Une plate-forme de conception et d'exploitation d'une grammaire d'arbres adjoints lexicalisés

Benoît Crabbé, Bertrand Gaiffe et Azim Roussanaly

LORIA
BP 239 – 54506 Vandoeuvre-lès-Nancy Cedex
{crabbe,gaiffe,azim}@loria.fr

Résumé – Abstract

Dans cet article, nous présentons un ensemble d'outils de conception et d'exploitation pour des grammaires d'arbres adjoints lexicalisés. Ces outils s'appuient sur une représentation XML des ressources (lexique et grammaire). Dans notre représentation, à chaque arbre de la grammaire est associé un *hypertag* décrivant les phénomènes linguistiques qu'il recouvre. De ce fait, la liaison avec le lexique se trouve plus compactée et devient plus aisée à maintenir.

Enfin, un analyseur permet de valider les grammaires et les lexiques ainsi conçus aussi bien de façon interactive que différée sur des corpus.

Keywords – Mots Clés

Grammaire, analyse syntaxique, ressources lexicales, LTAG, représentation compacte du lexique
Grammar, parser, lexical resources, LTAG, compact lexical representation

1 Introduction

La grammaire d'arbres adjoints lexicalisée est une grammaire fortement lexicalisée. La gestion du lexique de ce type de grammaire est *a priori* difficile. Les unités grammaticales élémentaires sont en effet très nombreuses. Les développements récents en matière de représentation compacte du lexique permettent désormais d'en faciliter la gestion, et en particulier la maintenance.

Dans cet article, nous présentons un ensemble d'outils effectivement implémentés qui permettent la gestion et l'utilisation d'une grammaire d'arbres adjoints lexicalisée. Nous centrons notre attention sur les aspects technique liés à la représentation du lexique. L'implémentation que nous proposons facilite, de notre point de vue, la maintenance de la grammaire. Nous commençons par rappeler brièvement l'architecture donnée au lexique dans les implémentations du formalisme (architecture XTAG). Nous montrons en quoi ce type

d'architecture pose des problèmes de maintenance et de réutilisation (liés à la notion d'ancrage). Enfin, nous discutons les choix retenus dans l'implémentation

2 Grammaire d'arbres adjoints (TAG)

La grammaire d'arbres adjoints (Joshi 1975, Joshi et Schabès 1997) est un système de composition d'arbres dont les unités sont des arbres élémentaires. Deux opérations de composition sont définies sur les arbres : l'adjonction et la substitution.

TAG est essentiellement utilisé dans sa version lexicalisée (LTAG). La condition de lexicalisation impose que tout arbre élémentaire soit ancré par une unité lexicale. En suivant cette approche, le lexique d'une grammaire TAG lexicalisée est un ensemble d'arbres élémentaires, comme illustré à la Figure 1[1].

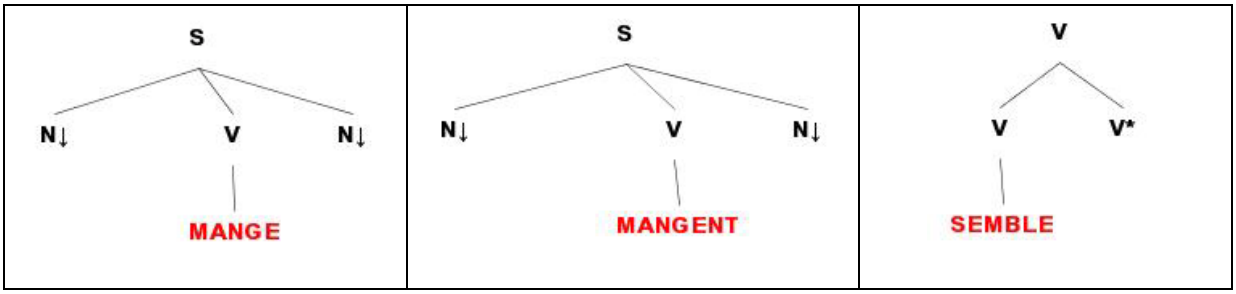

Figure 1 : Exemples d'arbres élémentaires

Si théoriquement un lexique TAG se compose d'un ensemble très important d'arbres élémentaires, en pratique les implémentations (Xtag 2001, Abeillé 2002) introduisent une factorisation : les unités lexicales sont stockées sur trois niveaux :

