TALN 2009, Senlis, 24–26 juin 2009

Analyse déductive pour les grammaires d'interaction

Joseph Le Roux
NCLT, Dublin City University
jleroux@computing.dcu.ie

Résumé. Nous proposons un algorithme d'analyse pour les grammaires d'interaction qui utilise le cadre formel de l'analyse déductive. Cette approche donne un point de vue nouveau sur ce problème puisque les méthodes précédentes réduisaient ce dernier à la réécriture de graphes et utilisaient des techniques de résolution de contraintes. D'autre part, cette présentation permet de décrire le processus de manière standard et d'exhiber les sources d'indéterminisme qui rendent ce problème difficile.

Abstract. We propose a parsing algorithm for Interaction Grammars using the deductive parsing framework. This approach brings new perspectives on this problem, departing from previous methods relying on constraint-solving techniques to interpret it as a graph-rewriting problem. Furthermore, this presentation allows a standard description of the algorithm and a fine-grained inspection of the sources of non-determinism.

Mots-clés : Analyse syntaxique, grammaires d'interaction.

Keywords: Parsing, Interaction Grammars.

1 Introduction

Une grammaire d'interaction (GI) (Guillaume & Perrier, 2008) permet de décrire la syntaxe d'une langue en insistant sur la valence, c'est-à-dire la capacité des mots à se combiner. Cette valence s'exprime au moyen de polarités qui décorent les syntagmes ou les traits associés à ces syntagmes. D'autres formalismes utilisent cette notion, comme les grammaires catégorielles (Lambek, 1958), les grammaires d'unification polarisées (Kahane, 2004) ou encore les grammaires minimalistes (Chomsky, 1995). Une autre caractéristique des GI est l'utilisation de structures dites sous-spécifiées. Les grammaires de (Duchier & Thater, 1999) utilisent aussi ces structures mais avec un système de polarités moins riche.

Dans cet article, nous présentons un algorithme proche de celui décrit par (Earley, 1970) pour l'analyse des grammaires hors-contexte. D'autres méthodes d'analyse existent pour les GI, comme par exemple la méthode *shift-reduce* de (Bonfante *et al.*, 2003) qui réduit l'analyse à la réécriture de graphes. Ici, nous considérons une approche radicalement différente. Nous introduisons notre algorithme en utilisant une méthode standard de ce domaine : l'analyse déductive (Shieber *et al.*, 1995). Cette approche évite d'inventer des notions et des notations ad hoc. Elle permet aussi aux lecteurs qui y sont habitués de comprendre rapidement les spécificités de l'analyse des GI. (Le Roux, 2007) décrit un algorithme très proche mais notre présentation, en décomposant la prédiction, est bien plus simple.

Le reste de l'article s'organise comme suit : nous présentons les GI (section 2), puis nous dé-

crivons l'algorithme (section 3). Nous discutons ensuite certains aspects techniques de l'algorithme (section 4) avant de conclure (section 5).

2 Les grammaires d'interaction

Nous reprenons la définition des GI donnée par (Guillaume & Perrier, 2008) [1]. Cependant, nous présentons ici une version sans structure de traits de manière à simplifier l'exposé.

2.1 Descriptions d'arbre polarisées

La structure fondamentale des GI est la description d'arbre polarisée (DAP) qui représente un fragment d'arbre d'analyse. Elle contient des nœuds polarisés, c'est-à-dire décorés de polarités.

Les GI distinguent 4 polarités $\mathbb{P} = \{\rightarrow, \leftarrow, =, \sim\}$, respectivement positive, négative, neutre et virtuelle. Un multi-ensemble de polarités est *saturé* s'il contient exactement une polarité positive et exactement une polarité négative, ou bien s'il ne contient aucune polarité positive ni négative et au moins une polarité neutre. Un multi-ensemble de polarités est *superposable* s'il contient au plus une polarité positive et au plus une polarité négative.

Les nœuds polarisés sont étiquetés par une catégorie et une polarité. Un ensemble de nœuds est saturé (resp. superposable) si tous ses éléments ont la même catégorie et si le multi-ensemble induit par leur polarité est saturé (resp. superposable).

Une DAP est un graphe. Il existe quatre relations binaires définies sur les nœuds polarisés d'une DAP : la parenté immédiate $>$, la parenté lâche $>^*$, la précédence immédiate $\prec$ et la précédence lâche $\prec^+$. De plus, pour être valide une DAP doit vérifier :
– $>$ et $>^*$ définissent une structure d'arbre,
– $\prec$ et $\prec^+$ ne sont définies que pour des nœuds ayant le même antécédent par $>$.
Dans chaque DAP il existe des nœuds feuilles (sans descendant par $>$ ou $>^*$), appelés ancres. Pour alléger l'exposé , si $n >^* m$, on appelle m un nœud *contraint* (par n) et pour un ensemble $\mathcal{N}$ de nœuds, on définit $\mathcal{N}^{>} = \{N | \exists M \in \mathcal{N}, M > N\}$ et $\mathcal{N}^{>^*} = \{N | \exists M \in \mathcal{N}, M >^* N\}$.

