Traitements d'ellipses : deux approches par les grammaires catégorielles abstraites

Pierre Bourreau[1] *
(1) SFB 991
Institut für Sprache und Information
Université Heinrich-Heine, 40225 Düsseldorf
bourreau@hhu.de

RÉSUMÉ

L'étude de phénomènes d'ellipses dans les modèles de l'interface syntaxe-sémantique pose certains problèmes du fait que le matériel linguistique effacé au niveau phonologique est néanmoins présent au niveau sémantique. Tel est le cas d'une ellipse verbale ou d'une élision du sujet, par exemple, phénomènes qui interviennent lorsque deux phrases reliées par une conjonction partagent le même verbe, ou le même sujet. Nous proposons un traitement de ces phénomènes dans le formalisme des grammaires catégorielles abstraites selon un patron que nous intitulons extraction/instanciation et que nous implémentons de deux manières différentes dans les ACGs.

ABSTRACT

Treating ellipsis : two abstract categorial grammar perspectives

The treatment of ellipsis in models of the syntax-semantics interface is troublesome as the linguistic material removed in the phonologic interpretation is still necessary in the semantics. Examples are particular cases of coordination, especially the ones involving verbal phrase ellipsis or subject elision. We show a way to use abstract categorial grammars so as to implement a pattern we call extraction/instantiation in order to deal with some of these phenomena ; we exhibit two different constructions of this principle into ACGs.

MOTS-CLÉS : ellipse, coordination, interface syntaxe-sémantique, grammaires catégorielles abstraites, grammaires d'arbres adjoints, grammaires IO d'arbres.

KEYWORDS: ellipsis, coordination, syntax-semantics interface, abstract categorial grammars, tree-adjoining grammars, IO tree-grammars.

∗. Ce travail a été financé par la DFG, dans le cadre du projet SFB 991 "Die Struktur von Repräsentationen in Sprache, Kognition und Wissenschaft".

1 Introduction

La description de la syntaxe du langage naturel par le biais de formalismes symboliques a donné lieu à la création de nombreux modèles tels que les grammaires non-contextuelles, comme première approximation, et plus récemment les grammaires catégorielles combinatoires (CCGs pour *combinatory categorial grammars*) (Steedman, 1987) ou les grammaires d'arbres adjoints (Joshi *et al.*, 1975; Joshi, 1985) (TAGs pour *tree-adjoining grammars*). Tous ces formalismes partagent la propriété de ne pas effacer, et de ne pas copier de matériel syntaxique ou phonologique : nous parlerons de propriété de linéarité. Cependant, certains phénomènes syntaxiques usuels semblent nécessiter des mécanismes de copie et/ou d'effacement :

(1) Marie mange une pizza, et Pierre ϵ des pâtes.
(2) Jean fait un footing et ϵ rattrape Marie.
(3) Jean prépare ϵ et Marie vend des crêpes.

Sur les exemples ci-dessus, des éléments phonologiques sont absents par économie de langage : le verbe "mange" en (1), et les syntagmes "Jean" en (2) et "des crêpes" en (3). Tous ces éléments sont néanmoins présents au niveau de l'arbre syntaxique (pour la correction grammaticale) ou de la sémantique. Par ailleurs, ces trois exemples partagent la présence de la conjonction de coordination "et" reliant deux phrases. Nous nous intéressons au traitement de ces phénomènes d'ellipse sous la présence de marqueurs de coordination.

Plusieurs solutions à ce problème ont été proposées afin d'étendre les formalismes grammaticaux cités ci-dessus. Ainsi, (Steedman, 1990) montre comment traiter de telles coordinations dans les CCGs ; ces idées ont ensuite été implémentées par (Sarkar et Joshi, 1996) dans les TAGs, en étendant le formalisme initial afin d'enrichir les arbres de dérivation par une notion de partage de noeuds, idée ensuite reprise dans (Seddah, 2008; Seddah *et al.*, 2010). Enfin, (Kobele, 2007) propose l'utilisation de grammaires non-contextuelles d'arbres avec copie IO (notées IO-CFTGs pour *IO context-free tree grammars*).

Nous proposons d'utiliser un formalisme plus expressif que les précédents, à savoir les grammaires catégorielles abstraites (ACGs pour *abstract categorial grammars*) (de Groote, 2001; Muskens, 2001). Il est en effet possible d'encoder des grammaires de chaînes ou des grammaires d'arbres dans les ACGs. Qui plus est, la notion de dérivation y est également relativement flexible puisqu'il est possible de considérer non seulement des arbres mais aussi des λ-termes comme structures de dérivation. En utilisant ces avantages, nous implémentons le principe suivant : une phrase où une ellipse intervient est d'abord partiellement construite, en omettant le constituant commun, qui est rajouté lors de l'étape suivante. Ce principe peut être naturellement réalisé dans le λ-calcul par le biais de la substitution de termes. En suivant ce principe, nous présentons deux méthodes, la première faisant intervenir la substitution au niveau des structures de dérivation (ou tectogrammaire), la seconde au niveau de la syntaxe (ou phénogrammaire). Nous discutons des avantages de chacune des deux méthodes, et en particulier de l'existence d'algorithmes d'analyse s'exécutant en temps polynomial pour chacune d'elles.

Le reste de cet article est structuré comme suit : en section 2, nous présentons les ACGs. En section 3, les deux approches que nous proposons seront détaillées et discutées. Enfin, en section 4, nous comparerons notre solution à celles existantes dans la littérature.

2 Grammaires Catégorielles Abstraites

Les grammaires catégorielles abstraites peuvent être vues comme des grammaires de λ-termes simplement typés. Étant donné un ensemble de types atomiques $\mathscr{A}$, nous définissons l'ensemble $\mathscr{T}(\mathscr{A})$ des types simples sur $\mathscr{A}$ par

$$\mathscr{T}(\mathscr{A}) ::= \mathscr{A} | (\mathscr{T}(\mathscr{A}) \rightarrow \mathscr{T}(\mathscr{A}))$$

Nous adopterons la notation usuelle permettant d'omettre certaines parenthèses : un type $(\alpha_1 \rightarrow (\alpha_2 \rightarrow \alpha_3))$ sera noté $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3$.

Une *signature d'ordre supérieur* est un tuple $\Sigma = (\mathscr{A}, C, \tau)$ où :
– $\mathscr{A}$ est un ensemble fini de types atomiques.
– C est un ensemble fini de constantes.
– τ est une fonction d'assignation de types de C dans $\mathscr{T}(\mathscr{A})$.
Afin de construire des termes sur une telle signature, nous nous donnons un ensemble de variables typées : la notation x^α désignera une variable x de type α. Étant donnés une signature $\Sigma = (\mathscr{A}, C, \tau)$ et un type $\alpha \in \mathscr{T}(\mathscr{A})$, l'ensemble $\Lambda_\alpha(\Sigma)$ des λ-termes de type α dans Σ se définit par induction :

1. une variable x^α appartient à $\Lambda_\alpha(\Sigma)$.
2. une constante $\mathbf{c}$ de C appartient à $\Lambda_\alpha(\Sigma)$ si $\tau(\mathbf{c}) = \alpha$.
3. si M est un terme de $\Lambda_{\alpha_2}(\Sigma)$ et si $\alpha = \alpha_1 \rightarrow \alpha_2$, alors $\lambda x^{\alpha_1}.M$ est un terme de $\Lambda_\alpha(\Sigma)$.
4. si M_1 appartient à $\Lambda_\beta(\Sigma)$ et M_2 à $\Lambda_{\beta\rightarrow\alpha}(\Sigma)$, alors $(M_1 M_2)$ appartient à $\Lambda_\alpha(\Sigma)$.

