Modélisation des mouvements explicites dans les ACG avec le produit dépendant

Florent Pompigne

RECITAL 2009, Senlis, 24–26 juin 2009

Modélisation des mouvements explicites dans les ACG avec le produit dépendant

Florent Pompigne[1]
(1) ENS Cachan / INRIA Nancy - Grand-Est
florent.pompigne@loria.fr

Abstract. Abstract Categorial Grammars (ACG) is a grammatical framework based on linear lambda-calculus. As in Muskens' Lambda Grammars, an abstract term in this kind of categorial grammar can be realized in different directions, such as syntactic and semantic ones. This structure provides autonomy for these different processings. ACG's architecture is independent from the logic used and so the type system is easily extensible in order to deal better with some linguistic phenomena. We will first introduce ACGs and the dependent product construction .This paper will then be concerned with the issue of overt grammatical movements, in particular extraction constraints in relative propositions, and how several close frameworks deal with it. Last we will show how to capture this phenomenon in extended ACG.

Mots-clés : syntaxe, grammaires catégorielles abstraites, types dépendant, mouvements explicites, extraction.

Keywords: Syntax, abstract categorial grammars, dependant product, overt movements, extraction.

1 Introduction

La description de la syntaxe d'une langue naturelle et de la relation entre ce niveau syntaxique et le niveau sémantique est traditionnellement opérée en Traitement Automatique des Langues par un formalisme grammatical. Les Grammaires Catégorielles Abstraites (ACG), proposées par de Groote (de Groote, 2001), se distinguent dans le paysage des formalismes existants par des propriétés héritées des grammaires catégorielles (sensibilité aux ressources) et des propriétés propres à leur architecture : elles s'appuient sur un langage abstrait, dont les termes correspondent à des structures de dérivation, et un langage objet dont les termes correspondent aux structures dérivées.

Les mouvements grammaticaux explicites désignent les déplacements qui ont lieu dans des constructions comme les formes interrogatives ou les propositions relatives par rapport aux phrases correspondantes. L'extraction d'un composant de la phrase par ces constructions peut s'effectuer à distance arbitraire, et selon des contraintes précises de la langue qui sont bien étudiées et transcrites dans de nombreux formalismes grammaticaux. Le propos de cet article est de présenter une modélisation de ce phénomène pour les ACG.

Dans la partie suivante on introduira le formalisme des ACG ainsi que l'extension du système de type avec le produit dépendant. Dans la partie 3 sera présenté le problème des mouvements explicites, à travers les contraintes à l'extraction dans les constructions relative en français, et les réponses apportées dans plusieurs formalismes voisins, et la partie 4 détaillera la solution proposée pour les ACG.

2 Grammaires Catégorielles Abstraites

Avec une approche similaire aux Lambda Grammars (Muskens, 2007) ou aux Convergent Grammars (Pollard, 2007) , (Pollard, 2008), chaque ACG introduit une fonction d'un espace de termes dits abstraits vers un espace de termes objets. Ces termes objets peuvent représenter différentes structures de données utilisées en traitement automatique des langues, comme des arbres ou des chaînes de caractère, et on choisit donc de les construire sur le λ-calcul linéaire.

2.1 Architecture du formalisme

Intuitivement, une ACG consiste en deux signatures, appelées vocabulaire abstrait et vocabulaire objet, qui déclarent des types et des constantes typées dans le fragment implicatif de la logique linéaire, et d'un homomorphisme, appelé lexique, de la première vers la seconde. Une Grammaire Catégorielle Abstraite génère deux langages, un langage abstrait et un langage objet. Le langage abstrait est défini comme l'ensemble des λ-termes linéaires (chaque abstraction lie une unique variable) clos construits sur le vocabulaire abstrait qui sont du type spécifique S. Le langage objet est alors l'image du langage abstrait par le lexique.

