Algorithme de Earley pour les grammaires d'interaction

Jonathan Marchand

NANCY 2 UNIVERSITE

UFR mathématiques et informatique

Algorithme de Earley pour les grammaires d'interaction

MÉMOIRE

présenté et soutenu publiquement le 30 juin 2006

dans le cadre du

Master Informatique de l'Université Nancy 2

(spécialité Traitement Automatique des Langues)

par

Jonathan Marchand

Composition du jury

Présidents :	Dominique Méry Guy Perrier
Encadrants :	Bruno Guillaume Guy Perrier

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Résumé

Les grammaires d'interaction sont un formalisme pour faire de l'analyse syntaxique et sémantique de la langue naturelle. Les objets syntaxiques de base sont des descriptions d'arbres polarisées qui spécifient partiellement des arbres syntaxiques. L'originalité ses grammaires d'interaction réside dans l'utilisation d'un système de polarités pour gérer la notion de ressources consommables. Dans ce contexte, l'analyse syntaxique est un procédé qui consiste à construire des modèles de descriptions sous la forme d'arbres syntaxiques complètement spécifiés et neutralisés.
Dans ce rapport, nous nous proposons d'adapter une stratégie d'analyse syntaxique descendante de type Earley aux grammaires d'interaction. Ce travail est le prolongement de la définition d'un algorithme de Earley pour une version simplifiée de ces grammaires par Joseph Le Roux et a fait l'objet d'une implémentation dans LEOPAR[1].

Mots-clés: grammaire d'interaction, description d'arbres, analyse syntaxique

[1] LEOPAR est un analyseur syntaxique pour les grammaires d'interaction développé par CALLIGRAMME (http ://www.loria.fr/equipes/calligramme/leopar/)

Remerciements

Je tiens à remercier :

– Guy Perrier, qui a permis que le master se déroule dans les meilleures conditions, pour sa gentillesse et tout le secours qu'il a pu m'apporter à plusieurs reprises.

– Bruno Guillaume pour sa disponibilité dans l'encadrement de mon mémoire et sa sympathie.

– Joseph Le Roux pour l'écoute qu'il m'a accordé et sa collaboration dans le mémoire.

– Les membres du projet CALLIGRAMME qui m'ont tous réservé un très bon accueil.

– Nadine Beurné qui m'a sauvé de plusieurs catastrophes administratives.

Table des matières

Introduction

Les grammaires d'interaction ont été conçues pour modéliser la syntaxe des langues naturelles. Elles s'inspirent de deux traditions différentes des grammaires formelles : les grammaires catégorielles et les grammaires d'arbres adjoints.
Des grammaires catégorielles, les grammaires d'interaction reprennent l'idée que les syntagmes sont des ressources consommables et qu'il y a une dualité entre celles-ci qui s'exprime dans le principe de composition. Dans un même temps, les grammaires d'interaction introduisent un assouplissement considérable dans le formalisme en ayant recours à la notion de *description d'arbres*. L'intérêt d'une telle approche est qu'elle permet une grande souplesse de composition et une expression économique de la sous-spécification. L'analyse syntaxique peut alors s'exprimer comme la construction d'un modèles de descriptions d'arbres par un processus de superposition d'arbres régi par la neutralisation de polarités opposées.

Les algorithmes d'analyse syntaxique les plus connus analysent des langages formels (souvent des langages de programmation). Cependant les langages naturels, contrairement aux langages formels, sont ambiguës. Les techniques les plus efficaces pour analyser les langages ambiguës s'appuient sur un principe de tabulation afin de ne pas analyser plusieurs fois les mêmes sous-arbres. C'est dans cette tradition qu'ont été développés les algorithmes de CYK et de Earley. Bien que ces algorithmes soient conçus à l'origine pour les grammaires hors-contextes, le principe de tabulation semble être voie intéressante pour une analyse efficace des grammaires d'interaction [Vil99].

Un logiciel d'analyse syntaxique basé sur les grammaires d'interaction est développé actuellement dans l'équipe CALLIGRAMME. Ce logiciel est conçu de façon modulaire afin de pouvoir tester différentes stratégies d'analyses. Actuellement, il existe deux stratégies d'analyse implémentées dans LEOPAR : une première stratégie utilise une procédure de type Shift/Reduce et la seconde de type CYK.
Nous nous proposons ici d'étudier une autre stratégie d'analyse bien connue en analyse syntaxique, il s'agit de l'algorithme de Earley. L'originalité de cette algorithme par rapport aux autres stratégies d'analyse est qu'il fait une recherche descendante dans la structure d'analyse.

Le rapport se présente de la manière suivante :

- Nous présentons tout d'abord les outils théoriques manipulés en rappelant le principe de l'analyse syntaxique descendante de Earley, puis en présentant une formalisation des grammaires d'interaction.

- Dans un deuxième temps, nous présentons l'algorithme de type Earley issu d'un travail préliminaire de Joseph Le Roux, doctorant dans l'équipe de Calligramme. Cet algorithme s'intéresse à une version primitive des grammaires d'interaction et est à la base de l'algorithme que nous développons ici.

- Dans un troisième temps, nous présentons un algorithme pour les grammaires d'interaction dans leur version complète. Ce travail a été fait conjointement avec Joseph Le Roux. Une longue phase d'implémentation dans LEOPAR a suivi.

– Enfin, nous présenterons les premiers résultats de cet analyseur dans LEOPAR par rapport aux stratégies déjà existantes et nous conclurons sur les perspectives envisageables pour améliorer cet algorithme.

Chapitre 1

Présentation des outils théoriques

1.1 Algorithme de Earley

L'algorithme de Earley [Ear70] est un algorithme d'analyse syntaxique pour les langages hors-contexte. Cet algorithme est une technique d'analyse principalement descendant qui balaie un énoncé de gauche à droite.
L'analyse se fait dans le pire des cas, par rapport à la longueur de l'énoncé à analyser, en temps cubique dans le cas général et en temps quadratique pour les grammaires non ambiguës.

1.1.1 Intuition

Nous présentons ici l'algorithme d'une manière légèrement différente de celle que J. Earley avait présentée en 1970 [Ear70] :
En partant du symbole initial de la grammaire, l'analyseur descend dans l'arbre d'analyse en explorant toutes les possibilités de gauche à droite (prédiction). A chaque fois que l'analyseur rencontre le mot attendu de l'énoncé, l'analyse avance d'un pas (balayage). Quand un sous-arbre d'analyse est analysé avec succès, l'analyseur remonte dans l'arbre d'analyse et avance d'un pas (complétion).
Pour mieux comprendre ces principes, il faut tout d'abord comprendre la notion de règles pointées.

1.1.2 Règles pointées et items manipulés

Soit G une grammaire hors-contexte telle que $G = < N, T, S, R >$ avec :
- N l'ensemble (fini et non vide) des symboles non-terminaux
- T l'ensemble (fini et non vide) des symboles terminaux
- S le symbole initial ($S \in N$)
- R un ensemble fini de règles de production de la forme $Y \longrightarrow \alpha$ où :
  - Y est un non-terminal
  - α est une suite de terminaux et de non-terminaux[1]

Soit $w_1 \dots w_n$ un énoncé.

Etant donné une règle de production $X \longrightarrow \alpha\beta$, la règle pointée $X \longrightarrow \alpha \bullet \beta$ représente la situation où α a déjà été analysée et où la séquence β est attendue.

Les items des règles d'inférence sont définis par :
- une règle pointée représentant un situation dans l'arbre l'analyse
- un couple d'entier i et j représentant les indices de la portion de l'énoncé analysé dans la règle pointée

[1] Par convention nous noterons les terminaux par des minuscules, les non-terminaux par des majuscules et les suites de terminaux et de non-terminaux par des lettres grecques.

Plus formellement, l'item $< A \longrightarrow \alpha \bullet \beta, (i, j) >$ caractérise les trois propriétés suivantes :
- $A \longrightarrow \alpha\beta$ appartient à R.
- $S \overset{*}{\Rightarrow} w_1 \dots w_i A\gamma$ où $\gamma \in (N|T)^*$
- $\alpha \overset{*}{\Rightarrow} w_{i+1} \dots w_j$

1.1.3 Les règles d'inférence

La règle axiome

L'algorithme étant descendant, l'analyse commence par initialiser les règles de production dont la tête est le symbole initial. L'analyse se faisant de gauche à droite, i et j sont initialisés à 0.

$$\frac{}{< S \longrightarrow \bullet\alpha, (0,0) >}, \text{ pour tout élément de } R \text{ ayant pour tête } S.$$

La règle de prédiction

Si la règle pointée d'un item I attend à l'indice i l'analyse d'un non-terminal B, alors l'analyseur prédit toutes les analyses possibles de ce non-terminal en produisant, pour toutes les règles de R ayant pour tête B, un item attendant l'analyse du corps de la règle correspondante.

$$\frac{< A \longrightarrow \alpha \bullet B\beta, (i, j) >}{< B \longrightarrow \bullet\gamma, (j, j) >}, \text{ pour toute règle } B \to \gamma \text{ de R.}$$

La règle de balayage

Si la règle pointée d'un item I attend à l'indice j l'analyse d'un terminal w, et que w est lu à l'indice j de l'énoncé, alors l'analyse avance d'un pas en produisant un nouvel item similaire à I où w est analysé et j est incrémenté.

$$\frac{< A \longrightarrow \alpha \bullet w\beta, (i, j) >}{< A \longrightarrow \alpha w \bullet \beta, (i, j+1) >}, \text{ si } w = w_j$$

La règle de complétion

Si la règle pointée d'un item I attend à l'indice j l'analyse d'un non-terminal B, et que B est la tête de la règle d'un item complètement analysé (le point est tout à droite de la règle pointée) entre j et k dans l'énoncé, alors l'analyseur produit un item similaire à I où B est analysé et où la portion de l'énoncé analysée est mise à jour.

$$\frac{< A \longrightarrow \alpha \bullet B\beta, (i, j) > \quad < B \longrightarrow \gamma\bullet, (j, k) >}{< A \longrightarrow \alpha B \bullet \beta, (i, k) >}$$

Résultat de l'analyse

Un énoncé est correct si l'analyse produit un item $< S \longrightarrow \alpha\bullet, (0, n) >$ où n est la longueur de l'énoncé.