2.2 Grammaires d'interaction

Une GI est un tuple $\mathcal{G} = \{\Sigma, \mathbb{C}, S, \mathcal{P}, phon\}$, où Σ est l'alphabet des symboles terminaux, $\mathbb{C}$ est l'alphabet des symboles non-terminaux (ou catégories), $S \in \mathbb{C}$ est le symbole initial, $\mathcal{P}$ est un ensemble de DAP dont les nœuds sont étiquetés par des éléments de $\mathbb{C} \times \mathbb{P}$ et $phon$ est une fonction partielle des nœuds de $\mathcal{P}$ vers Σ. Elle n'est définie que pour les ancres.

Le résultat d'une analyse syntaxique est un *arbre syntaxique*, c'est-à-dire un arbre totalement ordonné dans lequel tous les nœuds sont étiquetés par une catégorie. On note $lab(A)$ l'étiquette du nœud A. Certains nœuds feuilles F sont étiquetés par un terminal t. Dans ce cas, on notera $mot(F) = t$, et $mot(F) = \varepsilon$ sinon.

On notera $M \gg N$ si le nœud M est le père du nœud N et $N \gg [N_1, \ldots, N_k]$ si le nœud N est le père des nœuds de la liste ordonnée $[N_1, \ldots, N_k]$. L'ordre entre nœuds fils s'exprime à l'aide

1. Nous reportons les lecteurs à cet article pour ce qui concerne l'utilisation linguistique des GI.

de de la relation $\prec\!\!\prec$. $M \prec\!\!\prec N$ indique que N est le successeur immédiat de M. On notera $\prec\!\!\prec^+$ la clôture transitive de $\prec\!\!\prec$ et $\gg^*$ la clôture réflexive transitive de $\gg$.

Enfin nous avons besoin de la projection phonologique PP d'un nœud, définie récursivement :

$$PP(M) = \begin{cases} [t] \quad \text{si} \quad M \gg [] \quad \text{et} \quad mot(M) = t \\ [PP(N_1) \ldots PP(N_k)] \quad \text{si} \quad M \gg [N_1, \ldots, N_k] \end{cases}$$

Un arbre syntaxique T est un *modèle* du multi-ensemble de DAP $\mathcal{D}$ s'il existe une fonction totale I des nœuds de $\mathcal{D}$ (notés $\mathcal{ND}$) vers les nœuds de T (notés $\mathcal{NT}$) qui vérifie :

1. si $A \in \mathcal{NT}$ alors $I^{-1}(A)$ est saturé et non-vide.
2. si $M, N \in \mathcal{ND}$ et $M > N$ alors $I(M) \gg I(N)$
3. si $M, N \in \mathcal{ND}$ et $M >^* N$ alors $I(M) \gg^* I(N)$
4. si $M, N \in \mathcal{ND}$ et $M \prec N$ alors $I(M) \prec\!\!\prec I(N)$
5. si $M, N \in \mathcal{ND}$ et $M \prec^+ N$ alors $I(M) \prec\!\!\prec^+ I(N)$
6. si $A, B \in \mathcal{NT}$ et $A \gg B$ alors il existe $M \in I^{-1}(A)$ et $N \in I^{-1}(B)$ tels que $M > N$
7. si $A \in \mathcal{NT}$ alors $lab(A) = lab(M)$ pour tout $M \in I^{-1}(A)$
8. si $M \in \mathcal{ND}$ et $phon(M) = m$ alors $PP(I(M)) = [m]$

Étant donné une GI $\mathcal{G} = \{\Sigma, \mathbb{C}, S, \mathcal{P}, phon\}$ et une phrase $p = m_1, \ldots, m_n$ de Σ^*, un arbre syntaxique T est un arbre d'analyse pour p s'il existe un multi-ensemble $\mathcal{D}$ de DAP issues de $\mathcal{P}$ tel que T est un modèle de $\mathcal{D}$, la racine R de T est étiquetée par S et $PP(R) = [m_1, \ldots, m_n]$. Le langage engendré par $\mathcal{G}$ est l'ensemble des phrases de Σ^* pour lesquelles il existe un arbre d'analyse.

3 Définition de l'algorithme

Nous utilisons le cadre de l'analyse déductive (Shieber *et al.*, 1995) pour expliquer notre algorithme : un état de l'analyse est décrit par un item et des règles de déduction permettent d'obtenir de nouveaux items à partir d'items déjà créés. On applique ces règles jusqu'à stabilisation de l'ensemble d'items. L'analyse est amorcée par la création d'un item axiome. La phrase d'entrée appartient au langage si un item spécifique, appelé item but est créé durant l'analyse [2].

3.1 Les items

Nos items sont de la forme $[A(H, N, F) \rightarrow \alpha \bullet \beta, i, j, (O, U, D)]$. Ils sont constitués d'une règle pointée, de deux indices de position $0 \leq i \leq j \leq n$, où n est la longueur de la phrase d'entrée, et d'un triplet d'ensembles de nœuds qui contrôle l'utilisation des nœuds contraints.