- Une base de *schèmes* (ou *templates*) ou arbres élémentaires non instanciés. Un schème contient un nœud particulier, marqué par le symbole ◊ appelé *nœud ancre*, nœud sur lequel l'unité lexicale se place lors du processus d'ancrage. De plus, les schèmes sont classés en *familles*. Une famille est un ensemble d'arbres dont chaque élément représente la même structure prédicat-argument.

- Une base de *lemmes*, chaque lemme est associé à une ou plusieurs familles.

- Une base morphologique dans laquelle chaque forme fléchie est associée à un lemme.

Les arbres élémentaires sont construits dynamiquement lors du processus d'analyse syntaxique. Le processus de construction dynamique est appelé **ancrage**. Le principe est le suivant : la forme fléchie détermine le lemme correspondant. Ce lemme permet de sélectionner la famille. La forme fléchie est ensuite *greffée* aux nœuds ancres de chacun des schèmes composant la famille sélectionnée. Les traits associés à la forme fléchie, au lemme et

[1] Les arbres utilisés dans cet article sont inspirés de la grammaire proposée par (Abeillé 2002). De plus, nous utilisons une variante usuelle de TAG : Feature-Based TAG (Vijay-Shanker 1987), version dans laquelle chaque nœud d'un arbre est associé à une structure de trait *top* et une structure de traits *bottom*.

au nœud ancre sont alors unifiés. Il est donc possible que l'ancrage échoue suite à un échec d'unification.

2.1 Problématique de l'opération d'ancrage

Cette problématique est liée à la construction des structures de traits dans les arbre élémentaires. Elle est illustrée à la Figure 2.

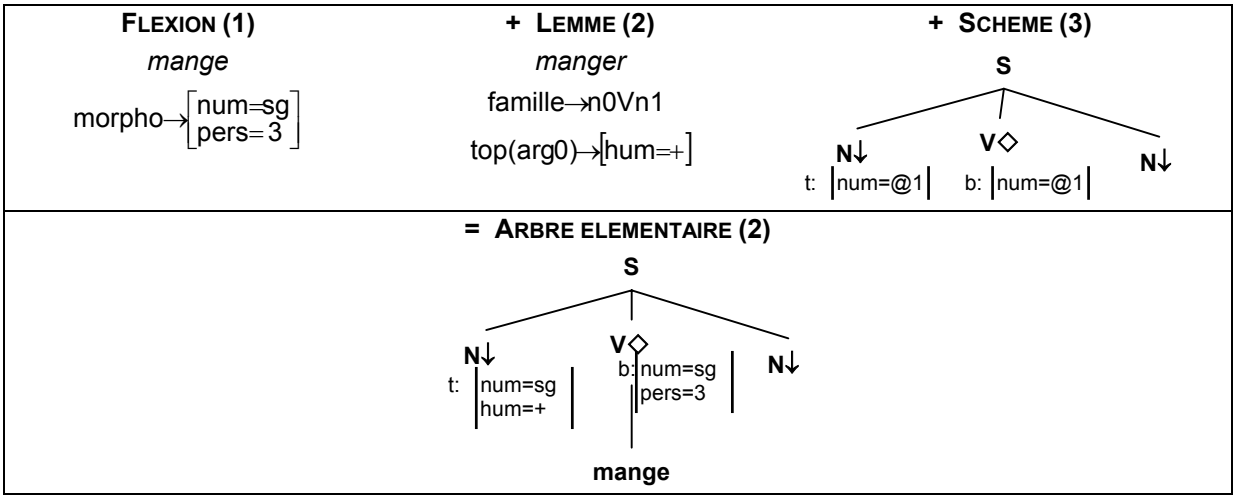

Figure 2 : Exemple d'ancrage

L'ancrage effectif d'une forme fléchie dans un schème fait intervenir des structures de traits issues de trois composants :

