TALN 2007, Toulouse, 5–8 juin 2007

Architecture compositionnelle pour les dépendances croisées

Alexandre DIKOVSKY
LINA-FRE CNRS 2729, Université de Nantes
Alexandre.Dikovsky@univ-nantes.fr

Résumé. L'article présente les principes généraux sous-jacent aux grammaires catégorielles de dépendances : une classe de grammaires de types récemment proposée pour une description compositionnelle et uniforme des dépendances continues et discontinues. Ces grammaires très expressives et analysées en temps polynomial, adoptent naturellement l'architecture multimodale et expriment les dépendances croisées illimitées.

Abstract. This article presents the general principles underlying the categorial dependency grammars : a class of type logical grammars recently introduced as a compositional and uniform definition of continuous and discontinuous dependences. These grammars are very expressive, are parsed in a reasonable polynomial time, naturally adopt the multimodal architecture and explain unlimited cross-serial dependencies.

Mots-clés : grammaires catégorielles de dépendances, grammaires multimodales, analyseur syntaxique.

Keywords: categorial dependency grammars, multimodal grammars, syntactic parser.

1 Introduction

L'intérêt principal des grammaires de types logiques dont les grammaires catégorielles (GC) est leur lien direct et transparent avec la sémantique formelle compositionnelle. Ce lien est établi pour une phrase générée à travers l'isomorphisme entre une preuve de correction du choix des types pour les mots dans la phrase et l'expession sémantique extraite de cette preuve. Les relations syntaxiques entre les mots définies par les types sont formaliseés par un calcul logique de types qui n'est pas spécifique à une grammaire mais à une classe de grammaires. On construit ainsi des interfaces simples et élégantes entre la syntaxe et la sémantique à la base de principes plus ou moins universels. Tant que les relations entre les mots (*dépendances*) s'accordent bien avec les relations de précédence (*ordre des mots*), à savoir lorsqu'elles ne dépassent jamais les limites des domaines syntaxiques l ocaux des mots (*dépendances projectives*), les preuves de correction sont isomorphes aux systèmes de constituents des phrases. A ce niveau de représentation syntaxique il est en principe possible de définir les types directement en termes de dépendances. En fait, les premières définitions des grammaires de dépendances (GD) (Gaifman, 1961) ont été similaires à celle des GC classiques (Bar-Hillel, 1953) [1]. Cependant, il existe dans

[1] Or, même à ce niveau, on peut remarquer qu'à la différence des grammaires de types logiques, les GD traitent les modifieurs comme adjoints.

toute langue des dépendances *non bornées* par les domaines locaux (*dépendances non projectives*). Elles sont dues aux formes et aux mots fonctionnels discontinus (comme les particules négatives, les pronoms comparatifs, etc.), ou à l'interférence des éléments des structures extra-syntaxiques telles que la structure communicative (cf. la topicalisation), la co-référence, les relations de portée, etc. ou, au contraire, sont dues au manque en surface, des membres des relations sémantiques (comme c'est le cas de la relativisation ou de l'extraction au cours de la coordination). Pour y faire face les calculs logiques sont complétés par des règles qui, d'un côté, rendent les preuves plus flexibles au détriment du lien direct avec les constituents, e.g. en montant les types (Lambek, 1961; Steedman, 1996), et d'un autre côté, les rendent plus sélectives, e.g. en choisissant les règles structurelles spécifiques en foncion de connecteurs différents (règles *multimodales* dues à Oehrle, Morrill, Moortgat et Hepple (Morrill, 1994; Moortgat, 1997)). Avec ces moyens on peut exprimer les dépendances non bornées tout en gardant l'interprétation sémantique compositionnelle. En même temps, à cause de l'expressivité accrue, la complexité des preuves devient exponentielle, voire pire.

Dans l'article (Dikovsky, 2004), nous avons proposé une nouvelle architecture compositionnelle de types invariables de dépendances (sans montée des types). Elle est établie sur la base de la distinction faite entre les types *neutres* des dépendances projectives, qui sont formalisés par la règle classique d'élimination d'arguments, et les types des dépendances non bornées (*valences*) dotés de *polarisation* et d'*orientation*, qui sont formalisés par une règle appelée **FA** (*first available*) de saturation (*appariement*) des valences. La base psycholinguistique de cette règle est l'hypothèse que les dépendances non bornées sont gérées par les piles dans la mémoire dynamique d'analyse. **FA** sélectionne la plus proche valence polarisée duale dans la direction indiquée. Elle est conforme avec la majorité des dépendances non projectives dans maintes langues. On a élaboré différents calculs de dépendances avec la règle **FA** (Dekhtyar & Dikovsky, 2004; Dekhtyar & Dikovsky, 2007). Les *Grammaires Catégorielles de Dépendances* (CDG) correspondantes s'avèrent expressives. En même temps, elles disposent d'algorithmes d'analyse en temps polynomial. Tout de même, la règle **FA** n'est pas universelle. Par exemple, elle n'est pas adaptée aux dépendances croisées illimitées du hollandais exposées dans (Bresnan *et al.*, 1982). C'est pourquoi, dans cet article nous explorons une autre règle d'appariement **FC** (*first cross*) qui sélectionne la première valence polarisée duale croisée dans la direction indiquée. Ainsi, la structure dynamique de mémoire qui correspond à cette règle est la file d'attente. **FC** explique les dépendances croisées illimitées en termes d'un langage simple de *structures de dépendances* et non en termes du langage de copies, comme d'habitude. A l'instar de grammaires multimodales de types, nous définissons les *CDG multimodales* (mmCDG) où les règles d'appariement sont considérées comme les modes de compositionnalité propres aux dépendances non projectives. Nous montrons que la règle **FC** est aussi efficace que la règle **FA** et nous présentons un algorithme d'analyse syntaxique de ces grammaires en temps polynomial.