L'ensemble des termes simplement typés de Σ est donné par $\Lambda(\Sigma) = (\Lambda_\alpha(\Sigma))_{\alpha\in\mathscr{T}(\mathscr{A})}$. Nous adopterons la convention usuelle suivante : un terme $(\ldots((M_1 M_2)M_3)\ldots M_n)$ sera écrit $M_1 M_2 M_3 \ldots M_n$. De plus, nous omettrons d'écrire les types des variables lorsqu'ils ne sont pas indispensables à la compréhension. Nous supposerons que les notions de variables libres et de β-réduction sont connues ; nous noterons $FV(M)$ l'ensemble des variables libres d'un terme M ; $M_1 \rightarrow_\beta^* M_2$ la β-réduction d'un terme M_1 en M_2 en un nombre arbitraire de β-contractions, et $|M|_\beta$ la forme β-normale d'un terme simplement typé. Pour plus de détails sur le λ-calcul simplement typé, le lecteur peut se référer à (Hindley, 1997).

L'ensemble $\mathcal{L}in_\alpha(\Sigma)$ des termes linéaires de type α dans Σ est défini par induction sur les règles 1. et 2. ci-dessus (en remplaçant $\Lambda_\alpha(\Sigma)$ par $\mathcal{L}in_\alpha(\Sigma)$) et :

3'. si M est un terme de $\mathcal{L}in_{\alpha_2}(\Sigma)$, si $\alpha = \alpha_1 \rightarrow \alpha_2$ et si $x^{\alpha_1} \in FV(M)$, alors $\lambda x^{\alpha_1}.M$ est un terme de $\mathcal{L}in_\alpha(\Sigma)$.

4'. si M_1 appartient à $\mathcal{L}in_\beta(\Sigma)$, M_2 à $\mathcal{L}in_{\beta\rightarrow\alpha}(\Sigma)$ et que $FV(M_1) \cap FV(M_2) = \emptyset$, alors $(M_1 M_2)$ appartient à $\mathcal{L}in_\alpha(\Sigma)$.

L'ensemble $\mathcal{QAff}_\alpha(\Sigma)$ des termes quasi-affines de type α dans Σ est construit par induction sur les règles 1., 2., 3. et la règle suivante :

4". si M_1 appartient à $\mathcal{QAff}_\beta(\Sigma)$, M_2 à $\mathcal{QAff}_{\beta\rightarrow\alpha}(\Sigma)$ et que pour toute variable $x^\beta \in FV(M_1) \cap FV(M_2)$, $\beta \in \mathscr{A}$, alors $(M_1 M_2)$ appartient à $\mathcal{QAff}_\alpha(\Sigma)$.

L'ordre $\mathtt{ord}(\alpha)$ d'un type α se définit par induction sur α : si $\alpha \in \mathscr{A}$ alors $\mathtt{ord}(\alpha) = 1$; sinon, $\alpha = \alpha_1 \rightarrow \alpha_2$ et $\mathtt{ord}(\alpha) = max(\mathtt{ord}(\alpha_1) + 1, \mathtt{ord}(\alpha_2))$. Par extension, l'ordre d'une signature $\Sigma = (\mathscr{A}, C, \tau)$ se définit comme : $\mathtt{ord}(\Sigma) = max_{\mathbf{c}\in C}(\mathtt{ord}(\tau(\mathbf{c})))$.

Remarquons qu'il est possible de voir une signature d'arbre comme une signature d'ordre 2 : dans une telle signature, tout terme M de type atomique et tel que $FV(M) = \emptyset$ peut effectivement être interprété comme un arbre. Par exemple, l'arbre $f(a, b, g(c))$ peut être représenté par le terme $f^{o\to o\to o\to o} a^o b^o (g^{o\to o} c^o)$, où o est un type atomique. De plus, si toutes les constantes d'une signature d'ordre 2 sont de type $o \to o$ (où o est un type atomique), les termes de la signature peuvent être interprétés comme des chaînes : la chaîne "Jean mange une pomme" est ainsi représentée par le terme $\lambda x^o.\mathbf{Jean}(\mathbf{mange}(\mathbf{une}(\mathbf{pomme}\ x)))$.

Étant données deux signatures $\Sigma_1 = (\mathscr{A}_1, C_1, \tau_1)$ et $\Sigma_2 = (\mathscr{A}_2, C_2, \tau_2)$, un *morphisme* $\mathscr{H}$ de Σ_1 vers Σ_2 est défini à partir d'un couple de fonctions $[\mathscr{H}_1; \mathscr{H}_2]$ vérifiant :
– étant donné un type $\alpha \in \mathscr{T}(\mathscr{A}_1)$:
  – $\mathscr{H}(\alpha) = \mathscr{H}_2(\alpha) \in \mathscr{T}(\mathscr{A}_2)$ si α appartient à $\mathscr{A}_1$.
  – $\mathscr{H}(\alpha) = \mathscr{H}(\alpha_1) \to \mathscr{H}(\alpha_2)$ si $\alpha = \alpha_1 \to \alpha_2$.
– étant donné un terme M de $\Lambda(\Sigma_1)$:
  – si $M = x^\alpha$, $\mathscr{H}(M) = x^{\mathscr{H}(\alpha)}$;
  – si $M = \mathbf{c} \in C_1$ et $\tau_1(\mathbf{c}) = \alpha$, $\mathscr{H}(M) \in \Lambda_{\mathscr{H}(\alpha)}(\Sigma_2)$;
  – si $M = \lambda x^\alpha.N$, $\mathscr{H}(M) = \lambda\mathscr{H}(x^\alpha).\mathscr{H}(N)$.
  – si $M = M_1 M_2$, alors $\mathscr{H}(M) = \mathscr{H}(M_1)\mathscr{H}(M_2)$.

Finalement, une ACG G est définie comme un tuple $(\Sigma_1, \Sigma_2, \mathscr{H}, s)$ où Σ_1 et Σ_2 sont deux signatures d'ordre supérieur (appelées respectivement signature abstraite et objet de G), $\mathscr{H}$ est un morphisme de Σ_1 vers Σ_2, et s est un type atomique de Σ_1. Une telle grammaire définit deux langages : un langage abstrait $A(G) = \{M \in \mathcal{L}in_s(\Sigma_1) \mid FV(M) = \emptyset\}$; un langage objet $O(G) = \{M \in \Lambda(\Sigma_2) \mid \exists N \in A(G), |\mathscr{H}(N)|_\beta = M\}$. De manière informelle, le langage abstrait correspond à l'ensemble des dérivations du langage $O(G)$ généré par la grammaire.

