Ambiguïté de portée et approche fonctionnelle des grammaires d'arbres adjoints

Sylvain POGODALLA
LORIA/INRIA Lorraine
sylvain.pogodalla@loria.fr

Résumé. En s'appuyant sur la notion d'arbre de dérivation des Grammaires d'Arbres Adjoints (TAG), cet article propose deux objectifs : d'une part rendre l'interface entre syntaxe et sémantique indépendante du langage de représentation sémantique utilisé, et d'autre part offrir un noyau qui permette le traitement sémantique des ambiguïtés de portée de quantificateurs sans utiliser de langage de représentation sous-spécifiée.

Abstract. Relying on the derivation tree of the Tree Adjoining Grammars (TAG), this paper has to goals : on the one hand, to make the syntax/semantics interface independant from the semantic representation language, and on the other hand to propose an architecture that enables the modeling of scope ambguities without using underspecified representation formalisms.

Mots-clés : interface syntaxe et sémantique, sémantique formelle, grammaires d'arbres adjoints, grammaires catégorielles.

Keywords: syntax/semantics interface, formal semantics, tree adjoining grammars, categorial grammars.

1 Introduction

La notion d'arbre de dérivation dans les grammaires d'arbres adjoints (TAG) (Joshi & Schabes, 1997; Abeillé, 1993) est censée représenter les dépendances entre les différents items lexicaux d'une phrase. À ce titre, l'arbre de dérivation apparaît comme le candidat privilégié pour réaliser le transfert structurel entre la syntaxe et la sémantique de manière compositionnelle. Or, sa représentation ne rendant pas explicite certains liens, il a été proposé, afin de le rendre opérationnel dans le cadre du calcul de la représentation sémantique, soit de l'étendre (Kallmeyer, 2002; Joshi *et al*., 2003), soit de ne pas l'utiliser et de calculer la représentation sémantique directement sur l'arbre dérivé (Frank & van Genabith, 2001; Gardent & Kallmeyer, 2003; Gardent, 2007).

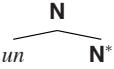
Cet article propose d'utiliser la notion d'arbre de dérivation telle qu'introduite dans (Pogodalla, 2004). En effet, cette notion, qui précise simplement la notion originale, y est montrée comme adéquate pour la représentation des dépendances longue distance. Néanmoins, le langage de représentation sémantique qui est utilisé est un formalisme sous-spécifié. Ces derniers posent parfois problème, comme dans le cas de la coordination de groupes nominaux quantifiés (Willis, 2007). De plus, nous voulons un cadre général qui laisse à l'utilisateur le choix d'utiliser ou

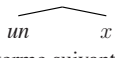
non de tels formalismes, tout en gardant la possibilité de modéliser les ambiguïtés. Ainsi, nous utilisons un formalisme plus proche de celui proposé par Montague (Montague, 1974) et une architecture qui permet de traiter des phénomènes d'ambiguïté. Nous nous appuyons sur les Grammaires Catégorielles Abstraites (ACG) (de Groote, 2001), et, *tout en gardant un seul arbre dérivé*, nous montrons comment le principe d'élévation de type des grammaires catégorielles permet d'obtenir plusieurs lectures sémantiques.

Dans les deux prochaines sections, nous présentons l'arbre de dérivation de (Pogodalla, 2004) sur des exemples. Puis nous définissons dans la section 4 la notion d'ACG et les architectures qu'elle rend possible pour l'interface entre la syntaxe et la sémantique. La section 5 met finalement en œuvre une telle architecture pour modéliser l'ambiguïté de portée des quantificateurs.

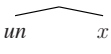
2 Lecture fonctionnelle de l'arbre dérivé

La présentation habituelle des arbres élémentaires des TAG en donne une vision relativement statique, les opérations de substitution et d'adjonction se chargeant de donner la dynamique du langage en combinant selon certaines règles les arbres entre eux. Dans cette section, nous nous proposons d'intégrer cette notion de dynamique aux arbres élémentaires eux-mêmes, en décrivant comment chacun prend part aux opérations de substitution et d'adjonction. Cette description se fait sur base d'exemples.