2 Grammaires catégorielles de dépendances

Les CDG sont des *grammaires catégorielles* (GC) qui, à la différence des GC classiques, définissent explicitement les relations de dépendance entre les mots dans la phrase et non les relations de dominance entre les constituants. Elles peuvent déterminer les *structures de dépendances* (SD) plus générales que les *arbres de dépendances* (AD). Une SD d'une phrase $w = w_1 \dots w_n$ est un graphe orienté dont les nœux sont les mots $w_1, \dots, w_n$ ordonnés par l'ordre dans w, avec un nœux sélectionné (la *tête*) et dont les arcs sont étiquetés par les noms

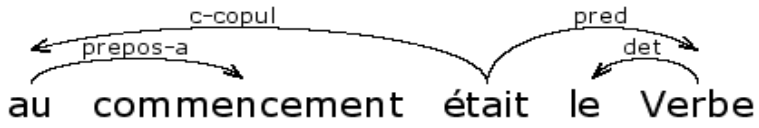


Figure 1

des *dépendances*. E.g., la SD en figure 1 est un AD dont la tête (sa racine) est le mot *était*. Comme toutes les GC, les CDG n'ont pas de règles. Une CDG peut être vue comme un *lexique* qui affecte à chaque mot un ensemble de *types de dépendances*. La particularité essentielle des types des CDG est la distinction faite entre les types de *dépendances projectives* qui relient le gouverneur à ses subordonnés appartenants à son domaine local, et les types de *dépendances non projectives* (*non bornées*) qui le relient aux subordonnés déplacés vers les domaines des autres mots. Les premiers sont définis par les sous types arguments des types du gouverneur, tandis que les derniers sont définis par les *valences* dotées d'une polarisation et d'une orientation (gauche / droite) dont l'ensemble constitue pour chaque type son *potentiel*. Formellement, les types de dépendances sont construits à partir d'un ensemble $\mathbf{C}$ de *types primitifs* et d'un ensemble $V(\mathbf{C})$ de *valences polarisées*. Les éléments de $\mathbf{C}$ sont les noms des relations de dépendance, dont un type sélectionné S (*l'axiome*). Les valences dans $V(\mathbf{C})$ sont orientées : $V(\mathbf{C}) = V^l(\mathbf{C}) \cup V^r(\mathbf{C})$, où $V^l(\mathbf{C})$ consiste des *valences gauches* $\swarrow d$ (*négative*), $\nearrow d$ (*positive*) et $V^r(\mathbf{C})$ consiste des *valences droites* $\nwarrow d$ (positive), $\searrow d$ (négative) où $d \in \mathbf{C}$.
Un *type (de dépendance)* est une expression α^P, où α est un type *basique* et P est un *potentiel*. $gCat(\mathbf{C})$ va noter l'ensemble des types sur $\mathbf{C}$. Les *types basiques* $\mathbf{B}(\mathbf{C})$ sur $\mathbf{C}$ sont les types fonctionnels traditionnels du 1^r ordre destinés à définir les dépendances projectives :

1. $\mathbf{C} \subset \mathbf{B}(\mathbf{C})$. 2. Si $\alpha \in \mathbf{C}$ et $\beta \in \mathbf{B}(\mathbf{C})$, alors $[\alpha\backslash\beta]$, $[\alpha* \backslash\beta]$, $[\beta/\alpha*]$, $[\beta/\alpha] \in \mathbf{B}(\mathbf{C})$. □

Les constructeurs $\backslash, /$ étant supposés associatifs, tout type basique peut être représenté sous la forme $[a_{lm}\backslash ... \backslash a_{l1}\backslash f/a_{r1}/.../a_{rn}]$. Intuitivement, f est la dépendance du gouverneur et a_{li}, a_{rj} correspondent aux dépendances des subordonnés gauches et droites. $d*$ correspond à la dépendance d itérée. $f = S$ est le type des SD correctes. Les *potentiels* sont les suites de valences polarisées. Ils sont destinés à définir les dépendances non projectives. Dans le cas de dépendances projectives, ils sont vides. Les types avec le potentiel vide sont *neutres*. Par exemple, l'AD projectif en figure 1 est défini par les types neutres suivants :

au $\mapsto [c{-}copul/prepos{-}a]$ *commencement* $\mapsto [prepos{-}a]$ *le* $\mapsto [det]$
était $\mapsto [c{-}copul\backslash S/pred]$ *Verbe* $\mapsto [det\backslash pred]$

Les valences $\swarrow d$ et $\nearrow d$, $d \in \mathbf{C}$, peuvent être vues comme les *parenthèses gauches*. Respectivement, $\nwarrow d$ et $\searrow d$ sont les *parenthèses droites*. Pour une valence gauche, e.g. $\swarrow d$, la valence correspondante (*duale*) droite, $\nwarrow d$, est notée $\breve{\swarrow} d = \nwarrow d$. Ensemble ces valences duales appariées définissent la dépendance non projective d. L'adjacence est exprimée en utilisant les types primitifs *d'ancrage* : pour *ancrer* une valence négative $v \in \{\swarrow d, \searrow d \,|\, d \in \mathbf{C}\}$ (la fin d'une dépendance non projective), c'est-à-dire la placer auprès d'un *mot d'appui*, sont utilisés les types primitifs particuliers *d'ancrage* : $\#(v)$ dont l'élimination signifie l'adjacence des mots et ne crée aucune dépendance. E.g., l'AD non projectif en figure 2 est défini par