1. la structure de traits morphologiques associée à la forme fléchie,
2. éventuellement, la structure de traits associée au lemme (dans notre exemple, il n'y en a pas).
3. un ensemble d'équations associées au lemme (équations d'ancrage) qui représentent des restrictions de sélections à projeter dans l'arbre.

Les structures de traits associées à la flexion et au lemme sont unifiées avec la structure de traits associée au nœud ancre. Leurs rôles respectifs sont prototypiquement le traitement de phénomènes d'accord et la mise en échec du processus d'ancrage avec certains schèmes (par échec d'unification). Ainsi, le lemme d'un verbe transitif de famille n_0Vn_1 non passivable comme *comporter* est associé à la structure de trait `[passif = -]`. Les schèmes correspondant aux formes passives contiennent quant à eux le trait `[passif = +]` sur le nœud ancre. On voit donc qu'un arbre associé à une forme fléchie n'est pas sélectionné uniquement sur la base de la famille, mais également par l'intermédiaire de traits liés à l'ancrage, placés à la fois dans la base de lemmes et dans les schèmes.

D'autre part, la valeur de certains traits est ambiguë. Ainsi un trait utilisé dans la grammaire pour traiter la concordance des temps tel `[mode = indicatif]` a également pour rôle de bloquer l'ancrage d'un schème représentant une forme impérative.

Les équations d'ancrage sont quant à elles associées aux lemmes. Leur particularité est de projeter des traits sur des nœuds de schèmes différents de l'ancre. Vu qu'un lemme est associé

à une famille, c'est-à-dire un ensemble de structures arborescentes topologiquement hétérogènes, il faut être capable de désigner des nœuds de façon symbolique pour l'ensemble des schèmes appartenant à cette famille. Les familles XTAG (FTAG) regroupent des arbres qui représentent les différentes réalisations d'une même structure prédicative ; dans ce cadre, les nœuds ciblés par les équations d'ancrage sont les nœuds représentant la position des arguments dans la structure arborescente. Ceux-ci sont repérés à l'aide d'indices représentant la position de l'argument dans la structure argumentale. Ainsi, la représentation XTAG (FTAG) des arguments du prédicat dans la famille {α,β}, illustré par la Figure 3, se fait par l'association au lemme *manger* de l'équation d'ancrage suivante : `top(arg0)` → `[hum=+]`.

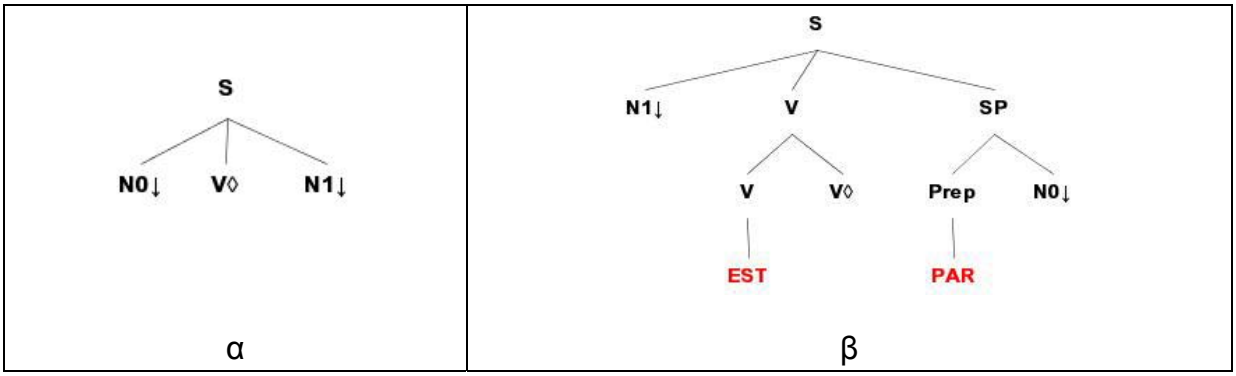

Figure 3 : Exemple de famille d'arbres

Ce qui signifie qu'une famille se définit non seulement par un ensemble d'arbres {α,β} mais également par un ensemble de correspondances :

