TALN 2008, Avignon, 9–13 juin 2008

Calculs d'unification sur les arbres de dérivation TAG

Sylvain Schmitz[1] Joseph Le Roux[2]
(1) LORIA, INRIA Nancy Grand Est, Nancy
(2) LORIA, Université Nancy 2, Nancy
Sylvain.Schmitz@loria.fr, Joseph.LeRoux@loria.fr

Résumé. Nous définissons un formalisme, les grammaires rationnelles d'arbres avec traits, et une traduction des grammaires d'arbres adjoints avec traits vers ce nouveau formalisme. Cette traduction préserve les structures de dérivation de la grammaire d'origine en tenant compte de l'unification de traits. La construction peut être appliquée aux réalisateurs de surface qui se fondent sur les arbres de dérivation.

Abstract. The derivation trees of a tree adjoining grammar provide a first insight into the sentence semantics, and are thus prime targets for generation systems. We define a formalism, feature based regular tree grammars, and a translation from feature based tree adjoining grammars into this new formalism. The translation preserves the derivation structures of the original grammar, and accounts for feature unification.

Mots-clés : Unification, grammaire d'arbres adjoints, arbre de dérivation, grammaire rationnelle d'arbres.

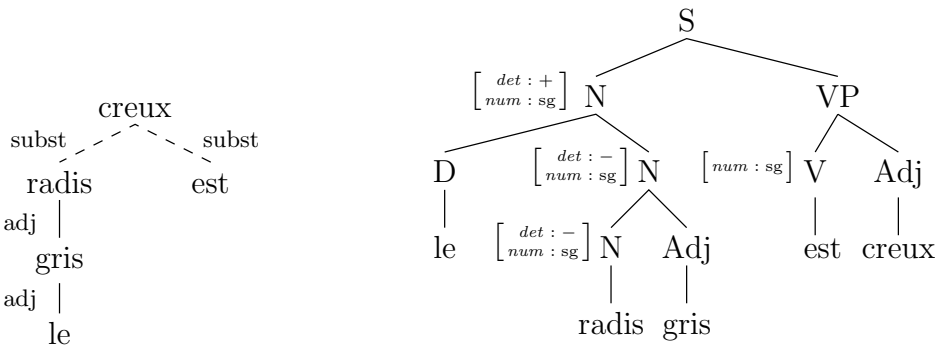
Keywords: Unification, tree adjoining grammar, derivation tree, regular tree grammar.

1 Introduction

Le processus de dérivation dans les grammaires d'arbres adjoints (Joshi & Schabes, 1997, TAG) produit deux arbres : l'*arbre dérivé* qui correspond à un arbre syntagmatique classique (voir figure 1b), et l'*arbre de dérivation*, qui présente par quelles opérations les arbres élémentaires de la grammaire ont été combinés pour obtenir l'arbre dérivé (voir figure 1a). Selon la tâche de traitement de la langue, il sera plus adéquat de considérer l'un ou l'autre, l'arbre dérivé étant en correspondance avec les lexèmes d'une phrase, tandis que l'arbre de dérivation donne une vue sémantique primitive de la phrase, comme le montrent par exemple Candito & Kahane (1998).

De fait, l'arbre de dérivation est privilégié dans plusieurs approches pour la réalisation de surface (Koller & Striegnitz, 2002; Koller & Stone, 2007). Il sert aussi de pivot à partir duquel représentation sémantique et arbre dérivé peuvent être générés dans les approches de de Groote (2002), Pogodalla (2004) et Kanazawa (2007) à base de grammaires catégorielles abstraites.

Ces travaux ne sont cependant pas immédiatement applicables à des grammaires réalistes qui emploient une variante des TAG à base de structures de traits (Vijay-Shanker, 1992, voir par exemple la figure 2). Cette variante munit les nœuds des arbres élémentaires de structures de traits, dont les unifications contraignent les opérations de substitution ou d'adjonction du nœud.



(a) Arbre de dérivation. (b) Arbre dérivé.

FIG. 1: Dérivation de la phrase « Le radis gris est creux. » avec la grammaire de la figure 2.

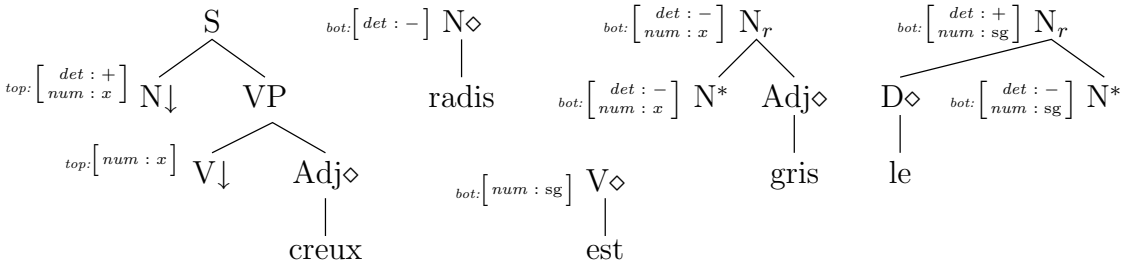


FIG. 2: Exemple de grammaire d'arbres adjoints avec structures de traits.