Figure 2

les types qui ancrent les clitiques *la*, *lui* sur l'auxiliaire *a* :

elle $\mapsto [pred]$ $\quad$ *a* $\mapsto [\#(\swarrow clit-iobj)\backslash\#(\swarrow clit-dobj)\backslash pred\backslash S/aux]$
la $\mapsto [\#(\swarrow clit-dobj)]^{\swarrow clit\text{–}dobj}$ $\quad$ *lui* $\mapsto [\#(\swarrow clit-iobj)]^{\swarrow clit\text{–}iobj}$
donnée $\mapsto [aux]^{\nwarrow clit\text{–}iobj \nwarrow clit\text{–}dobj}$

Le sens exact des types est défini par le *calcul de dépendances* suivant [2] :

$\mathbf{L^l}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$
$\mathbf{I^l}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1P_2}$
$\mathbf{\Omega^l}$. $[C^*\backslash\beta]^{P} \vdash [\beta]^{P}$
$\mathbf{D^l_M}$. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2}$, si $P_1(\swarrow C)P(\nwarrow C)P_2$ satisfait la règle d'appariement **M**.

$\mathbf{L^l}$ est la règle classique d'élimination. En éliminant le sous-type argument $C \neq \#(\alpha)$, elle crée la dépendance projective C et concatène les potentiels. $C = \#(\alpha)$ ne crée aucune dépendance. $\mathbf{I^l}$ crée $k > 0$ exemplaires de C. $\mathbf{\Omega^l}$ sert pour le cas $k = 0$ et pour éliminer le sous-type itéré. $\mathbf{D^l_M}$ apparie et élimine deux valences duales $\swarrow C$ et $\nwarrow C$ selon la règle d'appariement **M** et crée la dépendance non projective C. Voici deux règles importantes d'appariement :

$\mathbf{FA}^l$: P n'a pas d'occurrence de $\swarrow C, \nwarrow C$ (apparier à la plus proche valence duale disponible).

$\mathbf{FC}^l$: P_1 et P n'ont pas d'occurrences, respectivement, de $\swarrow C$ et de $\nwarrow C$ (apparier à la premiere valence duale croisée, c'est-à-dire à la plus lointaine disponible).

On voit que les valences ressamblent aux traits Slash des GPSG, HPSG, mais à la place de règles complexes de « propagation »des traits Slash les CDG utilisent les règles simples d'appariement **FA** et **FC**. En admettant que toute dépendance non projective C peut avoir sa propre règle d'appariement M_C nous considérons cette règle comme un mode de compositionnalité à travers C. Nous obtenons ainsi par analogie avec l'architecture multimodale pour les grammaires de Lambek (Morrill, 1994; Moortgat, 1997) la notion suivante de grammaire.

Définition 1 *Une* grammaire catégorielle multimodale de dépendances *(mmCDG) est une structure* $G = (W, \mathbf{C}, S, \delta, \mu)$, *où* W *est un vocabulaire,* δ *(le* lexique*) est une fonction qui affecte à chaque mot dans* W *un sous ensemble fini de types dans gCat(***C***) et* μ *est une fonction qui affecte une règle d'appariement à toute dépendance non projective dans* $\mathbf{C}$.
Le calcul de dépendances détermine la relation de prouvabilité correspondante $\vdash_\mu$ *sur les suites de types. La prouvabilité sans règles* **D** *(c'est-à-dire, au cas de dépendances projectives) est notée* $\vdash_{\mathbf{c}}$ *. Pour une SD* D *et une phrase* w, *la relation* $G(D, w)$ *signifie : « D est créée au cours d'une preuve* $\Gamma \vdash_\mu S$ *pour une suite de types* $\Gamma \in \delta(w)$ *».*
Le langage *et le* langage des SD *générés par* G *sont respectivement les ensembles* $L(G) =_{df} \{w \mid \exists D\ G(D, w)\}$ *et* $\Delta(G) =_{df} \{D \mid \exists w\ G(D, w)\}$. $mmCDG^\mu$ *et* $\mathcal{L}(mmCDG^\mu)$ *sont respectivement la famille des grammaires et des langages correspondants.*

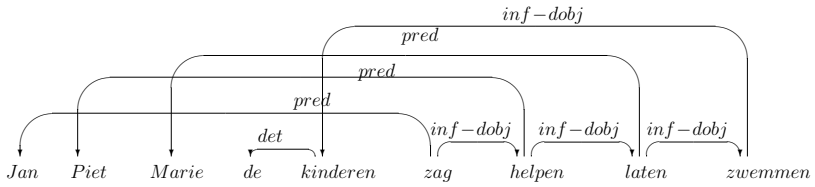
3 Expressivité des mmCDG

Les mmCDG sont très expressives. Avec la règle **FA** elles génèrent tous les langages non contextuels (algébriques), mais aussi maints langages contextuels dont $\{a^nb^nc^n \mid n > 0\}$, les langages $L^{(m)} = \{a_1^n a_2^n ... a_m^n \parallel n \geq 1\}$ (Dikovsky, 2004) qui sont faiblement contextuels mais non-TAG à partir de $m > 4$, le langage MIX, qui contient toutes permutations des motifs $a^nb^nc^n, n > 0$, $MIX = \{w \in \{a, b, c\}^+ \parallel |w|_a = |w|_b = |w|_c\}$. Or, selon l'hypothèse de E.

[2] Nous exposons les règles gauches. Les règles droites sont symétriques.

Bach, MIX n'est pas faiblement contextuel, ainsi il ne serait pas généré par une grammaire minimaliste, ou multi-TAG, etc. Dans (Dekhtyar & Dikovsky, 2007) on peut trouver d'autres exemples et une preuve du fait que $\mathcal{L}(mmCDG^{\mathbf{FA}})$ est une famille abstraite de langages (AFL).