1.1.4 Une analyse tabulaire

Afin que l'analyse soit efficace, il est nécessaire de ne pas analyser plusieurs fois les mêmes sous-arbres. Pour cela, les items produits lors de l'analyse sont stockés dans un tableau et les items à traiter sont ordonnancés dans un agenda. L'analyse se déroule de la façon suivante :

1. Initialiser l'agenda avec les items initiaux (produits par l'axiome).
2. Tant que l'agenda n'est pas vide :
   - Sélectionner un élément de l'agenda et le retirer.

- Produire tous les items possibles avec cet item et les règles d'inférence (en prenant un item dans le tableau pour la règle de complétion).
- Pour chaque item produit, s'il n'est pas déjà dans le tableau alors le rajouter dans l'agenda et le tableau.

1.1.5 Exemple de déroulement de l'algorithme

Nous exposons ici un exemple de l'analyse d'un énoncé avec une grammaire hors-contexte d'un petit fragment du français.

Soit $J =< N, T, S, R >$ la grammaire hors-contexte suivante :
- $N = \{S, V, SV, SN, N, Det\}$, les non-terminaux
- $T = \{Jean, mange, une, pomme\}$, les terminaux
- S le symbole initial
- et R l'ensemble des règles de production suivantes :

$$\begin{array}{lcl} S & \longrightarrow & SN\ SV \\ SV & \longrightarrow & V\ SN \\ SN & \longrightarrow & Det\ N \\ SN & \longrightarrow & Jean \\ V & \longrightarrow & mange \\ Det & \longrightarrow & une \\ N & \longrightarrow & pomme \end{array}$$

Et W l'énoncé suivant : ${}_0Jean_1mange_2une_3pomme_4$

L'analyse de Earley de W avec la grammaire J produit les items suivants :

0	$< S$	$\longrightarrow \bullet\ SN\ SV$	$, (0,0) >$	axiome
1	$< SN$	$\longrightarrow \bullet\ Det\ N$	$, (0,0) >$	prédiction 0
2	$< SN$	$\longrightarrow \bullet\ Jean$	$, (0,0) >$	prédiction 0
3	$< Det$	$\longrightarrow \bullet\ une$	$, (0,0) >$	prédiction 1
4	$< SN$	$\longrightarrow Jean\ \bullet$	$, (0,1) >$	balayage 2
5	$< S$	$\longrightarrow SN \bullet\ SV$	$, (0,1) >$	complétion 0 4
6	$< SV$	$\longrightarrow \bullet\ V\ SN$	$, (1,1) >$	prédiction 5
7	$< V$	$\longrightarrow \bullet\ mange$	$, (1,1) >$	prédiction 6
8	$< V$	$\longrightarrow mange\ \bullet$	$, (1,2) >$	balayage 7
9	$< SV$	$\longrightarrow V \bullet\ SN$	$, (1,2) >$	complétion 6 8
10	$< SN$	$\longrightarrow \bullet\ Det\ N$	$, (2,2) >$	prédiction 9
11	$< SN$	$\longrightarrow \bullet\ Jean$	$, (2,2) >$	prédiction 9
12	$< Det$	$\longrightarrow \bullet\ une$	$, (2,2) >$	prédiction 10
13	$< Det$	$\longrightarrow une\ \bullet$	$, (2,3) >$	balayage 12
14	$< SN$	$\longrightarrow Det \bullet\ N$	$, (2,3) >$	complétion 10 13
15	$< N$	$\longrightarrow \bullet\ pomme$	$, (3,3) >$	prédiction 14
16	$< N$	$\longrightarrow pomme\ \bullet$	$, (3,4) >$	balayage 15
17	$< SN$	$\longrightarrow Det\ N\ \bullet$	$, (2,4) >$	complétion 14 16
18	$< SV$	$\longrightarrow V\ SN\ \bullet$	$, (1,4) >$	complétion 6 17
19	$< S$	$\longrightarrow SN\ SV\ \bullet$	$, (0,4) >$	complétion 5 18

Le dernier item crée nous assure de la validité de l'énoncé dans la grammaire. Il est aussi possible de déduire à partir de ces items les arbres de dérivation des analyses correctes.

1.2 Les grammaires d'interaction

1.2.1 Introduction

À l'instar des grammaires d'arbres adjoints [JLT75], les grammaires d'interaction [Per02] sont un formalisme grammatical s'appuyant sur la notion de *description d'arbres*. Cette notion a été introduite par J. Rogers et K. Vijay-Shanker en 1992 [RVS92] et ce dernier l'a reprise pour représenter l'opération d'adjonction des grammaires d'arbres adjoints [VS92].
L'intérêt est de remplacer la manipulation d'arbres syntaxiques complètement spécifiés par la manipulation de spécifications partielles de ces arbres.
Une description d'arbres est définie par un ensemble de nœuds et de relations d'ascendance, de parenté et de précédence entre ces nœuds. Les nœuds représentent des syntagmes (éventuellement vides) et les relations expriment les dépendances entre ces syntagmes. Les propriétés morpho-syntaxiques de ce syntagmes sont décrites par des structures de traits.
Cette approche est bien adaptée à la flexibilité et à l'ambiguïté des langues naturelles.

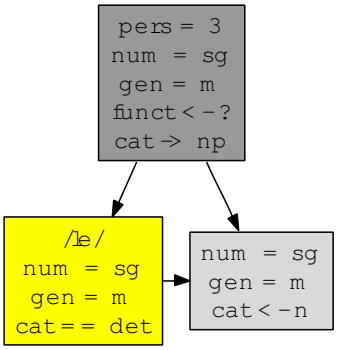

FIG. 1.1 – Description d'arbres élémentaire du déterminant "le"

Cependant, l'analyse syntaxique fondée sur des descriptions d'arbres peut être très coûteuse [KNT01]. En effet, dans cette approche, l'analyse syntaxique consiste à chercher des modèles de descriptions d'arbres sous forme d'arbres syntaxiques complètement spécifiés. Ce processus est hautement indéterministe.
Dans les formalismes réalistes fondés sur les descriptions d'arbres, cet indéterminisme est limité en contraignant la syntaxe des descriptions et le mécanisme de composition syntaxique.
L'originalité des grammaires d'interaction se trouve dans le mécanisme de composition syntaxique régi par les *polarités*. Cette notion de polarité est liée à la dualité besoins-ressources qui est à la base des grammaires catégorielles [Ré00] : certaines ressources munies de polarités négatives sont attendues alors que d'autres, munies de polarités positives, sont disponibles si bien que les premières vont chercher à rencontrer les secondes , c'est le principe de *neutralisation des polarités opposées*.

1.2.2 Les descriptions d'arbres polarisées

Les grammaires d'interaction ont pour objets syntaxiques de base les descriptions d'arbres polarisées [Per02].

Les descriptions d'arbres polarisées sont construites sur une signature $(\mathcal{N}, \mathcal{T})$ où $\mathcal{N}$ est un ensemble de nœuds syntaxiques et $\mathcal{T}$ une base de traits. $\mathcal{T}$ est un ensemble fini de couples (T, D_T) où pour un nom de trait T est associé l'ensemble fini D_T des valeurs atomiques qu'il peut prendre.

Les grammaires d'interaction peuvent être vues comme un raffinement des grammaires catégorielles en ce sens que la notion de polarité est descendue du niveau des syntagmes au niveau des traits grammaticaux qui le décrivent.
Ainsi à chaque trait, en plus d'une valeur, est associée un polarité pour indiquer éventuellement si c'est une ressource consommable ou un besoin. Une polarité peut être positive ($\rightarrow$), négative ($\leftarrow$) ou neutre ($=$). Il est nécessaire de distinguer les nœuds initialement neutres des nœuds issus d'une neutralisation. Dans ce dernier cas on note la polarité $\leftrightarrow$. La polarité $\perp$ indique l'échec de l'unification de deux polarités.

	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftarrow$	$\perp$	$\leftrightarrow$	$\leftarrow$	$\perp$
$\rightarrow$	$\leftrightarrow$	$\perp$	$\rightarrow$	$\perp$
$=$	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftrightarrow$	$\perp$	$\perp$	$\leftrightarrow$	$\perp$

TAB. 1.1 – Résultats de l'unification de deux polarités

Si nous faisons référence à la figure 1.1, nous constatons que la description d'arbres du déterminant "le" demande un nom et une fonction syntaxique et fournit un syntagme nominal.

