Génération de textes : G-TAG revisité avec les Grammaires Catégorielles Abstraites

Laurence Danlos, Aleksandre Maskharashvili, Sylvain Pogodalla

Génération de textes : G-TAG revisité avec les Grammaires Catégorielles Abstraites *

Laurence Danlos[1,2,3] Aleksandre Maskharashvili[4, 5, 6] Sylvain Pogodalla[4, 5, 6]
(1) Université Paris Diderot (Paris 7), Paris, F-75013, France
(2) ALPAGE, INRIA Paris–Rocquencourt, Paris, F-75013, France
(3) Institut Universitaire de France, Paris, F-75005, France
(4) INRIA, Villers-lès-Nancy, F-54600, France
(5) Université de Lorraine, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France
(6) CNRS, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France
{laurence.danlos}{aleksandre.maskharashvili}{sylvain.pogodalla}@inria.fr

Résumé. G-TAG est un formalisme dédié à la génération de textes. Il s'appuie sur les Grammaires d'Arbres Adjoints (TAG) qu'il étend avec des notions propres permettant de construire une forme de surface à partir d'une représentation conceptuelle. Cette représentation conceptuelle est indépendante de la langue, et le formalisme G-TAG a été conçu pour la mise en œuvre de la synthèse dans une langue cible à partir de cette représentation. L'objectif de cet article est d'étudier G-TAG et les notions propres que ce formalisme introduit par le biais des Grammaires Catégorielles Abstraites (ACG) en exploitant leurs propriétés de réversibilité intrinsèque et leur propriété d'encodage des TAG. Nous montrons que les notions clefs d'arbre de g-dérivation et de lexicalisation en G-TAG s'expriment naturellement en ACG. La construction des formes de surface peut alors utiliser les algorithmes généraux associés aux ACG et certaines constructions absentes de G-TAG peuvent être prises en compte sans modification supplémentaire.

Abstract. G-TAG is a formalism dedicated to text generation. It relies on the Tree Adjoining Grammar (TAG) formalism and extends it with several specific notions allowing for the construction of a surface form from a conceptual representation. This conceptual representation is independent from the target language. The goal of this paper is to study G-TAG and its specific notions from the perspective given by Abstract Categorial Grammars (ACG). We use the reversibility property of ACG and the encoding of TAG they offer. We show that the key G-TAG notions of g-derivation tree and lexicalization are naturally expressed in ACG. The construction of surface forms can then rely on the general ACG algorithms and some operations that G-TAG is lacking can be freely accounted for.

Mots-clés : TAG, G-TAG, génération, réalisation syntaxique, grammaires catégorielles abstraites.

Keywords: TAG, G-TAG, generation, syntactic realization, abstract categorial grammars.

1 Introduction

G-TAG (Danlos, 1998; Meunier, 1997; Danlos, 2000) est un formalisme dédié à la génération de textes. Il s'appuie sur les Grammaires d'Arbres Adjoints (TAG) (Joshi *et al.*, 1975; Joshi & Schabes, 1997) qu'il étend avec des notions propres, notamment les arbres de g-dérivation qui permettent de construire une forme de surface (arbre dérivé, ou chaîne de caractères) à partir d'une représentation conceptuelle. Cette représentation conceptuelle est indépendante de la langue, et le formalisme G-TAG a été conçu pour la mise en œuvre de la synthèse dans une langue cible de cette représentation. Ce formalisme a été implanté une première fois en ADA (Meunier, 1997) et plus récemment en .NET, et, dans cette dernière forme, utilisé et commercialisé auprès de Kantar Media, filiale de TNS-Sofres (Meunier *et al.*, 2011; Danlos *et al.*, 2011). Dans cette application, le but est d'accompagner les tableaux retraçant l'évolution des investissements publicitaires des clients par un texte de commentaires synthétisé automatiquement.

Une des motivations pour la définition du formalisme G-TAG était l'observation des différences entre les arbres de dérivation en TAG et les arbres de dépendances sémantiques (Schabes & Shieber, 1994). En analyse, cette observation a

∗. Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence (ANR-12-CORD-0004).

conduit à des propositions de modification de la notion de structure de dérivation (Kallmeyer, 2002; Joshi *et al.*, 2003; Rambow *et al.*, 2001; Chen-Main & Joshi, 2012). D'autres propositions ont néanmoins montré qu'il était possible de relier aux représentations sémantiques les arbres de dérivation de TAG, sans qu'il ne soit nécessaire de modifier ceux-ci. Soit en utilisant l'unification (Kallmeyer & Romero, 2004, 2007), soit en utilisant une représentation fonctionnelle des TAG (Pogodalla, 2004a, 2007, 2009) dans les Grammaires Catégorielles Abstraites (ACG) (de Groote, 2001). Ces dernières approches présentent l'intérêt d'être exprimées dans un cadre *intrinsèquement réversible*. C'est-à-dire que les grammaires utilisées et la nature des algorithmes mis en œuvre sont les mêmes, que l'on considère le problème de *l'analyse* pour passer de la forme de surface à une représentation sémantique, ou que l'on considère le problème de la *synthèse* pour passer de la représentation sémantique à la forme de surface.

L'objectif de cet article est donc d'étudier G-TAG, les notions propres que ce formalisme introduit, ainsi que ses limitations, par le biais des ACG en exploitant les propriétés de réversibilité de ces dernières et leur propriété d'encodage des TAG. Les ACG permettant d'exprimer également d'autres formalismes de la famille des TAG comme les *multiple component TAG* (MCTAG), certaines limitations de G-TAG peuvent être levées en utilisant ces formalismes sans qu'aucune autre adaptation ne soit nécessaire. Nous montrons que les notions clefs d'arbre de g-dérivation et de lexicalisation s'expriment naturellement en ACG. La construction des formes de surface peut alors utiliser les algorithmes généraux associés aux ACG, permettant d'analyser aussi bien des grammaires non contextuelles que des grammaires légèrement contextuelles (TAG, mais aussi les systèmes de réécriture linéaires non contextuels, LCFRS) (de Groote & Pogodalla, 2004) de manière optimisée (Kanazawa, 2007), et qui sont les mêmes pour l'analyse et la génération. Réciproquement, les principes de conception de G-TAG permettent d'éclairer les éléments opérationnels, notamment liés à la préférence de certaines réalisations, nécessaires à prendre en compte dans les algorithmes des ACG.

2 Génération de textes et G-TAG

L'architecture habituellement considérée dans les processus de génération de textes (Reiter & Dale, 1997) comporte trois sous-processus en cascade, chacun de ces sous-processus étant chargé de la réalisation de différentes tâches. À savoir : la planification du document (ou macro-planification : détermination du contenu, structuration du document), la micro-planification (agrégation, lexicalisation, génération des expressions référentielles), et la réalisation de surface (réalisation linguistique). La première tâche correspond à définir *Que dire ?* tandis que les deux autres réfèrent à *Comment le dire*. G-TAG est dédié à cette seconde tâche. L'entrée du processus de génération associé aux G-TAG, le *Que dire ?* est défini dans une représentation conceptuelle que décrit la section 2.1. Par la suite, le processus comporte trois étapes essentielles : la construction de l'arbre de g-dérivation, la construction de l'arbre g-dérivé, et les traitements ultérieurs pour la finalisation de la forme de surface.

2.1 Représentation conceptuelle

Le langage de représentation conceptuelle utilisé dans G-TAG est essentiellement un langage logique. Il est habituellement présenté sous une forme réifiée de premier ordre, que ce soit dans la logique typée du premier ordre *Login* (Aït-Kaci & Nasr, 1986) qui permet entre autre un contrôle de la bonne formation de la forme conceptuelle et d'abstraire l'ordre et le nombre des arguments par la présence des labels (nom des attributs), ou que ce soit sous forme de structures de représentation du discours segmenté (SDRS) (Danlos *et al.*, 2001).