D'un autre côté, nous croyons (Dikovsky, 2004; Dekhtyar & Dikovsky, 2004) que le langage de copies $L_{copy} = \{ww \mid w \in \{a,b\}^+\}$, qui est généré par une grammaire TAG, n'appartient pas à la famille $\mathcal{L}(mmCDG^{\mathbf{FA},\mathbf{FC}})$. Ce langage est d'un intérêt particulier parce qu'on croit qu'il est un modèle de la construction en néerlandais dite des « *dépendances croisées illimitées* ». Il s'agit des phrases $n_1 n_2 \dots n_m n_{m+1} v_1 v_{(inf)2} \dots v_{(inf)m}$, dont un exemple est en figure 3, où il y a une dépendance prédicative $n_1 \overset{pred}{\longleftarrow} v_1$ entre le verbe v_1 en forme finie et le nom n_1, les dépendances prédicatives $n_i \overset{pred}{\longleftarrow} v_{(inf)i}$ entre les verbes $v_{(inf)i}$ à l'infinitif et les noms n_i, pour tout $2 \leq i \leq m$, et éventuellement, une dépendance d'objet direct $n_{m+1} \overset{dobj}{\longleftarrow} v_{(inf)m}$ si le verbe $v_{(inf)m}$ est transitif et le nom n_{m+1} est présent (c'est-à-dire, $n_{m+1} \neq \varepsilon$).



**Jan Piet Marie les enfants a vu aider faire nager*

Figure 3.

Par ailleurs, une analyse plus approfondie de cette construction (Pulman & Ritchie, 1985) montre que l'accord des formes existe seulement entre n_1 et v_1. Sinon, la forme du nom subordonné est déterminée seulement par le verbe transitif $v_{(inf)m}$ et son argument n_{m+1}. Cela implique que le vrai modèle de cette construction n'est point le langage L_{copy}, mais le langage des SD $\Delta_{cross} = \{D^{(m)} \mid m > 0\}$ sur $W = N \cup V$, où $N \cap V = \emptyset$, $D^{(m)}$ est la SD en figure 4 et $n_{i_l} \in N, v_{j_r} \in V$. En même temps, le *langage* correspondant est algébrique (voire linéaire).

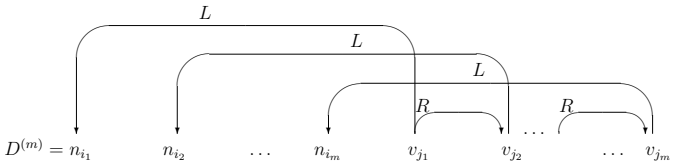


Figure 4. AD $D^{(m)}$

Le langage Δ_{cross} est généré par la $mmCDG^{\mathbf{FC}}$ suivante :

$$G_{cross} = \left\{ \begin{array}{lcll} n & \mapsto & [\#(L)]^{\swarrow_L}, [\#(L)\backslash\#(L)]^{\swarrow_L}, & \text{pour } n \in N \\ v & \mapsto & [\#(L)\backslash S/R]^{\searrow_L}, [R/R]^{\searrow_L}, [R]^{\searrow_L}, & \text{pour } v \in V \end{array} \right.$$

E.g., une preuve de $D^{(3)} \in \Delta_{cross}$ est montrée en figure 5.

$$\dfrac{\dfrac{\dfrac{[\#(L)]^{\swarrow_L}[\#(L)\backslash\#(L)]^{\swarrow_L}}{[\#(L)]^{\swarrow_L\swarrow_L}}(\mathbf{L}^l) \quad [\#(L)\backslash\#(L)]^{\swarrow_L}}{[\#(L)]^{\swarrow_L\swarrow_L\swarrow_L}}(\mathbf{L}^l) \quad \dfrac{[\#(L)\backslash S/R]^{\searrow_L} \quad \dfrac{[R/R]^{\searrow_L}[R]^{\searrow_L}}{[R]^{\searrow_L\searrow_L}}(\mathbf{L}^r)}{[\#(L)\backslash S]^{\searrow_L\searrow_L\searrow_L}}(\mathbf{L}^r)}{[S]^{\swarrow_L\swarrow_L\swarrow_L\searrow_L\searrow_L\searrow_L}}(\mathbf{L}^l)}{[S]}(\mathbf{D}^l_{\mathbf{FC}} \times 3)$$

Figure 5.

4 Fondements théoriques

Notre solution du problème des dépendances croisées repose sur l'indépendance des types basiques et des valences polarisées dans les preuves du calcul de dépendances. Cette propriété est exprimée en termes de *projections* et de suites de catégories *bien appariées*.

Pour une suite de catégories $\gamma \in gCat(\mathbf{C})^*$ ses projections *locale* $\|\gamma\|_l$ et *de valences* $\|\gamma\|_v$ sont définies ainsi : pour tous $\alpha \in gCat(\mathbf{C})$, $\gamma \in gCat(\mathbf{C})^*$ et $C^P \in gCAT(\mathbf{C})$,

1. $\|\varepsilon\|_l = \|\varepsilon\|_v = \varepsilon$; $\|\alpha\gamma\|_l = \|\alpha\|_l\|\gamma\|_l$ et $\|\alpha\gamma\|_v = \|\alpha\|_v\|\gamma\|_v$
2. $\|C^P\|_l = C$ et $\|C^P\|_v = P$.