Cette architecture permet à l'aide de deux ACG partageant un même langage abstrait (correspondant au niveau tectogrammatique) de réaliser ces termes abstraits vers un langage décrivant

leur sémantique et un langage décrivant leur forme de surface (correspondant au niveau phénogrammatique). Nous ne nous intéressons uniquement dans ce papier à la description de la syntaxe et donc à la relation entre tectogrammatique et phénogramatique. Donnons un exemple d'une telle ACG :

$\mathbf{N},\ \mathbf{NP},\ \mathbf{S} : type$ (vocabulaire abstrait)
$homme_c, village_c, ami_c : \mathbf{N}$
$joyeux_c : \mathbf{N} \multimap \mathbf{N}$
$un_c, le_c : \mathbf{N} \multimap \mathbf{NP}$
$dormir_c : \mathbf{NP} \multimap \mathbf{S}$
$aimer_c,\ rencontrer_c : \mathbf{NP} \multimap \mathbf{NP} \multimap \mathbf{S}$
$Jean_c,\ Marie_c,\ Paul_c : \mathbf{NP}$
$de_c : \mathbf{NP} \multimap \mathbf{N} \multimap \mathbf{N}$

string : $type$ (vocabulaire objet)
$homme,\ joyeux,\ un, ... :$ **string**

$\mathbf{N},\ \mathbf{NP},\ \mathbf{S} \longrightarrow$ **string** (lexique)
$homme_c \longrightarrow homme$
$joyeux_c \longrightarrow \lambda x.x + joyeux$
$un_c \longrightarrow \lambda x.un + x$
$dormir_c \longrightarrow \lambda x.x + dort$
$aimer_c \longrightarrow \lambda x, y.y + aime + x$
$Jean_c \longrightarrow Jean$
$de_c \longrightarrow \lambda x, y.y + de + x$

(on note par "+" l'opération de concaténation des chaînes de caractère)
On peut vérifier que le terme suivant appartient au langage abstrait

$$\underbrace{rencontrer_c\ \overbrace{(un_c\ \underbrace{(de_c\ Marie_c\ \ ami_c)}_{\mathbf{N}})}^{\mathbf{NP}}\ Jean_c}_{\mathbf{S}}$$

Son image par le lexique fait donc partie du langage objet

$rencontrer_c\ (un_c\ (de_c\ Marie_c\ \ ami_c))\ Jean_c$

$\longrightarrow (\lambda x, y.y + rencontre + x)\ ((\lambda x.un + x)\ ((\lambda x, y.y + de + x)\ Marie\ ami))\ Jean$

$\longrightarrow Jean\ rencontre\ un\ ami\ de\ Marie$ (β-réduction)

A l'inverse, la possibilité de partir d'un terme de surface et de décider s'il appartient au langage d'une ACG est un problème ouvert équivalent à la décidabilité du fragment exponentiel multiplicatif de la logique linéaire (Salvati, 2005). Cependant, si on se restreint à la classe des ACG du

second ordre, dans lesquelles les termes du langage abstrait sont de la forme $q_1 \multimap ... \multimap q_k \multimap p$ où les q_i sont des types atomiques , ce problème de l'appartenance est décidable en temps polynomial.

2.2 Extension par les produits dépendants

L'architecture des ACG ne dépend pas de la logique sur laquelle elles reposent. Contrairement aux autres constructions proposées par Philippe de Groote et Sarah Maarek (de Groote & Maarek, 2007)(implication non linéaire, produit cartésien, union disjointe et type unit) pour étendre le système de type, le produit dépendant ne correspond pas à une primitive de la logique linéaire. Il vient de la théorie intuitionniste des types de Martin-Löf (Martin-Löf, 1973) et a été utilisé dans le contexte de la logique linéaire par Cervesato et Pfenning (Cervesato & Pfenning, 2002). Le produit dépendant permet d'introduire des types qui dépendent de termes. Le formalisme ainsi étendu est Turing-complet, et le problème de l'appartenance est donc indécidable (de Groote *et al.*, 2007). Il reste à étudier quels fragments utilisant le pouvoir d'expression ainsi gagné restent décidables. Cet exemple montre comment utiliser les types dépendants pour instancier un groupe nominal par son genre.

