Projection et monotonie dans un langage de représentation lexico-grammatical

Benoit Crabbé
LORIA - Université Nancy 2
BP 239, 54506 Vandoeuvre-lès-Nancy Cedex
crabbe@loria.fr

Mots-clefs : Syntaxe, Lexique, Liage, Interface syntaxe sémantique, TAG

Keywords: Syntax, Lexicon, Linking, Syntax Semantics interface, TAG

Résumé Cet article apporte une méthode de développement grammatical pour la réalisation de grammaires d'arbres adjoints (TAG) de taille importante augmentées d'une dimension sémantique. La méthode que nous présentons s'exprime dans un langage informatique de représentation grammatical qui est déclaratif et monotone. Pour arriver au résultat, nous montrons comment tirer parti de la théorie de la projection dans le langage de représentation que nous utilisons. Par conséquent cet article justifie l'utilisation d'un langage monotone pour la représentation lexico-grammaticale.

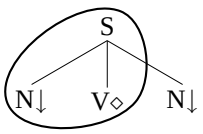
Abstract This papers provides a methodology for the grammatical development of large sized tree adjoining grammars (TAG) augmented with a semantic dimension. The provided methodology is expressed in a monotnic and declarative language designed for the compact representation of grammatical descriptions. To achieve the result, we show how to express a linking theory in the language used. Therefore this paper justifies the use of a monotonic language for lexico-grammatical representation.

1 Introduction

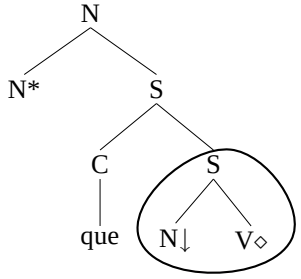
Dans cet article, nous donnons une méthode générale qui permet représenter de manière compacte un fragment significatif de grammaire TAG pour le français comportant une interface syntaxe-sémantique. TAG n'étant pas en tant que tel un formalisme d'analyse syntaxique comportant une spécification explicite de la représentation sémantique, nous utilisons ici la variante introduite par (GK03). Celle-ci utilise des unités élémentaires qui sont des structures à deux dimensions. Une dimension syntaxique qui est l'arbre TAG habituel, et une dimension sémantique qui représente l'information sémantique associée à cet arbre. La dimension sémantique est constituée de formules du langage de représentation sémantique utilisé par (GK03). Celui-ci est de la même famille que la *minimal recursive semantics*.

2 Enjeux méthodologiques

Pour représenter une grammaire TAG de manière non redondante, nous utilisons un langage informatique destiné à exprimer la grammaire de manière compacte (CD04). Celui-ci repose sur l'utilisation de macros (ou classes) réutilisables. Ce langage est destiné à servir de source à un interpréte effectivement implémenté qui réalise l'expansion de la description compacte (DLP04). Le langage utilisé pose que dans un système de description grammatical, il est souhaitable de reconnaître deux types de généralisations : les généralisations de structure d'une part et les généralisations d'alternatives d'autre part.



Jean *mange une pomme*



La pomme que **Jean** *mange*

On identifie dans la description grammaticale un certain nombre de constructions récurrentes que l'on souhaiterait pouvoir réutiliser. Par exemple dans les deux arbres suivants, nous avons encerclé l'information qui représente un sujet nominal réalisé en position canonique. D'autre part, et indépendamment du partage de structure, on souhaite capturer la notion d'alternative. Par exemple ou souhaitera identifier qu'une construction passive est une alternative d'une construction active. Les alternatives que nous identifions ont un statut particulier. En effet, ces généralisations contribuent à décrire des ensembles d'unités grammaticales mises en relation. Dans l'exemple donné ci-dessus, l'alternative entre un contexte transitif actif (a) et un contexte transitif passivisé (b) contribuent à décrire un ensemble de deux arbres qui partagent une sémantique commune (prédicat binaire transitif). La reconnaissance explicite des alternatives constitue un point important du langage que nous utilisons. En effet, une famille d'arbres (Abe02), n'est rien d'autre qu'un ensemble de réalisations alternatives d'une même structure prédicat-argument. Or la représentation des alernatives a souvent été négligée dans les propositions récentes pour la représentation grammaticale de TAG (Can99; Xia01; GCR02)[1].