Pour un potentiel P, sa projection $\|P\|_d$ sur une paire de valences duales $vd, \breve{vd}$ est définie comme $h(P)$ pour l'homomorphisme $h(\alpha) = \alpha$ si $\alpha \in \{vd, \breve{vd}\}$ et $h(\alpha) = \varepsilon$ sinon. P est dit *équilibré* si toute projection $\|P\|_d$ est bien appariée au sens habituel.

Soit $|P|_x$ le nombre d'occurrences de x dans P. Alors l'équilibre d'un potentiel P est incrémentalement vérifiable en utilisant les quantités suivantes pour toute $\alpha \in V^l(\mathbf{C})$ et $\breve{\alpha} \in V^r(\mathbf{C})$:

$$\Delta_\alpha(P) = max\{|P'|_\alpha - |P'|_{\breve{\alpha}} \parallel P' \; est \; un \; \text{suffixe} \; de \; P\},$$
$$\Delta_{\breve{\alpha}}(P) = max\{|P'|_{\breve{\alpha}} - |P'|_\alpha \parallel P' \; est \; un \; \text{préfixe} \; de \; P\}.$$

Elles expriment respectivement le *déficit* des $\alpha-$parenthèses droites et gauches dans P (c'est-à-dire, le nombre de parenthèses droites (gauches) qu'il faut rajouter à P de droite (de gauche) pour qu'il devienne équilibré. Les propriétés suivantes sont vérifiées (Dekhtyar & Dikovsky, 2004; Dekhtyar & Dikovsky, 2007) :

Lemme 1 *1. Quels que soient des potentiels P_1, P_2 et des valences $\alpha \in V^l(\mathbf{C})$, $\breve{\alpha} \in V^r(\mathbf{C})$,*

$$\Delta_\alpha(P_1P_2) = \Delta_\alpha(P_2) + max\{\Delta_\alpha(P_1) - \Delta_{\breve{\alpha}}(P_2), 0\},$$
$$\Delta_{\breve{\alpha}}(P_1P_2) = \Delta_{\breve{\alpha}}(P_1) + max\{\Delta_{\breve{\alpha}}(P_2) - \Delta_\alpha(P_1), 0\}.$$

2. Un potentiel P est équilibré ssi $\sum_{\alpha \in V(\mathbf{C})} \Delta_\alpha(P) = 0$.

La propriété suivante d'*indépendance des projections* (Dekhtyar & Dikovsky, 2004; Dekhtyar & Dikovsky, 2007) garantit l'existence d'un algorithme polynomial d'analyse de $mmCDG^{\mathbf{FA}}$.

Théorème 1 *Pour une mmCDG $G = (W, \mathbf{C}, S, \delta, \mu)$ avec le mode* **FA** *et $x \in W^+$, $x \in L(G)$ ssi il y a une suite $\Gamma \in \delta(x)$ telle que $\|\Gamma\|_l \vdash_{\mathbf{c}} S$ et $\|\Gamma\|_v$ est équilibré.*

Le seul point de sa preuve sensible aux modes est la proposition suivante vraie pour **FA** :

Lemme 2 *Un potentiel P est équilibré ssi pour toute catégorie α^P il y a une preuve $\alpha^P \vdash \alpha$ utilisant exclusivement les règles $\mathbf{D}^l_{\mathbf{M}}$ et $\mathbf{D}^r_{\mathbf{M}}$.*

Pour garantir l'indépendance des projections (et par conséquent, une analyse polynomiale) pour une $mmCDG^M$, il faut prouver ce lemme pour tout mode $\mathbf{M} \in M$. En prouvant le lemme 2 pour $\mathbf{FC}$, nous avons étendu le théorème 1 aux mmCDG avec les modes $\mathbf{FA}, \mathbf{FC}$:

Théorème 2 *Pour $x \in W^+$ et pour une $mmCDG^M$ $G = (W, \mathbf{C}, S, \delta, \mu)$ avec $M = \{\mathbf{FA}\}$, ou $M = \{\mathbf{FC}\}$ ou $M = \{\mathbf{FA}, \mathbf{FC}\}$, $x \in L(G)$ ssi il y a une suite $\Gamma \in \delta(x)$ telle que $\|\Gamma\|_l \vdash_{\mathbf{c}} S$ et $\|\Gamma\|_v$ est équilibré.*

Corollaire 1 $\mathcal{L}(mmCDG^{\mathbf{FA}}) = \mathcal{L}(mmCDG^{\mathbf{FC}}) = \mathcal{L}(mmCDG^{\mathbf{FA},\mathbf{FC}})$.

5 Analyse syntaxique, complexité

Dans l'article (Dekhtyar & Dikovsky, 2004) un algorithme d'analyse en temps polynomial a été décrit pour une version sous commutative du calcul de dépendances [3]. Dans l'article (Dekhtyar & Dikovsky, 2007) cet algorithme a été étendu aux $mmCDG^{\mathbf{FA}}$. Ce même algorithme à un détail près s'applique aussi aux $mmCDG^{\mathbf{FA},\mathbf{FC}}$. Nous l'exposons en figure 6.

Fonctions d'échec. Soit une $mmCDG^M$ $G = (W, \mathbf{C}, S, \delta, \mu)$ avec les valences polarisées gauches $V^l(\mathbf{C}) = \{v_1, \ldots, v_p\}$ et droites $V^r(\mathbf{C}) = \{\breve{v}_1, \ldots, \breve{v}_p\}$. Nous allons d'abord définir deux fonctions d'échec qui vont servir pour une optimisation de l'analyse. Soit $w = w_1 w_2 ... w_n \in W^+$. Alors, pour $1 \leq i \leq n$, $\alpha \in V^l(\mathbf{C})$ et $\beta \in V^r(\mathbf{C})$,

$$\pi^L(\alpha, i) = max\{\Delta_\alpha(\|\Gamma\|_v) \mid \Gamma \in \delta(w_1 ... w_i)\},$$
$$\pi^R(\beta, i) = max\{\Delta_\beta(\|\Gamma\|_v) \mid \Gamma \in \delta(w_{n-i+1} ... w_n)\}$$

sont les *fonction d'échec* gauche et droite. On suppose que $\pi^L(\alpha, 0) = \pi^R(\beta, 0) = 0$.