Dans le présent article, nous adoptons la logique d'ordre supérieur (à la Montague) comme langage de représentation conceptuelle. Cela permet notamment d'éviter les problèmes de quantification implicite sur les labels des objets réifiés et leur traitement compositionnel. Ce faisant, nous considérons :

- que l'ordre et le rôle des arguments des prédicats est pris en compte dans le lien entre arbre de dérivation (ou plutôt le terme qui le représente) et sa représentation conceptuelle (ou sa réalisation sémantique) ;
- que les informations morpho-syntaxiques, habituelles en TAG sous forme de trait, sont pris en compte dans les arbres de dérivations, mais pas dans la représentation conceptuelle ;
- que la génération des expressions référentielles (pronoms, articles définis) est hors du champ de notre proposition. C'est pour l'instant une propriété de l'implantation G-TAG, mais pas du formalisme, qui s'appuie sur la réification. Nous souhaitons pouvoir envisager différentes théories pour la génération de ces expressions.

L'entrée correspondant à (2), qui pourrait engendrer les deux phrases de (3) par exemple, sera donc (1).

$$\text{RWDING}(\text{HUMAN}(\textit{Marie}), \text{HUMAN}(\textit{Jean})) \tag{1}$$

$$
\begin{aligned}
E_{12} &:= \text{RWDING}[\text{RWDER} \mapsto H_2, \text{RWDEE} \mapsto H_1] \\
H_1 &:= \text{HUMAN}[\text{NAME} \mapsto \textit{Jean}, \text{SEX} \mapsto \text{MASC}] \\
H_2 &:= \text{HUMAN}[\text{NAME} \mapsto \textit{Marie}, \text{SEX} \mapsto \text{FEM}]
\end{aligned} \tag{2}
$$

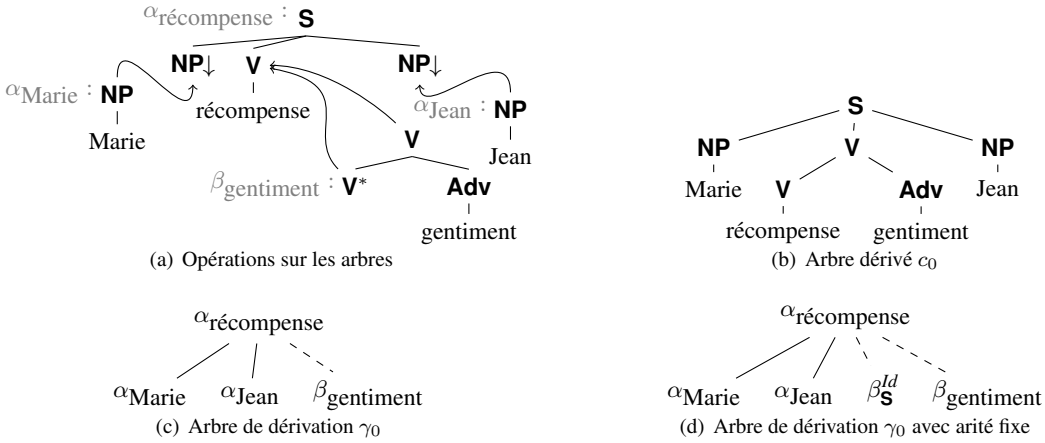
(3) a. Marie a récompensé Jean
b. Jean a été récompensé par Marie

2.2 Arbres de dérivation (TAG)

Les TAG sont des grammaires d'arbres qui mettent en œuvre deux opérations : la substitution et l'adjonction. La substitution permet d'étendre un arbre en remplaçant une feuille étiquetée d'un non terminal par un arbre dont la racine est étiquetée par ce même non terminal. L'adjonction permet *d'insérer* un arbre, appelé *arbre auxiliaire*, possédant un nœud racine et une feuille (nœud pied) étiquetés par un même non terminal. Cet arbre auxiliaire peut être inséré dans un autre arbre à un nœud de même étiquette que la racine de l'arbre auxiliaire. La figure 1(a) montre ces opérations : substitution des nœuds **NP** de l'arbre élémentaire de *récompense*, et adjonction à son nœud **V** de l'arbre auxiliaire de l'adverbe *gentiment*.

Le résultat obtenu est décrit à la figure 1(b), tandis que la figure 1(c) montre la *structure de dérivation*, ou *arbre de dérivation*, qui décrit les opérations effectuées sur les différents arbres :

— les constantes notées $\alpha_{\text{entrée lex}}$ sont les arbres initiaux associés à l'entrée lexicale *entrée lex* ;
— les constantes notées $\beta_{\text{entrée lex}}$ sont les arbres auxiliaires associés à l'entrée lexicale *entrée lex* ;
— un arc plein *parent-enfant* indique que l'arbre enfant a été substitué à une des feuilles de l'arbre parent ;
— un arc parent-enfant en pointillé indique que l'arbre enfant a été adjoint à l'arbre parent en l'un de ses nœuds.



(a) Opérations sur les arbres
(b) Arbre dérivé c_0
(c) Arbre de dérivation γ_0
(d) Arbre de dérivation γ_0 avec arité fixe

FIGURE 1: Analyse TAG de *Marie récompense gentiment Jean*

Remarque. *Dans la définition standard des arbres de dérivation (Vijay-Shanker, 1987), il n'y a pas d'adjonction multiple sur un même nœud, ceci ne changeant pas le pouvoir génératif de la grammaire. Cela signifie notamment qu'un arbre élémentaire peut faire l'objet d'un nombre maximum défini d'adjonctions et de substitutions. Autrement dit, comme étiquette d'un arbre de dérivation (par exemple $\alpha_{récompense}$ dans l'arbre de dérivation de la figure 1(c)), il a une arité* maximum. *Si l'on admet la présence d'éléments unités pour l'opération d'adjonction β^{Id}_X pour une adjonction à un nœud X, on peut même définir très précisément l'arité de ce symbole. L'arbre de la figure 1(c) devrait alors être représenté comme en 1(d).*

Plutôt qu'utiliser les adresses de Gorn pour repérer les nœuds où s'effectuent une opération, on peut fixer un ordre (arbitraire) des arguments et le faire correspondre systématiquement à l'ordre des arguments aussi bien dans la représentation conceptuelle que dans les arbres dérivés.

L'utilisation de la notion étendue de dérivation (Schabes & Shieber, 1994) est également possible dans ce cadre. Elle revient à dédoubler dans les arbres élémentaires chaque nœud pouvant recevoir une adjonction : en un nœud pour les arbres

auxiliaires modifieurs ; et en un autre nœud pour les arbres auxiliaires prédicatifs (selon la terminologie de (Schabes & Shieber, 1994)).

Compte tenu des remarques précédentes, nous insistons sur la convention que nous utiliserons : la notation $\alpha_{\text{entrée lex}}$ est utilisée aussi bien pour représenter l'arbre initial associé à l'entrée lexicale (ou ancré par) *entrée lex* que comme symbole d'arité fixe utilisé pour définir les termes que sont les arbres de dérivation. Si nécessaire, nous indiquerons l'arité du symbole en exposant : $\alpha^4_{\text{récompense}}$. Il en va de même pour la notation $\beta_{\text{entrée lex}}$ à propos des arbres auxiliaires. Alors l'arbre de dérivation de la figure 1(d) s'écrit comme le terme $\alpha^4_{\text{récompense}}(\alpha^0_{\text{Marie}}, \alpha^0_{\text{Jean}}, \beta^{Id}_{\mathbf{S}}, \beta^0_{\text{gentiment}})$ [1].

On constate alors que les arbres de dérivation, comme c'est explicité dans (Schabes & Shieber, 1994), sont des termes *clos*, c'est-à-dire dans lesquels n'apparaissent aucune variable. Or, le processus de génération de (Meunier, 1997) s'appuie sur la correspondance entre le concept à réaliser et les arbres exprimant ce concept pour synthétiser les textes. L'approche adoptée opère de haut en bas (*top-down*) : la relation conceptuelle la plus haute sélectionne un arbre dont les fils dans l'arbre de dérivation seront eux-mêmes générés récursivement par les concepts fils de la relation initiale. Le processus consiste donc à calculer un arbre de dérivation en commençant par la racine. Pour manipuler cet objet en cours de construction, une notion d'arbre de dérivation non complètement instancié est utile. L'arbre de g-dérivation est utilisé à cette fin.