Soit l'arbre auxiliaire suivant : 

. Lorsqu'il est adjoint à un autre arbre à un nœud $\mathbf{N}_0$, cet arbre remplace son propre nœud $\mathbf{N}^*$ par le sous-arbre de racine $\mathbf{N}_0$. Si l'on appelle x ce sous-arbre, on peut donc considérer l'arbre auxiliaire comme une fonction qui transforme un arbre x en un nouvel arbre 

. Soit, avec la notation à l'aide du λ-calcul, une représentation de cet arbre par le terme suivant :

$$c_{\mathit{un}} = \lambda x.$$ 

Considérons maintenant l'arbre initial suivant : 

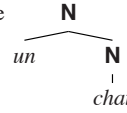
. Cet arbre peut se voir adjoindre un arbre auxiliaire au nœud **N**. Dans ce cas, il donnera comme argument à cet arbre auxiliaire (on a vu que l'arbre auxiliaire peut être décrit comme étant une fonction qui prend un arbre en argument et retourne un arbre) le sous arbre 

(dans le cas présent, le sous-arbre paramètre est l'arbre tout entier car l'adjonction a lieu au nœud racine).

On peut donc représenter l'arbre initial comme une fonction qui prend comme paramètre un arbre auxiliaire, c'est-à-dire *une fonction des arbres dans les arbres*. Soit, avec la notation en λ-calcul :

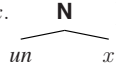
$$\lambda a.a($$ 

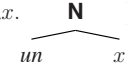
$$)$$

On constate alors que l'opération d'adjonction qui permet d'obtenir l'arbre 

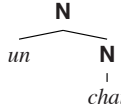
est décrite par l'application de la fonction (du terme) c_{chat} au terme c_{un}. En effet :

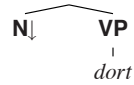
$c_{chat}c_{un} = (\lambda a.a($ 

$))(\lambda x.$ 

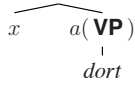
$) \rightarrow_\beta (\lambda x.$ 

$)($ 

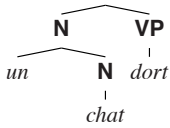
$) \rightarrow_\beta$ 

On peut finalement avoir un arbre qui combine la possibilité de subir une adjonction et une substitution. Prenons par exemple l'arbre initial suivant : 

. Sa particularité est qu'il attend un arbre qui peut être substitué au nœud **N** d'une part, et qu'il peut subir une adjonction au nœud **VP**. On choisit donc de le représenter comme une fonction qui prend en premier argument un arbre auxiliaire, c'est-à-dire une fonction, et en deuxième argument un arbre x qui est celui qui est substitué au nœud **N**. On peut alors le représenter de la manière suivante :

$c_{dort} = \lambda ax.$ 

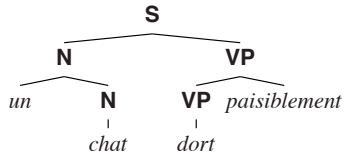
Bien entendu, il est possible qu'aucune adjonction n'ait lieu sur le nœud **VP**[1]. Dans l'optique que nous avons choisie, cela signifie que la fonction qui a été adjointe est l'identité $I = \lambda x.x$.

L'arbre dérivé 

est alors représenté par le terme $c_{dort}\, I\, (c_{chat}c_{un})$.

Avec une représentation adéquate de l'adverbe, par exemple $c_{paisiblement} = \lambda x.$ 

, on peut également construire l'arbre dérivé représenté par le terme $c_{dort}c_{paisiblement}(c_{chat}c_{un})$:



[1] Pour des raisons de clarté dans la présentation, nous avons omis la possibilité d'une adjonction sur le nœud **S**, et donc supprimé le paramètre correspondant. On voit également par là comment interdire des adjonctions.

Si l'on appelle γ le type des arbres, on voit que l'on a les constantes et le typage suivants :

$$\begin{array}{ll} c_{un} & : \gamma \multimap \gamma \\ c_{chat} & : (\gamma \multimap \gamma) \multimap \gamma \\ c_{dort} & : (\gamma \multimap \gamma) \multimap \gamma \multimap \gamma \end{array} \qquad \begin{array}{ll} c_{paisiblement} & : \gamma \multimap \gamma \\ I & : \gamma \multimap \gamma \end{array}$$

où $\multimap$ désigne l'implication linéaire[2].

3 Rôle de l'arbre de dérivation

En typant les constantes représentant les arbres auxiliaires et initiaux de cette manière, nous perdons toutefois une information importante : les arbres ont tous le même type γ, et aucune distinction n'est faire entre eux. Ainsi, la composition $c_{chat}c_{paisiblement}$ serait tout à fait licite. C'est pourquoi nous allons donner aux constantes un type plus abstrait[3], correspondant aux non-terminaux qui étiquettent leur racine. Nous nous donnons donc les types de base suivants : **VP**, **S**, **V**, **N** ainsi que les types qui correspondent aux racines des nœuds auxiliaires : $\mathbf{VP}_\mathsf{A}$, $\mathbf{S}_\mathsf{A}$, $\mathbf{V}_\mathsf{A}$, $\mathbf{N}_\mathsf{A}$.