Rejet des règles lexicales Le cadre que nous proposons ici se veut opérationnel. Dans ce contexte, la méthode que nous proposons se caractérise par le rejet des règles lexicales. La raison principale est que l'utilisation d'une mécanique procédurale dans le développement de grammaires d'arbres adjoints de tailles conséquentes pose en pratique de très sérieux problèmes d'ordonnancement de règles (Pro02). En lieu et place de règles lexicales, nous utilisons un langage purement déclaratif muni de deux opérateurs qui permettent de combiner des descriptions grammaticales fragmentaires : la conjonction et la disjonction.

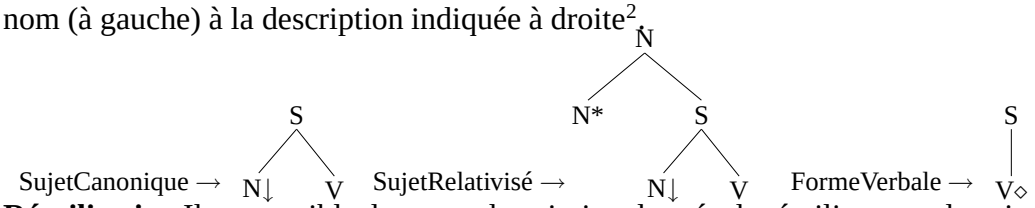
3 Langage utilisé

Le langage que nous utilisons (noté $\mathcal{L}_{\mathcal{C}}$) permet de tirer parti des deux types de généralisations que nous avons identifiés. Celui-ci repose crucialement sur la notion de *classe*. Une classe contient une *description* de structure(s) grammaticale(s) partielle(s). Pour une TAG, une description est une description partielle d'arbre potentiellement augmentée de structures de traits associées

[1] En particulier, mentionnons que (Can99; GCR02) permettent d'exprimer les alternatives de manière détournée à l'aide d'un algorithme de « croisement ». En pratique nous avons observé que cet algorithme complique considérablement le travail de conception de grammaires.

aux noeuds. Une classe a pour fonction de nommer la description D à laquelle elle est associée de telle sorte qu'il est possible de réutiliser D par ailleurs dans une description D' quelconque. Ce sont les classes qui nous permettent dans le langage de capturer les généralisations grammaticales. Ce langage détaillé par (CD04) comporte quatre aspects.

Nommage Le langage permet de nommer une description. Associer un nom à une description permet de la réutiliser par après. Ceci est illustré par les exemples suivants où on associe un nom (à gauche) à la description indiquée à droite[2].



Réutilisation Il est possible dans une description donnée de réutiliser une description déjà associée à un nom par ailleurs. Dans l'exemple suivant, on réutilise ainsi la « macro » FormeVerbale dans la description de VerbeActif.

VerbeActif → *FormeVerbale*

Alternative On peut exprimer dans le langage la notion d'alternative (ou de choix) en utilisant le symbole $\vee$. Ainsi, l'exemple suivant montre comment exprimer que la notion de sujet recouvre aussi bien la notion de sujet canonique que de sujet relativisé.

Sujet → *SujetCanonique* $\vee$ *SujetRelativisé*

Composition Il est également possible de combiner deux descriptions. Ainsi, on peut exprimer qu'une famille de constructions intransitives est faite d'un sujet et d'une forme verbale à l'actif de la manière suivante :