Ces structures ne posent en théorie aucun problème, car les domaines de valeur des différents traits sont finis et il suffit de démultiplier le nombre de symboles non-terminaux pour émuler les différentes structures possibles. Mais le nombre de ces structures s'accommode mal de cette vision naïve : par exemple, les vingt-huit traits syntaxiques utilisés dans la grammaire SEMFRAG du français (Gardent, 2006) décrivent un domaine, certes fini, mais comprenant plus de 214 milliards d'éléments. Enfin, l'argument du domaine fini ne tient tout simplement pas pour certains mécanismes de construction sémantique fondés sur l'unification de traits d'index sémantiques qui ont des domaines de valeur non finis (Gardent & Kallmeyer, 2003; Gardent, 2006).

Nous étudions dans cet article la traduction d'une grammaire d'arbres adjoints avec structures de traits en une grammaire rationnelle d'arbres de dérivation qui en préserve les mécanismes d'unification de traits. Plus en détail,

– nous rappelons comment traduire une grammaire TAG en une grammaire rationnelle d'arbres (RTG) qui en génère les arbres de dérivation (section 2.1),
– puis nous définissons un formalisme de grammaires rationnelles d'arbres enrichies par des structures de traits et montrons comment traduire une grammaire TAG dans ce nouveau formalisme (section 2.2) ;
– enfin, nous proposons une seconde traduction qui améliore l'efficacité de la génération des arbres de dérivation TAG (section 3).

Nous supposons que le lecteur est familier avec les aspects théoriques des grammaires d'arbres adjoints (Joshi & Schabes, 1997), des grammaires rationnelles d'arbres (Comon *et al.*, 2007) et de l'unification (Robinson, 1965)[1].

[1]Pour éviter toute confusion avec l'opération de substitution dans les TAG, la notion de substitution que l'on trouve associée à l'unification sera appelée u-substitution dans la suite.

2 Arbres de dérivation et unification

Un arbre de dérivation d'une grammaire d'arbres adjoints a des nœuds étiquetés par des arbres élémentaires de la grammaire et en guise d'arêtes les relations d'adjonctions et substitutions permises par la grammaire entre arbres élémentaires. Dans un premier temps, nous reformulons la description donnée par de Groote (2002) des arbres de dérivation qu'une grammaire d'arbres adjoints peut engendrer, en utilisant explicitement une grammaire rationnelle d'arbres. Dans un second temps, nous montrons comment les calculs d'unification de l'arbre dérivé peuvent s'intégrer simplement dans cette grammaire rationnelle.

2.1 Grammaire rationnelle des arbres de dérivation

Formellement, une grammaire d'arbres adjoints $\langle \Sigma, N, I, A, S \rangle$ est constituée d'un alphabet terminal Σ, d'un alphabet non-terminal N, d'un ensemble I d'arbres initiaux α, d'un ensemble A d'arbres auxiliaires β, et d'un non-terminal distingué S de N. Nous désignons par γ_r le nœud racine de l'arbre élémentaire γ et par β_f le nœud pied de l'arbre auxiliaire β.

Les nœuds d'un arbre élémentaire γ de $I \cup A$ qui nous intéressent sont étiquetés par des non-terminaux, et permettent une opération de substitution ou d'adjonction ; nous considérons en particulier que le pied d'un arbre auxiliaire ne permet pas d'adjonction[2]. Nous numérotons ces nœuds par un parcours arbitraire depuis la racine, de sorte que $\gamma_1 = \gamma_r$. Nous notons $\mathsf{lab}(\gamma_i)$ l'étiquette dans N du nœud γ_i.

Pour construire la grammaire rationnelle $\langle S, N \cup N_A, \mathcal{F}, R \rangle$ des arbres de dérivation, nous définissons :

- l'ensemble des arbres élémentaires comme notre alphabet ordonné $\mathcal{F} = I \cup A \cup \{\varepsilon\}$, où le rang $n = \mathsf{rg}(\gamma)$ d'un arbre élémentaire γ est le nombre de ses nœuds où une substitution ou une adjonction est possible, et où ε, de rang 0, représente une feuille vide ;
- l'alphabet non-terminal N et un duplicata $N_A = \{X_A \mid X \in N\}$ comme alphabet de la grammaire rationnelle ; à chaque nœud non terminal γ_i d'un arbre étiqueté par $X = \mathsf{lab}(\gamma_i)$, on associe un non terminal $\mathsf{nt}(\gamma_i)$ de forme X dans N s'il permet une substitution ou X_A dans N_A s'il permet une adjonction ;
- l'ensemble de règles R défini comme l'union

$$\begin{array}{ll} & \{X \to \alpha(\mathsf{nt}(\alpha_1), \ldots, \mathsf{nt}(\alpha_n)) \mid \alpha \in I, n = \mathsf{rg}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\ \cup & \{X_A \to \beta(\mathsf{nt}(\beta_1), \ldots, \mathsf{nt}(\beta_n)) \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \\ \cup & \{X_A \to \varepsilon \mid X_A \in N_A\} \end{array} \quad (1)$$