La règle pointée $A(H, N, F) \rightarrow \alpha \bullet \beta$ affirme qu'il existe un nœud A de l'arbre d'analyse, modèle de l'ensemble $H \cup N \cup F$. Les éléments A_l dans α sont également des nœuds de l'arbre syntaxique. Ils indiquent que des sous-analyses ont déjà été effectuées et que l'on a trouvé pour

2. Nous présentons ici un reconnaisseur. Pour en faire un analyseur, il faudrait garder l'historique des items.

ces nœuds des ensembles d'antécédents saturés. Les éléments $B_k(H_k)$de β signifient qu'il existe un nœud B_k dans l'arbre syntaxique dont un sous-ensemble des antécédents est H_k. De plus H_k est constitué uniquement de nœuds dans $(H \cup N \cup F)^>$. Cet item prédit que l'arbre syntaxique contient $A \gg [A_1 \dots A_k B_1 \dots B_l]$ et que $PP(A_1) \circ \dots \circ PP(A_k) = [m_{i+1} \dots m_j]$.

C'est le contrôle de l'utilisation des nœuds contraints qui complique la tâche de l'analyseur. Le triplet d'ensembles de nœuds (O, U, D) permet de vérifier les contraintes sur ces nœuds :
– Les ensemble O et D contiennent des nœuds contraints qui seront disponibles quand on cherchera des antécédents pour prédire l'existence d'un nouveau nœud dans l'arbre syntaxique.
– Les nœuds de O seront utilisés obligatoirement dans la sous-analyse courante. Pour qu'un item puisse compléter une analyse, il faudra impérativement que cet ensemble soit vide.
– L'ensemble U contient des nœuds qui étaient disponibles puis ont été utilisés sans que l'on ait encore vérifié à quelle sous-analyse ils devaient appartenir.

Par ailleurs, on utilisera 3 symboles supplémentaires :
– $\top$, la partie gauche de la règle pointée axiome. On peut voir $\top$ comme une racine ajoutée à l'arbre syntaxique durant sa construction.
– Le point $\bullet$ pourra devenir $\blacksquare$ ou $\blacklozenge$ dans les règles de préparation à la prédiction (p_1 et p_2) pour indiquer que les items les contenant ne peuvent pas être utilisés dans d'autres règles.

Nous aurons besoin de construire des suites d'ensembles de nœuds superposables qui respectent les relations de précédence des DAP. Étant donné un ensemble de nœuds $\mathcal{N}$, nous définissons :

$$
\begin{aligned}
ord(\mathcal{N}) = \{[\mathcal{N}_1 \dots \mathcal{N}_k] \mid\ & (\mathcal{N}_i)_{1 \leq i \leq k} \text{ est une partition de } \mathcal{N} \wedge \\
& 1 \leq i \leq k, \mathcal{N}_i \text{ est superposable} \wedge \\
& \text{si } n_1, n_2 \in \mathcal{N} \text{ et } n_1 \prec n_2 \text{ alors } \exists 1 \leq j < k \text{ t.q. } n_1 \in \mathcal{N}_j \text{ et } n_2 \in \mathcal{N}_{j+1} \wedge \\
& \text{si } n_1, n_2 \in \mathcal{N} \text{ et } n_1 \prec^+ n_2 \text{ alors } \exists 1 \leq i < j \leq k \text{ t.q. } n_1 \in \mathcal{N}_i \text{ et } n_2 \in \mathcal{N}_j\}
\end{aligned}
$$

3.2 Les règles de déduction

Dans cette section, on suppose vouloir analyser une phrase d'entrée $p = m_1, \dots, m_n$ avec une GI $\mathcal{G} = \{\Sigma, \mathbb{C}, S, \mathcal{P}, phon\}$.

Axiome C'est la règle de départ. On se prépare à prédire un nœud de catégorie initiale S. Aucun mot n'a été lu et il n'y a aucune contrainte sur les relations lâches.

$$\frac{}{[\top \rightarrow \bullet S(\emptyset), 0, 0, (\emptyset, \emptyset, \emptyset)]} ax$$

Prédiction C'est la règle qui permet de commencer une sous-analyse. Nous l'avons divisée en 3 sous-règles pour introduire les différentes contraintes séparément.

$$\frac{[A(H, N, F) \rightarrow \alpha \bullet C(H_c)\beta, i, j, (O, U, D)]}{[C(H_C, \emptyset, \emptyset) \rightarrow \blacksquare, j, j, (\emptyset, U, D \cup O)]} p_1$$

Dans cette première étape, on va commencer une nouvelle sous-analyse à la limite de l'analyse courante, en position j. On indique que le *focus* se situe sur un nœud C, pour lequel on a déjà choisi une partie des antécédents H_C.

Les nœuds de O, qui sont les nœuds à utiliser obligatoirement dans la sous-analyse de A deviennent des nœuds disponibles pour l'analyse de C et toutes les sous-analyses suivantes.

$$\frac{[C(H_C, \emptyset, \emptyset) \rightarrow \blacksquare, j, j, (\emptyset, U_1, D_1)]}{[C(H_C, N_C, \emptyset) \rightarrow \blacklozenge, j, j, (\emptyset, U_2, D_2)]} p_2 \begin{cases} H_C \cup N_C \neq \emptyset \\ H_C \cup N_C \text{ est superposable} \\ N_C \subset D_1 \cup \mathcal{P} \\ D_2 = D_1 - N_C \\ U_2 = U_1 \cup (D_1 \cap N_C) \end{cases}$$

