Convertir des grammaires d'arbres adjoints à composantes multiples avec tuples d'arbres (TT-MCTAG) en grammaires à concaténation d'intervalles (RCG)

Laura Kallmeyer, Yannick Parmentier

TALN 2008, Avignon, 9–13 juin 2008

Convertir des grammaires d'arbres adjoints à composantes multiples avec tuples d'arbres (TT-MCTAG) en grammaires à concaténation d'intervalles (RCG)

Laura Kallmeyer[1] Yannick Parmentier[2]
(1) SFB 441 / Université de Tübingen
(2) SfS-CL / SFB 441 / Université de Tübingen
Nauklerstr. 35, 72074 Tübingen, Allemagne
lk@sfs.uni-tuebingen.de, parmenti@sfs.uni-tuebingen.de

Résumé. Cet article étudie la relation entre les grammaires d'arbres adjoints à composantes multiples avec tuples d'arbres (TT-MCTAG), un formalisme utilisé en linguistique informatique, et les grammaires à concaténation d'intervalles (RCG). Les RCGs sont connues pour décrire exactement la classe PTIME, il a en outre été démontré que les RCGs « simples » sont même équivalentes aux systèmes de réécriture hors-contextes linéaires (LCFRS), en d'autres termes, elles sont légèrement sensibles au contexte. TT-MCTAG a été proposé pour modéliser les langages à ordre des mots libre. En général ces langages sont NP-complets. Dans cet article, nous définissons une contrainte additionnelle sur les dérivations autorisées par le formalisme TT-MCTAG. Nous montrons ensuite comment cette forme restreinte de TT-MCTAG peut être convertie en une RCG simple équivalente. Le résultat est intéressant pour des raisons théoriques (puisqu'il montre que la forme restreinte de TT-MCTAG est légèrement sensible au contexte), mais également pour des raisons pratiques (la transformation proposée ici a été utilisée pour implanter un analyseur pour TT-MCTAG).

Abstract. This paper investigates the relation between TT-MCTAG, a formalism used in computational linguistics, and RCG. RCGs are known to describe exactly the class PTIME ; « simple » RCG even have been shown to be equivalent to linear context-free rewriting systems, i.e., to be mildly context-sensitive. TT-MCTAG has been proposed to model free word order languages. In general, it is NP-complete. In this paper, we will put an additional limitation on the derivations licensed in TT-MCTAG. We show that TT-MCTAG with this additional limitation can be transformed into equivalent simple RCGs. This result is interesting for theoretical reasons (since it shows that TT-MCTAG in this limited form is mildly context-sensitive) and also for practical reasons (the proposed transformation has been used for implementing a parser for TT-MCTAG).

Mots-clés : Grammaires d'arbres adjoints à composantes multiples, grammaires à concaténation d'intervalles, légère sensibilité au contexte.

Keywords: Multicomponent Tree Adjoining Grammars, Range Concatenation Grammars, mild context-sensitivity.

1 Introduction

Il a été démontré que les grammaires d'arbre adjoints (TAGs) sont d'un grand interêt pour le traitement automatique des langues naturelles, et pour des raisons linguistiques (domaine de localité étendu), et pour des raisons formelles (complexité d'analyse). Pourtant, elles sont trop limitées dans leur expressivité pour traiter certains phenomènes linguistiques comme par exemple le brouillage d'arguments dans des langues dites à ordre de mots libre.

Cet article considère une variante des TAGs, les grammaires d'arbres adjoints à composantes multiples avec tuples d'arbres (TT-MCTAGs), cherchant à résoudre ce problème d'expressivité tout en gardant les propriétés principales des TAGs. Dans ce contexte, nous traitons les propriétés formelles de ce formalisme, en particulier sa relation avec les langages legèrement sensibles au contexte. En donnant la construction d'une grammaire à concaténation d'intervalles (RCG) simple pour une TT-MCTAG qui satisfait une certaine contrainte, nous obtenons deux résultats : d'abord nous prouvons que ces TT-MCTAGs restreintes sont en fait légèrement sensibles au contexte. Ensuite, en combinant cette construction avec un analyseur pour RCG, nous pouvons développer un analyseur pour TT-MCTAG. Nous pensons que l'idée de passer par les RCGs pour analyser certains formalismes grammaticaux au lieu de développer un analyseur différent pour chacun d'entre eux peut être utile d'une façon générale.

Le plan de cet article est le suivant. Dans un premier temps, nous définissons les formalismes sur lesquels nous nous basons, à savoir TAG (section 1.1) et RCG (section 1.2), et présentons brièvement un algorithme existant de conversion de TAG vers RCG (section 1.3). Ensuite, nous introduisons le formalisme des TT-MCTAGs (section 2), ainsi que leur forme restreinte. Finalement nous donnons l'algorithme de conversion de ces grammaires en RCG (section 3).

1.1 Tree Adjoining Grammars (TAG)

Les TAGs (Joshi & Schabes, 1997) sont des systèmes de réécriture d'arbres. Une TAG est un ensemble fini d'arbres (les arbres élémentaires) avec des étiquettes non-terminales et terminales (les derniers seulement pour des feuilles). Des arbres plus grands sont générés par substitution (remplacement d'une feuille par un nouvel arbre) et adjonction (remplacement d'un nœud interne par un nouvel arbre). En cas d'adjonction, l'arbre que l'on adjoint a exactement une feuille marquée comme nœud pied (marquée par une étoile). Un tel arbre s'appelle un arbre *auxiliaire*. Si cet arbre est adjoint à un nœud n, alors dans l'arbre résultat, le sous-arbre de racine n s'attache au nœud pied de l'arbre auxiliaire. Des arbres non-auxiliaires s'appellent des arbres *initiaux*. Chaque dérivation commence par un arbre initial. Dans un arbre dérivé terminal, toutes les feuilles doivent avoir des étiquettes terminales. Un exemple est donné Fig. 1.