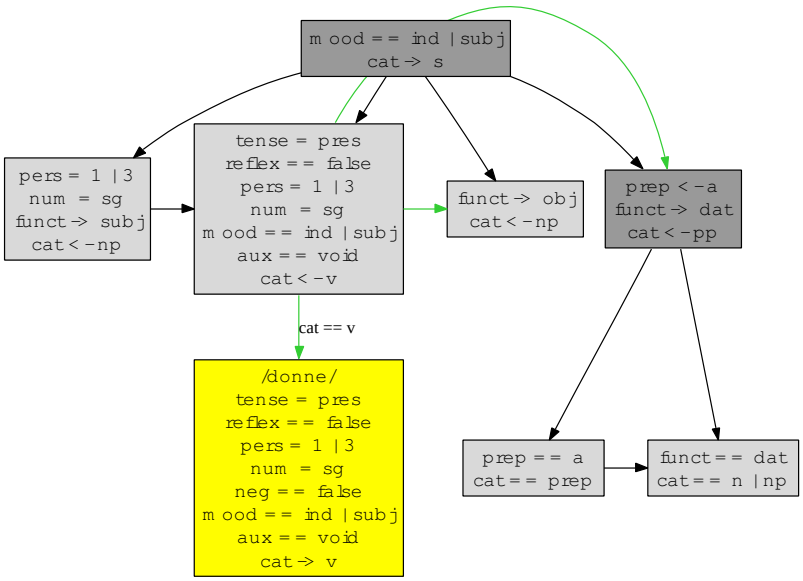

FIG. 1.2 – Description d'arbres élémentaire du verbe "donne" dans la phrase *qqn donne qqc à qqn*

Une description d'arbres polarisée sur une signature $(\mathcal{N}, \mathcal{T})$ est définie par un ensemble de nœuds $\mathcal{N}$ (chacun étiqueté par une structure de traits polarisés appartenant à $\mathcal{T}$) mais aussi par un ensemble de relations entre ces nœuds. Ces relations peuvent être de quatre types :

Relations de dominance immédiate $N_1 > N_2$ signifie que le syntagme N_2 est un constituant immédiat de N_1, ce qu'on représente graphiquement par un flèche de haut en bas. Dans la figure 1.2, le syntagme s se décompose en quatre constituants immédiats : $subj$, v, np, pp.

Relations de dominance sous-spécifiée $N_1 \overset{*}{>} N_2$ signifie que le syntagme N_2 est inclus dans N_1 à une profondeur indéterminée (éventuellement N_1 s'identifie à N_2), ce qu'on représente par une flèche grisée de haut en bas. Cette relation permet d'exprimer à la fois une dépendance syntaxique non bornée et

la possibilité d'appliquer des modifieurs à un syntagme. Dans les deux cas, on peut contraindre la relation qui devient $N_1 \overset{*}{>} [f_1 = v_1, \ldots, f_n = v_n]M$, ce qui signifie que tout syntagme qui est inclus dans N_1 et qui contient N_2 doit avoir sa structure de trait compatible avec $[f_1 = v_1, \ldots, f_n = v_n]$. Dans l'exemple de la figure 1.2, la relation de domination sous-spécifiée entre les deux nœuds v signifie la possibilité d'appliquer un modifieur de verbe à *donne*.

Relations de précédence immédiate $N_1 \prec N_2$ signifie que le syntagme N_1 précède immédiatement le syntagme N_2 dans l'ordre linéaire des mots de la phrase, ce qu'on représente graphiquement par une flèche horizontale. Dans l'exemple de la figure 1.2, le syntagme sujet précède immédiatement le syntagme verbal.

Relations de précédence sous-spécifiée $N_1 \overset{*}{\prec} N_2$ signifie que le syntagme N_1 précède le syntagme N_2 dans l'ordre linéaire des mots de la phrase, ce qu'on représente graphiquement par une flèche grisée horizontale. Dans l'exemple de la figure 1.2, le syntagme verbal précède le syntagme nominal et le syntagme prépositionnel.

1.2.3 Une grammaire lexicalisée

Les grammaires d'interaction sont lexicalisées. C'est-à-dire que chaque description d'arbres élémentaire est distinguée par son nœud ancre qui exprime la relation entre la description et le lexique. Ainsi, chaque item lexical est associé à un ensemble de descriptions d'arbres élémentaires.
Si nous prenons à nouveau l'exemple de la figure 1.1, le nœud ancre correspond au nœud de catégorie déterminant.

1.2.4 Construction de modèles de description d'arbres

La composition syntaxique de deux descriptions d'arbres est un processus de *neutralisation de nœuds opposés* dans lequel l'opération fondamentale peut être vue comme une fusion particulière de deux nœuds dans le cas où ceux-ci sont porteurs de traits opposés. Cette neutralisation ne se passant pas seulement entre la racine d'un arbre et une feuille d'une autre implique que les arbres ne sont pas seulement accrochés les uns aux autres mais qu'ils peuvent être aussi superposés partiellement.
Analyser une description d'arbres consiste à itérer l'opération de neutralisation des traits opposés pour spécifier progressivement la description initiale. Cela correspond à la recherche d'un modèle de description d'arbres :
Un modèle d'une description d'arbres D est une couple formé d'un arbre A et d'une interprétation I :
A est un arbre ordonné et ses nœuds sont étiquetés par des structures de traits.
I est une fonction d'interprétation de l'ensemble $|D|$ des nœuds de D dans l'ensemble $|A|$ des nœuds de A qui vérifient les conditions suivantes :

- Pour tout trait T d'un nœud N de D, $I(N)$ comprend un trait de même nom que T avec une valeur figurant dans la disjonction du trait.
- Si $N_1 > N_2$ alors $I(N_1)$ et le père de $I(N_2)$ dans A.
- Si $N_1 \overset{*}{>} [f_1 = v_1, \ldots, f_n = v_n]N_2$ alors $I(N_1)$ domine $I(N_2)$ (éventuellement $I(N_1) = I(N_2)$) dans A et pour tout nœud N' de A qui est un descendant de $I(N_1)$ et un ascendant de $I(N_2)$ au sens large, la structure de traits de $I(N')$ compatible avec $[f_1 = v_1, \ldots, f_n = v_n]$.
- Si $N_1 \prec N_2$, alors $I(N_1)$ précède immédiatement $I(N_2)$ dans A.
- Si $N_1 \overset{*}{\prec} N_2$, alors $I(N_1)$ précède $I(N_2)$ dans A.

L'analyse réussit si elle s'achève par un arbre complètement spécifié sans relation large où tous les traits ont été neutralisés. Cela correspond à un modèle de description d'arbres minimal et neutre.
Un modèle minimal et neutre A d'une description d'arbres D est un modèle satisfaisant les propriétés suivantes :

- *minimalité* : Si N_1 est le père d'un nœud N_2 dans A, il existe au moins une relation $N'_1 > N'_2$ dans D telle que $N_1 = I(N'_1)$ et $N_2 = I(N'_2)$.

– *neutralité* : Pour tout nœud N de A et pour tout trait F présent dans l'étiquette de N, soit il n'existe pas de nœud N' avec un trait F dans D à polarité positive ou négative tel que $I(N') = N$, soit il existe deux nœuds N' et N'' avec un trait F positive dans l'une et négative dans l'autre tels que $I(N') = N$ et $I(N'') = N$

Par exemple, la figure 1.4 représente le seul modèle valide de la description d'arbres illustrée à la figure 1.3.

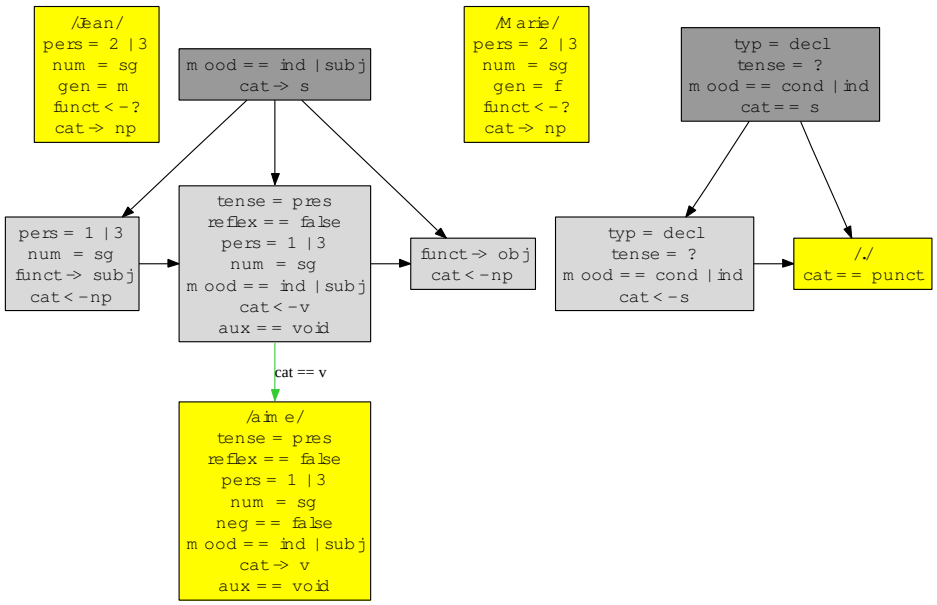

FIG. 1.3 – Description d'arbres de l'énoncé "Jean aime Marie."

Une description plus détaillée de ce processus de construction peut être trouvée dans l'article "Analyse syntaxique électrostatique" de Guillaume Bonfante, Bruno Guillaume et Guy Perrier [BGP03].