Ainsi, en reprenant les exemples ci-dessus et en introduisant de nouvelles constantes, nous avons les typages suivants :

$$\begin{array}{ll} C_{dort} & : \mathbf{VP}_\mathsf{A} \multimap \mathbf{N} \multimap \mathbf{S} \\ C_{chat} & : \mathbf{N}_\mathsf{A} \multimap \mathbf{N} \end{array} \qquad \begin{array}{ll} C_{un} & : \mathbf{N}_\mathsf{A} \\ C_{paisiblement} & : \mathbf{VP}_\mathsf{A} \\ I_{\mathbf{VP}} & : \mathbf{VP}_\mathsf{A} \end{array}$$

On peut alors avoir le terme $C_{dort}I_{\mathbf{VP}}(C_{chat}C_{un})$, de type **S**, tandis que le terme $C_{chat}C_{paisiblement}$ n'est pas typable. Il reste à établir le lien avec le terme $c_{dort}(c_{chat}c_{un})$ de la section précédente. Cela se fait par une fonction de conversion $:=_{\text{syntax}}$, le *lexique*, qui convertit les types et les constantes ainsi :

$$\begin{array}{llll} \mathbf{S} & :=_{\text{syntax}} & \gamma & C_{dort} :=_{\text{syntax}} c_{dort} \\ \mathbf{VP} & :=_{\text{syntax}} & \gamma & C_{un} :=_{\text{syntax}} c_{dort} \\ \mathbf{N} & :=_{\text{syntax}} & \gamma & C_{chat} :=_{\text{syntax}} c_{chat} \\ \mathbf{N}_\mathsf{A} & :=_{\text{syntax}} & \gamma \multimap \gamma & C_{paisiblement} :=_{\text{syntax}} c_{paisiblement} \\ \mathbf{VP}_\mathsf{A} & :=_{\text{syntax}} & \gamma \multimap \gamma & I_X :=_{\text{syntax}} \lambda x.x \text{ pour tout type } X \end{array}$$

TAB. 1 – Définition du lexique

On alors :

$$C_{dort}I_{\mathbf{VP}}(C_{chat}C_{un}) \quad :=_{\text{syntax}} \quad c_{dort}I(c_{chat}c_{un})$$

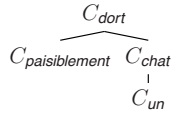
et

$$C_{dort}C_{paisiblement}(C_{chat}C_{un}) \quad :=_{\text{syntax}} \quad c_{dort}c_{paisiblement}(c_{chat}c_{un})$$

Si l'on adopte une représentation arborescente des λ-termes (il n'y a pas d'abstraction), on peut représenter $C_{dort}C_{paisiblement}(C_{chat}C_{un})$ par :

[2]Nous ne dirons rien du calcul logique sous-jacent, ni de la manière d'introduire de la non-linéarité avec l'implication intuitionniste habituelle $\rightarrow$. Nous renvoyons les lecteurs intéressés à (de Groote, 2001; Pogodalla, 2004).

[3]Car pouvant être réalisé, ou instancié, de différentes manières : arbre (γ) pour la syntaxe, mais aussi individu (e), prédicat ($e \multimap t$), etc. pour la sémantique.



Cet arbre rappelle très précisément *l'arbre de dérivation* tel qu'il est défini classiquement dans les TAG. En fait, il s'agit de la même notion où sont cependant explicités :
– l'ordre des arguments, qui doit être le même pour la constante qui est représentée dans l'arbre de dérivation et pour la constante qui lui est associée dans les arbres dérivés. Le choix est libre, mais une fois qu'il est fait, il doit être cohérent ;
– l'ordre des adjonctions lors d'une dérivation. Contrairement à la notion classique, où cet ordre n'est pas précisé, le résultat étant le même, ici l'ordre des opérations est spécifié. Cela ne change pas le pouvoir expressif, cela permet par contre de doter les TAG d'une sémantique compositionnelle basée sur l'arbre de dérivation.

Cette manière de représenter les arbres dérivés, les arbres de dérivation, et les relations qu'il y a entre eux, correspond en fait à la modélisation des TAG dans le formalisme des ACG.