Une ACG G est d'ordre $n \in \mathbb{N}$ si la signature abstraite de G est d'ordre n (nous écrirons que G est une n-ACG) ; de plus, une n-ACG est dite linéaire (*resp.* quasi-affine) si pour toute constante $\mathbf{c}$ de la signature abstraite de G, $\mathscr{H}(\mathbf{c})$ est un terme linéaire (*resp.* quasi-affine).

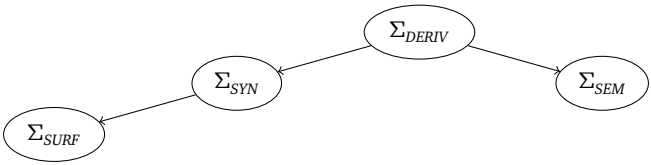

FIGURE 1 – Exemple de modélisation de l'interface syntaxe-sémantique par des ACGs

Grâce au pouvoir expressif du λ-calcul, (de Groote, 2002) et (de Groote et Pogodalla, 2004) ont montré qu'il est possible d'encoder de nombreux formalismes grammaticaux, dont les TAGs, comme des 2-ACG linéaires. Par ailleurs, lorsqu'on adopte l'hypothèse de compositionnalité, il est possible de représenter l'interface syntaxe-sémantique par l'intermédiaire de deux ACGs $G_1 = (\Sigma_{DERIV}, \Sigma_{SYN}, \mathscr{H}_{SYN}, s)$ et $G_2 = (\Sigma_{DERIV}, \Sigma_{SEM}, \mathscr{H}_{SEM}, s)$ (voire (de Groote, 2001; Pogodalla, 2004, 2007) pour plus de détails). Un des avantages de ce modèle est donc de représenter syntaxique et sémantique en parallèle, tout en traitant certains problèmes à des niveaux différents (par exemple, l'ordonnancement des mots peut être traité au niveau de la syntaxe, voire de la réalisation de surface, et non pas au niveau des dérivations). De plus, il est facile d'isoler les différentes représentations d'une phrase, tel que montré sur la figure 1.

3 Extraction et instanciation

Comme énoncé en introduction, nous souhaitons séparer la construction de phrases avec ellipses en deux étapes : la première consiste à construire une représentation incomplète ; la seconde à instancier cette représentation à l'aide de l'élément partagé. Au niveau de la représentation de surface seule une de ces occurrences sera réalisée. Ainsi, pour la phrase "*Jean mange une pizza et Pierre, des pâtes*", nous pouvons considérer que nous avons deux constituants incomplets, "*Jean* ϵ *une pizza*" et "*Pierre* ϵ *des pâtes*" qui sont reliés par la conjonction "et". Le verbe "mange" est ensuite rajouté à chacun de ces deux constituants, bien que non-réalisé phonétiquement pour le second. Le fait de garder une copie du constituant commun est nécessaire dans les ACGs puisque les dérivations des représentations syntaxiques et sémantiques sont symétriques, et qu'une copie de la forme sémantique du constituant commun est nécessaire, comme illustré dans la formule logique $(\exists x.\mathbf{Pizza}(x) \wedge \mathbf{Mange}(x,\mathbf{Jean}) \wedge (\exists y.\mathbf{PlatPates}(y) \wedge \mathbf{Mange}(y,\mathbf{Pierre}))$ représentant la sémantique de la phrase ci-dessus. La modélisation du principe extraction/instanciation est réalisée de manière relativement naturel dans le λ-calcul simplement typé. En effet, un objet incomplet peut être représenté par un terme de la forme $\lambda x.M$, où x est une variable dont les occurrences libres dans M représentent des emplacements vides de l'objet M. L'instanciation de ces emplacements par un objet N est ensuite simplement réalisée par application dans le λ-calcul et le terme $(\lambda x.M)N$ est donc l'objet M où les occurrences (libres) de x (dans M) sont substitués par N.

3.1 Enrichir les structures dérivationelles

Dans ce premier modèle, nous montrons comment la construction de constituants incomplets peut se réaliser au niveau de la signature des dérivations. Cette approche nous amène à écrire des ACGs dont l'ordre est supérieur à 2 ; en effet, à une phrase à laquelle il manque un verbe transitif sera associée une dérivation de la forme $\lambda x.M$, de type $(np \to np \to s) \to s$; la conjonction de deux phrases incomplètes implique de prendre deux termes de ce type en argument, et donc de manipuler des constantes d'ordre 4.

En guise d'exemple, nous considérons la grammaire suivante $G_{SURF} = (\Sigma_{DERIV}, \Sigma_{SURF}, \mathscr{H}_{SURF}, s)$, afin d'illustrer la modélisation d'ellipses verbales :

- $\Sigma_{DERIV} = \begin{cases} \mathscr{A}_{DERIV} = \{np, s\} \\ c_{Jean}, c_{Luc}, c_{Pierre}, c_{Mohamed} : np \quad c_{aime} : np \to np \to s \\ c_{et-TVel} : \alpha \to \alpha \to \alpha \qquad (\text{où } \alpha = (np \to np \to s) \to s) \end{cases}$
- $\Sigma_{SURF} = \{\mathscr{A}_{SURF} = \{\sigma\} \quad \mathbf{Jean}, \mathbf{Luc}, \mathbf{Pierre}, \mathbf{Mohamed}, \mathbf{aime}, \mathbf{et} : \sigma \to \sigma$
- $\mathscr{H}_{SURF} = \begin{cases} np, s := \sigma \to \sigma \text{ (noté } \sigma^2) \\ c_{Jean} := \lambda x^{\sigma}.\mathbf{Jean}x \qquad c_{Luc} := \lambda x^{\sigma}.\mathbf{Luc}x \\ c_{Mohamed} := \lambda x^{\sigma}.\mathbf{Mohamed}x \qquad c_{Pierre} := \lambda x^{\sigma}.\mathbf{Pierre}x \\ c_{aime} := \lambda P^{\sigma^2} Q^{\sigma^2} x^{\sigma}.Q(\mathbf{aime}(Px)) \\ c_{et-TVel} := \lambda P^{\beta \to \sigma^2} Q^{\beta \to \sigma^2} R^{\beta} x^{\sigma}.PR(\mathbf{et}(Q(\lambda S_1^{\sigma^2} S_2^{\sigma^2} y^{\sigma}.S_2(S_1 y))x)) \end{cases}$

  (où β désigne le type $\sigma^2 \to \sigma^2 \to \sigma^2$)

Le terme $M_{DERIV} = c_{et-TVel}(\lambda P^{np\to np\to s}.Pc_{Luc}c_{Jean})(\lambda P^{np\to np\to s}.Pc_{Mohamed}c_{Pierre})c_{aime}$ appartient à $\Lambda^s(\Sigma_{DERIV})$. De plus, il est possible de vérifier que $\mathscr{H}_{SURF}(M_{DERIV})$ se β-réduit en $\lambda x.\mathbf{Jean}(\mathbf{aime}(\mathbf{Luc}(\mathbf{et}(\mathbf{Pierre}(\mathbf{Mohamed}x)))))$. L'ACG ainsi obtenue est une 4-ACG linéaire.