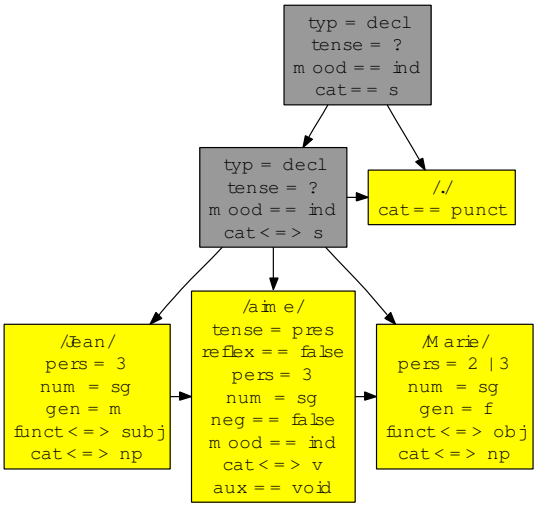

FIG. 1.4 – Modèle valide de la description d'arbres de l'énoncé "Jean aime Marie."

Chapitre 2

Un algorithme de Earley pour les grammaires d'interaction primitives

Les grammaires d'interaction et les grammaires d'arbres adjoints [JLT75] ont toutes les deux comme objets syntaxiques de base les *descriptions d'arbres*. C'est en partie la raison pour laquelle l'algorithme que nous étudions ici reprend en partie les travaux de Y. Schabes et A. Joshi qui proposent un algorithme de Earley pour les grammaires d'arbres adjoints [SJ88]. Cependant il existe une différence majeure entre les grammaires d'arbres adjoints et les grammaires d'interaction : tandis que les premières reposent sur deux opérations de composition syntaxiques (la substitution et l'adjonction d'arbres), ces dernières ne reposent que sur une opération plus générale : la superposition d'arbres.
Cet algorithme pour une version simplifiée des grammaires d'interaction est issu du travail de Joseph Le Roux, doctorant dans le projet Calligramme.

2.1 Les grammaires d'interaction primitives

Les grammaires d'interaction primitives présentent deux différences importantes avec les grammaires d'interaction :

- Les polarités ne sont plus portées par les traits mais par le nœud.
- Il existe deux types de nœuds, les nœuds étiquetés par des non-terminaux et les nœuds étiquetés par des terminaux. Seuls les nœuds étiquetés par des non-terminaux portent une polarité et celle-ci est soit positive, soit négative.

On peut ainsi définir une grammaire d'interaction primitive par un quadruplet $G =< N, T, \mathcal{D}, S >$ où N est l'ensemble des symboles non-terminaux, T l'ensemble des symboles terminaux ($N \cap T = \emptyset$), $\mathcal{D}$ l'ensemble des descriptions d'arbres élémentaires construits sur T, et S ($S \in N$) le symbole initial.

2.1.1 Les descriptions d'arbres polarisés élémentaires

Une description d'arbres polarisés élémentaire est définie par un ensemble de nœuds $\mathcal{N}$ et d'un ensemble de relations $\mathcal{R}$ entre les nœuds. Les nœuds d'une description d'arbres élémentaire sont étiquetés par les terminaux et les non-terminaux de la grammaire. A l'instar des grammaires d'interaction, les grammaires d'interaction primitives sont fortement lexicalisées. Ainsi, pour chaque description élémentaire de $\mathcal{D}$, il y a exactement un nœud de $\mathcal{N}$ qui appartienne à T, ce dernier étant une feuille de la description d'arbres. Tous les autres nœuds de $\mathcal{N}$ sont des nœuds non-terminaux de la grammaire et sont polarisés positivement ou négativement.
Les relations entre les nœuds des descriptions d'arbres dans les grammaires d'interaction sont équivalentes à celles des grammaires complètes à la différence qu'il n'est plus possible de contraindre une relation de dominance sous-spécifiée. On les définit comme suit :

- $N_1 > N_2$ si N_2 est un constituant immédiat de N_1
- $N_1 \overset{*}{>} N_2$ si N_2 est inclus dans N_1 à une profondeur indéterminée (N_1 pouvant s'identifier à N_2)

– $N_1 \prec N_2$ si N_1 précède immédiatement N_2 dans l'ordre linéaire des mots de la phrase
– $N_1 \overset{*}{\prec} N_2$ si N_1 précède N_2 dans l'ordre linéaire des mots de la phrase

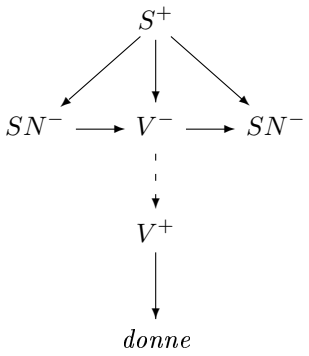

FIG. 2.1 – Description d'arbres élémentaire du verbe "donne"

2.1.2 Construction de modèles de descriptions d'arbres

Le principe de composition syntaxique est analogue à celui employé dans les grammaires d'interaction complètes, il s'agit de superposer des descriptions d'arbres qui présentent des nœuds de polarités opposées. Cependant, ce procédé est un peu plus restrictif dans les grammaires d'interaction primitives car seuls les nœuds étiquetés par le même non-terminal et de polarités opposées peuvent se neutraliser. Ainsi, les nœuds neutralisés et les nœuds étiquetés par des terminaux ne peuvent pas être les opérandes d'une telle opération.
Analyser un énoncé consiste à réitérer le procédé de superposition pour construire pas à pas une description d'arbres complètement spécifiée. Cela correspond à rechercher un modèle minimal et neutre de $\mathcal{D}$.
Un modèle minimal et neutre de G est un couple (A, I) où A est un arbre ordonné et I est une fonction surjective d'interprétation de l'ensemble des nœuds de $\mathcal{D}$ vers l'ensemble des nœuds de A qui vérifie les propriétés suivantes :

– Si $N_1 > N_2$ alors $I(N_1)$ et le père de $I(N_2)$ dans A.
– Si $N_1 \overset{*}{>} N_2$ alors $I(N_1)$ domine $I(N_2)$.
– Si $N_1 \prec N_2$, alors $I(N_1)$ précède immédiatement $I(N_2)$ dans A.
– Si N_1 est le père d'un nœud N_2 dans A, il existe au moins une relation $N_1' > N_2'$ dans $\mathcal{D}$ telle que $N_1 = I(N_1')$ et $N_2 = I(N_2')$.
– Si N est un non-terminal, alors il existe deux nœuds N' et N'' appartenant à $\mathcal{D}$ de polarités opposées et étiquetés par le même non-terminal tel que $I(N') = N$ et $I(N'') = N$.
– Si N est un terminal, alors il existe un unique nœud N' dans $\mathcal{D}$ qui est étiqueté par le même non-terminal que N tel que $I(N') = N$.

2.2 L'analyseur de Earley

2.2.1 Intuition

Comme un algorithme de Earley classique, l'algorithme construit l'arbre d'analyse de haut en bas et de gauche à droite en partant de l'axiome de la grammaire. A chaque étape de la descente, l'analyseur neutralise un nœud du modèle en construction avec un nœud racine des descriptions d'arbres visibles (prédiction). A chaque fois que l'analyseur rencontre le mot attendu de l'énoncé, l'analyse avance d'un pas (balayage). Quand un sous-arbre est analysé avec succès, l'analyseur remonte dans l'arbre d'analyse et avance d'un pas (complétion).

2.2.2 Les items manipulés

Lors de l'analyse, il est nécessaire de gérer spécifiquement les dominations larges des descriptions d'arbres. En effet, ces sous-arbres peuvent être composés à différents étages de l'analyse. Nous introduisons pour cela un triplet (S, U, D) qui va se rappeler des sous-arbres qui peuvent être composés et contraindre leur rattachement.

Les items manipulés sont de la forme $< A_{C_A} \longrightarrow \alpha \bullet B_{C_B}\beta, (i,j), (S,U,D) >$ où :
- C_A et C_B sont les contextes associés aux nœuds du modèle et sont notés par le terminal ou le non-terminal qui l'étiquette. Les contextes sont :
  - Pour les terminaux, le nœud interprété.
  - Pour les non-terminaux, un couple de nœuds interprétés (P,N) tel que P (de polarité positive) et N (de polarité négative) sont étiquetées par le même non-terminal. Ces contextes peuvent être sous-spécifiés ou partiellement sous-spécifiés. Quand un nœud n'est pas spécifié, on le note _.
- $A_{C_A} \longrightarrow \alpha \bullet B_{C_B}\beta$ est une règle pointée, la sémantique de cette règle est la suivante :
  - C_A est complètement spécifié et les nœuds qui le décrivent sont de polarités opposées et interprétés par A.
  - Si B est un terminal, alors C_B est décrit par un nœud terminal qui est interprété par B.
  - Si B est un non-terminal alors tout nœud défini qui décrit C_B est un nœud non-terminal et est interprété par B. Si C_B est complètement spécifié, alors les nœuds qui le décrivent sont de polarités opposées.
  - A est le père de tous les nœuds du corps de la règle pointée et ses fils sont ordonnés de gauche à droite dans le modèle en construction.
- i et j représentent les indices de la portion de l'énoncé analysé dans la règle pointée.
- le triplet (S, U, D) représente la situation des ressources de la grammaire à l'étape de l'analyse, il est défini respectivement par :
  - les descriptions d'arbres décrochées à l'étage précédent de l'analyse (Stop)
  - les descriptions d'arbres déjà utilisées lors de l'analyse (Up)
  - les descriptions d'arbres décrochées aux étages précédents de l'algorithme qui n'ont pas été utilisées par l'analyseur lors du processus de neutralisation (Down)