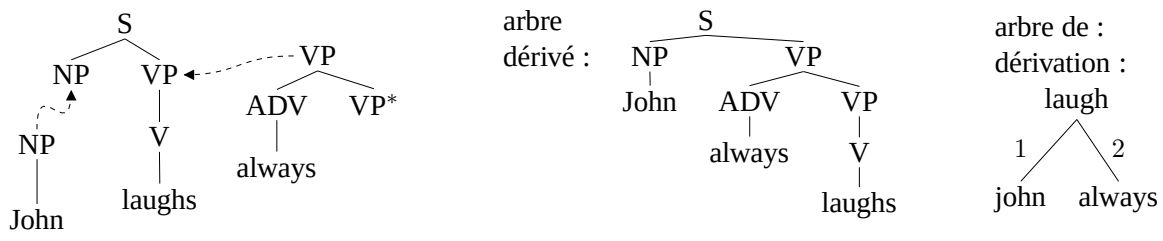


FIG. 1 – Dérivation TAG pour la phrase *John always laughs*

Définition 1. *Une* TAG *est un tuple $G = \langle I, A, N, T\rangle$ avec (i) N et T des alphabets disjoints, les nonterminaux et les terminaux, et (ii) I et A des ensembles finis d'arbres initiaux et auxiliaires respectivement avec non-terminaux N et terminaux T. En outre, les nœuds internes dans $I \cup A$ peuvent être marqués OA (adjonction obligatoire) ou NA (adjonction interdite).*

Définition 2. *Soient $G = \langle I, A, N, T\rangle$ une TAG, γ et γ' des arbres finis.*
- *$\gamma \Rightarrow \gamma'$ dans G ssi il existe une adresse de nœud p et un arbre γ'_0 qui est soit un arbre élémentaire, soit dérivé d'un arbre élémentaire tel que $\gamma' = \gamma[p, \gamma'_0]$[1]. $\stackrel{*}{\Rightarrow}$ est la clôture réflexive et transitive de $\Rightarrow$.*
- *Le langage d'arbres de G est $L_T(G) = \{\gamma \mid \alpha \stackrel{*}{\Rightarrow} \gamma$ pour quelque $\alpha \in I$, toutes les feuilles dans γ ont une étiquette terminale et γ ne contient pas de nœud marqué $OA\}$.*

Les dérivations en TAG sont représentées par des arbres de dérivation (des arbres non-ordonnés) qui décrivent comment les arbres élémentaires ont été combinés. L'arbre dérivé est alors le résultat de l'exécution de ces combinaisons, *i.e.*, l'arbre de dérivation décrit de façon unique un seul arbre dérivé. Chaque arc dans un arbre de dérivation représente une adjonction ou une substitution. Les arcs sont etiquetés avec des adresses de nœud de Gorn[2]. L'arbre de dérivation de la Fig. 1 par exemple, indique que l'arbre élémentarire de *John* a été substitué au nœud d'adresse 1 et *always* a été adjoint à l'adresse 2 (le fait que le premier est une substitution, le deuxième une adjonction peut être déduit du fait que l'adresse 1 est l'adresse d'une feuille tandis que 2 est l'adresse d'un nœud interne).

Définition 3. *Soit $G = \langle I, A, N, T\rangle$ une TAG, on définit γ un arbre dérivé d'un arbre élémentaire γ_0 dans G comme suit : $\gamma = \gamma_0[p_1, \gamma_1] \dots [p_k, \gamma_k]$ tel que les substitutions/adjonctions des $\gamma_1, \dots, \gamma_k$ sont les seules substitutions/adjunctions à γ_0 exécutées lors de la dérivation de γ. Dans ce cas-là, l'*arbre de dérivation *correspondant a une racine étiquetée γ_0 qui a k fils. Les arcs entre γ_0 et ces fils sont étiquetés $p_1, \dots, p_k$, et les fils sont les arbres de dérivation des dérivations de $\gamma_1, \dots, \gamma_k$.*

1.2 Range Concatenation Grammars (RCG)

Définition 4. *Une* RCG positive *(Boullier, 1999b; Boullier, 2000) est un tuple $G = \langle N, T, V, S, P\rangle$ tel que (i) N est un ensemble fini de prédicats, chacun d'arité fixe, $S \in N$ (d'arité 1) le prédicat initial, (ii) T et V sont des alphabets disjoints de terminaux et de variables, et (iii) P est un ensemble fini de* clauses *de la forme :*

$$A_0(x_{01}, \dots, x_{0a_0}) \rightarrow \epsilon$$
ou $$A_0(x_{01}, \dots, x_{0a_0}) \rightarrow A_1(x_{11}, \dots, x_{1a_1}) \dots A_n(x_{n1}, \dots, x_{na_n})$$
avec $n \geq 1$, $A_i \in N, x_{ij} \in (T \cup V)^*$ et a_i l'arité d'A_i.

Notons qu'une RCG avec une arité maximale n s'appelle une RCG d'arité n. Notons également que, dans ce papier nous n'utilisons que des RCGs positives, ainsi, dans ce qui suit, chaque mention des RCGs refère aux RCGs positives[3]. Dans un appel de clause par rapport à une chaîne $w = t_1 \dots t_n$, les arguments des prédicats sont instantiés par des sous-chaînes de w,

[1] Pour des arbres $\gamma, \gamma_1, \dots, \gamma_n$ et des adresses de nœuds différentes deux à deux $p_1, \dots, p_n$ dans γ, $\gamma[p_1, \gamma_1] \dots [p_n, \gamma_n]$ est le résultat des substitutions/adjonctions séquentielles de $\gamma_1, \dots, \gamma_n$ respectivement aux nœuds d'adresses $p_1, \dots, p_n$ dans γ.

[2] L'adresses de la racine est ϵ, et le n-ième fils d'un nœud avec adresse p a l'adresse $p.j$.

[3] La variante négative permet des appels de prédicats de la forme $\overline{A(\alpha_1, \dots, \alpha_n)}$. Un tel prédicat reconnaît le complément de $A(\alpha_1, \dots, \alpha_n)$, voir (Boullier, 2000).

plus précisement par les intervalles correpondants. Un intervalle $\langle i, j\rangle$ avec $0 \leq i < j \leq n$ correspond à la chaîne entre les positions i et j, *i.e.*, la chaîne $t_{i+1} \dots t_j$. Si $i = j$, $\langle i, j\rangle$ correspond à ϵ. Si $i > j$, $\langle i, j\rangle$ n'est pas défini.