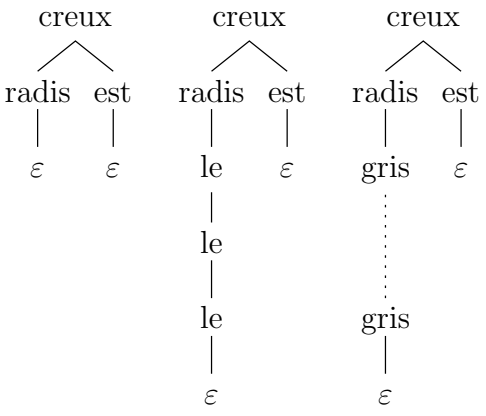
Les arbres initiaux de la grammaire TAG sont ainsi associés à des règles de la forme $X \to \alpha(Y_1, \ldots, Y_n)$ et les arbres auxiliaires à des règles $X_A \to \beta(Y_1, \ldots, Y_n)$, où X est le non terminal qui étiquette la racine de l'arbre élémentaire TAG. Enfin, la possibilité d'une adjonction non réalisée est simulée par les règles $X_A \to \varepsilon$. On peut observer que la taille de la grammaire RTG obtenue est équivalente à la taille de la grammaire TAG d'origine. La traduction elle-même peut être calculée en temps linéaire.

Puisque la grammaire TAG de la figure 2 ne propose pas d'arbre auxiliaire enraciné par S, VP, Adj ou D, on peut simplifier les règles en ignorant ces nœuds d'adjonction. La figure 3a montre

[2]Dans un souci de concision, nous ne traitons pas les contraintes d'adjonction sélective, qui ne posent aucune difficulté conceptuelle.

$\langle S, \{S, V, V_A, N, N_A\},$
$\{\text{creux}, \text{est}, \text{radis}, \text{gris}, \text{le}, \varepsilon\},$
$\{S \to \text{creux}(N, V),$
$V \to \text{est}(V_A),$
$V_A \to \varepsilon,$
$N \to \text{radis}(N_A),$
$N_A \to \text{gris}(N_A),$
$N_A \to \text{le}(N_A),$
$N_A \to \varepsilon\}\rangle$

(a) Grammaire rationnelle d'arbres.



(b) Exemples d'arbres générés.

FIG. 3: Grammaire rationnelle correspondant à la grammaire TAG de la figure 2.

la grammaire simplifiée pour les arbres de la figure 2. Il est aisé de vérifier que cette grammaire rationnelle génère les arbres enracinés par « creux », avec « radis » et « est » pour deux fils, et une combinaison arbitraire de nœuds « le » et « gris » comme descendance de « radis » (voir figure 3b) : la grammaire rationnelle génère les arbres de dérivation d'une version sans structures de traits de la grammaire TAG d'origine.

2.2 Calculs d'unification

Grammaire rationnelle d'arbres avec traits Afin de traduire les restrictions imposées par les structures de traits de la grammaire TAG, nous considérons dans notre RTG non plus de simples réécritures entre termes, mais des *surréductions* (Hanus, 1994), c'est-à-dire des réécritures assorties d'unifications, avec des variables dans $(N \cup N_A) \times \mathcal{D}$ où $\mathcal{D}$ désigne l'ensemble des structures de traits possibles[3].

Definition 1. Une *grammaire rationnelle d'arbres avec traits* $\langle S, N, \mathcal{F}, \mathcal{D}, R\rangle$ est composée d'un axiome S, d'un ensemble de symboles non-terminaux N contenant S, d'un alphabet ordonné de terminaux $\mathcal{F}$, d'un ensemble de structures de traits $\mathcal{D}$, et d'un ensemble de règles de forme $(A, d) \to a((B_1, d'_1), \ldots, (B_n, d'_n))$ avec $A, B_1, \ldots, B_n$ des non-terminaux de N, $d, d'_1, \ldots, d'_n$ des structures de traits de $\mathcal{D}$, et a un terminal d'arité n de $\mathcal{F}$.

La relation de dérivation $\Rightarrow$ associée à $G = \langle S, N, \mathcal{F}, \mathcal{D}, R\rangle$ met en relation des paires associant un terme[4] de $T(\mathcal{F}, N \times \mathcal{D})$ et une u-substitution, de telle sorte que $(s, e) \Rightarrow (t, e')$ si et seulement s'il existe un contexte[5] C, une règle $(A, d) \to a((B_1, d'_1), \ldots, (B_n, d'_n))$ dans R avec des variables fraîches dans les structures $d, d'_1, \ldots, d'_n$ et une u-substitution σ tels que

$$s = C[(A, d')],\ \sigma = \mathsf{mgu}(d, e(d')),\ t = C[a((B_1, \sigma(d'_1)), \ldots, (B_n, \sigma(d'_n)))] \text{ et } e' = \sigma \circ e$$

[3]Étant données deux structures de traits d et d' de $\mathcal{D}$, on désigne par l'u-substitution $\sigma = \mathsf{mgu}(d, d')$ leur *unificateur le plus général* s'il existe. Nous notons $\top$ l'élément le plus général de $\mathcal{D}$, et id l'u-substitution identité.

[4]L'ensemble des termes sur l'alphabet $\mathcal{F}$ et l'ensemble de variables $\mathcal{X}$ est noté $T(\mathcal{F}, \mathcal{X})$; en particulier $T(\mathcal{F}, \emptyset) = T(\mathcal{F})$ est l'ensemble des arbres sur $\mathcal{F}$.

[5]Un contexte C de $T(\mathcal{F}, \mathcal{X})$ est un terme de $T(\mathcal{F}, \mathcal{X} \cup \{x\})$, $x \notin \mathcal{X}$, qui ne contient qu'une seule occurrence de x, et le terme $C[t]$ pour un terme t de $T(\mathcal{F}, \mathcal{X})$ est obtenu en remplaçant cette occurrence par t dans C.