4 Modélisation des TAG dans les ACG

Nous ne reprenons pas ici le détail la modélisation systématique des TAG dans les ACG, donné dans (de Groote, 2002; Pogodalla, 2004). Nous allons simplement donner les définitions précises des ACG qui ont été mises en œuvre dans les exemples précédents, afin d'en tirer l'architecture générale que nous utiliserons pour modéliser les ambiguïtés de portée des quantificateurs.

Une ACG définit deux langages : un *langage abstrait*, qui peut être vu comme un ensemble abstrait de structures grammaticales, et un *langage objet*, représentant les formes réalisées des structures abstraites, qu'elle met en relation. Ici, le langage abstrait correspond à la structure grammaticale que l'on veut manipuler : l'arbre de dérivation. Dans l'exemple précédent, il est mis en relation avec le langage objet des arbres dérivés grâce au *lexique*.

Definition 1 (Signature d'ordre supérieur). *Une* signature d'ordre supérieure *est un triplet* $\Sigma = \langle A, C, \tau \rangle$ *où :*
– *A est un ensemble de types atomiques ;*
– *C est un ensemble fini de constantes ;*
– *$\tau : C \rightarrow \mathcal{T}(A)$ qui assigne à chaque constante de C un type de $\mathcal{T}(A)$ où $\mathcal{T}(A) ::= A | \mathcal{T}(A) \multimap \mathcal{T}(A)$.*

On appel Λ_Σ l'ensemble des λ-termes que l'on peut construire avec la signature Σ.

Ainsi, dans l'exemple précédent, nous avions deux signatures d'ordre supérieur. La première contenait les types atomiques $\mathbf{S}, \mathbf{N}, \mathbf{VP}_{\mathsf{A}} \ldots$ et les constantes $C_{chat}, C_{un} \ldots$ tandis que la deuxième signature d'ordre supérieur contenait l'unique type atomique γ et les constante $c_{chat}, c_{un} \ldots$

Definition 2 (Lexique). *Étant données une signature d'ordre supérieur $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ et une signature d'ordre supérieur $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, un* lexique $:=$ de Σ_1 vers Σ_2 *est défini par la donnée de $\overset{\tau}{:=}$ et $\overset{c}{:=}$ tels que :*

– $:\overset{\tau}{=}\ : A_1 \rightarrow \mathcal{T}(A_2)$ *est une fonction d'interprétation des types atomiques de* Σ_1 *comme des types implicatifs construits à partir de* A_2*. On appellera* $:\overset{\tau}{=}$ *également son extension homomorphique à tous les types de* $\mathcal{T}(A_1)$ *;*
– $:\overset{c}{=}\ : C_1 \rightarrow \Lambda_{\Sigma_2}$ *est une fonction d'interprétation des constantes de* Σ_1 *comme des* λ*-termes construits à partir de* Σ_2*. On appellera* $:\overset{c}{=}$ *également son extension homomorphique à tous les termes de* Λ_{Σ_1} *;*
– *les fonctions d'interprétation sont compatibles avec la relation de typage, c'est-à-dire que pour tout* $c \in C_1$ *et* $t : \alpha \in \Lambda_{\Sigma_2}$ *tels que* $c :\overset{c}{=} t$*, alors* $\tau_1(c) :\overset{\tau}{=} \alpha$ *(le type de l'image de* c *est l'image du type de* c*).*

Dans la suite, on utilisera sans ambiguïté $:=$ *pour* $:\overset{\tau}{=}$ *ou* $:\overset{c}{=}$*.*

Le tableau 1 définit bien un lexique. La colonne de gauche donne l'interprétation des types atomiques (on remarquera avec l'interprétation du type $\mathbf{VP_A}$ que l'interprétation d'un type atomique peut être un type non atomique). La colonne de droite donne l'interprétation des constantes.

Definition 3 (Grammaire catégorielle abstraite). *Une* grammaire catégorielle abstraite *est un quadruplet* $\mathcal{G} = \langle \Sigma_1, \Sigma_2, :=, s \rangle$ *où :*
– Σ_1 *est une signature d'ordre supérieure, et* Σ_2 *une signature d'ordre supérieure. Ils sont appelés* vocabulaire abstrait *et* vocabulaire objet *;*
– $:=\ : \Sigma_1 \rightarrow \Sigma_2$ *est un lexique ;*
– s *est un type atomique du vocabulaire abstrait, appelé le* type distingué *de la grammaire.*

Definition 4 (Langages abstrait et objet). *Soit* $\mathcal{G} = \langle \Sigma_1, \Sigma_2, :=, s \rangle$ *une grammaire catégorielle abstraite.*