Dans cette seconde sous-règle, les antécédents de C sont complétés avec l'ensemble de nœuds N_C, choisis parmi les nœuds disponibles D_1 et les racines des DAP de la GI dans $\mathcal{P}$ [3]. Le triplet de vérification est ensuite mis à jour. Les nœuds contraints choisis pour compléter C ne sont plus disponibles et sont ajoutés à l'ensemble des nœuds utilisés.

$$\frac{[C(H_C, N_C, \emptyset) \rightarrow \blacklozenge, j, j, (\emptyset, U, D)]}{[C(H_C, N_C, F_C) \rightarrow \bullet\gamma, j, j, (O, U, D)]} p_3 \begin{cases} H_C \cup N_C \cup F_C \text{ est saturé} \\ \gamma \in ord((H_C \cup N_C \cup F_C)^>) \\ F_C = \bigcup_i Q_i, Q_0 \subseteq (H_C \cup N_C)^{>^*}, Q_{i+1} \subseteq Q_i^{>^*} \\ O = (H_C \cup N_C \cup F_C)^{>^*} - F_C \\ \text{aucun nœud ancre dans } H_C \cup N_C \cup F_C \end{cases}$$

Dans la dernière étape de la prédiction, on complète les antécédents de C avec des nœuds contraints par des nœuds déjà antécédents de C. On peut aussi sélectionner des nœuds dans la clôture transitive de cette relation à condition que leur prédécesseurs aient été eux-mêmes sélectionnés. L'ensemble des antécédents doit alors être saturé. Il faut ensuite prédire la forme des prochaines sous-analyses. Pour cela, il faut grouper les fils des antécédents et respecter leur catégories et les relations de précédence qui existent entre eux. On choisit donc une partition de $(H_C, N_C, F_C)^>$ qui respecte ord. Enfin, les nœuds à utiliser obligatoirement dans les sous-analyses sont les nœuds contraints par les antécédents qui ne sont pas eux-mêmes antécédents.

Balayage C'est la règle qui vérifie les prédictions déjà effectuées par la présence d'un terminal à la position courante de l'analyse. C'est un cas particulier de la règle précédente quand l'un des antécédents de C est une ancre.

$$\frac{[C(H_C, N_C, \emptyset) \rightarrow \blacklozenge, j, j, (\emptyset, U, D)]}{[C(H_C, N_C, F_C) \rightarrow \bullet, j, j+1, (\emptyset, U, D)]} b \begin{cases} H_C \cup N_C \cup F_C \text{ est saturé} \\ (H_C \cup N_C \cup F_C)^> = \emptyset \\ F_C = \bigcup_i Q_i, Q_0 \subseteq (H_C \cup N_C)^{>^*}, Q_{i+1} \subseteq Q_i^{>^*} \\ (H_C \cup N_C \cup F_C)^{>^*} - F_C = \emptyset \\ \text{un unique nœud ancre } a \text{ dans } H_C \cup N_C \cup F_C \\ phon(a) = p_{j+1} \end{cases}$$

Si on lit sur la chaîne d'entrée le terminal attendu à la position courante, on fait progresser l'analyse. Cette règle ne s'applique que si tous les nœuds contraints par les antécédents sont eux-mêmes antécédents, puisqu'il n'y a pas de sous-analyse dans laquelle utiliser ces nœuds.

3. Par abus de langage, on notera de la même façon une DAP et sa racine. Il faut également noter qu'une racine peut être sélectionnée plusieurs fois (si plusieurs occurrences du même mot apparaissent dans la phrase à analyser par exemple) et qu'un renommage de nœuds peut s'imposer pour distinguer chaque occurrence.

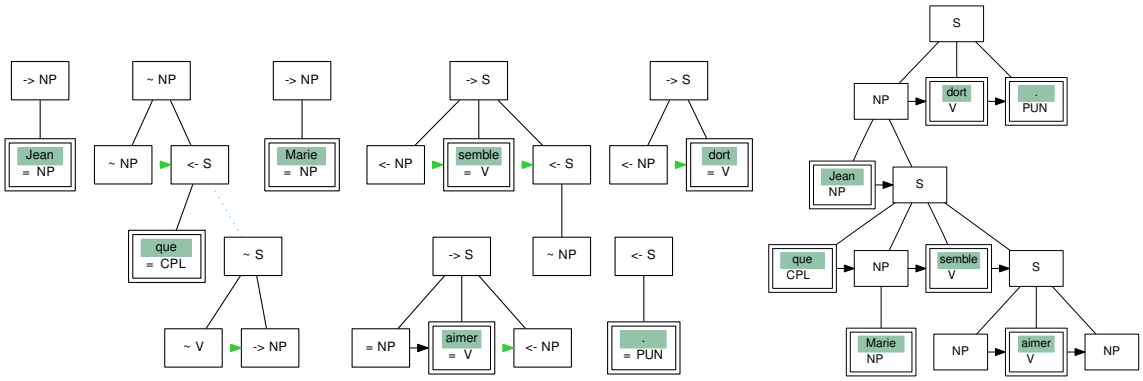

FIGURE 1 – les DAP et l'arbre d'analyse pour *Jean que Marie semble aimer dort.*

Complétion Cette règle permet de revenir d'une sous-analyse et d'étendre l'analyse courante.