Le langage généré par G est $L(G) = \{t \in T(\mathcal{F}) \mid \exists e, ((S, \top), id) \Rightarrow^* (t, e)\}$. □

La propagation des unifications de traits se fait hiérarchiquement par la recherche de l'unificateur le plus général mgu à chaque étape de dérivation. L'u-substitution e globale associée comme environnement à notre terme sert à communiquer les résultats des unifications dans les différentes branches du terme.

Traduction de TAG vers RTG avec traits Munis de cette définition opérationnelle d'une RTG avec unification, nous enrichissons notre traduction de TAG vers RTG pour tenir compte des structures de traits des nœuds des arbres TAG. Nous définissons $\mathsf{feats}(\gamma_i)$ comme la structure de traits de $\mathcal{D}$ associée au nœud γ_i de l'arbre élémentaire γ. Cette structure est composée de deux ensembles hauts et bas de traits atomiques $\mathsf{top}(\gamma_i)$ et $\mathsf{bot}(\gamma_i)$.

La traduction est établie sur la notion d'*interface* $\mathsf{in}(\gamma)$ offerte par chaque arbre élémentaire TAG γ, qui servira de structure de traits de la partie gauche des règles de la grammaire rationnelle d'arbres avec traits. Dans le cas d'un arbre initial α, la structure $[top : \mathsf{top}(\alpha_r)]$ doit s'unifier avec celle du nœud de substitution. Dans le cas d'un arbre auxiliaire β, la structure $[top : \mathsf{top}(\beta_r), bot : \mathsf{bot}(\beta_f)]$ doit s'unifier avec celle du nœud d'adjonction. Il reste à coindexer ces interfaces avec les structures de la partie droite de chaque règle ; le seul cas à traiter est celui de la racine de l'arbre élémentaire, pour laquelle nous définissons une fonction feats_r. Nous définissons ainsi pour tout α dans I, β dans A et γ dans $I \cup A$, à l'aide d'une variable fraîche t

$$\mathsf{in}(\alpha) = \left[\begin{smallmatrix} top : t \\ top : \mathsf{top}(\alpha_r) \end{smallmatrix}\right] \tag{2}$$

$$\mathsf{in}(\beta) = \left[\begin{smallmatrix} top : t \\ top : \mathsf{top}(\beta_r) \\ bot : \mathsf{bot}(\beta_f) \end{smallmatrix}\right] \tag{3}$$

$$\mathsf{feats}_r(\gamma_1) = \left[\begin{smallmatrix} top : t \\ bot : \mathsf{bot}(\gamma_1) \end{smallmatrix}\right] \tag{4}$$

Pour un nœud γ_i, nous définissons $\mathsf{tr}(\gamma_i) = (\mathsf{nt}(\gamma_i), \mathsf{feats}(\gamma_i))$ et $\mathsf{tr}_r(\gamma_1) = (\mathsf{nt}(\gamma_1), \mathsf{feats}_r(\gamma_1))$. L'ensemble de règles de notre grammaire rationnelle d'arbres avec traits pour une grammaire TAG $\langle \Sigma, N, I, A, S \rangle$ est alors

$$\begin{array}{ll} & \{(X, \mathsf{in}(\alpha)) \to \alpha(\mathsf{tr}_r(\alpha_1), \mathsf{tr}(\alpha_2), \ldots, \mathsf{tr}(\alpha_n)) \mid \alpha \in I, n = \mathsf{rg}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\ \cup & \{(X_A, \mathsf{in}(\beta)) \to \beta(\mathsf{tr}_r(\beta_1), \mathsf{tr}(\beta_2), \ldots, \mathsf{tr}(\beta_n)) \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \\ \cup & \{X_A \left[\begin{smallmatrix} top : x \\ bot : x \end{smallmatrix}\right] \to \varepsilon \mid X_A \in N_A, x \text{ variable de } \mathcal{D}\} \end{array} \tag{5}$$

Les règles dérivant la feuille vide ε effectuent l'unification finale entre traits hauts et bas des nœuds de la grammaire TAG.

Nous obtenons alors l'ensemble de règles suivant pour la grammaire rationnelle enrichie de structures de traits correspondant à la grammaire TAG de la figure 2 :

$$\begin{aligned} (S, \top) &\to \text{creux}\left(N\left[\begin{smallmatrix} top : \left[\begin{smallmatrix} det : + \\ num : x \end{smallmatrix}\right] \end{smallmatrix}\right], V\left[\begin{smallmatrix} top : [num : x] \end{smallmatrix}\right]\right) \\ V\left[\begin{smallmatrix} top : t \end{smallmatrix}\right] &\to \text{est}\left(V_A\left[\begin{smallmatrix} top : t \\ bot : [num : \text{sg}] \end{smallmatrix}\right]\right) \\ V_A\left[\begin{smallmatrix} top : x \\ bot : x \end{smallmatrix}\right] &\to \varepsilon \\ N\left[\begin{smallmatrix} top : t \end{smallmatrix}\right] &\to \text{radis}\left(N_A\left[\begin{smallmatrix} top : t \\ bot : [det : -] \end{smallmatrix}\right]\right) \\ N_A\left[\begin{smallmatrix} top : t \\ bot : \left[\begin{smallmatrix} det : - \\ num : x \end{smallmatrix}\right] \end{smallmatrix}\right] &\to \text{gris}\left(N_A\left[\begin{smallmatrix} top : t \\ bot : \left[\begin{smallmatrix} det : - \\ num : x \end{smallmatrix}\right] \end{smallmatrix}\right]\right) \\ N_A\left[\begin{smallmatrix} top : t \\ bot : \left[\begin{smallmatrix} det : - \\ num : \text{sg} \end{smallmatrix}\right] \end{smallmatrix}\right] &\to \text{le}\left(N_A\left[\begin{smallmatrix} top : t \\ bot : \left[\begin{smallmatrix} det : + \\ num : \text{sg} \end{smallmatrix}\right] \end{smallmatrix}\right]\right) \\ N_A\left[\begin{smallmatrix} top : x \\ bot : x \end{smallmatrix}\right] &\to \varepsilon \end{aligned} \tag{6}$$