$$\begin{cases} arg0 \rightarrow \alpha.1, \ \beta.3.2 \\ arg1 \rightarrow \alpha.3, \beta.1 \end{cases}$$

qui associe aux arguments 0 et 1 les adresses de Gorn sur lesquelles ils se projettent pour chacun des schèmes appartenant à la famille.

Une famille est donc plus qu'un ensemble d'arbres, elle définit également un ensemble de correspondances entre des désignations symboliques de nœuds et des adresses de Gorn reflétant la position de ces noms symboliques dans chaque arbre de la famille.

3 Vision alternative du processus d'ancrage

Une architecture de type XTAG suggère un découpage de la grammaire en termes de familles organisées autour de l'idée prédicat-argument. Bien que cette approche reflète une idée communément admise en syntaxe, nous voudrions montrer comment une organisation alternative du lexique fondée sur la notion *d'alternances lexicales* (Levin 1993) change la donne.

Le travail de (Levin 1993) suggère une organisation du lexique articulée autour de l'hypothèse du *déterminisme sémantique* : en syntaxe, cela signifie que les contextes syntaxiques (*alternances*) dans lesquels apparaît un mot sont déterminés par ses propriétés sémantiques. L'hypothèse stipule, par exemple, que tous les verbes de mouvement (*classe d'alternance*) partagent les mêmes propriétés syntaxiques. Dans ce cadre (Levin 1993) répertorie pour un

très grand nombre de verbes anglais les variations de diathèse effectivement attestées dans la langue (*alternances*) : ex. passif-long, actif et sujet nominal…).

Une adaptation au français du travail de Levin faite par (St Dizier 1996, St Dizier 1999) suggère d'associer un lexème à un ensemble d'alternances. En suivant cette approche, on associera au verbe *comporter* une alternance telle que : *actif et sujet nominal et objet nominal* mais on se garde de lui associer une alternance telle que *passif-long* comme l'illustrent les phrases : (*Le livre comporte deux chapitres* | * *Deux chapitres sont comportés par le livre*)

On remarque immédiatement que l'utilisation d'une paire de traits telle que `[passif = ±]` pour opérer un filtrage lors de l'ancrage des arbres n'a aucun intérêt dans ce cadre.

4 Représentation de la grammaire

4.1 Caractéristiques

Trois opérations apparaissent clairement pour définir les liens entre lemmes et schèmes :

- découpage de la grammaire en sous-ensembles caractérisés par des propriétés linguistiques communes,
- ancrage des lemmes sur les nœuds ancres des schèmes,
- projection de contraintes sémantiques associées aux lemmes dans les schèmes sélectionnés à l'aide des équations d'ancrage.

En vue de réaliser ces opérations de manière satisfaisante, nous proposons une représentation de la grammaire qui présente les avantages suivants :

- conciliation des différentes approches liées à la question de l'ancrage,
- gain d'efficacité de l'analyseur grâce à l'intégration d'un mécanisme permettant de filtrer l'ensemble des schèmes candidats à un ancrage pour éviter un échec d'unification plus tardif,
- amélioration de l'indépendance entre la base de lemmes et la base de schèmes ; cette propriété permet de dissocier la construction de la base de schèmes (en ayant recours par exemple à un compilateur de métagrammaire) de celle de la base de lemmes (en s'appuyant par exemple sur une extraction à partir d'un dictionnaire),
- utilisation de la technologie à base de XML facilitant ainsi le partage des ressources et le développement des outils.