Définition 5. *Une* instantiation *d'une clause C par rapport à une chaîne $w = t_1 \dots t_n$ est une fonction $f : \{t' \mid t'$ est l'occurrence d'un $t \in T$ dans les arguments de $C\} \cup V \rightarrow \{\langle i, j\rangle \mid i \leq j, i, j \in \mathbb{N}\}$ tel que :*

- *a) pour chaque occurrence t' d'un $t \in T$ dans C : $f(t') := \langle i, i+1\rangle$ pour quelque i, $0 \leq i < n$ tel que $t_i = t$,*
- *b) pour chaque $v \in V$: $f(v) = \langle j, k\rangle$ pour quelques $0 \leq j \leq k \leq n$, et*
- *c) si pour un argument $x_1 \dots x_k$ d'un prédicat dans C, $f(x_1) = \langle i_1, j_1\rangle, \dots, f(x_k) = \langle i_k, j_k\rangle$, alors $j_m = i_{m+1}$ pour $1 \leq m < k$. Par définition, nous disons alors que $f(x_1 \dots x_k) = \langle i_1, j_k\rangle$.*

La relation de dérivation pour RCG est définie comme suit :

Définition 6. *Pour chaque clause C : s'il existe une instantiation de cette clause par rapport à un $w \in T^*$, on peut alors, dans un pas de dérivation ($\dots \Rightarrow \dots$), remplacer la partie gauche de cette instantiation par sa partie droite. $\overset{*}{\Rightarrow}$ est la clôture reflexive et transitive de $\Rightarrow$. Le langage des chaînes d'une RCG G est $L(G) = \{w \mid S(\langle 0, |w|\rangle) \overset{*}{\Rightarrow} \epsilon$ par rapport à $w\}$.*

Comme exemple, prenons $G = \langle\{S, A, B\}, \{a, b\}, \{X, Y, Z\}, S, P\rangle$ avec
$S(X\,Y\,Z) \rightarrow A(X, Z)\,B(Y)$, $A(a\,X, a\,Y) \rightarrow A(X, Y)$, $B(b\,X) \rightarrow B(X)$,
$A(\epsilon, \epsilon) \rightarrow \epsilon$, $B(\epsilon) \rightarrow \epsilon$. $L(G) = \{a^n b^k a^n \mid k, n \in \mathbb{N}\}$.

La dérivation pour $w = aabaa$ est la suivante.
La clause S est instantiée comme suit : $S(\langle 0, 5\rangle) \Rightarrow A(\langle 0, 2\rangle, \langle 3, 5\rangle)B(\langle 2, 3\rangle)$

$$\begin{array}{ccccccccc} S(X & & Y & & Z) & \rightarrow & A(X, & Z) & B(Y) \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & \downarrow & \downarrow \\ \langle 0,2\rangle & & \langle 2,3\rangle & & \langle 3,5\rangle & & \langle 0,2\rangle & \langle 3,5\rangle & \langle 2,3\rangle \\ aa & & b & & aa & & aa & aa & b \end{array}$$

Après cela, $A(\langle 0, 2\rangle, \langle 3, 5\rangle) \Rightarrow A(\langle 1, 2\rangle, \langle 4, 5\rangle) \Rightarrow A(\langle 2, 2\rangle, \langle 5, 5\rangle) \Rightarrow \epsilon$:

$$\begin{array}{c} A(aXaY) \rightarrow A(X,Y) \\ \langle 0,1\rangle\langle 1,2\rangle\langle 3,4\rangle\langle 4,5\rangle\langle 1,2\rangle\langle 4,5\rangle \\ a\quad a\quad a\quad a\quad a\quad a \end{array} \qquad \text{et} \quad \begin{array}{c} A(aXaY) \rightarrow A(X,Y) \\ \langle 1,2\rangle\langle 2,2\rangle\langle 4,5\rangle\langle 5,5\rangle\langle 2,2\rangle\langle 5,5\rangle \\ a\quad \epsilon\quad a\quad \epsilon\quad \epsilon\quad \epsilon \end{array} \qquad \text{et } A(\epsilon, \epsilon) \rightarrow \epsilon$$

Pour le second prédicat, nous avons $B(\langle 2, 3\rangle) \Rightarrow B(\langle 3, 3\rangle) \Rightarrow \epsilon$:

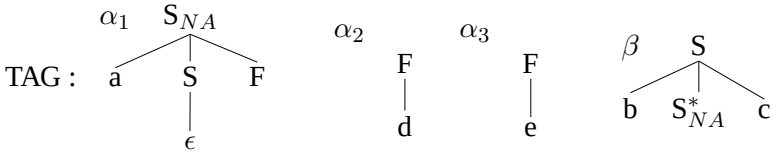
$$\begin{array}{ccccc} B(b & & X) & \rightarrow & B(X) \\ \downarrow & & \downarrow & & \downarrow \\ \langle 2,3\rangle & & \langle 3,3\rangle & & \langle 3,3\rangle \\ b & & \epsilon & & \epsilon \end{array} \qquad \text{et } B(\epsilon) \rightarrow \epsilon$$

Définition 7. *Une RCG est dite* non-combinatoire *si chaque argument d'un prédicat d'une partie droite de clause consiste en une seule variable. Elle est* linéaire *si aucune variable n'apparait plus d'une fois dans la partie gauche d'une clause ou plus d'une fois dans la partie droite d'une clause. Elle est* non-effaçante *si pour chaque clause, chaque variable dans la partie gauche se trouve également dans la partie droite, et vice versa. Une RCG est* simple *si elle est non-combinatoire, linéaire et non-effaçante.*

Les RCGs simples sont équivalentes aux systèmes de réécriture hors contextes linéaires (LCFRS, (Weir, 1988)), *c.f.* (Boullier, 1998). Par conséquent, les RCGs simples sont légèrement sensibles au contexte (Joshi, 1985).