Cette construction peut s'étendre à d'autres types d'ellipses, tels que les ellipses du sujet ou de l'objet : il suffit alors de rajouter des constantes c_{et-Sel} et c_{et-Oel} de type $(np\to s)\to(np\to s)\to np\to s$ dans Σ_{DERIV}. Comme alternative, nous pouvons envisager la généralisation de cette constante à un type $X\to X\to X$ tel que proposer dans (Steedman, 1990).

Afin de construire la représentation sémantique de cet exemple, il nous suffit de créer une seconde ACG $G_{SEM} = (\Sigma_{DERIV}, \Sigma_{SEM}, \mathscr{H}_{SEM}, s)$ comme suit :

– $\Sigma_{SEM} = \begin{cases} \mathscr{A}_{SEM} = \{e, t\} \\ \mathbf{J}, \mathbf{L}, \mathbf{P}, \mathbf{M} : e \qquad \mathbf{A} : e \to e \to t \\ \wedge : t \to t \to t \end{cases}$

– $\mathscr{H}_{SEM} = \begin{cases} np := (e\to t)\to t \text{ (noté } \gamma) & s := t \\ c_{Jean} := \lambda P^{e\to t}.P\mathbf{J} & c_{Luc} := \lambda P^{e\to t}.P\mathbf{L} \\ c_{Mohamed} := \lambda P^{e\to t}.P\mathbf{M} & c_{Pierre} := \lambda P^{e\to t}.P\mathbf{P} \\ c_{aime} := \lambda P^{(e\to t)\to t}Q^{(e\to t)\to t}.P(\lambda x^e.Q(\lambda y^e.\mathbf{A}xy) \\ c_{et-TVel} := \lambda P_1^{(\gamma\to\gamma\to t)\to t}P_2^{(\gamma\to\gamma\to t)\to t}R^{\gamma\to\gamma\to t}.\wedge(P_1R)(P_2R) \end{cases}$

Cette construction nous permet d'obtenir une 4-ACG $(\Sigma_{DERIV}, \Sigma_{SEM}, \mathscr{H}_{SEM}, s)$. Nous remarquerons, néanmoins, que cette dernière n'est ni linéaire, ni quasi-affine : en effet, la variable R a deux occurrences libres dans un sous-terme de $\mathscr{H}_{SEM}(c_{et-TVel})$.

Commentaires :

Les deux ACGs ainsi construites sont donc des n-ACGs où $n > 2$; ceci soulève un des inconvénients de cette méthode, puisque nous savons que, dans ce cas, le problème de l'appartenance est un problème NP-complet (Kanazawa et Yoshinaka, 2005a), lorsque l'ACG est linéaire.

Du point de vue de la modélisation linguistique, notons qu'il est possible de traiter des cas d'ellipses multiples d'un même constituant sans modifier notre modèle. Ainsi, afin de pouvoir dériver la phrase : "Jean aime Luc, Pierre, Mohamed et Paul, Valérie.", nous rajoutons la constante $c_{,TV-el}$ de type $\alpha\to\alpha\to\alpha$ (où $\alpha = (np\to np\to s)\to s$) à Σ_{DERIV} et telle que $\mathscr{H}_{SURF}(c_{,Tv-el}) = \lambda P^{\beta\to\sigma^2}Q^{\beta\to\sigma^2}R^{\beta}x^{\sigma}.PR(,(Q(\lambda S_1^{\sigma^2}S_2^{\sigma^2}y^{\sigma}.S_2(S_1y))x))$ (en considérant les notations de types ci-dessus). Il est intéressant de remarquer que nous obtenons alors deux termes M_1 et M_2 dans $\Lambda_s(\Sigma_{DERIV})$ tels que $|\mathscr{H}_{SURF}(M_1)|_\beta$ et $|\mathscr{H}_{SURF}(M_2)|_\beta$ sont égaux à $\lambda x.\mathbf{Jean}(\mathbf{aime}(\mathbf{Luc}(,(\mathbf{Pierre}(\mathbf{Mohamed}(\mathbf{et}(\mathbf{Paul}(\mathbf{Valérie}x)))))))))$. Ces deux termes sont :

1. $M_1 = c_{,TV-el}(\lambda P.Pc_{Luc}c_{Jean})(\lambda Q.c_{et-TVel}(\lambda R.Rc_{Mohamed}c_{Pierre})(\lambda R.Rc_{Valerie}c_{Paul})Q)c_{aime}$ correspondant à la dérivation de la phrase pour le parenthésage "[Jean aime Luc, [Pierre, Mohamed et Paul, Valérie]]" ;

2. $M_2 = c_{et-TVel}(\lambda Q.c_{,TV-el}(\lambda P.Pc_{Luc}c_{Jean})(\lambda P.Pc_{Mohamed}c_{Pierre})Q)(\lambda R.Rc_{Valerie}c_{Paul})c_{aime}$ correspondant à la dérivation de notre exemple pour le parenthésage "[[Jean aime Luc, Pierre, Mohamed] et Paul, Valérie]" ;

Par ailleurs, remarquons que le verbe n'est réalisé au niveau de la surface que dans le premier constituant gauche dominé par la coordination ; dans le second cas, il est remonté jusqu'au constituant correct de manière transitive. La grammaire reste alors une 4-ACG linéaire.

Cette construction peut s'étendre à l'analyse de phénomènes d'ellipses enchâssées comme dans la phrase suivante en Anglais :

(4) After seeing John running a marathon, Paul planned to ϵ_1, but Mary didn't ϵ_2.
Après avoir vu John courir un marathon, Paul a prévu de le faire, mais pas Marie.

En simplifiant quelque peu la dérivation syntaxique, cette phrase est traitée dans notre modèle par l'intermédiaire d'un terme $c_{after}M_1(\lambda P.c_{but}M_1'M_2'(c_{planned-to}P))c_{run}$; le morphisme est ensuite construit en suivant l'exemple précédent.

Il est important de remarquer la similitude entre cette construction et certains travaux antérieurs sur les ACGs. En effet, cette méthode repose sur le fait de retarder la concaténation de chaînes, de la même manière que (Pogodalla, 2007) utilise des ACGs d'ordre supérieur au niveau des dérivations afin de retarder l'ajout de matériel linguistique, permettant ainsi de modéliser les différentes portées des quantificateurs dans la représentation sémantique.

Remarquons enfin, que nous avons modélisé la réalisation de surface sans décrire la réalisation de l'arbre syntaxique ; cette construction ne nous apporte effectivement aucune information supplémentaire sur l'analyse de cette première modélisation.

Par ces divers exemples, nous montrons qu'il est possible de modéliser divers phénomènes d'ellipses de manière simple et élégante dans les ACGs, sans modifier le formalisme. Notre construction repose uniquement sur le fait de considérer des termes d'ordre supérieur au niveau des dérivations. Néanmoins, l'inconvénient d'une telle construction est que le traitement de ces phénomènes ne peut plus être réalisé en temps polynomial. Nous montrons à présent qu'une solution possible à ce problème consiste à considérer des ACGs d'ordre 2, et à enrichir le typage de la signature des dérivations, plutôt que la structure des termes.