4.2 Représentation axée sur les *hypertags*

Notre approche dans ce domaine rejoint celle de (Kinyon 2000). Il s'agit d'associer à chaque schème une structure de traits, appelée *hypertag*. Les hypertags ont pour but la description des phénomènes linguistiques encodés dans le schème. L'exemple de la Figure 4 montre, pour chaque schème, les structures de traits décrivant la voix de la phrase ainsi que les fonctions initiales, les fonctions finales des arguments.

Une telle structure sert de base au filtrage lors du processus d'ancrage dans la mesure où elle permet d'atteindre des ensembles d'arbres plus fins que les familles XTAG.

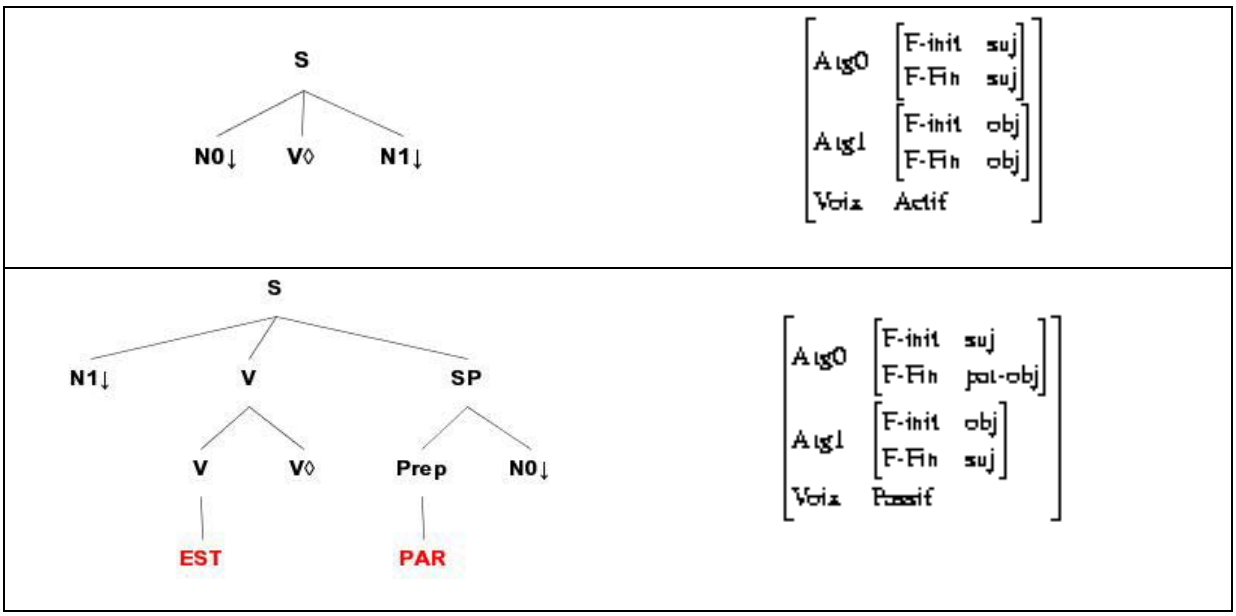

Figure 4 : Exemple d'*hypertags*

En suivant cette approche, si on associe au lemme *comporter* la structure de traits de la Figure 5 correspondant à la famille n_0Vn_1 , on sélectionne par succès d'unification les deux arbres de la Figure 4.