2.2.3 Règles d'inférence

La règle axiome

La règle axiome permet de démarrer l'analyse, S étant le symbole initial de la grammaire et $\mathcal{D}$ l'ensemble des descriptions d'arbres de la grammaire.

$$\frac{}{< \top \longrightarrow \bullet S_{(\_,\_)}, (0,0), (\mathcal{D}, \emptyset, \emptyset) >}$$

La règle de prédiction

A partir d'un item $< A \longrightarrow \alpha \bullet B_{(B_1,B_2)}\beta, (i,j), (S,U,D) >$, on génère un nouvel item qui contient une règle engendrée par les descriptions d'arbres de racines B_i en ordonnant complètement leurs fils. Cet étape peut générer beaucoup d'items si les fils de B_1 et B_2 ne sont pas ordonnés.
Comme le contexte de la tête de la règle pointée est complètement spécifié dans l'item produit, il est nécessaire de spécifier les nœuds B_1 et B_2 par un nœud racine des descriptions d'arbres visibles. Un nœud est dit *visible* :
- s'il est l'un des fils directs d'un des nœuds qui décrivent A.
- s'il est racine d'une description d'arbres élémentaires qui n'a pas encore été utilisée dans le processus de neutralisation.

Une fois le contexte de la tête de la règle pointée complètement spécifié, il faut mettre à jour la situation des ressources de la grammaire notamment en "décrochant" les fils larges non-utilisés par les nœuds du contexte de la tête dans le processus de neutralisation.

$$\frac{< A \longrightarrow \alpha \bullet B_{(B_1,B_2)}\beta, (i,j), (S,U,D) >}{< B_{(B'_1,B'_2)} \longrightarrow \bullet\gamma, (j,j), (S',U',D') >}, \text{ avec}$$

- $B'_i = B_i$ si B_i est défini, sinon B'_i est la racine d'une description d'arbres visible.
- γ est un ordre possible pour les fils directs de B'_1 et de B'_2.
- S' est l'ensemble des fils larges de B'_1 et B'_2. Si $B'_1 \overset{*}{>} B'_2$ (respectivement $B'_2 \overset{*}{>} B'_1$), alors S' est l'ensemble des fils larges de B'_1 et B'_2 privé de B'_2 (respectivement B'_1).
- U' est l'union de U et des racines des descriptions d'arbres visibles à la phase de prédiction qui forment le contexte de B (ie, $U' = U \cup \{N \in \mathcal{N} \mid N$ appartient à D et à $\{B'_1, B'_2\}\} \cup \{N \in \mathcal{N} \mid N$ appartient à S et à $\{B'_1, B'_2\}\}$).
- D' est l'ensemble des racines des descriptions d'arbres décrochées lors des phases de prédiction qui n'ont pas été utilisées lors des neutralisations précédentes (ie, $D' = D \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, B'_2\}\} \cup S \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, B'_2\}\}$).

La règle de balayage

Si on analyse le terminal correspondant au mot à lire dans l'énoncé, alors on génère l'item où le terminal a été analysé avec succès.

$$\frac{< A \longrightarrow \alpha \bullet w_w\beta, (i,j), (S,U,D) >}{< A \longrightarrow \alpha w_w \bullet \beta, (i,j+1), (S,U,D) >}, \text{ si } w = w_j$$

La règle de complétion

Cette règle permet de remonter dans l'arbre d'analyse en construction en validant un sous arbre d'analyse.
Pour valider le sous-arbre d'analyse, il faut vérifier que les descriptions d'arbres "décrochées" lors de la phase de prédiction ont bien été "raccrochées" quelque part dans le sous-arbre d'analyse.

$$\frac{< A \longrightarrow \alpha \bullet B_{(B_1,B_2)}\beta, (i,j), (S,U,D) > \quad < B_{(B'_1,B'_2)} \longrightarrow \gamma\bullet, (j,k), (S',U',D') >}{< A \longrightarrow \alpha B_{(B_1,B_2)} \bullet \beta, (i,k), (S'',U'',D'') >}, \text{ avec}$$

- $B'_i = B_i$ si B_i est défini
- $S' = \emptyset$
- S'' est l'ensemble des racines des descriptions d'arbres décrochées à l'étage précédent qui n'ont pas été encore utilisées dans l'analyse (ie, $S'' = S \setminus \{N \in \mathcal{N} \mid N$ appartient à S et à $U'\}$).
- $U'' = U' \setminus \{N \in \mathcal{N} \mid N$ appartient à $S\}$
- $D'' = D'$

Résultat de l'analyse

Un énoncé est correct si l'analyse produit un item $< \top \longrightarrow S_{(\_,\_)}\bullet, (0,n), (\emptyset,U,\emptyset) >$ où n est la longueur de l'énoncé.

2.2.4 Exemple de déroulement de l'algorithme

Soit $J =< N, T, D, S >$ avec :
- $N = \{S, SN, V\}$, les symboles non-terminaux
- $T = \{Jean, le, voit\}$, les symboles terminaux
- S, le symbole initial
- D, l'ensemble des descriptions d'arbres élémentaires représenté à la figure 2.2.

Et W l'énoncé suivant : ${}_0Jean_1le_2voit_3$

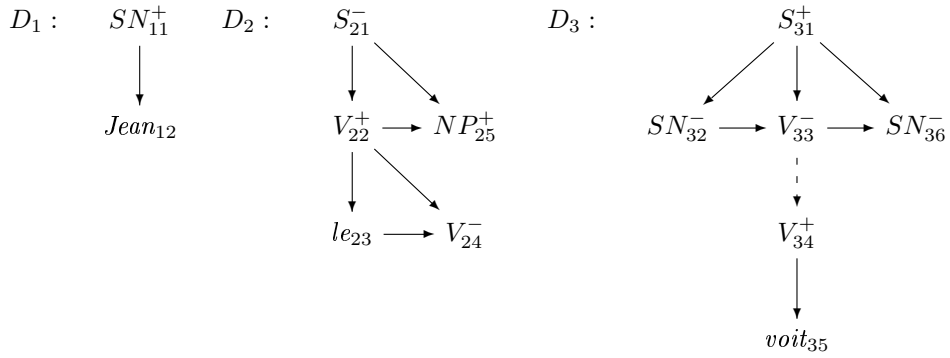

FIG. 2.2 – Ensemble des description d'arbres élémentaires de la grammaire J

L'analyse de Earley de W avec la grammaire J produit les items suivants :

0	$< \top \longrightarrow \bullet\ S_{(\_,\_)}\ , (0,0), (\{11,21,31\},\emptyset,\emptyset) >$	A
1	$< S_{(31,21)} \longrightarrow \bullet\ V_{(22,\_)}\ SN_{(25,\_)}\ SN_{(\_,32)}\ V_{(\_,33)}\ SN_{(\_,35)}\ , (0,0), (\emptyset,\{21,31\},\{11\}) >$	P 0
2	$< S_{(31,21)} \longrightarrow \bullet\ V_{(22,\_)}\ SN_{(25,32)}\ V_{(\_,33)}\ SN_{(\_,35)}\ , (0,0), (\emptyset,\{21,31\},\{11\}) >$	P 0
3	$< S_{(31,21)} \longrightarrow \bullet\ SN_{(\_,32)}\ V_{(22,33)}\ SN_{(25,35)}\ , (0,0), (\emptyset,\{21,31\},\{11\}) >$	P 0
4	$< S_{(31,21)} \longrightarrow \bullet\ SN_{(\_,32)}\ V_{(\_,33)}\ SN_{(\_,35)}\ V_{(22,\_)}\ SN_{(25,\_)}\ , (0,0), (\emptyset,\{21,31\},\{11\}) >$	P 0
5	$< SN_{(11,32)} \longrightarrow \bullet\ Jean_{12}\ , (0,0), (\emptyset,\{21,31,11\},\emptyset) >$	P 3/4
6	$< SN_{(11,32)} \longrightarrow Jean_{12}\ \bullet\ , (0,1), (\emptyset,\{21,31,11\},\emptyset) >$	B 5
7	$< S_{(31,21)} \longrightarrow SN_{(\_,32)}\ \bullet\ V_{(22,33)}\ SN_{(25,35)}\ , (0,1), (\emptyset,\{21,31,11\},\emptyset) >$	C 3 6
8	$< S_{(31,21)} \longrightarrow SN_{(\_,32)}\ \bullet\ V_{(\_,33)}\ SN_{(\_,35)}\ V_{(22,\_)}\ SN_{(25,\_)}\ , (0,1), (\emptyset,\{21,31,11\},\emptyset) >$	C 4 6
9	$< V_{(22,33)} \longrightarrow \bullet\ le_{23}\ V_{(\_,24)}\ , (1,1), (\{34\},\{21,31,11\},\emptyset) >$	P 7
10	$< V_{(34,33)} \longrightarrow \bullet\ voit_{35}\ , (1,1), (\emptyset,\{21,31,11,34\},\emptyset) >$	P 7
11	$< V_{(22,33)} \longrightarrow le_{23}\ \bullet\ V_{(\_,24)}, (1,2), (\{34\},\{21,31,11\},\emptyset) >$	B 9
12	$< V_{(24,34)} \longrightarrow \bullet\ voit_{35}\ , (2,2), (\emptyset,\{21,31,11,34\},\emptyset) >$	P 11
13	$< V_{(24,34)} \longrightarrow voit_{35}\ \bullet\ , (2,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	B 12
14	$< V_{(22,33)} \longrightarrow le_{23}\ V_{(\_,24)}\ \bullet\ , (1,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	C 11 13
15	$< S_{(31,21)} \longrightarrow SN_{(\_,32)}\ V_{(22,33)}\ \bullet\ SN_{(25,35)}\ , (0,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	C 7 14
16	$< SN_{(25,35)} \longrightarrow \bullet\ , (3,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	P 16
17	$< S_{(31,21)} \longrightarrow SN_{(\_,32)}\ V_{(22,33)}\ SN_{(25,35)}\ \bullet\ , (0,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	C 7 14
18	$< \top \longrightarrow S_{(\_,\_)}\ \bullet\ , (0,3), (\emptyset,\{21,31,11,34\},\emptyset) >$	C 0 17