$$\frac{[A(H,N,F) \rightarrow \alpha \bullet C(H_c)\beta, i, j, (O_1, U_1, D_1)] \quad [C(H_C, N_C, F_C) \rightarrow \gamma\bullet, j, k, (\emptyset, U_2, D_2)]}{[A(H,N,F) \rightarrow \alpha C \bullet \beta, i, k, (O_3, U_3, D_3)]}\ c \begin{cases} N_C \subseteq D_1 \cup O_1 \cup \mathcal{P} \\ D_2 \subseteq (D_1 \cup O_1) - N_C \\ U_1 \subseteq U_2 \\ O_3 = O_1 - U_2 \\ D_3 = D_1 - U_2 \\ U_3 = U_2 - O_1 \end{cases}$$

On doit ici s'assurer que la sous-analyse de C peut être *branchée* sur l'analyse courante :
– L'ensemble des nœuds disponibles dans la sous-analyse est un sous-ensemble des nœuds disponibles dans l'analyse principale. En d'autres termes, la sous-analyse a pu utiliser des nœuds qui étaient disponibles dans l'analyse principale mais n'a pas pu rendre de nouveaux nœuds disponibles sans les avoir utilisés.
– Pour les mêmes raisons, l'ensemble des nœuds utilisés dans l'analyse principale doit être un sous-ensemble des nœuds utilisés dans la sous-analyse.
– On retire des nœuds à utiliser obligatoirement ceux qui ont été utilisés dans la sous-analyse.

Item but L'analyse réussit si l'on obtient $[\top \rightarrow S\bullet, 0, n, (\emptyset, \emptyset, \emptyset)]$.

3.3 Exemple

Pour voir comment les relations de parenté lâche sont contrôlées, nous allons analyser la phrase *Jean que Marie semble aimer dort.* On peut voir les DAP utilisées pour l'analyse sur la Figure 1. Elles proviennent du logiciel LEOPAR [4]. Nous appellerons les descriptions j, q, m, s, a, d et p et nous désignerons les positions des nœuds par leur adresse de Gorn. Par exemple la racine de la DAP associée à *Marie* est m et le nœud le plus éloigné de la racine de la DAP associée à *semble* est s_{31}. Les relations $>$ et $>^*$ sont représentées par des traits, respectivement plein et pointillé. Les relations $\prec$ et $\prec^+$ sont représentées par des flèches, respectivement noire et colorée. Les items qui permettent d'arriver une analyse et les règles pour les produire sont listés dans la Table 1.

4. Ce logiciel est disponible à `http://leopar.loria.fr/`.

L'algorithme commence par prédire une racine S image de d et p (items 1, 2 et 3), puis ordonne les fils de ces nœuds (item 4). L'analyse se poursuit par la prédiction d'un nœud NP dont un des antécédents doit être d_1, puisque d a été choisi plus tôt. L'analyse se poursuit jusqu'à la prédiction du nœud S (item 12) dont la projection phonologique est *que Marie semble aimer.* Ce nœud a pour antécédents q_2 et s. Le nœud q_{22} contraint par q_2 n'est pas antécédent, il doit donc être obligatoirement utilisé dans cette sous-analyse et il est ajouté à l'ensemble O de cet item. Ce nœud devient ensuite disponible pour l'analyse de *que* et *semble* mais n'est pas utilisé. Il est donc toujours dans l'ensemble O de l'item 22, ainsi que dans celui de l'item 24.

Les items 25, 26 et 27 décrivent la prédiction du nœud S dont la projection phonologique est *aimer*. On sélectionne q_{22} comme antécédent qui devient utilisé (ensemble U). Lors de la complétion du S dont la projection phonologique est *que Marie semble aimer* (item 34), le *contrat* qui forçait l'utilisation de q_{22} a été rempli et on retire donc ce nœud des nœuds obligatoires. Le reste de l'analyse ne pose pas de problème particulier.

4 Discussion

4.1 Correction et complétude

L'algorithme présenté maintient un invariant tout au long de l'analyse. Chaque item de la forme $[A(H, N, F) \rightarrow \alpha \bullet \beta, i, j, (O, U, D)]$ assure que :

– A est modèle d'un ensemble saturé de nœuds qui ne sont plus disponibles pour être antécédents d'un autre nœud de l'arbre syntaxique en construction. Il en est de même des nœuds dans α. Les conditions 1, 7 et 3 (cas réflexif) que doit vérifier un modèle sont respectées.
– Les ensembles β_k sont superposables. On a $\beta_k \subseteq (A^{-1})^{>}$ (conditions 2 et 6)
– l'ordre des $\alpha\beta$ est compatible avec les relations d'ordre des DAP (conditions 4 et 5).
– $PP(\alpha_1) \circ \ldots \circ PP(\alpha_l) = [m_{i+1} \cdots m_j]$
– les nœuds de O sont des nœuds contraints en relation avec des nœuds de DAP qui sont antécédents de A et qui n'ont pas encore été utilisés comme antécédents.
– les nœuds de D sont des nœuds contraints en relation avec des nœuds de DAP qui sont antécédents de nœuds de l'arbre syntaxique situés entre sa racine et A et qui n'ont pas encore été utilisés comme antécédents
– un nœud N de U est un nœud contraint en relation par $>^*$ avec un nœud de DAP qui est antécédent d'un nœud situé à la fois entre la racine de l'arbre syntaxique et A, distinct de A et entre la racine et $I(N)$ (condition 3).