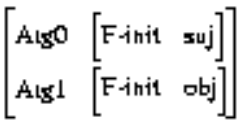

Figure 5 : Structure de traits n_0Vn_1

Si on associe au lemme un hypertag correspondant à l'alternance *actif et sujet nominal et objet nominal* (Figure 6), on ne lui associe plus que l'arbre de l'actif (Figure 4) :

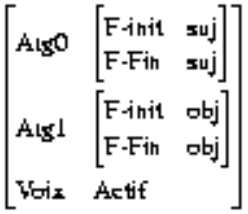

Figure 6 : Structure de traits *alternance actif et sujet nominal et objet nominal*

Notons que les hypertags présentent d'autres intérêts dans le cadre de la description des schèmes, notamment dans le domaine de la gestion d'une grammaire : extraction de sous-grammaires, interfaces de description du lexique,... (Kinyon 2000).

4.3 Mécanisme de correspondance

Nous avons besoin d'établir des correspondances entre nœuds dans des ensembles d'arbres filtrés par des hypertags. Le mécanisme de nommage de nœuds de XTAG doit donc être généralisé. La solution retenue est de prendre acte de la nécessité explicite d'un nommage des nœuds[2]. A titre d'illustration, le nommage présenté dans la Figure 7 permet de résoudre l'équation d'ancrage `top(#dest)` → `location=+`.

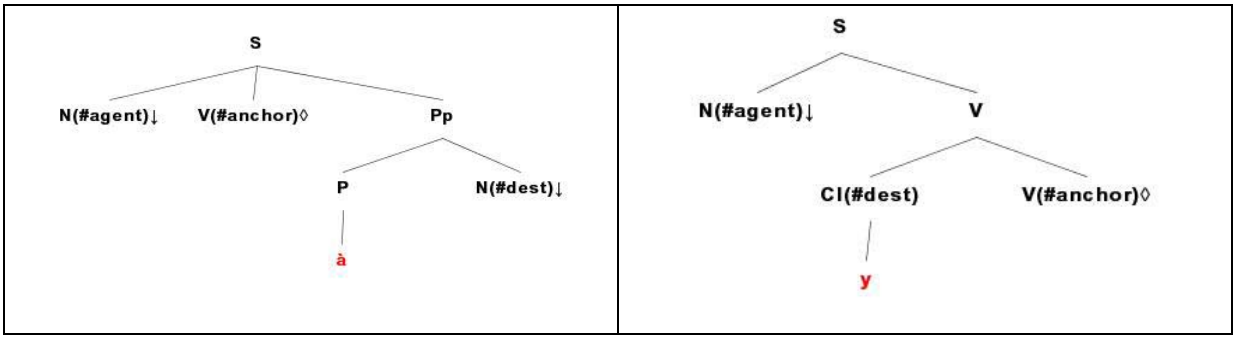

Figure 7 : Exemple de nommage de nœuds

Il est à noter que ce mécanisme est suffisamment général pour traiter les cas d'ancrages multiples dans un même schème. Les noms symboliques des nœuds vont servir à effectuer les correspondances relativement aux arbres d'une famille dans les éléments de représentation.

4.4 Structure de la représentation

La conséquence des principes de représentation à base d'hypertags et de nommage des correspondances que nous avons adoptés, est que le lexique est décrit par les éléments et les liens suivants:

- forme fléchie : cet élément permet de définir des liens entre une entrée de la base morphologique et, d'une part, le lemme auquel elle fait référence et, d'autre part, une structure de traits destinée à encoder les traits morphologiques,

- lexicalisation : cet élément permet essentiellement (1) de définir le lien entre un lemme et une structure de traits destinée à filtrer l'ensemble de schèmes dans lequel le lemme est ancré. Il contient également (2) une association avec le nom du noeud ancre. Il contient enfin (3) une association avec un doublet constitué d'un nom de nœud et d'une structure de traits destinée à effectuer l'opération de projection de la structure de traits dans les schèmes.

```
Exemple :(verbe, comporter)  → (1)    [Arg0 [F-init=suj]
                                       Arg1 [F-init=obj]]
                             → (2)    #anchor
                             → (3)    (#n0, [top [hum=+]])
```

[2] Ce nommage est réalisé via le compilateur de métagrammaire (*infra*)

5 Logiciels

La plate-forme que nous proposons, comporte deux applications centrales : le compilateur de métagrammaire et l'analyseur syntaxique. Autour de ces deux applications, nous fournissons un ensemble d'outils permettant d'éditer, de visualiser graphiquement et de tracer les différentes ressources utilisées dans ces applications.