VerbeIntransitif → *Sujet* $\wedge$ *VerbeActif*

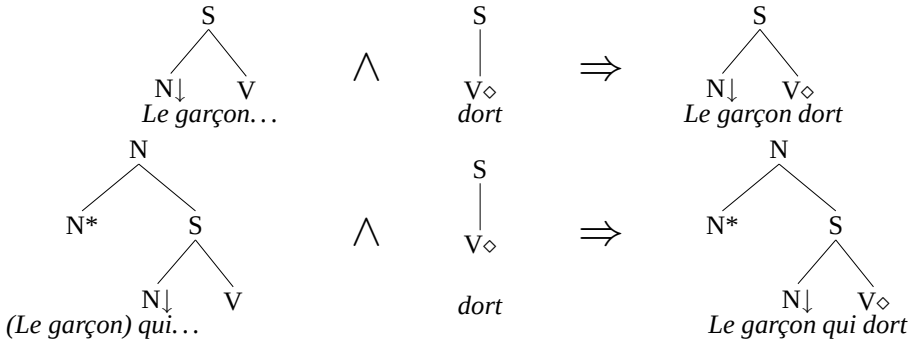


Figure 1: Deux arbres à contexte intransitif

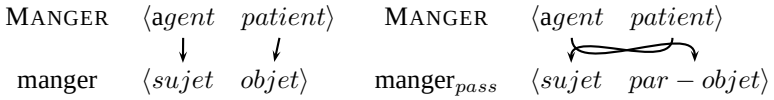
Interprétation du langage Formellement, comme l'indiquent (CD04), l'interprétation d'une description grammaticale d formulée en $\mathcal{L}_\mathcal{C}$ se ramène à un interpréter un programme logique de la famille *Definite Clause Grammar* (DCG). Cela se comprend en considérant que d est une grammaire dont les terminaux sont des descriptions arborescentes, que l'opération de concaténation est ici la conjonction et en imposant que la DCG sous-jacente à d soit non récursive (CD04). L'interprétation d'une description d engendre de manière indéterministe toutes les phrases, prises comme des conjonctions de descriptions, de la DCG sous-jacente à d. Dans l'exemple présenté jusqu'ici, en considérant la classe *VerbeIntransitif* comme axiome de la DCG, l'interprète engendre deux arbres représentant deux contextes de verbe intransitif, comme indiqué en figure 1. Où on illustre sur la gauche les deux conjonctions de descriptions engendrées par l'interprète et sur la droite le résultat de la composition des descriptions[3].

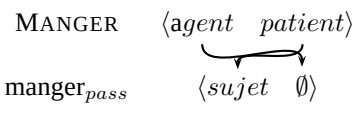
[2]Nous utilisons dans ce document une syntaxe abstraite pour illustrer notre propos. Un interprète concret pour ce langage a été implanté par (DLP04).

[3]Le mécanisme de composition est détaillé par (CD04)

4 Théorie de la projection lexicale

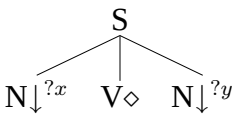
Le langage étant posé, nous donnons dans cette section une méthode générale qui permet d'exprimer de manière déclarative une grammaire à l'aide de ce langage. Cette méthode repose sur la théorie de la projection (*linking theory*). La fonction de projection associe chaque élément de la structure argumentale à un élément de la structure syntaxique, comme indiqué ici :





Approche monotone Utiliser la projection dans un langage tel que celui que nous proposons consiste à considérer que l'on génère un ensemble d'unités grammaticales complètement spécifiées par surspécifiation de la structure argumentale. Cette méthode permet de garantir un mécanisme monotone pour exprimer l'interface syntaxe sémantique. Pour illustrer ce point, prenons le cas classique du passif court qui, dans la littérature est souvent utilisé pour justifier l'utilisation d'une mécanique procédurale (Bec93; Can99; Xia01). Dans ce cas, *Marie est vue* est considéré comme variante de *Jean voit Marie* dans laquelle l'agent n'est pas exprimé. Le traitement par règle lexicale du passif court consiste à effacer l'agent (i.e. *Jean*). Dans le cas du modèle projectif, rien n'est effacé. Au contraire l'agent non exprimé est projeté sur une réalisation vide en syntaxe, comme illustré à gauche. L'approche projective ne fait qu'ajouter de l'information, et éventuellement de l'information vide à la structure argumentale.

Projection pour une grammaire TAG L'approche projective présentée jusqu'à présent s'adapte facilement à une grammaire semi-lexicalisée du type LFG. Ainsi dans une grammaire LFG, une structure syntaxique comme $manger\langle sujet \quad objet\rangle$ correspond directement à la f-structure de l'entrée lexicale d'un verbe.



◇(?x,?y)

En pratique, dans le cas d'une grammaire TAG à dimension sémantique, les entrées lexicales à décrire sont des schémas d'arbres comme illustré à droite. La dimension sémantique est prise comme contrepartie de la structure argumentale et la structure arborescente est prise comme contrepartie de la dimension syntaxique. La fonction de projection est représentée par le partage de valeurs des traits sémantiques associés aux noeuds de l'arbre et les variables d'individus de la prédication élémentaire (indiquée en gras).

