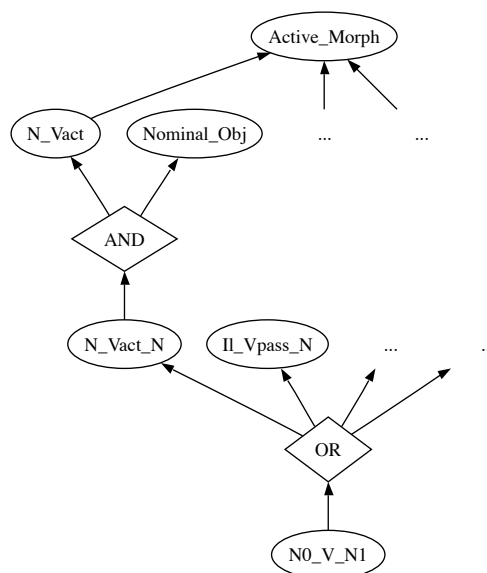


FIG. 2 – Héritage de classes avec XMG

modifier le module concerné puis de compiler à nouveau la grammaire objet.

L'outil XMG fournit des facilités pour construire et déboguer les grammaires. Il est notamment possible de compiler n'importe quel module séparément. Un outil graphique permet d'explorer la hiérarchie des modules.

XMG a été aussi utilisé pour développer une TAG du français (Crabbé, 2005) et il pourrait être facilement étendu pour permettre de construire des grammaires dans d'autres formalismes fondés sur les arbres.

#### 3.2 Ancrage de la grammaire objet dans le lexique

XMG produit une grammaire objet sous forme d'un ensemble d'arbres élémentaires sous-spécifiés. Ces arbres ne sont pas ancrés mais chaque arbre est produit avec une *interface*. Une interface est une structure de traits à deux niveaux qui décrit les contraintes morphologiques et syntaxiques utilisées pour sélectionner les mots ancrant l'arbre correspondant.

De façon indépendante, un lexique regroupe toutes les formes fléchies des mots de la langue, avec pour chacune d'elles une interface décrivant ses propriétés morphologiques et syntaxiques.

Le format des interfaces des mots du lexique est le même que celui des interfaces des arbres élémentaires de la grammaire, si bien qu'un arbre sélectionne un mot lorsque leurs interfaces s'unifient. Le module d'ancrage fournit donc en sortie

<sup>1</sup>XMG est librement disponible sous licence CeCILL <http://sourcesup.cru.fr/xmg>

un lexique d'arbres élémentaires ancrés.

Ce fonctionnement de l'ancrage est assez souple pour, d'une part, permettre d'écrire le lexique des mots grammaticaux en parallèle de la grammaire et, d'autre part, d'utiliser tout ou partie de lexiques existants pour les classes lexicales ouvertes (verbes, noms, adjectifs et adverbes). Par exemple, le lexique actuellement utilisé avec la grammaire du français utilise, pour les classes ouvertes, les lexiques *Lefff* (Sagot et al., 2006), *Morphalou* (Romary et al., 2004) et *ABU*<sup>2</sup>.

### 3.3 Désambiguïisation lexicale

La saturation de polarités est le principe qui guide l'analyse syntaxique dans la mesure où elle contrôle la composition syntaxique. Le même principe peut être utilisé pour la désambiguïisation lexicale. Pour une phrase d'entrée, ce que nous appelons *sélection lexicale* est le choix d'un arbre élémentaire ancré de la grammaire pour chaque mot de la phrase.

Le nombre de sélections lexicales possibles est en général une fonction exponentielle de la longueur de la phrase. Une façon de les filtrer est d'abstraire le formalisme initial  $F$  en un formalisme  $F_{abs}$  en oubliant une partie de l'information contenue dans les arbres élémentaires. L'analyse syntaxique avec le formalisme abstrait  $F_{abs}$  permet ensuite d'éliminer des sélections lexicales incorrectes à un coût minimal (Boullier, 2003).

#### 3.3.1 Désambiguïisation lexicale avec les polarités

(Bonfante et al., 2004) montre que l'on peut utiliser pour cela les polarités. Des arbres élémentaires, on conserve seulement l'ensemble des polarités positives et négatives et on oublie notamment la structuration en arbre. L'analyse syntaxique avec le formalisme abstrait revient alors à un simple comptage de polarités : la saturation de l'arbre syntaxique final impose que le nombre de polarités positives soit égal au nombre de polarités négatives.