On peut vérifier cet invariant par induction sur les règles. En d'autres termes un tel item affirme qu'il existe une fonction partielle J des nœuds d'un sous-ensemble des DAP d'une GI vers un arbre syntaxique de racine étiquetée par O et qui a pour projection phonologique $m_1 \ldots m_j$. Cette fonction J est similaire à la fonction I des modèles. Elle vérifie les mêmes propriétés mais les conditions 2–5 ne sont respectées que si les deux nœuds sont dans le domaine. L'algorithme étend cette fonction J jusqu'à (1) l'obtention d'une fonction totale et (2) la couverture complète de la chaîne d'entrée. J définit alors un arbre syntaxique qui est un arbre d'analyse.

D'autre part, s'il existe un arbre syntaxique pour une GI et une phrase d'entrée, un parcours préfixe de cet arbre permet de retrouver les items créés par l'algorithme

1	$[\top \rightarrow \bullet S(\emptyset), 0, 0, (\emptyset, \emptyset, \emptyset)]$	ax
2	$[S(\emptyset, \emptyset, \emptyset) \rightarrow \blacksquare, 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_1(1)$
3	$[S(\emptyset, \{d, p\}, \emptyset) \rightarrow \blacklozenge, 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_2(2)$
4	$[S(\emptyset, \{d, p\}, \emptyset) \rightarrow \bullet \textit{NP}(d_1)V(d_2)\textit{PUN}(p_1), 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_3(3)$
5	$[\textit{NP}(\{d_1\}, \emptyset, \emptyset) \rightarrow \blacksquare, 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_1(4)$
6	$[\textit{NP}(\{d_1\}, \{j, q\}, \emptyset) \rightarrow \blacklozenge, 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_2(5)$
7	$[\textit{NP}(\{d_1\}, \{j, q\}, \emptyset) \rightarrow \bullet \textit{NP}(j_1, q_1)S(q_2), 0, 0, (\emptyset, \emptyset, \emptyset)]$	$p_3(6)$
8	$[\textit{NP}(\{j_1, q_1\}, \emptyset, \emptyset) \rightarrow \bullet, 0, 1, (\emptyset, \emptyset, \emptyset)]$	$b \circ p_2 \circ p_1(7)$
9	$[\textit{NP}(\{d_1\}, \{j, q\}, \emptyset) \rightarrow \textit{NP} \bullet S(q_2), 0, 1, (\emptyset, \emptyset, \emptyset)]$	$c(7, 8)$
10	$[S(\{q_2\}, \emptyset, \emptyset) \rightarrow \blacksquare, 1, 1, (\emptyset, \emptyset, \emptyset)]$	$p_1(9)$
11	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \blacklozenge, 1, 1, (\emptyset, \emptyset, \emptyset)]$	$p_2(10)$
12	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \bullet \textit{CPL}(q_{21})\textit{NP}(s_1)V(s_2)S(s_3), 1, 1, (\{q_{22}\}, \emptyset, \emptyset)]$	$p_3(11)$
13	$[\textit{CPL}(\{q_{21}\}, \emptyset, \emptyset) \rightarrow \blacksquare, 1, 1, (\emptyset, \emptyset, \{q_{22}\})]$	$p_1(12)$
14	$[\textit{CPL}(\{q_{21}\}, \emptyset, \emptyset) \rightarrow \blacklozenge, 1, 1, (\emptyset, \emptyset, \{q_{22}\})]$	$p_2(13)$
15	$[\textit{CPL}(\{q_{21}\}, \emptyset, \emptyset) \rightarrow \bullet, 1, 2, (\emptyset, \emptyset, \{q_{22}\})]$	$b(14)$
16	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \textit{CPL} \bullet \textit{NP}(s_1)V(s_2)S(s_3), 1, 2, (\{q_{22}\}, \emptyset, \emptyset)]$	$c(12, 15)$
17	$[\textit{NP}(\{s_1\}, \emptyset, \emptyset) \rightarrow \blacksquare, 2, 2, (\emptyset, \emptyset, \{q_{22}\})]$	$p_1(16)$
18	$[\textit{NP}(\{s_1\}, \{m\}, \emptyset) \rightarrow \blacklozenge, 2, 2, (\emptyset, \emptyset, \{q_{22}\})]$	$p_2(17)$
19	$[\textit{NP}(\{s_1\}, \{m\}, \emptyset) \rightarrow \bullet \textit{NP}(m_1), 2, 2, (\emptyset, \emptyset, \{q_{22}\})]$	$p_3(18)$