$\mathbf{gender} = \{m, f\} : type$ (vocabulaire abstrait)
$\mathbf{NP}, \mathbf{N} : (\mathbf{gender})\, type$
$\mathbf{S} : type$

$village_c : \mathbf{N}\ m$
$ville_c : \mathbf{N}\ f$
$un_c : (\Pi x : gender)\ \mathbf{N}\ x \multimap \mathbf{NP}\ x$
$joyeux_c : (\Pi x : gender)\ \mathbf{N}\ x \multimap \mathbf{N}\ x$
$dormir_c : (\Pi x : gender)\ \mathbf{NP}\ x \multimap \mathbf{S}$

$\mathbf{gender} = \{m, f\} : type$ (vocabulaire objet)
$\mathbf{string} : type$
$ville\ joyeux,\ joyeuse,\ un,\ une, ... : \mathbf{string}$

$gender \longrightarrow gender$
$\mathbf{N},\ \mathbf{NP} \longrightarrow \lambda x.\mathbf{string}$ (lexique)
$\mathbf{S} \longrightarrow \mathbf{string}$
$ville_c \longrightarrow ville$
$village_c \longrightarrow village$
$un_c \longrightarrow \lambda g.\ \lambda x.\ \{m \rightarrow un,\ f \rightarrow une\}\ g + x$
$joyeux_c \longrightarrow \lambda g.\ \lambda x.\ x + \{m \rightarrow joyeux,\ f \rightarrow joyeuse\}\ g$
$dormir_c \longrightarrow \lambda g.\ \lambda x.\ x + dort$

On peut vérifier que le terme suivant appartient au langage abstrait.

$$\underbrace{dormir_c\ f\ \overbrace{(un_c\ f\ \underbrace{(joyeux_c\ f\ ville_c)}_{\mathbf{N}\ f})}^{\mathbf{NP}\ f}}_{S}$$

On en déduit que son image par le lexique appartient au langage objet

$dormir_c\ f\ (un_c\ f\ (joyeux_c\ f\ ville_c))$

$\longrightarrow une\ ville\ joyeuse\ dort$

3 Mouvements grammaticaux explicites

Les mouvements grammaticaux explicites sont ceux visibles au niveau de surface du langage : ce sont les extractions de constituants syntaxiques par des constructions comme les interrogatives en *qu*, les relatives, ou les semi-clivées en français. On s'interessera ici à la modélisation des contraintes d'extraction par les constructions relatives dans la langues française, ces contraintes étant très proches pour les autres mouvements explicites (Abeillé, 2002).

3.1 Propositions relatives en français dans les ACG

Dans les ACG le pronom relatif sujet *qui* serait représenté au niveau abstrait par le terme

$qui_c : (\mathbf{NP} \multimap \mathbf{S}) \multimap \mathbf{N} \multimap \mathbf{N}$

Le premier argument correspond à la suite de la proposition relative : c'est une phrase à laquelle il manque un sujet. Le second argument correspond au nom complété par cette proposition. L'image de cette constante par le lexique serait alors : $\lambda S.\ \lambda n.\ n\ +\ qui + (S\ \epsilon)$. On peut par exemple vérifier que la phrase suivante appartient au langage objet de surface :

$dormir_c\ (le_c\ (qui_c\ (\lambda x.aimer_c\ Marie_c\ x)\ homme_c))$

$\longrightarrow l'homme\ qui\ aime\ Marie\ dort$

Il faut noter que le système de type, contrairement à la tradition des grammaires catégorielles, n'opère aucun contrôle sur l'ordre des mots dans les ACG. Cet ordre des mots est géré par le lexique. On n'a donc pas ici de problème à générer des extractions non-périphériques (*l'homme que Marie aime passionnement dort*) ou à distance arbitraire (*l'homme que Jean dit que Pierre croit que Marie aime dort*). D'un autre côté, sans contrôle sur la construction de nos termes abstraits, on rencontre plusieurs phénomènes de surgénération. Le premier est lié à l'absence de contraintes sur le rôle du constituant grammatical extrait. Rien n'empêche en effet de reconnaître *l'homme qui Marie aime dort* de manière similaire à l'exemple initial. L'autre contrainte

qui doit être prise en compte est l'existence d'îlots d'extraction à l'intérieur desquels l'extraction est impossible depuis l'extérieur. Les propositions relatives elle-mêmes sont un exemple d'îlot d'extraction, et en effet *L'homme que ma soeur qui aime dort* ne devrait pas être reconnu comme une phrase de la langue française. Pourtant le terme

$$le\ (que\ (\lambda x.dormir\ (ma\ (qui\ (\lambda y.aime\ x\ y)\ soeur)))\ homme)$$

fait bien partie de notre langage abstrait. Les propositions complétives ne sont, elles, des îlots que pour l'extraction de sujet : *l'homme qui Marie crois que vient* est grammaticalement incorrect.