2.3 Arbres de g-dérivation (G-TAG)

Pour exprimer cette notion d'arbre en cours de construction, (Meunier, 1997, Chap. 3, p71) définit les arbres de g-dérivation comme des arbres dont les nœuds sont soit des constantes qui sont des noms d'arbres élémentaires, soit des variables. Deux types de variables sont originellement considérés. D'une part celles qui correspondent au nom des attributs des concepts, utilisés pour associer le rôle sémantique d'un argument et sa position dans l'arbre de dérivation. Et d'autre part celles qui correspondent à l'étiquette qui est valeur de cet attribut (la variable de réification), cette étiquette pouvant être considérée comme à mi-chemin entre la variable de départ et l'arbre de g-dérivation qui la remplacera. Cela correspond à l'évolution de l'arbre de g-dérivation décrite par la figure 2.

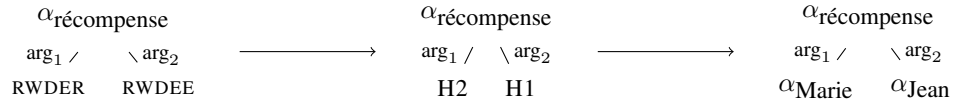
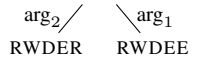


FIGURE 2: Évolution de l'arbre de g-dérivation avec variables vers un arbre instancié

Plus précisément, (Meunier, 1997) définit les constantes apparaissant dans les arbres de g-dérivation comme des noms *d'entrées lexicales* accompagnées de traits, appelés T-traits, plutôt que comme des noms d'arbres élémentaires. Cette distinction, importante pour la mise en œuvre et l'efficacité de G-TAG, permet par exemple de constituer une seule entrée lexicale qui ne distingue les arbres de *récompenser* à la forme active et à la forme passive que par le T-trait $[\texttt{T-trait} = \mp\texttt{passif}]$, réduisant ainsi le nombre de symboles à considérer sans rien changer pour les arguments. En effet, la correspondance entre l'argument d'un concept et l'argument d'une entrée qui le lexicalise est spécifiée par les étiquettes des arcs des arguments qui sont des rôles thématiques (arg_i) qui restent invariants quelle que soit la forme (par ex. active ou passive) de l'entrée lexicale. De plus, l'utilisation de rôles thématiques permet que l'argument conceptuel RWDEE corresponde à l'argument arg_2 des lexicalisations avec le verbe *récompenser*, mais à un argument arg_1 pour une lexicalisation avec *recevoir une récompense* : $\alpha_{\text{recevoir-récompense}}$. Dans l'approche que nous présentons section 3.2,



c'est la manière dont on interprète l'argument de l'arbre de dérivation soit syntaxiquement, soit sémantiquement, qui crée le lien entre le rôle thématique et le rôle conceptuel.

Les T-traits permettent donc de contrôler les constructions syntaxiques possibles. Par exemple, le T-trait $[\texttt{T-trait} = +\texttt{inf}]$ sur l'arbre de g-dérivation associé à la conjonction *pour (que)* spécifie que son deuxième argument doit être une infinitive, ce qui n'est possible que si le sujet vide de la subordonnée infinitive est interprété comme coréférent au sujet de la principale (*Jean passe l'aspirateur pour être récompensé par Marie*). Nous verrons à la section 4 comment modéliser en ACG cette contrainte sur la coréférence des sujets.

1. Pour simplifier les expressions, on fait ici l'hypothèse qu'il n'y a pas d'adjonction possible sur les nœuds **V** et **Adv** de l'adverbe. Sinon, il faudrait écrire le terme $\alpha^4_{\text{récompense}}(\alpha^0_{\text{Marie}}, \alpha^0_{\text{Jean}}, \beta^{Id}_{\mathbf{S}}, \beta^2_{\text{gentiment}}\ \beta^{Id}_{\mathbf{V}}\ \beta^{Id}_{\mathbf{Adv}})$ et l'arbre de dérivation correspondant.

Les termes ou arbres de g-dérivation correspondent donc à des *termes* ou *arbres avec variables*. Pour reprendre les notations habituelles (Comon *et al.*, 2007), si on appelle $\mathcal{F}$ l'ensemble des symboles avec arité, $\mathcal{F} = \bigcup_l \beta_l \cup \bigcup_l \alpha_l$, l'ensemble des arbres de g-dérivations est un sous-ensemble [2] de l'ensemble $T(\mathcal{F}, \mathcal{X})$ des termes construits sur $\mathcal{F}$ et $\mathcal{X}$ un ensemble de variables. Les arbres de dérivations sont les termes clos de cet ensemble. Cette représentation sous-forme d'arbres est isomorphe à celle que nous donnons à l'aide des ACG. Dans cette dernière représentation, les différentes contraintes sont exprimées à l'aides de types.

2.4 Autres notions de G-TAG

Base lexicale La base lexicale associée à un concept est essentiellement un ensemble d'arbres de g-dérivation capables d'exprimer ce concept, notamment parce que les fils de ces arbres expriment les rôles sémantiques associés aux arguments du concept. Cette base lexicale a un rôle opérationnel en permettant de réduire et de diriger la recherche des éléments autorisant la synthèse. Nous la mentionnons ici car elle joue un rôle analogue à la règle Scan dans les analyseurs à chartes. Nous ne donnons pas d'équivalent à cette notion de G-TAG dans la mesure où c'est l'implantation de l'algorithme d'inversion des lexiques dans les ACG, le même pour les tâches d'analyse et de génération, qui met éventuellement en œuvre cette notion si nécessaire [3]. Mais ce n'est pas décrit dans le formalisme lui-même.

Choix lexical Ce choix définit *la meilleure réalisation* (Meunier, 1997, Sect. 3.1.2, p70) d'un concept parmi les éléments de sa base lexicale. Un certain nombre de critères sémantiques, syntaxiques, mais aussi d'interaction entre les choix lexicaux, sont pris en compte à l'aide de différents tests. Quoique ce choix soit très important en pratique, nous considérons ici que toutes les solutions admissibles d'un point de vue syntaxique, c'est-à-dire que la grammaire TAG admettrait, sont possibles. Ce choix ne relève pas des propriétés combinatoires que décrivent la grammaire, et un traitement similaire à celui qui permet la désambiguïsation en syntaxe, avec d'autres critères bien sûr, est envisageable et pourrait bénéficier de toutes les avancées de ce domaine. Il ouvre cependant la question de l'intégration de ces critères aux algorithmes des ACGs, notamment si l'on souhaite qu'ils prennent en compte des informations linguistiques plutôt que statistiques.

Phrase et Discours Un point fort de G-TAG est d'autoriser la génération non de phrases isolées, mais de textes. La grammaire TAG associée contient donc des arbres élémentaires avec des signes de ponctuation, notamment le point. Ces arbres sont associés à des représentations conceptuelles exprimant la relation sémantique discursive entre deux phrases. Ainsi, les trois textes de (4) expriment la même représentation conceptuelle : SUCC(EAT(HUMAN JEAN), LEAVE(HUMAN MARIE)). Un certain nombre de contraintes sont exprimées par des traits, par exemple pour signifier que dans les constructions mentionnées, seul *ensuite* permet de coordonner deux phrases distinctes. Nous ne les utilisons pas ici pour simplifier les formules. Mais ils ne posent aucun problème théorique.

(4) a. Jean mange. Ensuite, Marie part.
b. Jean mange avant que Marie ne parte.
c. Jean mange avant le départ de Marie.

3 Génération et analyse dans les ACG

3.1 Généralités sur les ACG

Définitions et compositions Une ACG définit deux langages qu'elle met en relation : 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. Ici, le langage abstrait correspond à la structure grammaticale que l'on veut manipuler, c'est-à-dire l'arbre de dérivation. Il sera mis en relation à l'aide d'un premier *lexique* avec le langage objet des arbres dérivés (ou des chaînes de caractère), et à l'aide d'un deuxième lexique avec le langage objet des représentations conceptuelles.

Definition 1 (Signature d'ordre supérieur)**.** *Une* signature d'ordre supérieur *est un triplet* $\Sigma = \langle A, C, \tau \rangle$ *où :*
— A *est un ensemble de types atomiques ;*
— C *est un ensemble fini de constantes ;*

2. Certains termes n'étant pas possibles du fait des contraintes exprimées soit par les représentations conceptuelles, soit par les arbres dérivés.
3. Voir par exemple (Kanazawa, 2007) et son extension à une stratégie d'analyse particulière (Kanazawa, 2008).