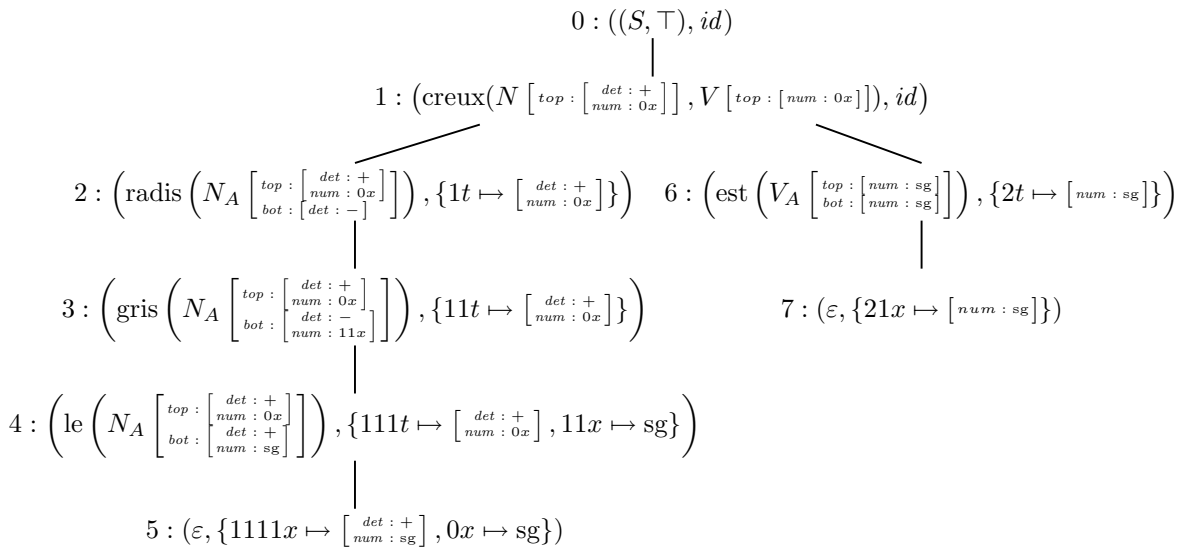


FIG. 4: Une dérivation dans la RTG enrichie pour la phrase « Le radis gris est creux. »

Exemple de dérivation Nous reprenons dans la figure 4 le cas de la phrase « Le radis gris est creux. » en employant les règles enrichies de structures de traits de l'équation (6). Chaque nœud de l'arbre de la figure est constitué d'une étiquette et d'un couple formé d'un terme de $T(\mathcal{F}, (N \cup N_A) \times \mathcal{D})$ et d'un environnement[6]. La création de variables fraîches utilise l'adresse de Gorn du nœud où la réécriture a lieu. Les étiquettes de chaque nœud indiquent l'ordre dans lequel s'effectuent les surréductions. Enfin, l'on remplace les variables par leur valeur associée dans l'environnement dès que possible.

On ne peut pas dériver l'arbre correspondant à « * Le radis gris sont creux. ». La partie gauche de la tentative de dérivation aurait été similaire. En revanche, dans la partie droite, le trait bot associé au nœud V_A de *sont* aurait eu pour valeur $num : \mathrm{pl}$ (pluriel). L'analyse aurait donc échoué à l'étape suivante, puisqu'en atteignant la feuille ε il aurait fallu unifier des traits top et bot avec respectivement sg et pl comme valeurs de num.

Bien sûr, pour une analyse qui visite d'abord le sous-arbre droit avant le sous-arbre gauche, le résultat serait le même, avec encore pour étape décisive du point de vue de l'unification la réécriture finale à ε.

3 Transformation par coin gauche

Comme nous venons de le voir, la génération d'un arbre de dérivation TAG à l'aide d'une grammaire RTG avec traits n'est pas très prédictive, dans le sens où il est nécessaire de patienter jusqu'à la réécriture à ε pour vérifier si une substitution réussit. Dans l'exemple de la figure 4, la substitution de « radis » dans « creux » n'est véritablement entérinée qu'au moment de la réécriture à ε, et potentiellement toutes les opérations intermédiaires seraient à défaire si cette réécriture n'avait pas été possible.

Le seul filtrage immédiatement exercé par l'arbre « radis » lors de sa substitution au nœud N de « creux » est l'unification de sa structure top avec la structure top de N. Or, l'arbre « radis »

[6]Nous ne faisons apparaître que les parties de l'environnement calculées à ce point de la dérivation.

suit l'usage dans les grammaires TAG, qui est que sa structure top est vide, et il n'y a en fait aucun filtrage par ce biais.