3.2 Le problème dans quelques autres formalismes

Les phénomènes d'extraction ont été considérés sous de nombreux angles différents. Examinons rapidement les solutions adoptées dans quelques formalismes voisins aux ACG.

Les Tree Adjoining Grammars permettent de bien rendre compte du phénomène (Abeillé, 2002). Le schéma de formation des arbres élémentaires à complément relativisé modifie les arbres à racine phrastique pour placer le complément en tête de phrase avec un trait <qu>. Ce complément pouvait se trouver à profondeur arbitraire dans l'arbre initial puisqu'il apparait sur le schéma sous un lien de dominance sous-spécifié. Dans le cas d'une proposition relative, l'arbre obtenu est dominé par un noeud de catégorie N qui domine un autre noeud de catégorie N décrivant l'antécédent. Dans tous les cas un noeud intermédiaire phrastique est ajouté : il permet avec l'opérateur d'adjonction d'ajouter un nombre arbitraire de constructions comme *Marie croit que*. L'extraction du sujet est traitée de manière similaire mais distincte.

Dans les grammaires d'interaction, chaque pronom relatif est décrit par un arbre dans lequel le complément extrait apparait mais est morphologiquement vide. Cette trace du complément pourra donc être unifiée avec les autres arbres où le complément est décrit, mais elle n'apparaitra pas en surface. Afin de permettre une dépendance à distance arbitraire entre le pronom relatif et le complément extrait, la trace est surmontée d'une relation de dépendance sous-spécifiée. Cependant, pour bloquer la traversée d'îlots d'extraction, cette relation est contrôlée : elle ne peut franchir que des noeuds étiquetés S et de fonction objet ou nulle (contraintes satisfaites par *que Marie aime* par exemple). L'arbre décrivant le pronom relatif sujet se distingue par l'absence de trace : le pronom assure la fonction de sujet. (Perrier, 2007).

Les grammaires catégorielles s'appuient traditionnellement sur le calcul de Lambek. Le connecteur implicatif est double : le / prend son argument directement à sa droite et le $\backslash$ directement à sa gauche. L'ordre des mots est donc géré dans le système de type. Ce système est trop rigide pour les constructions relatives puisqu'il empêche les extractions non-périphériques : l'argument d'un pronom relatif est une phrase à laquelle il manque un constituant à sa droite ou à sa gauche seulement. Une possibilité est d'ajouter un constructeur $\uparrow$ et sa règle d'introduction, A$\uparrow$B étant le type d'un terme qui complété quelque part d'un terme de type B serait de type A. On peut alors donner au pronom *qui* le type N$\backslash$N/(S$\uparrow$NP). On retrouve à ce moment la même

surgénération que dans les ACG. La modélisation des îlots d'extraction peut se faire à l'aide de constructeurs modaux rattachés à l'extracteur et à la trace qui doivent se combiner tandis que les îlots d'extraction jouent le rôle de portes verrouillées à ces constructeurs (Moortgat, 1996) , (Carpenter, 1997). C'est également l'approche présente dans les Lambda Grammars (Muskens, 2007). On verra que c'est de cette technique qu'on s'approchera le plus.

Dans les Convergent Grammars, les pronoms relatifs (et les autres constituants opérant des mouvements explicites) sont affectés d'une catégorie spéciale G[A,B,C] indiquant que ce terme se combine avec un terme de catégorie A contenant une variable libre de type B et renvoie un terme de type C tout en liant cette trace. Cette catégorie spéciale permet de ne construire aucun terme par lambda-abstraction. Il suffit donc de vérifier que des variables libres ne sortent pas des îlots d'extraction.

4 Modélisation des contraintes d'extraction dans les ACG étendues

Le système de type étendu va nous permettre de contrôler la construction des termes du langage abstrait dans le respect des contraintes de la langue. Le premier terme dont doivent dépendre les types de nos constantes est un marqueur de cas, qui vérifiera le rôle des constituants extraits (comme on ne s'intéresse ici qu'à la signature abstraite on omettra d'indicer chaque nom de constantes par c).