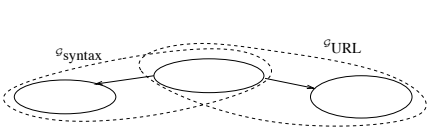
1. *Le* langage abstrait $\mathcal{A}(\mathcal{G})$ *engendré par* $\mathcal{G}$ *est défini par* $\mathcal{A}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_1} | t : s\}$
2. *Le* langage objet $\mathcal{O}(\mathcal{G})$ *engendré par* $\mathcal{G}$ *est défini par* $\mathcal{O}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_2} | \exists u \in \mathcal{A}(\mathcal{G}) \text{ avec } u := t\}$

Ainsi, les termes pris en exemple appartiennent bien aux vocabulaires abstrait et objet. Il est à noter que cette définition permet d'éviter que le terme $c_{\mathit{dort}}(c_{\mathit{chat}} c_{\mathit{paisiblement}})$, qui est bien un arbre (de type γ), appartienne effectivement au langage objet des arbres dérivés. En effet, il serait l'image de $C_{\mathit{dort}}(C_{\mathit{chat}} C_{\mathit{paisiblement}})$ qui n'est pas de type $\mathbf{S}$ (ce terme n'est même pas typable) et qui n'appartient donc pas au langage abstrait des arbres de dérivation.

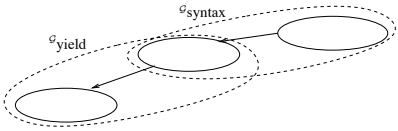
La définition des ACG permet de considérer différents types d'architecture. Par exemple, si deux ACG partagent le même vocabulaire abstrait, on aura le schéma de composition de la figure 1(a). C'est par exemple celui adopté dans (Pogodalla, 2004) pour doter les TAG d'une représentation sémantique sous-spécifiée.

On peut également composer deux ACG en faisant que le vocabulaire objet de l'une soit également le vocabulaire abstrait de l'autre (figure 1(b)). C'est par exemple le cas si l'on veut considérer le lien entre les arbres dérivés, cette fois vus comme un langage abstrait, et leur production (*yield* en anglais) comme langage de chaîne.

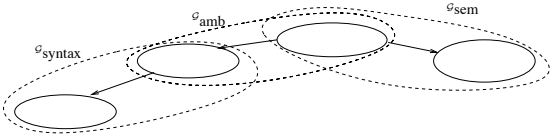
Bien entendu, on peut mélanger ces deux types de composition. La modélisation que nous proposons pour les phénomènes d'ambiguïté de portée des quantificateurs repose sur le schéma de la figure 1(c). Dans tous les cas, on retrouve un schéma classique du TAL, même si la relation est décrite par un autre formalisme : celui de la composition de *transducer*.



(a) Composition par partage de vocabulaire abstrait



(b) Composition par identification de vocabulaires abstrait et objet



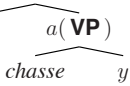
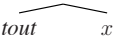
(c) Composition par combinaison des deux modes

FIG. 1 – Exemples d'architectures possibles

5 Composition d'ACG et modélisation des ambiguïtés de portée

Proposition. Notre objectif est de proposer pour les TAG un cadre dans lequel modéliser les ambiguïtés de portée sans utiliser de formalisme sous-spécifié (contrairement à (Pogodalla, 2004)), tout en gardant la contrainte d'avoir un unique arbre dérivé auquel peuvent être associées plusieurs représentations sémantiques. Pour l'architecture que nous proposons, il nous faut définir deux nouvelles ACG. La première, $\mathcal{G}_{\text{amb}}$, permettra d'associer à un arbre de dérivation unique deux structures plus profondes. La seconde, $\mathcal{G}_{\text{sem}}$, correspondra à la réalisation dans un langage de formes logiques du type de Montague de ces structures plus profondes.

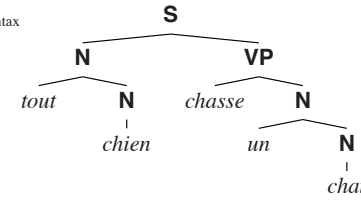
On pourra penser que ces différents niveaux ajoutent de la complexité conceptuelle. Nous pensons pour notre part que cela permet de modulariser les difficultés (en ne traitant qu'à leurs niveaux respectifs la syntaxe, avec $\mathcal{G}_{\text{syntax}}$, et la sémantique, avec $\mathcal{G}_{\text{amb}}$ et $\mathcal{G}_{\text{sem}}$). Par ailleurs, nous avons vu que cette notion de composition est déjà présente en TAL avec l'utilisation de *transducer* et de leur composition, pour le traitement de la morphologie par exemple. Si au final seule la relation entre les langages d'entrée et sortie nous intéresse, il est tout à fait possible de compiler deux ACG, par exemple $\mathcal{G}_{\text{amb}}$ et $\mathcal{G}_{\text{syntax}}$ en une seule, leur *composée*.