— $\tau : C \to \mathcal{T}(A)$ *assigne à chaque constante de* C *un type de* $\mathcal{T}(A)$ *où* $\mathcal{T}(A) ::= A | \mathcal{T}(A) \multimap \mathcal{T}(A)$ [4].
On appelle $\Lambda(\Sigma)$ *l'ensemble des* λ*-termes que l'on peut construire avec la signature* Σ *et* $t : \alpha$ *signifie que le terme* t *a le type* α.

Ainsi, pour obtenir les arbres de la figure 1, nous pouvons définir une première signature :

$$\Sigma_{\text{dérivations}} = \begin{cases} A_{\Sigma_{\text{dérivations}}} = \{\mathbf{NP}, \mathbf{V}, \mathbf{S}, \mathbf{Adv}, \mathbf{V_A}, \mathbf{S_A}, \ldots\} \\ C_{\Sigma_{\text{dérivations}}} = \{C_{\textit{récompense}}, C_{\textit{Marie}}, C_{\textit{gentiment}}, C_{\textit{Jean}}, \beta^{Id}_{\mathbf{S}}, \beta^{Id}_{\mathbf{V}} \ldots\} \\ C_{\textit{Marie}} : \mathbf{NP} \qquad C_{\textit{récompense}} : \mathbf{S_A} \to \mathbf{V_A} \to \mathbf{NP} \to \mathbf{NP} \to \mathbf{S} \\ C_{\textit{Jean}} : \mathbf{NP} \qquad C_{\textit{gentiment}} : \mathbf{V_A} \\ \beta^{Id}_{\mathbf{V}} : \mathbf{V_A} \qquad \beta^{Id}_{\mathbf{S}} : \mathbf{S_A} \end{cases}$$

Cette signature permet de construire les structures de dérivation que sont les arbres de dérivation. Les types $X_{\mathbf{A}}$ correspondent aux types des arbres prêts à être adjoints. Pour le détail de l'encodage systématique d'une TAG dans une ACG, nous renvoyons le lecteur à (de Groote, 2002; Pogodalla, 2004a, 2007, 2009).

Une autre signature nous permet de construire les arbres dérivés. Cette signature ne comporte qu'un seul type, le type τ des arbres.

$$\Sigma_{\text{dérivés}} = \begin{cases} A_{\Sigma_{\text{dérivés}}} = \{\tau\} \\ C_{\Sigma_{\text{dérivés}}} = \{\mathbf{S}_2, \mathbf{S}_3, \mathbf{V}_1, \mathbf{V}_2, \mathbf{Adv}_1, \mathbf{NP}_1, \mathbf{NP}_2, \textit{Marie}, \textit{Jean}, \textit{gentiment}, \textit{récompense}, \ldots\} \\ \textit{Marie}, \textit{Jean}, \textit{récompense}, \textit{gentiment} : \tau \\ \mathbf{NP}_1, \mathbf{V}_1, \mathbf{Adv}_1 : \tau \to \tau \\ \mathbf{NP}_2, \mathbf{S}_2, \mathbf{V}_2 : \tau \to \tau \to \tau \\ \mathbf{S}_3 : \tau \to \tau \to \tau \to \tau \end{cases}$$

Chaque constante permet de construire un arbre, de type τ. Elles sont distinguées par leur arité ($\mathbf{NP}_1$, $\mathbf{NP}_2$, etc.). Cela correspond aux différentes arités effectivement exprimées dans les arbres élémentaires de la grammaire TAG mais qui sont généralement laissées implicites.

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 :*

— $:\overset{\tau}{=} \; : A_1 \to \mathcal{T}(A_2)$ *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* $\mathcal{T}(A_1)$ *;*
— $:\overset{c}{=} \; : C_1 \to \Lambda(\Sigma_2)$ *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* $\Lambda(\Sigma_1)$ *;*
— *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* $:\overset{c}{=}$*. On définit également la fonction d'interprétation* $[\![\cdot]\!]$ *telle que pour tout* $t \in \Lambda(\Sigma_1)$ *et* $t := u$*,* $[\![t]\!] = u$.

Nous pouvons maintenant définir un lexique qui relie les termes abstraits de $\Lambda(\Sigma_{\text{dérivations}})$ aux termes objets de $\Lambda(\Sigma_{\text{dérivés}})$. Les tables 1 et 2 définissent bien un lexique. Le premier donne l'interprétation des types atomiques. On remarquera avec l'interprétation des types $\mathbf{V_A}$ et $\mathbf{S_A}$ que l'interprétation d'un type atomique peut être un type non atomique. Cela explicite le fait que l'adjonction est une fonction qui prend un (sous-)arbre et retourne un arbre qui sera à son tour inséré. La deuxième table donne l'interprétation des constantes. Pour l'interprétation de $C_{\textit{récompense}}$ (et des verbes en général), S correspond à l'arbre auxiliaire qui peut éventuellement s'adjoindre au nœud $\mathbf{S}$ et a à l'arbre auxiliaire qui peut éventuellement s'adjoindre au nœud $\mathbf{V}$, tandis que s et o correspondent aux arbres $\mathbf{NP}$ qui seront respectivement sujet et objet.

$\mathbf{NP}$	$\mathbf{S}$	$\mathbf{V}$	$\mathbf{V_A}$	$\mathbf{S_A}$
$:=_{\text{syntaxe}} \tau$	$:=_{\text{syntaxe}} \tau$	$:=_{\text{syntaxe}} \tau$	$:=_{\text{syntaxe}} \tau \to \tau$	$:=_{\text{syntaxe}} \tau \to \tau$

TABLE 1: Interprétation des types de $\Sigma_{\text{dérivations}}$ vers $\Sigma_{\text{dérivés}}$

Definition 3 (Grammaire catégorielle abstraite)**.** *Une* grammaire catégorielle abstraite *est un quadruplet* $\mathcal{G} = \langle \Sigma_1, \Sigma_2, :=, s \rangle$ *où :*

4. $\multimap$ est l'implication linéaire. Les variables abstraites par les λ ne peuvent être utilisées qu'une et une seule fois.

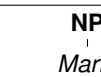
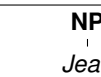
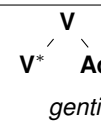
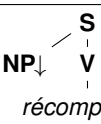
Interprétation des constantes		Arbre dérivé TAG correspondant
$C_{\textit{Marie}}$	$:=_{\text{syntaxe}} \mathbf{NP}_1\ \textit{Marie}$	
$C_{\textit{Jean}}$	$:=_{\text{syntaxe}} \mathbf{NP}_1\ \textit{Jean}$	
β^{Id}_X	$:= \lambda x.x \quad (X \in \{\mathbf{V_A}, \mathbf{S_A}, \ldots\})$	
$C_{\textit{gentiment}}$	$:=_{\text{syntaxe}} \lambda x.\mathbf{V}_2\ x\ (\mathbf{Adv}_1\ \textit{gentiment})$	
$C_{\textit{récompense}}$	$:=_{\text{syntaxe}} \lambda S\ a\ s\ o.S\ (\mathbf{S}_3\ s\ (a\ (\mathbf{V}_1\ \textit{récompense}))\ o)$	

TABLE 2: Lexique reliant les arbres de dérivation aux arbres dérivés

— Σ_1 *est une signature d'ordre supérieur, et* Σ_2 *une signature d'ordre supérieur. Elles sont appelés* vocabulaire abstrait *et* vocabulaire objet *;*
— $:= \quad : \Sigma_1 \to \Sigma_2$ *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* $\mathcal{O}(\mathcal{G}) = \{t \in \Lambda(\Sigma_2) | \exists u \in \mathcal{A}(\mathcal{G})\ \textit{avec}\ u := t\}$