1.3 De TAG à RCG

L'idée générale de la transformation d'une TAG en RCG (positive simple) proposée par (Boullier, 1999b; Boullier, 1999a) est la suivante : la RCG contient des prédicats de deux types, $\langle\alpha\rangle(X)$ et $\langle\beta\rangle(L, R)$, utilisés respectivement pour encoder les arbres initiaux et auxiliaires. X couvre la chaîne de l'arbre α en incluant tous les arbres ajoutés à α, tandis que L et R couvrent les parties de la chaîne de β (incluant tous les arbres ajoutés) qui se trouvent à gauche et à droite du nœud pied. Les clauses de la RCG réduisent les arguments de ces prédicats en identifiant la partie provenant de l'arbre α/β lui-même et les parties provenant des arbres ajoutés par substitution ou adjonction. Un exemple est donné Fig. 2.



RCG équivalente : $S(X) \rightarrow \langle\alpha_1\rangle(X) \mid \langle\alpha_2\rangle(X) \mid \langle\alpha_3\rangle(X)$ (chaque w est la chaîne d'un $\alpha \in I$)

$\langle\alpha_1\rangle(aF) \rightarrow \langle\alpha_2\rangle(F) \mid \langle\alpha_3\rangle(F)$ (la chaîne de α_1 est a suivie de la chaîne de l'arbre substitué au nœud F)

$\langle\alpha_1\rangle(aB_1B_2F) \rightarrow \langle\beta\rangle(B_1, B_2)\langle\alpha_2\rangle(F) \mid \langle\beta\rangle(B_1, B_2)\langle\alpha_3\rangle(F)$ (ou β est adjoint à S dans α ; la chaîne est alors a, suivie de la partie gauche de β, la partie droite de β et la chaîne substituée à F)

$\langle\beta\rangle(B_1b, cB_2) \rightarrow \langle\beta\rangle(B_1, B_2)$ (β peut être adjoint à sa racine ; alors la partie gauche est la partie gauche du β adjoint suivie de b ; la partie droite est c suivie par la partie droite du β adjoint)

$\langle\alpha_2\rangle(d) \rightarrow \epsilon \quad \langle\alpha_3\rangle(e) \rightarrow \epsilon \quad \langle\beta\rangle(b, c) \rightarrow \epsilon$ (les chaînes de α_2, α_3 et β sont d, e et la paire b (gauche) et c (droite) resp.)

FIG. 2 – Une TAG et la RCG équivalente.

2 TT-MCTAG

Pour représenter un ensemble de phenomènes linguistiques, une extension des grammaires d'arbres adjoints a été proposée, à savoir les TAGs à composantes multiples (MCTAG, (Weir, 1988)). Leur motivation réside dans le besoin de répartir la contribution d'un seul élément lexical (par exemple un verbe et ses arguments) sur plusieurs arbres élémentaires. Une MCTAG est donc composée d'ensembles d'arbres élémentaires. Si l'un de ces ensembles est utilisé lors d'une dérivation, chacun de ses éléments doit être utilisé.

Définition 8. *Une* MCTAG *est un tuple* $G = \langle I, A, N, T, \mathcal{A}\rangle$ *tel que* $G_{TAG} := \langle I, A, N, T\rangle$ *est une TAG, et* $\mathcal{A}$ *est une partition de* $I \cup A$.

Le type de MCTAG auquel nous sommes intéressé est *Tree-Tuple MCTAG with Shared Nodes* (TT-MCTAG, (Lichte, 2007)). Les TT-MCTAGs ont été introduites afin d'analyser des phénomènes d'ordre de mots libre dans des langues comme l'allemand. Un exemple est (1) où l'argument *es* de *reparieren* précède l'argument *der Mechaniker* de *verspricht* et n'avoisine donc pas le prédicat dont il dépend.

(1) ... dass es der Mechaniker zu reparieren verspricht
... que le le mécanicien réparer promet
'... que le mécanicien promet de le réparer'

Dans une TT-MCTAG, les ensembles élémentaires contiennent (1) un arbre lexicalisé γ, l'unique *arbre tête,* et (2) plusieurs arbres auxiliaires, les *arbres arguments*. Une liste composée d'une

tête et d'arguments s'appelle un tuple d'arbres. Lors de la dérivation, les arguments doivent soit s'adjoindre directement à leur tête, soit être liés par une chaîne d'adjonctions à des racines, à un arbre adjoint à leur tête. Autrement dit, dans l'arbre de dérivation TAG correpondant, la tête doit dominer ses arguments de telle manière que les adresses des nœuds sur le chemin, sauf la première, doivent être ϵ (*i.e.* le nœud racine). Cela traduit la notion d'adjonction avec partage de nœuds de (Kallmeyer, 2005).

Définition 9. *1. Une MCTAG $G = \langle I, A, N, T, \mathcal{A}\rangle$ est une* TT-MCTAG *ssi chaque $\Gamma \in \mathcal{A}$ a la forme $\{\gamma, \beta_1, \ldots, \beta_n\}$ ou γ (la* tête*) a au moins une feuille avec une étiquette terminale, et $\beta_1, \ldots, \beta_n$ sont des arbres auxiliaires, les* arguments*. Nous écrivons un tel ensemble $\langle\gamma, \{\beta_1, \ldots, \beta_n\}\rangle$. 2. Un arbre de dérivation D dans $\langle I, A, N, T\rangle$ est admis comme arbre de dérivation TAG dans G ssi*

(MC) *("multicomponent condition") Il existe k ($k \geq 1$) instances $\Gamma_1, \ldots, \Gamma_k$ de tuples élémentaires, différentes les unes des autres, tel que $\bigcup_{i=1}^{k} \Gamma_i$ est l'ensemble d'étiquettes dans D.*

(SN-TTL) *("tree-tuple locality with shared nodes") pour tous les nœuds $n_0, n_1, \ldots, n_m$ ($m > 1$) dans D avec l'étiquette du même tuple, tel que l'étiquette de n_0 est la tête : pour chaque $1 \leq i \leq m$: ou bien $\langle n_0, n_i\rangle \in \mathcal{P}_D$[4] ou bien il existe des $n_{i,1}, \ldots, n_{i,k}$ étiquettes d'arbres auxiliaires telles que $n_i = n_{i,k}$, $\langle n_0, n_{i,1}\rangle \in \mathcal{P}_D$ et pour $1 \leq j \leq k-1$: $\langle n_{i,j}, n_{i,j+1}\rangle \in \mathcal{P}_D$ avec ϵ comme étiquette d'arc.*

Sur la Fig. 3, l'arbre auxiliaire NP$_{nom}$ est adjoint directement à *verspricht* (sa tête) tandis que l'arbre NP$_{acc}$ est adjoint à la racine d'un arbre qui est adjoint à la racine d'un arbre qui est adjoint à *reparieren*.