Supposons que la grammaire $\mathcal{G}_{\text{syntax}}$ contiennent également les arbres suivants : $c_{\textit{chasse}} = \lambda axy.$  , $c_{\textit{chien}} = \lambda a.a($  $)$ et $c_{\textit{tout}} = \lambda x.$  , ainsi que les

constantes $C_{\textit{chien}} : \mathbf{N_A} \multimap \mathbf{N}$, $C_{\textit{chasse}} : \mathbf{VP_A} \multimap \mathbf{N} \multimap \mathbf{N} \multimap \mathbf{S}$ et $C_{\textit{tout}} : \mathbf{N_A}$ mis en relation par le lexique de la manière suivante : $C_{\textit{chasse}} :=_{\text{syntax}} c_{\textit{chasse}}$, $C_{\textit{chien}} :=_{\text{syntax}} c_{\textit{chien}}$ et $C_{\textit{tout}} :=_{\text{syntax}} c_{\textit{tout}}$.

Alors l'arbre de dérivation correspondant à l'arbre dérivé

$c_{\mathit{chasse}}(c_{\mathit{chien}}c_{\mathit{tout}})(c_{\mathit{chat}}c_{\mathit{un}}) :=_{\text{syntax}}$



est $t_0 = C_{\mathit{chasse}} I_{\mathbf{VP}} (C_{\mathit{chien}} C_{\mathit{tout}})(C_{\mathit{chat}} C_{\mathit{un}})$.

Définissons maintenant $\mathcal{G}_{\text{amb}}$, dont le vocabulaire objet est le vocabulaire abstrait de $\mathcal{G}_{\text{syntax}}$, et dont le vocabulaire abstrait contient les mêmes symboles de type que le vocabulaire objet mais les constantes typées $D_{\mathit{chasse}} : \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{N} \multimap \mathbf{N} \multimap \mathbf{S}$, $D_{\mathit{chien}} : \mathbf{N}_{\mathsf{A}} \multimap (\mathbf{N} \multimap \mathbf{S}) \multimap \mathbf{S}$, $D_{\mathit{chat}} : \mathbf{N}_{\mathsf{A}} \multimap (\mathbf{N} \multimap \mathbf{S}) \multimap \mathbf{S}$, $D_{\mathit{tout}} : \mathbf{N}_{\mathsf{A}}$, $D_{\mathit{un}} : \mathbf{N}_{\mathsf{A}}$ et $I^D_{\mathbf{VP}} : \mathbf{VP}_{\mathsf{A}}$. Le lexique $:=_{\text{amb}}$ est tel que pour tout type X, $X :=_{\text{amb}} X$ et :

$$\begin{array}{llll} D_{\mathit{chasse}} & :=_{\text{amb}} & C_{\mathit{chasse}} & \quad D_{\mathit{chien}} \quad :=_{\text{amb}} \quad \lambda a P.P(C_{\mathit{chien}}\, a) \\ D_{\mathit{tout}} & :=_{\text{amb}} & C_{\mathit{tout}} & \quad D_{\mathit{chat}} \quad :=_{\text{amb}} \quad \lambda a P.P(C_{\mathit{chat}}\, a) \\ D_{\mathit{un}} & :=_{\text{amb}} & C_{\mathit{un}} & \quad I^D_{\mathbf{VP}} \quad :=_{\text{amb}} \quad I_{\mathbf{VP}} \end{array}$$

Soit alors les termes :

$$\begin{array}{ll} t_1 & = (D_{\mathit{chien}} D_{\mathit{tout}})(\lambda x.(D_{\mathit{chat}} D_{\mathit{un}})(\lambda y. D_{\mathit{chasse}} I^D_{\mathbf{VP}}\, x\, y)) \\ t_2 & = (D_{\mathit{chat}} D_{\mathit{un}})(\lambda y.(D_{\mathit{chien}} D_{\mathit{tout}})(\lambda x. D_{\mathit{chasse}} I^D_{\mathbf{VP}}\, x\, y)) \end{array}$$

On pourra vérifier que t_1 et t_2 sont bien typés et que $t_1 :=_{\text{amb}} t_0$ et $t_2 :=_{\text{amb}} t_0$. Ainsi, nous avons désormais deux structures profondes (t_1 et t_2) reliées à un seul arbre de dérivation (t_0).