La méthodologie que nous proposons repose sur l'idée de base que les arbres élémentaires TAG sont décrits par assemblage de fragments de structure correspondant à la représentation des arguments et du prédicat. La méthodologie qui suit consiste ensuite à produire des familles d'arbres en assemblant ces différents fragments. Ce qui généralise l'exemple donné en figure 1. La méthode comporte quatre étapes : (1) Description des fragments (2) Description de fonctions syntaxiques (3) Description de changements de diathèse (4) Description de familles. Pour réaliser l'interface syntaxe-sémantique, l'idée consiste à établir le lien entre la structure prédicative et les valeurs de traits dans les arbres à l'aide de classes paramétrées[4]

La première étape de la description consiste à définir des fragments représentant différentes

[4]On se rappelle que le langage de contrôle $\mathcal{L}_\mathcal{C}$ est vu comme la contrepartie d'une DCG ce qui nous autorise à utiliser des paramètres.

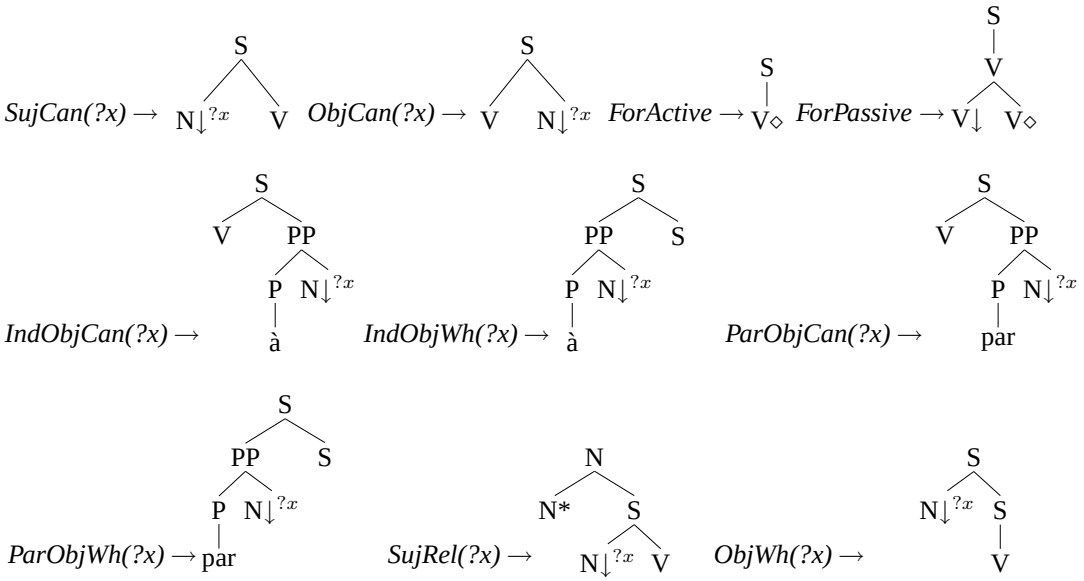


Figure 2: Fragments d'arbres utilisés comme « blocs de construction » de la grammaire

formes du prédicat et des arguments (Figure 2). Chacun de ces fragments est réutilisé pour la description des arbres. Par exemple, toute description d'arbre qui représente un contexte où le sujet est réalisé en position canonique utilisera le fragment *SujCan*. De plus, pour implémenter le liage syntaxe/sémantique, chaque classe décrivant un fragment arborescent coindexe la valeur du trait sémantique dans la structure arborescente avec un paramètre de classe approprié. La seconde étape regroupe les classes qui décrivent les fonctions syntaxiques. Celles-ci capturent le fait qu'une fonction syntaxique est une notion qui permet de s'abstraire de la question de l'ordre des mots.

(1) Sujet(?x) → SujetCan(?x) ∨ SujRel(?x)
Objet(?x) → ObjCan(?x) ∨ ObjWh(?x)
ParObjet(?x) → ParObjCan(?x) ∨ ParObjWh(?x)
ObjetIndirect(?x) → IndObjCan(?x) ∨ IndObjWh(?x)

Par exemple, la classe *Sujet* quoi que simplifiée ici capture le fait que le sujet recouvre alternativement une réalisation nominale devant le verbe (*SujCan*) ou en position relativisée (*SujRel*). En outre, les classes décrivant les variantes fonctionnelles contribuent à établir le liage entre syntaxe et sémantique en propageant la coindexation de valeurs par utilisation de paramètres.