Une façon particulièrement compacte de la faire est d'utiliser un automate d'états finis (Bonfante et al., 2004). Chaque état de l'automate exprime un bilan des polarités positives et négatives en un point de lecture de la phrase après sélection d'arbres élémentaires pour les premiers mots. Une transition correspond à la lecture d'un mot et la

sélection d'un arbre élémentaire correspondant. Il y a un seul état acceptant, celui où la phrase est totalement lue et le bilan de polarités est équilibré.

#### 3.3.2 Désambiguïisation lexicale avec les dépendances

Plus récemment, une nouvelle méthode de désambiguïisation (Bonfante et al., 2009) a été mise au point. Cette méthode utilise l'interprétation des polarités en terme de dépendances, les structures insaturées attendent pour interagir et créer une dépendance. Chaque polarité insaturée induit donc une contrainte qui doit être résolue par une autre description ; on calcule donc sur la grammaire objet (avant ancrage), pour chaque polarité insaturée, l'ensemble des descriptions candidates pour saturer cette polarité (on appelle de telles descriptions des compagnons). Ces informations ne dépendent que de la grammaire et peuvent donc être pré-calculées ; ensuite pour chaque phrase, on retire les sélections lexicales pour lesquelles une des descriptions n'a pas de compagnon dans la sélection. Dans (Bonfante et al., 2009), il est montré que cette façon de faire de la désambiguïisation était équivalente à utiliser un formalisme  $F_{abs}$  pour lequel les langages sont réguliers.

### 3.4 Analyse syntaxique

Le module suivant dans la chaîne de traitement est un analyseur syntaxique fondé sur le formalisme des GIs<sup>3</sup>.

En entrée de l'analyse d'une phrase, nous avons un automate dont les arcs sont décorés par des arbres élémentaires ancrés. Actuellement, on a le choix entre deux stratégies d'analyse : une stratégie de type *shift-reduce* et une stratégie de type *CYK*. Décrivons brièvement la première qui est la plus utilisée. L'automate est parcouru depuis l'état initial. Chaque pas consiste soit à franchir une transition dans l'automate en ajoutant l'arbre élémentaire correspondant (*shift*), soit à effectuer une réduction (*reduce*). Une réduction est déclenchée lorsque le nombre de polarités positives et négatives accumulées dépasse un seuil donné au départ. Elle consiste à fusionner deux nœuds pour neutraliser deux de leurs polarités. Bien entendu, en théorie, une telle stratégie est incomplète, mais en pratique, si l'on fixe une borne raisonnable,

<sup>2</sup><http://abu.cnam.fr/DICO/>

<sup>3</sup>LEOPAR est librement disponible sous licence CeCILL à <http://www.loria.fr/equipes/calligramme/leopar>

n'importe quelle phrase grammaticalement correcte d'un corpus peut être analysée.

### 3.5 Format des analyses

Lors de l'analyse syntaxique d'un énoncé, l'analyseur syntaxique construit pas à pas un arbre syntaxique saturé et complètement spécifié. Il s'agit de l'arbre syntagmatique de l'énoncé.

Durant cette analyse, c'est la saturation de polarités qui guide la superposition des arbres élémentaires qui construisent cet arbre. Ces saturations permettent également de déduire une structure en dépendances. En effet, les polarités expriment les interactions entre les mots, la saturation de ces polarités vient concrétiser ces interactions : chaque saturation de polarités donne lieu à une relation de dépendances entre les mots dont les arbres élémentaires portent ces polarités (Marchand et al., 2009).

Plus précisément, lors de la saturation de polarités de deux nœuds, ces polarités vont déterminer le type de la dépendance (s'il s'agit d'une relation tête-complément/tête-spécifieur ou d'une relation modifieur-modifié) ainsi que le gouverneur et le dépendant de cette relation. Les étiquettes de ces dépendances sont déduites des informations morpho-syntaxiques des nœuds contenant les polarités. Lors de l'analyse syntaxique d'un énoncé, l'ensemble des relations de dépendances issues de ces saturations forme le graphe de dépendances associé à cet énoncé.