Chapitre 3

Un algorithme de Earley pour les grammaires d'interaction

3.1 Introduction

L'algorithme présenté ici reprend en très grande partie l'algorithme développé pour les grammaires d'interaction primitives. Il s'agit de construire les modèles minimaux et neutres d'une description d'arbres en partant des racines des ses descriptions élémentaires et en descendant étape par étape dans l'arbre d'analyse par la neutralisation des nœuds en cours d'analyse et la sous-analyse de tous les modèles possibles du sous-arbre.

Si l'algorithme reste identique dans le principe, il doit cependant être adapté pour analyser les descriptions d'arbres des grammaires d'interaction dans leur version complète. En effet, l'opération de neutralisation est ici différente, il ne s'agit plus de superposer deux nœuds étiquetés par un même non-terminal et de polarité différentes, mais de superposer des nœuds dont l'unification des polarités de l'un des traits qu'ils ont en commun réussit (tableau 1.1) et dont leurs structures de traits associées sont compatibles. Ainsi, un nœud du modèle peut-être issu de la superposition de plusieurs nœuds de la description d'origine, dont certains ne peuvent avoir aucun traits polarisés positivement ou négativement (nœuds neutres).
D'un point de vue computationnel, cela signifie que lors de la prédiction, l'analyseur n'est pas guidé uniquement par les polarités positives ou négatives et doit donc étendre son espace de recherche avec des nœuds qui ne répondent pas à la notion de besoins-ressources, on peut qualifier ces derniers de nœuds optionnels.

Une autre différence provient de la possibilité de superposer des nœuds portant l'information lexical dans les grammaires d'interaction et non dans leur version primitive. En effet, dans les grammaires d'interaction primitives, on différenciait les nœuds qui portaient l'information lexical (dits terminaux) des autres (dits non-terminaux), et seuls les nœuds non-lexicalisés pouvaient être les opérandes d'une telle opération. Il faut donc adapter l'algorithme afin qu'il puisse savoir quand il rencontre une information lexicalisée et essayer de le faire correspondre avec le mot attendu de l'énoncé.

3.2 Les items manipulés

Soit $\mathcal{M} : (A, I)$ un modèle de description d'arbres. Un contexte $\{B_1, \ldots, B_n\}$ est l'image inverse d'un nœud B de A par I (ie, $I(B_i) = B$ pour tous les B_i du contexte).

Les items manipulés sont de la forme $< A_{C_A} \longrightarrow \alpha \bullet B_{C_B}\beta, (i, j), (S, U, D) >$ où :

- C_A et C_B sont les contextes associés aux nœuds du modèle, ils sont étiquetés par la structure de traits issue de l'unification des structures de traits des nœuds du contexte. Cependant, par souci d'espace, nous remplacerons cette structure de traits par un symbole arbitraire quand l'information

porté par la structure n'est pas utile. Un contexte est dit sous-spécifié s'il existe un trait de la structure de traits qui l'étiquette qui ne soit pas de polarité neutre (un trait est de polarité neutre s'il porte une polarité = ou ↔).

- $A_{C_A} \longrightarrow \alpha \bullet B_{C_B}\beta$ est une règle pointée, la sémantique de cette règle est la suivante :
  - C_A est complètement spécifié.
  - A est le père de tous les nœuds du corps de la règle pointée et ses fils sont ordonnés de gauche à droite dans le modèle en construction.
- i et j représentent les indices de la portion de l'énoncé analysé dans la règle pointée.
- le triplet (S, U, D) représente la situation des ressources de la grammaire à l'étape de l'analyse, il est défini respectivement par :
  - les descriptions d'arbres décrochées à l'étage précédent de l'analyse (Stop)
  - les descriptions d'arbres déjà utilisées lors de l'analyse (Up)
  - les descriptions d'arbres décrochées aux étages précédents de l'algorithme qui n'ont pas été utilisées par l'analyseur lors du processus de neutralisation (Down)

3.3 Les règles d'inférences

Soit $\mathcal{D}$ l'ensemble de descriptions d'arbres élémentaires de la grammaire et W l'énoncé à analyser. Nous supposons qu'il existe un trait dans la grammaire qui définisse un nœud initial des modèles de la grammaire.

3.3.1 La règle axiome

La règle axiome permet de démarrer l'analyse. Nous créons pour cela autant d'items qu'il y a de possibilité de nœuds racine du modèle.

$$\frac{}{< \top \longrightarrow \bullet S_N, (0,0), (\mathcal{D}, \emptyset, \emptyset) >}, \text{ pour}$$

– toutes les partitions N des nœuds initiaux de la grammaire qui forment des contextes complètement spécifiés.

3.3.2 La règle de prédiction

A partir d'un item $< A \longrightarrow \alpha \bullet B_{\{B_1,\dots B_n\}}\beta, (i,j), S, U, D >$,on génère un nouvel item qui contient une règle engendrée par les descriptions d'arbres de racines B_i en ordonnant complètement leurs fils. Cet étape peut générer beaucoup d'items si les fils de $\{B'_1, \dots B'_{n+m}\}$ ne sont pas ordonnés.
Comme le contexte de la tête de la règle pointée est complètement spécifié dans l'item produit, il est nécessaire de compléter les nœuds B_i par des nœuds racines des descriptions d'arbres visibles.
De plus, une fois le contexte de la tête de la règle pointée complètement spécifié, il faut mettre à jour la situation des ressources de la grammaire notamment en "décrochant" les fils larges non-utilisés par les nœuds du contexte de la tête dans le processus de neutralisation.

$$\frac{< A \longrightarrow \alpha \bullet B_{\{B_1,\dots B_n\}}\beta, (i,j), S, U, D >}{< B_{\{B'_1,\dots B'_{n+m}\}} \longrightarrow \bullet\gamma, (j,j), (S', U', D') >}, \text{ avec}$$

– $B'_i = B_i$ si $i \leq n$, sinon B'_i est la racine d'une description d'arbres visible.
– Aucun des B'_i n'est une ancre.
– γ est un ordre possible pour les fils directs de $\{B'_1, \dots B'_{n+m}\}$.
– S' est l'ensemble des fils larges de $\{B'_1, \dots B'_{n+m}\}$ privé de tous les nœuds B_j pour lesquels il existe une relation $B_i \overset{*}{>} B_j$ dans D.
– U' est l'union de U et des racines des descriptions d'arbres visibles à la phase de prédiction qui forment le contexte de B (ie, $U' = U \cup \{N \in \mathcal{N} \mid N$ appartient à D et à $\{B'_1, \dots B'_{n+m}\}\} \cup \{N \in \mathcal{N} \mid N$ appartient à S et à $\{B'_1, \dots B'_{n+m}\}\}$).

– D' est l'ensemble des racines des descriptions d'arbres décrochées lors des phases de prédiction qui n'ont pas été utilisées lors des neutralisations précédentes (ie, $D' = D \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, \dots B'_{n+m}\}\} \cup S \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, \dots B'_{n+m}\}\}$).

3.3.3 La règle de balayage

Les grammaires d'interactions ne proposent pas de distinction franche terminaux/non-terminaux comme dans sa version primitive. Pour palier ce problème, nous opérons le balayage dans la phase de prédiction.
Lors de la phase prédictive nous repérons les contextes $\{B'_1, \dots B'_{n+m}\}$ qui ne possèdent qu'un ancre et qui n'ont pas de fils larges ou strictes. Si l'ancre correspond à l'item lexical attendu dans l'énoncé, alors nous prédisons l'item déjà analysé en incrémentant de 1 la portion de l'énoncé analysé.

$$\frac{< A \longrightarrow \alpha \bullet B_{\{B_1,\dots B_n\}}\beta, (i,j), S, U, D >}{< B_{\{B'_1,\dots B'_{n+m}\}} \longrightarrow \bullet, (j, j+1), (S', U', D') >}, \text{ avec}$$

– $B'_i = B_i$ si $i \leq n$, sinon B'_i est la racine d'une description d'arbres visible.
– Un seul des B'_i est une ancre.
– Les nœuds de $\{B'_1, \dots B'_{n+m}\}$ n'ont pas de fils ni au sens strict ni au sens large.
– $S' = \emptyset$.
– U' est l'union de U et des racines des descriptions d'arbres visibles à la phase de prédiction qui forment le contexte de B (ie, $U' = U \cup \{N \in \mathcal{N} \mid N$ appartient à D et à $\{B'_1, \dots B'_n\}\} \cup \{N \in \mathcal{N} \mid N$ appartient à S et à $\{B'_1, \dots B'_n\}\}$).
– D' est l'ensemble des racines des descriptions d'arbres décrochées lors des phases de prédiction qui n'ont pas été utilisées lors des neutralisations précédentes (ie, $D' = D \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, \dots B'_n\}\} \cup S \setminus \{N \in \mathcal{N} \mid N$ appartient à $\{B'_1, \dots B'_n\}\}$).