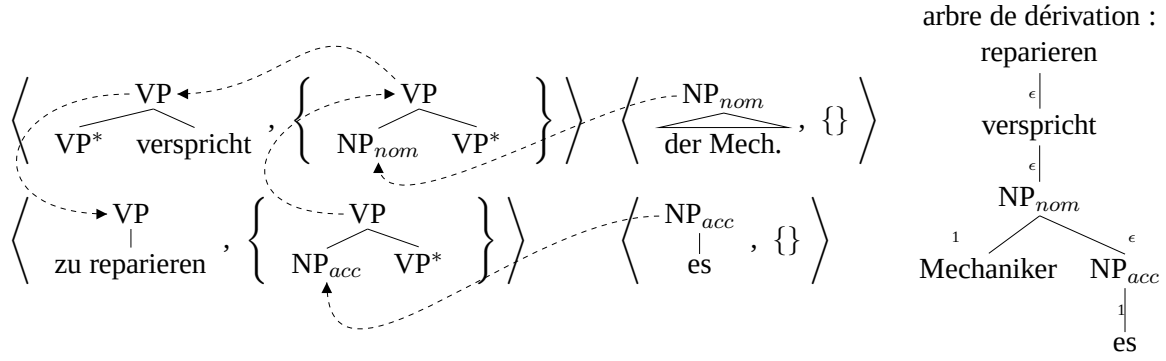


FIG. 3 – Dérivation en TT-MCTAG pour (1)

Le problème de la reconnaissance pour TT-MCTAG est NP-difficile (Søgaard *et al.*, 2007). Nous introduisons ici une contrainte supplémentaire pour les TT-MCTAG, fondée sur une idée de (Søgaard *et al.*, 2007) : les TT-MCTAG sont de rang k si, à chaque moment de la dérivation, le nombre d'arguments qui dépendent de têtes précédemment utilisées dans la dérivation et qui donc attendent leur adjonction, est limité à k.[5]

Définition 10. *Une TT-MCTAG $G = \langle I, A, N, T, \mathcal{A}\rangle$ est de rang k (appelée aussi k-*TT-MCTAG*) ssi pour chaque arbre de dérivation TAG D admis dans G :*
(TT-k) *Il n'existe pas de nœuds $n, h_0, \ldots, h_k, a_0, \ldots, a_k$ dans D tels que l'étiquette de a_i réfère à un argument de l'étiquette de h_i et $\langle h_i, n\rangle, \langle n, a_i\rangle \in \mathcal{P}_D^+$ pour $0 \leq i \leq k$.*

[4] $\mathcal{P}_D$ est la relation de parenté, i.e., $\langle x, y\rangle \in \mathcal{P}_D$ ssi y est nœud fils de x dans D.

[5] Une idée similaire est utilisée dans V-TAG, (Rambow, 1994), où, lors de l'analyse, on doit connaître l'ensemble des liens de dominance qui restent encore à satisfaire. En limitant cet ensemble, on obtient une analyse polynomiale.

3 De k-TT-MCTAG à RCG

Nous construisons une RCG simple équivalente à une k-TT-MCTAG de manière similaire à la conversion en RCG pour TAG. Nous considérons des prédicats $\langle\gamma\rangle$ référant aux contributions des arbres élémentaires (et non des ensembles) utilisés dans une analyse. Rappelons que chaque TT-MCTAG est une TAG contrainte, ainsi une dérivation TT-MCTAG est une dérivation pour la TAG sous-jacente. En conséquence, nous pouvons construire une RCG pour la TAG sous-jacente, tout en enrichissant les prédicats de façon à conserver l'information « doit être adjoint » pour les arbres arguments, information contraignant les clauses instantiables dans la dérivation RCG. Dans notre cas, la production d'un prédicat $\langle\gamma\rangle$ contient non seulement la production de γ et de ses arguments, mais également celle des arguments des prédicats précédemment instantiés dans la dérivation, et étant adjoints par la suite[6]. Notre conversion mène à une RCG d'arité 2, et dont les noms de prédicats sont complexes. Afin de maintenir le nombre de noms de prédicats nécessaires finis, la limite k est cruciale. Un prédicat $\langle\gamma\rangle$ doit encoder l'ensemble des arbres arguments dépendant d'arbres têtes précédemment consommés dans la dérivation. Cette ensemble est appelé *liste des arguments en attente* (*List of Pending Arguments, LPA*). Ces arbres doivent soit être adjoints à la racine de γ ou être passé à la LPA des arbres adjoints. Afin de réduire le nombre de clauses, nous distingons, comme le propose (Boullier, 1999b), les clauses d'arbres (prédicats $\langle\gamma...\rangle$), des clauses dites de « branchement » (prédicats $\langle adj\ldots\rangle$ and $\langle sub\ldots\rangle$). Nous avons ainsi trois types de prédicats :