20	$[\textit{NP}(\{m_1\}, \emptyset, \emptyset) \rightarrow \bullet, 2, 3, (\emptyset, \emptyset, \{q_{22}\})]$	$b \circ p_2 \circ p_1(19)$
21	$[\textit{NP}(\{s_1\}, \{m\}, \emptyset) \rightarrow \textit{NP}\bullet, 2, 3, (\emptyset, \emptyset, \{q_{22}\})]$	$c(19, 20)$
22	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \textit{CPL NP} \bullet V(s_2)S(s_3), 1, 3, (\{q_{22}\}, \emptyset, \emptyset)]$	$c(16, 21)$
23	$[V(\{s_2\}, \emptyset, \emptyset) \rightarrow \bullet, 3, 4, (\emptyset, \emptyset, \{q_{22}\})]$	$b \circ p_2 \circ p_1(22)$
24	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \textit{CPL NP } V \bullet S(s_3), 1, 4, (\{q_{22}\}, \emptyset, \emptyset)]$	$c(22, 23)$
25	$[S(\{s_3\}, \emptyset, \emptyset) \rightarrow \blacksquare, 4, 4, (\emptyset, \emptyset, \{q_{22}\})]$	$p_1(24)$
26	$[S(\{s_3\}, \{a, q_{22}\}, \emptyset) \rightarrow \blacklozenge, 4, 4, (\emptyset, \{q_{22}\}, \emptyset)]$	$p_2(25)$
27	$[S(\{s_3\}, \{a, q_{22}\}, \emptyset) \rightarrow \bullet \textit{NP}(a_1, s_{31})V(a_2, q_{221})\textit{NP}(q_{222}, a_3), 4, 4, (\emptyset, \{q_{22}\}, \emptyset)]$	$p_3(26)$
28	$[\textit{NP}(\{a_1, s_{31}\}, \emptyset, \emptyset) \rightarrow \bullet, 4, 4, (\emptyset, \{q_{22}\}, \emptyset)]$	$p_3 \circ p_2 \circ p_1(27)$
29	$[S(\{s_3\}, \{a, q_{22}\}, \emptyset) \rightarrow \textit{ NP} \bullet V(a_2, q_{221})\textit{NP}(q_{222}, a_3), 4, 4, (\emptyset, \{q_{22}\}, \emptyset)]$	$c(27, 28)$
30	$[V(\{a_2, q_{221}\}, \emptyset, \emptyset) \rightarrow \bullet, 4, 5, (\emptyset, \{q_{22}\}, \emptyset)]$	$b \circ p_2 \circ p_1(29)$
31	$[S(\{s_3\}, \{a, q_{22}\}, \emptyset) \rightarrow \textit{ NP } V \bullet \textit{NP}(q_{222}, a_3), 4, 5, (\emptyset, \{q_{22}\}, \emptyset)]$	$c(29, 30)$
32	$[\textit{NP}(\{q_{222}, a_3\}, \emptyset, \emptyset) \rightarrow \bullet, 5, 5, (\emptyset, \{q_{22}\}, \emptyset)]$	$p3 \circ p_2 \circ p_1(31)$
33	$[S(\{s_3\}, \{a, q_{22}\}, \emptyset) \rightarrow \textit{ NP } V \textit{ NP}\bullet, 4, 5, (\emptyset, \{q_{22}\}, \emptyset)]$	$c(31, 32)$
34	$[S(\{q_2\}, \{s\}, \emptyset) \rightarrow \textit{CPL NP } V \textit{ } S\bullet, 1, 5, (\emptyset, \emptyset, \emptyset)]$	$c(24, 33)$
35	$[\textit{NP}(\{d_1\}, \{j, q\}, \emptyset) \rightarrow \textit{NP } S\bullet, 0, 5, (\emptyset, \emptyset, \emptyset)]$	$c(9, 34)$
36	$[S(\emptyset, \{d, p\}, \emptyset) \rightarrow \textit{NP} \bullet V(d_2)\textit{PUN}(p_1), 0, 5, (\emptyset, \emptyset, \emptyset)]$	$c(4, 35)$
37	$[V(\{d_2\}, \emptyset, \emptyset) \rightarrow \bullet, 5, 6, (\emptyset, \emptyset, \emptyset)]$	$b \circ p_2 \circ p_1(36)$
38	$[S(\emptyset, \{d, p\}, \emptyset) \rightarrow \textit{NP } V \bullet \textit{PUN}(p_1), 0, 6, (\emptyset, \emptyset, \emptyset)]$	$c(36, 37)$
39	$[\textit{PUN}(\{p_1\}, \emptyset, \emptyset) \rightarrow \bullet, 6, 7, (\emptyset, \emptyset, \emptyset)]$	$b \circ p_2 \circ p_1(38)$
40	$[S(\emptyset, \{d, p\}, \emptyset) \rightarrow \textit{NP } V \textit{ PUN}\bullet, 0, 7, (\emptyset, \emptyset, \emptyset)]$	$c(38, 39)$
41	$[\top \rightarrow S\bullet, 0, 7, (\emptyset, \emptyset, \emptyset)]$	$c(1, 40)$

TABLE 1 – Items pour l'analyse de *Jean que Marie semble aimer dort.*

4.2 Complexité

Le problème de l'analyse des GI est un problème NP-difficile dans le cas lexicalisé et même en l'absence d'ambiguïté lexicale (Bonfante *et al.*, 2003).