3.3.4 La règle de complétion

Cette règle permet de remonter dans l'arbre d'analyse en construction en validant un sous-arbre d'analyse.
Pour valider le sous-arbre d'analyse, il faut vérifier que les descriptions d'arbres "décrochées" lors de la phase de prédiction aient bien été "raccrochées" quelque part dans le sous-arbre d'analyse.

$$\frac{< A \longrightarrow \alpha \bullet B_{\{B_1,\dots B_n\}}\beta, (i,j), (S,U,D) > \quad < B_{\{B'_1,\dots B'_{n+m}\}} \longrightarrow \gamma\bullet, (j,k), (S', U', D') >}{< A \longrightarrow \alpha B_{\{B_1,\dots B_{n+m}\}} \bullet \beta, (i,k), (S'', U'', D'') >}, \text{ avec}$$

– $B'_i = B_i$ si B_i si $i \leq n$, sinon B'_i était une description d'arbres visible à l'étage précédent de l'analyse.
– $S' = \emptyset$
– S'' est l'ensemble des racines des descriptions d'arbres décrochées à l'étage précédent qui n'ont pas été encore utilisées dans l'analyse (ie, $S'' = S \setminus \{N \in \mathcal{N} \mid N$ appartient à S et à U').
– $U'' = U' \setminus \{N \in \mathcal{N} \mid N$ appartient à $S\}$
– $D'' = D'$

Résultat de l'analyse

Un énoncé est correct si l'analyse produit un item $< \top \longrightarrow S_N\bullet, (0,n), (\emptyset, U, \emptyset) >$ où n est la longueur de l'énoncé.

3.4 Implémentation dans LEOPAR

L'algorithme tel qu'il a été spécifié précédemment fait l'objet d'un implémentation dans LEOPAR. Cependant, il a subi quelques modifications pour correspondre aux structures utilisées dans LEOPAR.

Les grammaires d'interaction telles qu'elles sont décrites dans LEOPAR comportent quelques propriétés supplémentaires par rapport à ce qui a été décrit en 1.2. En effet, les nœuds des descriptions d'arbres portent une information d'arité sur son nombre de constituants immédiats et deux nœuds peuvent partager une même valeur de trait.
Si ce dernier point est invisible pour l'analyse d'une grammaire, la contrainte d'arité nous permet de diminuer la combinatoire lorsqu'on génère tous les ordres possibles des fils de l'ensemble des nœuds d'un contexte en phase prévisionnel.

LEOPAR, lors de l'étiquetage syntaxique de l'énoncé (procédé qui pour un item lexical sélectionne les descriptions d'arbres élémentaires correspondants), crée un automate à états fini où chaque transition correspond à une description d'arbres élémentaire et où chaque chemin correspond à un étiquetage possible de l'énoncé. La version actuelle de l'algorithme analyse chaque chemin indépendament sans profiter de la structure en automate de l'énoncé. Un travail prochain sera d'analyser directement l'automate pour profiter de la tabulation de tous les sous-arbres analyses.

3.5 Exemple de déroulement de l'algorithme

Soit D la description d'arbres reprentée à la figure 3.1.

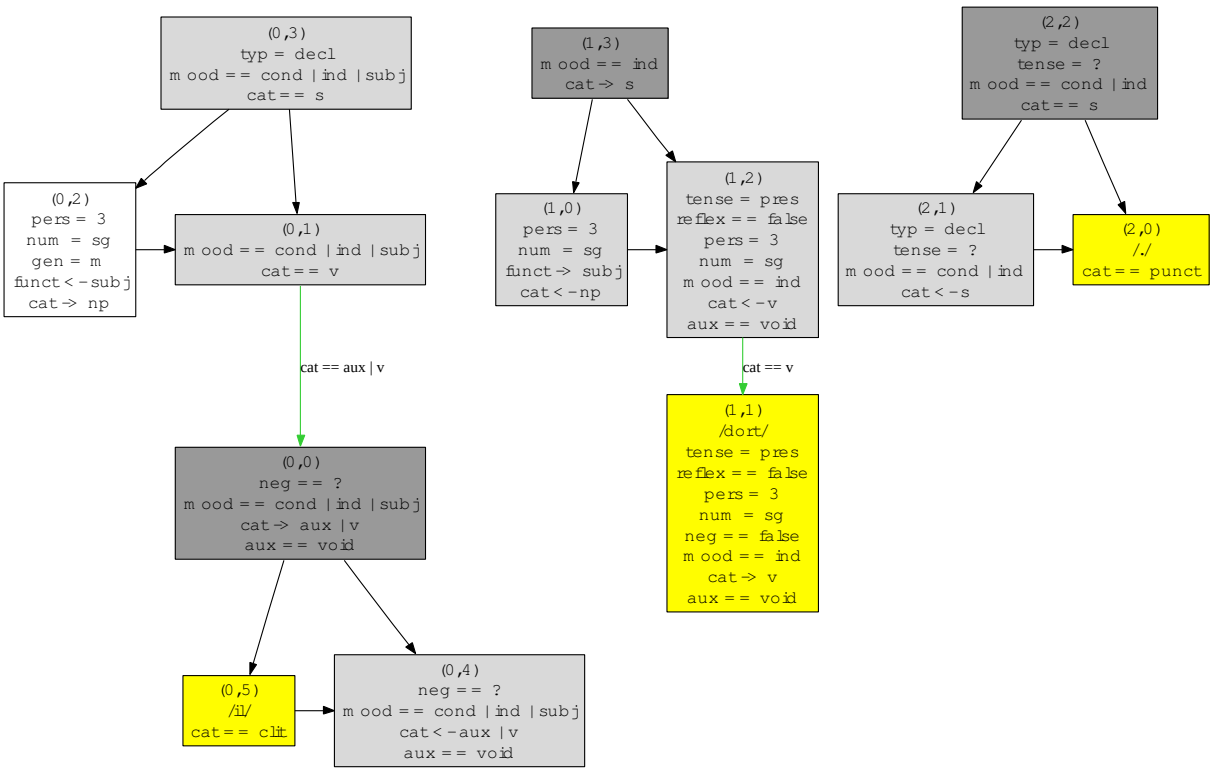

FIG. 3.1 – Description d'arbres de l'énoncé "Il dort ."

Et W l'énoncé suivant : ${}_0Il_1dort_{2\cdot 3}$

L'analyse de Earley de W avec la grammaire D produit les items suivants :

0	$< \top \longrightarrow \bullet\ S_{\{(0,3)\}}\ , (0,0), (\{(1,3),(2,2)\},\{(0,3)\},\emptyset) >$	A
1	$< \top \longrightarrow \bullet\ S_{\{(2,2)\}}\ , (0,0), (\{(0,3),(1,3)\},\{(2,2)\},\emptyset) >$	A
2	$< \top \longrightarrow \bullet\ S_{\{(0,3,)(2,2)\}}\ , (0,0), (\{(1,3)\},\{(0,3),(2,2)\},\emptyset) >$	A
3	$< S_{\{(0,3)\}} \longrightarrow \bullet\ NP_{\{(0,2)\}}\ V_{\{(0,1)\}}, (0,0), (\emptyset,\{(0,3)\},\{(1,3),(2,2)\}) >$	P 0
4	$< S_{\{(2,2)\}} \longrightarrow \bullet\ S_{\{(2,1)\}}\ \cdot_{\{(2,0)\}}, (0,0), (\emptyset,\{(2,2)\},\{(0,3),(1,3)\}) >$	P 1
5	$< S_{\{(1,3),(2,1)\}} \longrightarrow \bullet\ NP_{\{(1,0)\}}\ V_{\{(1,2)\}}, (0,0), (\emptyset,\{(1,3),(2,2)\},\{(0,3)\}) >$	P 4
6	$< S_{\{(0,3),(1,3),(2,1)\}} \longrightarrow \bullet\ NP_{\{(0,2),(1,0)\}}\ V_{\{(0,1),(1,2)\}}, (0,0), (\emptyset,\{(0,3),(1,3),(2,2)\},\emptyset) >$	P 4
7	$< NP_{\{(0,2),(1,0)\}} \longrightarrow \bullet, (0,0), (\emptyset,\{(0,3),(1,3),(2,2)\},\emptyset) >$	P 7
8	$< S_{\{(0,3),(1,3),(2,1)\}} \longrightarrow NP_{\{(0,2),(1,0)\}} \bullet\ V_{\{(0,1),(1,2)\}}, (0,0), (\emptyset,\{(0,3),(1,3),(2,2)\},\emptyset) >$	C 6 8
9	$< V_{\{(0,0),(0,1),(1,2)\}} \longrightarrow \bullet\ il_{\{(0,5)\}}\ V_{\{(0,4)\}}, (0,0), (\{(1,1)\},\{(0,3),(1,3),(2,2)\},\emptyset) >$	P 8
10	$< il_{\{(0,5)\}} \longrightarrow \bullet, (0,1), (\emptyset,\{(0,3),(1,3),(2,2)\},\{1,1\}) >$	B 9
11	$< V_{\{(0,0),(0,1),(1,2)\}} \longrightarrow il_{\{(0,5)\}} \bullet\ V_{\{(0,4)\}}, (0,1), (\{(1,1)\},\{(0,3),(1,3),(2,2)\},\emptyset) >$	C 10 11
12	$< dort_{\{(0,4),(1,1)\}} \longrightarrow \bullet, (1,2), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	B 11
13	$< V_{\{(0,0),(0,1),(1,2)\}} \longrightarrow il_{\{(0,5)\}}\ dort_{\{(0,4),(1,1)\}} \bullet, (0,2), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	C 11 12
14	$< S_{\{(0,3),(1,3),(2,1)\}} \longrightarrow NP_{\{(0,2),(1,0)\}}\ V_{\{(0,0),(0,1),(1,2)\}} \bullet, (0,2), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	C 7 13
15	$< S_{\{(2,2)\}} \longrightarrow S_{\{(0,3),(1,3),(2,1)\}} \bullet\ \cdot_{\{(2,0)\}}, (0,2), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	C 4 14
16	$< \cdot_{\{(2,0)\}} \longrightarrow \bullet, (2,3), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	B 15
17	$< S_{\{(2,2)\}} \longrightarrow S_{\{(0,3),(1,3),(2,1)\}}\ \cdot_{\{(2,0)\}} \bullet, (0,3), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	C 15 16
18	$< \top \longrightarrow S_{\{(2,2)\}} \bullet, (0,3), (\emptyset,\{(0,3),(1,1),(1,3),(2,2)\},\emptyset) >$	C 1 17