1. $\langle\gamma, LPA\rangle$. Ces prédicats ont une arité 2 si γ est un arbre auxiliare (contributions gauche et droite du nœud pied), une arité 1 s'il s'agit d'un arbre initial. Les clauses $\langle\gamma, LPA\rangle$ distribuent les variables référant aux productions des arbres substitués ou adjoints à γ via une partie droite composée de prédicats adj et sub. La LPA est passée au prédicat adj du nœud racine, et les arguments de γ distribués à l'ensemble des prédicats d'adjonction.
2. $\langle adj, \gamma, dot, LPA\rangle$ est un prédicat de branchement d'arité 2. Ici, la LPA contient a) la liste des arguments des têtes précédemment consommées si $dot = \epsilon$ (*i.e.* nœud racine), et b) certains des arguments de γ (toutes les distributions des arguments sur les nœuds d'adjonction ont été calculées, en tenant compte des contraintes d'étiquette du nœud dot). Les clauses $\langle adj, \gamma, dot, LPA\rangle$ adjoignent un arbre γ' sur le nœud dot de γ. Si γ' était dans la LPA, le prédicat d'arbre appelé reçoit $LPA \setminus \{\gamma'\}$, dans le cas contraire, γ' est une tête, et la LPA reste inchangée.
3. $\langle sub, \gamma, dot\rangle$ est une clause de branchement d'arité 1, représentant les arbres pouvant être substitués au nœud dot de γ.

Plus précisément, la conversion se passe comme suit : comme dans (Boullier, 1999b), nous définissons une chaîne de décoration σ_γ pour chaque arbre élémentaire γ. Les nœuds internes qui ne sont pas étiquetés NA recoivent deux variables L et R représentant les productions provenant d'une adjonction au nœud en question, les nœuds feuilles une variable X représentant la production provenant d'une substitution. Dans un parcours de type *depth-first* de γ, les variables sont collectées (les nœuds pieds apportant un caractère de séparation « , »), pour créer le(s) argument(s) du prédicat $\langle\gamma\rangle$.

1. Nous ajoutons un prédicat de départ S et les clauses $S(X) \rightarrow \langle\alpha, \emptyset\rangle(X)$ pour l'ensemble des arbres initiaux α.

[6] Notons qu'une alternative à ces prédicats enrichis serait d'augmenter l'arité des prédicats pour contenir les contributions des arguments restant à adjoindre. La conversion en RCG devient alors bien plus complexe.

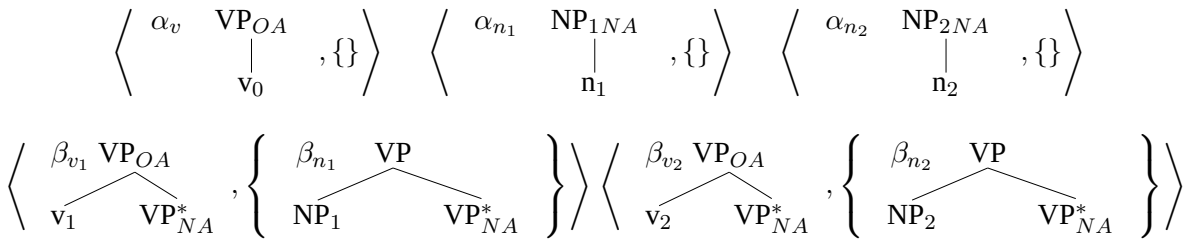


FIG. 4 – TT-MCTAG

2. Pour chaque $\gamma \in I \cup A$, soient L_p, R_p les symboles gauche et droite dans σ_γ pour le nœud d'adresse p, s'il s'agit d'un nœud d'adjonction, et X_p s'il s'agit d'un nœud de substitution. Soient $p_1, \ldots, p_k$ les nœuds d'adjonction potentiels, $p_{k+1}, \ldots, p_l$ les nœuds de substitution dans γ. La RCG produite contient alors toutes les clauses :
$\langle \gamma, LPA\rangle(\sigma_\gamma) \rightarrow \langle adj, \gamma, p_1, LPA_{p_1}\rangle(L_{p_1}, R_{p_1}) \ldots \langle adj, \gamma, p_k, LPA_{p_k}\rangle(L_{p_k}, R_{p_k})$
$\langle sub, \gamma, p_{k+1}\rangle(X_{p_{k+1}}) \ldots \langle sub, \gamma, p_l\rangle(X_{p_l})$ telles que :
   – si $LPA \neq \emptyset$, alors $\epsilon \in \{p_1, \ldots, p_k\}$ et $LPA \subseteq LPA_\epsilon$,
   – $\bigcup_{i=0}^{k} LPA_{p_i} = LPA \cup \Gamma(\gamma)$ où $\Gamma(\gamma)$ est soit l'ensemble des arguments de γ (si γ est un arbre tête), soit $\emptyset$ si γ est lui-même un argument.
3. Pour tous les prédicats $\langle adj, \gamma, dot, LPA\rangle$, la RCG contient les clauses $\langle adj, \gamma, dot, LPA\rangle(L, R) \rightarrow \langle \gamma', LPA'\rangle(L, R)$ telles que γ' peut être adjoint au nœud dot de γ et :
   – soit $\gamma' \in LPA$ et $LPA' = LPA \setminus \{\gamma'\}$,
   – soit $\gamma' \notin LPA$, γ' est un arbre tête, et $LPA' = LPA$.
4. Pour tous les prédicats $\langle adj, \gamma, dot, \emptyset\rangle$ où le nœud dot de γ n'est pas un nœud étiqueté OA, la RCG contient une clause $\langle adj, \gamma, dot, \emptyset\rangle(\epsilon, \epsilon) \rightarrow \epsilon$.
5. Pour tous les prédicats $\langle sub, \gamma, dot\rangle$ et tous les γ' qui peuvent être substitués au nœud dot de γ, la RCG contient la clause $\langle sub, \gamma, dot\rangle(X) \rightarrow \langle \gamma', \emptyset\rangle(X)$.