### 3.6 LEOPAR

En plus d'une interface sous forme de ligne de commande, l'analyseur offre une interface graphique de dialogue avec l'utilisateur. Cette interface permet non seulement de piloter l'analyse mais aussi de visualiser les ressources lexicales et grammaticales, ce qui est très utile pour le débogage. Sous une forme ou sous une autre, l'interface utilisateur fournit différents paramètres qui permettent de personnaliser l'utilisation de l'analyseur.

L'analyse peut être effectuée de façon totalement automatique ou selon un mode manuel. Dans ce cas, c'est l'utilisateur qui effectue la sélection lexicale et qui ensuite choisit, à chaque étape de l'analyse, les nœuds à fusionner. Une interface permet de visualiser l'état d'analyse à chaque étape et de revenir éventuellement en arrière pour la reprendre différemment.

## 4 Exemple

Nous donnons ici pour la phrase (1) des illustrations de différents points évoqués dans la section précédente :

(1) « *L'entreprise dans laquelle il travaille ferme.* »

La figure 3 montre les résultats obtenus par les différentes méthodes de désambiguïsation lexicale. Avec plus de 180 millions de sélections au départ, on en garde finalement, en utilisant les deux méthodes, seulement 16 sur lesquelles le parsing profond doit être effectué. Pour chacune des deux méthodes, on dispose d'approximations plus rapides que les versions complètes ; on peut ainsi appliquer les approximations des deux méthodes puis les méthodes complètes sur le résultat c'est pour cela que le temps de calcul avec les deux méthodes (0,23s) est inférieur au temps de chacune d'elle.

Les deux formats de sortie en arbre syntagmatique et en structure de dépendances sont illustrés par les figures 4 et 5.

L'arbre de la figure 4 décrit la structure en constituants de la phrase. Cette structure utilise des nœuds vides (les feuilles qui ne contiennent pas de mots et qui sont représentés avec un fond blanc) comme trace pour les constituants qui ne sont pas en position canonique.

La structure en dépendances de la figure 4 montre le type de dépendances que l'on obtient. Ces dépendances ne sont pas tout à fait celles qui sont décrites habituellement. Ceci est dû au fait qu'elles suivent de près les interactions syntaxiques qui donnent un rôle important aux mots grammaticaux. La dépendance  *sujet*  entre « *ferme* » et « *entreprise* » donne lieu à deux dépendances dans notre représentation et c'est la composition de ces deux relations qui redonne la dépendance attendue. La même remarque s'applique à la dépendance  *mod*  entre « *travaille* » et « *laquelle* » qui passe par la préposition « *dans* ».

## 5 Résultats

Notre chaîne de traitement a été utilisée tout d'abord pour produire une GI du français à large couverture. La plupart des constructions grammaticales du français sont couvertes, et parmi elles, un certain nombre qui sont non triviales : coordination, extraction avec "pied piping" et barrières, négation (en français, la négation est en général

Méthode de désambiguïsation	Nombre de sélections lexicales	temps
sans désambiguïsation	181 984 320	0s
polarité	1 666	0,44s
dépendances	200	2,85s
polarité + dépendances	16	0.23s

FIG. 3 – Désambiguïsation pour la phrase « *L'entreprise dans laquelle il travaille ferme.* »