Nous présentons dans cette section une transformation du langage d'arbres de dérivation qui permet d'inverser l'ordre des réécritures, en commençant par ε, en opérant à toutes les adjonctions à la racine, et en finissant par l'arbre initial. Comme nous avons convenu que la racine d'un arbre élémentaire TAG apparaissait en fils gauche dans nos arbres de dérivation, cette transformation revient à une transformation par coin gauche (Rosenkrantz & Lewis II, 1970) appliquée à nos grammaires rationnelles d'arbres de dérivation. Cette transformation est simple, et nous semble plus naturelle que la transformation correspondante sur les arbres dérivés.

3.1 Grammaire rationnelle transformée

Les règles que nous souhaitons transformer sont de la forme $X \to \alpha(X_A, \dots)$, $X_A \to \beta(X_A, \dots)$ ou $X_A \to \varepsilon$. À l'issue de la transformation, un appel à X devra commencer par invoquer ε, puis les adjonctions β en ordre inverse, et enfin α en dernier lieu, dont l'arité est décrémentée. Pour notre grammaire (figure 3a), cela revient simplement à utiliser de nouveaux non-terminaux N_S et V_S et les règles

$$\begin{aligned} S &\to \text{creux}(N_S, V_S) \\ N_S &\to \varepsilon(N) \\ N &\to \text{radis} \mid \text{gris}(N) \mid \text{le}(N) \\ V_S &\to \varepsilon(V) \\ V &\to \text{est} \end{aligned} \tag{7}$$

Il manque à ces règles la possibilité d'une adjonction ailleurs qu'à la racine d'un arbre initial ; il suffit alors de conserver les règles $X_A \to \beta(X_A, \dots)$ et $X_A \to \varepsilon$ qui s'appliqueront comme auparavant.

Nous pouvons ensuite éliminer les ε-termes ; la grammaire de la figure 3a transformée est alors :

$$\begin{aligned} S &\to \text{creux}(N, V) \\ N &\to \text{radis} \mid \text{gris}(N) \mid \text{le}(N) \\ N_A &\to \text{gris}(N_A) \mid \text{le}(N_A) \mid \varepsilon \\ V &\to \text{est} \end{aligned} \tag{8}$$

Les règles de dérivation de N_A sont cependant inutiles puisqu'il n'y a jamais d'adjonction sur un nœud de catégorie N qui n'est pas une racine dans notre grammaire d'arbres adjoints.

Formellement, étant donnée une grammaire d'arbres adjoints $\langle \Sigma, N, I, A, S \rangle$, sa grammaire rationnelle d'arbres de dérivation transformée par coin gauche $G_{\text{lc}} = \langle S, N \cup N_A, \mathcal{F}_{\text{lc}}, R_{\text{lc}} \rangle$ utilise un alphabet terminal $\mathcal{F}_{\text{lc}} = I \cup A \cup \{\varepsilon\}$ mais où l'arité d'un arbre initial α est $\mathsf{rg}(\alpha) - 1$, et un ensemble de règles R_{lc} défini comme l'union

$$\begin{aligned} & \{X \to \alpha(\mathsf{nt}(\alpha_2), \dots, \mathsf{nt}(\alpha_n)) \mid \alpha \in I, n = \mathsf{rg}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\ \cup\ & \{X \to \beta(X, \mathsf{nt}(\beta_2) \dots, \mathsf{nt}(\beta_n)) \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \\ \cup\ & \{X_A \to \beta(\mathsf{nt}(\beta_1), \dots, \mathsf{nt}(\beta_n)) \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \end{aligned} \tag{9}$$

La taille de cette grammaire est au pire doublée par rapport à la grammaire rationnelle d'arbres de dérivation puisque chaque arbre auxiliaire apparaît maintenant deux fois. En pratique, les règles utiles dans la grammaire obtenue sont probablement moins nombreuses. Par exemple,

dans la grammaire SEMFRAG et en se basant sur l'existence de nœuds d'adjonction ailleurs qu'à la racine pour chaque catégorie syntaxique, seuls un tiers des arbres auxiliaires, soit encore un dixième des arbres élémentaires, est concerné par cette duplication.

Notons enfin que la transformation est aisément réversible. Nous définissons pour cela la fonction lc^{-1} de $T(\mathcal{F}_{\mathrm{lc}})$ dans $T(\mathcal{F})$ par

$$\mathsf{lc}^{-1}(t) = \mathsf{revlc}(t, \varepsilon) \tag{10}$$

$$\mathsf{revlc}(\beta(t_1, t_2, \ldots, t_n), t) = \mathsf{revlc}(t_1, \beta(t, f_{\beta_2}(t_2), \ldots, f_{\beta_n}(t_n))) \tag{11}$$

$$\mathsf{revlc}(\alpha(t_1, \ldots, t_n), t) = \alpha(t, f_{\alpha_2}(t_1), \ldots, f_{\alpha_{n+1}}(t_n)) \tag{12}$$

$$f_{\gamma_i}(t) = \begin{cases} \mathsf{recur}(t) & \text{si } \gamma_i \text{ est un nœud d'adjonction} \\ \mathsf{lc}^{-1}(t) & \text{si } \gamma_i \text{ est un nœud de substitution} \end{cases} \tag{13}$$

$$\mathsf{recur}(\gamma(t_1, \ldots, t_n)) = \gamma(f_{\gamma_1}(t_1), \ldots, f_{\gamma_n}(t_n)) \tag{14}$$

On peut ainsi procéder à la génération d'un arbre dérivé dans $L(G_{\mathrm{lc}})$ et retrouver l'arbre correspondant de $L(G)$ en lui appliquant lc^{-1}.