Prenons la TT-MCTAG de la Fig. 4. Pour celle-là, on obtient (entre autres) les clauses RCG

– $\langle \alpha_v, \emptyset\rangle(L\ v_0\ R) \rightarrow \langle adj, \alpha_v, \epsilon, \emptyset\rangle(L, R)$ (une seule adjonction sur la racine d'adresse ϵ)
– $\langle adj, \alpha_v, \epsilon, \emptyset\rangle(L, R) \rightarrow \langle \beta_{v_1}, \emptyset\rangle(L, R) \mid \langle \beta_{v_2}, \emptyset\rangle(L, R)$ (β_{v_1} ou β_{v_2} peuvent être adjoints à ϵ dans α_v, LPA (ici vide) est transmise)
– $\langle \beta_{v_1}, \emptyset\rangle(L\ v_1, R) \rightarrow \langle adj, \beta_{v_1}, \epsilon, \{\beta_{n_1}\}\rangle(L, R)$ (dans β_{v_1}, il y a un unique nœud d'adjonction d'adresse ϵ ; l'argument est passé à la nouvelle LPA)
– $\langle adj, \beta_{v_1}, \epsilon, \{\beta_{n_1}\}\rangle(L, R) \rightarrow \langle \beta_{n_1}, \emptyset\rangle(L, R) \mid \langle \beta_{v_1}, \{\beta_{n_1}\}\rangle(L, R) \mid \langle \beta_{v_2}, \{\beta_{n_1}\}\rangle(L, R)$ (soit β_{n_1} est adjoit et retiré de la LPA ou bien un autre arbre (β_{v_1} ou β_{v_2}) est adjoint)
– $\langle \beta_{v_1}, \{\beta_{n_1}\}\rangle(L\ v_1, R) \rightarrow \langle adj, \beta_{v_1}, \epsilon, \{\beta_{n_1}, \beta_{n_1}\}\rangle(L, R)$ (ici encore, il y a un seul nœud d'adjonction dans β_{v_1} ; l'argument β_{n_1} est ajouté à la LPA)
– $\langle \beta_{n_1}, \emptyset\rangle(L\ X, R) \rightarrow \langle adj, \beta_{n_1}, \epsilon, \emptyset\rangle(L, R)\ \langle sub, \beta_{n_1}, 1, \rangle(X)$ (adjonction à la racine et substitution au nœud 1 dans β_{n_1})
– $\langle adj, \beta_{n_1}, \epsilon, \emptyset\rangle(\epsilon, \epsilon) \rightarrow \epsilon$ (l'adjonction à la racine de β_{n_1} n'est pas obligatoire tant que la LPA est vide)
– $\langle sub, \beta_{n_1}, 1, \rangle(X) \rightarrow \langle \alpha_{n_1}, \emptyset\rangle(X)$ (substitution de α_{n_1} au nœud d'adresse 1)
– $\langle \alpha_{n_1}, \emptyset\rangle(n_1) \rightarrow \epsilon$ (aucune adjonction ou substitution dans α_{n_1})

La dérivation RCG pour la chaîne d'entrée $n_1 n_2 n_1 v_2 v_1 v_1 v_0$ procède comme suit[7] :

$S(n_1\ n_2\ n_1\ v_2\ v_1\ v_1\ v_0) \Rightarrow \langle \alpha_v, \emptyset\rangle(n_1\ n_2\ n_1\ v_2\ v_1\ v_1\ v_0)$

[7] Nous remplaçons les intervalles par les sous-chaînes correspondantes pour rendre l'exemple plus lisible.

$\Rightarrow \langle adj, \alpha_v, \epsilon, \emptyset\rangle(n_1\ n_2\ n_1\ v_2\ v_1\ v_1, \epsilon)$ (adjonction au nœud ϵ, v_0 est consommé)

$\Rightarrow \langle \beta_{v_1}, \emptyset\rangle(n_1\ n_2\ n_1\ v_2\ v_1\ v_1, \epsilon)$ (β_{v_1} est adjoint)

$\Rightarrow \langle adj, \beta_{v_1}, \epsilon, \{\beta_{n_1}\}\rangle(n_1\ n_2\ n_1\ v_2\ v_1, \epsilon)$ (adj. au nœud ϵ, v_1 est consommé, β_{n_1} est placé dans la LPA)

$\Rightarrow \langle \beta_{v_1}, \{\beta_{n_1}\}\rangle(n_1\ n_2\ n_1\ v_2\ v_1, \epsilon)$ (β_{v_1} est adjoint)

$\Rightarrow \langle adj, \beta_{v_1}, \epsilon, \{\beta_{n_1}, \beta_{n_1}\}\rangle(n_1\ n_2\ n_1\ v_2, \epsilon)$ (adj. au nœud ϵ, v_1 est consommé, β_{n_1} est placé dans la LPA)

$\Rightarrow \langle \beta_{v_2}, \{\beta_{n_1}, \beta_{n_1}\}\rangle(n_1\ n_2\ n_1\ v_2, \epsilon)$ (β_{v_2} est adjoint)

$\Rightarrow \langle adj, \beta_{v_2}, \epsilon, \{\beta_{n_2}, \beta_{n_1}, \beta_{n_1}\}\rangle(n_1\ n_2\ n_1, \epsilon)$ (adj. au nœud ϵ, v_2 est consommé, β_{n_2} est placé dans la LPA)

$\Rightarrow \langle \beta_{n_1}, \{\beta_{n_2}, \beta_{n_1}\}\rangle(n_1\ n_2\ n_1, \epsilon)$ (β_{n_1} issu de la LPA est adjoint)

$\Rightarrow \langle adj, \beta_{n_1}, \epsilon, \{\beta_{n_2}, \beta_{n_1}\}\rangle(n_1\ n_2, \epsilon)\ \ \langle sub, \beta_{n_1}, 1, \rangle(n_1)$ (adj. au nœud ϵ, subst. au nœud 1)

$\Rightarrow \langle adj, \beta_{n_1}, \epsilon, \{\beta_{n_2}, \beta_{n_1}\}\rangle(n_1\ n_2, \epsilon)\ \ \langle \alpha_{n_1}, \emptyset\rangle(n_1)$ (subst. de α_{n_1})

$\Rightarrow \langle adj, \beta_{n_1}, \epsilon, \{\beta_{n_2}, \beta_{n_1}\}\rangle(n_1\ n_2, \epsilon)\ \ \epsilon \Rightarrow \langle \beta_{n_2}, \{\beta_{n_1}\}\rangle(n_1\ n_2, \epsilon)$ (n_1 consommé, β_{n_2} adjoint)