3.2 Enrichir les types des dérivations

Dans cette seconde approche, nous construisons des modèles de représentation de la structure de surface à partir des structures de dérivation de manière indirecte, par l'intermédiaire des structures syntaxiques arborescentes. Ceci nous permettra, en particulier, de mettre en avant la complexité des morphismes utilisés, cette propriété ayant un impact sur la complexité de l'analyse dans les ACGs [1].

Pour ce faire, nous introduisons un opérateur **DEL** d'effacement au niveau des arbres syntaxiques, à la manière de (Kobele, 2007) ou de l'opérateur de "*deanchoring*" sur les structures de dérivation dans (Lichte et Kallmeyer, 2010). Dans notre cas, un sous-arbre dominé par cet opérateur sera interprété comme la chaîne vide ϵ au niveau de la représentation de surface. Cet opérateur n'est donc pas indispensable à notre modèle, mais nous permet néanmoins de faire apparaître l'élément effacé dans l'arbre syntaxique.

Nous donnons un exemple d'un tel arbre en Figure 2, dérivé par la grammaire $G_{SURF} = (\Sigma_{SYN}, \Sigma_{SURF}, \mathcal{H}_{SURF}, o)$ définie par :

1. Nous aurions pu procéder de la même manière à l'étape précédente, mais les ACGs étant alors d'ordre 3, l'analyse n'est, a priori, déjà plus réalisable en temps polynomial

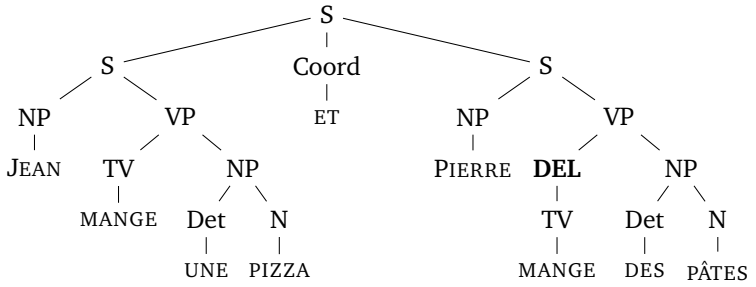

FIGURE 2 – Arbre dérivé pour la phrase "*Jean mange une pizza et Pierre, des pâtes.*"

$$
\begin{aligned}
&- \ \Sigma_{SYN} = \begin{cases} \mathscr{A}_{SYN} = \{o\} \\ S_{Conj} : o \to o \to o \to o \qquad NP_1, N, Det, Coord, TV, \mathbf{DEL} : o \to o \end{cases} \\
&- \ \Sigma_{SURF} = \begin{cases} S, NP_2, VP : o \to o \to o \quad \text{JEAN, MANGE, PIERRE, UNE, DES, PIZZA, PÂTES, ET} : o \\ \mathscr{A}_{SURF} = \{\sigma\} \quad \mathbf{Jean}, \mathbf{mange}, \mathbf{une}, \mathbf{des}, \mathbf{pizza}, \mathbf{pâtes}, \mathbf{et} : \sigma \to \sigma \end{cases} \\
&- \ \mathscr{H}_{SURF} = \begin{cases} o := \sigma \to \sigma \\ S_{Conj} := \lambda P_1^{\sigma\to\sigma} P_2^{\sigma\to\sigma} P_3^{\sigma\to\sigma} x^\sigma . P_1(P_2(P_3 x)) \\ S, NP_2, VP := \lambda P_1^{\sigma\to\sigma} P_2^{\sigma\to\sigma} x^\sigma . P_1(P_2 x) \\ N, NP_1, Det, Coord, TV := \lambda P^{\sigma\to\sigma} x^\sigma . Px \qquad \mathbf{DEL} : \lambda P^{\sigma\to\sigma} x^\sigma . x \\ \text{JEAN} := \lambda x^\sigma . \mathbf{Jean} x, \text{PIERRE} := \lambda x^\sigma . \mathbf{Pierre} x \quad \text{MANGE} := \lambda x^\sigma . \mathbf{mange} x \\ \text{UNE} := \lambda x^\sigma . \mathbf{une} x, \text{DES} := \lambda x^\sigma . \mathbf{des} x \quad \text{PIZZA} := \lambda x^\sigma . \mathbf{pizza} x \\ \text{PÂTES} := \lambda x^\sigma . \mathbf{pâtes} x \quad \text{ET} := \lambda x^\sigma . \mathbf{et} x \end{cases}
\end{aligned}
$$

Nous remarquerons que l'opérateur **DEL** réalise l'effacement au niveau de la chaîne de caractères ; en effet, l'image de ce terme par le morphisme $\mathscr{H}_{SURF}$ est un terme quasi-affine, effaçant sur son premier argument. Ainsi, pour tout terme M, nous avons $\mathscr{H}_{SURF}(\mathbf{DEL}M) \to^*_\beta \lambda x.x$. Nous pouvons alors vérifier que le terme M_{SYN} correspondant à l'arbre de la figure 2 vérifie $\mathscr{H}_{SURF}(M_{SYN}) \to^*_\beta \lambda x.\mathbf{Jean}(\mathbf{mange}(\mathbf{une}(\mathbf{pizza}(\mathbf{et}(\mathbf{Pierre}(\mathbf{des}(\mathbf{pâtes}x)))))))$ [2]. Par ailleurs, l'ACG ainsi présentée n'est pas lexicalisée : l'image de certaines constantes de Σ_{SYN} par $\mathscr{H}_{SURF}$ ne contient pas de constantes. Néanmoins, nous savons qu'il est possible de construire une 2-ACG lexicalisée générant le même langage (Kanazawa et Yoshinaka, 2005b).

Nous décrivons à présent, une seconde implémentation du principe d'extraction/instanciation, en créant de nouveaux types dans la signature des dérivations : le fait qu'un constituant d'une certaine catégorie syntaxique soit incomplet pour une autre catégorie syntaxique sera effectivement dénoté par un type distinct.

En reprenant l'exemple de la figure 2, nous souhaitons donc pouvoir dériver un terme de la forme $\lambda x^o.M_1$ et un terme $\lambda x^o.M_2$ représentant chacun les contextes d'arbre pour "Jean x une pizza" et pour "Pierre x des pâtes", sachant que pour ce dernier, l'occurrence de x est dominée par une occurrence de l'opérateur **DEL**. Nous créons donc un type (noté s_{TVel}) pour désigner les contextes d'arbre sur un verbe transitif, au niveau des dérivations. De plus,

2. Il est possible de lexicaliser $\mathscr{H}_{SURF}(\mathbf{DEL})$, en $\lambda Px.,x$ par exemple, de manière à faire apparaître le signe de ponctuation ",".

$\mathcal{H}_{SYN}(s_{TVel}) = o \to o$, afin de rendre compte du fait que la dérivation d'un terme de type s_{TVel} est un contexte d'arbre, tel que nous le codons dans le λ-calcul.

Une constante c_{et} est ensuite nécessaire à Σ_{DERIV} afin de réaliser l'étape d'instanciation, mais cette fois au niveau des termes des arbres syntaxiques ; il suffit donc de typer cette constante par $s_{TVel} \to s_{TVel} \to tv \to s$, un type d'ordre 2. On notera que pour ce faire, nous modifions le type associé aux verbes transitifs de $np \to np \to s$ en tv. Intuitivement, ceci revient à associer à un verbe le plus grand sous-arbre dont l'unique racine est la réalisation phonologique associée au verbe.