Il nous reste à transformer ces structures en formules logiques à l'aide d'une nouvelle ACG $\mathcal{G}_{\text{sem}}$. Celle-ci partage son vocabulaire abstrait avec $\mathcal{G}_{\text{amb}}$, et, au niveau objet, met en œuvre les types habituels e et t pour les représentations à la Montague. Avec le lexique $:=_{\text{sem}}$[4] suivant[5] :

$$\begin{array}{lll} & & D_{\mathit{chasse}} \quad :=_{\text{sem}} \lambda aso.(a(\lambda x.\mathbf{chasse}\, x\, o))s \\ \mathbf{S} & :=_{\text{sem}} t & D_{\mathit{tout}} \quad :=_{\text{sem}} \lambda PQ.\forall x.P\, x \Rightarrow Q\, x \\ \mathbf{N} & :=_{\text{sem}} e & D_{\mathit{un}} \quad :=_{\text{sem}} \lambda PQ.\exists x.P\, x \wedge Q\, x \\ \mathbf{N}_{\mathsf{A}} & :=_{\text{sem}} (e \to t) \to (e \to t) \to t & D_{\mathit{chat}} \quad :=_{\text{sem}} \lambda q.q(\lambda x.\mathbf{chat}\, x) \\ \mathbf{VP}_{\mathsf{A}} & :=_{\text{sem}} (e \to t) \to (e \to t) & D_{\mathit{chien}} \quad :=_{\text{sem}} \lambda q.q(\lambda x.\mathbf{chien}\, x) \\ & & I^D_{\mathbf{VP}} \quad :=_{\text{sem}} \lambda x.x \end{array}$$

Nous laissons le lecteur vérifier que l'on obtient bien alors les deux lectures :

$$\begin{array}{ll} t_1 & :=_{\text{sem}} \quad \forall x.\mathbf{chien}\, x \Rightarrow (\exists y.\mathbf{chat}\, y \wedge \mathbf{chasse}\, x\, y) \\ t_2 & :=_{\text{sem}} \quad \exists y.\mathbf{chat}\, y \wedge (\forall x.\mathbf{chien}\, x \Rightarrow \mathbf{chasse}\, x\, y) \end{array}$$

Faute de place, nous ne pouvons illustrer également la coordination de groupes nominaux quantifiés avec les constantes $C_{\mathit{et}} : \mathbf{N} \multimap \mathbf{N} \multimap \mathbf{N}$, $D_{\mathit{et}} : ((\mathbf{N} \multimap \mathbf{S}) \multimap \mathbf{S}) \multimap ((\mathbf{N} \multimap$

[4] On suppose présentes dans la signature objet les constantes $\mathbf{chasse} : e \multimap e \multimap t$, $\mathbf{chien} : e \multimap t$, $\mathbf{chat} : e \multimap t$, $\forall : (e \to t) \multimap t$ et $\exists : (e \to t) \multimap t$.

[5] Notons que c'est la présence du paramètre a dans la formule sémantique qui réalise D_{chasse} qui permet, en intégrant la contribution des éventuels sous-arbres adjoints, la prise en compte des dépendances longue distance.

$\mathbf{S}) \multimap \mathbf{S}) \multimap ((\mathbf{N} \multimap \mathbf{S}) \to \mathbf{S})$ et leur réalisation $D_{\mathit{et}} :=_{\text{amb}} \lambda PQr.P(\lambda x.Q(\lambda y.r(C_{\mathit{et}}\, x\, y)))$ et $D_{\mathit{et}} :=_{\text{sem}} \lambda PQr.P\, r \wedge Q\, r$. On aurait par exemple les deux termes

$$\begin{aligned} t_3 &= D_{\mathit{et}}(D_{\mathit{chat}} D_{\mathit{tout}})(D_{\mathit{chien}} D_{\mathit{un}})(\lambda x.(D_{\mathit{souris}} D_{\mathit{une}})(\lambda y.D_{\mathit{chasse}} I^{D}_{\mathbf{VP}}\, x\, y)) \\ t_4 &= (D_{\mathit{souris}} D_{\mathit{une}})(\lambda y.D_{\mathit{et}}(D_{\mathit{chat}} D_{\mathit{tout}})(D_{\mathit{chien}} D_{\mathit{un}})(\lambda x.D_{\mathit{chasse}} I^{D}_{\mathbf{VP}}\, x\, y)) \end{aligned}$$

qui donneraient les deux lectures attendues pour *tout chat et un chien chassent une souris*. Contrairement au problème soulevé par les représentations sous-spécifiées dans (Willis, 2007), on n'a pas la lecture où *tout chat* a une portée différente de *un chien* vis à vis de la portée de *une souris*. On obtient ainsi une architecture dans laquelle modéliser les phénomènes d'ambiguïté de portée sans imposer l'utilisation de formalismes sous-spécifiés.