$\Rightarrow \langle adj, \beta_{n_2}, \epsilon, \{\beta_{n_1}\}\rangle(n_1, \epsilon)\ \ \langle sub, \beta_{n_2}, 1, \rangle(n_2)$ (adj. au nœud ϵ, subst. au nœud 1)

$\Rightarrow \langle adj, \beta_{n_2}, \epsilon, \{\beta_{n_1}\}\rangle(n_1, \epsilon)\ \ \langle \alpha_{n_2}, \emptyset\rangle(n_2)$ (subst. de α_{n_2})

$\Rightarrow \langle adj, \beta_{n_2}, \epsilon, \{\beta_{n_1}\}\rangle(n_1, \epsilon)\ \ \epsilon \Rightarrow \langle \beta_{n_1}, \emptyset\rangle(n_1, \epsilon)$ (n_2 est consommé, β_{n_1} adjoint)

$\stackrel{*}{\Rightarrow} \langle adj, \beta_{n_1}, \epsilon, \emptyset\rangle(\epsilon, \epsilon)\ \ \langle \alpha_{n_1}, \emptyset\rangle(n_1) \stackrel{*}{\Rightarrow} \epsilon$ (subst. de α_{n_1}, n_1 consommé)

Cette exemple nécessite une LPA de capacité maximale 3, *i.e.* une 3-TT-MCTAG. Notons qu'avec cette construction, le groupement des arbres en tuples est perdu. Ainsi, dans notre exemple, nous ne savons plus quel arbre n_1 provient de quel ensemble. Cependant, en pratique, nous créons une RCG pour une sous-grammaire TT-MCTAG sélectionnée à partir de la chaîne d'entrée. Si plusieurs occurrences d'un tuple donné sont sélectionnées, celles-ci se voient affectées un identifiant unique.

Avec cette construction, on peut démontrer le théorème suivant (voir (Kallmeyer & Parmentier, 2008)). Comme corollaire, les k-TT-MCTAGs sont légèrement sensibles au contexte.

Théorème 1. *Pour chaque k-TT-MCTAG G, il existe une RCG G' simple telle que $L(G) = L(G')$*[8].

Conclusion

Cet article a mis en relation deux formalismes grammaticaux, à savoir les grammaires TT-MCTAG et grammaires RCG. TT-MCTAG est un formalisme de réécriture d'arbres qui permet de modéliser des phénomènes d'ordre des mots libre (existant par exemple en allemand). Les RCG, quant à elles, sont connues pour leur propriétés formelles avantageuses : les RCG sont analysables en un temps polynomial, et les RCG simples sont légèrement sensibles au contexte. De plus, des algorithmes d'analyse pour RCGs simples sont connus.

[8]Nous suspectons que l'inverse ne soit pas toujours vrai. Autrement dit, nous suspectons que les langages k-TT-MCTAG soient strictement contenus dans l'ensemble des langages RCG simples. Un exemple de langage qui n'est probablement pas dans k-TT-MCTAG est le langage de double copie $\{www \mid w \in \{a, b\}^*\}$. Si les trois copies d'un terminal appartiennent à un même tuple, deux d'entre elles doivent être adjointes par partage de nœud, comment contraindre alors toutes les dépendances pour n'avoir que celles attendues ?

Nous avons montré comment convertir une TT-MCTAG restreinte (appelée k-TT-MCTAG) en une RCG simple équivalente. Le résultat formel de cette conversion est que la classe des langages de chaînes générés par une k-TT-MCTAG est contenue dans la classe des langages générés par les RCGs simples. En particulier, k-TT-MCTAG est légèrement sensible au contexte.

Le résultat pratique de cette conversion réside dans l'implantation d'un analyseur syntaxique pour TT-MCTAG utilisant un analyseur RCG comme noyau. Plus précisément, la TT-MCTAG est d'abord convertie en une RCG équivalente, utilisée alors pour l'analyse. Le résultat de l'analyse RCG est alors converti en analyse TT-MCTAG par interprétation des noms de prédicats des clauses instanciées. L'analyseur en question est utilisé pour développer une grammaire TT-MCTAG de l'allemand, et est disponible librement sous licence GPL[9].

Références

BOULLIER P. (1998). *A Proposal for a Natural Language Processing Syntactic Backbone*. Rapport interne 3342, INRIA.

BOULLIER P. (1999a). *On TAG and Multicomponent TAG Parsing*. Rapport interne 3668, Institut National de Recherche en Informatique et en Automatique (INRIA).

BOULLIER P. (1999b). On TAG Parsing. In *TALN 99, 6^e conférence annuelle sur le Traitement Automatique des Langues Naturelles*, p. 75–84, Cargèse, Corse.

BOULLIER P. (2000). Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, p. 53–64, Trento, Italy.

JOSHI A. K. (1985). Tree adjoining grammars : How much contextsensitivity is required ro provide reasonable structural descriptions ? In D. DOWTY, L. KARTTUNEN & A. ZWICKY, Eds., *Natural Language Parsing*, p. 206–250. Cambridge University Press.

JOSHI A. K. & SCHABES Y. (1997). Tree-Adjoning Grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages*, p. 69–123. Berlin : Springer.

KALLMEYER L. (2005). Tree-local multicomponent tree adjoining grammars with shared nodes. *Computational Linguistics*, **31**(2), 187–225.

KALLMEYER L. & PARMENTIER Y. (2008). On the relation between Multicomponent Tree Adjoining Grammars with Tree Tuples (TT-MCTAG) and Range Concatenation Grammars (RCG). In *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications LATA*, Tarragona, Spain.

LICHTE T. (2007). An MCTAG with Tuples for Coherent Constructions in German. In *Proceedings of the 12th Conference on Formal Grammar 2007*, Dublin, Ireland.

RAMBOW O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

SØGAARD A., LICHTE T. & MAIER W. (2007). The complexity of linguistically motivated extensions of tree-adjoining grammar. In *Recent Advances in Natural Language Processing 2007*, Borovets, Bulgaria.

WEIR D. J. (1988). *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania.

[9] Voir `http://www.sfb441.uni-tuebingen.de/emmy/tulipa`