Afin d'illustrer notre proposition, nous donnons la grammaire $G_{SYN} = (\Sigma_{DERIV}, \Sigma_{SYN}, \mathcal{H}_{SYN}, s)$ définie ci-dessous. Afin de mieux dissocier les deux étapes de notre méthode, nous isolons l'étape d'instanciation par l'intermédiaire d'une constante distincte, c_{SUB}, le type de la variable c_{et} s'en trouvant alors modifié :

– $\Sigma_{DERIV} = \begin{cases} \mathcal{A}_{DERIV} = \{s, s_{TVel}, vp_{TVel}, n, np, v\} & \\ c_{et} : s_{TVel} \to s_{TVel} \to s_{TVel} & c_{SUB} : s_{TVel} \to tv \to s \\ c_1 : np \to vp_{TVel} \to s_{TVel} & c_2 : np \to vp_{TVel} \\ c_3 : n \to det \to np & \\ c_{Jean}, c_{Pierre} : np & c_{mange} : tv \\ c_{une}, c_{des} : det & c_{pizza}, c_{pates} : n \end{cases}$

– $\mathcal{H}_{SYN} = \begin{cases} s, tv, n, np, det := o & s_{TVel}, vp_{TVel} := o \to o \\ c_{SUB} := \lambda P^{o\to o} x^o . Px & \\ c_{et} := \lambda P_1^{o\to o} P_2^{o\to o} x^o . S(P_1 x)(Coord\ \text{ET})(P_2(\mathbf{DEL}x)) & \\ c_1 := \lambda t^o P^{o\to o} x^o . St(Px) & c_2 := \lambda t^o x^o . VPxt \\ c_3 := \lambda t_1^o t_2^o . NP_2 t_2 t_1 & \\ c_{Jean} := NP_1 \text{JEAN}, c_{Pierre} := NP_1 \text{PIERRE} & c_{mange} : V\text{MANGE} \\ c_{une} := Det\text{UNE}, c_{des} := Det\text{DES} & \\ c_{pizza} : N\text{PIZZA}, c_{pates} : N\text{PÂTES} & \end{cases}$

En considérant la signature abstraite Σ_{DERIV}, nous obtenons un terme M_{deriv} appartenant au langage abstrait et tel que $M_{deriv} = c_{SUB} M_1 M_2$ où

1. $M_1 = c_{et}(c_1 c_{Jean}(c_2(c_3 c_{une} c_{pizza})))(c_1 c_{Pierre}(c_2(c_3 c_{des} c_{pates})))$ et
2. $M_2 = c_{mange}$.

Nous remarquerons que $\mathcal{H}_{SYN}(M_1)$ s'interprète alors comme un contexte d'arbre, de la forme $\lambda x^o . T$, T étant l'arbre de la figure 2 où les occurrences du sous-arbre VMANGE sont remplacées par x.

Commentaires

Tout d'abord, remarquons que l'ACG $G = (\Sigma_{DERIV}, \Sigma_{SYN}, \mathcal{H}_{SYN}, s)$ est une 2-ACG quasi-affine. D'après (Bourreau et Salvati, 2011; Bourreau, 2011) ou (Kanazawa, 2007; Yoshinaka, 2006), nous savons que le problème de l'analyse, dans ce cas, peut être résolu en temps polynomial. Ce point différencie donc les deux approches présentées. Ensuite, nous remarquerons qu'il est à nouveau possible de généraliser le type associé à la conjonction c_{et} au niveau des dérivations en $\alpha \to \alpha \to \alpha$, avec, $\alpha \in \mathcal{A}_{DERIV}$.

Néanmoins, comme nous l'avons remarqué, l'ACG $(\Sigma_{SYN}, \Sigma_{SURF}, \mathscr{H}_{SURF}, o)$ n'est pas lexicalisée. Le fait de considérer l'ACG lexicalisée équivalente $(\Sigma'_{SYN}, \Sigma_{SURF}, \mathscr{H}'_{SURF}, o)$ de (Kanazawa et Yoshinaka, 2005b) peut a priori avoir un certain impact sur la construction de l'ACG $(\Sigma_{DERIV}, \Sigma'_{SYN}, \mathscr{H}'_{SYN}, s)$, cette question demandant à être étudiée plus en détails.

Par ailleurs, le choix d'introduire l'opérateur **DEL** n'est destiné qu'à faire apparaître l'occurrence de constituant effacée dans l'arbre syntaxique. En effet, il est possible de modifier notre modèle de sorte que $\mathscr{H}_{SYN}(c_{et}) = \lambda P_1^{o\to o} P_2^{o\to o} x^o . S(P_1 x)(Conj\ \text{ET})(P_2 \epsilon)$, où ϵ est alors une constante de Σ_{SYN}, de type o et telle que $\mathscr{H}_{SURF}(\epsilon) = \lambda x^\sigma . x^\sigma$. Dans cette proposition alternative, l'ACG obtenue reste une 2-ACG linéaire.

La gestion de l'effacement par enrichissement des types peut également s'étendre à l'analyse de phénomènes d'ellipses enchâssées comme dans la phrase (4) de la section précédente. Un tel cas peut-être traité dans notre proposition en rajoutant une étape supplémentaire d'extraction/instanciation, par l'intermédiaire d'une constante c'_{SUB} de type $s_V \to vp_{VPinf} \to s_V$, et telle que $\mathscr{H}_{SYN}(c'_{SUB}) = \lambda P_1^{o\to o} P_2^{o\to o} t^o . P_1(P_2 t)$. L'utilisation d'une telle constante réalise alors l'instanciation d'emplacements vides dans un arbre de type s_V par un contexte d'arbre de type vp_{VPinf}. La dérivation de cet exemple est donc réalisée en construisant d'abord deux contextes d'arbre : Le terme $\mathscr{H}_{SYN}(c'_{SUB})$ permet alors de substituer les occurrences de x dans

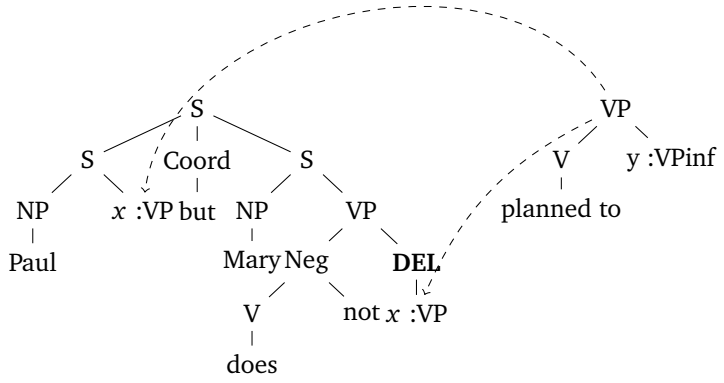

FIGURE 3 – Représentation de la dérivation pour "John planned to but Mary does not"

le premier arbre par le second ; il suffit ensuite de suivre la procédure sur notre exemple initiale pour obtenir l'arbre souhaité. Nous pouvons alors voir que l'inconvénient principal de cette méthode est de devoir créer de nombreux types afin de prendre en compte les différents cas d'ellipses possibles, selon le constituant effacé. Notons également qu'il est possible de construire la sémantique associée à une phrase où une ellipse a été réalisée, de manière similaire à la construction précédente. Finalement, il apparaît que les deux constructions présentées sont liées : notre deuxième proposition repose sur le fait de décomposer les arbres syntaxiques en unité plus petite, ce qui aboutit à considérer une ACG non-lexicalisée, et à considérer un nombre de types plus grand. Cependant, il nous faudra étudier ce lien, et la possibilité d'abaisser l'ordre d'une ACG tout en préservant le langage généré.