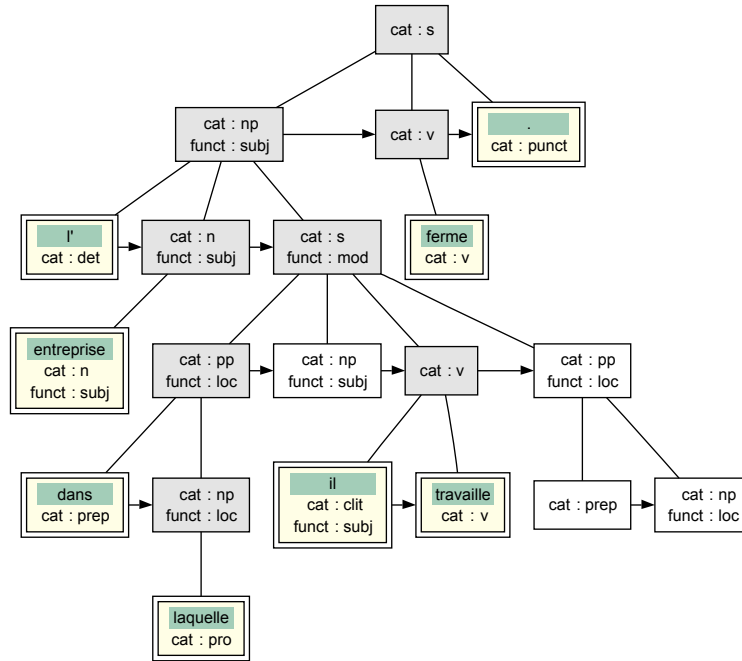


FIG. 4 – Arbre d'analyse pour la phrase « *L'entreprise dans laquelle il travaille ferme.* »

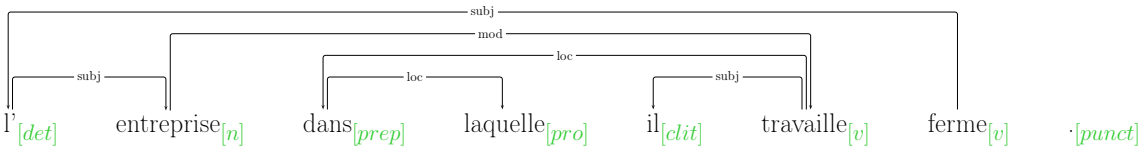


FIG. 5 – Structure de dépendances pour la phrase « *L'entreprise dans laquelle il travaille ferme.* »

exprimée par deux mots dont le positionnement est parfois complexe) ... La grammaire objet, dans son état actuel, contient 2380 arbres élémentaires sous-spécifiés non ancrés issus de 373 modules de la grammaire source.

La grammaire du français a été testée sur la TSNLP du français (Test Suite for Natural Language Processing) (Lehmann et al., 1996). Cette suite contient environ 1300 phrases grammaticales et 1600 phrases non grammaticales. Le fait que notre grammaire soit fondée sur des connaissances linguistiques lui assure une bonne précision et limite la surgénération : 88% des phrases grammaticales sont analysées correctement et 85% des phrases non grammaticales sont rejetées par notre

grammaire.

Une ébauche de grammaire de l'anglais a été effectuée à partir d'une abstraction de la grammaire du français et en y ajoutant ensuite les spécificités de l'anglais. Cette tâche a été effectuée en à peine deux mois parce que nous avons pu exploiter la modularité de la grammaire source du français. La grammaire a été testée sur la TSNLP de l'anglais : 85% des phrases grammaticales ont été correctement analysées et 84% des non grammaticales ont été rejetées.

## 6 Participation à la campagne d'évaluation Passage

Pour tester notre analyseur et nos grammaires, il est impératif d'analyser de grands volumes de corpus. C'est dans cette optique que l'analyseur LEOPAR participe actuellement à la campagne d'évaluation des systèmes d'analyse syntaxique du français Passage<sup>4</sup>. De façon évidente, deux obstacles majeurs rendent l'analyseur actuel peu adapté à cette campagne :

- LEOPAR n'est pas conçu pour faire de l'analyse robuste, il ne fournit donc aucune information partielle pour des phrases qui ne sont pas grammaticales au sens de notre grammaire.
- LEOPAR n'utilise pas de statistiques, il fait une recherche exhaustive parmi les nombreuses ambiguïtés possibles pour une phrase d'un corpus ; les méthodes de désambiguïsation développées permettent d'analyser des phrases d'environ 20 mots mais nous ne pouvons pas pour l'instant analyser les phrases plus longues.

Néanmoins, il nous semble évident que cette participation est un premier jalon qui nous permettra de mesurer à l'avenir les évolutions de nos outils par rapport à ces deux aspects.

## 7 Perspectives

La chaîne de traitement que nous avons présentée ici permet de produire des grammaires et des lexiques à large couverture à partir de connaissances linguistiques. Cela justifie le fait que nous ayons écarté toute méthode statistique dans une première étape : dans les deux étapes de la désambiguïsation lexicale et de l'analyse syntaxique, nous voulons garder toutes les solutions possibles, même les moins probables.