Limitations. Actuellement, nous ne savons pas exprimer les contraintes de portée des quantificateurs, telles celles des îlots de portée. Ce problème est comparable à celui rencontré par les grammaires de types logiques. En effet, l'approche proposée ici repose sur le principe de l'élévation de type, qui est à la base de la prise en compte des ambiguïtés de portée dans ces grammaires. Ici, nous avons gardé la contrainte supplémentaire que, bien entendu, l'arbre dérivé et l'arbre de dérivation restent uniques. La solution que nous envisageons repose sur une extension du système de type des ACG, et va bien au-delà du sujet de cet article[6].

6 Conclusion

Nous avons montré comment, en se basant sur la définition précise de l'arbre de dérivation de (Pogodalla, 2004), nous pouvons définir un calcul des représentations sémantiques pour les TAG qui ne nécessite pas l'usage de formalismes sous-spécifiés tout en permettant le traitement de l'ambiguïté. Cela nous permet d'une part de renforcer l'indépendance entre le formalisme syntaxique des TAG et le formalisme choisi par l'utilisateur pour la représentation sémantique, et d'autre par de confirmer l'importance de cette notion d'arbre de dérivation. Par ailleurs, notre approche a de forts liens avec les approches de Glue Semantics (Dalrymple, 2001), et la proposition (Frank & van Genabith, 2001) (utilisant les principes de Glue Semantics depuis l'arbre dérivé) pourrait sans doute être reconsidérée avec cette notion d'arbre de dérivation.

Références

ABEILLÉ A. (1993). *Les nouvelles syntaxes*. Paris : Armand Colin Éditeur.

DALRYMPLE M. (2001). *Lexical Functional Grammar*, volume 42 of *Syntax and Semantics series*. Academic Press.

DE GROOTE P. (2001). Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, p. 148–155.

DE GROOTE P. (2002). Tree-adjoining grammars as abstract categorial grammars. In *TAG+6, Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*, p. 145–150 : Università di Venezia.

[6] Faute de place, nous ne pouvons pas exposer comment l'architecture proposée ici permet également de dépasser la limitation mentionnée dans (Pogodalla, 2004) pour les verbes à contrôle.

FRANK A. & VAN GENABITH J. (2001). Glue tag : Linear logic based semantics construction for LTAG - and what it teaches us about the relation between LFG and LTAG. In M. BUTT & T. H. KING, Eds., *Proceedings of the LFG '01 Conference*, Online Proceedings : CSLI Publications. `http://cslipublications.stanford.edu/LFG/6/lfg01.html`.

GARDENT C. (2007). Tree adjoining grammar, semantic calculi and labelling invariants. In (Getzen *et al*., 2007).

GARDENT C. & KALLMEYER L. (2003). Semantic construction in feature-based tag. In *Proceedings of the 10th Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*.

J. GETZEN, E. THIJSSE, H. BUNT & A. SCHIFFRIN, Eds. (2007). *Proceedings of the Seventh International Workshop on Computational Semantics, IWCS-7*. Tilburg University.

JOSHI A. K., KALLMEYER L. & ROMERO M. (2003). Flexible composition in ltag : Quantifier scope and inverse linking. In H. BUNT, I. VAN DER SLUIS & R. MORANTE, Eds., *Proceedings of the Fifth International Workshop on Computational Semantics IWCS-5*.

JOSHI A. K. & SCHABES Y. (1997). Tree-adjoining grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of formal languages*, chapter 2. Springer.

KALLMEYER L. (2002). Using an enriched tag derivation structure as basis for semantics. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*.

MONTAGUE R. (1974). The proper treatment of quantification in ordinary english. In P. PORTNER & B. H. PARTEE, Eds., *Formal Semantics : The Essential Readings*, chapter 1. Blackwell Publishers. 2002 edition.

POGODALLA S. (2004). Computing semantic representation : Towards ACG abstract terms as derivation trees. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, p. 64–71.

WILLIS A. (2007). NP coordination in underspecified scope representations. In (Getzen *et al*., 2007).