Le terme correspondant à l'arbre de dérivation de la figure 1(c) est le terme : $t_0 = C_{\textit{récompense}}\ \beta^{Id}_{\mathbf{S}}\ C_{\textit{gentiment}}\ C_{\textit{Marie}} C_{\textit{Jean}}$. Il est bien typé, de type $\mathbf{S}$. Il appartient donc bien au langage abstrait de l'ACG $\mathcal{G}_{\text{syntaxe}}$ dont le lexique est donné dans les tables 1 et 2. On peut donc calculer son image par le lexique. Par définition, elle appartiendra au langage objet :

$$t_0 :=_{\text{syntaxe}} \mathbf{S}_3\ (\mathbf{NP}_1\ \textit{Marie})\ (\mathbf{V}_2\ (\mathbf{V}_1 \textit{récompense})\ (\mathbf{Adv}_1\ \textit{gentiment}))\ (\mathbf{NP}_1\ \textit{Jean})$$

La définition des ACG permet de considérer différents types d'architecture. On peut par exemple composer deux ACG de sorte que le vocabulaire objet de l'une soit également le vocabulaire abstrait de l'autre. C'est le cas si l'on veut considérer le lien entre les arbres dérivés, cette fois vus comme un langage abstrait, et leur production (*yield*) comme langage de chaînes. On utilisera une nouvelle ACG $\mathcal{G}_{\text{surface}}$ qui aura comme vocabulaire abstrait $\Sigma_{\text{dérivés}}$ le vocabulaire objet de $\mathcal{G}_{\text{syntaxe}}$ et comme vocabulaire objet la signature Σ_{string} dont le seul type est le type des chaînes de caractère σ, qui possède comme constantes l'opération de concaténation, l'élément vide, et les chaînes *Marie*, *Jean*, *récompense*, ... En utilisant le lexique défini dans la table 3 on obtient la chaîne de caractères associée à t_0 :

$$\begin{aligned} t_0 &:=_{\text{syntaxe}} \mathbf{S}_3\ (\mathbf{NP}_1\ \textit{Marie})\ (\mathbf{V}_2\ (\mathbf{V}_1 \textit{récompense})\ (\mathbf{Adv}_1\ \textit{gentiment}))\ (\mathbf{NP}_1\ \textit{Jean}) \\ &:=_{\text{surface} \circ \text{syntaxe}} \textit{Marie} + \textit{récompense} + \textit{gentiment} + \textit{Jean} \end{aligned}$$

Cette composition d'ACG est illustrée dans la partie gauche de la figure 3.

Analyse dans les ACG On peut maintenant préciser ce que l'on entend par problème d'analyse dans les ACG. Soit une ACG $\mathcal{G} = \langle \Sigma_1, \Sigma_2, :=, s \rangle$. Analyser un terme $u \in \Lambda(\Sigma_2)$, c'est-à-dire un terme construit sur le vocabulaire objet, revient à *trouver un terme* t *tel que* $t \in \mathcal{A}(\mathcal{G})$ *et* $t := u$. Il s'agit donc *d'inverser le lexique*.

Les propriétés des ACG dites d'ordre 2 [5] ont été particulièrement étudiées. Elles permettent d'encoder les formalismes faiblement contextuelles comme les TAG, les systèmes de réécriture linéaires non contextuels, les grammaires non contextuelles multiples (de Groote & Pogodalla, 2004; Salvati, 2006; Kanazawa, 2009) et la complexité de l'analyse est polynomiale (Kanazawa, 2008).

Or ces résultats ne dépendent que de l'ordre de la signature abstraite, mais pas du vocabulaire objet. L'inversion du morphisme reste possible dans les mêmes conditions y compris si le vocabulaire objet permet de construire des formules logiques pour la représentation conceptuelle. C'est ce qui permet de qualifier les ACG d'intrinsèquement réversibles.

5. Ce sont les ACG dont les types des constantes abstraites sont au plus d'ordre 2, avec $\text{ord}(a) = 1$ si a est un type atomique et $\text{ord}(\alpha \to \beta) = \max(\text{ord}(\beta), \text{ord}(\alpha) + 1)$. Autrement dit, ce sont les ACG dont les structures de dérivations, les termes abstraits, sont des arbres.

τ	$:=_{\text{surface}}\ \sigma$	*Marie*	$:=_{\text{surface}}\ \textit{Marie}$
Jean	$:=_{\text{surface}}\ \textit{Jean}$	*récompense*	$:=_{\text{surface}}\ \textit{récompense}$
gentiment	$:=_{\text{surface}}\ \textit{gentiment}$	$\mathbf{NP}_1, \mathbf{V}_1, \mathbf{Adv}_1$	$:=_{\text{surface}}\ \lambda x.x$
$\mathbf{NP}_2, \mathbf{S}_2, \mathbf{V}_2$	$:=_{\text{surface}}\ \lambda x\ y.x + y$	$\mathbf{S}_3$	$:=_{\text{surface}}\ \lambda x\ y\ z.x + y + z$

TABLE 3: Lexique pour les formes de surfaces à partir des formes dérivées

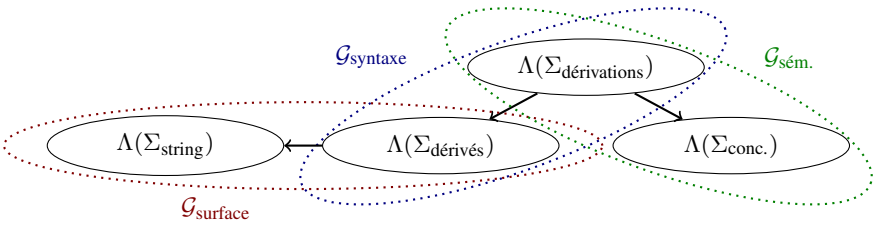


FIGURE 3: Architecture ACG utilisée pour les TAG et leur interface syntaxe-sémantique

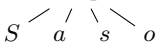
Interface syntaxe-sémantique La réalisation de l'interface syntaxe-sémantique avec les ACG s'obtient par un mode de composition des ACG différent du précédent. Il s'agit de partager la structure de dérivation, autrement dit de considérer deux ACG qui partagent un même vocabulaire abstrait. La partie droite de la figure 3 illustre cette composition. Le vocabulaire abstrait $\Sigma_{\text{dérivations}}$ étant déjà défini, il nous suffit de définir le vocabulaire objet $\Sigma_{\text{conc.}}$ et le lexique de $\mathcal{G}_{\text{sém.}}$. $\Sigma_{\text{conc.}}$ comprend les deux types atomiques THING (entités) et PROP (propositions), qui correspondent respectivement aux types e et t chez Montague, ainsi que les constantes typées de la table 4. Le lecteur pourra vérifier que :

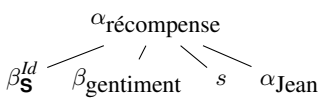
$$t_0 :=_{\text{sém.}} \text{KINDLY}(\text{RWDING}(\text{HUMAN MARIE}, \text{HUMAN JEAN}))$$

$\mathbf{S}$	$:=_{\text{sém.}}\ \text{PROP}$	$C_{\textit{Marie}}$	$:=_{\text{sém.}}\ \lambda P.P\ (\text{HUMAN MARIE})$
$\mathbf{N}$	$:=_{\text{sém.}}\ (\text{THING} \to \text{PROP}) \to \text{PROP}$	$C_{\textit{Jean}}$	$:=_{\text{sém.}}\ \lambda P.P\ (\text{HUMAN JEAN})$
$\mathbf{V}_{\mathsf{A}}$	$:=_{\text{sém.}}\ \text{PROP} \to \text{PROP}$	$\beta^{Id}_{\mathbf{V}}$	$:=_{\text{sém.}}\ \lambda x.x$
$\mathbf{S}_{\mathsf{A}}$	$:=_{\text{sém.}}\ \text{PROP} \to \text{PROP}$	$\beta^{Id}_{\mathbf{S}}$	$:=_{\text{sém.}}\ \lambda x.x$
$C_{\textit{gentiment}}$	$:=_{\text{sém.}}\ \lambda v\ s.\text{KINDLY}(v\ s)$	$C_{\textit{récompense}}$	$:=_{\text{sém.}}\ \lambda S\ a\ s\ o.S(s(\lambda x.o(\lambda y.a\ (\text{RWDING}(x, y)))))$

TABLE 4: Lexique de $\mathcal{G}_{\text{sém.}}$ pour la représentation conceptuelle