cas = $\{nom, acc, gen, comp\}$: type
NP, **N** : (cas) type
S : type
Jean, Marie, Paul, Laurent : $(\Pi c : \textbf{cas})\ \textbf{NP}\ c$
chien, maison, homme : $(\Pi c : \textbf{cas})\ \textbf{N}\ c$
un, le : $(\Pi c : \textbf{cas})\ \textbf{N}\ c \multimap \textbf{NP}\ c$
aimer, rencontrer : $\textbf{NP}\ acc \multimap \textbf{NP}\ nom \multimap \textbf{S}$
dormir : $\textbf{NP}\ nom \multimap \textbf{S}$
qui : $(\Pi c : \textbf{cas})\ (\textbf{NP}\ nom \multimap \textbf{S}) \multimap \textbf{NP}\ c \multimap \textbf{NP}\ c$

La phrase suivante n'appartient plus au langage : *aimer* contraint la variable x à prendre comme argument le cas *acc*, et *qui* le cas *nom*. Le terme ne peut donc pas être typé.

$dormir_c\ (le_c\ nom\ (qui_c\ (\lambda x.aimer_c\ x\ acc\ Marie_c\ nom)\ homme_c\ nom))$

$\longrightarrow l'homme\ qui\ Marie\ aime\ dort$

On introduit un autre marqueur, de type **ext** pour extraction. L'idée est que les pronoms relatifs forcent les propositions relatives à étiqueter la variable trace de l'extraction par ce marqueur à 0. Ce marqueur est propagé aux niveaux syntaxiques supérieurs mais ne peut pas franchir les îlots d'extraction. Il reste alors au pronom relatif de contrôler que la proposition relative est bien étiquetée par 0. Une troisième valeur du marqueur permet d'indiquer qu'un îlot à l'extraction

de sujet a été franchit. Cette valeur intermédiaire sera donc également acceptée par tous les pronoms sauf le *qui* extracteur de sujet.

ext = $\{0, 1, 2\}$: type
NP : $(ext)\,(cas)$ type
N : $(ext)\,(cas)$ type
S : (ext) type
S* : (ext) type
$chien, maison, homme$: $(\Pi c : \textbf{cas})\ \textbf{N}\ 2\ c$
un, le, mon :$(\Pi c : \textbf{cas})\ \textbf{N}\ 2\ c \multimap \textbf{NP}\ 2\ c$
$marcher, dormir$:$(\Pi y : \textbf{ext})\ \textbf{NP}\ y\ nom \multimap \textbf{S}\ y$
$aimer, rencontrer$:$(\Pi y_1, y_2 : \textbf{ext})\ \textbf{NP}\ y_1\ acc \multimap \textbf{NP}\ y_2\ nom \multimap \textbf{S}\ \min(y_1\ y_2)$
qui :$(\Pi c : \textbf{cas})\ (\textbf{NP}\ 0\ nom \multimap \textbf{S}\ 0) \multimap \textbf{N}\ 2\ c \multimap \textbf{N}\ 2\ c$
que :$(\Pi y : \textbf{ext}, c : \textbf{cas})\ (\textbf{NP}\ 0\ acc \multimap \textbf{S}\ y - 1) \multimap \textbf{N}\ 2\ c \multimap \textbf{N}\ 2\ c$
que' :$(\Pi y : \textbf{ext})\ \textbf{S}\ y \multimap \textbf{S}\ *\ max(y, 1)$
(je crois *que* l'idée est bonne)
$croire, dire$:$(\Pi y_1, y_2 : \textbf{ext})\,\textbf{S}\ *\ y_1 \multimap \textbf{NP}\ y_2\ nom \multimap \textbf{S}\ min\,(y_1\ y_2)$

On peut vérifier que ce terme n'est plus typé par la grammaire : (on omet d'écrire les termes du produit dépendant pour une meilleure lecture)

L'homme que ma soeur qui aime dort
$le\ (que\ (\lambda x.dormir\ (ma\ (qui\ (\lambda y.aime\ x\ y)\ soeur)))\ homme)$

En effet les deux variables qui jouent le rôle de traces x et y doivent porter le marqueur 0 (exigence du type des pronoms relatifs). Ce marqueur va être consommé par le *qui*, et le marqueur d'extraction qui arrivera au *que* ne pourrait être que 2, ce qui rend le terme intypable. A l'inverse le terme suivante reste toujours typable :