Algorithme d'analyse syntaxique. mmCdgPars est un algorithme typique de « programmation dynamique ». Il s'applique à une $mmCDG^M$ et à une phrase $w = w_1 w_2 ... w_n \in W^+$ et remplit une matrice triangulaire M dont la dimension est $n \times n$. L'élément $M[i, j]$, $i \leq j$, de M correspond à l'intervalle $w_i ... w_j$ de la phrase et représente un ensemble fini d'« items ». Un *item* est une expression $I = \langle C, \Delta^L, \Delta^R, I^l, I^r \rangle$ qui code une catégorie C^P, où C est une catégorie basique ($C \in \mathbf{B}(\mathbf{C})$), $\Delta^L = (\Delta_{v_1}, \ldots, \Delta_{v_p})$ et $\Delta^R = (\Delta_{\breve{v}_1}, \ldots, \Delta_{\breve{v}_p})$ sont les vecteurs entiers dont chaque composante i correspond à la valence v_i, respectivement $\breve{v}_i$, et vaut le déficit correspondant des v_i-parenthèses droites (gauches) dans le potentiel P. Finalement, I^l, I^r sont les identificateurs des items dans les angles gauches et droites de M à partir desquelles est calculé l'item I (pour tout $I \in M[i, i]$ $I^l = I^r = \emptyset$).

Complexité. Pour une $mmCDG^M$ $G = (W, \mathbf{C}, S, \delta, \mu)$, soit $l_G = |\delta|$ le nombre d'affectations des catégories aux mots dans le lexique, soit $a_G = max\{k \mid \exists x \in W\ ([\alpha_k \backslash ... \backslash \alpha_1 \backslash C / \beta]^P \in \delta(x) \ \vee\ [\beta \backslash C / \alpha_1 / ... / \alpha_k]^P \in \delta(x))\}$ le nombre maximal de sous types arguments dans les catégories affectées, soit $p_G = |V^l(\mathbf{C})| = |V^r(\mathbf{C})|$ le nombre de valences polarisées et $\Delta_G = max\{\Delta_\alpha(P) \mid \exists x \in W(C^P \in \delta(x) \vee \alpha \in V(\mathbf{C}))\}$ le déficit maximal des valences parenthèses dans les catégories affectées. Finalement, soit n la longueur de la phrase analysée.

Théorème 3 *L'algorithme* $\mathbf{mmCdgPars}$ *a une complexité en temps* $\mathbf{O}(\mathsf{l_G} \cdot \mathsf{a_G^2} \cdot (\Delta_\mathsf{G} \cdot \mathsf{n})^{2\mathsf{p_G}} \cdot \mathsf{n}^3)$.

Remarque 1 *1. Pour une grammaire fixée G, les valeurs l_G, a_G, p_G et Δ_G sont constantes. Si G varie, alors le problème d'appartenance devient NP-complèt (Dekhtyar & Dikovsky, 2004).*
2. Si G est sans valence polarisée, alors la complexité est $\mathbf{O}(\mathsf{n}^3)$.
3. Soit le déficit maximal de valences $\sigma_G(n)$ des potentiels survenants dans les preuves des phrases dont la longueur est limitée par n. Si $\sigma_G(n)$ est bornée par une constante c, alors G peut être transformée en une mmCDG G' sans valence polarisée dont le langage est algébrique (Dikovsky, 2001). Or, la taille de G' est exponentielle par rapport à G. Si, de plus, le nombre des dépendances non bornées dans une SD engendrée par G n'est jamais supérieur à une borne constante uniforme (ce qui est typique pour maintes langues), alors la complexité est $\mathbf{O}(\mathsf{n}^3)$ pour la même grammaire G.
4. D'un autre côté, même si **toute** *dépendance de G (sauf S) était définie par une valence polarisée, la complexité serait toujours polynomiale. Cette remarque explique que les mmCDG sont bien adaptées aux langages avec l'ordre flexible. Les limites de cet article ne nous laissent pas faire une analyse plus détaillée de ce cas important.*

[3] L'algorithme a été réalisé en LISP par Darin et Hristian Todorov et en en $C\#$ par Ilya Zaytsev.

Algorithme $\mathbf{mmCdgPars}$
//Entrée : mmCDG G, phrase $w = w_1...w_n$
//Sortie : $\langle$"$succès$", DS D$\rangle$ ssi $w \in L(G)$
{
CalcFailFuncL() ;
CalcFailFuncR() ;
for $(k = 1, \ldots, n)$
{
Propose(k)
}
for $(l = 2, \ldots, n)$
{
for $(i = 1, \ldots, n - l)$
{
$j := i + l - 1$;
for $(k = i, \ldots, j - l)$
{
SubordinateL(i, k, j) ;
SubordinateR(i, k, j) ;
}
}
}
if ($I = \langle S, (0, 0, \ldots, 0), (0, 0, \ldots, 0), I^l, I^r\rangle \in M[1, n]$)
return $\langle$"$succès$", $Expand(I)\rangle$;
//procedure Expand(I) *calcule la SD de sortie.*
//Elle seule est sensible aux règles d'appariement
*//***FA**,**FC**. *Elle est technique et n'est pas incluse*
else
return $\langle$"$échec$", $\emptyset\rangle$;
}

CalcFailFuncL()
{
foreach $(v \in V^l(\mathbf{C}))$
{
$\pi^L[v, 0] := 0$;
for $(i = 1, \ldots, n)$
{
$\pi_{max} := 0$;
foreach $(C^P \in \delta(w_i))$
{
$\pi_{max} := max\{\pi_{max}, \Delta_v(P) +$
$max\{\pi^L[v, i - 1] - \Delta_{\breve{v}}(P), 0\}\}$;
}
$\pi^L[v, i] := \pi_{max}$;
}
}
}

CalcFailFuncR() est similaire.