En regardant les règles de notre algorithme, on peut voir plusieurs sources d'indéterminisme :
– dans la règle p_2, il faut choisir de nouveaux antécédents (l'ensemble N_C) qui soient superposables avec les antécédents hérités des choix précédents (l'ensemble H_C). Ces nouveaux antécédents sont à choisir parmi les nœuds disponibles et les racines des DAP de la GI utilisée. Il y a un nombre exponentiel de tels choix, d'autant plus grand que les GI réalistes contiennent plus de 2000 DAP.
  Cependant, en pratique, on va filtrer les choix possibles grâce aux catégories et aux polarités. De plus, les GI utilisées dans l'implantation LEOPAR sont lexicalisées. On ne va donc considérer qu'un sous-ensemble des DAP de la grammaire, qui correspond aux DAP qui ont pour ancre un mot de la phrase d'entrée. Enfin, des techniques de filtrages lexical (*supertagging*) très efficaces ont été développées pour les GI, comme (Bonfante *et al.*, 2006) qui permettent de restreindre de façon drastique le nombre de DAP qui peuvent être utilisées.
– dans la règle p_3 et la règle b, il faut choisir un sous-ensemble de nœuds contraints par des antécédents déjà choisis. Ici encore il existe un nombre exponentiel de choix. Cependant, dans les GI utilisées en pratique, les nœuds des DAP ont au plus un successeur par la relation $>^*$ et il n'existe pas de chaîne de nœuds reliés par $>^*$. On peut donc borner le nombre de nœuds dans F_C par le nombre de nœuds dans $H_C \cup N_C$.
– dans la règle p_3, on doit ordonner et partitionner les fils par $>$ des antécédents. Dans le cas où il n'existe aucune relation de précédence entre ces fils, il y a à nouveau un nombre exponentiel de possibilité. Cependant, en pratique, le nombre de nœuds à ordonner/partitionner est petit. On peut imaginer calculer l'ordre de façon paresseuse en l'étendant à chaque complétion, comme le proposent (Nederhof *et al.*, 2003) pour les pomset-CFG.

On remarque que la règle qui fait intervenir le plus d'indices de position est la complétion et qu'il n'y a pas d'indéterminisme à cette étape. Ce n'est donc pas directement la taille de la phrase qui rend le problème de l'analyse des GI difficile mais la taille de la GI et des DAP à considérer. La longueur de la phrase ne joue qu'un rôle indirect, le nombre de DAP utilisables augmentant avec le nombre de mots.

5 Conclusion

Nous avons présenté un algorithme d'analyse pour les GI. Bien que nous ayons utilisé une version sans structure de traits, nous pensons qu'il n'y a aucune difficulté majeure à y ajouter un mécanisme d'unification.

L'originalité de notre travail réside dans l'utilisation du cadre formel de l'analyse déductive pour un formalisme qui se réclame de l'approche par *théorie des modèles* (Pullum & Scholz, 2001). Ce cadre formel permet de distinguer les sources de l'indéterminisme qui rendent difficile le problème de l'analyse dans les GI. Ce travail est donc un premier pas vers une étude plus approfondie de sa complexité.

À l'avenir, il sera intéressant de rechercher, comme pour la méthode shift-reduce ou comme pour les (k-)TT-MCTAG (Kallmeyer & Parmentier, 2008), des approximations de l'algorithme ou du formalisme qui ne considèrent qu'un nombre borné de nœuds à chaque étape, de manière

à rendre l'analyse efficace (polynomiale).

Références

BONFANTE G., GUILLAUME B. & PERRIER G. (2003). Analyse syntaxique électrostatique. *Traitement Automatique des Langues*.

BONFANTE G., LE ROUX J. & PERRIER G. (2006). Lexical disambiguation with polarities and automata. In O. H. IBARRA & H.-C. YEN, Eds., *The 11th International Conference on Implementation and Application of Automata (CIAA 2006)*.

CHOMSKY N. (1995). *The Minimalist Program*. MIT Press.

DUCHIER D. & THATER S. (1999). Parsing with tree descriptions : a constraint based approach. In *Natural Language Understanding and Logic Programming NLULP'99, Dec 1999, Las Cruces, New Mexico*.

EARLEY J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, **13**(2), 94–102.

GUILLAUME B. & PERRIER G. (2008). *Interaction Grammars*. Research Report RR-6621, INRIA.

KAHANE S. (2004). Grammaires d'unification polarisées. In *TALN2004, Fès, Maroc*, p. 233–242.

KALLMEYER L. & PARMENTIER Y. (2008). Convertir des grammaires d'arbres adjoints à composantes multiples avec tuples d'arbres (TT-MCTAG) en grammaires à concaténation d'intervalles (RCG). In *15e Conférence sur le Traitement Automatique des Langues Naturelles*, Avignon France : ATALA.

LAMBEK J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, **65**(3), 154–170.

LE ROUX J. (2007). *La coordination dans les grammaires d'interaction*. PhD thesis, Ecole Doctorale IAEM Lorraine ; Institut National Polytechnique de Lorraine - INPL.

NEDERHOF M., SATTA G. & SHIEBER S. (2003). Partially ordered multiset context-free grammars and ID/LP parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies*, p. 171–182, Nancy, France.

PULLUM G. & SCHOLZ B. (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *Proccedings of the 4th conference on Logical Aspects of Computational Linguistics*.

SHIEBER S., SCHABES Y. & PEREIRA F. P. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, **24**(1–2), 3–36.