3.2 Unification dans la grammaire transformée

Nous procédons maintenant à la définition d'une grammaire rationnelle d'arbres de dérivation transformée par coin gauche avec structures de traits. En reprenant les règles transformées (7) de la section 3.1, nous obtenons dans un premier temps les règles transformées avec structures de traits

$$\begin{aligned}
(S, \top) &\to \text{creux}\left(N_S\left[\mathit{top} : \left[\begin{smallmatrix}\mathit{det} : +\\ \mathit{num} : x\end{smallmatrix}\right]\right], V\left[\mathit{top} : [\mathit{num} : x]\right]\right) \\
N_S\left[\mathit{top} : t\right] &\to \varepsilon\left(N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : t\end{smallmatrix}\right]\right) \\
N\left[\mathit{bot} : [\mathit{det} : -]\right] &\to \text{radis} \\
N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right] &\to \text{gris}\left(N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right]\right) \\
N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : +\\ \mathit{num} : \mathrm{sg}\end{smallmatrix}\right]\end{smallmatrix}\right] &\to \text{le}\left(N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : \mathrm{sg}\end{smallmatrix}\right]\end{smallmatrix}\right]\right) \\
V_S\left[\mathit{top} : t\right] &\to \varepsilon\left(V\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : t\end{smallmatrix}\right]\right) \\
V\left[\mathit{bot} : [\mathit{num} : \mathrm{sg}]\right] &\to \text{est}
\end{aligned} \tag{15}$$

Comme la récursion au sein des arbres auxiliaires est inversée, les structures de traits de la partie gauche de chaque règle sont les structures de son nœud racine dans la grammaire TAG, et inversement (on observe ce changement pour la règle qui dérive « le »).

Nous pouvons comme auparavant éliminer les règles dérivant ε, ce qui a pour effet de copier la structure de traits top des nœuds de substitution dans la structure bot. Nous obtenons l'ensemble de règles suivant pour la grammaire TAG de la figure 2 :

$$\begin{aligned}
(S, \top) &\to \text{creux}\left(N\left[\begin{smallmatrix}\mathit{top} : \left[\begin{smallmatrix}\mathit{det} : +\\ \mathit{num} : x\end{smallmatrix}\right]\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : +\\ \mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right], V\left[\begin{smallmatrix}\mathit{top} : \left[\begin{smallmatrix}\mathit{num} : x\end{smallmatrix}\right]\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right]\right) \\
N\left[\mathit{bot} : [\mathit{det} : -]\right] &\to \text{radis} \\
N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right] &\to \text{gris}\left(N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : x\end{smallmatrix}\right]\end{smallmatrix}\right]\right) \\
N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : +\\ \mathit{num} : \mathrm{sg}\end{smallmatrix}\right]\end{smallmatrix}\right] &\to \text{le}\left(N\left[\begin{smallmatrix}\mathit{top} : t\\ \mathit{bot} : \left[\begin{smallmatrix}\mathit{det} : -\\ \mathit{num} : \mathrm{sg}\end{smallmatrix}\right]\end{smallmatrix}\right]\right) \\
V\left[\mathit{bot} : [\mathit{num} : \mathrm{sg}]\right] &\to \text{est}
\end{aligned} \tag{16}$$

Cette grammaire d'arbres avec traits est bien plus lisible que celle décrite dans l'équation (6) : le premier fils de « creux » ne peut être que « le » de par la présence du trait $det = +$ dans la structure bot associée à N. Les seuls fils de « le » possibles sont « gris » et « radis », seuls compatibles avec le trait $det = -$. Le filtrage dû aux unifications est maintenant immédiat.

Construction de la grammaire rationnelle transformée Nous définissons les variantes suivantes des fonctions de calcul de structures de traits, pour tout arbre auxiliaire β de A et pour tout nœud γ_i d'un arbre élémentaire γ de $I \cup A$:

$$\mathsf{in}_{\mathrm{lc}}(\beta) = \left[\begin{smallmatrix} top : t \\ bot : \mathsf{bot}(\beta_f) \end{smallmatrix}\right] \tag{17}$$

$$\mathsf{feats}_{\mathrm{lc}}(\gamma_i) = \begin{cases} \left[\begin{smallmatrix} top : \mathsf{top}(\gamma_i) \\ bot : \mathsf{top}(\gamma_i) \end{smallmatrix}\right] & \text{si } \gamma_i \text{ est un nœud de substitution,} \\ \left[\begin{smallmatrix} top : t \\ top : \mathsf{top}(\gamma_r) \\ bot : \mathsf{bot}(\gamma_r) \end{smallmatrix}\right] & \text{si } \gamma_i = \gamma_r, \\ \mathsf{feats}(\gamma_i) & \text{sinon.} \end{cases} \tag{18}$$

Pour un nœud γ_i, nous notons $\mathsf{tr}_{\mathrm{lc}}(\gamma_i)$ la paire $(\mathsf{nt}(\gamma_i), \mathsf{feats}_{\mathrm{lc}}(\gamma_i))$.