3.2 TAG, G-TAG, et ACG

Cette architecture permet donc d'utiliser un vocabulaire abstrait partagé entre deux ACG pour réaliser l'interface entre syntaxe et sémantique. Or, dans le cas de l'encodage des TAG avec les ACG, le langage abstrait des structures de dérivation, c'est-à-dire les termes de type $\mathbf{S}$, correspond aux arbres de dérivation des TAG. Par ailleurs, on a vu dans la section 2.3 que les arbres de g-derivation de G-TAG sont des arbres de dérivation TAG non complètement instanciés. Cela correspond en fait aux termes de $\Lambda(\Sigma_{\text{dérivations}})$ d'ordre 2. Ainsi, le terme $C_{\textit{récompense}} : \mathbf{S}_{\mathsf{A}} \to \mathbf{V}_{\mathsf{A}} \to \mathbf{NP} \to \mathbf{NP} \to \mathbf{S}$ peut être écrit sous-forme η-longue $\lambda S\ a\ s\ o.C_{\textit{récompense}}\ S\ a\ s\ o$. Il correspond à l'abre de g-dérivation (à l'ordre des arguments près : la modélisation TAG en ACG place habituellement les adjonctions en début, alors que les TAG les placent à la fin) :  . La même correspondance peut s'observer sur des termes et des arbres partiellement instanciés.

Par exemple entre le terme $\lambda s.C_{\textit{récompense}}\ \beta^{Id}_{\mathbf{S}}\ C_{\textit{gentiment}}\ s\ C_{\textit{Jean}}$ et l'arbre  où l'argument

correspondant au sujet n'est pas encore instancié.

$\mathcal{G}_{\text{sém.}}$ permet également d'exprimer la notion de base lexicale d'un concept C. L'ensemble des arbres de g-dérivation capables d'exprimer ce concept peut être défini comme l'ensemble des constantes du vocabulaire abstrait dont l'image par $:=_{\text{sém.}}$ admet le concept C comme sous-terme : $BL(\text{C}) = \{c \in C_{\Sigma_{\text{dérivations}}} \mid c :=_{\text{sém.}} u \text{ et C est un sous-terme de } u\}$.

Montrer l'équivalence formelle entre les arbres de g-dérivation et les termes de $\Lambda(\Sigma_{\text{dérivations}})$ d'ordre 2 irait au-delà de cet article. Mais l'approche que nous proposons consiste donc à *utiliser l'architecture des ACG pour construire les arbres de dérivations (ou arbres de g-dérivation complètement instanciés) à partir des termes représentant les formes conceptuelles*. Cela nous permet d'utiliser les propriétés de réversibilité des ACG de second ordre pour construire ces arbres de dérivations, ainsi que les techniques d'optimisation qui y sont liées.

Cette approche nous permet également de dépasser la limitation de G-TAG concernant les verbes ponts (verbes à complétive permettant l'extraction hors de celle-ci), généralement représentés par des arbres auxiliaires en TAG. Dans les dérivations, ils apparaissent donc comme dépendants de la tête de la complétive, alors que conceptuellement leur prédicat a portée sur le verbe de la complétive. En G-TAG, il a été choisi de ne pas modéliser ces verbes qui n'apparaissent pas dans les textes techniques (Danlos, 1998, 2000). On voit bien la difficulté technique liée à l'inversion de l'ordre des arguments entre l'arbre de g-dérivation et la représentation conceptuelle. Les études formelles sur les ACG ont montré que ce problème apparent n'en était pas vraiment un à l'ordre 2 [6] tant que les lexiques sont presque linéaires (*almost linear*) [7]. Comme il a été montré par ailleurs que l'arbre de dérivation permet d'exprimer de manière adéquate ces inversions de portée (Pogodalla, 2004b,a), il n'est plus nécessaire de faire des hypothèses sur la grammaire TAG utilisée.

4 Exemples

Les exemples de cette section exprimés au second ordre utilisent une ACG et l'algorithme d'analyse général du toolkit ACG [8]. Le premier exemple a pour objectif d'illustrer le lien qui doit être fait entre les arguments du terme abstrait et ses réalisations syntaxique et sémantique. Nous étendons donc $\mathcal{G}_{\text{syntaxe}}$ et $\mathcal{G}_{\text{sém.}}$ avec les interprétations des tables 5 et 6. Nous laissons le lecteur vérifier que :

$$C_{\textit{récompense}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}}\ C_{\textit{Marie}}\ C_{\textit{Jean}} :=_{\text{sém.}} \textsc{rwding}(\textsc{human Marie}, \textsc{human Jean})$$
$$C_{\textit{être récompensé}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}}\ C_{\textit{Jean}}\ C_{\textit{Marie}} :=_{\text{sém.}} \textsc{rwding}(\textsc{human Marie}, \textsc{human Jean})$$
$$C_{\textit{donne récompense}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}}\ C_{\textit{Marie}}\ C_{\textit{Jean}} :=_{\text{sém.}} \textsc{rwding}(\textsc{human Marie}, \textsc{human Jean})$$
$$C_{\textit{reçoit récompense}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}}\ C_{\textit{Jean}}\ C_{\textit{Marie}} :=_{\text{sém.}} \textsc{rwding}(\textsc{human Marie}, \textsc{human Jean})$$

Les constantes abstraites $C_{\textit{pour que}}$ et $C_{\textit{pour}}$ de la table 5 permettent d'engendrer les textes de (5) [9] à partir de la représentation conceptuelle : $\textsc{goal}(\textsc{vacc}(\textsc{human jean}), \textsc{rwding}(\textsc{human Marie}, \textsc{human Jean}))$. Il est à noter que $C_{\textit{pour}}$, à strictement parler, ne correspond pas à un arbre TAG mais permet la combinaison de plusieurs d'entre eux pour former l'expression p_1 *pour* p_2 où p_1 est le résultat de la substitution de **NP**$\downarrow$ dans s_1 par le **NP** donné en paramètre, tandis que p_2 est le résultat de la substitution du **NP**$\downarrow$ dans s_2 par l'arbre $\begin{array}{c}\mathbf{NP}\\ |\\ \mathbf{Pro}\end{array}$ [10] (syntaxiquement la chaîne vide alors que sémantiquement le sujet est le même que celui donné en argument à s_1 pour former p_1). Cela permet notamment d'assurer que le sujet est bien partagé sémantiquement. Cela requiert simplement un nouveau type **Sws** (et des règles lexicales) dont les interprétations sont telles que $\mathbf{Sws} :=_{\text{syntaxe}} \tau \to \tau$ et $\mathbf{Sws} :=_{\text{sém.}} [\![\mathbf{NP}]\!] \to \textsc{Prop}$ avec $\mathbf{NP} :=_{\text{sém.}} [\![\mathbf{NP}]\!] = (\textsc{Thing} \to \textsc{Prop}) \to \textsc{Prop}$. (5d) n'est pas engendré car il n'y a pas de constante abstraite pour le passif de *donner une récompense* qui n'est pas passivable en français. Or, comme l'interprétation sémantique de $C_{\textit{pour}}$ requiert que les deux propositions aient même sujet syntaxique, pour que le sujet de la deuxième soit le patient du prédicat RWDING, il faut un passif.

(5) a. Jean passe l'aspirateur pour que Marie récompense Jean
b. Jean passe l'aspirateur pour être récompensé par Marie
c. Jean passe l'aspirateur pour que Marie donne une récompense à Jean
d. *Jean passe l'aspirateur pour être donné une récompense par Marie

6. En fait, on peut lier cela à une complexité qui reste polynomiale mais dont le coefficient du polynôme croît avec la complexité du lexique exprimée par l'ordre maximum des termes réalisant les constantes abstraites. C'est ce qui explique la différence de complexité de l'analyse pour les grammaires non contextuelles et les grammaires TAG. Si des bornes maximales sont connues dans le cas des langages de chaînes et d'arbres, ce n'est pas le cas pour des langages objets en général.

7. C'est-à-dire qu'aucune variable n'est effacée (λ-abstraction vide) et que seules les variables de type atomique peuvent apparaître plusieurs fois. Dans le cas général, cela reste décidable mais extrêmement complexe (Salvati, 2010) et lève la restriction de monotonie sémantique (Shieber, 1988).