Maintenant, dans un futur proche, nous avons l'ambition de pouvoir analyser de larges corpus bruts. Pour le français, nous disposons d'une grammaire et d'un lexique à large couverture, qui sont essentiels pour une telle tâche. L'introduction de statistiques dans les modules de désambiguïsation et d'analyse permettra d'accroître l'efficacité des calculs. Par ailleurs, nous devons prendre en compte la robustesse dans nos stratégies d'analyse. Nous avons aussi l'ambition d'intégrer la sémantique dans le formalisme.

<sup>4</sup><http://atoll.inria.fr/passage/eval2.fr.html>

Notre expérience avec l'anglais est un premier pas vers la prise en compte de la multilingualité. Le point crucial est de faire évoluer nos grammaires vers une architecture dans laquelle on puisse distinguer un noyau multilingue de ses instantiations vers des langues particulières comme le fait (Ranta, 2004) avec le "Grammatical Framework".

Enfin, pour ouvrir la chaîne de traitement à d'autres formalismes, il est nécessaire tout d'abord d'étendre XMG vers davantage de généralité, ce qui ne paraît pas difficile. Par ailleurs, il faut pouvoir polariser les formalismes les plus courants afin de leur appliquer nos méthodes de désambiguïsation et d'analyse fondées sur les polarités. (Kahane, 2006) propose une polarisation de la plupart de ces formalismes dont nous pouvons nous inspirer et (Kow, 2007) en utilisant une polarisation des TAGs montre que la voie est prometteuse.

## Bibliographie

- G. Bonfante, B. Guillaume, and G. Perrier. 2004. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *CoLing'2004, 2004*, pages 303–309, Geneva, Switzerland.
- G. Bonfante, B. Guillaume, and M. Morey. 2009. Dependency constraints for lexical disambiguation. In *proceedings of IWPT 09*, Paris, France.
- P. Boullier. 2003. Supertagging : A non-statistical parsing-based approach. In *IWPT 03*, pages 55–65, Nancy, France.
- J. Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford.
- A. Copestake. 2001. *Implementing Typed Feature Structure Grammars*. CSLI Publications.
- B. Crabbé. 2005. *Représentation informatique de grammaires fortement lexicalisées : application à la grammaire d'arbres adjoints*. Thèse d'université, Université Nancy 2.
- D. Duchier, J. Le Roux, and Y. Parmentier. 2004. The metagrammar compiler : A NLP Application with a Multi-paradigm Architecture. In *Second International Mozart/Oz Conference - MOZ 2004, Charleroi, Belgium*.
- B. Guillaume and G. Perrier. 2009. Interaction Grammars. *Research on Language and Computation*. A paraître. Un rapport préliminaire est disponible à l'URL <http://www.loria.fr/~guillaum/publications/RR-6621.pdf>.
- S. Kahane. 2006. Polarized unification grammars. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Sydney, Australia.

- E. Kow. 2007. *Surface realisation : ambiguity and determinism*. Thèse d'université, Université Nancy 2.
- S. Lehmann, S. Oepen, S. Regnier-Pros, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion, J. Baur, L. Balkan, and D. Arnold. 1996. TSNLP — Test Suites for Natural Language Processing. In *CoLing 1996, Copenhagen*.
- J. Marchand, B. Guillaume, and G. Perrier. 2009. Analyse en dépendances à l'aide des grammaires d'interaction. In *Actes de TALN 09*, Senlis, France, June. poster.
- G. Perrier. 2000. Interaction grammars. In *CoLing '2000, Sarrebrücken*, pages 600–606.
- G. Perrier. 2003. *Les grammaires d'interaction*. Mémoire d'habilitation à diriger des recherches, Université Nancy 2.
- C.J. Pollard and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.
- A. Ranta. 2004. Grammatical Framework : A Type-Theoretical Grammar Formalism. *Journal of Functional Programming*, 14(2) :145–189.
- C. Retoré. 2000. The Logic of Categorical Grammars. ESSLI'2000, Birmingham.
- L. Romary, S. Salmon-Alt, and G. Francopoulo. 2004. Standards going concrete : from lmf to morphalou. In *Workshop on Electronic Dictionaries, Coling 2004, Geneva, Switzerland*.
- B. Sagot, L. Clément, E. de La Clergerie, and P. Boulrier. 2006. The lefff 2 syntactic lexicon for french : architecture, acquisition, use. In *LREC 06, Genova, Italy*.