Tous ces logiciels partagent des structures de données encodées en XML. Le format que nous avons retenu est TAGML2 (définie par une DTD) car il offre un cadre normalisé à la représentation des lexiques et des grammaires TAG. Tous ces logiciels sont disponibles sur le site de l'équipe L&D du LORIA (`www.loria.fr/equipes/led/outils.html`)

5.1 Compilateur de métagrammaire

Pour obtenir les associations <*schème, structure de trait, correspondance*> nécessaires à notre proposition, nous avons développé un compilateur de métagrammaires (Gaiffe et al. 2002), fortement inspiré du travail de (Candito 1999). Dans une approche de ce type, le linguiste décrit sa grammaire via des ensemble de hiérarchies de classes. Chaque classe comprend des contraintes topologiques devant être vérifiées par les arbres relevant de la classe et une description de la classe par une structure de traits. Les hiérarchies de classes sont combinées entre elles pour enrichir le graphe d'héritage[3]. Enfin, les contraintes accumulées aux feuilles de ce graphe d'héritage sont résolues en l'ensemble des arbres qui les valident.

On obtient en résultat les associations visées ; en effet :

- les structures de traits décorant les classes sont unifiées le long du graphe d'héritage jusqu'aux classes feuilles où elles sont associées aux arbres résultats (ce sont nos hypertags);

- les contraintes topologiques contenues dans les classes sont exprimées sur des noms symboliques des nœuds[4], (exemple *#agent, #dest, #anchor*). et le compilateur fournit pour chaque arbre solution les adresses où chaque nom se projette dans l'arbre. On obtient ainsi les correspondance mentionnées au paragraphe 4.3.

Bien entendu, les résultats sont fournis par ce compilateur de métagrammaires au format TAGML2. Enfin, un outil annexe permet de décrire les associations lemmes, structures de traits de filtrage, équations symboliques et de les traduire en TAGML2.

5.2 Analyseur syntaxique

L'analyseur LTAG du Loria (LLP2) est développée à partir de la version précédente écrite par Lopez (Lopez 1999). Les caractéristiques de cette nouvelle version sont les suivantes :

---

[3] Moyennant des contraintes de compatibilité

[4] Ce sont des constantes

- elle permet le traitement des structures de traits (FB-LTAG) (Vijay-Shanker 1987),
- elle est limitée aux grammaires d'arbres insérés (TIG) (Schabes et Waters 1995), une version restrictive des TAG,
- elle prend en charge des ressources décrite au format TAGML2 ; les schèmes créés par le compilateur de métagrammaires sont, de ce fait, directement exploitables par l'analyseur,
- elle dispose d'une interface graphique interactive permettant de tester une grammaire, et offre également la possibilité d'un fonctionnement en mode lignes de commandes; ceci permet de lancer globalement l'analyse d'un lot de phrases (par exemple, un corpus),
- elle intègre une représentation XML des lots de phrases à analyser en vue :
  - de traiter un corpus annoté par des informations morpho-syntaxiques : l'idée ici est de faciliter l'interfaçage entre l'analyseur et un annotateur morpho-syntaxique de sorte à réduire considérablement la taille des ressources lexicales, notamment celle de la base morphologique,
  - de traiter un corpus incluant des syntagmes partiellement analysés : l'objectif visé est de prétraiter les phrases à analyser en s'appuyant sur les résultats d'un analyseur de surface. Cette opération vise a améliorer les performances de l'analyseur, mais elle permet également d'utiliser l'analyseur pour valider des hypothèses lexicales sur les mots inconnus dans le cadre d'un apprentissage,
  - de traiter certaines ambiguïtés lexicales, il s'agit plus précisément ici du cas particulier de l'occurrence de plusieurs candidats pour un mot ou groupe de mots ; cette aptitude de l'analyseur peut être mise à profit dans deux cas : (1) Le premier cas provient d'un problème de segmentation (tokenisation) des mots composés ; par exemple, dans le cas de l'expression pomme de terre, l'analyse peut se dérouler parallèlement sur les hypothèses de 3 mots distincts ou d'un seul mot composé. (2) Le deuxième cas correspond à l'entrée bruitée d'un système de reconnaissance de la parole où l'ambiguïté est inhérente au dispositif d'acquisition.