8. http://www.loria.fr/equipes/calligramme/acg/#Software

9. Nous donnons les exemples sans pronom puisque nous ne les traitons pas pour l'instant, mais ils seraient nécessaires.

10. Pour des raisons de simplicité, nous omettons les traits morpho-syntaxiques tels que subjonctif ou infinitif.

Le dernier exemple que nous souhaitons évoquer concerne *ensuite*. La constante abstraite que nous avons définie permet d'attacher l'adverbe à la proposition. (Danlos, 2000) indique que l'obtention de l'attachement au **V** passe par un post-traitement, ou alors requiert un formalisme plus expressif que TAG. Nous pouvons exprimer ceci grâce à l'ordre supérieur avec une constante $C'_{\textit{ensuite}} : \mathbf{S} \to (\mathbf{V_A} \to \mathbf{S}) \to \mathbf{S}$ dont les interprétations sont : $C'_{\textit{ensuite}} :=_{\text{sém.}} \lambda s_1\ s_2.\text{SUCC}(s_1, s_2\ (\lambda x.x))$ et $C'_{\textit{ensuite}} :=_{\text{syntaxe}} \lambda s_1\ s_2.\mathbf{S}_2\ s_1\ (s_2(\lambda x.\mathbf{V}_2\ x\ \textit{ensuite}))$. Nous laissons le lecteur vérifier que les termes t_1 et t_2 donnés en (6) dérivent bien la même représentation sémantique $\text{SUCC}(\text{VACC}(\text{HUMAN JEAN}), \text{RWDING}(\text{HUMAN MARIE}, \text{HUMAN JEAN}))$ et engendrent respectivement les deux phrases de (7).

$$
\begin{aligned}
t_1 &= C_{\textit{ensuite}}(C_{\textit{passe l'aspirateur}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}} C_{\textit{Jean}})\ (C_{\textit{donne récompense}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}}\ C_{\textit{Marie}}\ C_{\textit{Jean}}) \\
t_2 &= C'_{\textit{ensuite}}(C_{\textit{passe l'aspirateur}}\ \beta^{Id}_{\mathbf{S}}\ \beta^{Id}_{\mathbf{V}} C_{\textit{Jean}})\ (\lambda v.C_{\textit{donne récompense}}\ \beta^{Id}_{\mathbf{S}}\ v\ C_{\textit{Marie}}\ C_{\textit{Jean}})
\end{aligned}
\tag{6}
$$

(7) a. Jean passe l'aspirateur. Ensuite Marie donne une récompense à Jean.
b. Jean passe l'aspirateur. Marie donne ensuite une récompense à Jean.

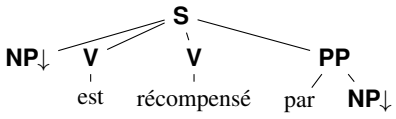
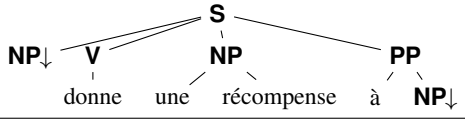
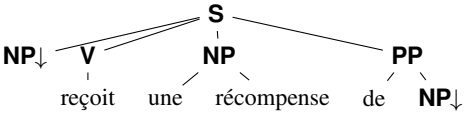
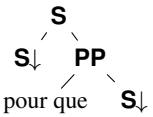
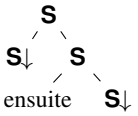
$C_{\textit{être récompensé}}$	$: \mathbf{S_A} \to \mathbf{V_A} \to \mathbf{NP} \to \mathbf{NP} \to \mathbf{S}$	
$C_{\textit{donne récompense}}$	$: \mathbf{S_A} \to \mathbf{V_A} \to \mathbf{NP} \to \mathbf{NP} \to \mathbf{S}$	
$C_{\textit{reçoit récompense}}$	$: \mathbf{S_A} \to \mathbf{V_A} \to \mathbf{NP} \to \mathbf{NP} \to \mathbf{S}$	
$C_{\textit{pour que}}$	$: \mathbf{S} \to \mathbf{S} \to \mathbf{S}$	
$C_{\textit{pour}}$	$: \mathbf{Sws} \to \mathbf{Sws} \to \mathbf{NP} \to \mathbf{S}$	
$C_{\textit{ensuite}}$	$: \mathbf{S} \to \mathbf{S} \to \mathbf{S}$	

TABLE 5: Arbres élémentaires et constantes abstraites pour les différentes expressions du concept RWDING

$C_{\textit{être récompensé}}$	$:=_{\text{syntaxe}} \lambda S\ a\ s\ o.S\ (\mathbf{S}_4\ (\mathbf{NP}_1\ s)\ (a\ (\mathbf{V}_2\ \textit{est}))\ (\mathbf{V}_1\ \textit{récompensé})\ (\mathbf{PP}_2\ \textit{par}\ o))$ $:=_{\text{sém.}} \lambda S\ a\ s\ o.S(a(\text{RWDING}(o, s)))$
$C_{\textit{donne récompense}}$	$:=_{\text{syntaxe}} \lambda S\ a\ s\ o.S\ (\mathbf{S}_4\ (\mathbf{NP}_1\ s)\ (a\ (\mathbf{V}_1\ \textit{donne}))\ (\mathbf{NP}_2\ \textit{une récompense})\ (\mathbf{PP}_2\ \textit{à}\ o))$ $:=_{\text{sém.}} \lambda S\ a\ s\ o.S(a(\text{RWDING}(s, o)))$
$C_{\textit{reçoit récompense}}$	$:=_{\text{syntaxe}} \lambda S\ a\ s\ o.S\ (\mathbf{S}_4\ (\mathbf{NP}_1\ s)\ (a\ (\mathbf{V}_1\ \textit{reçoit}))\ (\mathbf{NP}_2\ \textit{une récompense})\ (\mathbf{PP}_2\ \textit{de}\ o))$ $:=_{\text{sém.}} \lambda S\ a\ s\ o.S(a(\text{RWDING}(o, s)))$
$C_{\textit{pour que}}$	$:=_{\text{syntaxe}} \lambda s_1\ s_2.\mathbf{S}_2\ s_1\ (\mathbf{PP}_2\ (\textit{pour que})\ s_2)$ $:=_{\text{sém.}} \lambda s_1\ s_2.\text{GOAL}(s_1, s_2)$
$C_{\textit{pour}}$	$:=_{\text{syntaxe}} \lambda s_1\ s_2\ s.\mathbf{S}_2\ (s_1\ s)\ (\mathbf{PP}_2\ \textit{pour}\ (s_2\ (\mathbf{NP}_1\ \mathbf{Pro})))$ $:=_{\text{sém.}} \lambda s_1\ s_2\ n.\text{GOAL}(s_1\ n, s_2\ n)$
$C_{\textit{ensuite}}$	$:=_{\text{syntaxe}} \lambda s_1\ s_2.\mathbf{S}_2\ s_1\ (\mathbf{PP}_2\ (\textit{ensuite})\ s_2)$ $:=_{\text{sém.}} \lambda s_1\ s_2.\text{SUCC}(s_1, s_2)$

TABLE 6: Différentes expressions du concept RWDING

5 Conclusion et perspectives

Nous avons étudié G-TAG et ses notions propres, notamment l'arbre de g-dérivation, et montré que ce dernier correspondait aux termes abstraits d'une ACG de second ordre. Cela nous permet : d'une part d'utiliser les propriétés de réversibilité intrinsèque de ce formalisme ; d'autre part de généraliser l'approche à des phénomènes non traités (verbes ponts) tout en restant dans les TAG ; enfin de créer facilement des liens vers des formalismes plus expressifs (cf l'attachement de *ensuite* au nœud **V**). Cependant, un certain nombre de traitements propres aux implantations G-TAG ne sont pas encore traduits ici. La génération des expressions référentielles en est un. Une autre limitation concerne la parallélisation, ou la factorisation de certains évènements dans certaines constructions. En effet, la représentation conceptuelle que nous avons adoptée associe la représentation SUCC(GOAL(VACC(HUMAN JEAN), RWDING(HUMAN MARIE, HUMAN JEAN), NAP(HUMAN JEAN)) au texte *Jean a passé l'aspirateur pour être récompensé par Marie. Ensuite il a fait la sieste*. Or le prédicat GOAL ne devrait pas être dans la portée de SUCC. Ceci est traité à l'aide d'opérations particulières en G-TAG (Meunier, 1997, Section 6.1.2). Nous souhaitons éviter les opérations spécifiques et pour cela utiliser dans des travaux ultérieurs la représentation sémantique d'ordre supérieur décrite par (Danlos, 2009). Par ailleurs, le problème de guider le choix des réalisations lexicales, notamment par des règles et des connaissances linguistiques comme en G-TAG, et d'indiquer des préférences aux algorithmes utilisés dans les ACG est ouvert. Enfin, la clarification du statut de l'arbre de g-dérivation du point de vue des propriétés formelles des langages engendrés nous permet d'envisager de comparer plus précisément cette approche à celles qui considèrent les arbres de dérivation comme les arbres engendrés par une grammaire régulière d'arbres (Schmitz & Le Roux, 2008) dont les dérivations sont utilisées comme pivot pour la génération (Gardent & Perez-Beltrachini, 2010).