L'homme que ma soeur qui dort aime

$$\overbrace{le\ (que\ (\lambda x.\ aime\ \underbrace{x}_{\textbf{NP}\ 0\ acc}\ \overbrace{(ma\ (qui\ (\lambda y.\ \overbrace{dort\ \underbrace{y}_{\textbf{NP}\ 0\ nom}}^{\textbf{S}\ 0})\ soeur))}^{\textbf{N}\ 2\ nom})\ homme))}^{\textbf{S}\ 0}$$
$$\textbf{N}\ 2\ c$$

Enfin, on vérifie que ce groupe nominal n'est plus typable :

Pierre qui je crois que vient
$$\underbrace{qui_2(\lambda x.croire\ \underbrace{(que'(venir\ x))}_{\textbf{S}\ *\ 1}\ je)}_{\textbf{NP}\ 0\ nom \multimap \textbf{S}\ 1}\ Pierre$$

Le marqueur 0 de la variable trace devient 1 en passant par le *je crois que*. Le terme résultant n'a donc pas le bon type pour être argument de *qui*.

Pierre que je crois que Marie aime

a la même forme de dérivation de type, à la différence de *que* qui, lui, accepte ce type pour son argument.

Ce contrôle s'étend aisément aux autres pronoms relatifs et aux autres îlots d'extraction, comme les *si* ou les propositions infinitives. Les îlots lexicalisés, comme le verbe *hurler* ne posent pas plus de problèmes. De plus les autres phénomènes de mouvements explicites (interrogatives en *qu*, semi-clivées) peuvent utiliser le même marqueur et la même structure de contrôle. Notons enfin que l'on pourrait avoir démultiplié les entrées lexicales plutôt qu'utiliser les types dépendants, on est donc resté dans un fragment réversible des ACG.

5 Conclusion

Un atout des ACG est de proposer directement un contrôle sur les structures de dérivations, qui constituent notre langage abstrait. Il est ainsi possible de modéliser des phénomènes linguistiques sans quitter le formalisme ni le lambda-calcul typé. Cette technique de modélisation des contraintes à l'extraction propose une illustration pratique des possibilités offertes par l'extension du système de type des ACG.

D'autres phénomènes de déplacement peuvent être modélisés selon le même schéma. En particulier, les mouvements grammaticaux implicites, qui sont visibles au niveau de la réalisation sémantique, présentent des caractéristiques proches des mouvements présentés dans cet article, sur la nature des extractions (opérées dans ce cas par les prises de portée des quantificateurs) comme sur celle des îlots d'extraction (Ruys & Winter, 2008).

Références

ABEILLÉ A. (2002). *Une grammaire électronique du français*. CNRS Editions.

CARPENTER B. (1997). *Type-Logical Semantics*. The MIT Press.

CERVESATO I. & PFENNING F. (2002). A linear logical framework. *Information and Computation*, **179**(1), 19–75.

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. & MAAREK S. (2007). Type-theoretic extensions of abstract categorial grammars. In *New Directions in Type-Theoretic Grammars, proceedings of the workshop*, p. 18–30.

DE GROOTE P., YOSHINAKA R. & MAAREK S. (2007). On two extensions of abstract categorial grammars. In N. DERSHOWITZ & A. VORONKOV, Eds., *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, p. 273–287 : Springer.

MARTIN-LÖF P. (1973). An intuitionistic theory of types : Predicative part. In F.ROSE & J.SHEPERDSON, Eds., *Logic Colloquium '73*, p. 73–118.

MOORTGAT M. (1996). Categorial type logics. In J. VAN BENTHEM & A. TER MEULEN, Eds., *Handbook of Logic and Language*, p. 93–177. Amsterdam : Elsevier Science Publishers.

MUSKENS R. (2007). Separating syntax and combinatorics in categorial grammar. *Research on language and computation*.

PERRIER G. (2007). A french interaction grammar. In *6th International Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria, 2007.*, p. 463–467.

POLLARD C. (2007). The logic of pied-piping. Presentation at the Colloquium in Honour of Alain Lecomte.

POLLARD C. (2008). An introduction to convergernt grammar. Presentation at Calligramme Seminar.

RUYS E. & WINTER Y. (2008). Quantifier scope in formal linguistics. To appear in D. Gabbay (ed.), Handbook of Philosophical Logic - Second Edition.

SALVATI S. (2005). *Problèmes de filtrage et problèmes d'analyse pour les grammaires catégorielles abstraites*. PhD thesis, Institut National Polytechnique de Lorraine.