//For $1 \leq i \leq n$
Propose(i)
{
(loop) foreach $(C^P \in \delta(w_i)$
{
foreach $(v \in V^l(\mathbf{C}))$
{
$\Delta^L[v] := \Delta_v(P)$;
if ($\Delta^L[v] > \pi^R[\breve{v}, n - j]$) next (loop) ;
$\Delta^R[\breve{v}] := \Delta_{\breve{v}}(P)$;
if ($\Delta^R[\breve{v}] > \pi^L[v, i - 1]$) next (loop) ;
}
AddItem($M[i, i], \langle C, \Delta^L, \Delta^R, \emptyset, \emptyset\rangle$) ;
}
}

AddItem($M[i, j], \langle C, \Delta^L, \Delta^R, I^l, I^r\rangle$)
{
$M[i, j] := M[i, j] \cup \{\langle C, \Delta^L, \Delta^R, I^l, I^r\rangle\}$;
if ($C = [C' * \backslash\beta]$)
{
AddItem($M[i, j], \langle[\beta], \Delta^L, \Delta^R, I^l, I^r\rangle$) ;
}
if ($C = [\beta/C'*]$)
{
AddItem($M[i, j], \langle[\beta], \Delta^L, \Delta^R, I^l, I^r\rangle$) ;
}
}

//For $1 \leq i \leq k \leq j \leq n$
SubordinateL(i, k, j)
{
(loop) foreach $(I_1 = \langle \alpha_1, \Delta_1^L, \Delta_1^R, I_1^l, I_1^r\rangle \in M[i, k]$,
$I_2 = \langle \alpha_2, \Delta_2^L, \Delta_2^R, I_2^l, I_2^r\rangle \in M[k + 1, j]$)
{
foreach $(v \in V^l(\mathbf{C}))$
{
$\Delta^L[v] := \Delta_2^L(v) + max\{\Delta_1^L(v) - \Delta_2^R(v), 0\}$;
if ($\Delta^L[v] > \pi^R[\breve{v}, n - j]$) next (loop) ;
$\Delta^R[\breve{v}] := \Delta_1^R(\breve{v}) + max\{\Delta_2^R(\breve{v}) - \Delta_1^L(\breve{v}), 0\}$;
if ($\Delta^R[\breve{v}] > \pi^L[v, i - 1]$) next (loop) ;
}
if ($\alpha_1 = C$ and $\alpha_2 = [C\backslash\beta]$)
{
AddItem($M[i, j], \langle[\beta], \Delta^L, \Delta^R, I_1, I_2\rangle$) ;
}
elseif (($\alpha_1 = C$ and $\alpha_2 = [C * \backslash\beta]$) or $\alpha_1 = [\varepsilon]$)
{
AddItem($M[i, j], \langle \alpha_2, \Delta^L, \Delta^R, I_1, I_2\rangle$) ;
}
}
}

SubordinateR(i, k, j) est similaire.

Figure 6. Algorithme $\mathbf{mmCdgPars}$

6 Comparaison, discussion

Certes, il y a des grammaires où l'expression des dépendances non bornées ne pose pas problème, e.g. HPSG (Pollard & Sag, 1988), les extensions multimodales des grammaires de Lambek (Morrill, 1994; Moortgat, 1997), dont certaines visent notamment les dépendances (Kruijff, 2001) et leur fournissent une interface compositionnelle avec la sémantique. Or, l'analyse avec ces formalismes expressifs est très complexe et parfois nécessite l'utilisation des systèmes de démonstration des théorèmes. C'est aussi le cas des grammaires qui représentent $PTIME$, dont RCG (Boullier, 2003). A la différence de mmCDG, ces grammaires n'ont pas d'algorithme universel d'analyse en temps $\mathbf{O}(\mathsf{n}^{\mathsf{k}})$, où k dépend de l'alphabet. Cela concerne aussi les grammaires basées sur l'unification et les contraintes, e.g. (Duchier, 1999). Contrairement à ces formalismes, les mmCDG n'utilisent que les moyens primitifs d'une complexité faible. E.g., les Grammaires Topologiques de Dépendances (Duchier & Debusmann, 2001) (voir aussi (Bröker, 1998; Duchier *et al.*, 2004)) utilisent les hierarchies des domaines de l'ordre des mots (WO-domains) qui, en cas de discontinuité, servent à exprimer les contraintes de contiguïté, de distance entre un gouverneur et son modifieur etc. Dans beaucoup des cas, ces contraintes sont exprimées dans mmCDG par le moyen de sous types d'ancrage placés dans les positions correspondantes d'un type du gouverneur.