Conclusion

L'encodage des ressources lexicales en TAGML2, constitue un socle commun à l'ensemble d'outils présentés dans cet article. Par ailleurs, TAGML2 fournit une représentation normalisée des ressources. La DTD originale a été conçue dans le cadre de l'Action de recherche concertée : *Ressources lexicales pour LTAG* (RLT : `http://atoll.inria.fr/RLT/arc.html`). TAGML2 a été conçue spécifiquement à partir de TAGML1 (Bonhomme et Lopez 2000) dans le but de permettre une représentation plus compacte du lexique LTAG. La mise à disposition de bibliothèques logicielles vise à faciliter, dans le futur, l'intégration des ressources LTAG existants dans divers formats.

Par ailleurs, à court terme, nous projetons d'utiliser cette plate-forme pour constituer des lexiques destinés à être utilisés dans des applications restreintes. Notre but, à plus long terme, est de concevoir une grammaire à plus large couverture. Grammaires que les hypertags

permettront de filtrer pour obtenir des sous-grammaires d'applications spécifiques. Le traitement des équations d'ancrage proposé devrait alors permettre de projeter les restrictions de sélection liées à l'application sur cette grammaire.

Références

Abeillé Anne, (2002), *Une grammaire d'arbres adjoints pour le français*, Editions du CNRS, Paris

Bonhomme P. et Lopez P (2000), TagML : codage XML et ressources pour les grammaires d'arbres adjoints lexicalisés, *LREC 2000* Athènes

Candito Marie-Hélène, (1999), *Organisation Modulaire et Paramétrable de Grammaires Electroniques Lexicalisées,* Thèse de l'Université de Paris 7

Gaiffe B., Crabbé B. et Roussanaly A. (2002), A new metagrammar compiler, in proc *TAG+6* Venise

Joshi Aravind K. and Schabès Yves, (1997), Tree Adjoining Grammars, *Handbook of Formal Languages*, eds G. Rozenberg and A. Salomaa, Berlin, Springer Verlag

Joshi Aravind K., Levy Leon S. and Takahashi M., (1975), Tree Adjunct Grammars, *Journal of the Computer and System Sciences*, vol.10, pp.136-163

Kinyon Alexandra, (2000), Hypertags, *COLING-00* Sarrebrück

Levin Beth, (1993) *English Verb Classes and Alternations*, The University of Chicago Press

Saint-Dizier Patrick, (1999) Predicative Forms in Natural Language and Lexical Knowledge Bases in *Alternations and Verb Semantic Classes for French : Analysis and Class Formation*, ed. Patrick Saint-Dizier, Kluwer, Dordrecht

Saint-Dizier Patrick, (1996), *Verb Semantic Classes in French Version 2*, Tech. Report., IRIT - CNRS",

Schabès Y. et Waters R (1995), Tree Inserted Grammar : a Cubic-time Parsable Formalism that lexicalizes Context Free Grammar without Changing the Trees Produced, *Computational Linguistics* MIT Press 1995.

Vijay-Shanker K, (1987) *A study of Tree Adjoining Grammar*, PhD. Th. University of Pennsylvania, Department of computer and information science

XTAG Research Group (2001) *A Lexicalized Tree Adjoining Grammar for English*, IRCS, University of Pennsylvania, num. IRCS-01-03