La troisième étape où l'on décrit les changements de diathèse constitue le point important. Pour capturer l'alternance actif/passif, nous utilisons la définition suivante :

(2) AlternancePassive(?x, ?y) →
(Sujet(?x) ∧ FormeActive ∧ Objet(?y))
∨(Sujet(?y) ∧ FormePassive ∧ ParObjet(?x))
∨(Sujet(?y) ∧ FormePassive)

Ce qui indique que les réalisations alternatives de l'actif et du passif sont soit faites d'un sujet, d'une forme verbale active et d'un objet; soit d'un sujet, d'une forme verbale passive et optionellement d'un complément d'agent. Cette classe définit deux paramètres $?x$ et $?y$ qui sont coindexés avec les paramètres des classes de fonctions syntaxiques. Les valeurs des paramètres $?x$ et $?y$ représentent les valeurs associées aux arguments dans la structure argumentale. Ainsi la première ligne, qui représente l'actif, a pour effet de lier le premier argument ($?x$) au sujet et le second ($?y$) à l'objet. Par contre, la seconde ligne, qui représente le passif, lie le premier argument au complément d'agent et le second au sujet. Finalement la dernière ligne, qui représente le passif sans agent, lie uniquement le second argument au sujet. Le premier argument n'est tout simplement pas lié.

En dernier lieu, la définition des familles (Abe02) demande d'introduire explicitement la représentation sémantique et de lier les variables de la prédication élémentaire avec les paramètres de classe appropriés. On définira une famille ditransitive de la manière suivante :

(3) FamilleDitransitive →
AlternancePassive(?x, ?y) ∧ ObjetIndirect(?z)
∧ <sem> $\diamond(?x, ?y, ?z)$

où <sem> représente la description d'un littéral sémantique dont les variables d'individu sont coindexées avec les paramètres de classes appropriés Le littéral sémantique ainsi décrit est la contrepartie dans notre langage de ce que la théorie de la projection appelle la structure argumentale. À chacun des niveaux de description les variables sont coindexées de manière appropriée. La génération de l'ensemble des arbres de la famille produit bien un ensemble de couples (arbre, représentation sémantique) dans lesquels la projection est bien réalisée.

5 Conclusion

Dans cet article, nous avons proposé une méthode de développement de grammaires d'arbres adjoints qui s'exprime dans un cadre monotone. Cette méthode permet de justifier l'usage d'un langage monotone pour la représentation lexico-grammaticale. En second lieu elle ouvre la porte au développement de grammaires d'arbres adjoints à large couverture comportant une interface syntaxe-sémantique. Ces grammaires doivent être interfacées avec les analyseurs TAG existants (GP05). Le langage ainsi que la méthode proposés ont permis de générer une grammaire d'arbre adjoints de taille importante (environ 4000 arbres). De plus le cadre proposé ici s'adapte facilement au développement de grammaires pour d'autres formalismes fortement lexicalisés. Ainsi, G. Perrier a pu réutiliser cette méthode pour produire une grammaire d'interaction. Nous avons également mené quelques expériences encourageantes qui ont permis de créer une petite grammaire XDG du français.

Références

Anne Abeillé. (Abe02) *Une grammaire d'arbres adjoints pour le français*. CNRS, Paris, 2002.

Tilman Becker. (Bec93) *HyTAG: A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Language*. PhD thesis, Universität des Saarlandes, 1993.

Marie-Hélène Candito. (Can99) *Organisation Modulaire et Paramétrable de Grammaires Electroniques Lexicalisées*. PhD thesis, Université de Paris 7, 1999.

Benoit Crabbé and Denys Duchier. (CD04) Metagrammar redux. In *Constraint Solving and Language Processing*, Copenhagen, 2004.

Denys Duchier, Joseph Leroux, and Yannick Parmentier. (DLP04) The metagrammar compiler: An nlp application with a multi-paradigm architecture. In *Mozart 2004*, Charleroi, 2004.

Bertrand Gaiffe, Benoit Crabbé, and Azim Roussanaly. (GCR02) A new metagrammar compiler. In *Proc. TAG+6*, Venise, 2002.

Claire Gardent and Laura Kallmeyer. (GK03) Semantic construction in feature-based tree adjoining grammar. In *Proc. EACL*, 2003.

Claire Gardent and Yannick Parmentier. (GP05) Large scale semantic construction for tree adjoining grammar. In *Proc. LACL 2005*, Bordeaux, 2005.

Carlos Prolo. (Pro02) Systematic grammar development in the XTAG project. In *COLING'02*, 2002.

Fei Xia. (Xia01) *Automatic Grammar Generation from two Different Perspectives*. PhD thesis, University of Pennsylvania, 2001.