4 Méthodes existantes

Une première possibilité de traitement de certains phénomènes d'ellipse consiste à reprendre les idées de (Steedman, 1990) pour les grammaires catégorielles combinatoires, et à implémenter ces idées dans les ACGs. Steedman suggère, en particulier, l'utilisation d'un combinateur **T** de "type raising" afin de traiter des phénomènes d'élision du sujet ou de l'objet ; de plus, les ellipses verbales nécessitent l'introduction d'un combinateur supplémentaire **Bx** qui permet de rompre avec la directionnalité du calcul logique sous-jacent, et un opérateur de décomposition de type, permettant d'extraire le verbe d'une phrase.

Dans le cadre des ACGs, l'opérateur **Bx** n'est pas nécessaires, puisque les types ne sont pas dirigés. l'utilisation du combinateur **T** revient à modifier les types assignés aux constantes de la signature des structures de dérivation. Sur un exemple, nous pouvons décrire une signature des dérivations faite des constantes $c_{Jean}, c_{Marie} : (np \to s) \to s$, $c_{court} : np \to s$, $c_{rattrape} : np \to np \to s$ et $c_{et} : (np \to s) \to (np \to s) \to (np \to s)$. Cette signature permet de dériver un terme $c_{Marie}(\lambda y^{np}.c_{Jean}(c_{et}\, c_{court}(c_{rattrape}\, y)))$. Grâce au morphisme $\mathscr{H}_{sem}$ ci-dessous, il est ensuite possible d'associer la forme sémantique souhaitée pour la phrase "Jean court et rattrape Marie" :

– $\mathscr{H}_{sem} = \begin{cases} np := e & s := t \\ c_{Jean} := \lambda P^{e\to t}.P\mathbf{J} & c_{Marie} := \lambda P^{e\to t}.P\mathbf{M} \\ c_{rattrape} := \lambda x^e y^e.\mathbf{R}xy & c_{court} := \lambda x^e.\mathbf{C}x \\ c_{et} := \lambda P_1^{e\to t} P_2^{e\to t} x^e. \wedge (P_1 x)(P_2 x) & \end{cases}$

Nous remarquerons néanmoins que, la signature des dérivations que nous décrivons ci-dessus est d'ordre supérieur à 2. Par ailleurs, il ne semble pas souhaitable, dans le cas des ACGs, de typer tous les syntagmes nominaux par un type $(np \to s) \to s$, ce qui revient à considérer des ACGs d'ordre supérieur à 2 pour des cas très simples, sans phénomènes d'ellipses. Enfin, l'opérateur de décomposition est nécessaire dans le cas d'ellipses verbales pour les langues de type SVO, car il permet d'extraire le verbe de la phrase en partie gauche de la conjonction. Cet opérateur permet en fait d'effectuer le même traitement que nous réalisons, c.a.d. de construire des constituants incomplets puis de les composer avec le constituant commun. Qui plus est, cet opérateur de décomposition semble poser un problème du point de vue calculatoire car il introduit deux nouvelles formules, ce qui va à l'encontre de la propriété de la sous-formule. Enfin, notons que les ACGs permettent l'implémentation du même principe de manière plus élégante puisque, de par l'indépendance entre dérivations et ordre des mots, nous n'avons pas eu besoin d'enrichir le formalisme initial de nouveaux opérateurs.

Des extensions des TAGs ont également été proposées, tout d'abord dans (Sarkar et Joshi, 1996) qui proposent une implémentation des idées de (Steedman, 1990) dans les grammaires d'arbres adjoints, en y rajoutant une opération de conjonction. Par ailleurs, l'objectif des auteurs est de construire des structures dérivées qui sont des arbres avec partage de noeud. Qui plus est, ils rendent compte de ce partage de matériel syntaxique au niveau des dérivations, les structures de dérivation étant également des arbres avec partage de noeuds. Ceci est dû au fait que les structures de dérivation dans les TAGs sont censés être plus proches de la structure prédicat/argument de représentation sémantique d'une phrase. D'autres propositions sont celles de (Seddah, 2008) ou (Seddah *et al.*, 2010), qui considèrent des grammaires de tuples d'arbres et requièrent des opérations plus complexes ; par exemple, le traitement d'ellipses multiples se fait en ajoutant un nombre arbitraire d'arbres non-lexicalisés (appelés "ghost trees"

par les auteurs). L'originalité de notre méthode, par rapport à celles-ci, est de pouvoir traiter les phénomènes d'ellipses que nous avons étudiés sans modifier le formalisme des grammaires catégorielles abstraites. Par ailleurs, le modèle de l'interface syntaxe sémantique dans les ACGs permet de séparer explicitement les structures de dérivation, de la représentation sémantique. Le partage d'information nécessaire au niveau des dérivations dans les TAGs, est donné au niveau de la signature Σ_{sem} dans notre cas.

Enfin, (Kobele, 2007) décrit plusieurs méthodes possibles dont les deux suivantes : la première consiste à construire des contextes d'arbres, car du matériel syntaxique est absent aux emplacements où une ellipse a été réalisée ; l'information manquante doit alors être retrouvée dans l'arbre (dans le cas d'une ellipse verbale, dans le premier constituant dominé par une conjonction de coordination). La deuxième approche de (Kobele, 2007) consiste à utiliser des grammaires non-contextuelles d'arbres avec copie IO.

Les deux approches que nous proposons semblent assez proches des propositions de Kobele, à la différence que, plutôt que de rechercher le matériel effacé dans l'arbre, nous mettons en place un mécanisme permettant de le copier. Par ailleurs, les ACGs de notre seconde approche peuvent être réduites à des grammaires IO-CFTGs. Bien que le patron de dérivation ne soit pas le même que celui utilisé par Kobele, il semblerait que nous ne puissions pas traiter plus de phénomènes que dans son approche. En particulier, Kobele montre qu'une des limites de l'approche par des IO-CFTGs est de ne pas pouvoir traiter des phénomènes tels que :

(5). "John wants to climb Mt. Kilimanjaro and Mary to sail around the world, and while I know that John will ϵ_1 and Mary won't ϵ_2, Bill doesn't ϵ_3"

John veut grimper le Kilimanjaro et Marie naviguer autour du monde, et alors que je sais que John le fera et pas Marie, Bill ne le sait pas

D'après cette construction, il serait nécessaire de garder l'ensemble des verbes utilisés dans le constituant à gauche d'une conjonction afin de pouvoir le réutiliser dans les constituants en partie droite ; qui plus est, ce nombre de verbes est potentiellement infini, ce qui, dans notre première approche nous amène à considérer un nombre de constantes infinies c^n_{et}, $n \in \mathbb{N}$; dans notre seconde approche, il nous faudrait considérer un nombre de types infini dans la signature des dérivations. Ces cas d'ellipses mettent en avant la limite des traitements proposées. Par ailleurs, ce type d'ellipses paraît maladroit en Français, où les pronoms sont utilisés afin de se référer à un syntagme précédemment utilisé. Une solution à envisager est donc d'adapter des techniques de résolution d'anaphores, à partir de continuations dans le λ-calcul, par exemple (de Groote, 2006), afin de résoudre les phénomènes d'ellipse.