Les mmCDG représentent une alternative intéressante aux TAG (et équivalentes : CCG, HG (Vijay-Shanker & Weir, 1994)) et aux grammaires faiblement contextuelles (Joshi *et al.*, 1991), telles multi-TAG, non contextuelles multi-composantes, minimalistes, etc. Tout comme ces dernières, les mmCDG disposent d'une analyse syntaxique en temps polynomial. On peut même constater, qu'en pratique l'algorithme **mmCdgPars** va avoir une complexité $\mathbf{O}(\mathsf{n}^3)$. Leur avantage décisif est l'architecture compositionnelle de dépendances où toutes les dépendances, projectives comme non bornées, sont définies par les types fonctionnels, ce qui crée la base nécessaire pour une sémantique fonctionnelle de dépendances. En même temps, cette architecture adopte naturellement la multimodalité des dépendances non bornées correspondant aux règles de saturation des valences spécifiques aux différentes langues. Il est important de noter que cette flexibilité syntaxique est atteinte sans explosion du coût de l'analyse syntaxique (par contraste avec les grammaires de Lambek). Malgré leur simplicité, les mmCDG sont très expressives. On a vu que pour exprimer les dépendances croisées illimitées on n'a pas besoin du langage de copies, mais d'un langage des SD facilement exprimé par les mmCDG. Et le fait que MIX est un langage $mmCDG^{\mathbf{FA}}$ montre que ces grammaires sont adaptées aux langues naturelles avec l'ordre des mots flexible.

Enfin, il est difficile de comparer les mmCDG par l'expressivité avec les autres GD qui traitent les dépendances non bornées et qui les analysent en temps polynomial, e.g. (Kahane *et al.*, 1998; Bröker, 2000). Le pouvoir de ces grammaires n'est pas déterminée. Leurs définitions sont opérationnelles (cf. le « lifting »). L'avantage des mmCDG est leur transparence et leur architecture compositionnelle de dépendances.

Références

BAR-HILLEL Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, **29**(1), 47–58.

BOULLIER P. (2003). Counting with range concatenation grammars. *Theoretical Computer Science*, **293**, 391–416.

BRESNAN J., KAPLAN R., PETERS S. & ZAENEN A. (1982). Cross-serial dependencies in dutch. *Linguistic Inquiry*, **13**(4), 613–635.

BRÖKER N. (1998). Separating surface order and syntactic relations in a dependency grammar. In *Proc. COLING-ACL*, p. 174–180, Montreal.

BRÖKER N. (2000). Unordered and non-projective dependency grammars. *Traitement Automatique des Langues (TAL)*, **41**(1), 245–272.

DEKHTYAR M. & DIKOVSKY A. (2004). Categorial dependency grammars. In M. MOORTGAT & V. PRINCE, Eds., *Proc. of Intern. Conf. on Categorial Grammars*, p. 76–91.

DEKHTYAR M. & DIKOVSKY A. (2007). Generalized categorial dependency grammars. In submission, `www.sciences.univ-nantes.fr/info/perso/permanents/dikovsky/`.

DIKOVSKY A. (2001). Polarized non-projective dependency grammars. In P. DE GROOTE, G. MORILL & C. RETORÉ, Eds., *Proc. of the Fourth Intern. Conf. on Logical Aspects of Computational Linguistics*, volume 2099 of *LNAI*, p. 139–157 : Springer.

DIKOVSKY A. (2004). Dependencies as categories. In *"Recent Advances in Dependency Grammars". COLING'04 Workshop*, p. 90–97.

DUCHIER D. (1999). Axiomatizing dependency parsing using set constraints. In *Sixth Meeting on Mathematics of Language (MOL-6)*, p. 115–126, Orlando, Florida.

DUCHIER D. & DEBUSMANN R. (2001). Topological dependency trees : A constraint-based account of linear precedence. In *Proc. of the Intern. Conf. ACL'2001*, p. 180–187 : ACL & Morgan Kaufman.

DUCHIER D., DEBUSMANN R. & KRUIJFF G.-J. M. (2004). Extensible dependency grammar : A new methodology. In *COLING'04 Workshop*, p. 78–84, Geneva.

GAIFMAN H. (1961). *Dependency systems and phrase structure systems*. Report p-2315, RAND Corp. Santa Monica (CA). Published in Information and Control, 1965, v. 8, n ° 3, pp. 304-337.

JOSHI A. K., SHANKER V. K. & WEIR D. J. (1991). The convergence of mildly context-sensitive grammar formalisms. In P. SELLS, S. SHIEBER & T. WASOW, Eds., *Foundational issues in natural language processing*, p. 31–81, Cambridge, MA : MIT Press.

KAHANE S., NASR A. & RAMBOW O. (1998). Pseudo-projectivity : A polynomially parsable non-projective dependency grammar. In *Proc. COLING-ACL*, p. 646–652, Montreal.

KRUIJFF G.-J. M. (2001). *A Categorial-Modal Logical Architecture of Informativity : Dependency Grammar Logic & Information Structure*. PhD thesis, Charles University, Prague.

LAMBEK J. (1961). On the calculus of syntactic types. In R. JAKOBSON, Ed., *Structure of languages and its mathematical aspects*, p. 166–178. Providence RI : American Mathematical Society.

MOORTGAT M. (1997). Categorial type logics. In J. VAN BENTHEM & A. TER MEULEN, Eds., *Handbook of Logic and Language*, chapter 2, p. 93–177. Elsevier, The MIT Press.

MORRILL G. V. (1994). *Type Logical Grammar. Categorial Logic of Signs*. Kluwer.

POLLARD C. & SAG I. (1988). *An Information Based Approach to Syntax and Semantics, Part I*. Stanford, California : CSLI.

PULMAN S. & RITCHIE G. (1985). Indexed grammars and interesting dependencies. *UEA Papers in Linguistics*, **23**, 21–38.

M. STEEDMAN, Ed. (1996). *Surface Structure and Interpretation*. The MIT Press.

VIJAY-SHANKER K. & WEIR D. (1994). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, **27**, 511–545.