Formellement, l'ensemble de règles de notre grammaire rationnelle d'arbres avec traits transformée pour une grammaire TAG $\langle \Sigma, N, I, A, S \rangle$ est alors

$$\begin{aligned} &\{(X, \mathsf{feats}(\alpha_1)) \rightarrow \alpha(\mathsf{tr}_{\mathrm{lc}}(\alpha_2), \ldots, \mathsf{tr}_{\mathrm{lc}}(\alpha_n)) \mid \alpha \in I, n = \mathsf{rg}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\ \cup\, &\{(X, \mathsf{feats}_{\mathrm{lc}}(\beta_1)) \rightarrow \beta((X, \mathsf{in}_{\mathrm{lc}}(\beta)), \mathsf{tr}_{\mathrm{lc}}(\beta_2), \ldots, \mathsf{tr}_{\mathrm{lc}}(\beta_n)) \\ &\qquad\qquad \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \\ \cup\, &\{(X_A, \mathsf{in}(\beta)) \rightarrow \beta(\mathsf{tr}_r(\beta_1), \mathsf{tr}_{\mathrm{lc}}(\beta_2), \ldots, \mathsf{tr}_{\mathrm{lc}}(\beta_n)) \mid \beta \in A, n = \mathsf{rg}(\beta), X = \mathsf{lab}(\beta_r)\} \end{aligned} \tag{19}$$

4 Conclusion

Les grammaires rationnelles d'arbres avec structures de traits permettent de générer aisément les arbres de dérivation d'une grammaire TAG avec structures de traits. Les grammaires transformées par coin gauche permettent de plus de filtrer plus efficacement les opérations d'adjonction et de substitution possibles à partir d'un arbre élémentaire.

Si des calculs d'unification sur arbres de dérivation ont déjà été considérés par le passé de manière spécialisée (Kallmeyer & Romero, 2004), les mécanismes que nous avons définis sont suffisamment généraux pour traduire fidèlement l'unification dans les grammaires d'arbres adjoints.

Parmi les perspectives ouvertes par ce traitement des structures de traits dans les arbres de dérivation, on pourra mentionner des calculs d'accessibilité plus fins entre les arbres élémentaires, utiles par exemple pour vérifier qu'une TAG est dans la classe restreinte des grammaires d'arbres par insertion (Schabes & Waters, 1995, TIG) ou sous forme rationnelle (Rogers, 1994, RFTAG). On pourrait par ailleurs imaginer étendre notre approche à l'analyse syntaxique, pour peu que les informations topologiques d'ordre entre les ancres soient calculées dans nos arbres de dérivation (Kuhlmann, 2007).

Références

CANDITO M.-H. & KAHANE S. (1998). Une grammaire TAG vue comme une grammaire Sens-Texte précompilée. In P. ZWEIGENBAUM, Ed., *TALN'98*, p. 102–111: ATALA.

COMON H., DAUCHET M., GILLERON R., LÖDING C., JACQUEMARD F., LUGIEZ D., TISON S. & TOMMASI M. (2007). *Tree Automata Techniques and Applications*.

DE GROOTE P. (2002). Tree-adjoining grammars as abstract categorial grammars. In R. FRANK, Ed., *TAG+6*, p. 145–150.

GARDENT C. (2006). Intégration d'une dimension sémantique dans les grammaires d'arbres adjoints. In P. MERTENS, C. FAIRON, A. DISTER & P. WATRIN, Eds., *TALN'06*, p. 149–158: Presses universitaires de Louvain.

GARDENT C. & KALLMEYER L. (2003). Semantic construction in feature-based TAG. In *EACL'03*, p. 123–130: ACL Press.

HANUS M. (1994). The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, **19–20**, 583–628.

JOSHI A. K. & SCHABES Y. (1997). Tree-adjoining grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages*, volume 3: Beyond Words, chapter 2, p. 69–124. Springer.

KALLMEYER L. & ROMERO M. (2004). LTAG semantics with semantic unification. In O. RAMBOW & M. STONE, Eds., *TAG+7*, p. 155–162.

KANAZAWA M. (2007). Parsing and generation as Datalog queries. In *ACL'07*, p. 176–183: ACL Press.

KOLLER A. & STONE M. (2007). Sentence generation as a planning problem. In *ACL'07*, p. 336–343: ACL Press.

KOLLER A. & STRIEGNITZ K. (2002). Generation as dependency parsing. In *ACL'02*, p. 17–24: ACL Press.

KUHLMANN M. (2007). *Dependency Structures and Lexicalized Grammars*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

POGODALLA S. (2004). Vers un statut de l'arbre de dérivation : exemples de construction de representations sémantiques pour les grammaires d'arbres adjoints. In P. BLACHE, Ed., *TALN'04*, p. 377–386: LPL.

ROBINSON J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**(1), 23–41.

ROGERS J. (1994). Capturing CFLs with tree adjoining grammars. In *ACL'94*, p. 155–162: ACL Press.

ROSENKRANTZ D. J. & LEWIS II P. M. (1970). Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory*, p. 139–152: IEEE Computer Society.

SCHABES Y. & WATERS R. C. (1995). Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, **21**(4), 479–513.

VIJAY-SHANKER K. (1992). Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, **18**(4), 481–517.