5 Conclusion

Les phénomènes d'ellipses sont fréquents dans le langage naturel et sont des exemples de phénomènes non-linéaires au niveau de l'interface syntaxe-sémantique. Nous avons proposé deux approches pour le traitement d'ellipses sous coordination dans les ACGs, en utilisant le principe d'extraction pour la construction d'une phrase incomplète, suivi d'un mécanisme d'instanciation, modélisé par la substitution dans le λ-calcul. Dans la première approche, ce principe est directement codé au niveau des termes des structures de dérivation ; de manière élégante, nous pouvons alors traiter de nombreux cas d'ellipses, mais la signature des dérivations étant d'ordre supérieur à 2, le problème de l'analyse est, au meilleur des

cas, NP-complet. Dans la deuxième approche, nous conservons une ACG d'ordre 2, mais les mécanismes d'extraction sont encodés au niveau des types utilisés dans la signature. Ceci nous amène alors à considérer un ensemble de types très grand, mais nous permet de réutiliser des algorithmes d'analyse connus pour s'exécuter en temps polynomial.

Les deux approches ainsi proposées ne nécessitent pas d'étendre le formalisme des ACGs, contrairement aux solutions proposées dans la littérature, pour les TAGs ou les CCGs. Néanmoins, les modélisations que nous proposons ne prétendent pas résoudre des phénomènes d'ellipses complexes, tels que les ellipses de verbes prenant différentes catégories en argument (dans "Jean est un républicain, et fier de l'être"), ou encore celles faisant intervenir un zeugma (dans "Napoléon a pris du poids et beaucoup de pays", discuté dans (Seddah, 2008)). Dans ce dernier cas, une piste est de tenter de distinguer deux signatures des dérivations $\Sigma_{DERIV-EXPR}$ et $\Sigma_{DERIV-STR}$ contrôlant les dérivations de l'arbre syntaxique, la première s'assurant de la construction d'expressions figées.

Par ailleurs, les modèles que nous proposons reposent essentiellement sur la présence d'une coordination dominant l'occurrence du syntagme effacé, et ne saurait résoudre des cas d'ellipses ou ce principe n'est pas vérifié. Enfin, et comme discuté dans (Kobele, 2007), les deux approches semblent trop limités afin de résoudre certains cas d'ellipses faisant intervenir de multiples verbes, et des méthodes de résolution d'anaphores pourrait se montrer plus efficaces.

Finalement, cette étude demande à être approfondie afin d'étudier plus en détails le lien entre les deux propositions présentées. En particulier, il serait intéressant de savoir quand, et à quel coût, il est possible de diminuer l'ordre d'une ACG tout en préservant le langage généré.

Remerciements : Je remercie les rapporteurs anonymes qui ont grandement aidé à l'amélioration de ce travail. Je tiens également à remercier Laura Kallmeyer et Timm Lichte pour les discussions qui m'ont amenées à m'intéresser à ce problème.

Références

BOURREAU, P. (2011). *Jeux de typage et analyse de λ-grammaires non-contextuelles*. Thèse de doctorat, Laboratoire Bordelais d'Informatique.

BOURREAU, P. et SALVATI, S. (2011). A Datalog recognizer for almost affine λ-CFGs. *In* (Kanazawa *et al.*, 2011), pages 21–38.

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*, pages 148–155.

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

de GROOTE, P. (2006). Towards a montagovian account of dynamics. *In Proceedings of Semantics and Linguistic Theory XVI*.

de GROOTE, P. et POGODALLA, S. (2004). On the expressive power of abstract categorial grammars : Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438.

HINDLEY, R. J. (1997). *Basic Simple Type Theory*. Cambridge Press University.

JOSHI, A. K. (1985). Tree-adjoining grammars : How much context-sensitivity is required to provide reasonable strucutral descriptions ? *Natural Language Parsing : Psychological, Computational and Theoretical Perspectives*, pages 206–250.

JOSHI, A. K., LEVY, L. S. et TAKAHASHI, M. (1975). Tree adjunct grammars. *Journal of Comput. Syst. Sci.*, 10(1):136–163.

KANAZAWA, M. (2007). Parsing and generation as Datalog queries. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 176–183, Prague. Association for Computational Linguistics.

KANAZAWA, M., KORNAI, A., KRACHT, M. et SEKI, H., éditeurs (2011). *The Mathematics of Language - 12th Biennal Conference, MOL 12, Nara, Japan, September 2011. Proceedings*, volume 6878 de *Lecture Notes in Artificial Intelligence*. Springer.

KANAZAWA, M. et YOSHINAKA, R. (2005a). The complexity and generative capacity of lexicalised abstract categorial grammars. *In* (Kanazawa *et al.*, 2011), pages 330–346.

KANAZAWA, M. et YOSHINAKA, R. (2005b). Lexicalization of second-order ACGs. Rapport technique NII-2005-012E, NII, National Institute of Informatics, Tokyo.

KOBELE, G. M. (2007). Parsing ellipsis. Unpublished Manuscript.

LICHTE, T. et KALLMEYER, L. (2010). Gapping through TAG derivations. *In Procedings of the 10th International Workshop on Tree-Adjoining Grammar and Related Formalisms*.

MUSKENS, R. (2001). Lambda Grammars and the Syntax-Semantics Interface. *In* van ROOY, R. et STOKHOF, M., éditeurs : *Proceedings of the Thirteenth Amsterdam Colloquium*, pages 150–155, Amsterdam.

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)*, pages 64–71.

POGODALLA, S. (2007). Generalizing a proof-theoretic account of scope ambiguity. *In proceedings of IWCS-7*.

SARKAR, A. et JOSHI, A. (1996). Coordination in tree adjoining grammars : formalization and implementation. *In Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 610–615, Stroudsburg, PA, USA. Association for Computational Linguistics.

SEDDAH, D. (2008). The use of MCTAG to process elliptic coordination. *In Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms*, TAG+9.

SEDDAH, D., SAGOT, B. et DANLOS, L. (2010). Control verb, argument cluster coordination and multi component TAG. *In Proceedings of the 10th International Conference on Tree Adjoining Grammars and Related Formalisms*, TAG+10.

STEEDMAN, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5:403–439.

STEEDMAN, M. (1990). Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–264.

YOSHINAKA, R. (2006). Linearization of affine abstract categorial grammars. *In Proceedings of the 11th Conference on Formal Grammar*, pages 185–199, Malaga, Spain.