Cette analyse correspond à la construction du modèle présenté à la figure 3.2.

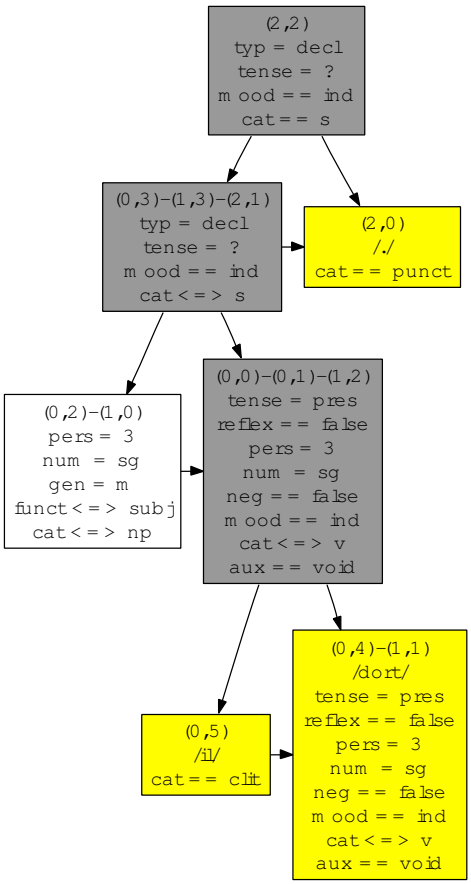

FIG. 3.2 – Modèle valide calculé par l'algorithme de l'énoncé "Il dort ."

Chapitre 4

Résultats expérimentaux et conclusions

4.1 Premiers résultats

L'implémentation actuelle de l'algorithme dans LEOPAR n'est pas assez robuste pour évaluer les performances réelles de l'algorithme comparé aux stratégies d'analyse déjà implémentées (il existe actuellement deux analyseurs, un analyseur de type Shift/Reduce et un autre de type CYK). Cependant de premiers résultats nous permettent d'effectuer quelques points de comparaison :

– Ainsi, la vitesse d'analyse pour les phrases peu ambiguës est sensiblement la même pour toutes les stratégies d'analyse. Cependant l'analyseur de Earley à tendance à être moins efficace lorsque un grand nombre des racines des descriptions d'arbres élémentaires de la grammaire peuvent être potentiellement inclus dans le contexte du nœud racine d'un modèle. Cela implique une combinatoire forte dés le début de l'analyse, et donc une légère baisse de performances.

– De plus, comme pour l'instant l'algorithme analyse chaque étiquetage d'un énoncé et non un automate, celui-ci ne tabule pas d'un étiquetage à l'autre. C'est pour cela que pour les phrases fortement ambiguës, nous remarquons une sensible chute de performances comparé à son homologue tabulaire CYK. Cependant, et même si cela devra être vérifié par une évaluation à plus grande échelle, nous pensons que lorsque l'algorithme tabulera sur l'automate en entrée, nous devrions avoir un gain de performance assez important de sorte que le temps d'analyse soit comparable au temps d'analyse de CYK.

– Un autre problème est lié à l'analyse tabulaire. En effet, que ce soit pour l'algorithme CYK ou Earley, il y a assez vite une explosion de l'espace mémoire utilisé par l'analyseur. De plus, de ce point de vue là, l'algorithme de Earley est tout de même plus gourmand que CYK. Cela est dû à la création de trop d'items inutiles lors de la phase de prédiction. Non seulement cette surgénération fait baisser les performances de l'analyse mais elle implique aussi une explosion de la taille du tableau.

4.2 Pistes d'amélioration

L'algorithme de Earley tel qu'il a été présenté, bien que moins performant que les stratégies d'analyse qui existent déjà pour les grammaires d'interaction, propose une alternative crédible. En effet, certaines pistes d'amélioration possibles nous laissent à penser que l'algorithme puisse être aussi performant que ses homologues existants :

– L'introduction d'un "symbole initial" dans la grammaire permettrait de sensiblement diminuer la combinatoire en début d'analyse, et permettrait ainsi un gain de performance notable pour les phrases les plus longues.

– La tabulation, quand elle sera correctement implémentée pour l'algorithme de Earley dans LEOPAR, devrait théoriquement permettre d'obtenir des performances comparables à CYK.

– Enfin, la grande faiblesse du Earley développé reste la surgénération d'items lors de la phase de prédiction. Cela entraîne un perte d'efficacité en temps et en espace. Nous proposons deux premières pistes pour diminuer cette combinatoire :

- Une première idée serait de diminuer cette combinatoire en essayant d'en descendre une partie le plus bas possible dans l'analyse. Pour cela, ce serait peut-être intéressant lors de la phase de prédiction de ne pas construire de contextes avec des nœuds n'apportant aucune polarité (les nœuds neutres) et de les recoller seulement s'ils deviennent nécessaires pour avancer dans le sous-arbre d'analyse. Ce procédé renforcerait l'analyse du point de vue de la dualité besoins-ressources.
- Une deuxième idée pourrait être d'effectuer une analyse de Earley guidée. En effet, de façon similaire à ce que propose Pierre Boulier pour une analyse de Earley guidée pour les grammaires d'arbres adjoints ([Bou03]), il serait possible de créer un guide à partir des descriptions d'arbres de la grammaire qui restreindrait le choix des items produits lors de la prédiction.

4.3 Conclusion

Nous avons présenté dans ce rapport un algorithme d'analyse descendante pour les grammaires d'interaction. Cet algorithme a fait l'objet d'une implémentation dans LEOPAR. Ceci nous a permis de nous rendre compte que cet analyseur pouvait être une alternative crédible aux stratégies d'analyses existantes au vue des améliorations qu'on pourrait y apporter.

Bibliographie

[BGP03] G. Bonfante, B. Guillaume, and G. Perrier. Analyse syntaxique électostatique. *Traitement Automatique des Langues*, 44 :3 Évolutions en analyse syntaxique, 2003.

[Bou03] Pierre Boullier. Guided Earley parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 43–54, Nancy, France, April 2003.

[Ear70] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2) :94–102, 1970.

[JLT75] Aravind K. Joshi, Leon S. Levy, and M. Takahashi. Tree adjunt grammars. *Journal of Computer and System Sciences*, 10(1) :136–162, February 1975.

[KNT01] Alexander Koller, Joachim Niehren, and Ralf Treinen. Dominance constraints : Algorithms and complexity. In M. Moortgat, editor, *Third International Conference on Logical Aspects of Computational Linguistics (Dec. 1998, Grenoble, France)*, volume 2014 of *Lecture Note in Artificial Intelligence*, pages 106–125, Heidelberg, 2001. Springer-Verlag.

[Per02] Guy Perrier. Descriptions d'arbres avec polarités : les grammaires d'interation. In *TALN02*, 2002.

[RVS92] J. Rogers and K. Vijay-Shanker. Reasoning with descriptions of trees, 1992.

[Ré00] Christian Rétoré. Systèmes déductifs et traitement des langues :un panorama des grammaires catégorielles. *Traitement automatique du langage naturel*, 20(3) :301–336, 2000.

[SJ88] Yves Schabes and Aravind K. Joshi. An Earley-type parsing algorithm for tree adjoining grammars. In *Proc. of 26th Annual Meeting of the Association for Computational Linguistics*, pages 258–269, Buffalo, NY, USA, June 1988. ACL.

[Vil99] Éric Villemonte de la Clergerie. Tabulation et traitement de la langue. ATALA, Cargèse, Corse, France, July 1999. Tutoriel présenté à la $6^{\text{ème}}$ conférence annuelle sur le Traitement Automatique des Langues Naturelles (TALN'99).

[VS92] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Comput. Linguist.*, 18(4) :481–517, 1992.