Références

AÏT-KACI H. & NASR R. (1986). LOGIN : A logic programming language with built-in inheritance. *The Journal of logic programming*, **3**(3), 185–215. doi :10.1016/0743-1066(86)90013-0.

CHEN-MAIN J. & JOSHI A. K. (2012). A dependency perspective on the adequacy of tree local multi-component tree adjoining grammar. *Journal of Logic and Computation*. doi :10.1093/logcom/exs012.

COMON H., DAUCHET M., GILLERON R., LÖDING C., JACQUEMARD F., LUGIEZ D., TISON S. & TOMMASI M. (2007). Tree Automata Techniques and Applications. Available on : http://www.grappa.univ-lille3.fr/tata. Release October, 12th 2007.

DANLOS L. (1998). G-TAG : Un formalisme lexicalisé pour la génération de textes inspiré de TAG. *Traitement Automatique des Langues*, **39**(2). http://hal.inria.fr/inria-00098489.

DANLOS L. (2000). G-TAG : A lexicalized formalism for text generation inspired by Tree Adjoining Grammar. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree Adjoining Grammars : Formalisms, Linguistic Analysis, and Processing*, volume 107 of *CSLI Lecture Notes*, p. 343–370 : CSLI Publications.

DANLOS L. (2009). D-STAG : un formalisme d'analyse automatique de discours basé sur les TAG synchrones. *T.A.L.*, **50**(1), 111–143. http://hal.inria.fr/inria-00524743/en/.

DANLOS L., GAIFFE B. & ROUSSARIE L. (2001). Document sructuring à la SDRT. In H. HORACEK, N. NICOLOV & L. WANNER, Eds., *Proceedings of the ACL 2001 Eighth European Workshop on Natural Language Generation (EWNLG)*. http://aclweb.org/anthology/W/W01/W01-0803.pdf.

DANLOS L., MEUNIER F. & COMBET V. (2011). EasyText : an operational NLG system. In *ENLG 2011, 13th European Workshop on Natural Language Generation*. http://hal.inria.fr/inria-00614760/en/.

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. http://aclweb.org/anthology/P/P01/P01-1033.pdf.

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. http://www.loria.fr/equipes/calligramme/acg/publications/2002-tag+6.pdf.

DE GROOTE P. & 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. doi :10.1007/s10849-004-2114-x.

GARDENT C. & PEREZ-BELTRACHINI L. (2010). RTG based surface realisation for TAG. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, p. 367–375, Beijing, China : Coling 2010 Organizing Committee. http://www.aclweb.org/anthology/C10-1042.

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 IWCS-5*.

JOSHI A. K., LEVY L. S. & TAKAHASHI M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Sciences*, **10**(1), 136–163. doi :10.1016/S0022-0000(75)80019-5.

JOSHI A. K. & SCHABES Y. (1997). Tree-adjoining grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of formal languages*, volume 3, chapter 2. Springer.

KALLMEYER L. (2002). Using an Enriched TAG Derivation Structure as Basis for Semantics. In *Proceedings of TAG+6*.

KALLMEYER L. & ROMERO M. (2004). LTAG Semantics with Semantic Unification. In *Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms - TAG+7*, p. 155–162. http://www.sfs.uni-tuebingen.de/~lk/papers/kallmrom-tag+7.pdf.

KALLMEYER L. & ROMERO M. (2007). Scope and Situation Binding for LTAG. *Research on Language and Computation*, **6**(1), 3–52. doi :10.1007/s11168-008-9046-6.

KANAZAWA M. (2007). Parsing and Generation as Datalog Queries. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, p. 176–183, Prague, Czech Republic : Association for Computational Linguistics. http://www.aclweb.org/anthology/P/P07/P07-1023.

KANAZAWA M. (2008). A prefix-correct Earley recognizer for multiple context-free grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, p. 49–56, Tuebingen, Germany.

KANAZAWA M. (2009). Second-Order Abstract Categorial Grammars as Hyperedge Replacement Grammars. *Journal of Logic, Language, and Information*, **19**(2), 137–161. doi :10.1007/s10849-009-9109-6.

MEUNIER F. (1997). *Implantation du formalisme de génération G-TAG*. PhD thesis, Université Paris 7 — Denis Diderot.

MEUNIER F., DANLOS L. & COMBET V. (2011). EasyText : un système opérationnel de génération de textes. In *Actes de la 18e conférence sur le Traitement Automatique des Langues Naturelles*, Montpellier, France. http://hal.inria.fr/inria-00607708.

POGODALLA S. (2004a). Computing Semantic Representation : Towards ACG Abstract Terms as Derivation Trees. In *Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms - TAG+7*, p. 64–71, Vancouver, BC, Canada. http://hal.inria.fr/inria-00107768.

POGODALLA S. (2004b). Vers un statut de l'arbre de dérivation : exemples de construction de représentations sémantiques pour les Grammaires d'Arbres Adjoints. In *Traitement Automatique des Langues Naturelles - TALN'04*, p. 10 p, Fès, Maroc : none. http://hal.inria.fr/inria-00107767.

POGODALLA S. (2007). Ambiguïté de portée et approche fonctionnelle des TAG. In *Traitement Automatique des Langues Naturelles - TALN 2007*, p. 10 p., Toulouse, France. http://hal.inria.fr/inria-00141913.

POGODALLA S. (2009). Advances in Abstract Categorial Grammars : Language Theory and Linguistic Modeling. ESSLLI 2009 Lecture Notes, Part II. http://hal.inria.fr/hal-00749297.

RAMBOW O., VIJAY-SHANKER K. & WEIR D. (2001). D-Substitution Grammars. *Computational Linguistics*, **27**(1), 87–121. http://aclweb.org/anthology/J/J01/J01-1004.pdf.

REITER E. & DALE R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, **3**(1), 57–87. doi :10.1017/S1351324997001502.

SALVATI S. (2006). Encoding second order string ACG with Deterministic Tree Walking Transducers. In SHULY WINTNER, Ed., *Proceedings of The 11th conference on Formal Grammar FG 2006*, FG Online Proceedings, p. 143–156, Malaga Espagne : CSLI Publications. http://cslipublications.stanford.edu/FG/2006/salvati.pdf.

SALVATI S. (2010). On the membership problem for non-linear Abstract Categorial Grammars. *Journal of Logic, Language and Information*, **19**(2), 163–183. doi :10.1007/s10849-009-9110-0.

SCHABES Y. & SHIEBER S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, **20**(1), 91–124. http://aclweb.org/anthology/J/J94/J94-1004.pdf.

SCHMITZ S. & LE ROUX J. (2008). Feature Unification in TAG Derivation Trees. In C. GARDENT & A. SARKAR, Eds., *Proceedings of the 9th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+'08)*, p. 141–148, Tübingen, Germany. http://arxiv.org/abs/0804.4584.

SHIEBER S. M. (1988). A Uniform Architecture for Parsing and Generation. In *Proceedings of the 12th International Conference on Computational Linguistics*, volume 2, p. 614–619, Budapest. http://dash.harvard.edu/bitstream/handle/1/2265286/Shieber_UniformArchitecture.pdf.

VIJAY-SHANKER K. (1987). *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania.