# Étiquetage grammatical symbolique et interface syntaxe-sémantique des formalismes grammaticaux lexicalisés polarisés

Mathieu Morey

Nancy-Université
Université Nancy 2

**Département de formation doctorale en informatique** **École doctorale IAEM Lorraine**
**UFR mathématiques et informatique**

# Étiquetage grammatical symbolique et interface syntaxe-sémantique des formalismes grammaticaux lexicalisés polarisés

## THÈSE

présentée et soutenue publiquement le 3 novembre 2011

pour l'obtention du

**Doctorat de l'université Nancy 2**

**(spécialité informatique)**

par

Mathieu Morey

**Composition du jury**

| | | |
|---|---|---|
| *Rapporteurs :* | Philippe Blache | Directeur de Recherche au CNRS, Aix-en-Provence |
| | Alexis Nasr | Professeur à l'Université de la Méditerranée, Marseille |
| *Examinateurs :* | Guillaume Bonfante | Maître de Conférences à l'INPL, Nancy |
| | Gérard Huet | Directeur de Recherche à l'INRIA, Rocquencourt |
| | Sylvain Kahane | Professeur à l'Université Paris Ouest, Nanterre |
| | Guy Perrier (directeur) | Professeur à l'Université Nancy 2, Nancy |
| | Alain Polguère | Professeur à l'Université Nancy 2, Nancy |

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

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

# Remerciements

Je tiens à remercier en premier lieu Philippe Blache et Alexis Nasr de m'avoir fait l'honneur d'accepter de rapporter mon manuscrit et d'avoir eu la gentillesse de tolérer les multiples retards qui ont parsemé sa finalisation. Je remercie tout autant Gérard Huet et Alain Polguère d'avoir accepté de participer à mon jury de thèse. Je remercie particulièrement Sylvain Kahane pour l'attention qu'il a témoignée depuis plusieurs années maintenant à mon travail et les discussions, peu nombreuses mais fructueuses, que nous avons eues à ce sujet.

Je remercie également Guy Perrier d'avoir dirigé mon stage de master puis cette thèse. Depuis le début, Guy m'a toujours témoigné la même confiance et m'a toujours laissé la même liberté. Sa ténacité et son pragmatisme m'ont beaucoup appris. Je remercie Guillaume Bonfante d'avoir apporté autant d'idées et de s'être autant investi dans l'élaboration et le déroulement de mon travail. Le contenu scientifique de cette thèse doit énormément à sa rigueur formelle et à sa créativité, bien que ce manuscrit ne rende justice ni à l'une, ni à l'autre. Je remercie enfin Bruno Guillaume d'avoir rendu cette thèse possible. Sa capacité à mener de front formalisation et expérimentation avec la même efficacité est aussi impressionnante que précieuse.

Je remercie Maxime Amblard et Joseph Le Roux d'avoir donné de leur temps pour relire des version préliminaires de ce manuscrit. Je remercie les membres de l'équipe-projet Calligramme puis Sémagramme de m'avoir fourni un environnement scientifique et humain de grande qualité pendant quatre ans et demi. Je remercie en particulier Paul Masson, dont le travail de développement logiciel a été primordial. Je remercie également Novak Novakovic pour avoir supporté de partager le même bureau que moi pendant trois ans tout en m'accordant son amitié.

À l'UHP, je remercie les membres du département informatique de l'UFR STMIA de m'avoir accueilli en tant que moniteur puis ATER, notamment Didier Galmiche et Odile Mella.

Je remercie tous les stagiaires, les doctorants, les ingénieurs, les permanents, les assistantes, les techniciens et les membres du service de restauration que j'ai eu la chance de croiser au LORIA, à l'INRIA Nancy Grand Est, au CIES de Lorraine et aux différentes conférences auxquelles j'ai assisté, et qui m'ont accordé leur considération ou leur amitié.

Je remercie ceux auprès desquels j'ai contracté, faute de disponibilité, de temps ou de présence, des dettes morales immenses : ma famille et mes amis, qui m'ont toujours entouré et soutenu. Je remercie en particulier ceux qui forment, depuis neuf ans pour certains, ma grande famille lorraine. C'est grâce à eux que je me sens chez moi à Nancy.

Enfin, je remercie Anne-Sophie de m'avoir porté durant la bien trop longue période de rédaction de ce manuscrit, et plus globalement de partager ma vie et mes rêves. C'est grâce à elle que je me sens chez moi.

*À Thérèse et Michel, à Cécile et Jean, mes grands-parents,*
*À Françoise et Didier, mes parents*

# Table des matières

# Table des figures

# Introduction

Les travaux exposés dans ce manuscrit se situent dans le domaine du Traitement Automatique des Langues (TAL) et portent plus précisément sur la tâche de l'analyse syntaxique et sémantique de la phrase. L'objectif est de faire de l'analyse syntaxique et sémantique à grande échelle sur des corpus écrits, journalistiques et littéraires, sans sacrifier la précision et la finesse des analyses produites. L'accent mis sur la précision justifie l'emploi de méthodes symboliques.

Plus concrètement, ce travail s'inscrit dans le contexte du développement d'une chaîne d'analyse linguistique autour de l'analyseur syntaxique LEOPAR pour les Grammaires d'Interaction (IG). Les IG sont un formalisme grammatical lexicalisé dans lequel la notion de polarité joue un rôle central.

## 1 Contexte et motivation de la thèse

### 1.1 Les formalismes grammaticaux lexicalisés

Les formalismes grammaticaux lexicalisés fournissent un cadre de modélisation de la syntaxe des langues naturelles qui ancre la grammaire dans le lexique. Dans un formalisme grammatical lexicalisé, une grammaire est composée de descriptions syntaxiques élémentaires qui sont toutes ancrées par (au moins) un mot. Dans ce cadre, une phrase est grammaticale si et seulement si (1) l'on peut associer une description syntaxique élémentaire à chacun de ses mots (2) de telle façon que la séquence de ces descriptions syntaxiques élémentaires ait pour modèle une structure syntaxique dont la forme phonologique est égale à la phrase. Les deux points de cette formulation de la grammaticalité correspondent aux deux phases de l'analyse syntaxique pour les grammaires lexicalisées : l'étiquetage grammatical (ou supertagging), qui associe une description à chaque mot, et l'analyse syntaxique proprement dite, qui compose ces descriptions [SAJ88].

### 1.2 Les formalismes grammaticaux polarisés

Les formalismes grammaticaux polarisés mettent l'accent sur l'état de saturation de l'information syntaxique. Ils utilisent pour cela un système de polarités [Kah06]. Ce système de polarités varie selon les formalismes mais les polarités les plus répandues sont les polarités positives, négatives et saturées. Les polarités positives modélisent les informations disponibles, les polarités négatives les informations attendues, et les polarités saturées les informations saturées.

Le but du processus d'analyse est de produire par composition des polarités une structure totalement neutre, qui ne contient plus aucune polarité positive ou négative. Pour saturer une polarité positive, il faut la composer avec une polarité négative. Pour saturer une polarité négative, il faut la composer avec une polarité positive.

Les polarités ont émergé dans les travaux menés sur les logiques de ressources [Gir87]. Elles sont donc présentes de façon sous-jacente dans les formalismes grammaticaux de la famille des Grammaires Catégorielles (CG) [Ret96, Ret00]. Les CG utilisent les polarités pour modéliser l'état de saturation des syntagmes. En réalité, des travaux subséquents ont montré que d'autres formalismes, dont les Grammaires d'Arbres Adjoints (TAG) [JS97] les Grammaires Syntagmatiques Guidées par les Têtes (HPSG) [PS94] et les Grammaires Lexicales Fonctionnelles (LFG) [KB95], peuvent être vus comme des formalismes polarisés [BGP04, Kah06, Kow07].

## 1.3 Les Grammaires d'Interaction

Les Grammaires d'Interaction (IG) sont un formalisme grammatical lexicalisé polarisé qui appartient au courant de description syntaxique fondé sur la *théorie des modèles*, appelé en anglais *Model-Theoretic Syntax* (MTS) [PS01]. Dans ce paradigme, une grammaire est un ensemble de contraintes et une phrase est grammaticale si elle admet un modèle qui satisfait les contraintes de la grammaire.

Les IG reposent sur deux idées principales que sont la polarisation des traits et la sous-spécification structurelle. Alors que la plupart des formalismes, comme les CG et les TAG polarisées, utilisent les polarités pour modéliser l'état de saturation des syntagmes, la particularité des IG est de descendre la polarisation au niveau des traits. La sous-spécification structurelle, quant à elle, permet de représenter en une seule structure générale plusieurs structures construites à partir des mêmes fragments, en sous-spécifiant les relations que ces fragments entretiennent entre eux [VS92].

## 1.4 Étiquetage grammatical symbolique pour les formalismes polarisés

### Approches statistique et symbolique

Les travaux sur l'étiquetage grammatical dessinent deux perspectives très distinctes sur ce problème. L'approche majoritaire est l'approche statistique [BJ10], qui voit l'étiquetage comme un problème de *classification*. Ces méthodes visent à associer à la séquence des mots de la phrase la ou les bonnes séquences d'étiquettes. L'approche statistique est extrêmement efficace, cependant elle est susceptible d'écarter des étiquetages peu probables et néanmoins corrects. La deuxième approche, à la suite des travaux de Boullier [Bou10], est l'approche symbolique, qui refuse de perdre la moindre analyse possible pour une phrase et voit l'étiquetage comme un problème de *filtrage*. L'objectif de ces méthodes est d'écarter, parmi l'ensemble des étiquetages naïvement possibles pour la phrase, tous ceux qui sont en fait absolument impossibles. La difficulté consiste alors à trouver les critères exacts de filtrage les plus discriminants possibles. C'est

cette seconde perspective que nous adoptons.

#### Filtrage par bilan de polarités

En IG, l'analyse syntaxique proprement dite est un problème NP-difficile [BGP03]. Concrètement, cela signifie que la durée de l'analyse augmente exponentiellement avec la longueur de la phrase. Le refus d'écarter la moindre analyse possible pour une phrase a alors un coût potentiel important. Il est donc d'une importance cruciale que la phase d'étiquetage grammatical soit la plus discriminante possible : plus elle est efficace, plus l'espace de recherche du processus d'analyse syntaxique est restreint.

Les méthodes de filtrage développées jusqu'à présent pour les formalismes grammaticaux polarisés en général [BGP04] et pour les IG en particulier [BLRP06], exploitent l'idée qu'un étiquetage grammatical ne peut avoir de solution si son bilan de polarités n'est pas équilibré. En effet, dans les formalismes grammaticaux polarisés, le produit de l'analyse est une structure neutre. Cela signifie que toutes les polarités non neutres d'un étiquetage grammatical doivent être neutralisées. En particulier, les polarités positives et négatives se neutralisent en formant des couples. Pour pouvoir produire une structure neutre, un étiquetage grammatical doit donc contenir autant de polarités positives que négatives. Cette propriété est un invariant du processus d'analyse, elle peut donc être utilisée pour filtrer les étiquetages grammaticaux possibles d'une phrase.

### 1.5 Interface syntaxe - sémantique

Dans la plupart des formalismes grammaticaux, l'analyse syntaxique et l'analyse sémantique sont étroitement liées. Les deux analyses peuvent être conduites dans un même processus, c'est ce que nous qualifions d'architecture intégrée, ou dans deux processus menés en parallèle qui se contraignent mutuellement, c'est ce que nous qualifions d'architecture parallèle. Une troisième possibilité consiste à appliquer les processus d'analyse de façon séquentielle, l'analyse sémantique étant effectuée à partir du résultat de l'analyse syntaxique.

On distingue principalement deux niveaux d'analyse sémantique. Le premier est celui de l'analyse sémantique superficielle, où la question est d'identifier les prédicats d'un texte, et pour chacun de ces prédicats, arriver à déterminer qui a fait quoi à qui, où et comment. Cette détermination n'est pas fine, l'enjeu est de retrouver le groupe de mots qui contient chacune de ces informations. Le deuxième niveau est celui de l'analyse sémantique profonde, qui associe à une phrase une ou plusieurs structures qui représentent le sens de la phrase. Ces structures contiennent la contribution sémantique de chacun des mots qui composent la phrase. La sémantique profonde qui nous intéresse ici est la sémantique vériconditionnelle compositionnelle. Une phrase est caractérisée par ses conditions de vérité logique, c'est-à-dire que son sens est représenté par une formule logique. Cette formule logique est construite de façon compositionnelle, à partir de la contribution du sens des mots de la phrase et du mode de composition syntaxique de ces mots. Plus encore que l'analyse syntaxique, l'ambiguïté inhérente à l'analyse

sémantique pose des problèmes d'ingénierie linguistique complexes. Afin de limiter l'explosion de l'ambiguïté sémantique d'une phrase, l'une des solutions les plus répandues en linguistique informatique consiste à représenter un ensemble de formules logiques par une seule structure. Une telle structure est qualifiée de représentation sémantique sous-spécifiée. Il est possible d'utiliser ces structures de façon compacte, sans énumérer les formules qu'elles représentent, pour des tâches de raisonnement et d'inférence.

# 2 Problèmes

## 2.1 Limites du filtrage par bilan de polarités

La capacité discriminante des méthodes de filtrage par bilan de polarités est encore insuffisante pour les phrases longues et ambiguës rencontrées dans les corpus comme le French Tree-Bank [ACT03]. Cela constitue un frein important à l'utilisation des Grammaires d'Interaction à grande échelle.

Les méthodes de filtrage par bilan de polarités réduisent les descriptions syntaxiques élémentaires à l'ensemble de leurs polarités positives et négatives. Elles ignorent donc les autres polarités insaturées, comme les polarités virtuelles en IG ou les polarités blanches de Kahane [Kah06] qui doivent être composées avec une polarité saturée. Ces polarités modélisent un contexte obligatoire, ce qui permet par exemple de représenter la relation entre un modifieur et son gouverneur en grammaires de dépendances. Le modifieur requiert la présence de son gouverneur mais ne change pas l'état de saturation de ce dernier. Les méthodes de filtrage par bilan de polarités oublient également les structures qui organisent l'information syntaxique : structures de traits, arbres, connecteurs logiques orientés... En orientant les polarités les unes par rapport aux autres et par rapport à l'ancre, ces structures permettent de contrôler la directionnalité des compléments d'un mot et l'ordre de ces compléments. En regroupant les polarités, elles permettent de spécifier le genre et le nombre de ces compléments. Le filtrage par bilan de polarités est donc incapable de filtrer un étiquetage qui ne respecte pas les contraintes de contexte, l'ordre des mots ou l'accord en genre et en nombre. La prise en compte de ces informations constitue une piste intéressante d'amélioration de l'efficacité de la phase d'étiquetage grammatical.

## 2.2 Interface syntaxe - sémantique

La plupart des propositions d'interface syntaxe - sémantique présentent deux limites. Premièrement, elles n'utilisent qu'une partie de l'information produite par l'analyse syntaxique. Ainsi, de nombreuses grammaires calculent les relations grammaticales de contrôle ou les sujets profonds des infinitifs et des participes, mais cette information est ignorée par l'interface syntaxe - sémantique. Deuxièmement, le lien entre les processus d'analyse syntaxique et sémantique est très fort, ce qui signifie que l'interface syntaxe - sémantique elle-même est très liée au formalisme ou à la grammaire. Ces deux points signifient que l'effort de développement d'une interface syntaxe - sémantique est conséquent et peu générique.

Or, la composition de structures syntaxiques est très souple en IG et la grammaire FRIGRAM du français est en développement constant. Cela signifie qu'il n'est possible de poser d'hypothèse forte ni sur le formalisme, ni sur la grammaire, pour développer une interface syntaxe - sémantique. Par ailleurs, il n'existe pas pour le français de corpus de référence annoté avec des structures sémantiques. Cela renchérit d'autant le coût potentiel de développement d'une interface syntaxe - sémantique.

## 3 Contributions de la thèse

Les contributions de cette thèse s'articulent en trois parties.

Dans la première partie, nous exploitons les propriétés de lexicalisation et de polarisation des formalismes grammaticaux pour extraire des informations statiques de la grammaire. Plus précisément, nous exploitons les besoins exprimés par les polarités attachées aux structures d'une grammaire pour calculer, pour chaque structure de la grammaire, l'ensemble des structures de la grammaire qui sont capables de répondre à l'un de ses besoins de composition. Nous définissons ainsi la notion de *compagnon hypothétique* d'une polarité, que nous exploitons immédiatement pour définir la notion de *quasi-réduction d'une grammaire polarisée*. Quasi-réduire une grammaire consiste à en ôter toutes les structures qui ne peuvent pas participer à une analyse, car elles ont au moins un besoin de composition qui ne peut être rempli par aucune structure de la grammaire. La quasi-réduction est applicable à toute grammaire polarisée. Cette notion, ainsi que celle de compagnon hypothétique, trouve des applications dans le développement et la maintenance de grammaires.

Dans la deuxième partie, nous proposons des méthodes qui améliorent l'efficacité de la première phase de l'analyse syntaxique des formalismes lexicalisés : l'étiquetage grammatical. Nous exploitons la notion de compagnon hypothétique pour formuler un critère de correction des étiquetages grammaticaux, le *principe du compagnonnage*, et nous en dérivons deux méthodes de filtrage symbolique. De ces méthodes dérive une troisième méthode de filtrage, qui est elle réservée à certaines polarités : les polarités linéaires. Nous implantons ces trois méthodes sur automates et nous montrons que, combinées aux méthodes existantes, elles améliorent grandement l'efficacité de la phase d'étiquetage grammatical. Ces méthodes sont applicables à tout formalisme grammatical lexicalisé polarisé.

Dans la troisième partie, nous adoptons une perspective séquentielle sur l'interface syntaxe - sémantique. Nous proposons de calculer une représentation sémantique à partir d'une structure de dépendance syntaxique enrichie de relations grammaticales. Ce choix a l'avantage d'être neutre par rapport au formalisme et à la grammaire utilisés. Pour cela, nous proposons un calcul de réécriture de graphes modulaire à base de commandes et nous l'utilisons pour définir un programme de réécriture qui calcule des représentations sémantiques sous-spécifiées au format *Dependency* MRS [Cop09] à partir de structures de dépendance syntaxique au format du *French Treebank* [CCD10] enrichies de relations grammaticales. Ces relations grammati-

cales représentent des informations syntaxiques profondes, comme les sujets des infinitifs. Nous proposons ensuite un deuxième programme de réécriture qui enrichit un arbre de dépendance syntaxique de surface en lui ajoutant ces relations grammaticales. En composant ces deux programmes de réécriture, il est possible de produire des représentations sémantiques sous-spécifiées à partir de phrases annotées par des dépendances syntaxiques de surface.

## 3.1 Publications

Les résultats contenus dans ce manuscrit ont été obtenus et publiés en collaboration avec Guillaume Bonfante, Bruno Guillaume et Guy Perrier. J'ai développé, intégré et remis en perspective ces résultats pour ce manuscrit.

- *Word Order Constraints for Lexical Disambiguation of Interaction Grammars*. ESSLLI Workshop on Parsing with Categorial Grammars. Bordeaux, France, 2009. Travail réalisé en collaboration avec Guillaume Bonfante et Bruno Guillaume.
- *Dependency Constraints for Lexical Disambiguation*. 11th International Conference on Parsing Technology (IWPT'09). Paris, France, 2009. Travail réalisé en collaboration avec Guillaume Bonfante et Bruno Guillaume.
- *Réécriture de graphes de dépendances pour l'interface syntaxe-sémantique*. 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2010). Montréal, Canada, 2010. Travail réalisé en collaboration avec Guillaume Bonfante, Bruno Guillaume et Guy Perrier.
- *Modular Graph Rewriting to Compute Semantics*. 9th International Conference on Computational Semantics (IWCS 2011). Oxford, Royaume-Uni, 2011. Travail réalisé en collaboration avec Guillaume Bonfante, Bruno Guillaume et Guy Perrier.
- *Enrichissement de structures en dépendances par réécriture de graphes*. 18e Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2011). Montpellier, France, 2011. Travail réalisé en collaboration avec Guillaume Bonfante, Bruno Guillaume et Guy Perrier.

# Première partie

# Formalismes grammaticaux lexicalisés polarisés

# Chapitre 1

# Polarisation des formalismes grammaticaux lexicalisés

Dans ce chapitre, nous posons le cadre général dans lequel s'inscrit notre travail : les formalismes grammaticaux lexicalisés polarisés. Nous présentons en section 1.1 les formalismes grammaticaux lexicalisés, en dégageant leur origine informatique et leurs caractéristiques générales puis en les définissant formellement. Nous introduisons ensuite, en section 1.2, la notion de polarité que nous intégrons aux formalismes grammaticaux lexicalisés, pour aboutir en section 1.3 aux formalismes grammaticaux lexicalisés polarisés. Nous présentons enfin en section 1.4 quelques formalismes grammaticaux lexicalisés polarisés.

## 1.1 Formalismes grammaticaux lexicalisés

### 1.1.1 Un très bref aperçu de l'origine informatique des formalismes grammaticaux

Les formalismes proposés pour la modélisation informatique de la syntaxe des langues naturelles forment trois grands courants qui correspondent aux trois sous-domaines de l'informatique théorique et des mathématiques dont ils adoptent le point de vue, les outils et les techniques.

Historiquement, le premier courant à s'être développé est issu de l'algèbre, puis de la logique et plus précisément de la *théorie de la démonstration* : ce sont les Grammaires Catégorielles (CG), développées à partir des travaux algébriques d'Ajdukiewicz [Ajd35] et Bar-Hillel [BH53] puis formalisées en un système déductif par Lambek [Lam58]. Un formalisme grammatical est assimilé à une logique, les opérations de composition grammaticale sont les règles de déduction. Analyser une phrase revient à construire une démonstration de sa grammaticalité. Ce courant est représenté par les Grammaires Catégorielles (CG) [Mon70] et ses principales variantes que sont les Grammaires de Types Logiques (TLG) [Mor94] et les Grammaires Catégorielles Combinatoires (CCG) [SB09], ainsi que les grammaires de prégroupes [Lam99].

Le deuxième courant est issu de la Grammaire Générative de Chomsky qui utilise, dans

la perspective de la linguistique structuraliste américaine, des travaux sur les *langages formels* et les *systèmes de réécriture* [Cho02]. Une grammaire formelle est assimilée à un système de réécriture, les opérations de composition syntaxique sont les règles de réécriture. Analyser une phrase revient à dériver cette phrase d'un symbole de départ, en un nombre fini de réécritures. Appartiennent à ce courant des formalismes comme le Programme Minimaliste [Cho95] et les Grammaires d'Arbres Adjoints (TAG) [JS97].

Ces deux premiers cadres permettent de générer tous les énoncés bien-formés selon la grammaire fournie ; ils forment ce que Pullum [PS01] appelle le courant *génératif-énumératif* de la syntaxe des langues naturelles (en anglais Generative-Enumerative Syntax, GES).

Le troisième cadre rompt avec ce côté procédural, calculatoire en considérant la syntaxe sous l'angle de la *théorie des modèles* ; c'est ce que Pullum, après Rogers [Rog96], appelle la *syntaxe fondée sur la théorie des modèles* [1] (en anglais Model-Theoretic Syntax, MTS). Une grammaire formelle est assimilée à un système de contraintes, la composition syntaxique est la résolution de contraintes. Analyser une phrase revient à construire un modèle de cette phrase qui respecte les contraintes de la grammaire [Bla07]. En termes logiques, un formalisme grammatical est une théorie logique, et analyser une phrase revient à construire un modèle dans cette théorie [BGM03]. Ce courant est représenté par les formulations les plus récentes des Grammaires Syntagmatiques Guidées par les Têtes (HPSG) [Pol99] et des Grammaires Lexicales Fonctionnelles (LFG) [KB95], ainsi que des formalismes purement MTS comme les Grammaires de Dépendances eXtensibles (XDG) [Deb06], les Grammaires de Propriétés (GP) [Bla05] et les Grammaires d'Interaction (IG) [Per03] que nous présentons au chapitre 2.

### 1.1.2 Lexicalisation des formalismes grammaticaux

La classification de Pullum fournit une grille de lecture pertinente des formalismes d'analyse syntaxique, fondée sur différentes visions du calcul : le calcul comme démonstration, le calcul comme réécriture, et le calcul comme construction d'un modèle. Ces trois courants ne sont cependant pas irréconciliables et les structures que leurs formalismes produisent ne sont pas incomparables, notamment car ces formalismes ont tous été traversés par un mouvement de *lexicalisation*.

La lexicalisation consiste à ancrer la majeure partie de l'information et de la variabilité syntaxique dans le lexique. Concrètement, une grammaire est *lexicalisée* si elle est constituée de (i) un ensemble fini de structures qui sont chacune associées à un élément lexical appelé l'*ancre* de la structure, et (ii) une ou des opérations de composition de ces structures [SAJ88].

L'adoption commune de cette division du travail syntaxique facilite l'établissement de liens formels et la comparaison d'analyses entre formalismes qui reposent sur des principes calculatoires différents ou qui construisent des structures syntaxiques différentes.

Ainsi, la lexicalisation des Grammaires d'Arbres Adjoints (TAG), qui sont issues de la réécri-

1. Nous reprenons ici la traduction de l'expression anglaise utilisée par Denys Duchier et ses collègues dans [DDPL10].

ture, a favorisé la comparaison d'analyses et le partage de techniques d'analyse et d'étiquetage grammatical avec les Grammaires Catégorielles (CG) [BJ10], qui sont issues de la théorie de la démonstration. Le nombre restreint d'opérations de composition sur les structures induit par la lexicalisation a également favorisé l'émergence de la syntaxe MTS, par la reformulation MTS de formalismes GES lexicalisés [PS01].

De même, la lexicalisation facilite la comparaison d'analyses entre les *grammaires syntagmatiques* et les *grammaires de dépendance*. Les grammaires syntagmatiques, comme les CFG, les TAG et les CG, modélisent la phrase comme une boîte divisée récursivement en boîtes de plus en plus petites : les syntagmes. Si l'on renverse la perspective, cette modélisation correspond à la formation de paquets de mots de plus en plus gros. Les grammaires de dépendance (DG) [Mel88, Hud84] modélisent quant à elles la phrase comme un réseau de dépendances entre mots, dans lequel chaque mot permet ou requiert la présence d'autres mots. La lexicalisation des grammaires syntagmatiques a pour effet de rattacher l'information syntagmatique aux mots. Analyser une phrase consiste alors à combiner les structures syntagmatiques associées aux mots de cette phrase, ce qui produit par effet de bord une structure de dépendance entre les mots de la phrase [Kah01].

Enfin, la lexicalisation de la grammaire permet de garantir qu'une phrase de longueur finie est finiment ambiguë, du point de vue de l'analyse syntaxique [SAJ88]. Cette propriété a des répercussions importantes sur la complexité de l'analyse.

Pour ces raisons, tant linguistiques que calculatoires, les formalismes grammaticaux lexicalisés occupent actuellement une place centrale en analyse syntaxique.

### 1.1.3 Présentation formelle des formalismes grammaticaux lexicalisés

**Définition 1.1.1** (Vocabulaire)**.** Un **vocabulaire** est un ensemble fini de mots du langage naturel modélisé.

**Définition 1.1.2** (Phrase)**.** Une **phrase** est une liste finie, notée $[w_1, \ldots, w_n]$, de mots du vocabulaire.

Nous définissons ensuite ce qu'est une **structure**, sur le modèle de la définition de Kahane [Kah06].

**Définition 1.1.3** (Structure)**.** Une **structure** est composée de :
- un ensemble d'**objets**, noté $\mathcal{O}$,
- et un ensemble d'**applications** de $\mathcal{O}$ dans $\mathcal{O}$, noté $\pi$.

Par exemple, un graphe orienté est constitué de deux types d'objets : un ensemble $X$ de nœuds et un ensemble $U$ d'arêtes, tels que $X \cup U = \mathcal{O}$. La structure de graphe est alors définie par deux applications $\pi_1$ et $\pi_2$ de $U$ dans $X$, qui associent à toute arête respectivement son nœud source et son nœud destination [Kah06].

Nous définissons les formalismes grammaticaux lexicalisés de façon suffisamment abstraite pour que cette définition s'applique à tous les formalismes GES et MTS. L'abstraction de cette

définition implique que les opérations de composition de structures, mentionnées dans la définition originale des grammaires lexicalisées [SAJ88], ne soient pas explicitement définies ici. La définition des opérations de composition de structures est en fait incluse dans la définition concrète de l'ensemble de fonctions d'interprétation $\mathcal{J}$.

**Définition 1.1.4** (Formalisme grammatical lexicalisé)**.** Un **formalisme grammatical lexicalisé** est un 6-uplet $(\mathcal{V}, \mathcal{S}, \mathcal{G}, \mathbf{anc}, \mathcal{SF}, \mathcal{J})$ où :

- $\mathcal{V}$ est le **vocabulaire** ;
- $\mathcal{S}$ est l'ensemble des **structures syntaxiques** utilisées par le formalisme ;
- $\mathcal{G} \subset \mathcal{S}$ est la **grammaire** : l'ensemble fini de structures syntaxiques initiales ; une liste finie $[t_1, \ldots, t_n]$ d'éléments de $\mathcal{G}$ est appelée un **étiquetage grammatical** ;
- $\mathbf{anc} : \mathcal{G} \to \mathcal{V}$ est une application des structures syntaxiques initiales vers leurs ancres ;
- $\mathcal{SF} \subset \mathcal{S}$ est l'ensemble des structures syntaxiques finales que le processus d'analyse construit (par exemple des arbres) ;
- $\mathcal{J} = \{I : \mathcal{O}_E \to \mathcal{O}_F\}$ est l'ensemble des **fonctions d'interprétation** $I$ des objets d'un étiquetage grammatical (ensemble noté $\mathcal{O}_E$) vers les objets d'une structure finale (ensemble noté $\mathcal{O}_F$).

Chaque fonction d'interprétation $I$ met en correspondance un étiquetage et une structure finale. Pour un formalisme donné, l'ensemble $\mathcal{J}$ est défini en intension par un ensemble de contraintes que toute fonction d'interprétation de ce formalisme doit vérifier.

Notons que la fonction **anc** impose la lexicalisation de la grammaire : toute structure initiale dans $\mathcal{G}$ est associée à un élément de $\mathcal{V}$. S'ensuit la définition du lexique.

**Définition 1.1.5** (Lexique)**.** Le **lexique** est la fonction, notée $\ell$, de $\mathcal{V}$ vers des sous-ensembles de $\mathcal{G}$ définie par :

$$\ell(w) = \{t \in \mathcal{G} \mid \mathbf{anc}(t) = w\}.$$

La définition de la fonction **anc** impose qu'une structure syntaxique initiale n'a qu'une seule ancre. Dans certains formalismes, comme les LTAG, les structures syntaxiques peuvent contenir, en plus, d'autres mots du vocabulaire, qui sont appelés des *co-ancres*. Dans les grammaires LTAG, les co-ancres sont utilisées couramment pour représenter les particules verbales ou les prépositions régies. Dans ce qui suit, afin de garder des définition simples, nous ne considérerons que des grammaires qui n'emploient pas de co-ancres. Nous qualifions ces grammaires de **strictement lexicalisées** [2].

**Définition 1.1.6** (Étiquetage grammatical d'une phrase)**.** Un étiquetage grammatical $E = [t_1, \ldots, t_n]$ est un étiquetage grammatical de la phrase $[\mathbf{anc}(t_1), \ldots, \mathbf{anc}(t_n)]$.

2. Notre emploi de ce qualificatif est plus restreint que celui que l'on trouve dans la littérature LTAG, où une grammaire strictement lexicalisée est une grammaire dont absolument tous les arbres respectent la contrainte de lexicalisation.

**Définition 1.1.7** (Solutions de l'analyse d'une phrase)**.** L'ensemble des **solutions de l'analyse** d'un étiquetage $E$, noté $\mathbf{p}(E)$, est formé des structures finales $F \in \mathcal{SF}$ telles qu'il existe une fonction d'interprétation $I : \mathcal{O}_E \to \mathcal{O}_F$ qui appartienne à $\mathcal{J}$.

**Définition 1.1.8** (Analyse d'une phrase)**.** Une **analyse d'une phrase** $[w_1, \ldots, w_n]$ est un triplet $(E, F, I)$, avec :

- $E = [t_1, \ldots, t_n]$ un **étiquetage grammatical** de $[w_1, \ldots, w_n]$ ;
- $F$ une **structure syntaxique finale** solution de $E$ ;
- $I : \mathcal{O}_E \to \mathcal{O}_F$ une **fonction d'interprétation** qui associe aux objets de $E$ leurs objets correspondants dans $F$.

La fonction d'interprétation $I$ contient une forme d'historique, qui est la trace de l'analyse.

**Définition 1.1.9** (Langage engendré par une grammaire lexicalisée)**.** Le **langage engendré** par une grammaire lexicalisée $\mathcal{G}$ est :

$$\mathcal{L} = \{[w_1, \ldots, w_n] \mid \exists E = [t_1, \ldots, t_n] \text{ tel que } \forall i, 1 \leq i \leq n, t_i \in \ell(w_i) \text{ et } \mathbf{p}(E) \subset \mathcal{SF}\}$$

Intuitivement, le langage engendré par une grammaire lexicalisée est l'ensemble des phrases pour lesquelles il existe un étiquetage grammatical qui a une solution.

## 1.2 Polarités

Nous introduisons dans cette section la notion de *polarité*, qui est employée dans de nombreux formalismes grammaticaux.

### 1.2.1 Généralités

Les polarités sont indissociables de la notion de ressource, qui est fondamentale tant en logique linéaire qu'en syntaxe des langues [Ret00].

À l'origine, les polarités en syntaxe modélisent la notion de *valence*, qui a été transposée de la chimie au cours des années 1930 par les travaux de Tesnière [Tes34], Jespersen [Jes37] et Ajdukiewicz [Ajd35]. Dans la métaphore chimique, les mots sont des atomes, ils ont une valence qui exprime leur capacité à se combiner à d'autres mots. Le processus de composition syntaxique correspond ainsi à une réaction chimique dont le résultat ultime est la formation d'une molécule : la phrase, dans laquelle les valences de tous les atomes sont saturées.

Cette métaphore, à travers les travaux d'Ajdukiewicz, est à l'œuvre dans les Grammaires Catégorielles (CG), dont les Grammaires Catégorielles Combinatoires (CCG) [SB09] et les Grammaires de Types Logiques (TLG) [Ret00, Mor10]. Les CG utilisent la valence pour modéliser le type des constituants. Par exemple, si on associe aux phrases le type $s$ et aux groupes nominaux le type $np$, alors on peut voir les verbes intransitifs comme des phrases auxquelles il

manque un groupe nominal : leur type est $np\backslash s$ [3], qui correspond à la formule logique polarisée $-np \parr +s$ [Ret00].

De façon générale, pour modéliser l'état de saturation de l'information syntaxique, on utilise un ensemble de polarités $\mathcal{P}$, dont un sous-ensemble $\mathcal{N}$ est distingué qui correspond aux polarités neutres [Kah06]. Le but du processus d'analyse est de produire par composition des polarités une structure totalement neutre.

Remarquons enfin que les polarités ont également été utilisées pour contrôler la compilation de méta-grammaires [DLP05a].

### 1.2.2 Systèmes de polarités

**Définition 1.2.1** (Système de polarités)**.** Un **système de polarités** $\mathcal{Q}$ est un triplet $(\mathcal{P}, \oplus, \mathcal{N})$ tel que :

- $\mathcal{P}$ est un ensemble fini de polarités ;
- $\oplus$ est une opération binaire associative et commutative sur $\mathcal{P}$ ;
- $\mathcal{N} \subset \mathcal{P}$ est le sous-ensemble fini des polarités neutres.

Certains systèmes de polarités permettent de composer plus de deux polarités. Dans les formalismes qui utilisent ces systèmes, une analyse peut donc composer plusieurs objets polarisés. Il est alors nécessaire de définir une opération de composition sur des multi-ensembles de polarités.

Or, l'opération de composition de polarités $\oplus$ est associative et commutative. Par conséquent, elle peut sans risque être généralisée en une opération $\bigoplus$ sur des multi-ensembles non vides de polarités, par itération de l'opération binaire :

- $\bigoplus(\{p\}) = p$ pour les singletons,
- $\bigoplus(\mathcal{M} \uplus p) = \bigoplus(\mathcal{M}) \oplus p$ .

Un multi-ensemble $\mathcal{M}$ de polarités est dit **équilibré** si $\bigoplus(\mathcal{M}) \in \mathcal{N}$.

Comme le but de l'analyse syntaxique dans les formalismes polarisés est de produire une structure neutre, il faut que le processus d'analyse forme des multi-ensembles de polarités équilibrés.

**Exemples de systèmes de polarités** Le système de polarités utilisé varie selon les formalismes. La figure 1.1a présente la définition de l'opération de composition du système de polarités des IG et la figure 1.1b présente celle du système de polarités des Grammaires d'Unification Polarisées (GUP) [Kah06]. Les GUP sont un formalisme générique de combinaison de structures. Le système de polarités de [Kah06] est l'ensemble $\mathcal{P} = \{■, □, -, +, ■\}$ avec $\mathcal{N} = \{■, ■\}$, les polarités étant appelées comme suit :

3. Nous utilisons la notation fractionnaire et non celle des Grammaires Catégorielles Combinatoires (CCG) : ici, les catégories des arguments sont au dénominateur et le résultat au numérateur.

■ : grise (*absolument neutre*) ;
□ : blanche (*contexte obligatoire*) ;
− : négative ;
+ : positive ;
■ : noire (*saturée*).

Sur cette figure, ⊥ dénote l'impossibilité de composer deux polarités.

| ⊕ | = | ∼ | ← | → | ↔ | ⊥ |
|---|---|---|---|---|---|---|
| = | = | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |
| ∼ | ⊥ | ∼ | ← | → | ↔ | ⊥ |
| ← | ⊥ | ← | ⊥ | ↔ | ⊥ | ⊥ |
| → | ⊥ | → | ↔ | ⊥ | ⊥ | ⊥ |
| ↔ | ⊥ | ↔ | ⊥ | ⊥ | ⊥ | ⊥ |
| ⊥ | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |

(a) IG

| ⊕ | ■ | □ | − | + | ■ | ⊥ |
|---|---|---|---|---|---|---|
| ■ | ■ | □ | − | + | ■ | ⊥ |
| □ | □ | □ | − | + | ■ | ⊥ |
| − | − | − | ⊥ | ■ | ⊥ | ⊥ |
| + | + | + | ■ | ⊥ | ⊥ | ⊥ |
| ■ | ■ | ■ | ⊥ | ⊥ | ⊥ | ⊥ |
| ⊥ | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |

(b) GUP

FIGURE 1.1 – Systèmes de polarités

Le système de polarités des GUP est très semblable à celui des IG, où $\{■, □, -, +, ■\}$ correspond à $\{=, \sim, \leftarrow, \rightarrow, \leftrightarrow\}$, à une différence près. Alors qu'en GUP toutes les polarités peuvent interagir avec la polarité grise ■, le système de polarités des IG est constitué de deux sous-systèmes distincts qui ne peuvent pas être composés : la polarité = d'un côté, les autres polarités de l'autre. Ainsi, un trait est soit « réellement polarisé », soit absolument neutre.

Dans ces deux systèmes, les polarités absolument neutres (grise ■ des GUP et = des IG) équivalent en fait à une absence de polarité. C'est pourquoi, comme Lareau [Lar08], nous ne désignerons, dans la suite de ce manuscrit, par « système de polarités » que son sous-système réellement polarisé.

### 1.2.3 Systèmes de polarités monotones

La monotonie de l'analyse est une propriété formelle importante, que certains formalismes comme les IG ou les GUP rendent obligatoire. Pour que l'analyse syntaxique dans un formalisme polarisé soit monotone, il faut que le système de polarités utilisé soit monotone. Pour qu'un système de polarités soit monotone, l'ensemble $\mathcal{P}$ des polarités du système doit être muni d'un ordre partiel, qui est tel que l'opération de composition des polarités, qui est associative et commutative, vérifie également une **propriété de monotonie** [Kah06].

**Définition 1.2.2** (Système de polarités monotone)**.** Un **système de polarités monotone** est un quadruplet $(\mathcal{P}, \oplus, \leq, \mathcal{N})$ tel que :

(i) $(\mathcal{P}, \oplus, \mathcal{N})$ est un système de polarités ;

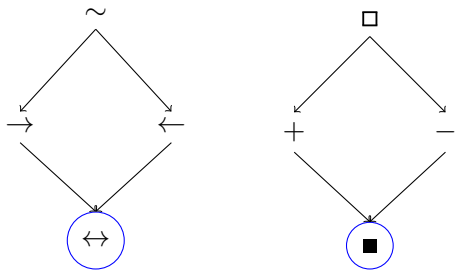
(ii) $\leq$ est un ordre partiel sur $\mathcal{P}$ ;

(iii) $\oplus$ vérifie la propriété de monotonie sur $\leq$ :
$\forall p, q \in \mathcal{P},\ p \leq (p \oplus q)$ ;

(iv) tout élément maximal $g$ de $\leq$ est une polarité neutre :
$\forall g, \forall x \in \mathcal{P},\ (g \leq x \Rightarrow g = x) \Rightarrow g \in \mathcal{N}$ ;

**Exemple d'ordre partiel sur des systèmes de polarités**

Pour les IG et les GUP, l'ordre $\leq$ sur l'ensemble des polarités $\mathcal{P}$ est défini par les treillis de la figure 1.2. Dans chaque treillis, le sous-ensemble $\mathcal{N}$ des polarités neutres est constitué des nœuds entourés.




(a) IG (b) GUP

FIGURE 1.2 – Ordre partiel sur les polarités

L'ordre partiel sur le système de polarités des IG est donné en figure 1.2a. L'élément maximal est la polarité $\leftrightarrow$ qui est une polarité neutre. Il est trivial de vérifier que la propriété de monotonie est respectée.

Pour le système de polarités des GUP dont l'ordre partiel est défini figure 1.2b, l'élément maximal est la polarité $\blacksquare$ qui est une polarité neutre. Là encore, il est trivial de vérifier que la propriété de monotonie est respectée.

### 1.2.4 Classification des polarités

Les systèmes de polarités des IG et des GUP combinent deux sous-systèmes de polarités distincts : les polarités linéaires et les polarités non linéaires. Ces deux sous-systèmes ont d'ailleurs émergé dans deux traditions formelles distinctes de la syntaxe des langues, que nous avons évoquées précédemment : les grammaires de dépendance et les grammaires de types logiques.

**Ressources consommables et réutilisables**

Les polarités des grammaires de types logiques modélisent des ressources consommables [Mor10]. Le système de polarités est constitué des polarités positive, négative et saturée : $\mathcal{P} = \{+, -, \blacksquare\}$ et $\mathcal{N} = \{\blacksquare\}$. La polarité positive correspond à une ressource disponible, la polarité négative à une ressource attendue et la polarité saturée au résultat de la consommation d'une ressource disponible par une ressource attendue.

Les polarités des grammaires de dépendance modélisent des ressources réutilisables. Les arbres de dépendance de Nasr [Nas95, Nas96] utilisent un système de polarités constitué des

polarités blanche et noire : $\mathcal{P} = \{\square, \blacksquare\}$, $\mathcal{N} = \{\blacksquare\}$. La polarité blanche utilise une polarité noire sans la consommer. Une polarité noire peut donc être utilisée par plusieurs polarités blanches.

#### Illustration linguistique

Cette différence entre ressources consommables et réutilisables est illustrée par la façon dont les grammaires de types logiques et les grammaires de dépendance modélisent respectivement les modifieurs, par exemple les adjectifs épithètes.

En grammaires de dépendance, deux épithètes utilisent, sans le consommer, le même nom gouverneur. Dans les grammaires de types logiques, le nom est consommé par son premier épithète, qui produit immédiatement un nouveau nom. Ce nouveau nom est consommé par le deuxième épithète, qui produit à son tour un nouveau nom.

Cette différence a un parallèle dans les grammaires d'arbres : la consommation correspond à l'opération d'adjonction des TAG et l'utilisation sans consommation à l'opération d'adjonction de sœur (*sister-adjunction*) des D-Tree Grammars (DTG) [RVSW95], Grammaires à Insertion d'Arbres Lexicalisées (LTIG) [Chi00] et LTAG-spinal [SCJ08].

#### Terminologie : polarités linéaires et non linéaires

La distinction entre ressources consommables et réutilisables sépare les polarités non neutres en deux catégories, selon leur capacité de combinaison.

Dans le système de polarités des GUP (et des IG), la première catégorie est constituée des polarités $+$ et $-$. Il est impossible de composer deux polarités $+$ avec la même polarité $-$ ; il est également impossible de composer deux polarités $-$ avec la même polarité $+$. Toute polarité $+$ doit se composer avec exactement une polarité $-$ et réciproquement. Par analogie avec la logique linéaire [Gir87], nous qualifions les polarités $+$ et $-$ de *linéaires*.

La seconde catégorie est constituée par les polarités $\blacksquare$. Plusieurs polarités $\blacksquare$ peuvent être composées ensemble, et avec, soit une polarité $\blacksquare$, soit deux polarités $\square$ et $\blacksquare$. Les polarités $\blacksquare$ sont donc qualifiées de *non linéaires*.

### 1.2.5 Interprétation des polarités par des polarités composites

Les interactions entre ressources consommables et réutilisables, polarités linéaires et non linéaires, sont éclairées par l'interprétation des polarités des GUP et des IG par des polarités composites.

**Définition 1.2.3** (Polarité composite)**.** Une **polarité composite** $p$ de taille $n$ est un n-uplet de polarités atomiques : $p = (p_1, \ldots, p_n)$ avec $1 \leq i \leq n, p_i \in \mathcal{P}_A$.

L'opération de composition sur les polarités composites fonctionne composante par composante.

**Définition 1.2.4** (Composition de polarités composites)**.** La composition de $m$ polarités composites $p^1, \ldots, p^m$ de taille $n$ est définie par :

$$\bigoplus(p^1, \ldots, p^m) = (\bigoplus(p_1^1, \ldots, p_1^m), \ldots, \bigoplus(p_n^1, \ldots, p_n^m))$$

Comme l'indique Kahane [Kah06], le système de polarités $\mathcal{P}_{GUP} = \{\square, -, +, \blacksquare\}$ peut être interprété par le système de polarités composites $\mathcal{P}' = \{(\square, \square), (\square, \blacksquare), (\blacksquare, \square), (\blacksquare, \blacksquare)\}$ défini sur $\mathcal{P}_A = \{\square, \blacksquare\}$. Le système de polarités des IG $\mathcal{P}_{IG} = \{\sim, \leftarrow, \rightarrow, \leftrightarrow\}$ peut être interprété, lui aussi, par le système $\mathcal{P}'$. Le treillis de la figure 1.3 représente l'ordre sur les polarités composites qui interprètent les polarités des IG et des GUP.

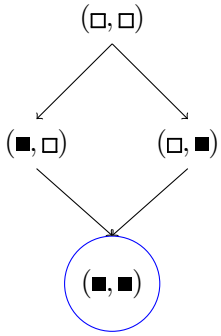



FIGURE 1.3 – Ordre partiel sur les polarités composites

En d'autres termes, le (sous-)système de polarités $\{-, +, \blacksquare\}$ qui modélise des ressources consommables peut être interprété à l'aide du seul (sous-)système de polarités $\{\square, \blacksquare\}$, qui modélise des ressources réutilisables. Dans la suite de ce manuscrit, nous utiliserons cette interprétation des systèmes de polarités des IG et des GUP : les polarités utilisées dans ces formalismes sont des couples de polarités atomiques sur $\{\square, \blacksquare\}$.

De plus, le terme « polarité » référera, selon le contexte, soit aux polarités qui sont des éléments d'un système de polarités, soit aux instances de ces polarités qui sont attachées à des objets syntaxiques.

## 1.3 Formalismes grammaticaux lexicalisés polarisés

Intégrer les polarités à un formalisme grammatical revient à attacher des polarités aux objets des structures syntaxiques du formalisme.

**Définition 1.3.1** (Formalisme grammatical lexicalisé polarisé)**.** Un **formalisme grammatical lexicalisé polarisé** est un 8-uplet $(\mathcal{V}, \mathcal{S}, \mathcal{G}, \mathbf{anc}, \mathcal{SF}, \mathcal{J}, \mathcal{Q}, \mathbf{pol})$ où :

- $(\mathcal{V}, \mathcal{S}, \mathcal{G}, \mathbf{anc}, \mathcal{SF}, \mathcal{J})$ est un formalisme grammatical ;
- $\mathcal{Q} = (\mathcal{P}, \oplus, \leq, \mathcal{N})$ est un système de polarités monotone ;

- la fonction de **polarisation**, notée **pol**, est une application des objets des structures syntaxiques de $\mathcal{G}$ vers des éléments de $\mathcal{P}$ ;
- toute structure $F \in \mathcal{SF}$ est neutre : $\forall o_f \in \mathcal{O}_F, \mathbf{pol}(o_f) \in \mathcal{N}$ ;
- toute fonction d'interprétation $I \in \mathcal{J}$ respecte une **contrainte de neutralité** : dans toute analyse $(E, F, I)$, pour tout objet $o_f \in \mathcal{O}_F$, l'ensemble d'objets polarisés $\{o_1, \ldots, o_k\} \subset \mathcal{O}_E$ tel que $I(o_1) = \ldots = I(o_k) = o_f$ vérifie $\bigoplus(\mathbf{pol}(o_i), \ldots, \mathbf{pol}(o_k)) \in \mathcal{N}$.

La fonction **pol** associe à chaque objet au plus une polarité. Toute instance de polarité est donc identifiée de façon unique par un couple $(o, p), o \in \mathcal{O}_t, t \in \mathcal{G}$ et $p \in \mathcal{P}$.

La contrainte de neutralité sur la fonction $I$ garantit que les objets polarisés qui ont la même image par $I$ forment un ensemble neutre, c'est-à-dire que le résultat de la composition de leurs polarités est une polarité neutre.

Nous aurons besoin dans la suite de ce manuscrit de pouvoir référer à l'ensemble des instances de polarités attachées aux objets d'une structure syntaxique polarisée. Nous définissons donc l'application **pols** qui associe à toute structure syntaxique polarisée l'ensemble de ses instances de polarités $(o, p)$.

Plus simplement, nous utiliserons le raccourci de notation $p \in t_i$ pour désigner une polarité $p$ telle que $(o, p) \in \mathbf{pols}(t_i)$. Nous utiliserons également ce raccourci de notation pour les polarités atomiques d'une polarité composite : $p_k \in p$ désigne le $k$-ième élément d'une polarité $p$, et par transitivité, $p_k \in t_i$ désigne le $k$-ième élément d'une polarité composite $p \in t_i$.

## 1.4 Exemples de formalismes grammaticaux polarisés

Plusieurs formalismes entrent actuellement dans le cadre des formalismes lexicalisés polarisés. Certains d'entre eux, comme les arbres de dépendance de Nasr et les IG, utilisent explicitement des polarités dans leur formulation originale. La plupart des autres, comme les Grammaires Catégorielles (CG) et les Grammaires d'Arbres Adjoints (TAG) peuvent être vus comme des formalismes polarisés, comme le montre le cadre unificateur des Grammaires d'Unification Polarisées (GUP). Dans cette section, nous donnons l'intuition de la formulation polarisée de ces formalismes.

### 1.4.1 Arbres de dépendance de Nasr

Nasr propose un formalisme lexicalisé pour construire des arbres de dépendance à partir d'arbres élémentaires dont les nœuds sont soit noirs, soit blancs [Nas95]. Les couleurs utilisées correspondent aux polarités noires ■ et blanches □ du système de polarités des GUP : tout nœud blanc doit être composé avec un nœud noir ; un nœud noir peut être composé avec une infinité de nœuds blancs, mais pas avec un nœud noir. Dans ce formalisme, les arbres d'une grammaire sont construits de la façon suivante. Chaque arbre élémentaire contient :

- un nœud noir qui représente son ancre ;
- autant de nœuds blancs que l'ancre a de compléments ;

– un nœud blanc pour le gouverneur de l'ancre, si cette dernière est un modificateur ;
– et autant de nœuds blancs que nécessaire pour décrire le contexte de l'ancre, c'est-à-dire les nœuds sur lesquels l'ancre impose des contraintes lexicales ou sémantiques.

### 1.4.2 Calcul de Lambek et Grammaires Catégorielles

La logique linéaire étend le calcul de Lambek qui est à la base des Grammaires Catégorielles (CG) [Abr96]. Chaque formule du calcul de Lambek a donc une formule équivalente en logique linéaire [Ret00]. Par exemple, le type du verbe transitif $NP\backslash S/NP$ est traduit par $-NP \parr +S \parr -NP$, qui équivaut à la formule de logique linéaire $NP^{\perp} \parr S \parr NP^{\perp}$. Les polarités de la logique linéaire correspondent au sous-système de polarités $\{+, -, \blacksquare\}$, avec $\blacksquare$ pour les liens axiomes. Une analyse syntaxique revient à créer un réseau de démonstration. Dans ce cadre, la création d'un lien axiome équivaut à la neutralisation de polarités.

### 1.4.3 Grammaires d'Arbres Adjoints polarisées

Les Grammaires d'Arbres Adjoints fondées sur les structures de traits, notées FTAG pour *Feature Structures based Tree Adjoining Grammars*) [VSJ88] sont une version des Grammaires d'Arbres Adjoints (TAG) qui repose sur l'unification de structures de traits. Une FTAG consiste en un ensemble d'arbres élémentaires sur lesquels sont définies deux opérations de composition : la substitution et l'adjonction. Il y a deux types d'arbres : les arbres initiaux et les arbres auxiliaires. La substitution insère un arbre $t$ de racine $r$ à un nœud feuille $l$ d'un autre arbre $t'$, à la condition que $l$ soit marqué comme un lieu de substitution et que les structures de traits de $l$ et $r$ soient compatibles. L'adjonction insère un arbre auxiliaire $t$ dans un arbre $t'$ en séparant en deux un nœud $n$ de $t'$, à la condition que les structures de traits des nœuds tête et pied de $t$ soient compatibles avec les structures de traits *top* et *bottom* de $n$.

La procédure de polarisation des FTAG [BGP04, Kow07] utilise le sous-système de polarités $\{+, -, \blacksquare\}$. L'idée de cette procédure est la suivante. Pour chaque arbre initial $t$, sa racine de catégorie $C$ est marquée comme ayant la polarité $+C$, et ses nœuds de substitution de catégorie $S$ sont marqués avec la polarité $-S$. Ainsi, la substitution d'un arbre $t'$ dans l'arbre $t$ correspond à la composition de la polarité $-S$ du nœud de substitution de $t$ avec la polarité $+S$ de la racine de $t'$.

Cette procédure peut être étendue aux arbres auxiliaires, afin de polariser l'opération d'adjonction. Il existe plusieurs façons de réaliser cette extension selon la formulation des FTAG utilisée : quasi-arbres [VS92], arbres tridimensionnels [Rog03] ...

### 1.4.4 Grammaires d'Unification Polarisées

La contribution la plus significative et la plus synthétique à l'étude des formalismes polarisés est apportée par Kahane au travers des Grammaires d'Unification Polarisées (GUP) [Kah06], que nous avons déjà évoquées dans ce chapitre. Les GUP sont un formalisme générique qui

étend les grammaires d'unification en leur intégrant un système de polarités. Kahane définit ce formalisme et montre qu'il permet de simuler fortement les arbres de dépendance de Nasr, les systèmes de réécriture algébriques, les TAG, HPSG, LFG et les grammaires synchrones.

Les travaux de Kahane montrent que, même si les systèmes de polarités apparaissent rarement dans la formulation originale des formalismes grammaticaux, de nombreux formalismes peuvent être vus comme des formalismes polarisés. Cette contribution est importante car elle établit la généralité des méthodes proposées dans ce manuscrit pour les formalismes grammaticaux polarisés. Les polarités sont en effet au cœur des contributions présentées au chapitre 3 et dans la partie II.

## Conclusion

Nous avons dressé dans ce chapitre le cadre général de notre travail, qui sont les formalismes grammaticaux lexicalisés polarisés. Dans un formalisme, l'ajout de polarités a pour effet de rendre explicites les modalités et les besoins de composition des structures. Comme l'ont montré notamment les travaux de Kahane, de nombreux formalismes grammaticaux peuvent être présentés comme des formalismes polarisés. Les méthodes que nous proposons dans ce manuscrit sont donc applicables à de nombreux formalismes grammaticaux.

# Chapitre 2

# Les Grammaires d'Interaction

Nous présentons dans ce chapitre un exemple de formalisme grammatical lexicalisé polarisé : les Grammaires d'Interaction (IG). Les IG sont le formalisme d'application et d'expérimentation de notre travail. Cette présentation reprend pour l'essentiel la présentation la plus récente du formalisme faite par Bruno Guillaume et Guy Perrier [GP09], que le lecteur intéressé pourra consulter.

Les IG se situent à l'articulation entre les trois grands courants de formalismes grammaticaux évoqués au chapitre précédent, et sont donc apparentées autant aux formalismes génératifs-énumératifs (GES) qu'aux formalismes fondés sur la théorie des modèles (MTS) [PS01]. Historiquement, les IG sont issues d'une généralisation des (réseaux de démonstration des) CG, dont elles étendent le système de polarités. Comme les TAG, les IG combinent des descriptions d'arbres syntagmatiques. Enfin, le caractère purement déclaratif et monotone des IG les inscrit dans la syntaxe MTS.

En section 2.1, nous commençons ce chapitre par une présentation générale des IG dans une perspective MTS. Nous définissons ensuite formellement, en section 2.2, le système de contraintes des IG, qui nous permet de construire ses objets initiaux que sont les *descriptions d'arbres polarisées* et finaux que sont les *arbres syntaxiques*. Le lien entre ces deux types d'objets est rendu explicite par la notion de *fonction d'interprétation*, qui associe à une description d'arbre polarisée les arbres syntaxiques qui sont ses modèles *minimaux* et *saturés*. En section 2.3, nous complétons cette présentation en adoptant une vision opérationnelle du formalisme, dans l'esprit des formalismes GES. Nous concluons cette présentation en section 2.4 par une description de la chaîne logicielle qui a été développée pour les IG et en particulier l'analyseur syntaxique LEOPAR.

## 2.1 Les IG : un formalisme syntaxique fondé sur la théorie des modèles

### 2.1.1 Arbre syntaxique

En IG, le résultat de l'analyse syntaxique d'une phrase est un arbre ordonné dont les nœuds représentent des syntagmes, décrits par des structures de traits.

La figure 2.1 contient l'arbre d'analyse de la phrase *« Paul le voit. »*.

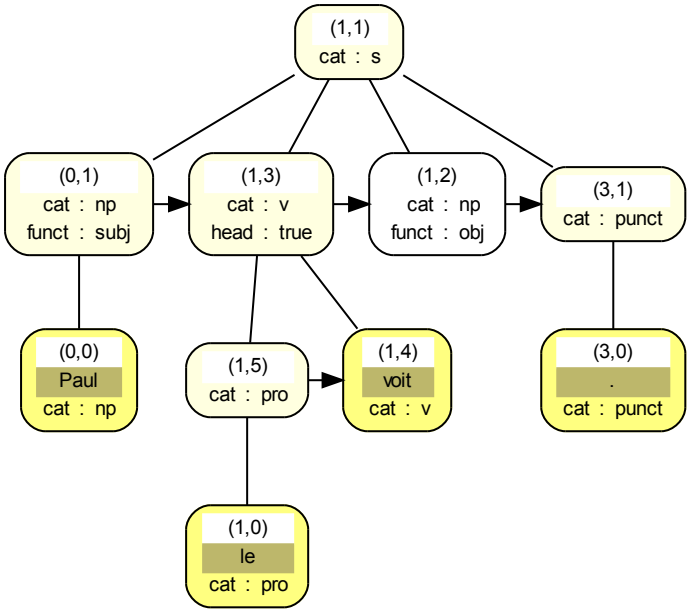



FIGURE 2.1 – Arbre d'analyse pour *« Paul le voit. »*

C'est un arbre ordonné, c'est-à-dire un ensemble de nœuds reliés par deux types de relations : la domination et la précédence. La *domination* indique les relations père-fils, elle est matérialisée par des traits pleins sans flèche et un étagement vertical. Ainsi, le nœud (1,1) domine les nœuds (0,1), (1,3), (1,2) et (3,1). La *précédence* indique l'ordre entre les nœuds frères, elle est matérialisée par des flèches. Ainsi, le nœud (1,5) précède le nœud (1,4).

Chaque nœud contient une structure de traits, c'est-à-dire un ensemble de couples `attribut : valeur` comme `cat : np`, `funct : subj` dans le nœud (0,1). Certains nœuds feuilles contiennent également une forme phonologique, celle-ci est alors inscrite sur fond marron, comme « Paul » dans le nœud (0,0). Les nœuds feuilles dont la forme phonologique est vide, comme (1,2), sont appelés *nœuds vides*. Les autres nœuds ont pour forme phonologique (non matérialisée ici) la concaténation des formes phonologiques de leurs fils, suivant leur ordre de précédence. Ainsi la forme phonologique de (1,3) est « le voit ».

Plusieurs raisons linguistiques et calculatoires amènent à considérer des variantes de cette formulation, dans lesquelles le résultat de l'analyse syntaxique n'est plus un arbre. Ainsi, une structure de graphe acyclique orienté (DAG) paraît plus adaptée à la modélisation de la coordination et des phénomènes liés à la ponctuation [DL10]. Plus globalement, Blache considère que la représentation de l'information linguistique requiert toute la généralité des structures de graphe [Bla07]. Ce besoin est particulièrement évident pour l'annotation de la langue parlée : une phrase comme *« Ma mère, sa moto, la selle, elle est noire »* [4] peut difficilement être annotée par un arbre syntaxique.

### 2.1.2 Description d'arbre polarisée

Les IG sont un formalisme syntaxique fondé sur la théorie des modèles, par conséquent une grammaire d'interaction particulière est un ensemble de contraintes qui sont vérifiées par toute phrase grammaticale. Les IG utilisent cinq types de contraintes : la domination, la précédence, la forme phonologique, les traits et les polarités. La domination et la précédence sont nécessaires à l'expression d'un arbre ordonné. La forme phonologique et les traits font partie du contenu des nœuds de l'arbre. Ces quatre types de contraintes permettent de décrire l'arbre syntaxique qui est le résultat de l'analyse d'une phrase en IG. Les polarités, ou contraintes de saturation, ont une nature plus combinatoire sur laquelle nous reviendrons plus loin.

Ces contraintes permettent de construire les objets manipulés par le processus d'analyse en IG que sont les *descriptions d'arbres polarisées* (notées DAP). Ainsi, la grammaire d'interaction strictement lexicalisée du français FRIGRAM [5], développée par Guy Perrier, associe à la phrase *« Paul le voit. »* la DAP de la figure 2.2. Cette DAP est l'union des DAP élémentaires associées à chacun des mots de cette phrase.

#### Les contraintes de domination et de précédence

Les contraintes de *domination* et de *précédence* correspondent aux relations présentes dans l'arbre final : ainsi, la contrainte de domination entre les nœuds (0,1) et (0,0) dans la DAP de la figure 2.2 implique que (0,1) doit dominer (0,0) dans l'arbre final.

Ces deux contraintes, domination et précédence, peuvent être sous-spécifiées dans une DAP : on parle de *domination large* et de *précédence large*. Par exemple, le nœud (1,3) précède largement le nœud (1,2), ce qui est marqué par la flèche pointillée verte. Cela signifie que (1,3) précède (1,2) mais que d'autres nœuds peuvent s'insérer entre eux.

La domination et la précédence peuvent également être bornées. Ainsi le double cadre autour du nœud (0,0) signifie que (0,0) doit être une feuille de l'arbre final, et le rectangle orange situé à droite à l'intérieur du nœud (3,1) signifie que (3,1) ne précède aucun nœud, c'est le fils le plus à droite du nœud (3,2).

4. Exemple de Joseph Le Roux.
5. http://wikilligramme.loria.fr/doku.php?id=frigram:frigram

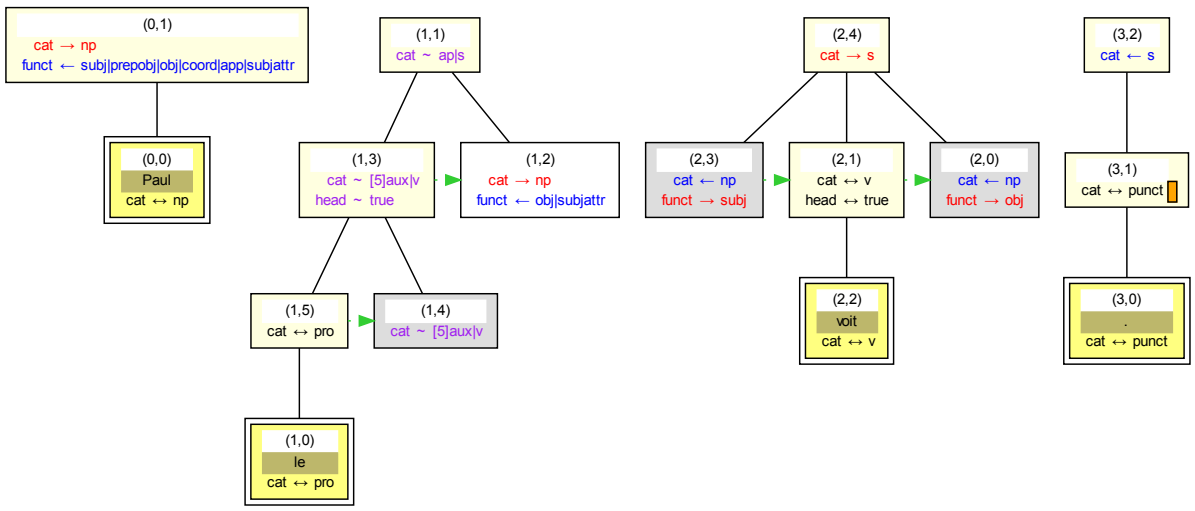



FIGURE 2.2 – DAP pour *« Paul le voit. »*

### Les contraintes de forme phonologique

Les contraintes de forme phonologique portent sur la forme phonologique des nœuds de l'arbre final. Ainsi, le fond uniformément blanc de (1,2) signale que ce nœud a une forme phonologique vide, c'est-à-dire que tous les nœuds feuilles de son sous-arbre dans l'arbre final ont une forme phonologique vide.

### Les contraintes de traits

Les contraintes de traits portent sur le contenu des nœuds de l'arbre final et sont exprimées dans les DAP par des traits polarisés : le trait polarisé `cat` $\rightarrow$ `np` dans (0,1) signifie que (0,1) doit contenir le trait `cat : np` dans l'arbre final. Les contraintes de traits peuvent être sous-spécifiées elles aussi : le trait polarisé `funct` $\leftarrow$ `subj|prepobj|obj|coord|app |subjattr` dans (0,1) signifie que (0,1) devra contenir un trait qui associe à l'attribut `funct` l'une des six valeurs `subj`, `prepobj`, `obj`, `coord`, `app` ou `subjattr`, dans l'arbre final.

### Les polarités ou contraintes de saturation

Les polarités modélisent la notion de *valence*, comme nous l'avons évoqué au chapitre précédent. L'originalité des IG est de généraliser la notion de valence et donc l'utilisation des polarités. En effet, au-delà du type des syntagmes, les autres informations grammaticales sont susceptibles d'être polarisées, comme la fonction syntaxique de ces syntagmes ou la négation. Dans des formalismes comme HPSG [PS94] ou FTAG [VSJ88], ces informations sont intégrées dans une

structure de traits. En IG, les structures de traits contiennent donc des traits polarisés.

Du point de vue de l'analyse syntaxique, les traits polarisés permettent de contrôler les identifications de nœuds de la description d'arbre polarisée dans l'arbre final. On distingue deux types d'interactions polarisées : les interactions *linéaires* et *non linéaires*, selon le type des polarités impliquées.

Ainsi le trait polarisé *positif* `cat` $\rightarrow$ `np` de (0,1) signifie que ce nœud doit être identifié à un autre nœud de la description d'arbre polarisée qui contient son trait dual *négatif* `cat` $\leftarrow$ `np`. C'est une interaction entre des polarités linéaires, donc une interaction linéaire.

Le trait polarisé *virtuel* `cat` $\sim$ `ap|s` de (1,1) signifie que ce nœud doit être identifié à un autre nœud de la description d'arbre polarisée qui contient un trait polarisé non virtuel, d'attribut `cat` et de valeur `ap` ou `s` (`cat` $\leftrightarrow$ `ap`, `cat` $\rightarrow$ `ap`, `cat` $\leftarrow$ `ap`, `cat` $\leftrightarrow$ `s`, `cat` $\rightarrow$ `s`, `cat` $\leftarrow$ `s`). C'est une interaction entre une polarité non linéaire et une autre polarité, donc une interaction non linéaire.

Les polarités sont essentielles aux IG pour deux raisons. Premièrement, les polarités fournissent un principe calculatoire à l'analyse des phrases mais également au filtrage des descriptions syntaxiques élémentaires, comme nous le verrons plus loin dans les chapitres suivants. Deuxièmement, l'utilisation méthodique des polarités dans une grammaire d'interaction permet d'interpréter cette grammaire dans une théorie linguistique. Ainsi la grammaire FRIGRAM peut être interprétée comme une grammaire de dépendance [MGP10].

### 2.1.3 L'arbre syntaxique et la fonction d'interprétation comme modèle d'une description d'arbre polarisée

Analyser une phrase en IG revient à associer à chacune de ses DAP possibles les arbres syntaxiques finaux qui satisfont les contraintes de cette DAP.

Pour reprendre la terminologie de la théorie des modèles, les nœuds d'un arbre syntaxique constituent le *domaine* (ou l'*univers*) *d'interprétation* d'une DAP. La correspondance entre les nœuds d'une DAP et ceux d'un arbre syntaxique prend la forme d'une fonction que l'on appelle *fonction d'interprétation* : elle associe à tout nœud de la DAP un nœud de l'arbre. Un arbre et une fonction d'interprétation, pris ensemble, constituent un *modèle* d'une DAP.

La fonction d'interprétation doit vérifier les contraintes de la DAP (domination, précédence, traits, saturation), ainsi que deux contraintes additionnelles : la contrainte de minimalité et la contrainte sur la précédence large.

**Contrainte 2.1.1** (Contrainte de minimalité)**.** La fonction d'interprétation ne peut ajouter ni nœud, ni domination immédiate, ni trait, ni forme phonologique à la DAP.

Dans notre exemple, la fonction d'interprétation qui va des nœuds de la figure 2.2 vers les nœuds de la figure 2.1 est donnée figure 2.3.

**Contrainte 2.1.2** (Contrainte sur la précédence large)**.** Si un nœud A précède largement un

| | | | |
|---|---|---|---|
| {(1,1), (2,4), (3,2)} → (1,1) | {(0,1), (2,3)} → (0,1) | {(1,3), (2,1)} → (1,3) | {(1,2), (2,0)} → (1,2) |
| {(3,1)} → (3,1) | {(0,0)} → (0,0) | {(1,5)} → (1,5) | {(1,4), (2,2)} → (1,4) |
| {(3,0)} → (3,0) | {(1,0)} → (1,0) | | |

FIGURE 2.3 – Fonction d'interprétation des nœuds de la figure 2.2 aux nœuds de la figure 2.1

nœud B dans la description d'arbre polarisée, alors les images de A et B par la fonction d'interprétation, dans l'arbre syntaxique, doivent être distinctes.

Notons que la contrainte de minimalité et la contrainte sur la précédence large reflètent une différence de statut entre les relations larges de domination et de précédence. La contrainte de minimalité interdit de transformer une domination large en domination immédiate lors de la construction du modèle ; de plus, deux nœuds liés par une domination large peuvent avoir la même image dans le modèle. Par contraste, la contrainte sur la précédence large interdit à deux nœuds liés par une précédence large d'avoir la même image dans le modèle ; une précédence large peut être transformée en précédence immédiate lors de la construction du modèle.

## 2.2 Définition formelle des IG

Nous définissons d'abord les structures produites par l'analyse : les arbres syntaxiques. Nous présentons ensuite le système de polarités utilisé, puis les descriptions d'arbres polarisées, qui sont les structures polarisées que l'analyse accepte en entrée et manipule. Nous introduisons ensuite les fonctions d'interprétation, qui font le lien entre les DAP et les arbres syntaxiques. Enfin, nous définissons ce qu'est une grammaire d'interaction.

### 2.2.1 Arbre syntaxique

**Définition 2.2.1** (Signature de traits)**.** Une **signature de traits** est définie par :
- un ensemble fini $\mathcal{F}$ de constantes appelées **noms de traits** ;
- pour chaque nom de trait $f$ dans $\mathcal{F}$, un ensemble fini $\mathcal{A}_f$ de constantes appelées **valeurs atomiques**.

Les traits sont construits relativement à une signature de traits.

**Définition 2.2.2** (Trait)**.** Un **trait** est une paire $(f, v)$ avec $f \in \mathcal{F}$ et $v \in \mathcal{A}_f$ ; dans ce qui suit, on écrit le trait $(f, v)$ comme $f : v$.

**Définition 2.2.3** (Structure de traits)**.** Une **structure de traits** est un ensemble de traits de noms différents.

Cette définition implique que les structures de traits en IG ne sont pas récursives, à la différence de formalismes comme HPSG.

**Définition 2.2.4** (Arbre syntaxique)**.** Un **arbre syntaxique** est un arbre totalement ordonné dont chaque nœud $M$ comporte :

- une structure de traits,
- une forme phonologique, notée $FP(M)$, qui est une chaîne de caractères ; cette chaîne peut être vide (notée $\epsilon$).

La forme phonologique est définie inductivement sur la structure de l'arbre. Un nœud $M$ qui n'est pas une feuille a pour forme phonologique la chaîne de caractères construite par concaténation des formes phonologiques de ses fils, séparées par des espaces. La forme phonologique d'un arbre est celle de sa racine. Ce processus de construction étant systématique, dans ce qui suit, la forme phonologique ne sera spécifiée que pour les nœuds feuilles.

**Notations** Dans les arbres syntaxiques, la relation de domination immédiate est notée $M \gg N$ ($M$ est le père de $N$), la relation de précédence immédiate est notée $M \prec\!\!\prec N$ ($M$ et $N$ ont même père et $M$ est situé juste avant $N$ dans l'ordre des frères).

Notons $\gg^*$ la clôture transitive et réflexive de $\gg$. Si $M \gg^* M'$ alors nous définissons le $chemin(M, M')$ comme la liste de nœuds de M à M' :

$$chemin(M, M') = \{N_i\}_{1 \leq i \leq n} \text{ tel que } \begin{cases} N_1 = M \\ N_i \gg N_{i+1} \quad \text{pour } 1 \leq i < n \\ N_n = M' \end{cases}$$

Notons $\prec\!\!\prec^+$ la clôture transitive[6] de la relation $\prec\!\!\prec$.

### 2.2.2 Système de polarités

#### Définition du système de polarités

Les IG, telles qu'elles ont été implantées, utilisent un système de polarités qui a été défini dans le but de modéliser la sensibilité des langues naturelles aux ressources. L'ensemble des polarités est $\mathcal{P} = \{=, \rightarrow, \leftarrow, \leftrightarrow\}$, parmi lesquelles $\mathcal{N} = \{=, \leftrightarrow\}$ est le sous-ensemble des polarités neutres :

- **inerte** (notée $=$) : un trait avec une polarité inerte n'est pas considéré comme une ressource, il se comporte comme un trait non polarisé et ne peut être combiné qu'avec d'autres traits inertes ;
- **positive** (notée $\rightarrow$) : un trait avec une polarité positive marque une ressource disponible ;
- **négative** (notée $\leftarrow$) : un trait avec une polarité négative marque une ressource requise ;
- **virtuelle** (notée $\sim$) : un trait avec une polarité virtuelle est non saturé : il attend d'interagir avec un autre trait non inerte et non virtuel ; les polarités virtuelles servent à exprimer des contraintes sur le contexte dans lequel une DAP peut apparaître ;

6. La contrainte sur la précédence large 2.1.2, page 28, implique que la réflexivité n'est pas une propriété pertinente pour la clôture transitive de la relation $\prec\!\!\prec$.

– **neutralisée** (notée $\leftrightarrow$) : un trait avec une polarité neutralisée est saturé ; ce trait ne peut pas être composé avec un trait inerte, positif, négatif ou neutralisé.

La composition des polarités est définie par le tableau 2.4, où $\perp$ marque l'impossibilité de composition de polarités.

| $\oplus$ | $=$ | $\sim$ | $\leftarrow$ | $\rightarrow$ | $\leftrightarrow$ | $\perp$ |
|---|---|---|---|---|---|---|
| $=$ | $=$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ |
| $\sim$ | $\perp$ | $\sim$ | $\leftarrow$ | $\rightarrow$ | $\leftrightarrow$ | $\perp$ |
| $\leftarrow$ | $\perp$ | $\leftarrow$ | $\perp$ | $\leftrightarrow$ | $\perp$ | $\perp$ |
| $\rightarrow$ | $\perp$ | $\rightarrow$ | $\leftrightarrow$ | $\perp$ | $\perp$ | $\perp$ |
| $\leftrightarrow$ | $\perp$ | $\leftrightarrow$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ |
| $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ |

FIGURE 2.4 – Système de polarités des IG

### Multi-ensembles de polarités équilibrés

À partir de cette définition, il est facile de vérifier qu'un multi-ensemble $\mathcal{M}$ de polarités est équilibré si et seulement si le résultat de sa composition est soit inerte, soit saturé, c'est-à-dire :

– $\bigoplus \mathcal{M}$ vaut $=$, ce qui signifie que le multi-ensemble ne contient que des polarités inertes et qu'il en contient au moins une ;

– ou $\bigoplus \mathcal{M}$ vaut $\leftrightarrow$, ce qui signifie que le multi-ensemble contient soit une unique polarité positive et une unique polarité négative, soit une unique polarité neutralisée.

Le tableau 2.5 récapitule les différentes configurations qui permettent d'obtenir un multi-ensemble équilibré.

| $\#_=$ | $\#_\rightarrow$ | $\#_\leftarrow$ | $\#_\leftrightarrow$ | $\#_\sim$ | $\bigoplus$ |
|---|---|---|---|---|---|
| $n \geq 1$ | 0 | 0 | 0 | 0 | $=$ |
| 0 | 1 | 1 | 0 | $n \geq 0$ | $\leftrightarrow$ |
| 0 | 0 | 0 | 1 | $n \geq 0$ | $\leftrightarrow$ |

FIGURE 2.5 – Configurations de multi-ensembles de polarités équilibrés en IG

La première ligne du tableau 2.5 correspond au cas où le résultat de la composition des polarités du multi-ensemble est la polarité inerte. Ce résultat ne peut être obtenu qu'en composant un multi-ensemble de polarités qui contient au moins une polarité inerte $=$, et aucune autre polarité.

Les deuxième et troisième lignes correspondent aux deux façons d'obtenir une polarité saturée : soit la polarité saturée est produite par composition d'une polarité positive et d'une polarité négative, soit la polarité saturée appartient au multi-ensemble de départ. La deuxième ligne décrit les multi-ensembles qui contiennent exactement une polarité positive et une polarité négative, aucune polarité inerte, aucune polarité saturée, et un nombre quelconque de polarités virtuelles. La troisième ligne décrit les multi-ensembles qui contiennent exactement une polarité

saturée, aucune polarité inerte, aucune polarité positive, aucune polarité négative, et un nombre indifférent de polarités virtuelles.

#### Sous-systèmes de polarités

L'analyse des configurations de multi-ensembles équilibrés révèle que le système de polarités des IG peut être découpé en deux sous-ensembles absolument indépendants $P_1 = \{=\}$ et $P_2 = \{\sim, \leftarrow, \rightarrow, \leftrightarrow\}$, dont chacun a sa propre polarité neutre : $N_1 = \{=\}$ et $N_2 = \{\leftrightarrow\}$.

Les algorithmes d'analyse et de filtrage des étiquetages grammaticaux qui ont été développés pour les IG, et que nous verrons dans la suite de ce manuscrit, exploitent le système de polarités que nous venons de présenter.

### 2.2.3 Description d'arbre polarisée

Nous définissons ici ce qu'est une description d'arbre polarisée (DAP). Sommairement, une DAP est un graphe, c'est-à-dire un ensemble de nœuds et de relations sur ces nœuds.

#### Trait polarisé et structure de traits polarisée

Alors que dans les arbres syntaxiques, les traits sont définis comme des couples nom-valeur, dans les DAP, les valeurs de traits peuvent être sous-spécifiées (par une disjonction de valeurs atomiques) et une polarité est attachée à chaque trait. Les traits sont construits sur une signature de traits augmentée de systèmes de polarités : pour chaque nom de trait $f \in \mathcal{F}$, en plus de l'ensemble $\mathcal{A}_f$ des valeurs atomiques, nous considérons un système de polarités $\mathcal{P}_f$.

**Définition 2.2.5** (Trait polarisé)**.** Un **trait polarisé** est défini par un triplet composé de :
- un nom de trait $f$ pris dans $\mathcal{F}$,
- une polarité $p$ prise dans $\mathcal{P}_f$,
- une valeur de trait qui est une disjonction de valeurs atomiques prises dans $\mathcal{A}_f$.

**Notations** Une valeur de trait est écrite comme une liste de valeurs atomiques séparées par le symbole *pipe* (noté `|` ). Le point d'interrogation (`?`) dénote la disjonction de toutes les valeurs de $\mathcal{A}_f$. Un trait polarisé est écrit comme la concaténation de ces trois composants. Par exemple, `cat` $\rightarrow$ `np|pp` et `funct` $\leftarrow$ `?` sont des traits polarisés.

Il est également possible de donner des contraintes supplémentaires sur les valeurs de traits en utilisant des coréférences.

**Définition 2.2.6** (Coréférence)**.** Une **coréférence** est un entier appelé **index** (noté $[i]$).

Par exemple, `funct` $\leftarrow$ `[4]obj|subjattr` est un trait coréférencé. La portée de ces coréférences est la DAP : une valeur de trait peut être partagée entre deux nœuds d'une même DAP et

elle est définie modulo renommage des indices. Lorsque nous considérons l'union disjointe d'un multi-ensemble de DAP élémentaires, nous supposons que les indices sont locaux à chaque DAP élémentaire du multi-ensemble. Dans la pratique, de telles unions disjointes peuvent requérir un renommage.

### Trait filtrant et structure de traits filtrante

Les traits filtrants sont utilisés pour représenter des contraintes sur les relations de domination.

**Définition 2.2.7** (Trait filtrant)**.** Un **trait filtrant** est un trait polarisé avec une polarité virtuelle ($\sim$) ou inerte ($=$) et sans coréférence.

Les traits filtrants ont pour but de contraindre les valeurs possibles de certains traits. C'est la raison pour laquelle seules les polarités qui sont des éléments neutres de l'opération de composition de polarités, $\sim$ et $=$, sont autorisées dans les traits filtrants [7].

**Définition 2.2.8** (Structure de traits filtrante)**.** Une **structure de traits filtrante** est un ensemble de traits filtrants qui ont des noms différents.

**Définition 2.2.9** (Compatibilité avec une structure de traits filtrante)**.** Une structure de traits $\phi$ est **compatible** avec une structure de traits filtrante $\psi$ si, pour tout nom de trait $f$ défini dans $\psi$ avec la valeur $v$, $\phi$ contient le trait $f$ avec une valeur incluse dans la disjonction $v$.

Le tableau ci-dessous compare les trois types de traits que nous avons définis jusqu'ici.

| | Trait | Trait polarisé | Trait filtrant |
|---|---|---|---|
| Emplacement | nœuds des arbres syntaxiques | nœuds des DAPs | relations de domination dans les DAPs |
| Polarité | non | oui (toutes) | $=$ ou $\sim$ |
| Valeur | atomique | disjonction | disjonction |
| Coréférences | non | oui | non |

### Description de nœud

Une **description de nœud** est constituée d'une structure de traits polarisée et, éventuellement, d'un type de nœud.

Les types de nœuds expriment des contraintes sur la forme phonologique des nœuds dans le modèle. Il y a trois types de nœuds :

– **ancre**($w$), avec $w$ une chaîne de caractères non vide : l'image de la description de nœud par la fonction d'interprétation doit être un nœud feuille de forme phonologique $w$ ;

7. Une alternative serait de définir les traits filtrants comme des traits non polarisés, identiques à ceux qui sont utilisés dans les arbres syntaxiques.

- **vide** : l'image de la description de nœud doit avoir une forme phonologique vide ; les descriptions de nœuds vides et les nœuds vides sont dessinés sur fond blanc ;
- **non vide** : l'image de la description de nœud doit avoir une forme phonologique non vide.

**Description d'arbre polarisée**

Une description d'arbre polarisée (DAP) est définie par un ensemble de descriptions de nœuds et un ensemble de relations sur ces nœuds. Nous considérons quatre types de relations entre les nœuds d'une DAP :

**Domination immédiate** La relation $M > N$ contraint l'image de $M$ à être le nœud père de l'image de $N$. Cette relation peut être précisée, en imposant la contrainte supplémentaire que l'image de $N$ soit le fils le plus à gauche (resp. le plus à droite) de l'image de $M$, ce qui est noté $M > \bullet N$ (resp. $M > N\bullet$). L'arité de l'ensemble des fils d'un nœud peut également être précisée : $M > \{N_1, N_2, \ldots, N_k\}$ impose que l'image de $M$ dans le modèle ait exactement $k$ fils qui soient les images des $N_i$. Cette contrainte d'arité n'impose pas d'ordre sur les $k$ fils de $M$.

**Domination** $M >^* N$ contraint l'image de $N$ à faire partie du sous-arbre enraciné à l'image de $M$. Cette relation de domination peut porter une contrainte supplémentaire sur les nœuds du chemin entre $M$ et $N$ dans le modèle : $M >^*_\psi N$, où $\psi$ est une structure de traits filtrante, impose que l'image de $N$ soit dans le sous-arbre enraciné à l'image de $M$ et que tous les nœuds du chemin entre les deux images contiennent une structure de traits qui soit compatible avec $\psi$.

**Précédence immédiate** $M \prec N$ contraint les images de $M$ et $N$ à être des nœuds fils du même nœud dans le modèle, l'image de $M$ devant être le frère gauche immédiat de l'image de $N$.

**Précédence** $M \prec^+ N$ contraint les images de $M$ et $N$ à être des nœuds fils du même nœud dans le modèle, l'image de $M$ devant précéder l'image de $N$ dans l'arbre ordonné ; cette précédence est stricte et les deux images doivent donc être distinctes.

Alors que dans les arbres syntaxiques, la relation $\gg^*$ est la clôture transitive et réflexive de la relation $\gg$ et la relation $\prec\!\!\prec^+$ la clôture transitive de la relation $\prec\!\!\prec$, dans les DAP, les relations $>^*$ et $\prec^+$ ne sont pas les clôtures des relations $>$ et $\prec$. Les quatre relations des DAP sont distinctes.

L'**union disjointe** $\coprod \mathcal{M}$ d'un multiensemble $\mathcal{M}$ de DAP est définie comme l'union disjointe de l'ensemble des nœuds et relations, avec renommage des coréférences si nécessaire. Par définition, $\coprod \mathcal{M}$ est une DAP.

### 2.2.4 Fonction d'interprétation

Le lien entre les DAP et les arbres syntaxiques repose sur la notion de fonction d'interprétation.

Un modèle d'une DAP $\mathcal{D}$ est un couple formé d'un arbre syntaxique $\mathcal{T}$ et d'une **fonction d'interprétation** $\mathcal{I}$ des nœuds $\mathcal{ND}$ de $\mathcal{D}$ vers les nœuds $\mathcal{NT}$ de $\mathcal{T}$ telle que :

**Satisfaction de domination immédiate**

- si $M, N \in \mathcal{ND}$ et $M > N$ alors $\mathcal{I}(M) \gg \mathcal{I}(N)$ ;
- si $M, N \in \mathcal{ND}$ et $M > \bullet N$ alors $\mathcal{I}(M) \gg \mathcal{I}(N)$ et $N$ est le fils le plus à gauche de $M$ ;
- si $M, N \in \mathcal{ND}$ et $M > N\bullet$ alors $\mathcal{I}(M) \gg \mathcal{I}(N)$ et $N$ est le fils le plus à droite de $M$ ;
- si $M, N_1, \ldots, N_k \in \mathcal{ND}$ et $M > \{N_1, \ldots, N_k\}$ alors $\mathcal{I}(M)$ a exactement $k$ fils différents $\{\mathcal{I}(N_1), \ldots, \mathcal{I}(N_k)\}$.

**Satisfaction de domination**

- si $M, N \in \mathcal{ND}$ et $M >^* N$ alors $\mathcal{I}(M) \gg^* \mathcal{I}(N)$ ;
- si $M, N \in \mathcal{ND}$ et $M >^*_\psi N$ alors $\mathcal{I}(M) \gg^* \mathcal{I}(N)$ et pour tout nœud $P$ dans $chemin(\mathcal{I}(M), \mathcal{I}(N))$, la structure de traits $\phi$ de $\mathcal{I}^{-1}(P)$ est compatible avec $\psi$.

**Satisfaction de précédence immédiate**

- si $M, N \in \mathcal{ND}$ et $M \prec N$ alors $\mathcal{I}(M) \lessdot \mathcal{I}(N)$. Par définition de la relation $\lessdot$ dans les arbres syntaxiques, $\mathcal{I}(M)$ et $\mathcal{I}(N)$ ont alors le même nœud père.

**Satisfaction de précédence**

- si $M, N \in \mathcal{ND}$ et $M \prec^+ N$ alors $\mathcal{I}(M) \lessdot^+ \mathcal{I}(N)$. $\mathcal{I}(M)$ et $\mathcal{I}(N)$ ont alors le même nœud père.

**Satisfaction de trait**

- si $M \in \mathcal{NT}$ et `f : v` est un trait de $M$ alors, pour tout nœud $N$ dans $\mathcal{I}^{-1}(M)$, soit le trait $f$ existe dans $N$ et la disjonction associée à $f$ contient $v$, soit $N$ ne contient pas le nom de trait $f$ ;
- si $M, N \in \mathcal{ND}$ sont deux nœuds qui contiennent chacun un trait $f$ avec la même coréférence, alors les valeurs associées à $f$ dans $\mathcal{I}(M)$ et $\mathcal{I}(N)$ sont identiques.

**Satisfaction de type de nœud**

- si $M \in \mathcal{ND}$ est de type ancre($w$) alors $\mathcal{I}(M)$ est un nœud feuille et $PF(\mathcal{I}(M)) = w$ ;
- si $M \in \mathcal{ND}$ est de type vide alors $PF(\mathcal{I}(M)) = \epsilon$ ;
- si $M \in \mathcal{ND}$ est de type non vide alors $PF(\mathcal{I}(M)) \neq \epsilon$.

**Saturation**

- le multi-ensemble de polarités associé à un nom de trait $f$ dans l'ensemble de nœuds de $\mathcal{I}^{-1}(M)$ qui contient le trait $f$ est équilibré.

**Minimalité**

- $\mathcal{I}$ est surjective : $\forall M \in \mathcal{NT}, \exists M' \in \mathcal{ND},\ \mathcal{I}(M') = M$ ;
- si $M, N \in \mathcal{NT}$ et $M \gg N$ alors il existe $M' \in \mathcal{I}^{-1}(M)$ et $N' \in \mathcal{I}^{-1}(N)$ tels que $M' > N'$ ;
- si $M \in \mathcal{NT}$ et `f : v` est un trait de $M$ alors au moins un nœud de $\mathcal{I}^{-1}(M)$ contient un trait de nom $f$ ;
- si $M \in \mathcal{ND}$ est un nœud feuille de forme phonologique non vide *phon*, alors $\mathcal{I}^{-1}(M)$ contient exactement un nœud de type ancre($w$).

Les quatre points qui définissent la minimalité garantissent que rien n'est ajouté lors de la construction du modèle. Ils contrôlent l'absence de création de nœud, de création de relation de domination immédiate, de création de trait et de création de forme phonologique.

Dans ce manuscrit, nous serons amené à considérer d'autres relations $\mathcal{R}', \mathcal{R}", \ldots$ entre les nœuds des DAP qui pourront être canoniquement interprétées comme relations entre les nœuds du modèle :

$$\text{pour } \mathcal{R} \in \{\mathcal{R}', \mathcal{R}", \ldots\},\ a\ \mathcal{I}(\mathcal{R})\ b \iff \exists a' \in \mathcal{I}^{-1}(a), \exists b' \in \mathcal{I}^{-1}(b) \text{ tels que } a'\ \mathcal{R}\ b'$$

### 2.2.5 Grammaires d'interaction

**Définition 2.2.10** (Grammaire d'interaction)**.** Une **grammaire d'interaction** $\mathcal{G}$ est un ensemble de descriptions d'arbres polarisées élémentaires (DAPE).

**Définition 2.2.11** (Langage d'arbres défini par une IG)**.** Le **langage d'arbres** défini par la grammaire $\mathcal{G}$ est l'ensemble des arbres syntaxiques qui sont les modèles des DAP produites par union disjointe d'un multi-ensemble de DAPE.

**Définition 2.2.12** (Langage de chaînes défini par une IG)**.** Le **langage de chaînes** défini par la grammaire $\mathcal{G}$ est l'ensemble des formes phonologiques des arbres du langage d'arbres défini par $\mathcal{G}$.

**Définition 2.2.13** (Analyse)**.** Un arbre syntaxique $\mathcal{T}$ est un **arbre d'analyse** d'une phrase $S$ pour une grammaire $\mathcal{G}$ si :

- il existe une DAP $\mathcal{D} = \coprod \mathcal{M}$, $\mathcal{M}$ étant un multi-ensemble de DAPE de $\mathcal{G}$,
  et une fonction d'interprétation $\mathcal{I}$ des nœuds $\mathcal{ND}$ de $\mathcal{D}$ vers les nœuds $\mathcal{NT}$ de $\mathcal{T}$,
  tels que $(\mathcal{T}, \mathcal{I})$ est un modèle de $\mathcal{D}$ ;
- et $PF(\mathcal{T}) = S$.

Un tel triplet $(\mathcal{D}, \mathcal{T}, \mathcal{I})$ est une **analyse** de la phrase $S$ pour la grammaire $\mathcal{G}$.

**Définition 2.2.14** (IG lexicalisée)**.** Une grammaire d'interaction $\mathcal{G}$ est dite **lexicalisée** si toute DAPE contient au moins un nœud de type ancre($w$).

**Définition 2.2.15** (IG strictement lexicalisée)**.** Une grammaire d'interaction $\mathcal{G}$ est dite **strictement lexicalisée** si toute DAPE contient exactement une ancre.

Dans le cas des grammaires strictement lexicalisées, le lien avec les mots du langage peut être vu comme une fonction qui associe un mot au sous-ensemble des DAPE dont l'ancre a ce mot pour forme phonologique.

## 2.3 La vision opérationnelle des IG

Jusqu'ici, la présentation que nous avons faite des IG était très orientée vers la syntaxe MTS. Concrètement, cette présentation se contente de décrire le lien entre la description d'arbre polarisée d'une phrase et son modèle, sans rien dire sur la façon dont on peut construire un tel modèle à partir d'une DAP. L'analyse est vue comme une fonction dont on n'évoque que les entrées et les sorties.

Un point de vue complémentaire consiste à considérer l'analyse comme un processus qui opère sur la DAP pour la transformer progressivement en un arbre syntaxique. Dans cette perspective, l'analyse correspond au processus de construction de modèle. C'est la vision opérationnelle des Grammaires d'Interaction.

### 2.3.1 Fusion de nœuds

La principale différence entre la DAP de la figure 2.2 et l'arbre syntaxique de la figure 2.1 est le nombre de nœuds que contiennent respectivement ces deux structures. Cela implique que, pour obtenir l'arbre, il faut fusionner des nœuds de la DAP. Nous définissons donc l'opération de *fusion de nœuds*, qui est une opération atomique sur les DAPs.

**Définition 2.3.1** (Fusion de nœuds)**.** Soit $\mathcal{D}$ une DAP et $N$ et $M$ deux nœuds distincts de $\mathcal{D}$, la DAP $\mathcal{D}_{\{N=M\}}$ est la description $\mathcal{D}$ dans laquelle les deux nœuds $N$ et $M$ sont remplacés par un unique nœud $NM$ et les relations de domination et de précédence sont modifiées en conséquence. La structure de traits polarisée du nœud $NM$ est obtenue à partir des structures de traits polarisées des nœuds $N$ et $M$, par unification des valeurs de traits et par composition des polarités.

Cette définition vérifie deux propriétés importantes.

**Propriété 2.3.1** (correction)**.** *Tout modèle $(\mathcal{T},\mathcal{I})$ de $\mathcal{D}_{\{N=M\}}$ est un modèle de $\mathcal{D}$.*

**Propriété 2.3.2** (complétude)**.** *Si $\mathcal{D}$ est non saturée et que $(\mathcal{T},\mathcal{I})$ est un modèle de $\mathcal{D}$, alors il existe deux nœuds $N$ et $M$ tels que $(\mathcal{T},\mathcal{I})$ est un modèle de $\mathcal{D}_{\{N=M\}}$.*

D'après la première propriété, la fusion de nœuds est une opération correcte pour la construction de modèles. D'après la deuxième propriété, la fusion de nœuds est une opération complète pour la construction de modèles, c'est-à-dire qu'elle permet de construire tout modèle $(\mathcal{T},\mathcal{I})$ d'une DAP $\mathcal{D}$. Plus concrètement, des applications successives de l'opération de fusion de nœuds permettent de construire une DAP saturée $\mathcal{D}'$ à partir de $\mathcal{D}$ ; il suffit ensuite de vérifier que $(\mathcal{T},\mathcal{I})$ est un modèle de $\mathcal{D}'$. Remarquons que la façon dont nous venons de définir la construction de

modèles à partir de l'opération de fusion de nœuds, correspond exactement à la description d'un problème NP-difficile. De fait, le problème de l'analyse en IG est NP-difficile [BGP03].

L'effet d'une fusion ne s'arrête pas aux nœuds fusionnés : il se propage à leur contexte, ce qui simplifie la DAP. Ainsi, si l'on fusionne deux nœuds $N$ et $M$ qui ont pour pères respectifs $N'$ et $M'$, alors dans la description $\mathcal{D}_{\{N=M\}}$, $N'$ et $M'$ sont tous deux des nœuds pères de $NM$. Un arbre syntaxique qui fait partie d'un modèle de $\mathcal{D}_{\{N=M\}}$ fait donc également partie d'un modèle de $\mathcal{D}_{\{N=M\}\{N'=M'\}}$, et l'on peut par conséquent fusionner les nœuds $N'$ et $M'$. Par exemple, dans la figure 2.2, la fusion des nœuds (1,4) et (2,2) entraîne la fusion de leurs nœuds pères (1,3) et (2,1). Mais ces nœuds ont eux-mêmes des pères, les nœuds (1,1) et (2,4), qui fusionnent donc à leur tour. Les relations de précédence propagent la fusion de la même façon que les relations de domination : si deux nœuds fusionnent, leurs voisins immédiats à gauche comme à droite doivent fusionner.

La propagation des contraintes a pour effet de réduire l'espace de construction du modèle. Plus la structure à construire est spécifique, plus la propagation est efficace. De fait, si les relations de domination et de précédence ne décrivaient pas un arbre mais un graphe acyclique orienté (DAG), la propagation aurait un impact moindre et la construction de modèles serait beaucoup plus coûteuse.

### 2.3.2 Polarités

La fusion de deux nœuds n'est possible que si leurs structures de traits polarisées sont compatibles, c'est-à-dire si, pour chaque trait, les valeurs des deux nœuds sont unifiables et leurs polarités sont composables.

La notion de polarité est centrale dans les IG. Les polarités présentent un double intérêt, linguistique et calculatoire : elles permettent d'écrire des descriptions qui sont faciles à interpréter linguistiquement et qui contiennent pourtant l'opérationnalité du processus d'analyse syntaxique.

#### Polarités linéaires

Les polarités des IG représentent des ressources. Une ressource consommable est soit disponible (polarité positive, notée $\rightarrow$) soit requise (polarité négative, notée $\leftarrow$). Dans un modèle, toute ressource disponible doit être consommée et tout besoin de ressource doit être satisfait. L'appariement d'une ressource disponible et d'une ressource requise correspond à une neutralisation des deux polarités linéaires. Le produit de cette neutralisation est la polarité saturée, notée $\leftrightarrow$.

En CG comme en IG, ce mécanisme de ressources est utilisé pour contrôler l'interaction d'un déterminant avec un nom, d'une préposition avec un syntagme nominal, ou encore d'un verbe, nom ou adjectif avec les arguments contenus dans son cadre de sous-catégorisation.

Ce mécanisme est également utilisé en IG pour contrôler d'autres types d'interactions, comme celles qui existent au sein des paires de mots grammaticaux comme « ne ... pas », entre une

ponctuation et une construction, entre un pronom réflexif et son verbe ou encore entre un auxiliaire et son participe.

### Polarités non linéaires et contexte

Le système de polarités des IG contient, en plus des polarités linéaires et saturée, une polarité non linéaire : la polarité *virtuelle* notée $\sim$. Plusieurs polarités virtuelles peuvent être composées ensemble. Pour saturer une polarité virtuelle, il faut la composer avec, soit une polarité saturée, soit une polarité positive et une polarité négative. La polarité virtuelle permet de modéliser le contexte d'un nœud.

La plus simple utilisation linguistique des polarités virtuelles est la modification. Par exemple, un nom peut être modifié par un adjectif. La simple présence du nom permet la présence de l'adjectif, sans que le second ne modifie la valence du premier.

Les polarités virtuelles permettent également de décrire des contextes beaucoup plus complexes que pour la modification. C'est le cas sur la figure 2.2 : les polarités virtuelles des nœuds (1,1), (1,3) et (1,4) spécifient le contexte dans lequel le pronom clitique *« le »* peut s'appliquer. Le pronom clitique (nœuds (1,0) et (1,5)) doit être situé à gauche de l'auxiliaire ou du verbe (nœud (1,4)), au sein du noyau verbal (nœud (1,3)), et il fournit à la phrase (nœud (1,1)) l'objet ou l'attribut du sujet (nœud (1,2)) de son verbe principal, qui est canoniquement à droite du noyau verbal. Les nœuds (2,4), (2,1) et (2,2) de la description d'arbre polarisée, qui sont associés au verbe *« voit »*, contiennent les polarités non virtuelles requises pour saturer les polarités virtuelles des nœuds (1,1), (1,3) et (1,4).

### Traits polarisés

Alors qu'en CG les polarités portent uniquement sur la catégorie des syntagmes, en IG n'importe quel trait qui décrit un syntagme peut être polarisé. Il est donc possible d'utiliser les polarités pour contrôler des informations syntaxiques autres que la catégorie. Ainsi, la grammaire FRIGRAM représente le mode comme un trait polarisé (appelé `mood`) afin de contrôler l'interaction entre l'auxiliaire et son participe. Un autre exemple est fourni par la négation, qui est généralement portée en français par l'interaction de deux mots comme « ne ... pas » ou « aucun ... ne ». La négation est donc modélisée par un trait polarisé (appelé `neg`).

Pouvoir polariser plusieurs traits des syntagmes n'est pas un mouvement anodin car les polarités guident le processus d'analyse syntaxique. En CG, un syntagme n'a qu'un trait polarisé qui est sa catégorie. Par conséquent, du point de vue de l'analyse, le syntagme peut être assimilé à sa catégorie. Le syntagme est alors une ressource. En IG, un syntagme a plusieurs traits polarisés, comme sa catégorie et sa fonction. Un syntagme dont plusieurs traits sont polarisés ne peut plus être assimilé à une ressource. Un syntagme doit plutôt être considéré comme le lieu de l'interaction entre des ressources.

C'est ce que montre la figure 2.6 qui est une partie de la description d'arbre polarisée associée à la phrase *« Jean pense que Marie vient. »* par la grammaire FRIGRAM. Dans cette phrase,

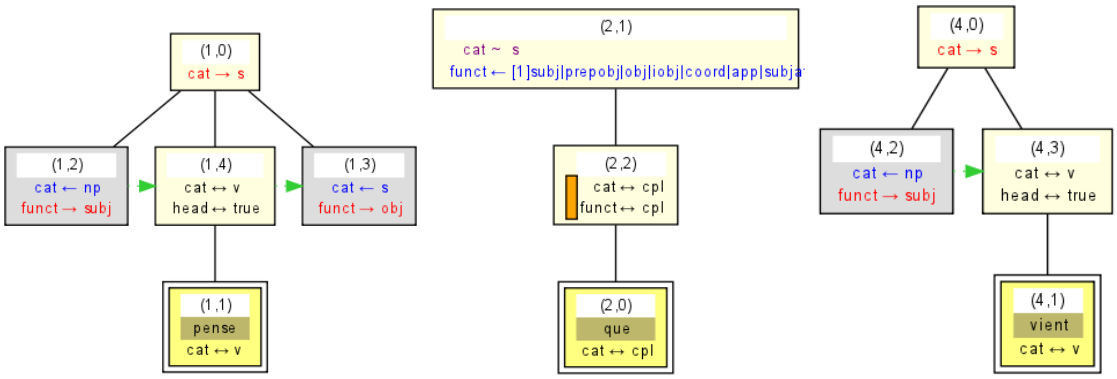



FIGURE 2.6 – DAP pour *« pense que ... vient »*

le verbe principal *« pense »* attend au nœud (1,3) une proposition (trait `cat` ← `s`) à laquelle il donnera la fonction d'objet (trait `funct` → `obj`). La catégorie de la proposition est fournie (`cat` → `s`) par le nœud (4,0) qui est apporté par le verbe *« vient »*. La fonction de la proposition est consommée (`cat` ∼ `s`, `funct` ← `obj`) par le nœud (2,1) qui est apporté par le complémenteur *« que »*. Le complémenteur *« que »* ne peut en effet être utilisé que pour introduire des propositions en position de complément, qui ont une fonction syntaxique. La polarité virtuelle du nœud (2,1) permet d'éviter la surgénération en contraignant l'interaction du trait `funct`.

L'objet phrastique de *« pense »* est donc le lieu de deux interactions →/← distinctes, l'une sur la catégorie et l'autre sur la fonction, qui impliquent trois nœuds de la description d'arbre polarisée. Ces trois nœuds sont apportés respectivement par le verbe principal, le complémenteur et le verbe du complément phrastique. La figure 2.7 montre le résultat de ces deux interactions, par fusion des nœuds (1,3), (4,0) et (2,1).

### 2.3.3 Superposition d'arbres

L'opération de fusion de nœuds et le système de polarités distinguent les IG des autres formalismes d'analyse syntaxique basés sur des arbres, comme les grammaires algébriques (CFG) et les Grammaires d'Arbres Adjoints (TAG).

Les CFG ont une unique opération de composition qui est la substitution. Elle consiste à remplacer une feuille $L$ d'un premier arbre par la racine $R$ d'un deuxième arbre. Les contraintes qui pèsent sur l'opération de composition sont localisées dans les nœuds $L$ et $R$. Les TAG possèdent deux opérations de composition : la substitution et l'adjonction. L'adjonction consiste à remplacer un nœud $N$ d'un arbre $U$ par un arbre auxiliaire $V$, dont le nœud racine $R$ et le nœud pied $F$ ont le même symbole que $N$. Les contraintes qui pèsent sur l'opération de composition sont localisées dans les nœuds $N$, $R$ et $F$. L'adjonction est en fait une substitution du deuxième ordre [Mön97].

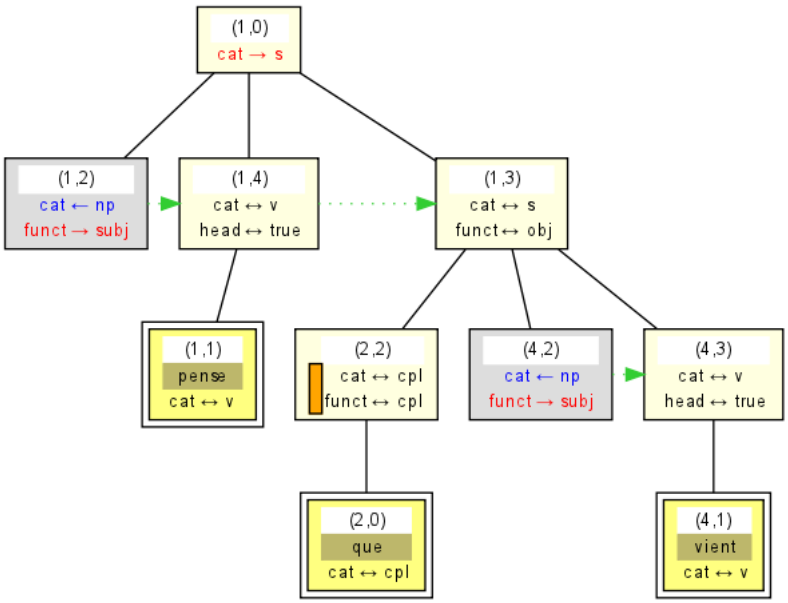



FIGURE 2.7 – DAP pour *« pense que ... vient »* après fusion de trois nœuds

En comparaison, l'opération de composition des IG, la fusion de nœuds, est plus souple et son effet est moins strictement localisé. En effet, alors que les opérations de composition des CFG et des TAG remplacent un nœud par un arbre entier, en IG l'opération de composition ne fait que fusionner des nœuds. Deux nœuds peuvent être fusionnés à la seule condition que leurs structures de traits polarisés soient composables. L'effet d'une fusion de nœuds se propage ensuite au contexte de ces nœuds. La propagation des contraintes a pour effet une superposition partielle des structures d'arbres autour des deux nœuds fusionnés.

### 2.3.4 Structures sous-spécifiées

Le deuxième trait distinctif des Grammaires d'Interaction, outre la superposition d'arbres, est la sous-spécification des structures arborescentes. Cette sous-spécification porte sur les relations de précédence et de domination contenues dans les descriptions d'arbres polarisées. Elle permet de décrire le contexte d'un nœud au-delà de son contexte immédiat, sans spécification de la distance à laquelle ce contexte se trouve dans le modèle.

Grâce à la sous-spécification de la relation de précédence, une seule description d'arbre polarisée suffit à modéliser les différents ordres possibles des arguments d'un verbe. La figure 2.8 contient la description d'arbre polarisée élémentaire associée au verbe *« demande »*, dans laquelle l'ordre de précédence entre les objets direct et indirect est sous-spécifié. L'objet direct (nœud (0,0)) et l'objet indirect (nœud (0,5)) sont tous deux, en position canonique, à droite du verbe (nœud (0,1)). Cette description est utilisée pour les deux phrases *« Jean demande à Marie une invitation »* et *« Jean demande une invitation à Marie »*.

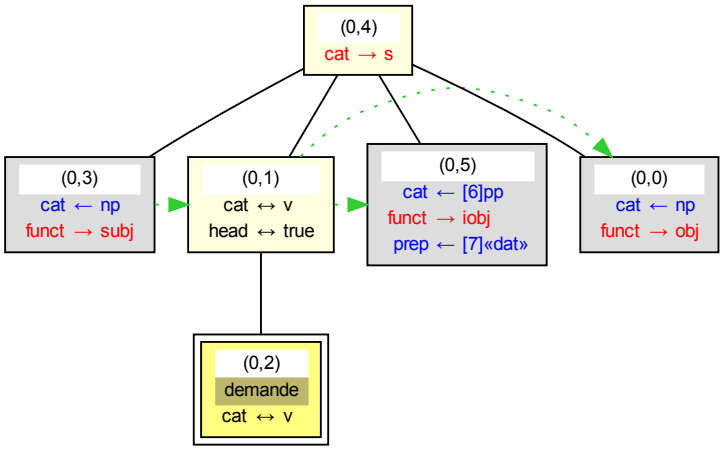



FIGURE 2.8 – DAP élémentaire pour *« demande »*

La sous-spécification de la relation de domination permet de modéliser les dépendances non bornées, telle celle qui existe entre le pronom relatif *« que »* et le verbe dont il est l'objet. La figure 2.9 montre la DAP élémentaire associée au pronom relatif *« que »* lorsque celui-ci est extrait. Cette DAP élémentaire est utilisée pour analyser la phrase *« Jean que Pierre croit que Marie aime vient »*. Le pronom *« que »* est extrait d'une proposition (nœud (1,7)) qui contient sa trace (nœud vide (1,9)). Cette proposition peut être l'objet direct (nœud (1,6)) du verbe de la proposition introduite par *« que »*, ou être imbriquée dans cet objet direct à une profondeur arbitraire. C'est cette profondeur arbitraire, voire nulle, que représente la relation de domination sous-spécifiée entre les nœuds (1,6) et (1,7).

Pour modéliser les contraintes d'îlots, la domination sous-spécifiée doit être contrôlée. Les IG utilisent pour cela des structures de traits filtrantes, qui sont des structures de traits polarisées qui ne contiennent que des polarités virtuelles et inertes. Une relation de domination large $M >^* N$ filtrée par une structure $\psi$ signifie que, dans le modèle, le nœud $M$ doit dominer $N$ et que tout nœud du modèle appartenant au chemin entre $M$ et $N$ doit être compatible avec $\psi$. Sur la figure 2.9, la structure filtrante est écrite à droite du trait pointillé qui représente la domination large entre les nœuds (1,6) et (1,7). Elle permet de garantir que l'extraction ne se fait qu'à travers des nœuds de propositions. La même DAP élémentaire est utilisée pour analyser la phrase *« Jean que Marie pense que Paul croit qu'Anne aime vient »*, qui contient un niveau d'imbrication supplémentaire.

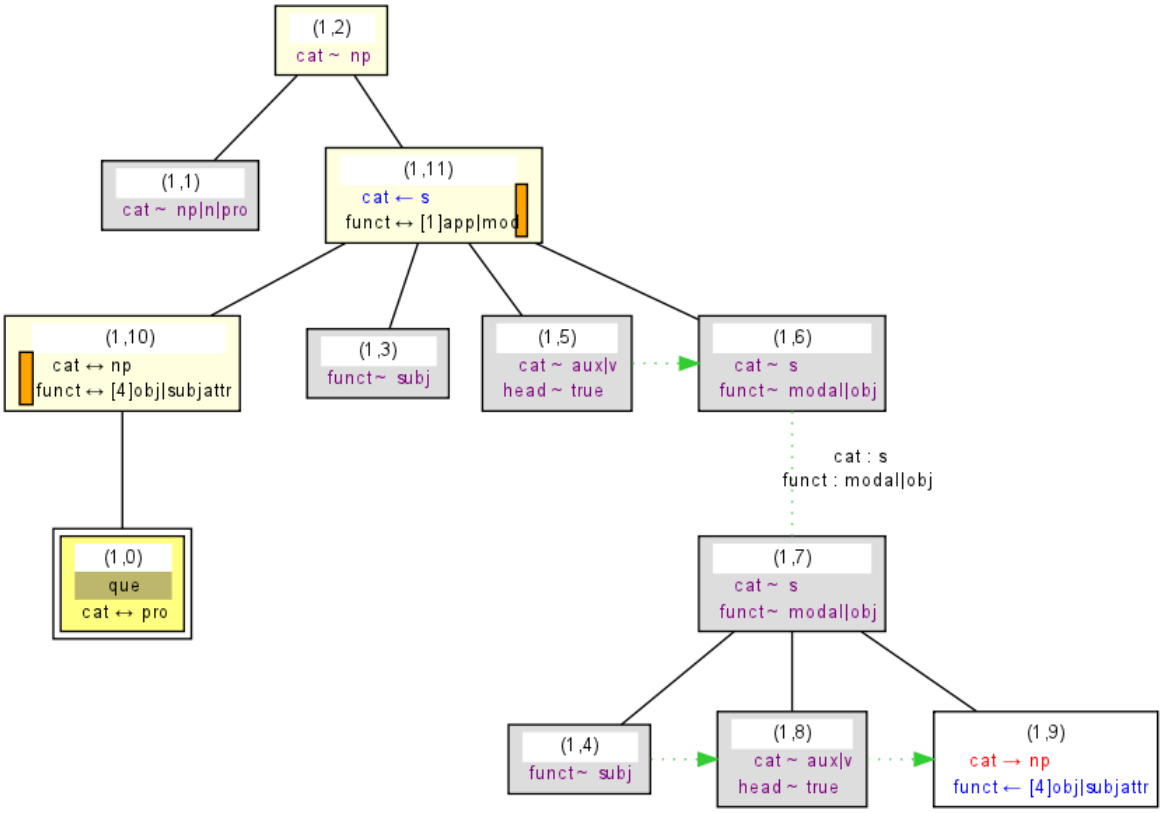



FIGURE 2.9 – DAP élémentaire pour *« que »*

## 2.4 Implantation

Les Grammaires d'Interaction, ainsi que leurs méthodes de filtrage et d'analyse, sont implantées dans l'analyseur syntaxique LEOPAR [8]. Cet analyseur facilite le développement de grammaires, comme la grammaire du français strictement lexicalisée FRIGRAM [9], et permet l'expérimentation et l'évaluation de ces grammaires sur corpus, comme le corpus arboré de Paris 7, appelé *French Treebank* (FTB) [ACT03].

Le développement d'une grammaire d'interaction de grande taille (plusieurs milliers de descriptions d'arbres polarisées pour FRIGRAM) repose sur le compilateur de métagrammaires XMG [DLP05b], dont la sortie est une grammaire d'interaction non ancrée. Afin d'ancrer cette grammaire d'interaction par des ressources lexicales indépendantes comme Dicovalence [vdEM03] ou Le*fff* [SCdlCB06], on associe à chaque DAP une interface, qui est une description syntaxique similaire à un *hypertag* [Kin00]. La grammaire d'interaction ancrée est alors produite en associant à chaque mot l'ensemble des DAP dont l'interface est compatible avec l'une des interfaces de ce mot dans le lexique.

Dans ce manuscrit, la version de la grammaire FRIGRAM que nous utilisons est celle du 23 juin 2011. Cette grammaire contient 3 788 descriptions d'arbres non ancrées. Elle couvre 88% des phrases grammaticales et rejette 85% des phrases agrammaticales du corpus TSNLP [LORP+96]. Contrairement aux grammaires documentées de façon exhaustive et figées comme la grammaire XTAG de l'anglais [XTA95], les principes de modélisation linguistique qui sous-tendent la grammaire FRIGRAM ne sont pas figés. Cette grammaire est en développement continu et elle connaît régulièrement des évolutions significatives.

Le développement continu de la grammaire FRIGRAM a des répercussions importantes sur les travaux présentés dans ce manuscrit. Il nous oblige en effet à proposer des méthodes qui ne font aucune hypothèse forte sur la grammaire ou sur le formalisme. Ainsi, les propositions que nous faisons au chapitre 3 et dans la partie II exploitent uniquement des propriétés générales du formalisme, et les travaux de la partie III exploitent des informations linguistiques sans rien présupposer de leur réalisation concrète dans une grammaire ou un formalisme donnés.

## Conclusion

Nous avons présenté dans ce chapitre les IG, qui sont un formalisme grammatical lexicalisé polarisé tel que nous l'avons défini au chapitre précédent. Les IG, avec l'analyseur LEOPAR et la grammaire FRIGRAM, constituent le support concret des travaux présentés dans ce manuscrit, qui sont définis et applicables à tout formalisme grammatical lexicalisé polarisé. Le développement continu de la grammaire FRIGRAM a un impact important sur la suite de ce manuscrit, en nous obligeant à proposer des méthodes qui ne font aucune hypothèse forte, ni sur la grammaire, ni sur le formalisme.

8. http://leopar.loria.fr
9. http://wikilligramme.loria.fr/doku.php?id=frigram:frigram

# Chapitre 3

# Le compagnonnage

Nous avons présenté dans le chapitre 1 le cadre général de la polarisation des formalismes grammaticaux lexicalisés et nous l'avons illustré au chapitre 2 avec le formalisme grammatical des Grammaires d'Interaction. Dans les formalismes grammaticaux polarisés, le processus d'analyse a pour but de produire une structure totalement neutre : l'analyse est dirigée par la saturation des instances de polarités non neutres attachées aux structures syntaxiques. Pour saturer une instance de polarité non neutre, il faut la composer avec une ou plusieurs autres instances de polarités. La présence d'une instance de polarité non neutre dans une analyse dépend donc de la présence d'une ou plusieurs autres instances de polarités.

Dans ce chapitre, nous explorons cette relation de dépendance qui existe entre des instances de polarités non neutres et d'autres instances de polarités. Cela nous amène à définir, en section 3.1, la notion de *compagnon* d'une polarité non neutre dans une analyse. Nous remontons la notion de compagnon du niveau de l'analyse à celui de la grammaire et nous considérons alors les *compagnons potentiels* d'une polarité dans une grammaire, en section 3.2, puis les *compagnons hypothétiques* d'une polarité dans une grammaire en section 3.3. La notion de compagnon hypothétique nous permet de définir la notion de *grammaire polarisée quasi-réduite*. Nous intégrons la directionnalité dans les compagnons hypothétiques en section 3.4. Des raisons pratiques nous amènent à calculer les compagnons hypothétiques sur la grammaire non ancrée, à partir de la section 3.5. Nous illustrons en section 3.6 sur un exemple la notion de compagnons hypothétiques dirigés d'une polarité dans la grammaire non ancrée. Nous présentons enfin, dans la section 3.7, quelques utilisations possibles des compagnons hypothétiques et de la quasi-réduction de grammaire pour le développement de grammaires et l'analyse syntaxique.

## 3.1 Dans une analyse : les compagnons

Nous avons vu au chapitre 1 qu'il était possible d'interpréter les polarités utilisées dans les formalismes grammaticaux polarisés comme des polarités composites, qui sont des n-uplets de polarités atomiques. Dans ce qui suit, nous situerons notre propos aux deux niveaux des polarités atomiques et des polarités composites.

### 3.1.1 La notion de compagnon

#### Polarités non neutres

Les différents systèmes de polarités reposent tous sur une partition entre polarités neutres et non neutres. C'est la neutralisation des polarités non neutres attachées aux objets des structures syntaxiques de la grammaire qui, dans le cadre des grammaires polarisées, dirige le processus d'analyse. Pour neutraliser une polarité, il faut la composer avec une ou plusieurs autres polarités jusqu'à atteindre un état saturé. Ce principe est valable aussi bien pour les polarités « composites » (ici les couples de polarités) que pour les polarités atomiques qui les composent. Dans le cas des GUP et des IG, il y a une polarité atomique non neutre, la polarité blanche □, qui doit être neutralisée par composition avec une polarité atomique noire ■. Une même polarité atomique noire peut neutraliser plusieurs polarités atomiques blanches, mais il est impossible de composer des polarités atomiques noires entre elles.

#### Compagnon d'une polarité atomique non neutre

Dans une analyse de phrase, la neutralisation d'un ensemble de polarités blanches par composition avec une unique polarité noire se traduit par le fait que la fonction d'interprétation $I$ associe la même image à un ensemble de polarités blanches et à exactement une polarité noire. Alternativement, on peut donc considérer que la fonction d'interprétation $I$ associe à chaque polarité atomique blanche de chaque description élémentaire utilisée, exactement une polarité atomique noire. Cette polarité atomique noire est appelée le *compagnon* de chacune de ces polarités atomiques blanches dans l'analyse.

**Définition 3.1.1** (Compagnon d'une polarité atomique)**.** Dans une analyse de phrase $(E, F, I)$, le **compagnon** d'une polarité atomique blanche □ (attachée à un objet d'un élément) de $E$ est l'unique polarité atomique noire ■ (attachée à un objet d'un élément) de $E$ qui a la même image que cette polarité atomique blanche par la fonction d'interprétation $I$.

#### Compagnon d'une polarité composite non neutre

L'extension de la définition 3.1.1 aux polarités composites est évidente, si l'on distingue autant de compagnons que la polarité composite comporte d'éléments.

**Définition 3.1.2** (k-ième compagnon d'une polarité composite)**.** Dans une analyse de phrase $(E, F, I)$, le **k-ième compagnon** d'une polarité composite $p = (p_1, \ldots, p_{k-1}, \square, \ldots, p_n)$ de $E$ est l'unique polarité composite $q = (q_1, \ldots, q_{k-1}, \blacksquare, \ldots, q_n)$ de $E$ qui a la même image que $p$ par la fonction d'interprétation $I$.

Les polarités composites des IG et GUP sont des couples de polarités atomiques. Chaque instance de polarité a donc au plus deux compagnons, notés *premier compagnon* et *second compagnon*, dans une analyse.

Ainsi dans une analyse en IG, une polarité :

1. $\leftarrow =$ $(\blacksquare, \square)$ a uniquement un second compagnon, qui doit être une polarité $\rightarrow =$ $(\square, \blacksquare)$ ;
2. $\rightarrow =$ $(\square, \blacksquare)$ a uniquement un premier compagnon, qui doit être une polarité $\leftarrow =$ $(\blacksquare, \square)$ ;
3. $\sim =$ $(\square, \square)$ a pour premier et second compagnons
   (a) soit une polarité $\leftarrow =$ $(\blacksquare, \square)$ et une polarité $\rightarrow =$ $(\square, \blacksquare)$,
   (b) soit une seule polarité $\leftrightarrow =$ $(\blacksquare, \blacksquare)$.

### 3.1.2 Analyse syntaxique dirigée par les compagnons

La notion de compagnon d'une instance de polarité nous permet de formuler plus explicitement ce qui dirige l'analyse syntaxique dans les formalismes grammaticaux polarisés. La formulation « classique », qui synthétise plusieurs variantes [Kah06, GP09], est la suivante.

**Thèse 3.1.1** (Formulation classique du principe directeur de l'analyse en grammaires polarisées)**.** Dans les formalismes grammaticaux polarisés, l'analyse syntaxique est dirigée par la saturation des instances de polarités non neutres attachées aux objets des structures syntaxiques.

Ce qui est implicite dans cette définition, c'est qu'une instance de polarité est saturée par composition avec certaines autres polarités. Ces dernières correspondent précisément à notre définition de l'ensemble des compagnons d'une instance de polarité non neutre. La notion de compagnon(s) concrétise ainsi l'idée de « saturateur(s) », partiels ou totaux, d'une instance de polarité non neutre. Nous pouvons par conséquent reformuler le principe directeur de l'analyse syntaxique des formalismes grammaticaux polarisés de la façon suivante.

**Thèse 3.1.2** (Reformulation du principe directeur de l'analyse en grammaires polarisées avec les compagnons)**.** Dans les formalismes grammaticaux polarisés, l'analyse est dirigée par la recherche de l'ensemble des compagnons des instances de polarités non neutres attachées aux objets des structures syntaxiques.

Cette reformulation a le mérite de rendre explicite le fait que le processus d'analyse est dirigé par des interactions entre des instances de polarités. Ce léger changement de perspective met en évidence une question charnière entre les grammaires polarisées et leur processus d'analyse : quelles structures syntaxiques d'une grammaire sont susceptibles d'être en interaction dans une analyse ?

## 3.2 Dans une grammaire : les compagnons potentiels

### 3.2.1 La notion de compagnon potentiel

La notion de compagnon est liée au résultat de l'analyse syntaxique : une polarité (attachée à un objet) d'une structure syntaxique a un compagnon dans une analyse de phrase. Au cours du processus d'analyse, si l'on considère une polarité attachée à un objet d'une structure élémentaire,

ce qui est utile, c'est de connaître l'ensemble des polarités (attachées à des objets des structures élémentaires) de la grammaire, qui sont susceptibles d'être son compagnon. Ces polarités sont appelées les *compagnons potentiels* de la polarité dans la grammaire.

**Définition 3.2.1** (Compagnon potentiel d'une polarité atomique)**.** Une polarité atomique noire ■ d'une structure $t_j \in \mathcal{G}$ est un **compagnon potentiel** d'une polarité atomique blanche □ d'une structure $t_i \in \mathcal{G}$ dans la grammaire $\mathcal{G}$ si et seulement s'il existe une analyse de phrase $(E, F, I)$, avec $t_i, t_j \in E$, dans laquelle ces deux polarités atomiques ont la même image par $I$.

Notons que dans cette définition, $t_i$ et $t_j$ peuvent être identiques, c'est-à-dire qu'il est possible d'une même structure contienne à la fois une polarité atomique et l'un des compagnons potentiels de cette dernière.

La définition 3.2.1 peut être étendue aux polarités composites de façon triviale, en procédant de la même façon que pour la définition du compagnon.

### 3.2.2 Réduction des grammaires polarisées

De la définition 3.2.1, on peut déduire que si une polarité atomique d'une description élémentaire n'a aucun compagnon potentiel dans la grammaire, alors il n'existe aucune analyse de phrase dans laquelle cette polarité atomique est neutralisée. Par conséquent, il n'existe aucune analyse de phrase qui utilise la description élémentaire qui contient cette polarité atomique. Cette description élémentaire est donc inutile du point de vue de la capacité générative de la grammaire.

La détection des descriptions élémentaires inutiles dans une grammaire polarisée rappelle la détection des variables (ou non-terminaux) inutiles dans une grammaire algébrique, qui permet de réduire cette grammaire algébrique [Car08]. Dans une grammaire algébrique, une variable est utile si elle est productive (i.e. elle engendre un langage non vide) et accessible (i.e. elle est dérivable de l'axiome et son contexte engendre un langage non vide). Les notions de variable utile et de grammaire réduite ainsi que les algorithmes de détection et de réduction correspondants ont été transposés dans le cadre, plus général que les grammaires algébriques, des Grammaires à Concaténation d'Intervalles (RCG) [Bou98]. Or, les LIG, les TAG et les HG peuvent être codées en RCG. Par conséquent, les notions et les algorithmes que nous venons de mentionner sont définis pour ces formalismes, qui appartiennent tous au courant GES. Ces notions et algorithmes ne sont cependant pas définis pour tous les formalismes, et en particulier pour les formalismes MTS. Dans le cadre des formalismes polarisés, la notion correspondante serait celle de *grammaire polarisée réduite*, qui serait une grammaire qui ne contient aucune description élémentaire inutile.

**Définition 3.2.2** (Grammaire polarisée réduite)**.** Une grammaire polarisée $\mathcal{G}$ est dite **réduite** si pour toute description élémentaire $t_i \in \mathcal{G}$, pour toute polarité atomique $p_k = \square, p_k \in t_i$, il existe une polarité atomique $q_k = \blacksquare, q_k \in t_j, t_j \in \mathcal{G}$ telle que $q_k$ est un compagnon potentiel de $p_k$.

Cependant, les définitions des compagnons potentiels et des grammaires réduites semblent difficilement exploitables en dehors des formalismes GES codables en RCG. Or, les formalismes grammaticaux utilisés pour les langues naturelles sont très divers. Dans le cas le plus général, calculer les compagnons potentiels requiert d'énumérer toutes les analyses que la grammaire est capable de générer, c'est-à-dire de tester toutes les combinaisons possibles des descriptions élémentaires de la grammaire. Pour une grammaire de production infinie, il est impossible de calculer ces compagnons potentiels.

## 3.3 Dans une grammaire : les compagnons hypothétiques

### 3.3.1 La notion de compagnon hypothétique

Il est toutefois possible de calculer une approximation par sur-ensemble des compagnons potentiels des polarités d'une grammaire. Nous allons définir ce sur-ensemble directement, mais il est plus facile d'en donner l'intuition en passant par son complémentaire.

Intuitivement, une polarité atomique noire ■ ne peut pas être un compagnon potentiel d'une polarité atomique blanche □, si aucune des compositions possibles de ces polarités ne produit de description qui a un modèle. Cela se produit dans deux cas de figure :

- soit la composition de ces deux polarités réussit, mais (la ou) les descriptions produites par cette composition ne participe à la construction d'aucun modèle,
- soit la composition de ces deux polarités échoue.

Le deuxième cas est facile à détecter : il suffit de vérifier la composabilité des polarités des descriptions élémentaires de la grammaire, deux à deux. C'est de cette façon que nous calculons les *compagnons hypothétiques* d'une polarité.

**Définition 3.3.1** (Compagnon hypothétique)**.** Étant données deux structures $t_i, t_j \in \mathcal{G}$, une polarité atomique noire $q_k = ■,\ q_k \in t_j$ est un **compagnon hypothétique** d'une polarité atomique blanche $p_k = □,\ p_k \in t_i$ si et seulement s'il existe une composition de $t_i$ et $t_j$ qui compose $p_k$ et $q_k$ et qui n'échoue pas.

L'appartenance à l'ensemble des compagnons hypothétiques d'une polarité ne garantit pas qu'il existe une analyse qui compose ces deux polarités. C'est néanmoins une condition nécessaire.

### 3.3.2 Cas d'échec de la composition de deux polarités

Voyons concrètement comment la composition de deux polarités peut échouer en IG. En IG une polarité atomique $p_k$ (resp. $q_k$) :

1. fait partie d'une polarité composite $p$ (resp. $q$),
2. qui est attachée à un trait $t$ (resp. $u$),
3. qui appartient à une structure de traits $f$ (resp. $g$),
4. qui appartient à un nœud $M$ (resp. $N$),

5. qui appartient à une DAP $D$ (resp. $E$).

On considère que la composition de $p_k$ et $q_k$ échoue si :

1. les polarités composites $p$ et $q$ ne sont pas composables,
2. les traits polarisés $t$ et $u$ ne sont pas composables,
3. les structures de traits polarisées $f$ et $g$ ne sont pas composables,
4. les nœuds du contexte immédiat de $M$ et $N$ dans $D$ et $E$ ne sont pas composables.

Afin d'illustrer simplement ces cas d'échec, nous utilisons ici des descriptions jouets qui n'ont pas de signification linguistique précise.

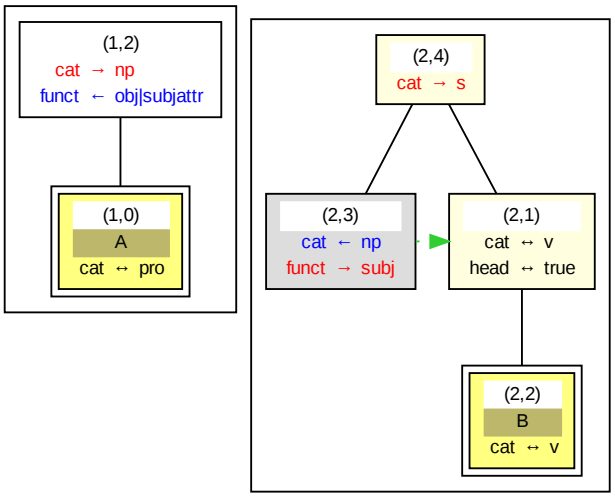



FIGURE 3.1 – Structures de traits non composables

Dans la figure 3.1, les structures de traits polarisées des nœuds (1,2) de la description de gauche et (2,3) de la description de droite ne sont pas composables. Concrètement, le nœud (1,2) contient un trait `cat` $\rightarrow$ `np` et le nœud (2,3) un trait `cat` $\leftarrow$ `np`. Ces deux nœuds ne sont cependant pas composables car les valeurs de leurs traits `funct` ne sont pas unifiables : (1,2) contient `funct` $\leftarrow$ `obj|subjattr` alors que (2,3) contient `funct` $\rightarrow$ `subj`. Le trait `cat` $\rightarrow$ `np` de (1,2) n'est donc pas un compagnon hypothétique du trait `cat` $\leftarrow$ `np` de (2,3).

La figure 3.2 permet d'illustrer deux configurations d'échec au niveau des contextes de nœuds. Supposons que les nœuds (3,2) de la description de gauche et (1,2) de la description de droite

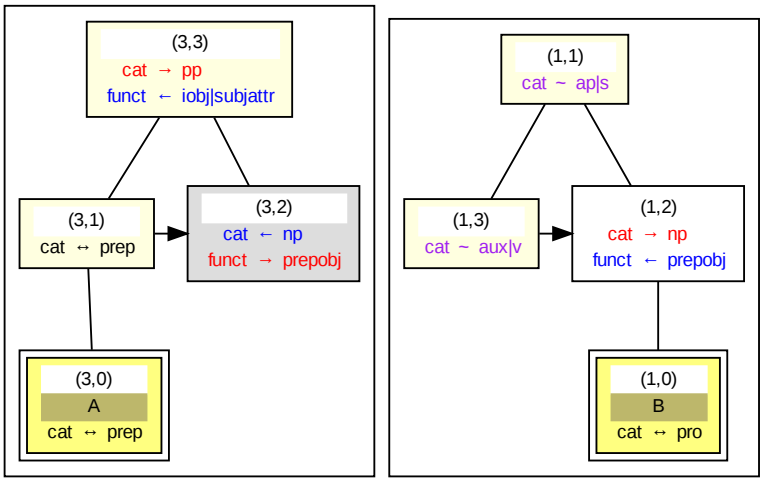



FIGURE 3.2 – Contextes non composables

fusionnent. À première vue, cette fusion paraît possible car les contenus des nœuds sont composables : (3,2) contient `cat` $\leftarrow$ `np` et `funct` $\rightarrow$ `prepobj`, alors que (1,2) contient `cat` $\rightarrow$ `np` et `funct` $\leftarrow$ `prepobj`. Mais, si ces nœuds fusionnent, leurs contextes doivent fusionner. Le contexte de chacun de ces nœuds consiste en un frère gauche immédiat et un père.

Les structures de traits des frères gauches immédiats (3,1) et (1,3) ne sont pas composables : (3,1) contient `cat` $\leftrightarrow$ `prep`, alors que (1,3) contient `cat` $\sim$ `aux|v`. Les nœuds (3,1) et (1,3) ne peuvent donc pas fusionner.

Les structures de traits des pères (3,3) et (1,1) ne sont pas composables non plus : (3,3) contient `cat` $\rightarrow$ `pp` et `funct` $\leftarrow$ `iobj|subjattr` alors que (1,1) contient `cat` $\sim$ `ap|s`. Les nœuds (3,3) et (1,1) ne peuvent donc pas fusionner non plus.

Au final, les contextes des nœuds (3,2) et (1,2) ne sont pas composables et par conséquent les nœuds (3,2) et (1,2) ne sont pas composables. Par conséquent, le trait `cat` $\rightarrow$ `np` de (1,2) n'est pas un compagnon hypothétique du trait `cat` $\leftarrow$ `np` de (3,2).

### 3.3.3 Quasi-réduction des grammaires polarisées

L'ensemble des compagnons potentiels d'une polarité étant un sous-ensemble des compagnons hypothétiques de cette polarité, nous définissons, sur le modèle de la section précédente, la notion de *grammaire polarisée quasi-réduite* et proposons la *quasi-réduction* d'une grammaire.

**Définition 3.3.2** (Grammaire polarisée quasi-réduite)**.** Une grammaire polarisée $\mathcal{G}$ est dite

**quasi-réduite** si pour toute structure $t_i \in \mathcal{G}$, pour toute polarité atomique $p_k = \square, p_k \in g$, il existe une polarité atomique $q_k = \blacksquare, q_k \in t_j, t_j \in \mathcal{G}$ telle que $q_k$ est un compagnon hypothétique de $p_k$.

**Proposition 3.3.1** (Quasi-réduction d'une grammaire polarisée)**.** *Pour toute grammaire polarisée $\mathcal{G}$, il existe une grammaire quasi-réduite $\mathcal{G}'$ telle que $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.*

*Démonstration.* La démonstration consiste à établir l'algorithme de quasi-réduction. Cet algorithme enlève de $\mathcal{G}$ toutes les descriptions élémentaires $g \in \mathcal{G}$ qui n'ont pas de compagnon hypothétique, en itérant jusqu'à atteindre un point fixe. L'itération jusqu'au point fixe est nécessaire car enlever des descriptions élémentaires peut avoir pour (seul) effet de faire disparaître tous les compagnons hypothétiques de certaines polarités. □

**Note** Les notions de compagnon hypothétique et de quasi-réduction de grammaire sont importantes pour la suite de ce manuscrit. Nous exploiterons ces notions dans les chapitres 5 et 6, pour proposer de nouvelles méthodes symboliques d'étiquetage grammatical applicables à tous les formalismes lexicalisés polarisés.

## 3.4 Intégration de la directionnalité dans les compagnons hypothétiques

Le degré de précision des compagnons hypothétiques peut être affiné en prenant en compte l'ordre linéaire des mots dans la phrase.

L'ordre des mots dans une phrase est une information syntaxique importante. Prenons pour exemples deux principes grammaticaux en français. Le premier principe est toujours valable : un déterminant doit précéder le nom qui le gouverne. Le deuxième principe est plus souple car il concerne la position canonique des éléments dans la phrase : un verbe transitif trouve son sujet à sa gauche et son objet direct à sa droite (c'est-à-dire que le français est, en typologie syntaxique, une langue SVO). Ces contraintes d'ordre sont codées dans les descriptions syntaxiques de la grammaire. Le codage est plus ou moins direct et plus ou moins élaboré selon les formalismes : les IG utilisent des relations de précédence immédiates et larges, alors que les Grammaires de Types Logiques (TLG) utilisent des connecteurs logiques orientés. Il est donc nécessaire de vérifier, pour chaque compagnon hypothétique d'une polarité, s'il peut remplir son rôle lorsqu'il est ancré dans la phrase à gauche ou à droite de la polarité considérée.

**Définition 3.4.1** (Compagnon hypothétique à gauche (resp. à droite))**.** Étant données deux structures $t_i, t_j \in \mathcal{G}$, une polarité atomique noire $q_k = \blacksquare$, $q_k \in t_j$ est un **compagnon hypothétique à gauche** (resp. à droite) d'une polarité atomique blanche $p_k = \square$, $p_k \in t_i$ si et seulement s'il existe une composition de $t_i$ et $t_j$ qui compose $p_k$ et $q_k$, qui n'échoue pas et qui produit une description qui admet au moins un modèle de phrase dans lequel $\mathbf{anc}(t_j)$ précède $\mathbf{anc}(t_i)$ (resp. $\mathbf{anc}(t_i)$ précède $\mathbf{anc}(t_j)$).

Les compagnons hypothétiques d'une polarité atomique $p$ forment donc deux sous-ensembles non nécessairement disjoints : ses compagnons hypothétiques à gauche, qui forment l'ensemble $\mathcal{L}_p$, et ses compagnons hypothétiques à droite, l'ensemble $\mathcal{R}_p$. L'ensemble des compagnons hypothétiques de la polarité atomique $p$ correspond alors au couple d'ensembles $(\mathcal{L}_p, \mathcal{R}_p)$.

Cette introduction de la non-commutativité dans les compagnons hypothétiques est également généralisée aux polarités composites. Une polarité composite $p = (p_1, \ldots, p_n)$ a autant d'ensembles de compagnons hypothétiques $(\mathcal{L}_{pk}, \mathcal{R}_{pk})$ que de polarités atomiques blanches $p_k, 1 \leq k \leq n$.

### 3.4.1 Illustration de l'impact de la directionnalité

Dans le cas des IG lexicalisées, les DAP sont ancrées, chaque ancre correspondant à une feuille de l'arbre syntaxique. Or, l'arbre syntaxique produit par une analyse en IG doit vérifier en particulier une condition : l'ordre linéaire de ses feuilles doit correspondre à l'ordre linéaire des ancres dans la phrase. Si la description d'arbre produite en composant deux polarités n'a aucun modèle qui respecte cette condition, alors on peut considérer que la composition de ces deux polarités échoue.

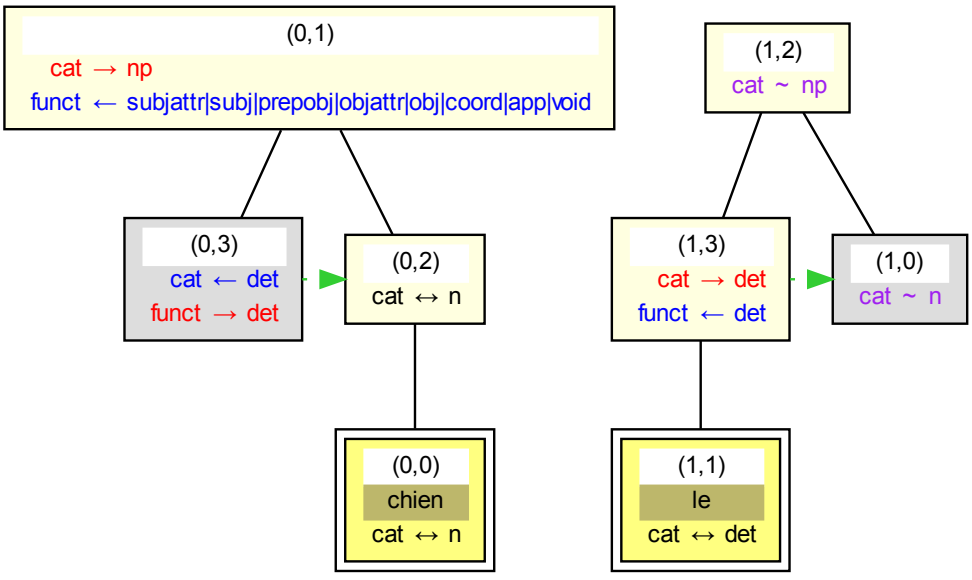



FIGURE 3.3 – DAP pour *« chien … le »*

Prenons l'exemple d'un déterminant, *« le »*, situé à droite d'un nom, *« chien »*. La figure 3.3 contient la DAP correspondante. Cette DAP est l'union des DAP élémentaires ancrées par *« chien »* et *« le »*, respectivement notées $DAPE_{chien}$ et $DAPE_{le}$. Pour que *« le »* soit le déter-

minant de « *chien* », il faut que le trait polarisé $p$ `cat` $\leftarrow$ `det` du nœud (0,3) de $DAPE_{chien}$ soit composé avec le trait $q$ `cat` $\rightarrow$ `det` du nœud (1,3) de $DAPE_{le}$. Cette composition produit la figure 3.4. Dans cette DAP, les nœuds (0,3) et (1,3), ainsi que leurs nœuds pères (0,1) et (1,2), ont fusionné. Or, la DAP issue de ces fusions de nœuds n'a aucun modèle dont la forme phonologique contient « *le* » après « *chien* », comme l'indique la précédence large (en pointillés verts) entre les nœuds (0,3) et (0,2). Cette DAP ne respecte donc pas la condition d'ordre sur les ancres. Par conséquent, la composition de $p$ et $q$ dans cette configuration échoue : aucune analyse de phrase contenant « *chien ... le* » ne contient la composition de $p$ et $q$. On dit que $q$ ne peut pas être le *compagnon droit* de $p$.

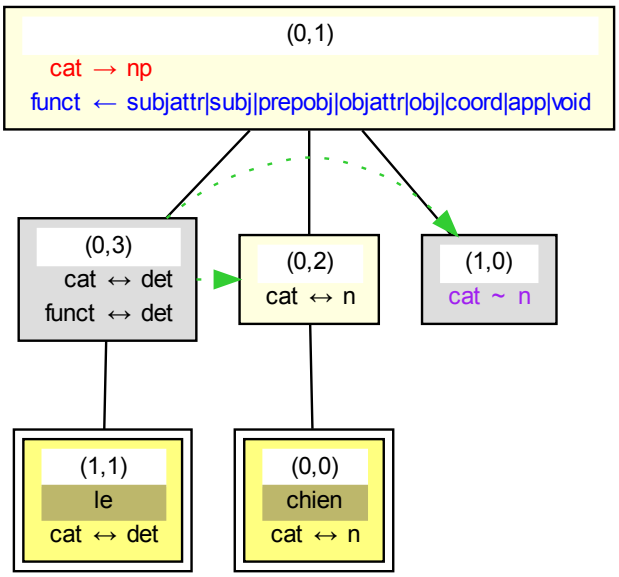



FIGURE 3.4 – DAP pour « *chien ... le* » après fusion des nœuds (0,3) et (1,3)

## 3.5 Utilisation de la grammaire non ancrée pour le calcul des compagnons hypothétiques

Dans la pratique, calculer les ensembles de compagnons hypothétiques des polarités d'une grammaire lexicalisée est un problème. Si nous prenons comme exemple la grammaire d'interaction du français FRIGRAM, cette grammaire contient 3 788 descriptions non ancrées. Chacune contient une dizaine de polarités atomiques non saturées, et donc, $m$ étant le nombre de polarités non saturées et $\mathcal{U}$ désignant la grammaire non ancrée, $m > 10.|\mathcal{U}| > 10^4$. L'ancrage de ces descriptions par croisement avec le lexique FRILEX produit de l'ordre d'un million de descriptions

ancrées, donc $m > 10.|\mathcal{G}| > 10^7$ sur la grammaire ancrée $\mathcal{G}$. Calculer à l'avance et manipuler $10^7$ ensembles de compagnons hypothétiques n'est pas réaliste.

La solution consiste alors à partitionner l'ensemble des polarités de $\mathcal{G}$ en classes d'équivalence et à travailler sur l'ensemble quotient ainsi créé. La création de classes d'équivalence est, elle aussi, une méthode classique dans les techniques d'approximation surensembliste [Ned00]. C'est ce que nous faisons ici : plutôt que de calculer les ensembles de compagnons hypothétiques des polarités de la grammaire ancrée $\mathcal{G}$, nous calculons ceux de la grammaire non ancrée $\mathcal{U}$. En effet, chaque polarité $q_k \in u, u \in \mathcal{U}$ peut être vue comme représentant une classe d'équivalence formée de l'ensemble des polarités correspondantes $p_k \in t_i, t_i \in \mathcal{G}$ des descriptions $t_i$ produites par l'ancrage de $u$.

Concrètement, le processus d'ancrage ajoute à une description non ancrée des informations de lemme, de genre et de nombre. C'est un processus monotone, qui ne change ni la structure, ni les polarités de la description non ancrée, mais précise certaines valeurs de traits. La monotonie de l'ancrage permet de garantir que le langage reconnu par la grammaire non ancrée est un surensemble strict du langage reconnu par la grammaire ancrée : $\mathcal{L}_\mathcal{G} \subset \mathcal{L}_\mathcal{U}$.

Remarquons que cette approximation surensembliste a une visée très modeste, qui est seulement de diminuer la taille des ensembles manipulés. La création de classes d'équivalence a ici un impact moins grand que dans les méthodes qui forment des classes d'équivalence dans le but d'utiliser un modèle de calcul moins puissant, par exemple pour transformer un automate à pile en un automate fini [Bou03, Ned00].

L'approximation surensembliste permise par la grammaire non ancrée est efficace : une grammaire non ancrée contient typiquement de l'ordre de $10^3$ descriptions élémentaires, donc de l'ordre de $10^4$ ensembles de compagnons hypothétiques (contre $10^7$ pour la grammaire ancrée).

## 3.6 Exemples de compagnons hypothétiques

Pour illustrer le détail des compagnons hypothétiques, considérons la grammaire jouet non ancrée $\mathcal{G}'$ de la figure 3.4, pages 60 à 63. Les ensembles de compagnons hypothétiques des polarités de la description `Det`, rappelée en figure 3.6, dans cette grammaire, sont indiqués dans la figure 3.5.

| | id($p$) | | | $k$ | $\mathcal{L}_{pk}$ | $\mathcal{R}_{pk}$ |
|---|---|---|---|---|---|---|
| 1 | Det | (0,3) | cat → det | 1 | ∅ | (NC, (0,3), cat ← det) |
| 2 | | | funct ← det | 2 | ∅ | (NC, (0,3), funct → det) |
| 3 | | (0,0) | cat ∼ n | 1 | ∅ | (NC, (0,2), cat ↔ n) |
| 4 | | | | 2 | ∅ | (NC, (0,2), cat ↔ n) |
| 5 | | (0,2) | cat ∼ np | 1 | (VTr, (0,0), cat ← np) | (VTr, (0,3), cat ← np), (VIntr, (0,2), cat ← np) |
| 6 | | | | 2 | (NC, (0,1), cat → np) | (NC, (0,1), cat → np) |

FIGURE 3.5 – Compagnons hypothétiques des polarités de la DAP `Det`

Une polarité composite $p$ est identifiée par le nom de la description, l'identifiant du nœud et le trait auquel elle est attachée. Les polarités atomiques contenues dans une polarité composite sont identifiées par leur rang, noté $k$, dans le n-uplet. À chaque polarité atomique sont associés ses ensembles de compagnons hypothétiques $\mathcal{L}_{pk}$ et $\mathcal{R}_{pk}$.

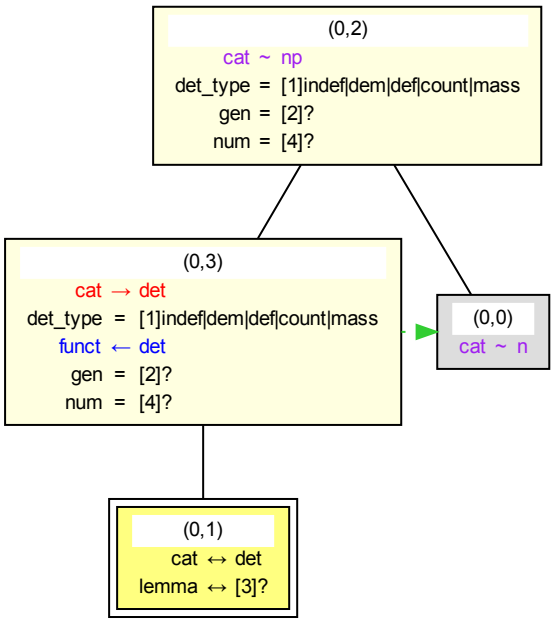



FIGURE 3.6 – `Det`

Par exemple, la première ligne de ce tableau signifie que la polarité composite → attachée au trait `cat : det` du nœud (0,3) de la description `Det` n'a aucun $1^{er}$ compagnon hypothétique à gauche dans la grammaire, et un seul $1^{er}$ compagnon hypothétique à droite qui est la polarité composite ← attachée au trait `cat : det` du nœud (0,3) de la description `NC`.

## 3.7 Utilisation des compagnons dans les chaînes d'analyse syntaxique

Nous concluons ce chapitre en montrant, dans cette section, l'intérêt concret des notions que nous avons introduites (compagnons, compagnons potentiels et hypothétiques, réduction et quasi-réduction des grammaires polarisées), pour les chaînes d'analyse syntaxique des formalismes grammaticaux polarisés. Nous relions ces notions à quatre applications : les tests de correction et complétude d'une grammaire, l'amélioration de la spécificité des compagnons hypothétiques d'une grammaire pour l'analyse syntaxique, la détection d'incohérences dans une grammaire et enfin l'intégration des compagnons hypothétiques à l'étiquetage grammatical et à

l'analyse syntaxique.

### 3.7.1 Correction et complétude d'une grammaire

Les compagnons hypothétiques fournissent un moyen de vérification et d'exploration d'une grammaire. Pour chaque objet polarisé de la grammaire, les développeurs de la grammaire peuvent vérifier quels sont ses compagnons hypothétiques à gauche, à droite ou des deux côtés indifféremment.

Cette possibilité est d'une grande aide lors des tests de correction et de complétude de la grammaire. Les tests de complétude ont pour objectif de garantir que la grammaire produit toutes les analyses attendues pour un ensemble de phrases. Dans l'idéal, cet ensemble comprend toutes les phrases du langage ; dans la réalité, on utilise une suite de tests représentative comme la TSNLP [LORP+96]. Au cours de ces tests, lorsque la grammaire ne produit pas une analyse attendue pour une phrase, il peut s'avérer difficile d'isoler la cause de cet échec.

Les tests de correction ont pour objectif de garantir que la grammaire ne produit pas d'analyse incorrecte. Cela signifie que la grammaire ne reconnaît pas comme grammaticale une phrase qui ne l'est pas et qu'elle n'attribue à une phrase grammaticale que les structures attendues.

### 3.7.2 Spécificité des compagnons hypothétiques pour l'analyse syntaxique

Vérifier la liste des compagnons hypothétiques d'une polarité est également utile pour améliorer la *spécificité* des compagnons hypothétiques d'une grammaire par rapport aux compagnons, c'est-à-dire au processus d'analyse. En statistique, la validité d'un test est déterminée par deux valeurs. La *sensibilité* est la capacité du test à donner un résultat positif dans tous les cas réellement positifs. La *spécificité* est la capacité du test à donner un résultat négatif dans tous les cas réellement négatifs.

Dans notre cas, le test correspond au calcul des compagnons hypothétiques :

- les *vrais positifs* ($VP$) sont les compagnons hypothétiques d'une polarité qui sont également ses compagnons potentiels,
- les *vrais négatifs* ($VN$) sont les polarités qui ne sont pas des compagnons hypothétiques et qui ne sont pas des compagnons potentiels,
- les *faux positifs* ($FP$) sont les compagnons hypothétiques d'une polarité qui ne sont pas ses compagnons potentiels,
- les *faux négatifs* ($FN$) sont les polarités qui ne sont pas des compagnons hypothétiques et qui sont des compagnons potentiels.

La sensibilité d'un test est [10] :

$$sensib = \frac{VP}{VP + FN}$$

10. La formule de la sensibilité est la même que celle du rappel.

La spécificité d'un test est :

$$specif = \frac{VN}{VN + FP}$$

La définition des compagnons hypothétiques implique que leur calcul ne produit aucun faux négatif. Ce test a par conséquent une sensibilité de 100%. A contrario, la spécificité du calcul des compagnons hypothétiques n'est pas garantie car les compagnons hypothétiques d'une grammaire peuvent contenir plus ou moins de faux positifs.

Une façon naïve mais sûre de détecter les faux positifs est d'examiner les ensembles de compagnons hypothétiques des polarités de la grammaire, à la recherche de compagnons hypothétiques inattendus au regard des principes linguistiques qui sous-tendent la grammaire. Souvent, les faux positifs sont produits par des structures initiales qui sont moins contraintes que dans l'intention du grammairien. Deux cas de figure se présentent alors :

- soit les autres structures de la grammaire compensent ce manque local de contrainte, et la grammaire ne surgénère pas,
- soit les autres structures ne compensent pas ce manque local de contrainte, et la grammaire surgénère.

Dans le premier cas, même si le manque de contrainte n'a pas d'impact sur le résultat, il a un impact clair sur les performances du processus d'analyse car il augmente inutilement la taille de l'espace de recherche. Dans les deux cas, il est possible que le grammairien ait été limité par l'expressivité du formalisme qu'il utilise.

Le relevé des faux positifs dans les compagnons hypothétiques d'une grammaire permet, dans le meilleur des cas, de corriger la grammaire. Le cas échéant, ces observations peuvent également suggérer des modifications dans la définition du formalisme, afin par exemple d'en augmenter l'expressivité.

### 3.7.3 Détection d'incohérences dans une grammaire

Les compagnons hypothétiques permettent de calculer, pour toute grammaire lexicalisée polarisée $\mathcal{G}$, la grammaire quasi-réduite $\mathcal{G}'$ qui lui est fortement équivalente. La grammaire quasi-réduite est le sous-ensemble de la grammaire, tel que toute polarité de toute structure a au moins un compagnon hypothétique dans la grammaire.

Or, il est rare qu'un linguiste écrive intentionnellement une grammaire qui contienne des structures orphelines, c'est-à-dire une grammaire qui ne soit pas d'emblée quasi-réduite. Souvent, ce cas de figure témoigne d'un oubli ou d'une erreur. Ces oublis ou erreurs sont d'autant plus fréquents, et la procédure de quasi-réduction de la grammaire est d'autant plus utile, que la grammaire est de grande taille : plusieurs milliers de structures initiales pour les grammaires LTAG [XTA95] et IG [Per07]. L'écriture de grammaires de cette taille peut être facilitée par l'utilisation d'un compilateur de méta-grammaires [Can96, DLP05a]. Cependant, ce compilateur ne vérifie pas, et donc ne garantit aucunement, que la grammaire produite est quasi-réduite.

Le test de quasi-réduction de la grammaire a été intégré à notre chaîne de développement de

ressources linguistiques. Une vérification automatique de la quasi-réduction est effectuée pour toute nouvelle version de la grammaire FRIGRAM du français. Cette vérification s'est avérée être un indicateur fiable d'oubli ou d'erreur dans la grammaire.

### 3.7.4 Intégration à l'étiquetage grammatical et à l'analyse syntaxique

Les compagnons hypothétiques permettent également de guider l'étiquetage grammatical et l'analyse syntaxique. En particulier, toutes les polarités d'un étiquetage qui n'ont qu'un seul compagnon hypothétique dans cet étiquetage peuvent être systématiquement associées ou composées avec ce compagnon. Ces compositions déterministes réduisent d'autant l'espace de recherche du modèle.

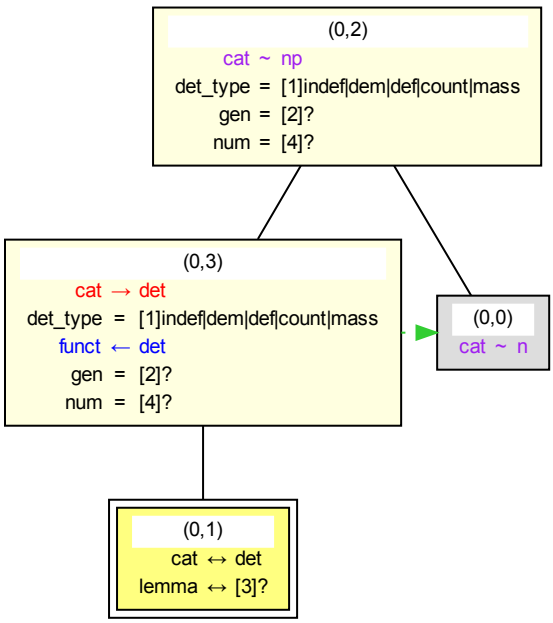
Dans la deuxième partie de ce manuscrit, nous utiliserons les notions de compagnons hypothétiques et de quasi-réduction d'une grammaire polarisée pour concevoir de nouvelles méthodes de filtrage des étiquetages grammaticaux.

## Conclusion

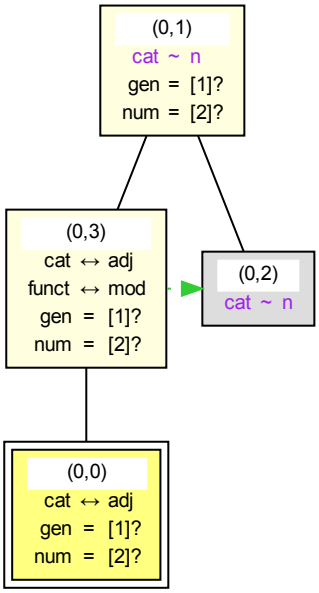
Dans ce chapitre, nous avons défini les notions de compagnon hypothétique d'une polarité dans une grammaire et de grammaire polarisée quasi-réduite. Des notions similaires, plus fortes, existent pour les formalismes GES codables en RCG. Les notions que nous définissons sont valables et peuvent être utilisées dans tout formalisme grammatical lexicalisé polarisé. Nous avons présenté quelques utilisations possibles de ces notions pour la conception et la maintenance de grammaires, ainsi que pour l'analyse syntaxique.

Dans la partie II, nous utiliserons ces notions pour concevoir de nouvelles méthodes de filtrage des étiquetages grammaticaux, applicables là encore à tout formalisme grammatical lexicalisé polarisé. Dans la partie III, nous proposerons une interface syntaxe - sémantique générique pour les formalismes grammaticaux lexicalisés dans laquelle les polarités joueront un rôle moindre.

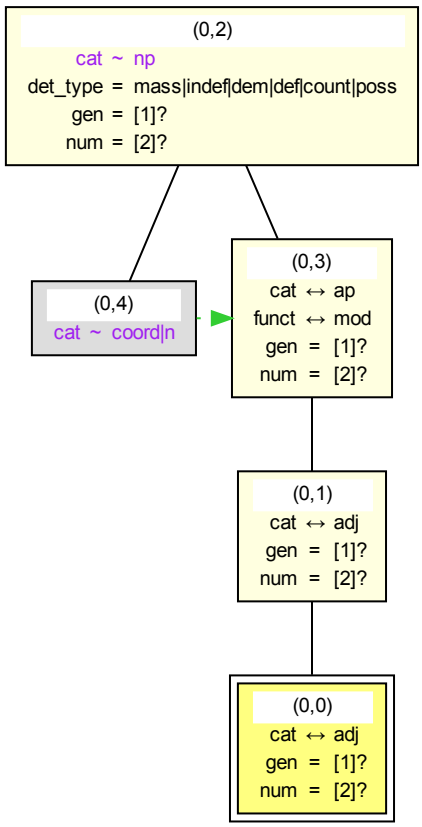
## 3.8 Annexe : grammaire jouet non ancrée



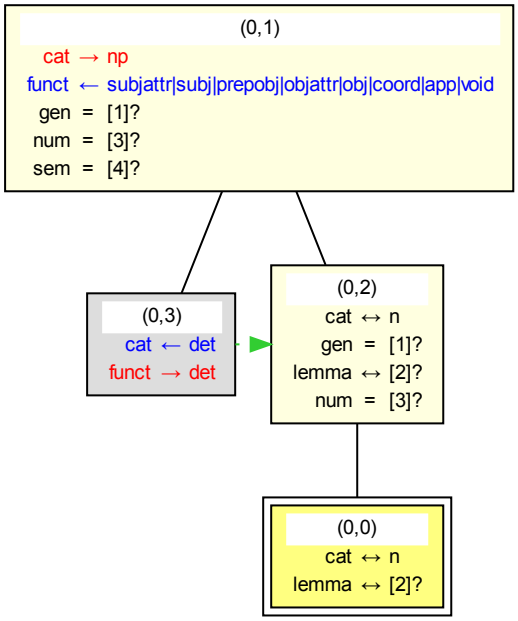

(a) `Det`



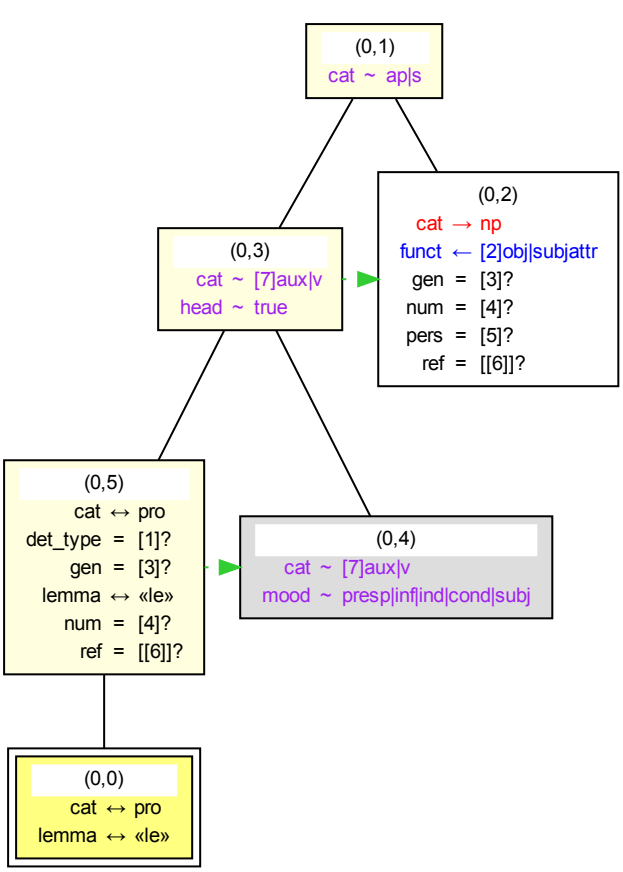

(b) `AdjG`


(0,2)
cat ~ np
det_type = mass|indef|dem|def|count|poss
gen = [1]?
num = [2]?
(0,4)
cat ~ coord|n
(0,3)
cat ↔ ap
funct ↔ mod
gen = [1]?
num = [2]?
(0,1)
cat ↔ adj
gen = [1]?
num = [2]?
(0,0)
cat ↔ adj
gen = [1]?
num = [2]?

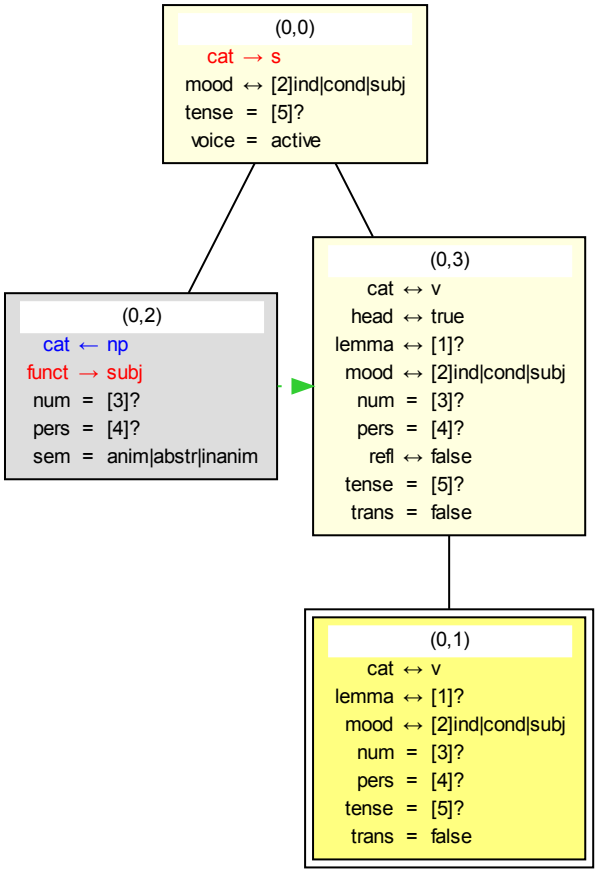
(c) `AdjD`


(0,1)
cat → np
funct ← subjattr|subj|prepobj|objattr|obj|coord|app|void
gen = [1]?
num = [3]?
sem = [4]?
(0,3)
cat ← det
funct → det
(0,2)
cat ↔ n
gen = [1]?
lemma ↔ [2]?
num = [3]?
(0,0)
cat ↔ n
lemma ↔ [2]?

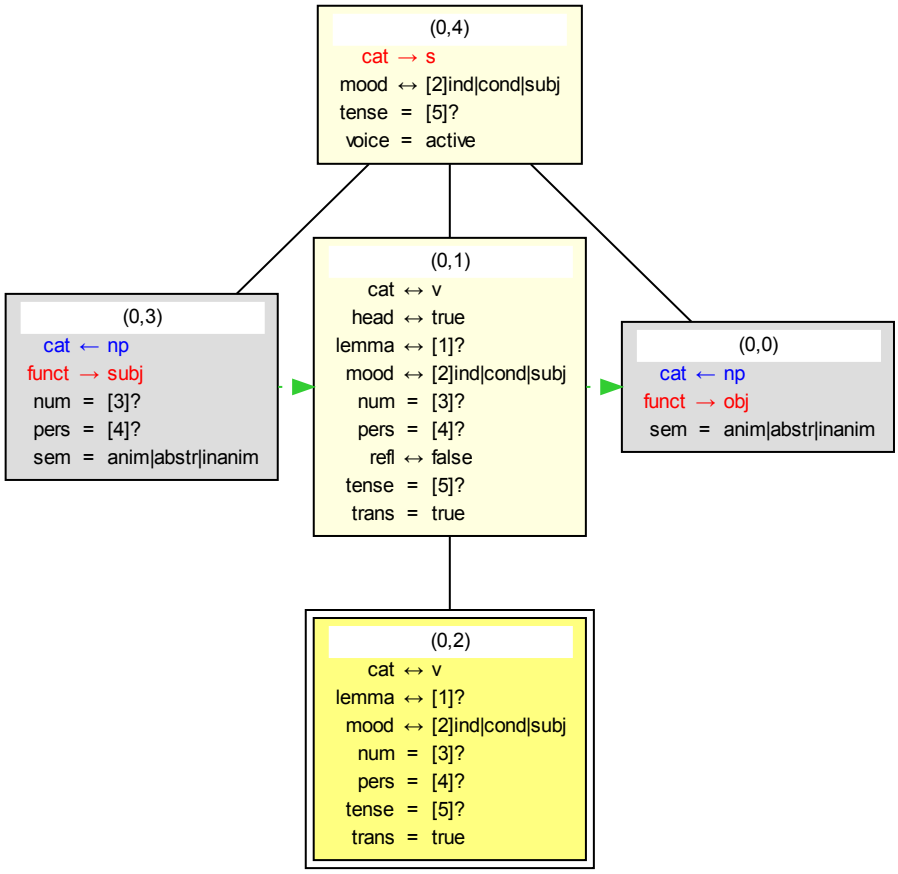
(d) `NC`




(e) `Clit`




(f) `VIntr`




(g) `VTr`

FIGURE 3.4 – Grammaire jouet non ancrée

# Deuxième partie

# Étiquetage grammatical symbolique pour les formalismes lexicalisés polarisés

# Chapitre 4

# Étiquetage grammatical symbolique pour les formalismes polarisés

L'analyse syntaxique des grammaires lexicalisées peut être découpée en deux phases : l'*étiquetage grammatical* ou *supertagging*, et l'analyse syntaxique proprement dite [SAJ88, Nas04, BJ10]. L'étiquetage grammatical sélectionne dans la grammaire les descriptions élémentaires utiles à l'analyse. Ensuite, l'analyse syntaxique proprement dite n'a plus qu'à composer les descriptions élémentaires de l'étiquetage. Nous nous intéressons dans ce chapitre à la phase d'étiquetage grammatical, dont nous présentons le principe en section 4.1. En section 4.2, nous nous concentrons sur les méthodes symboliques d'étiquetage grammatical. En section 4.3, nous illustrons ce principe en étiquetant grammaticalement une phrase exemple, dans le cadre des IG. Enfin, nous présentons en section 4.4, une méthode symbolique de filtrage des étiquetages grammaticaux qui exploite la polarisation. Cette méthode ne conserve que les étiquetages dont le bilan de polarités est équilibré.

## 4.1 Étiquetage grammatical

### 4.1.1 Origine

L'étiquetage grammatical ou supertagging a été proposé par Bangalore et Joshi [BJ99] pour les Grammaires d'Arbres Adjoints Lexicalisées (LTAG) et adapté à de nombreux formalismes lexicalisés [BJ10] comme les Grammaires Catégorielles Combinatoires (CCG) [SB09], les Grammaires de Dépendances à Contraintes (CDG) [Mar90] ou les Grammaires de Types Logiques (TLG) [Mor94]. Le supertagging étend le principe de l'étiquetage morphosyntaxique (*POS tagging* pour *part-of-speech tagging* en anglais). L'étiquetage morphosyntaxique attribue à un mot, par exemple *« chats »*, son étiquette morphosyntaxique, « nom commun masculin pluriel ». L'étiquetage grammatical attribue quant à lui à un mot un *supertag*, c'est-à-dire une description riche contenant des informations sur les propriétés lexicales et syntaxiques du mot.

### 4.1.2 Jeux d'étiquettes grammaticales

Chaque structure syntaxique d'une grammaire lexicalisée contient les propriétés combinatoires de son ancre. Ces propriétés sont essentiellement la liste de ses compléments requis et modifieurs potentiels, leur catégorie syntaxique, leur fonction, leur genre et leur nombre. La structure peut également comporter des informations topologiques sur ces compléments et adjoints, qui modélisent leur position dans l'ordre linéaire de la phrase par rapport à l'ancre. L'ensemble de ces informations décrit, donc contraint, le contexte dans lequel le mot qui ancre la structure peut apparaître. Ce contexte précis est ce que Joshi appelle le **domaine de localité** du mot ancre.

Prenons l'exemple des structures syntaxiques ancrées par le participe passé *« mangées »* en emploi verbal. La structure utilisée à la voix active implique la présence d'un objet direct féminin pluriel situé avant l'auxiliaire *« avoir »* : *« Jean les a mangées. »*. La structure utilisée à la voix passive implique la présence d'un sujet féminin pluriel et de l'auxiliaire *« être »* : *« Les pommes sont mangées. »*.

Une façon simple de définir un jeu d'étiquettes grammaticales est de faire correspondre une étiquette à chaque structure de la grammaire : (quasi-)arbre en TAG, type composé en CCG, DAP en IG... Étant donnée une phrase, le problème du choix des structures à utiliser pour l'analyser est alors réduit à un problème d'étiquetage.

La taille des jeux d'étiquettes grammaticales est en général bien plus grande que celle des jeux d'étiquettes morphosyntaxiques. À titre d'exemple, le jeu d'étiquettes morphosyntaxiques du Penn Treebank pour l'anglais compte 45 étiquettes, contre 425 étiquettes grammaticales dans la grammaire CCG de l'anglais extraite de ce même corpus [CCV06], et de l'ordre de 1000 à 5000 étiquettes pour les grammaires TAG [BJ10] et les Grammaires à Insertion d'Arbres (TIG) [BBN+09]. Le grand nombre d'étiquettes possibles rend le problème de l'étiquetage grammatical difficile. De plus, les erreurs d'étiquetage n'étant ni compensables ni défaisables par l'analyse, elles peuvent empêcher une phrase d'être analysée. Il est alors nécessaire de conserver dans l'étiquetage grammatical d'une phrase une certaine ambiguïté.

## 4.2 Étiquetage grammatical symbolique

Nous situons dans cette section les méthodes symboliques d'étiquetage grammatical par rapport aux méthodes statistiques. Nous présentons ensuite le principe qui est à la base des méthodes symboliques : l'approximation surensembliste. Enfin, nous motivons l'utilisation d'automates pour représenter des ensembles d'étiquetages dans les méthodes symboliques.

### 4.2.1 Approches statistique et symbolique

Les propriétés combinatoires que contient une structure syntaxique imposent des contraintes sur le contexte de cette structure. Par conséquent, l'étiquetage d'une phrase ne se fait pas de

façon indépendante pour chaque mot. Le problème de l'étiquetage grammatical est actuellement abordé sous deux angles complémentaires.

L'approche majoritaire est l'approche statistique [BJ10]. L'étiquetage est vu comme un problème de *classification*. Le principe de ces méthodes est d'utiliser un corpus de phrases déjà analysées syntaxiquement afin d'entraîner un classifieur, qui attribue ensuite à toute nouvelle phrase une ou plusieurs séquences d'étiquettes. L'hypothèse sous-jacente est que les dépendances entre étiquettes sont capturées par le modèle de classification. Les premiers modèles étudiés utilisaient des fréquences de n-grammes d'étiquettes [BJ99]. Cette approche est extrêmement efficace, cependant elle est susceptible d'écarter des étiquetages peu probables et néanmoins corrects.

La deuxième approche, à la suite des travaux de Boullier [Bou10], est l'approche symbolique. L'étiquetage est vu comme un problème de *filtrage* des étiquetages impossibles car incohérents. Étant donnée une grammaire lexicalisée, le processus d'étiquetage grammatical le plus naïf consiste à associer à chaque mot d'une phrase l'ensemble des descriptions élémentaires dont ce mot est l'ancre. Or, si un étiquetage grammatical ne respecte pas les contraintes imposées par chacune de ses descriptions syntaxiques élémentaires, il ne peut pas être utilisé pour produire une analyse de la phrase. La phase d'étiquetage filtre les étiquetages pour ne garder que ceux qui sont potentiellement cohérents, c'est-à-dire dont l'analyse syntaxique a une chance d'aboutir. Cette démarche a pour effet d'éviter de perdre la moindre analyse possible pour une phrase. C'est dans cette seconde perspective que s'inscrivent les travaux de cette thèse.

### 4.2.2 Étiquetage symbolique par approximation surensembliste

Les méthodes d'étiquetage grammatical symboliques, comme celle présentée par Boullier [Bou10], filtrent les étiquetages en calculant des approximations surensemblistes du langage engendré par la grammaire source.

La méthode de Boullier produit, à partir d'une grammaire LTAG $\mathcal{G}$ donnée, une grammaire CFG $\mathcal{G}_{CF}$, dont le langage engendré contient le langage engendré par la grammaire LTAG : $\mathcal{L}_{\mathcal{G}} \subset \mathcal{L}_{\mathcal{G}_{CF}}$. La grammaire $\mathcal{G}_{CF}$ ainsi produite est une approximation surensembliste, du point de vue du langage engendré, de la grammaire $\mathcal{G}$.

La production de la grammaire $\mathcal{G}_{CF}$ à partir de la grammaire $\mathcal{G}$ se fait en parcourant chaque arbre $t$ de la grammaire $\mathcal{G}$. Pour chaque arbre initial de la grammaire, on génère une règle de production de $\mathcal{G}_{CF}$ ; pour chaque arbre auxiliaire, on en génère deux : une pour la partie de l'arbre à gauche de l'épine et une autre pour la partie à droite de l'épine.

Une même règle de production $r \in \mathcal{G}_{CF}$ peut être générée à partir de plusieurs arbres $t \in \mathcal{G}$. Il existe donc une fonction qui, à toute règle de production $r \in \mathcal{G}_{CF}$, associe l'ensemble des arbres $t \in \mathcal{G}$ qui génèrent cette règle.

La grammaire $\mathcal{G}_{CF}$ ainsi générée peut être utilisée pour analyser des phrases. Si une phrase $S$ n'a pas d'analyse par $\mathcal{G}_{CF}$, comme $\mathcal{L}_{\mathcal{G}} \subset \mathcal{L}_{\mathcal{G}_{CF}}$, cette phrase n'a pas non plus d'analyse par $\mathcal{G}$. Si $S \in \mathcal{L}_{\mathcal{G}_{CF}}$, on peut extraire de chaque analyse un ensemble d'étiquetages grammaticaux dans $\mathcal{G}$, à partir de la liste des règles de $\mathcal{G}_{CF}$ utilisées pour produire cette analyse. En effet, pour

chaque mot $w_i$ de la phrase $S$, il existe une règle $r \in \mathcal{G}_{CF}$ utilisée dans l'analyse qui produit le terminal $w_i$. On peut alors associer à $w_i$ un ensemble d'étiquettes, qui correspond à l'ensemble des arbres $t \in \mathcal{G}$ qui génèrent la règle $r$.

Afin d'obtenir une méthode d'étiquetage encore plus rapide, en temps linéaire, Boullier calcule à partir de $\mathcal{G}_{CF}$ une grammaire régulière $\mathcal{G}_{FA}$ ($FA$ car elle équivaut à un automate fini), dont le langage engendré $\mathcal{L}_{\mathcal{G}_{FA}}$ est un surensemble de $\mathcal{L}_{\mathcal{G}_{CF}}$.

Chacune de ces deux approximations surensemblistes permet de définir une méthode d'étiquetage grammatical symbolique, sans perte d'étiquetage et sans apprentissage.

### 4.2.3 Représentation des étiquetages par des automates

Adopter le point de vue de l'étiquetage grammatical symbolique sans perte amène à considérer, plus souvent que des étiquetages, des ensembles d'étiquetages. Or, dans les grammaires à large couverture en CCG, TAG ou IG, un mot est généralement associé à une dizaine d'étiquettes en moyenne. Pour une (courte) phrase de dix mots, il y a donc de l'ordre de $10^{10}$ étiquetages grammaticaux possibles. Le nombre d'étiquetages initiaux croît de façon exponentielle avec la longueur de la phrase.

Traiter chacun de ces étiquetages individuellement n'est pas réaliste. La solution classique consiste à représenter les ensembles d'étiquetages grammaticaux par des ensembles de chemins d'un automate acyclique, dans lequel les transitions sont étiquetées par des éléments de la grammaire $\mathcal{G}$. Un tel automate est appelé un **automate d'étiquetages grammaticaux** (AEG). De façon générale, ces automates permettent un gain d'espace élevé. Par exemple, pour une phrase $[w_1, \ldots, w_n]$, le nombre d'étiquetages au début de la phase d'analyse est de $\Pi_{1 \leq i \leq n} |\ell(w_i)|$. Cet ensemble d'étiquetages peut être efficacement représenté par l'ensemble de chemins de l'automate à $n+1$ états $s_0, \ldots, s_n$, dans lequel il y a une transition de $s_{i-1}$ à $s_i$ avec l'étiquette $t$ pour chaque $t \in \ell(w_i)$. Cet automate a $\sum_{1 \leq i \leq n} |\ell(w_i)|$ transitions.

Cette représentation présente plusieurs avantages. Premièrement, elle permet le partage de préfixes et de suffixes et par conséquent, la taille d'un automate d'étiquetages grammaticaux est linéaire par rapport à l'ambiguïté lexicale initiale. L'utilisation d'automates permet également de bénéficier des procédures de programmation dynamique, et par conséquent d'un gain exponentiel en temps et en espace. Enfin, la représentation des ensembles d'étiquetages par des automates permet de réaliser l'application d'une méthode de filtrage par une intersection d'automates, entre l'automate d'étiquetages de la phrase et l'automate de la méthode de filtrage.

## 4.3 Exemple d'étiquetage grammatical en Grammaires d'Interaction

Dans cette section, nous illustrons sur un exemple le principe de l'étiquetage grammatical et l'utilité des méthodes de filtrage des étiquetages. Pour une phrase, nous allons voir quels étiquetages lui sont initialement associés (dans une grammaire jouet) et quels étiquetages doivent

lui être finalement associés à la fin de la phase d'étiquetage grammatical. La différence entre l'ensemble des étiquetages initiaux et l'ensemble des étiquetages finaux correspond à l'effet d'une méthode de filtrage idéale.

Considérons la phrase ambiguë 4.1 :

(4.1) La belle ferme la porte.

### Grammaire

Nous utilisons pour analyser cette phrase la grammaire jouet $\mathcal{G}$ décrite par le tableau 4.1, qui définit le croisement entre :

- une grammaire de structures syntaxiques non ancrées $\mathcal{G}' = \{\texttt{Det}, \texttt{AdjG}, \texttt{AdjD}, \texttt{NC}, \texttt{Clit}, \texttt{VTr}, \texttt{VIntr}\}$, chaque élément noté ici étant le nom d'une description élémentaire de $\mathcal{G}'$ dont le détail est donné en figure 3.4, pages 60 et suivantes,
- et un vocabulaire $\mathcal{V} = \{\text{la, belle, ferme, porte}\}$.

Chaque ligne correspond à une description élémentaire non ancrée de $\mathcal{G}'$. Chaque colonne correspond à un mot de $\mathcal{V}$. Chaque croix correspond à une description élémentaire ancrée de $\mathcal{G}$.

| | la | belle | ferme | porte |
|---|---|---|---|---|
| Det | × | | | |
| AdjG | | × | × | |
| AdjD | | × | × | |
| NC | × | × | × | × |
| Clit | × | | | |
| VTr | | | × | × |
| VIntr | | | × | × |

FIGURE 4.1 – Ancrage de la grammaire jouet

Dans ce tableau,

- `Det` signifie « déterminant »,
- `AdjG` signifie « adjectif épithète gauche »,
- `AdjD` signifie « adjectif épithète droit »,
- `NC` signifie « nom commun »,
- `Clit` signifie « pronom clitique »,
- `VTr` signifie « verbe transitif » et
- `VIntr` signifie « verbe intransitif ».

Par exemple, le mot *« belle »* peut être un adjectif épithète gauche, un adjectif épithète droit ou un nom commun. Trois descriptions élémentaires sont donc associées à ce mot. Elles sont notées respectivement $\texttt{AdjG}_{belle}$, $\texttt{AdjD}_{belle}$ et $\texttt{NC}_{belle}$. L'ancrage ajoute à la description élémentaire non ancrée des informations comme le genre et le nombre, ainsi $\texttt{AdjG}_{belle}$ est un adjectif épithète gauche féminin singulier.

### Étiquetages grammaticaux initiaux

Nous considérons qu'une étiquette grammaticale correspond à chaque description élémentaire de la grammaire. D'après le lexique, la phrase 4.1 peut être initialement étiquetée de $3 \times 3 \times 5 \times 3 \times 3 = 405$ façons différentes. Chaque chemin de l'automate de la figure 4.2 est l'un de ces 405 étiquetages.

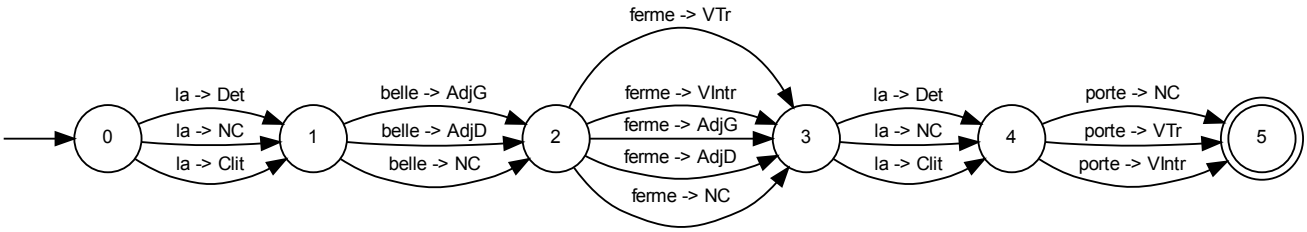



FIGURE 4.2 – Automate des étiquetages initiaux de *« la belle ferme la porte »*

### Étiquetages grammaticaux produisant une analyse

Seulement 3 des 405 étiquetages grammaticaux initiaux de la phrase 4.1 ont une solution. Ces trois étiquetages correspondent chacun à l'une des paraphases approximatives suivantes :

(4.2) *La belle ferme la porte.*
`Det NC VTr Det NC`
'La jolie fille referme la porte.'

(4.3) *La belle ferme la porte.*
`Det NC AdjD Clit VTr`
'La jolie fille (au tempérament) ferme porte la gerbe de fleurs.'

(4.4) *La belle ferme la porte.*
`Det AdjG NC Clit VTr`
'(Le bois de) la jolie ferme porte la patine du temps.'

Ces trois étiquetages grammaticaux sont décrits par l'automate de la figure 4.3.

## 4.4 Étiquetage symbolique par comptage de polarités

Des méthodes de filtrage ont été développées pour les formalismes grammaticaux polarisés en général [BGP04] et pour les IG en particulier [BLRP06]. Ces méthodes exploitent l'idée qu'un étiquetage grammatical ne peut avoir de solution si son comptage de polarités n'est pas équilibré.

En effet, dans les formalismes grammaticaux polarisés, le produit de l'analyse est une structure neutre. Toutes les polarités non neutres d'un étiquetage grammatical doivent donc être

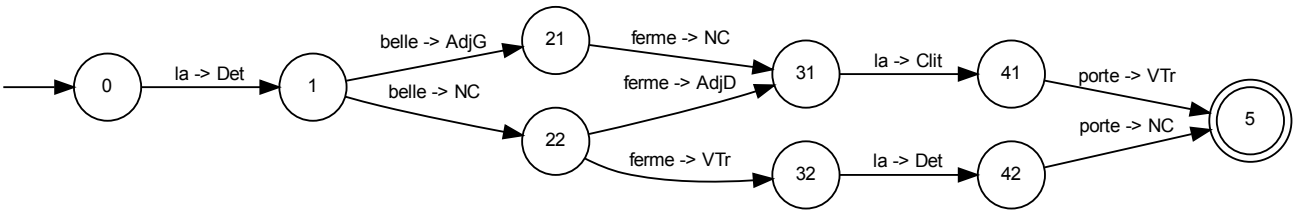



FIGURE 4.3 – Automate des étiquetages analysés de l'exemple 4.1

neutralisées par l'analyse. En particulier, les polarités positives et négatives se neutralisent en formant des couples. Une condition nécessaire pour produire une structure neutre est donc qu'un étiquetage grammatical contienne, pour chaque ensemble d'objets composables, autant de polarités positives que négatives.

### 4.4.1 L'invariant de comptage de van Benthem

Ce principe est en fait une redécouverte et une généralisation de l'*invariant de comptage* de van Benthem [vB86, vB91] pour les démonstrations dans les systèmes logiques sans contraction.

Nous reprenons ici la procédure de comptage et l'invariant tels que formulés dans [vB91].

**Définition 4.4.1** (Comptage des types primitifs)**.** Pour tout type de base $x$, on définit le *comptage de* $x$ $\#_x(a)$ de tout type $a$ récursivement de la façon suivante :

- $\#_x(x) = 1$,
- $\#_x(y) = 0$, pour tous les types de base $y$ différents de $x$,
- $\#_x((a, b)) = \#_x(b) - \#_x(a)$.

Pour des séquences de types comme pour des produits de types, le comptage se fait en additionnant les comptages des éléments de la séquence ou du produit. Ainsi, pour un produit de types : $\#_x(a \cdot b) = \#_x(a) + \#_x(b)$.

L'invariant suivant est valide :

**Proposition 4.4.1** (Invariant de comptage)**.** *Pour tout séquent dérivable $X \Rightarrow a$ dans $LP$, tous les comptages de types primitifs doivent être égaux des deux côtés de la flèche.*

$LP$ est le calcul de Lambek non directionnel : les implications gauche et droite sont confondues en une seule implication.

L'application de l'invariant de comptage aux CG est valable non seulement pour l'étiquetage grammatical d'une phrase (on ne garde que les étiquetages qui vérifient l'invariant) [vdWH88], mais également pour réduire l'espace de recherche exploré par l'analyse : quand l'application

d'une règle requiert de partitionner un séquent, l'invariant de comptage doit être vrai sur chacun des deux séquents créés [Moo88].

### 4.4.2 Filtrage par bilan de polarités

L'invariant de van Benthem a été redécouvert et généralisé à tous les objets polarisés, au travers des travaux sur le *filtrage par bilan de polarités* [BGP04, BLRP06]. Dans les formalismes grammaticaux polarisés, le comptage peut être défini de façon plus directe que pour les types composés.

**Définition 4.4.2** (Comptage des objets polarisés)**.** Pour tout type d'objet polarisé $x$, on définit le *comptage de* $x$, noté $\#_x(a)$, de toute structure $a$ d'une grammaire comme le résultat de la somme suivante :

- chaque objet de $a$ qui est de type $x$ et polarisé positivement $(+)$ compte pour $+1$,
- chaque objet de $a$ qui est de type $x$ et polarisé négativement $(-)$ compte pour $-1$,
- tout autre objet de $a$ compte pour 0.

Les travaux sur le filtrage par bilan de polarités font essentiellement cinq contributions.

Premièrement, le cadre d'application de la méthode de filtrage est agrandi pour inclure différents formalismes grammaticaux. Il suffit en effet, pour pouvoir appliquer le filtrage par bilan de polarités à un formalisme, de définir un *morphisme de polarisation* pour ce formalisme grammatical.

Deuxièmement, la méthode de filtrage est généralisée afin de pouvoir s'appliquer à d'autres objets que de simples multi-ensembles d'objets polarisés. Il est possible de définir différents *morphismes d'abstraction* pour chaque formalisme, qui conservent une plus ou moins grande partie de l'information contenue dans les structures du formalisme. Le filtrage par bilan de polarités « classique » implique alors l'application préalable d'un morphisme de destructuration, noté *destr*, qui oublie l'information structurelle et transforme les structures de la grammaire en multi-ensembles d'objets polarisés.

Troisièmement, le comptage est généralisé à tous les types d'objets polarisés. Par exemple, dans le cadre des IG, ce ne sont pas seulement les catégories qui sont polarisées, mais tous les traits.

Quatrièmement, afin de gérer l'indéterminisme, le comptage utilise une arithmétique d'intervalles. Par exemple, la valeur d'un trait dans une DAP de IG peut être une disjonction de valeurs atomiques, comme dans le trait polarisé `cat` $\rightarrow$ `n|np|s`. Ce trait contribue alors, aux trois comptages de polarités pour les traits `cat : n`, `cat : np` et `cat : s`, un intervalle $[0, +1]$ qui indique que, selon la valeur de trait qui sera sélectionnée par l'analyse (`n`, `np` ou `s`), ce trait contribuera $+1$ dans un comptage et 0 dans les deux autres.

Cinquièmement, l'implantation des comptages est faite sur automates. Ainsi, chaque comptage est fait non pas sur un étiquetage grammatical, mais sur l'ensemble des étiquetages grammaticaux, et le comptage global correspond alors à l'intersection des automates de comptage de

chaque polarité.

**Note** La notion de *morphisme*, et en particulier de *morphisme d'abstraction* présente dans le deuxième point, est importante. Un morphisme d'abstraction permet en effet de transformer une grammaire $\mathcal{G}$ en une autre grammaire $\mathcal{G}'$ plus simple et qui engendre un langage strictement plus grand : $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}'_{\mathcal{G}}$. Nous exploiterons de nouveau cette possibilité dans le prochain chapitre.

## Conclusion

Dans ce chapitre, nous avons abordé le problème posé par la phase d'étiquetage grammatical dans les formalismes lexicalisés. Nous avons développé l'approche symbolique dans laquelle l'étiquetage grammatical d'une phrase est un problème de filtrage de l'ensemble des étiquetages possibles. Nous avons enfin présenté les méthodes d'étiquetage symbolique existantes pour les formalismes polarisés. Ces méthodes comptent les polarités afin d'éliminer tous les étiquetages déséquilibrés, qui n'ont aucune chance de produire une analyse. Dans les deux prochains chapitres, nous présenterons de nouvelles méthodes de filtrage des étiquetages grammaticaux pour les formalismes polarisés. Ces méthodes exploitent la notion de compagnon hypothétique d'une polarité dans une grammaire que nous avons présentée au chapitre 3.

# Chapitre 5

# Filtrage booléen des étiquetages grammaticaux fondé sur les compagnons

Au chapitre 3, nous avons introduit la notion de compagnon hypothétique et nous l'avons utilisée pour définir la quasi-réduction d'une grammaire polarisée. Nous avons vu au chapitre 4 que la première phase de l'analyse syntaxique, l'étiquetage grammatical, correspond à la selection d'une grammaire restreinte. Dans ce chapitre, nous utilisons cette correspondance entre étiquetage grammatical et sélection d'une grammaire, pour dériver de l'idée de quasi-réduction d'une grammaire polarisée un principe de filtrage des étiquetages grammaticaux pour les formalismes lexicalisés polarisés : le *principe du compagnonnage*.

En section 5.1, nous formulons le principe du compagnonnage et nous montrons que ce principe peut être utilisé pour filtrer les étiquetages grammaticaux. Nous montrons en section 5.2 que ce principe permet un filtrage efficace, car il définit un langage régulier qui est une approximation surensembliste des étiquetages grammaticaux qui ont une solution. Nous montrons enfin comment utiliser concrètement le principe du compagnonnage pour filtrer les étiquetages grammaticaux, en utilisant des booléens. Pour cela, nous proposons deux implantations sur automates du filtrage des étiquetages fondé sur le principe du compagnonnage : une méthode exacte dans la section 5.3 et une méthode approximée dans la section 5.4.

## 5.1 Principe du compagnonnage

### 5.1.1 Le principe du compagnonnage

Comme nous l'avons vu au chapitre 4, l'analyse syntaxique peut être découpée en deux phases : l'*étiquetage grammatical* ou *supertagging*, et l'analyse syntaxique proprement dite. L'étiquetage grammatical sélectionne dans la grammaire les descriptions élémentaires utiles à l'analyse. Ensuite, l'analyse syntaxique proprement dite n'a plus qu'à composer les descriptions

élémentaires de l'étiquetage. Chaque étiquetage grammatical d'une phrase correspond donc à la sélection d'une grammaire restreinte, plus précisément d'un multi-ensemble de descriptions élémentaires de la grammaire, qui sera utilisé par l'analyse.

Or, nous avons vu au chapitre 3 qu'il était possible de restreindre une grammaire polarisée, en éliminant les structures qui ne peuvent pas être utilisées pour produire une analyse. Cette restriction, que nous avons appelée quasi-réduction, se fonde sur l'idée que, pour avoir une chance de produire une analyse, une structure ne doit contenir aucune polarité qui n'a aucun compagnon hypothétique dans la grammaire.

La correspondance entre étiquetage grammatical et sélection d'une grammaire restreinte suggère de transposer l'idée de la quasi-réduction d'une grammaire polarisée au problème du filtrage des étiquetages grammaticaux. Nous dérivons donc de la notion de grammaire polarisée quasi-réduite un principe sur les étiquetages, que nous appelons le *principe du compagnonnage*. Intuitivement, ce principe exprime le fait que, dans tout étiquetage qui a une solution, toutes les polarités des structures de l'étiquetage ont au moins un compagnon hypothétique gauche à leur gauche, ou un compagnon hypothétique droit à leur droite, dans l'étiquetage.

**Proposition 5.1.1** (Principe du compagnonnage)**.** *Si un étiquetage grammatical* $[t_1, \ldots, t_n]$ *a une solution alors pour toute polarité atomique* $p_k \in t_i$ *:*

- $\bigcup_{1 \leq j \leq i-1} \mathbf{pols}(t_j) \cap \mathcal{L}_{pk} \neq \emptyset$,
- *ou* $\bigcup_{i+1 \leq j \leq n} \mathbf{pols}(t_j) \cap \mathcal{R}_{pk} \neq \emptyset$.

Dans le cas des grammaires lexicalisées, on peut par conséquent lister naïvement tous les étiquetages grammaticaux possibles d'une phrase, puis les filtrer pour ne garder que ceux qui contiennent au moins un compagnon hypothétique pour chacune des polarités atomiques des descriptions élémentaires qu'ils contiennent. Le principe du compagnonnage devient alors une méthode de filtrage.

### 5.1.2 Exemple

À titre d'exemple, utilisons le principe du compagnonnage pour filtrer les étiquetages grammaticaux de la phrase 4.1 à l'aide des informations du tableau 3.5. Ce tableau contient les compagnons hypothétiques, dans la grammaire jouet non ancrée, des polarités contenues dans la description associée aux déterminants `Det`.

**Ligne 1** La première ligne du tableau indique que la polarité $\rightarrow$ attachée au trait `cat : det` du nœud (0,3) de la description `Det` n'a aucun $1^{er}$ compagnon hypothétique à gauche et un seul $1^{er}$ compagnon hypothétique à droite dans la grammaire : la polarité $\leftarrow$ attachée au trait `cat : det` du nœud (0,3) de la description `NC`.

Par conséquent, le principe du compagnonnage pour les étiquetages grammaticaux implique que tout étiquetage qui a une solution et qui contient une description `Det` doit également contenir une description `NC` à droite de celle-ci.

**Lignes 2 à 4** Les deuxième, troisième et quatrième lignes du tableau amènent à la même conclusion que la première ligne, en examinant respectivement la polarité ← attachée au trait `funct : det` du nœud (0,3) et la polarité ∼ attachée au trait `cat : n` du nœud (0,0).

Par conséquent, tout étiquetage qui a une solution et qui contient une description `Det` doit également contenir une description `NC` à droite de celle-ci.

**Lignes 5 et 6** Les cinquième et sixième lignes contiennent les compagnons hypothétiques de la polarité ∼ attachée au trait `cat : np` du nœud (0,2) de `Det`. Cette polarité composite cherche deux compagnons. Ses $1^{ers}$ compagnons hypothétiques sont, à gauche, la polarité ← du trait `cat : np` du nœud (0,0) de `VTr`, et à droite, les polarités ← des traits `cat : np` des nœuds (0,3) de `VTr` et (0,2) de `VIntr`. Son $2^{e}$ compagnon hypothétique est, à gauche ou à droite, la polarité → du trait `cat : np` du nœud (0,1) de `NC`.

Par conséquent, tout étiquetage qui a une solution et qui contient une description `Det` doit également contenir :

1. une description `VIntr` à droite de `Det` ou une description `VTr` à gauche ou à droite de `Det`,
2. et une description `NC` à gauche ou à droite de `Det`.

**Bilan** La conjonction de ces contraintes sur les étiquetages nous donne une formulation concise des contraintes sur les étiquetages qui contiennent la description `Det`. Il est important de noter que ces contraintes ne sont valables que dans le cadre de notre grammaire jouet et ne prétendent pas refléter des principes linguistiques universels.

Tout étiquetage qui a une solution et qui contient une description `Det` doit également contenir :

- une description `NC` à droite de `Det`,
- et, soit une description `VIntr` à droite de `Det`, soit une description `VTr`(à gauche ou à droite de `Det`).

Prenons un étiquetage en particulier : [$\texttt{Det}_{la}$, $\texttt{NC}_{belle}$, $\texttt{VTr}_{ferme}$, $\texttt{Det}_{la}$, $\texttt{VTr}_{porte}$] n'a pas de solution, car le $\texttt{Det}_{la}$ situé en avant-dernière position n'a pas de `NC` à sa droite.

Au total, sur les 405 étiquetages de départ, seuls 248 vérifient les deux contraintes concises posées par les déterminants. Ces étiquetages correspondent aux chemins de l'automate 5.1.

## 5.2 Le langage du principe du compagnonnage pour les étiquetages

Un étiquetage grammatical est, plus précisément, un élément du langage formel $\mathcal{G}^*$. Nous pouvons alors considérer les trois langages suivants :

- Le premier est $\mathcal{G}^*$ lui-même.

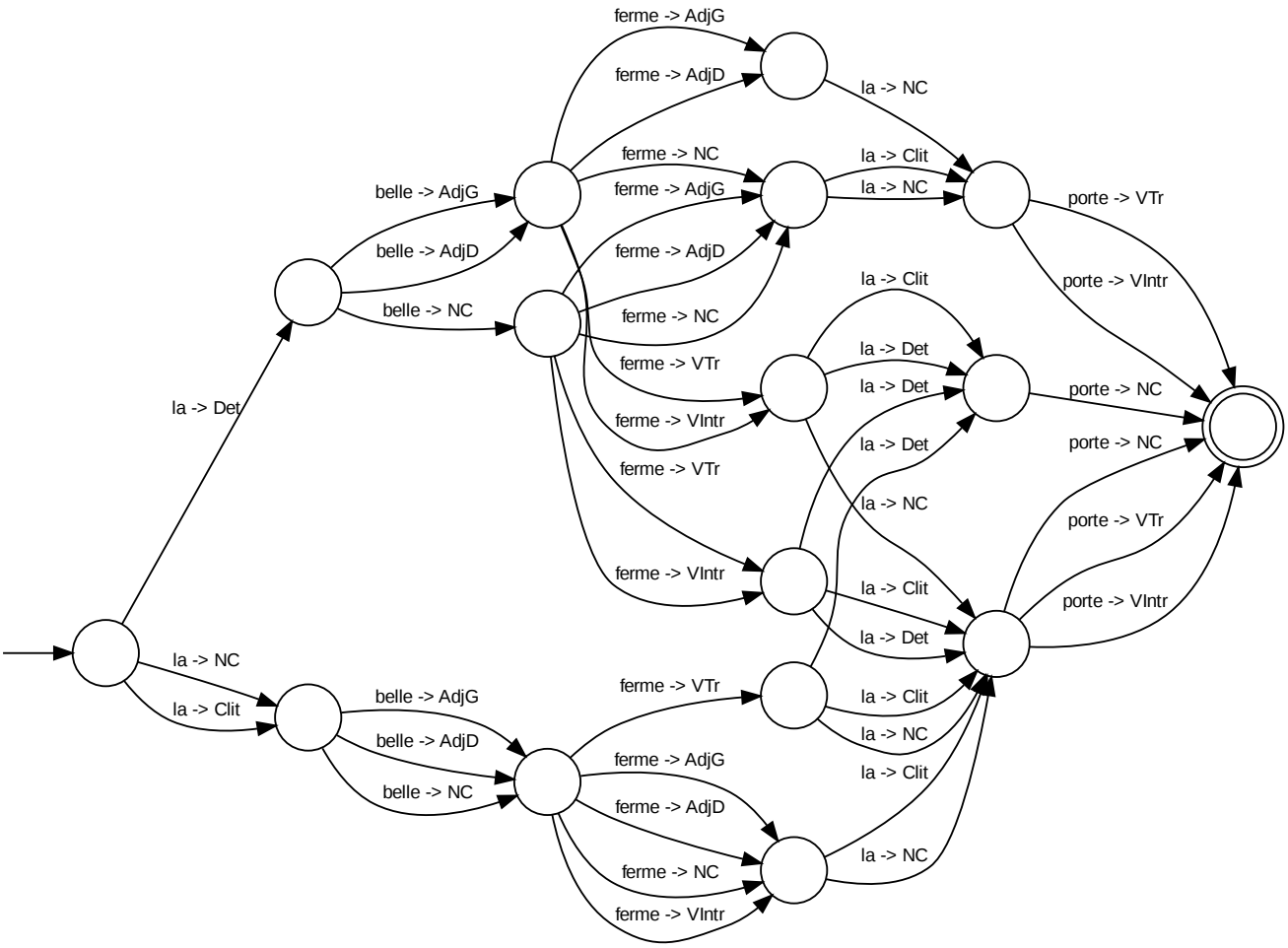



FIGURE 5.1 – Étiquetages grammaticaux qui respectent les contraintes des déterminants

– Le deuxième est l'ensemble $C \subseteq \mathcal{G}^*$. Il correspond aux étiquetages grammaticaux qui sont analysables. Le but de la phase d'étiquetage grammatical est alors d'énumérer pour toute phrase $[w_1, \ldots, w_n]$ tous les étiquetages grammaticaux qui sont dans $C$.
– Le troisième est le langage $\mathcal{P}$ des étiquetages grammaticaux qui vérifient le principe du compagnonnage.

$\mathcal{P}$ se situe entre les deux langages précédents : $C \subseteq \mathcal{P} \subseteq \mathcal{G}^*$. Nous retrouvons la démarche d'approximation surensembliste que nous avons présentée au chapitre précédent.

Considérons une description élémentaire $t$ et une polarité atomique $p_k$ appartenant à cette description. Alors, l'ensemble des étiquetages grammaticaux qui vérifient la contrainte de compagnonnage posée par cette polarité atomique peut être décrit comme :

$$L_{t:pk} = \complement((\complement\mathcal{L}_{pk})^* t (\complement\mathcal{R}_{pk})^*)$$

où $\complement$ dénote le complémentaire d'un ensemble.

Littéralement, l'ensemble des étiquetages qui respectent le principe du compagnonnage pour la polarité $p_k$ de la structure $t$ est le complémentaire des étiquetages qui contiennent $t$ mais aucun $t'$ contenant un compagnon hypothétique gauche à sa gauche ou un compagnon hypothétique droit à sa droite.

$\mathcal{P}$ étant défini comme l'ensemble des étiquetages grammaticaux vérifiant les contraintes de toutes les polarités atomiques de $\mathcal{G}$, $\mathcal{P}$ est un langage régulier défini par :

$$P = \bigcap_{t \in \mathcal{G},\, p_k \in t} L_{t:pk}$$

Comme $C$ n'est a priori pas un langage régulier, du moins pour les langues naturelles, $\mathcal{P}$ est donc une meilleure approximation régulière de $C$ que le langage trivial $\mathcal{G}^*$.

Du principe du compagnonnage pour les étiquetages, nous dérivons un principe de filtrage des étiquetages grammaticaux qui teste simplement les étiquetages candidats par rapport à $\mathcal{P}$. En théorie, $\mathcal{P}$ peut être calculé statiquement à partir de la grammaire.

Dans notre exemple, l'automate décrivant $\mathcal{P}$ pour la grammaire jouet $\mathcal{G}$ est donné en figure 5.2, où :

– `c` correspond à `Clit`,
– `d` correspond à `Det`,
– `i` correspond à `VIntr`,
– `l` correspond à `AdjG`,
– `n` correspond à `NC`,
– `t` correspond à `VTr`,
– `r` correspond à `AdjD`.

Une approximation grossière de la taille de l'automate correspondant à $\mathcal{P}$ peut facilement être calculée. Comme nous le verrons dans la section 5.3, chaque automate $L_{t:pk}$ a 4 états, donc $\mathcal{P}$ a au plus $4^m$ états, où $m$ est le nombre de polarités atomiques de la grammaire. Calculer

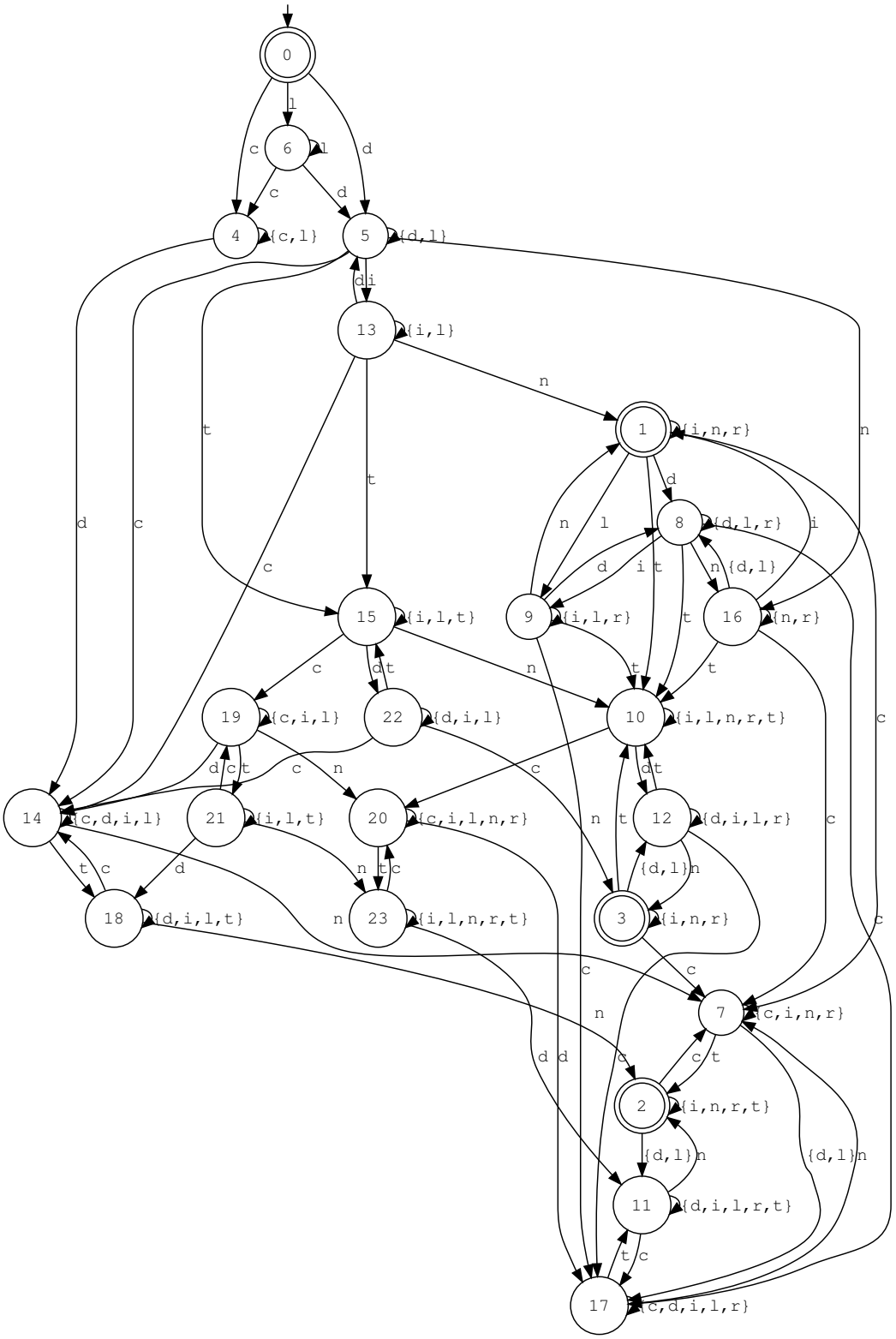



FIGURE 5.2 – Le langage $\mathcal{P}$ pour la grammaire jouet $\mathcal{G}$

l'automate qui reconnaît $\mathcal{P}$ pour toute une grammaire lexicalisée, même non ancrée, n'est pas réaliste d'un point de vue calculatoire. Par exemple, la grammaire FRIGRAM contient 3 788 descriptions non ancrées. Chaque description contenant en moyenne une dizaine de polarités, $m > 10.|\mathcal{U}| > 10^4$, $\mathcal{U}$ étant la grammaire non ancrée. Comme Boullier [Bou10], nous calculons en fait à la volée l'automate qui reconnaît la restriction de $\mathcal{P}$ nécessaire à chaque ensemble d'étiquetages d'une phrase.

## 5.3 Filtrage booléen fondé sur le Principe du Compagnonnage (*BCP*)

### 5.3.1 Intuition

Pour ne garder que les étiquetages grammaticaux d'une phrase qui vérifient le principe du compagnonnage, il suffit de vérifier tous les étiquetages un par un et d'éliminer ceux qui contiennent une polarité mais aucun de ses compagnons hypothétiques.

Nous appelons cette méthode *Filtrage Booléen fondé sur le Principe du Compagnonnage*, notée *BCP* pour *Boolean Companionship Principle*.

### 5.3.2 Implantation sur automate

Comme nous représentons les ensembles d'étiquetages par des automates, nos méthodes de filtrage correspondent à des opérations sur les automates. Considérons un AEG $\mathcal{A}$ pour une phrase $[w_1, \ldots, w_n]$. Pour chaque étiquette de transition $t$ [11] et pour toute polarité atomique $p_k \in t$, nous construisons un automate $\mathcal{A}_{t,\mathcal{L}_{pk},\mathcal{R}_{pk}}$ comme suit.

Tout état $s$ de $\mathcal{A}_{t,\mathcal{L}_{pk},\mathcal{R}_{pk}}$ est étiqueté par un triplet composé d'un état de l'automate $\mathcal{A}$ et de deux booléens :

- le premier booléen vaut vrai si, dans tous les chemins qui passent par $s$, on a emprunté avant $s$ une transition $t$ dont la polarité $p_k$ n'a pas encore trouvé de compagnon hypothétique ; autrement dit, les chemins qui entrent dans cet état contiennent une polarité $p_k$ qui attend impérativement un compagnon hypothétique à sa droite ;
- le second booléen vaut vrai si, dans tous les chemins qui passent par $s$, on a emprunté avant $s$ une transition $\mathcal{L}_{pk}$. Dans ce cas, toute transition $t$ empruntée après $s$ est assurée d'avoir un compagnon hypothétique pour ses $p_k$.

**Notation** Lorsque nous considérons un étiquetage pour une polarité atomique $p_k$ d'une description $t$, nous utilisons les notations suivantes :

- x correspond à une instance de la polarité $p_k$,
- l correspond à une instance d'un compagnon hypothétique gauche $q_k$ de $p_k$ : $q_k \in \mathcal{L}_{pk}$,

11. Un étiquetage peut contenir plusieurs fois la même étiquette de transition, c'est-à-dire la même description élémentaire de la grammaire. Par exemple, il est très probable qu'un étiquetage de la phrase *« Le lapin aime le foin »* associe la même description de la grammaire aux deux déterminants *« le »*.

– R correspond à une instance d'un compagnon hypothétique droit $q_k$ de $p_k$ : $q_k \in \mathcal{R}_{pk}$.

Afin de simplifier la présentation, nous utilisons dans ce chapitre cette même notation de façon moins fine dans les étiquetages :

– X désigne une description $t$, qui contient la polarité $p_k$,
– L désigne une description contenant un compagnon hypothétique gauche $q_k$ de $p_k$,
– R désigne une description contenant un compagnon hypothétique droit $q_k$ de $p_k$.

**Construction d'un automate** L'état initial est étiqueté $(s_0, \mathtt{F}, \mathtt{F})$, où $s_0$ est l'état initial de $\mathcal{A}$, et les autres états sont construits récursivement, pas à pas, à partir de l'état initial de la façon suivante. Si $s \xrightarrow{u} s'$ dans $\mathcal{A}$, alors on ajoute dans $\mathcal{A}_{t,\mathcal{L}_{pk},\mathcal{R}_{pk}}$ les transitions suivantes avec leurs états destination :

1. $(s, b_1, \mathtt{F}) \xrightarrow{u} (s', \mathtt{T}, \mathtt{F})$ si $u = \text{X}$,
2. $(s, b_1, \mathtt{T}) \xrightarrow{u} (s', b_1, \mathtt{T})$ si $u = \text{X}$,
3. $(s, b_1, b_2) \xrightarrow{u} (s', \mathtt{F}, b_2)$ si $u = \text{R}$,
4. $(s, b_1, b_2) \xrightarrow{u} (s', b_1, \mathtt{T})$ si $u = \text{L}$,
5. $(s, b_1, b_2) \xrightarrow{u} (s', b_1, b_2)$ si $u \neq \{\text{X}, \text{L}, \text{R}\}$,

où $b_1, b_2 \in \{\mathtt{F}, \mathtt{T}\}$.

L'automate de la figure 5.3 décrit les changements de valeur des deux booléens.

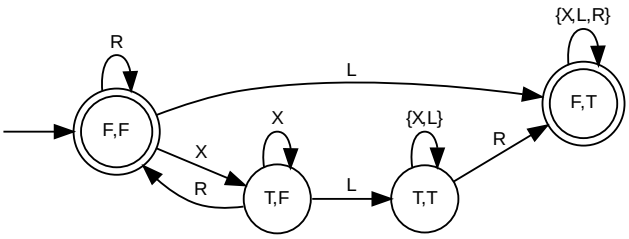



FIGURE 5.3 – Automate des booléens pour le filtrage *BCP*

**Propriétés d'un automate** Il est très simple de montrer que, pour tout état étiqueté $(s, b_1, b_2)$ :

– $b_1$ vaut T si et seulement si tous les chemins allant de l'état initial jusqu'à $s$ contiennent une transition X sans L à sa gauche ni R à sa droite,
– $b_2$ vaut T si et seulement si tous les chemins allant de l'état initial jusqu'à $s$ contiennent une transition L.

Par conséquent, tout chemin dont le dernier état est étiqueté $(s_f, \mathtt{T}, \mathtt{F})$, avec $s_f$ un état final de $\mathcal{A}$, est de la forme $\text{R}^*(\text{X}^*\text{R})^*\text{X}^+$ ; tout chemin dont le dernier état est étiqueté $(s_f, \mathtt{T}, \mathtt{T})$ est de la forme $\text{R}^*(\text{X}^*\text{R})^*\text{X}^+\text{L}(\text{L}|\text{X})^*$. Ces chemins contiennent au moins un X qui n'a pas trouvé de

compagnon hypothétique bien placé dans l'étiquetage, ils sont caractérisés par des derniers états du type $(s_f, \texttt{T}, b_2)$.

Par contraste, les états finaux sont du type $(s_f, \texttt{F}, b_2)$. Les chemins qui y aboutissent, soit ne contiennent aucun x, soit contiennent un ou des x qui ont tous un compagnon hypothétique.

Comme il peut y avoir pour chaque état de l'automate initial au plus quatre états dans l'automate construit, la taille de ces automates construits est inférieure ou égale à $4n$, où $n$ est la taille de $\mathcal{A}$.

**Exemple** La figure 5.4 ci-dessous représente l'automate $\mathcal{A}_{\texttt{Det},\mathcal{L}_1,\mathcal{R}_1}$ : la polarité étudiée est située à la $1^{re}$ ligne du tableau 3.5.

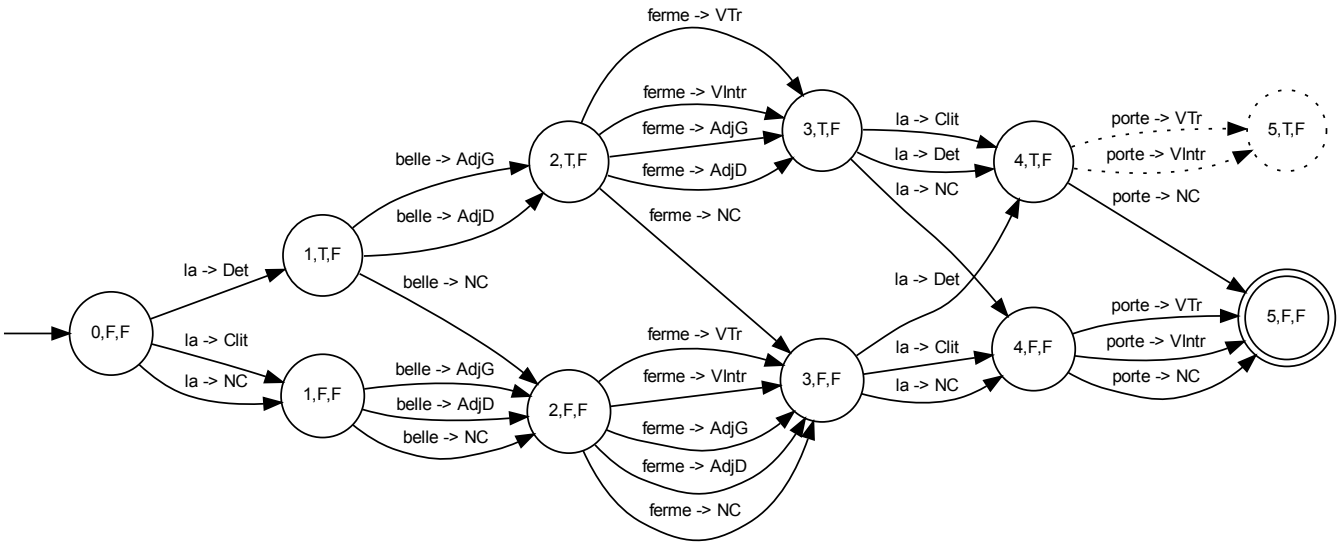



FIGURE 5.4 – Étiquetages grammaticaux qui respectent le principe du compagnonnage pour la polarité 1 de `Det`

Les transitions et l'état marqués en pointillés n'amènent pas à un état final. Ces éléments peuvent par conséquent être enlevés de l'automate, ce qui supprime 106 chemins. La partie de l'automate marquée en trait plein contient 299 chemins. Au final, l'application de la méthode *BCP* à cet exemple, pour la seule polarité 1 de `Det`, permet d'enlever 106 des 405 étiquetages grammaticaux initiaux.

**Construction de l'automate global** Pour chaque transition $t$ de l'AEG et pour chaque polarité $p_k \in t$, nous construisons l'automate $\mathcal{A}_{t,\mathcal{L}_{pk},\mathcal{R}_{pk}}$. L'intersection de ces automates représente l'ensemble des étiquetages possibles de la phrase qui respectent le principe du compagnonnage.

Nous calculons donc :

$$\mathcal{A}_{BCP} = \bigcap_{t \in \mathcal{A},\, p_k \in t} \mathcal{A}_{t,\mathcal{L}_{pk},\mathcal{R}_{pk}}$$

Il peut être démontré que cet automate est le même que l'automate obtenu par intersection entre l'automate des étiquetages initiaux $\mathcal{A}$ et l'automate du langage $\mathcal{P}$ défini précédemment, en section 5.2 :

$$\mathcal{A}_{BCP} = \mathcal{A} \cap \mathcal{P}$$

**Exemple** Pour notre exemple, l'intersection des 38 automates correspondant aux 38 polarités atomiques de la grammaire non ancrée $\mathcal{G}'$ produit l'automate de la figure 5.5. Cet automate a

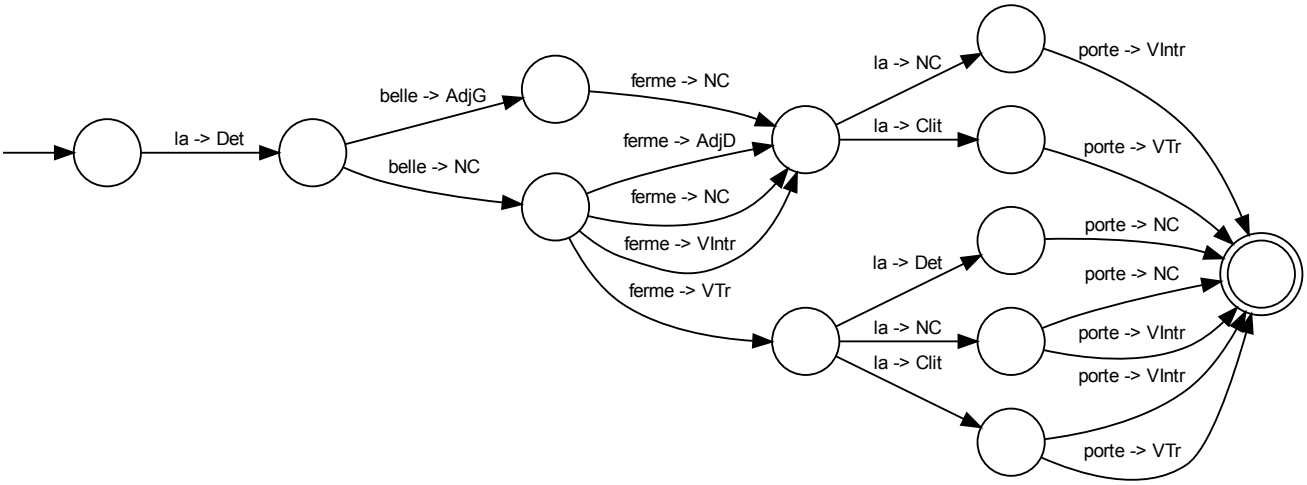



FIGURE 5.5 – Automate produit par le filtrage *BCP*

13 chemins, ce qui signifie que 13 étiquetages grammaticaux, sur les 405 initiaux, vérifient le principe du compagnonnage.

## 5.4 Approximation du filtrage booléen fondé sur les compagnons (*QCP*)

### 5.4.1 Motivation

La méthode *BCP* est précise mais présente le désavantage d'utiliser un grand nombre d'automates : $O(n)$, où $n$ est la taille de la phrase. Chacun de ces automates est lui-même de taille $O(n)$, ce qui amène la complexité théorique de l'intersection à $O(n^n)$. C'est la raison pour laquelle nous proposons un algorithme approximé de *BCP*, que nous appelons *Filtrage Rapide fondé sur le Principe du Compagnonnage*, noté *QCP* pour *Quick Companionship Principle*.

### 5.4.2 Intuition

Alors que le *BCP* vérifie les étiquetages un par un, le *QCP* examine les descriptions élémentaires. Plus précisément, en termes d'automates, alors que le *BCP* vérifie les chemins un par un, le *QCP* vérifie les transitions de l'automate. Ainsi, le *QCP* considère d'un coup tous les étiquetages qui partagent une même transition dans l'automate. Pour chaque transition de l'automate, l'algorithme *QCP* vérifie qu'il existe au moins un chemin qui emprunte cette transition et qui vérifie le principe du compagnonnage.

Cet algorithme ne fait qu'enlever des transitions dans l'automate qu'il prend en entrée. Il présente donc l'avantage de ne pas changer le nombre d'états de l'automate. Cette propriété s'avère particulièrement intéressante sur des automates plats, comme ceux qui représentent l'ensemble des étiquetages initiaux naïfs d'une phrase. C'est la raison pour laquelle le *QCP* occupe une place particulière dans nos expérimentations, comme nous le verrons au chapitre 7.

### 5.4.3 Implantation sur automate

Considérons un AEG $\mathcal{A}$. Nous notons $\prec_{\mathcal{A}}$ la relation de précédence sur les transitions dans un automate $\mathcal{A}$. Nous définissons $l_{\mathcal{A}}(t) = \{u \in \mathcal{G}, u \prec_{\mathcal{A}} t\}$ et $r_{\mathcal{A}}(t) = \{u \in \mathcal{G}, t \prec_{\mathcal{A}} u\}$. Pour chaque transition $s \xrightarrow{t} s'$ et pour chaque polarité $p_k \in t$, si $l_{\mathcal{A}}(t) \cap \mathcal{L}_{pk} = \emptyset$ et $r_{\mathcal{A}}(t) \cap \mathcal{R}_{pk} = \emptyset$ alors aucun des étiquetages grammaticaux qui utilisent la transition $t$ n'a de solution et la transition $t$ peut être enlevée de l'automate.

Cet algorithme peut être calculé par une double boucle `pour` : pour chaque polarité atomique de chaque transition, on vérifie que soit le contexte gauche, soit le contexte droit de la transition contient une description syntaxique contenant un compagnon hypothétique de cette polarité. Remarquons que le coût de cet algorithme est $O(m^2)$, où $m$ est la taille (i.e. le nombre de transitions) de l'automate fourni en entrée.

Cet algorithme doit être itéré jusqu'à atteindre un point fixe. En effet, cet algorithme peut enlever une transition qui fournit un compagnon hypothétique à une polarité d'une autre transition. Néanmoins, comme nous enlevons au moins une transition à chaque itération jusqu'au point fixe, nous itérons la double boucle `pour` au plus $O(m)$ fois. La complexité totale de cet algorithme est donc $O(m^3)$. En pratique, sur la grammaire FRIGRAM, nous avons observé que la complexité était plus proche de $O(m^2)$ : seules 2 ou 3 itérations suffisent à atteindre le point fixe.

### 5.4.4 Exemple

Appliquons la méthode *QCP* à l'automate initial de la figure 4.2, que nous rappelons en figure 5.6. La première passe enlève la transition $0 \xrightarrow{la->\texttt{NC}} 1$, qui n'a aucun compagnon hypothétique pour sa polarité `cat` $\leftarrow$ `det`. En effet, dans la grammaire $\mathcal{G}$, la seule description adéquate serait un `Det` situé à gauche de ce `NC`. La deuxième passe enlève la transition $1 \xrightarrow{belle->\texttt{AdjD}} 2$, qui n'a aucun compagnon hypothétique pour sa polarité `cat` $\sim$ `n`. En effet, la seule description

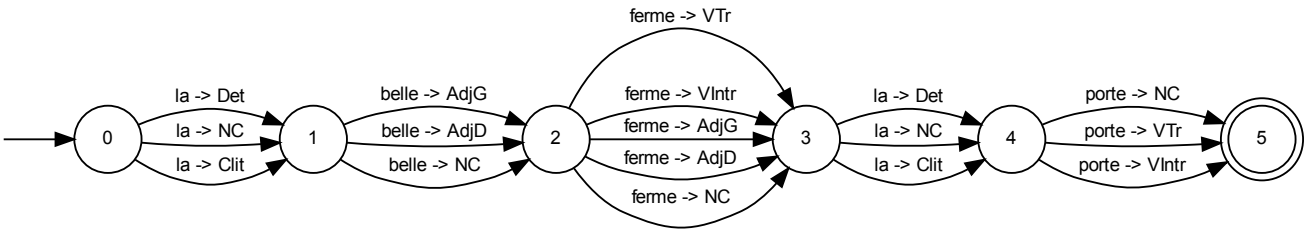



FIGURE 5.6 – Automate des étiquetages initiaux de *« la belle ferme la porte »*

adéquate serait un `NC` situé à sa gauche, qui a été enlevé par la passe précédente. Le point fixe est alors atteint et l'automate produit par la méthode *QCP* est celui de la figure 5.7 qui contient 180 chemins, sur les 405 initiaux.

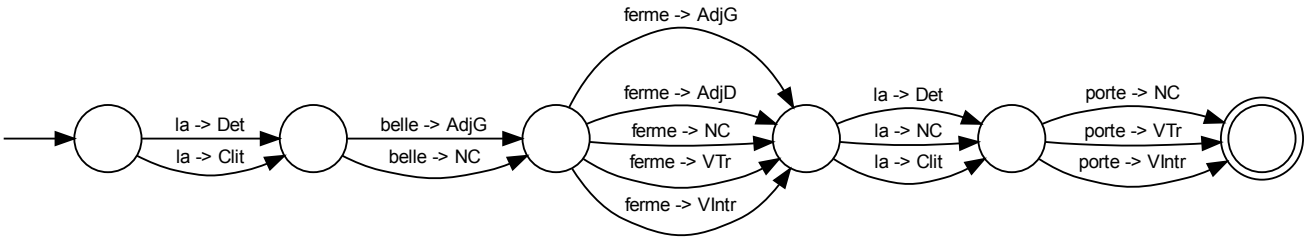



FIGURE 5.7 – Automate produit par le filtrage QCP

## Conclusion

Nous avons présenté dans ce chapitre un principe de filtrage des étiquetages grammaticaux pour les formalismes polarisés, qui utilise la notion de compagnon hypothétique présentée au chapitre 3. Nous avons dérivé de ce principe deux méthodes de filtrage, une exacte et une approximée, que nous avons implantées sur automate. Ces méthodes ont l'avantage d'intégrer des informations qui n'étaient pas prises en compte par les méthodes de filtrage existantes. Ces nouvelles méthodes sont donc complémentaires des méthodes existantes. Leur efficacité sera évaluée sur corpus au chapitre 7.

Le critère de filtrage des méthodes présentées dans ce chapitre est booléen : un étiquetage est conservé si, pour chacune de ses polarités, il contient un compagnon hypothétique bien

placé. Ce critère de filtrage ne fait pas la différence entre les polarités qui monopolisent leur compagnon hypothétique et celles qui laissent leur compagnon hypothétique libre de se composer avec d'autres polarités. Pour cela, il faut compter les polarités qui sont en concurrence et les compagnons hypothétiques contenus dans un étiquetage. C'est l'objet du prochain chapitre.

# Chapitre 6

# Filtrage entier des étiquetages grammaticaux fondé sur les compagnons

Dans ce chapitre, nous proposons une troisième méthode de filtrage des étiquetages grammaticaux fondée sur les compagnons hypothétiques. À la différence des deux premières, cette méthode est spécifique aux polarités linéaires.

La section 6.1 introduit la problématique générale du filtrage entier fondé sur les compagnons et donne une intuition de cette méthode de filtrage. La section 6.2 donne les définitions et notations qui sont nécessaires pour la suite. Dans la section 6.3, nous définissons le langage des compagnons qui permet de formaliser notre méthode de filtrage. Ce langage est prototypique au sens où il comporte le nombre minimum de lettres nécessaire pour être considéré. Nous montrons que ce langage n'est pas un langage algébrique et, dans la section 6.4, qu'il est équivalent à un langage défini par un pomonoïde. Dans la section 6.5, nous donnons enfin l'algorithme par flot (*streaming algorithm*) qui décide de l'appartenance d'un mot au langage des compagnons et nous prouvons la correction de cet algorithme. Nous montrons enfin, en section 6.6, comment utiliser l'appartenance au langage des compagnons pour filtrer les étiquetages grammaticaux.

## 6.1 Principe du filtrage entier fondé sur les compagnons

### 6.1.1 Polarités linéaires et concurrentes

Nous avons vu au chapitre 1 que les polarités non saturées peuvent être séparées en deux catégories, selon leur capacité de combinaison : les polarités $(■, □)$ et $(□, ■)$ sont qualifiées de *linéaires* et les polarités $(□, □)$ de *non linéaires*. Plusieurs polarités $(□, □)$ peuvent être composées et se partager le ou les mêmes compagnons. A contrario, il est impossible de composer deux polarités $(■, □)$ avec la même polarité $(□, ■)$ ; il est également impossible de composer deux polarités $(□, ■)$ avec la même polarité $(■, □)$. Chaque polarité $(■, □)$ *consomme* une polarité $(□, ■)$.

Les polarités linéaires ne peuvent donc pas se partager un même compagnon.

Quand une description est utilisée plusieurs fois dans un même étiquetage, cet étiquetage contient deux copies de la même polarité. Si ce sont des copies d'une polarité linéaire, elles cherchent leur compagnon parmi les mêmes compagnons hypothétiques : elles sont concurrentes.

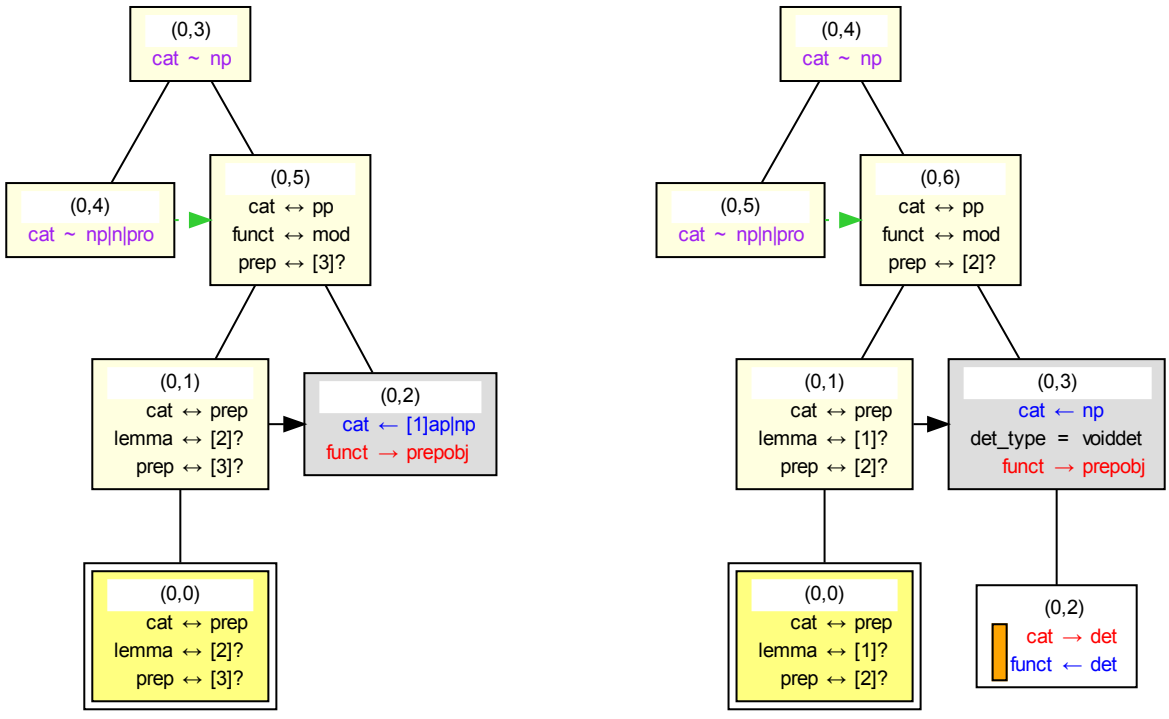
Plus généralement, nous appelons **polarités concurrentes** toutes les instances de polarités linéaires qui ont le même ensemble de compagnons hypothétiques.

### Exemple

Considérons l'exemple suivant :

(6.1) Les perspectives de redémarrage de l'activité évoluent.

**Grammaire** Pour analyser cette phrase, nous utilisons la grammaire d'interaction $\mathcal{G}'$ de la figure 3.4, pages 60 à 63, à laquelle nous ajoutons deux descriptions pour les prépositions qui introduisent des compléments du nom. Ces descriptions sont représentées dans la figure 6.1.




(a) `PrepNP` (b) `PrepN`

FIGURE 6.1 – Descriptions pour les prépositions

La description `PrepNP`, figure 6.1a, représente une préposition qui introduit un syntagme

nominal à sa droite : le nœud (0,2) contient les traits `cat` ← `ap|np` et `funct` → `prepobj` ; ce nœud n'a pas de fils dans cette description.

La description `PrepN`, figure 6.1b, représente une préposition qui introduit un syntagme nominal à sa droite et qui fournit un déterminant déficient à ce syntagme nominal : le nœud (0,3) contient les traits `cat` ← `np`, `funct` → `prepobj` et `det_type = voiddet` ; ce nœud a pour fils le nœud (0,2) qui contient les traits `cat` → `det` et `funct` ← `det`.

La principale différence entre ces deux descriptions est donc que `PrepN` apporte un déterminant alors que `PrepNP` n'en apporte pas. La description `PrepN` permet de former les compléments du nom en [de NP] [12].

**Étiquetages grammaticaux** Après application des méthodes de filtrage par bilan de polarités et des méthodes fondées sur les compagnons (*QCP* et *BCP*), il reste pour notre exemple les deux étiquetages grammaticaux représentés par l'automate de la figure 6.2.

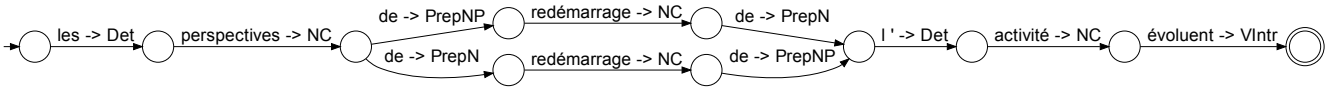



FIGURE 6.2 – Étiquetages restants avant le filtrage entier

Ces deux étiquetages sont quasiment identiques. Ils diffèrent uniquement sur l'attribution des descriptions `PrepN` et `PrepNP` aux deux *« de »*. Dans *« Les perspectives **de** redémarrage **de** l'activité évoluent »*, est-ce que c'est le premier *« de »* qui est une préposition simple (`PrepNP`), et le deuxième *« de »* qui est une préposition avec déterminant déficient (`PrepN`), ou l'inverse ?

Intuitivement, nous connaissons la réponse à cette question : c'est le premier *« de »* qui est une préposition avec déterminant déficient (`PrepN`), alors que le deuxième *« de »* est une préposition simple (`PrepNP`). Le raisonnement est le suivant :

(i) l'exemple contient trois noms communs, étiquetés `NC` : *« perspectives »*, *« redémarrage »* et *« activité »* ;

(ii) chacun de ces trois noms communs doit trouver un déterminant à sa gauche ;

(iii) or, la seule possibilité pour *« perspectives »* de trouver un déterminant à sa gauche est de se lier à *« les »* ;

(iv) donc, la seule possibilité pour *« redémarrage »* de trouver un déterminant à sa gauche est que ce soit le premier *« de »* qui le lui fournisse ;

(v) donc le premier *« de »* est une préposition avec déterminant déficient (`PrepN`).

L'étiquetage correspondant est représenté par la figure 6.3. Dans ce raisonnement, nous avons exploité le fait que la description du nom commun `NC`, rappelée en figure 6.4, est utilisée trois fois dans chaque étiquetage de la phrase. Chaque étiquetage contient donc trois exemplaires

12. Voir [Kni09] pour une analyse détaillée de ces compléments du nom.

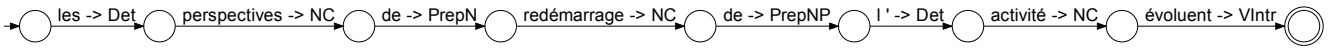



FIGURE 6.3 – Étiquetages corrects

de chaque polarité de la description `NC`. En particulier, cela signifie que ces chaque étiquetage contient trois exemplaires du trait polarisé linéaire `cat` ← `det`. Ces trois exemplaires sont donc en concurrence pour les traits `cat` → `det`, et un seul des deux étiquetages possibles permet à chacune de ces polarités concurrentes de trouver un compagnon hypothétique différent.

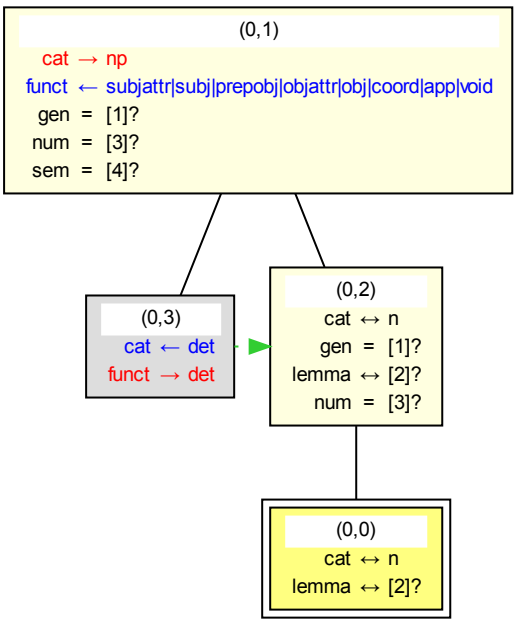



FIGURE 6.4

### 6.1.2 Filtrage pour les polarités concurrentes

#### Motivation

La méthode *BCP* garantit que, dans un étiquetage, toute polarité prise isolément a au moins un compagnon potentiel. Cette méthode ne garantit cependant pas que, si un étiquetage contient des polarités concurrentes, chacune d'entre elles puisse trouver dans l'étiquetage un compagnon distinct. Dans ce chapitre, nous proposons une méthode de filtrage des étiquetages

grammaticaux, qui est spécifique aux polarités concurrentes et qui apporte une telle garantie. Nous appelons cette méthode *Filtrage Entier fondé sur le Principe du Compagnonnage* (*ICP* pour *Integer Companionship Principle*).

### Intuition

Pour formaliser cette méthode, nous utilisons le langage à trois symboles L, X et R que nous avons introduit à la section 5.3 et que nous rappelons ici.

**Notation** Lorsque nous considérons un étiquetage pour une classe d'équivalence de polarités, nous utilisons les notations suivantes :

- X correspond à une instance de polarité $p$ de la classe d'équivalence,
- L correspond à une instance d'un compagnon hypothétique gauche $q$ de $p$ : $q \in \mathcal{L}_p$,
- R correspond à une instance d'un compagnon hypothétique droit $q$ de $p$ : $q \in \mathcal{R}_p$.

Considérons un étiquetage $E$. L'ensemble des polarités linéaires contenues dans $E$ peut être partitionné en classes d'équivalence, chaque classe d'équivalence correspondant à un ensemble de polarités concurrentes. Pour chaque classe de polarités concurrentes, on construit un mot sur $\{\text{L}, \text{X}, \text{R}\}$ qui représente le contenu de $E$ pour cette classe de polarités.

**Propriété 6.1.1** (Satisfiabilité des polarités concurrentes)**.** *Chaque polarité concurrente* X *doit trouver exactement un compagnon hypothétique* L *à sa gauche ou un compagnon hypothétique* R *à sa droite, et tout compagnon hypothétique (*L *ou* R*) peut être lié à au plus un* X*.*

S'il existe une classe de polarités concurrentes pour laquelle le mot correspondant à $E$ ne vérifie pas cette propriété, alors $E$ n'a aucune chance de produire une analyse et peut donc être filtré.

### Exemple

Appliquons maintenant cette méthode à l'exemple *« Les perspectives de redémarrage de l'activité plongent »*, et considérons pour cela les deux étiquetages de la figure 6.2 du point de vue des polarités `cat` $\leftarrow$ `det` contenues dans les descriptions `NC`.

À chaque étiquetage correspond un mot sur $\{\text{L}, \text{X}, \text{R}\}$ :

- la description du nom commun `NC` contient un trait polarisé `cat` $\leftarrow$ `det` ; chaque exemplaire de la description du nom commun contribue donc un X ;
- les descriptions du déterminant `Det` et de la préposition avec déterminant déficient `PrepN` contiennent un compagnon hypothétique gauche (une polarité `cat` $\rightarrow$ `det`) ; chaque exemplaire de ces descriptions contribue donc un L ;
- les autres descriptions ne contiennent pas de compagnon hypothétique pour cette polarité.

Les mots correspondant à chaque étiquetage sont représentés à la figure 6.5.

| $\texttt{Det}_{les}$ | $\texttt{NC}_{perspectives}$ | $\texttt{PrepNP}_{de}$ | $\texttt{NC}_{redemarrage}$ | $\texttt{PrepN}_{de}$ | $\texttt{Det}_{l'}$ | $\texttt{NC}_{activite}$ | $\texttt{VIntr}_{evoluent}$ |
|---|---|---|---|---|---|---|---|
| L | X | | X | L | L | X | |

| $\texttt{Det}_{les}$ | $\texttt{NC}_{perspectives}$ | $\texttt{PrepN}_{de}$ | $\texttt{NC}_{redemarrage}$ | $\texttt{PrepNP}_{de}$ | $\texttt{Det}_{l'}$ | $\texttt{NC}_{activite}$ | $\texttt{VIntr}_{evoluent}$ |
|---|---|---|---|---|---|---|---|
| L | X | L | X | | L | X | |

FIGURE 6.5 – Mots correspondant aux étiquetages dans le langage des compagnons

Le mot correspondant à l'étiquetage incorrect est représenté avec une association de lettres possible, mais incomplète, en figure 6.6. Les deux premiers X sont en concurrence directe pour le premier L, l'un de ces deux X restera donc sans L.


FIGURE 6.6 – Mot correspondant à un étiquetage incorrect

Le mot correspondant à l'étiquetage correct est représenté avec ses associations de lettres en figure 6.7. Dans ce mot, chaque X peut trouver un L, en s'associant au L qui le précède immédiatement.

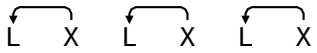



FIGURE 6.7 – Mot correspondant à un étiquetage correct

### Formalisation

Filtrer les étiquetages grammaticaux d'une phrase avec la méthode *ICP* consiste donc à construire des mots sur $\{\text{L}, \text{X}, \text{R}\}$ et à déterminer s'ils vérifient ou non la propriété 6.1.1.

Dans les sections suivantes de ce chapitre, nous définissons formellement ce problème de décision comme un problème d'appartenance à un langage. La section 6.2 donne les définitions et notations qui sont nécessaires pour construire des mots, des associations de lettres dans un mot et des permutations de lettres dans un mot. Dans la section 6.3, nous définissons le langage des compagnons, qui est prototypique au sens où il comporte le nombre minimum de lettres nécessaire pour représenter notre problème. Nous montrons que ce langage n'est pas un langage algébrique et, dans la section 6.4, qu'il est équivalent à un langage défini par un pomonoïde.

Dans la section 6.5, nous donnons enfin l'algorithme par flot (*streaming algorithm*) qui décide de l'appartenance d'un mot au langage des compagnons et nous prouvons la correction de cet algorithme. Nous montrons enfin, en section 6.6, comment utiliser l'appartenance au langage des compagnons pour filtrer les étiquetages grammaticaux.

## 6.2 Notions préliminaires et notations

### 6.2.1 Mots

Fixons au préalable quelques notations sur les mots. Étant donné un *alphabet* $\Sigma$, $(\Sigma^*, \epsilon, \cdot)$ désigne le monoïde des mots finis sur $\Sigma$. $\epsilon$ est le mot vide et $\cdot$ est l'opération de concaténation de mots. Étant donné $w \in \Sigma^*$, $|w|$ désigne sa taille, c'est-à-dire son nombre de lettres ; étant donnée une lettre $a \in \Sigma$, $|w|_a$ désigne le nombre d'occurences de la lettre $a$ dans $w$. Étant donnés $w \in \Sigma^*$ et $n \in \mathbb{N}$, $w^n$ désigne le mot formé par $n$ concaténations de $w$. Plus formellement, $w^0 = \epsilon$ et $w^{n+1} = w \cdot w^n$. Pour finir avec les notations, $w[k]$ désigne la $k$-ième lettre de $w$ et $w[i, \ldots, j] = w[i] \cdots w[j]$. Alors, $w = w[1] \cdot w[2] \cdots w[|w|] = w[1, \ldots, |w|]$.

### 6.2.2 Fonctions entières

Pour gérer les positions dans les mots, nous introduisons quelques notations sur les fonctions de $\mathbb{N}$ dans $\mathbb{N}$. Premièrement, nous écrivons $\mu(n) = \bot$ pour signifier que la fonction $\mu$ n'est pas définie pour $n$. Alors, le *domaine* $\mathcal{D}(\mu)$ de $\mu$ est défini par $\mathcal{D}(\mu) = \{n \in \mathbb{N} \mid \mu(n) \neq \bot\}$ et son *codomaine* $\mathcal{C}(\mu)$ est défini par $\{\mu(n) \in \mathbb{N} \mid n \in \mathcal{D}(\mu)\}$. La composition de fonctions est notée $\mu \circ \tau$. Pour une fonction injective $\mu$, nous notons $\mu^{-1}$ la fonction inverse de domaine $\mathcal{D}(\mu^{-1}) = \mathcal{C}(\mu)$ qui associe à $i$ l'unique $j$ tel que $\mu(j) = i$. Une fonction $\mu$ est monotone si pour tous $i, j \in \mathcal{D}(\mu), i < j \implies \mu(i) < \mu(j)$. Dans ce contexte, les fonctions monotones sont injectives.

Soit **Id** la fonction identité. Étant donnés $i, j \in \mathbb{N}$ et une fonction $\mu$, la fonction $\mu[i \mapsto j]$ associe $j$ à $i$ et $\mu(x)$ à tout $x \neq i$. Nous écrivons $\mu[i \mapsto \bot]$ pour positionner de façon explicite la valeur de $\mu$ en $i$ comme indéfinie. La notation entre crochets est étendue directement aux séquences : $\mu[i_1 \mapsto j_1, \ldots, i_k \mapsto j_k]$.

Dans ce qui suit, nous utilisons des fonctions pour décrire des relations entre des lettres d'un mot et pour gérer les transformations sur les mots telles que des permutations de symboles. Nous utilisons la lettre $\mu$ dans le premier cas et $\sigma$ dans le second. Le *décalage* d'une relation $\mu$ par une transformation $\sigma$ est défini comme étant $\mu^\sigma = \sigma \circ \mu \circ \sigma^{-1}$.

Un *enchâssement* (en anglais *embedding*) d'un mot $w$ dans $w'$ est donné par une fonction monotone $\sigma$ telle que pour tout $1 \leq i \leq |w|$, $w'[\sigma(i)] = w[i]$. Un tel enchâssement est noté $w \to_\sigma w'$.

## 6.3 Le langage des compagnons

Soit $C$ l'alphabet de trois *lettres* $C = \{\text{X}, \text{L}, \text{R}\}$. La lettre L (respectivement R) est appelée le compagnon hypothétique gauche (resp. droit) de X.

**Définition 6.3.1.** Le *langage des compagnons* CL est l'ensemble de mots $w \in C^*$ tel qu'il existe une fonction injective de *correspondance* (en anglais *matching*) $\mu : \mathbb{N} \to \mathbb{N}$ qui vérifie :
- $\forall\ 1 \leq i \leq |w|,\ w[i] = \text{X} \Rightarrow w[\mu(i)] \in \{\text{L}, \text{R}\}$,
- $\forall\ 1 \leq i \leq |w|,\ (w[i] = \text{X} \wedge w[\mu(i)] = \text{L}) \Rightarrow \mu(i) < i$,
- $\forall\ 1 \leq i \leq |w|,\ (w[i] = \text{X} \wedge w[\mu(i)] = \text{R}) \Rightarrow \mu(i) > i$.

Intuitivement, un mot $w$ appartient à CL ssi chaque lettre X de $w$ peut être liée soit à une lettre L située à sa gauche, soit à une lettre R à sa droite et toute lettre L ou R est liée à au plus une lettre X.

Dans la définition ci-dessus, il est clair que l'appartenance d'un mot $w$ à CL ne dépend pas de la valeur des correspondances pour les indices $i$ tels que $w[i] \neq \text{X}$ ou $i > |w|$. Par conséquent, dans ce qui suit, le domaine de toute correspondance sur un mot $w$ est considéré être $\{i \mid w[i] = \text{X}\}$. Pour montrer qu'un mot est dans CL, nous le représentons avec sa correspondance de la façon suivante (notons qu'il peut y avoir plus d'une correspondance, nous en donnons deux ci-dessous) :


où les flèches indiquent la correspondance. Quelques autres exemples de mots appartenant à CL sont donnés par la proposition suivante.

**Proposition 6.3.1.** *L'ensemble intersection* $\text{CL} \cap \{\text{L}, \text{X}\}^*$ *est l'ensemble des préfixes des mots bien équilibrés.* L *joue le rôle de la parenthèse ouvrante et* X *celui de la parenthèse fermante.* $\text{CL} \cap \{\text{X}, \text{R}\}^*$ *est l'ensemble des suffixes des mots bien équilibrés. Dans ce cas,* X *joue le rôle de la parenthèse ouvrante et* R *celui de la parenthèse fermante.*

*Démonstration.* Un mot $w$ est bien équilibré ssi $w$ contient exactement autant de parenthèses ouvrantes que de fermantes et tout préfixe de $w$ contient plus de parenthèses ouvrantes que de fermantes. Ainsi, un mot est un préfixe d'un mot bien équilibré ssi lui-même et chacun de ses préfixes contient plus de parenthèses gauches que de parenthèses droites. Cette propriété est une conséquence immédiate de la Définition 6.3.1. □

**Théorème 6.3.2.** *Le langage des compagnons n'est pas un langage algébrique.*

Pour démontrer ce théorème, nous utilisons deux lemmes, dont le Lemme de la pompe pour les langages algébriques (notés CFL pour *Context Free Languages*) [HMU06] :

**Lemme 6.3.3.** *Soit* $\mathcal{L}$ *un CFL et* $\mathcal{M}$ *un langage régulier.* $\mathcal{L} \cap \mathcal{M}$ *est un CFL.*

**Lemme 6.3.4** (Lemme de la pompe pour les CFL)**.** *Soit $\mathcal{L}$ un CFL. Il existe une constante $n \in \mathbb{N}$ telle que pour tout mot $w \in \mathcal{L}$ avec $|w| > n$, il existe des mots $x, u, y, v$ et $z$ tels que $w = xuyvz$ et :*

1. $|uyv| \leq n$ ;
2. $|uv| > 0$ ;
3. $\forall i \geq 0, xu^i yv^i z \in \mathcal{L}$.

*Théorème 6.3.2.* Supposons que CL est un CFL, alors le langage $\text{CL}_\diamond = \text{CL} \cap (\text{X}^+\text{L}^+\text{R}^+\text{X}^+)$, qui est son intersection avec un langage régulier, est encore un CFL. Dans le langage $\text{CL}_\diamond$, les lettres X au début du mot sont nécessairement liées aux lettres R et les X à la fin du mot sont nécessairement liées aux lettres L ; cela implique que $\text{CL}_\diamond = \{\text{X}^i\text{L}^j\text{R}^k\text{X}^l \mid 1 \leq i \leq k \text{ and } 1 \leq l \leq j\}$. Étant donné $w = \text{X}^i\text{L}^j\text{R}^k\text{X}^l \in \text{CL}_\diamond$, les sous-mots $\text{X}^i$, $\text{L}^j$, $\text{R}^k$ et $\text{X}^l$ de $w$ sont appelés les *segments* de $w$.

Prouvons maintenant que $\text{CL}_\diamond$ ne satisfait pas le Lemme de la pompe. Par l'absurde, supposons $n$ défini comme dans le Lemme 6.3.4. Soit $w = \text{X}^n\text{L}^n\text{R}^n\text{X}^n$ et sa décomposition suivant le Lemme de la pompe $w = xuyvz$.

Premièrement, observons que ni $u$ ni $v$ ne peut être composé de deux lettres différentes. En effet, si c'était le cas, par le Lemme 6.3.4 nous aurions $xu^2yv^2z \in \text{CL}_\diamond$. Cependant $xu^2yv^2z \notin (\text{X}^+\text{L}^+\text{R}^+\text{X}^+)$, ce qui contredit le Lemme. Pour conclure, $u$ et $v$ sont des sous-mots de segments de $w$.

Deuxièmement, à cause de l'item 1 du Lemme 6.3.4, comme les segments sont de longueur $n$, $u$ et $v$ sont nécessairement des sous-mots du même segment ou de deux segments consécutifs dans $w$. Les deux observations ci-dessus restreignent l'analyse par cas aux cas suivants :

- si $u = \text{X}^i$ et $v = \text{X}^j$ (au début du mot), alors $xu^2yv^2z = \text{X}^{n+i+j}\text{L}^n\text{R}^n\text{X}^n \notin \text{CL}_\diamond$ par l'item 2 du Lemme 6.3.4 ;
- si $u = \text{X}^i$ et $v = \text{L}^j$, alors $xu^2yv^2z = \text{X}^{n+i}\text{L}^{n+j}\text{R}^n\text{X}^n \notin \text{CL}_\diamond$ ;
- si $u = \text{L}^i$ et $v = \text{L}^j$, alors $xu^0yv^0z = \text{X}^n\text{L}^{n-i-j}\text{R}^n\text{X}^n \notin \text{CL}_\diamond$ ;
- si $u = \text{L}^i$ et $v = \text{R}^j$, alors $xu^0yv^0z = \text{X}^n\text{L}^{n-i}\text{R}^{n-j}\text{X}^n \notin \text{CL}_\diamond$ ;
- les autres cas sont symétriques aux précédents.

Par conséquent, quelle que soit la décomposition, l'item 3 du Lemme 6.3.4, $\forall i \geq 0, xu^iyv^iz \in \text{CL}_\diamond$, n'est pas vérifié. Finalement, $\text{CL}_\diamond$ n'est pas un CFL et donc CL n'est pas un CFL. □

## 6.4 Présentation algébrique

D'un point de vue algébrique, le triplet $(\text{CL}, \epsilon, \cdot)$ est un monoïde. Cependant, comme CL n'est pas un langage algébrique, il n'est pas régulier, et donc, d'après le Théorème de Myhill-Nerode, il n'existe aucune présentation monoïdale finie de CL. CL n'a pas non plus la structure d'un prégroupe [Lam99] : de fait, les prégroupes et les grammaires algébriques sont faiblement équivalents [Bus01]. Toutefois, le langage CL peut être présenté comme un monoïde partiellement ordonné, ou *pomonoïde*. Un pomonoïde est un quadruplet $(X, 1, \times, \preceq)$ tel que $(X, 1, \times)$ est un

monoïde et $(X, \preceq)$ est un ordre partiel. De plus, pour tout $x \preceq y$, pour tout $z : x \times z \preceq y \times z$ et $z \times x \preceq z \times y$. Le langage défini par un pomonoïde est le sous-ensemble $L \subseteq X$ tel que $x \in L$ ssi $x \preceq 1$.

**Théorème 6.4.1.** CL *est le langage correspondant au pomonoïde* $(C^*, \epsilon, \cdot, \leq)$ *avec* $\leq$ *défini inductivement par (les variables* $x, y, z, t$ *appartenant à* $C^*$*) :*

$$\frac{}{\mathrm{L} \cdot y \cdot \mathrm{X} \leq y}(\mathrm{Li}) \qquad \frac{}{\mathrm{X} \cdot y \cdot \mathrm{R} \leq y}(\mathrm{Ri}) \qquad \frac{}{\mathrm{L} \leq \epsilon}(\mathrm{Lw}) \qquad \frac{}{\mathrm{R} \leq \epsilon}(\mathrm{Rw})$$

$$\frac{}{x \leq x}(\mathrm{Ref}) \qquad \frac{x \leq y \wedge y \leq z}{x \leq z}(\mathrm{Tr}) \qquad \frac{x \leq y \wedge z \leq t}{x \cdot z \leq y \cdot t}(\mathrm{Mon})$$

*Démonstration.* Supposons que $w \in$ CL, nous montrons que $w \leq \epsilon$ par induction sur la longueur de $w$.

- Si $w = \epsilon$, la conclusion est immédiate par (Ref).
- Sinon $w = A.w'$.
  - Si $w$ ne contient aucun X, $w' \in$ CL et par induction $w' \leq \epsilon$. Par (Rw) si $A =$ R ou (Lw) si $A =$ L, et (Mon), $w \leq \epsilon$.
  - Sinon $w$ contient une lettre X à une position $i$. Soit $\mu$ une correspondance sur $w$, nous supposons $\mu(i) < i$, c'est-à-dire que $w = w[1, \ldots, \mu(i) - 1]\mathrm{L}w[\mu(i) + 1, \ldots, i - 1]\mathrm{X}w_3$. Comme $u = w[1, \ldots, \mu(i) - 1]w[\mu(i) + 1, \ldots, i - 1]w_3 \in$ CL, par induction, $u \leq \epsilon$. Par (Li) et (Tr), $\mathrm{L}w[\mu(i) + 1, \ldots, i - 1]\mathrm{X} \leq \epsilon$. En appliquant deux fois (Mon), $w \leq \epsilon$. Le cas $\mu(i) > i$ est traité de façon similaire.

Dans le sens opposé, nous commençons par montrer par induction sur la structure de l'ordre $\leq$ que si $w \leq w'$, alors $w$ est une extension compatible de $w'$. Par extension compatible, nous entendons un couple $(\sigma, \nu)$ tel que $\sigma$ est un enchâssement $w' \to_\sigma w$, et $\nu$ est une correspondance partielle sur les lettres qui sont dans $w$ mais pas dans $\sigma(w')$, c'est-à-dire que $\nu$ est une fonction injective dont le domaine est $\{i \leq |w| \mid w[i] = \mathrm{X} \wedge i \notin \mathcal{C}(\sigma)\}$, telle que $\mathcal{C}(\nu) \cap \mathcal{C}(\sigma) = \emptyset$, et soit $\nu(i) < i$ et $w[\nu(i)] =$ L, soit $\nu(i) > i$ et $w[\nu(i)] =$ R.

La propriété est immédiate pour les règles (Lw, Rw, Ref).

Pour (Li) (resp. (Ri)), posons $\sigma = x \in [1, \ldots, |y|] \mapsto x + 1$ et $\nu(|y| + 2) = 1$ (resp. $\nu(1) = |y| + 2$).

Pour (Tr), soient $(\sigma_1, \nu_1)$ et $(\sigma_2, \nu_2)$ les deux extensions compatibles obtenues par induction pour $x \leq y$ et $y \leq z$ respectivement. Alors, $\sigma = \sigma_1 \circ \sigma_2$ et

$$\nu : \begin{cases} i \mapsto \nu_1(i) \text{ si } i \in \mathcal{D}(\nu_1) \\ i \mapsto \nu_2^{\sigma_1}(i) \text{ si } i \in \mathcal{C}(\sigma_1) \cap \mathcal{D}(\nu_2) \end{cases}$$

Pour (Mon), soient $(\sigma_1, \nu_1)$ et $(\sigma_2, \nu_2)$ les deux extensions compatibles pour $x \leq y$ et $z \leq t$

respectivement. Alors, $(\sigma, \nu)$ définie ci-dessous est une extension compatible pour $x \cdot z \leq y \cdot t$.

$$\sigma : \begin{cases} i \mapsto \sigma_1(i) \text{ si } 0 < i \leq |y| \\ i \mapsto \sigma_2(i - |y|) + |x| \text{ si } |y| < i \leq |y| + |t| \end{cases}$$

et

$$\nu : \begin{cases} i \mapsto \nu_1(i) \text{ si } 0 < i \in \mathcal{D}(\nu_1) \\ i \mapsto \nu_2(i - |y|) + |x| \text{ si } i > |y| \text{ et } i - |y| \in \mathcal{D}(\nu_2) \end{cases}$$

Pour conclure, soient $w \leq \epsilon$ et $(\sigma, \nu)$ une extension compatible comme celles que nous venons de construire. $\mu = \nu$ est en fait une correspondance sur $w$. □

## 6.5 Algorithme par flot (*Streaming algorithm*)

### 6.5.1 Problème d'appartenance

**Définition 6.5.1** (Formes canoniques)**.** Soit CC le sous-ensemble de mots $w$ de $\Sigma^*$, tel que $w = \mathrm{R}^h(\mathrm{XR})^i\mathrm{X}^j(\mathrm{LX})^k\mathrm{L}^m$, $(h, i, j, k, m) \in \mathbb{N}^5$.

**Proposition 6.5.1.** *Un mot* $w = \mathrm{R}^h(\mathrm{XR})^i\mathrm{X}^j(\mathrm{LX})^k\mathrm{L}^m \in$ CC *appartient à* CL *ssi* $j = 0$.

*Démonstration.* Supposons $j = 0$, soit $\mu$ défini par : $\mu(h + 2 * \ell - 1) = h + 2 * \ell$ pour $\ell \in 1, \ldots, i$ et $\mu(h + 2 * i + 2 * \ell) = h + 2 * i + 2 * \ell - 1$ pour $\ell = 1, \ldots, k$. C'est une correspondance sur $w$. Dans le sens opposé, supposons $j \neq 0$, considérons l'occurrence de la lettre $\mathrm{X} = w[h + 2 * i + 1]$. Il n'y a ni L sur sa gauche, ni R sur sa droite. Donc, $w \notin$ CL. □

On peut observer que tout mot dans CC ∩ CL a une unique fonction de correspondance.

### 6.5.2 Lemmes techniques

Les quatre lemmes suivants établissent des résultats de permutation : certaines modifications locales peuvent être utilisées pour transformer le mot sans risque de modifier son statut d'appartenance à CL.

**Lemme 6.5.2** (Indépendance de L et R par rapport à la distance)**.**
1. *Si* $w_1 \cdot A\mathrm{L} \cdot w_2 \in$ CL *alors* $w_1 \cdot \mathrm{L}A \cdot w_2 \in$ CL *;*
2. *Si* $w_1 \cdot \mathrm{R}A \cdot w_2 \in$ CL *alors* $w_1 \cdot A\mathrm{R} \cdot w_2 \in$ CL.

*Démonstration.* Soient $k = |w_1|$ et $\sigma = \mathbf{Id}[k + 1 \mapsto k + 2, k + 2 \mapsto k + 1]$.

1. Il est simple de vérifier que, si $\mu$ est une correspondance sur $w_1 \cdot A\mathrm{L} \cdot w_2$, alors $\mu^\sigma$ est une correspondance sur $w_1 \cdot \mathrm{L}A \cdot w_2$.
2. Il est également simple de vérifier que, si $\mu$ est une correspondance sur $w_1 \cdot \mathrm{R}A \cdot w_2$, alors $\mu^\sigma$ est une correspondance sur $w_1 \cdot A\mathrm{R} \cdot w_2$.

□

**Lemme 6.5.3** (Convertibilité LR/RL). $w_1 \cdot \text{LR} \cdot w_2 \in \text{CL}$ *ssi* $w_1 \cdot \text{RL} \cdot w_2 \in \text{CL}$

*Démonstration.* Soient $k = |w_1|$ et $\sigma = \mathbf{Id}[k+1 \mapsto k+2, k+2 \mapsto k+1]$. Il est simple de vérifier que, si $\mu$ est une correspondance sur $w_1 \cdot \text{LR} \cdot w_2$, alors $\mu^\sigma$ est une correspondance sur $w_1 \cdot \text{RL} \cdot w_2$. La réciproque est donnée par le Lemme 6.5.2. □

**Lemme 6.5.4** (Convertibilité LLX/LXL). $w_1 \cdot \text{LLX} \cdot w_2 \in \text{CL}$ *ssi* $w_1 \cdot \text{LXL} \cdot w_2 \in \text{CL}$.

*Démonstration.* Soient $k = |w_1|$ et $\mu$ une correspondance sur $w_1 \cdot \text{LLX} \cdot w_2$ ; par analyse par cas :

- Si $\mu(k+3) = k+2$ (*i.e.* le X est lié au second L), alors pour $\sigma_1 = \mathbf{Id}[k+2 \mapsto k+1, k+3 \mapsto k+2, k+1 \mapsto k+3]$, $\mu^{\sigma_1}$ est une correspondance sur $w_1 \cdot \text{LXL} \cdot w_2$.
- Si $\mu(k+3) = k+1$ (*i.e.* le X est lié au premier L), alors pour $\sigma_2 = \mathbf{Id}[k+3 \mapsto k+2, k+2 \mapsto k+3]$, $\mu^{\sigma_2}$ est une correspondance sur $w_1 \cdot \text{LXL} \cdot w_2$.
- Sinon (*i.e.* le X est lié à $w_1$ ou $w_2$), alors $\sigma_1$ et $\sigma_2$ peuvent être tous deux utilisés pour construire une correspondance (soit $\mu^{\sigma_2}$ soit $\mu^{\sigma_1}$) sur $w_1 \cdot \text{LXL} \cdot w_2$.

La réciproque est donnée par le Lemme 6.5.2. □

**Lemme 6.5.5** (Convertibilité LXX/XLX). $w_1 \cdot \text{LXX} \cdot w_2 \in \text{CL}$ *ssi* $w_1 \cdot \text{XLX} \cdot w_2 \in \text{CL}$.

*Démonstration.* Soient $k = |w_1|$ et $\mu$ une correspondance sur $w_1 \cdot \text{LXX} \cdot w_2$ ; par analyse par cas :

- Si $\mu(k+2) = k+1$ (*i.e.* le premier X est lié au L), alors avec $\sigma_1 = \mathbf{Id}[k+1 \mapsto k+2, k+2 \mapsto k+3, k+3 \mapsto k+1]$, $\mu^{\sigma_1}$ est une correspondance sur $w_1 \cdot \text{XLX} \cdot w_2$.
- Sinon (*i.e.* le second X est lié au L ou aucun des deux X n'est lié au L), alors pour $\sigma_2 = \mathbf{Id}[k+2 \mapsto k+1, k+1 \mapsto k+2]$, $\mu^{\sigma_2}$ est une correspondance sur $w_1 \cdot \text{XLX} \cdot w_2$.

La réciproque est donnée par le Lemme 6.5.2. □

Lorsqu'on travaille sur des préfixes de mots, les permutations permises sont plus nombreuses. C'est le point du lemme suivant.

**Lemme 6.5.6** (Convertibilité des préfixes LXR/XRL). $\text{LXR} \cdot w \in \text{CL}$ *ssi* $\text{XRL} \cdot w \in \text{CL}$.

*Démonstration.* La première implication requiert une analyse par cas, soit $\mu$ une correspondance sur $\text{LXR} \cdot w$ :

- Si $\mu(2) = 1$, observons que $\mu^{-1}(3) = \bot$ (*i.e.* $w_1[3] = \text{R}$ ne peut être utilisé dans $\mu$) ; donc $\mu'$ peut être construit par une modification locale $\mu' = \mu[1 \mapsto 2, 2 \mapsto \bot]$.
- Si $\mu(2) \neq 1$, la correspondance $\mu$ peut être décalée sans risque en une correspondance $\mu^\sigma$ sur $w_2$, par la permutation $\sigma = \mathbf{Id}[1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 2]$.

La réciproque est donnée par le Lemme 6.5.2. □

Le lemme qui suit établit que les préfixes R et XR peuvent être supprimés sans danger de changer la situation d'appartenance du mot à CL.

**Lemme 6.5.7** (Préfixes R). *1.* $\text{R} \cdot w \in \text{CL}$ *ssi* $w \in \text{CL}$ *;*

*2.* $\text{XR} \cdot w \in \text{CL}$ *ssi* $w \in \text{CL}$ *;*

*3.* $\text{R}^*(\text{XR})^* \cdot w \in \text{CL}$ *ssi* $w \in \text{CL}$.

*Démonstration.* 1. Considérons $\sigma$ tel que $\sigma(1) = \bot$ et $\sigma(i) = i - 1$ pour tout $i > 1$. Si $\mu$ est une correspondance sur $\text{R} \cdot w$ alors $\mu^\sigma$ est une correspondance sur $w$. Réciproquement, si $\mu$ est une correspondance sur $w$ alors $\mu^{\sigma^{-1}}$ est une correspondance sur $\text{R} \cdot w$.

2. Considérons $\sigma$ tel que $\sigma(1) = \sigma(2) = \bot$ et $\sigma(i) = i - 2$ pour tout $i > 2$. Si $\mu$ est une correspondance sur $\text{XR} \cdot w$ alors $\mu^\sigma$ est une correspondance sur $w$. Réciproquement, si $\mu$ est une correspondance sur $w$ alors $\mu^{\sigma^{-1}}[1 \mapsto 2]$ est une correspondance sur $\text{XR} \cdot w$.

3. Conséquence immédiate des deux items précédents.

□

Le lemme qui suit garantit que pour tout $w \in \text{CL}$, le R le plus à gauche de $w$, quand il est utilisé pour saturer un X, peut en réalité être associé au X le plus à gauche.

**Lemme 6.5.8** (R le plus à gauche)**.** *Soit* $w = w_1 \cdot \text{X} \cdot w_2 \cdot \text{X} \cdot w_3 \cdot \text{R} \cdot w_4$ *avec* $w_2 \in \{\text{L}, \text{X}\}^*$ *;*

*Si* $w$ *a une correspondance* $\mu$*, avec* $\mu(|w_1|+|w_2|+2) = |w_1|+|w_2|+|w_3|+3$ *(i.e. le second* X *est lié au* R*), alors il existe une correspondance* $\mu'$ *sur* $w$ *telle que* $\mu(|w_1|+1) = |w_1|+|w_2|+|w_3|+3$ *(i.e. le premier* X *est lié au* R*).*

*Démonstration.* Nous définissons $i = |w_1| + 1$ comme l'index du premier X et $j = |w_1| + |w_2| + 2$ l'index du second X. Soit $\mu' = \mu[i \to \mu(j), j \to \mu(i)]$. Il reste uniquement à vérifier que $\mu'$ est une correspondance. Comme $w_2 \in \{\text{L}, \text{X}\}^*$, nous avons soit $\mu(i) < i$ et $w[\mu(i)] = \text{L}$ soit $\mu(i) > j$ et $w[\mu(i)] = \text{R}$. Dans les deux cas, $\mu'$ est clairement une correspondance sur $w$. □

**Lemme 6.5.9** (première transition R)**.** *Pour* $(j, k, m) \in \mathbb{N}^3$

$$w_1 = \text{X} \cdot \text{X}^j(\text{LX})^k \cdot \text{L}^m \cdot \text{R} \cdot w \in \text{CL} \iff w_2 = \text{XR} \cdot \text{X}^j(\text{LX})^k\text{L}^m \cdot w \in \text{CL}.$$

*Démonstration.* Soit $r = j + 2k + m + 2$ l'index du R le plus à gauche dans $w_1$. Soit $\sigma$ le décalage entre $w_1$ et $w_2$ défini par $\sigma(1) = 1$, $\sigma(i) = i + 1$ pour $1 < i < r$, $\sigma(r) = 2$, et $\sigma(i) = i$ for $i > r$. Soit $\mu$ une correspondance sur $w_1$ ; par analyse par cas :

- Si $\mu^{-1}(r) = \bot$ (*i.e.* le R n'est pas utilisé dans la correspondance), alors $\mu^\sigma$ est une correspondance sur $w_2$.
- Si $\mu^{-1}(r) \neq \bot$, par le Lemme 6.5.8, nous pouvons construire une nouvelle correspondance $\mu'$ sur $w_1$ telle que $\mu'(1) = r$ et alors $\mu'^\sigma$ est une correspondance sur $w_2$.

Réciproquement, soit $\mu$ une correspondance sur $w_2$, alors $\mu^{\sigma^{-1}}$ est une correspondance sur $w_1$. □

**Lemme 6.5.10** (seconde $R$-transition)**.** *Pour* $(k, m) \in \mathbb{N}^2$*,*

$$w_1 = (\text{LX}) \cdot (\text{LX})^k \cdot \text{L}^m \cdot \text{R} \cdot w \in \text{CL} \iff w_2 = (\text{XR}) \cdot (\text{LX})^k \cdot \text{L}^{m+1} \cdot w \in \text{CL}.$$

*Démonstration.* Considérons $w_a = (\text{LX}) \cdot \text{R} \cdot (\text{LX})^k \cdot \text{L}^m \cdot w$ et $w_b = (\text{XR}) \cdot \text{L} \cdot (\text{LX})^k \cdot \text{L}^m \cdot w$. En fait, nous montrons

$$w_1 \in \text{CL} \iff w_a \in \text{CL} \iff w_b \in \text{CL} \iff w_2 \in \text{CL}$$

- $w_1 \in \text{CL} \iff w_a \in \text{CL}$ : la démonstration est similaire à celle du Lemme 6.5.9, à la différence que le X le plus à gauche est à l'index 2 et non 1.
- $w_a \in \text{CL} \iff w_b \in \text{CL}$ : par le Lemme 6.5.6.
- $w_b \in \text{CL} \iff w_2 \in \text{CL}$ : comme les deux mots $w_b$ et $w_2$ sont LLX/LXL-convertibles, l'équivalence est donnée par plusieurs applications du Lemme 6.5.4.

□

### 6.5.3 L'algorithme

Nous proposons un algorithme par flot (*streaming algorithm*), c'est-à-dire un algorithme en temps linéaire, espace logarithmique et une seule passe de gauche à droite pour lire l'entrée, qui reconnaît le langage des mots analysables. Si nous voulons récupérer les liens atomes, l'algorithme est toujours en temps linéaire et une passe mais utilise un espace linéaire.

À tout mot dans $\mathcal{C}^*$, nous associons un vecteur dans $\mathbb{N}^3$ qui peut être calculé de façon incrémentale, à coût constant à chaque étape : c'est la fonction $f$ de la Définition 6.5.2 ci-dessous. Les trois composantes de chaque vecteur correspondent respectivement à :

- le nombre de X en attente (comme le mot est lu de gauche à droite, la seule façon de saturer un tel X est de trouver un R sur sa droite) ;
- le nombre d'appariements "LX" construits jusque là ;
- le nombre de L disponibles.

**Définition 6.5.2.**

Soit $f : C^* \to \mathbb{N}^3$ définie par :

$$\frac{}{f(\epsilon) = (0,0,0)}[Ax] \qquad \frac{f(w) = (j,k,m)}{f(w \cdot \text{L}) = (j,k,m+1)}[\text{L}]$$

$$\frac{f(w) = (j,k,m+1)}{f(w \cdot \text{X}) = (j,k+1,m)}[\text{X}_1] \qquad \frac{f(w) = (j,k,0)}{f(w \cdot \text{X}) = (j+1,k,0)}[\text{X}_2]$$

$$\frac{f(w) = (j+1,k,m)}{f(w \cdot \text{R}) = (j,k,m)}[\text{R}_1] \qquad \frac{f(w) = (0,k+1,m)}{f(w \cdot \text{R}) = (0,k,m+1)}[\text{R}_2] \qquad \frac{f(w) = (0,0,m)}{f(w \cdot \text{R}) = (0,0,m)}[\text{R}_3]$$

Soit $f(w) = (j,k,m)$, nous définissons $w^{\Downarrow}$ comme étant le mot $\text{X}^j \cdot (\text{LX})^k \cdot \text{L}^m$.

**Lemme 6.5.11** (Lemme de lecture)**.** *Pour $a \in C$, alors $w_1^{\Downarrow} \cdot a \cdot w_2 \in \text{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 \in \text{CL}$.*

*Démonstration.* Nous faisons une analyse par cas, les cas correspondant aux 6 dernières règles de la Définition 6.5.2 :

- règle [L] :
  $a = \mathrm{L}$ et $w_1^{\Downarrow} \cdot a = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{L}^{m+1} = (w_1 \cdot a)^{\Downarrow}$.
  L'équivalence $w_1^{\Downarrow} \cdot a \cdot w_2 \in \mathrm{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 \in \mathrm{CL}$ est triviale.

- règle [X$_1$] :
  $a = \mathrm{X}$, $w_1^{\Downarrow} \cdot a = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{L}^{m+1} \cdot \mathrm{X}$ et $(w_1 \cdot a)^{\Downarrow} = \mathrm{X}^j \cdot (\mathrm{LX})^{k+1} \cdot \mathrm{L}^m$.
  En utilisant $m$ fois le Lemme de convertibilité LLX/LXL,

$$w_1^{\Downarrow} \cdot a \cdot w_2 = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{L}^{m+1} \cdot \mathrm{X} \cdot w_2 \in \mathrm{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 = \mathrm{X}^j \cdot (\mathrm{LX})^{k+1} \cdot \mathrm{L}^m \cdot w_2 \in \mathrm{CL}$$

- règle [X$_2$] :
  $a = \mathrm{X}$, $w_1^{\Downarrow} \cdot a = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{X}$ et $(w_1 \cdot a)^{\Downarrow} = \mathrm{X}^{j+1} \cdot (\mathrm{LX})^k$.
  En utilisant $k$ fois le Lemme de convertibilité LXX/XLX,

$$w_1^{\Downarrow} \cdot a \cdot w_2 = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{X} \cdot w_2 \in \mathrm{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 = \mathrm{X}^{j+1} \cdot (\mathrm{LX})^k \cdot w_2 \in \mathrm{CL}$$

- règle [R$_1$] :
  $a = \mathrm{R}$, $w_1^{\Downarrow} \cdot a = \mathrm{X}^{j+1} \cdot (\mathrm{LX})^k \cdot \mathrm{L}^m \cdot \mathrm{R}$ and $(w_1 \cdot a)^{\Downarrow} = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{L}^m$.
  Par le Lemme 6.5.9 :

$$w_1^{\Downarrow} \cdot a \cdot w_2 = \mathrm{X}^{j+1} \cdot (\mathrm{LX})^k \cdot \mathrm{L}^m \cdot \mathrm{R} \cdot w_2 \in \mathrm{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 = \mathrm{X}^j \cdot (\mathrm{LX})^k \cdot \mathrm{L}^m \cdot w_2 \in \mathrm{CL}$$

- règle [R$_2$] :
  $a = \mathrm{R}$, $w_1^{\Downarrow} \cdot a = (\mathrm{LX})^{k+1} \cdot \mathrm{L}^m \cdot \mathrm{R}$ et $(w_1 \cdot a)^{\Downarrow} = (\mathrm{LX})^k \cdot \mathrm{L}^{m+1}$.
  Par le Lemme 6.5.10 :

$$w_1^{\Downarrow} \cdot a \cdot w_2 = (\mathrm{LX})^{k+1} \cdot \mathrm{L}^m \cdot \mathrm{R} \cdot w_2 \in \mathrm{CL} \iff (w_1 \cdot a)^{\Downarrow} \cdot w_2 = (\mathrm{LX})^k \cdot \mathrm{L}^{m+1} \cdot w_2 \in \mathrm{CL}$$

- règle [R$_3$] :
  $a = \mathrm{R}$, $w_1^{\Downarrow} \cdot a = \mathrm{L}^m \cdot \mathrm{R}$ et $(w_1 \cdot a)^{\Downarrow} = \mathrm{L}^m$.
  La propriété attendue est démontrée en deux étapes :

$$\begin{aligned} \mathrm{L}^m \cdot \mathrm{R} \cdot w_2 \in \mathrm{CL} &\iff \mathrm{R} \cdot \mathrm{L}^m \cdot w_2 \in \mathrm{CL} && \text{par } m \text{ applications du Lemme de convertibilité LR/RL,} \\ &\iff \mathrm{L}^m \cdot w_2 \in \mathrm{CL} && \text{par le Lemme 6.5.7(1).} \end{aligned}$$

□

**Proposition 6.5.12.** $w \in \mathrm{CL} \iff w^{\Downarrow} \in \mathrm{CL}$

*Démonstration.* Par induction sur $k$, nous montrons que $(w[1,\ldots,k])^{\Downarrow} w[k{+}1,\ldots,|w|] \in \mathrm{CL} \iff w \in \mathrm{CL}$. Pour $k = 0$, cette équivalence est immédiate ; pour $k > 0$ c'est un cas particulier du lemme précédent. □

**Machine** Nous pouvons maintenant donner l'algorithme de la machine qui reconnaît les mots du langage.

```
bool xlr(w){
 var x = 0, lx = 0, l = 0;
 for(i = 1; i < |w|; i++){
  switch(w[i]){
     case 'L':
       l = l + 1;
     case 'X':
       if(l == 0)
         x = x + 1;
       else
         lx = lx + 1;
         l = l - 1;
    case 'R':
       if (x > 0)
         x = x - 1;
       else if (lx > 0)
         lx = lx - 1;
         l = l + 1;
  }
 }
 return (x == 0)
}
```

**Théorème 6.5.13.** *$w \in$ CL ssi $w$ est reconnu par la machine.*

*Démonstration.* La machine implante la fonction $f$ de la Définition 6.5.2 ; c'est-à-dire qu'à la fin de la boucle `for`, étant donné $f(w) = (j, k, m)$, nous avons $\mathtt{x} = j$, $\mathtt{lx} = k$ et $\mathtt{l} = m$. Alors, par la Proposition 6.5.12 et la Proposition 6.5.1, $w \in$ CL $\iff \mathtt{x} = 0$. □

## 6.6 Filtrage des étiquetages par appartenance au langage des compagnons

### 6.6.1 Projection d'un étiquetage pour une polarité

Étant donnée une grammaire $\mathcal{G}$ et des structures initiales $t_i, t_j \in \mathcal{G}$, nous notons :
- $\#_{\mathrm{L}}(t_i : p\ ,\ t_j)$ le nombre de polarités de $t_j$ qui appartiennent à $\mathcal{L}_{t_i:p}$ ;
- $\#_{\mathrm{R}}(t_i : p\ ,\ t_j)$ le nombre de polarités de $t_j$ qui appartiennent à $\mathcal{R}_{t_i:p}$.

**Définition 6.6.1** (Projection LXR)**.** Dans une grammaire $\mathcal{G}$, la **projection lxr** d'une structure initiale $t_j \in \mathcal{G}$ pour une polarité $p$ d'une structure initiale $t_i \in \mathcal{G}$, est :

$$\rho(t_i : p, t_j) = \begin{cases} \text{R}^a\text{XL}^b & \text{si } t_j = t_i \\ \text{R}^a\text{L}^b & \text{sinon} \end{cases}$$

avec $a = \#_{\text{R}}(t_i : p\ ,\ t_j)$ et $b = \#_{\text{L}}(t_i : p\ ,\ t_j)$.

Étant donnée une polarité $p$ d'une structure $t_i$ d'une grammaire $\mathcal{G}$, la projection LXR permet d'associer à toute structure initiale $t_j$ de $\mathcal{G}$ un mot sur $C = \{\text{L}, \text{X}, \text{R}\}^*$. Ce mot représente le nombre de compagnons hypothétiques de $p$ contenus dans $t_j$, ainsi que leur directionnalité.

La concaténation, dans l'ordre, des projections LXR des structures d'un étiquetage pour une polarité donnée, permet de construire un mot sur $C$. Ce mot contient autant de X que d'instances de la polarité $p$ et autant de L et R que de compagnons hypothétiques gauches et droits de $p$ dans l'étiquetage. La concaténation suivant l'ordre des ancres des étiquettes, l'ordre des lettres reflète correctement les possibilités d'accrochage de l'analyse.

### 6.6.2 Principe du compagnonnage entier

Si un étiquetage a une solution, alors tous les mots que l'on peut construire de la sorte pour toutes les polarités de toutes les descriptions de l'étiquetage appartiennent au langage des compagnons.

**Proposition 6.6.1** (Principe du compagnonnage entier)**.** *Si un étiquetage grammatical* $[t_1, \ldots, t_n]$ *a une solution alors pour toute polarité linéaire* $p \in t_i$ *:*

$$\rho(t_i : p\ ,\ t_1) \cdot \rho(t_i : p\ ,\ t_2) \cdot \ldots \cdot \rho(t_i : p\ ,\ t_n) \in \text{CL}$$

Ce principe nous permet, comme dans le chapitre précédent, de filtrer les étiquetages grammaticaux.

### 6.6.3 Implantation sur automate du filtrage

L'algorithme par flot que nous avons présenté dans la section précédente permet de construire à la volée, pour chaque polarité, un automate d'étiquetages grammaticaux $\mathcal{A}_{t,\mathcal{L}_p,\mathcal{R}_p}$ à partir d'un automate d'étiquetages $\mathcal{A}$. Tout état de ce nouvel automate est étiqueté par un couple composé d'un état de l'automate $\mathcal{A}$ et du vecteur de $\mathbb{N}^3$ calculé par l'algorithme par flot.

## Conclusion

Dans ce chapitre, nous avons présenté une troisième méthode de filtrage des étiquetages grammaticaux qui est spécifique aux polarités linéaires. Cette méthode tient compte de la concurrence qui existe entre les polarités linéaires pour trouver un compagnon hypothétique. Pour formaliser

cette méthode, nous avons défini un langage, le langage des compagnons, dont nous avons montré qu'il n'était pas un langage algébrique et qu'il était équivalent à un langage défini par un pomonoïde. Nous avons enfin présenté un algorithme par flot qui décide de l'appartenance d'un mot au langage des compagnons et nous avons montré comment utiliser cet algorithme pour filtrer les étiquetages grammaticaux d'une phrase.

Dans le chapitre prochain, nous faisons le bilan, théorique et expérimental, des méthodes de filtrage, anciennes et nouvelles, des étiquetages pour les formalismes grammaticaux lexicalisés polarisés.

# Chapitre 7

# Bilan des méthodes de filtrage fondées sur les compagnons

Dans ce chapitre, nous faisons le bilan des méthodes symboliques de filtrage des étiquetages grammaticaux pour les formalismes lexicalisés polarisés. Nous évaluons d'abord, par des expérimentations sur corpus, les méthodes de filtrage proposées aux chapitres 5 et 6, en combinaison avec les méthodes de filtrage par bilan de polarités du chapitre 4. Nous présentons notre dispositif expérimental dans la section 7.1, puis les résultats des expérimentations dans la section 7.2. Nous résumons ensuite en section 7.3 ce que ces nouvelles méthodes apportent par rapport à l'existant, et nous pointons en section 7.4 leurs limites. Nous présentons enfin quelques-unes des perspectives ouvertes par ce travail en section 7.5, à la fois sur le plan théorique et sur le plan expérimental. Nous concluons enfin cette partie dans la section 7.5.6.

## 7.1 Dispositif d'évaluation

### 7.1.1 Ressources

Nous avons évalué nos méthodes de filtrage avec l'analyseur LEOPAR, utilisant la grammaire d'interaction du français FRIGRAM [Per07] pour étiqueter une partie du corpus du journal *Le Monde*.

#### Corpus

Le corpus utilisé est un ensemble de 42 038 phrases appartenant au corpus du journal *Le Monde*. La figure 7.1, page 113, contient entre autres informations le nombre de phrases de chaque longueur dans le corpus utilisé.

#### Grammaire

La version de la grammaire FRIGRAM utilisée contient $|\mathcal{U}| = 3\,788$ descriptions d'arbres non ancrées. Chaque DAP contient une dizaine de polarités ; au total, la grammaire contient 49 822 polarités composites non neutres.

### 7.1.2 Préparatifs

#### Compagnons hypothétiques pour les polarités virtuelles

Les compagnons hypothétiques ont été calculés sur la grammaire non ancrée. Nous avons toutefois procédé à une approximation supplémentaire : nous n'avons calculé et utilisé que les $1^{\text{ers}}$ compagnons hypothétiques des polarités virtuelles. Une polarité virtuelle requiert en effet la présence d'une ou de deux polarités : une seule polarité si c'est une polarité saturée, deux polarités si c'est un couple formé d'une polarité positive et une polarité négative. Comme toute polarité positive doit trouver une polarité négative et réciproquement, nous considérons, par approximation, qu'une polarité virtuelle cherche essentiellement soit une polarité saturée, soit une polarité positive qui va elle-même chercher une polarité négative.

Cette approximation limite le nombre d'ensembles de compagnons hypothétiques à manipuler et diminue la redondance de l'information.

#### Calcul des compagnons hypothétiques

Le calcul des compagnons hypothétiques de FRIGRAM prend 5 jours sur une machine de bureau et 25 minutes en utilisant 40 nœuds d'un cluster. Chaque nœud comporte 2 processeurs à 4 cœurs (Intel Xeon L5420 à 2.5 GHz) et 16 Go de RAM. Un cœur de l'un des nœuds exécute le processus qui est chargé du dispatching des processus de calcul sur les autres nœuds et de la fusion des données. Le calcul de la matrice se fait donc en parallèle dans (40 nœuds $*$ 2 processeurs $*$ 4 cœurs ) $-$ 1 cœur = 319 processus. Le temps total de calcul est donc de $319 * 25 = 7975$ minutes, soit 132 heures et 55 minutes. Le résultat est une matrice binaire, dont la compression produit un fichier de 800 Ko.

Il est évident que plus l'ensemble des compagnons hypothétiques d'une polarité est restreint, plus cet ensemble est intéressant pour le filtrage des étiquetages. Voici quelques chiffres concernant la grammaire FRIGRAM :

- 55 % des polarités n'ont des compagnons hypothétiques que d'un seul côté, c'est-à-dire que soit $\mathcal{L}$ soit $\mathcal{R}$ est vide ;
- le nombre médian de compagnons hypothétiques est de 39, c'est-à-dire que 50 % des polarités insaturées ont entre 1 et 39 compagnons hypothétiques dans la grammaire ;
- 81 % des polarités ont entre 1 et 417 compagnons hypothétiques ;
- une polarité a en moyenne 311 compagnons hypothétiques dans la grammaire ;
- 4 polarités ont chacune 6 380 compagnons hypothétiques, ce qui est le nombre maximal dans la grammaire.

### 7.1.3 Méthode

#### Filtres appliqués

Nous avons appliqué à chaque phrase du corpus les méthodes de filtrage dans l'ordre suivant :

1. *QCP*, sur l'automate initial des étiquetages grammaticaux ;
2. *BCPV* : *BCP* pour les seules polarités virtuelles ;
3. *QCP*, qui après *BCPV* filtre uniquement sur les polarités linéaires ;
4. *DET* : filtrage déterministe par bilan de polarités ;
5. *QCP* ;
6. *ACTIVE* : filtrage par bilan de polarités ;
7. *QCP* ;
8. *ICP*.

Cet ordre d'application des filtres a été déterminé de façon empirique ; il s'est avéré le plus efficace en moyenne sur ce corpus et avec cette grammaire. Notons que la deuxième étape est l'application de la méthode *BCP* aux seules polariéts virtuelles. Nos expérimentations préliminaires ont en effet montré que l'application de la méthode *BCP* aux polarités linéaires prenait autant de temps que l'application de la méthode *ICP* pour une efficacité moindre. Les applications successives du *QCP* sont liées à son fonctionnement particulier. Cette méthode peut en effet être appliquée à plusieurs automates d'étiquetage successifs en ayant à chaque fois un impact. De plus, cette méthode est intéressante car elle ne fait pas grossir l'automate auquel elle s'applique.

La méthode *ACTIVE* correspond au filtrage par bilan de polarités que nous avons présenté au chapitre 4. La méthode *DET* est un cas particulier de la méthode *ACTIVE*, qui construit uniquement les automates de bilan de polarités qui n'utilisent pas l'arithmétique d'intervalles (et qui sont donc « réellement déterministes »). Les automates de ce sous-ensemble sont moins coûteux à calculer, ce qui fait de *DET* une première étape intéressante avant d'appliquer *ACTIVE*.

#### Délai d'expiration

Nos méthodes de filtrage construisent un grand nombre d'automates puis font leur intersection, ce qui peut être très long. Nous avons par conséquent limité le temps total de filtrage à trois minutes par phrase. Si, au bout de trois minutes, l'application des méthodes de filtrage aux étiquetages d'une phrase n'est pas terminée, nous considérons que le filtrage ne termine pas en un temps raisonnable.

Nous avons déterminé ce seuil de façon expérimentale, par rapport aux temps de calcul de nos méthodes de filtrage. Avec un seuil plus bas, le délai n'était pas suffisant pour de nombreuses phrases. Au-delà de ce seuil, le nombre de phrases pour lesquelles l'ensemble des filtres est appliqué n'augmente que très lentement : les chances de terminer le filtrage dans la minute suivante sont très faibles.

**Traitement des résultats**

Un dernier commentaire doit être fait sur le traitement des résultats pour les nombres moyens d'étiquetages par phrase. Comme nous l'avons dit au début de cette partie, le nombre $n$ d'étiquetages grammaticaux est a priori exponentiel par rapport à la longueur de la phrase. Nous considérons par conséquent son logarithme en base 10 ($\log_{10}$, noté simplement log sur les graphes). De plus, comme nous utilisons un corpus brut, certaines phrases sont considérées comme incorrectes par la grammaire ; dans ce cas, il peut arriver que les méthodes de filtrage enlèvent tous les étiquetages. Pour éviter les valeurs indéfinies lorsque $n = 0$, nous considérons en fait $\log_{10}(1 + n)$.

## 7.2 Résultats expérimentaux

### 7.2.1 Applicabilité des méthodes de filtrage

La figure 7.1 montre l'applicabilité des méthodes de filtrage selon la longueur des phrases. La partie grise de chaque barre représente le nombre de phrases sur lesquelles toutes les méthodes de filtrage ont pu être appliquées avant l'expiration du délai de 3 minutes ; les parties colorées représentent le nombre de phrases pour lesquelles le délai a expiré. La couleur de chaque partie indique la méthode qui était en cours d'application lors de l'expiration du délai. Sur toutes les phrases de longueur inférieure à 15 mots, tous les filtres s'appliquent en moins de trois minutes. Jusqu'à 21 mots, l'ensemble des filtres s'applique en moins de trois minutes pour 97.8 % des phrases de chaque longueur. Au-delà de 21 mots, la proportion de phrases pour lesquelles le filtrage est terminé au bout de trois minutes diminue rapidement : 90 % pour les phrases de longueur 24, 79.3 % pour les phrases de longueur 27, 69 % pour les phrases de longueur 30.

Les méthodes les plus fréquemment interrompues par l'expiration du délai sont, par ordre de fréquence décroissante, la méthode *ICP*, en magenta, la méthode *ACTIVE*, en kaki, la méthode *BCPV*, en bleu et les différentes applications de la méthode *QCP*. Les méthodes *ICP*, *ACTIVE* et *BCPV* construisent des automates très dépliés et potentiellement très différents les uns des autres, dont l'intersection est potentiellement très coûteuse en temps et en espace.

Les différentes applications du *QCP* sont également régulièrement interrompues par l'expiration du délai. Comme nous l'avons évoqué au chapitre 5, le temps de calcul de la méthode *QCP* dépend directement de la taille de l'automate. Or, les méthodes de filtrage autres que *QCP* déplient l'automate. Par conséquent, le *QCP* peut prendre un temps très important lorsqu'il est appliqué à l'automate produit par une autre méthode de filtrage.

### 7.2.2 Coût en temps des méthodes de filtrage

La figure 7.2 représente le temps moyen d'application des méthodes de filtrage pour chaque longueur de phrase. La courbe correspondant à une méthode de filtrage est en trait plein tant que cette méthode n'a jamais été interrompue sur une phrase par l'expiration du délai de 3 minutes.

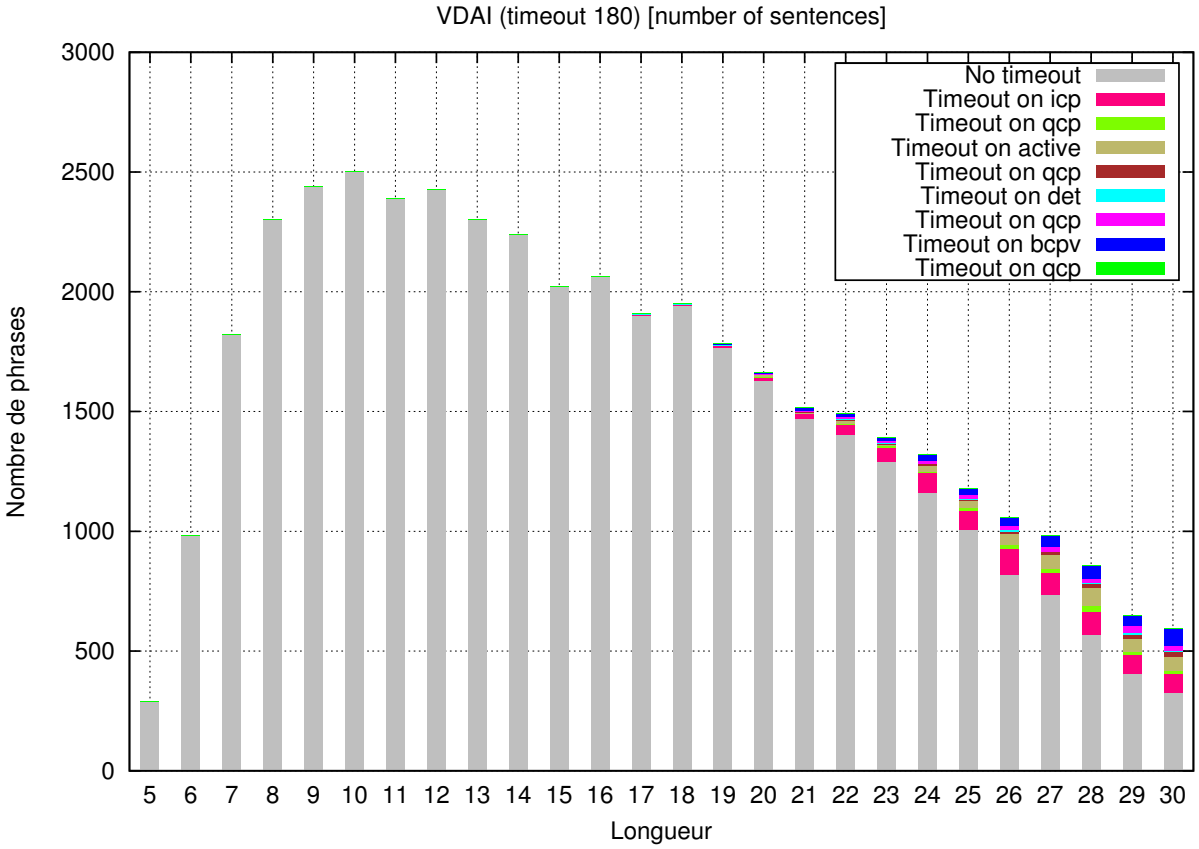



FIGURE 7.1 – Applicabilité des méthodes de filtrage selon la longueur des phrases

Seule l'application du *QCP* sur l'automate initial s'applique à toutes les phrases, quelle que soit leur longueur. Les autres méthodes s'appliquent à toutes les phrases sans exception jusqu'à la longueur 15 (16 pour *BCPV*).

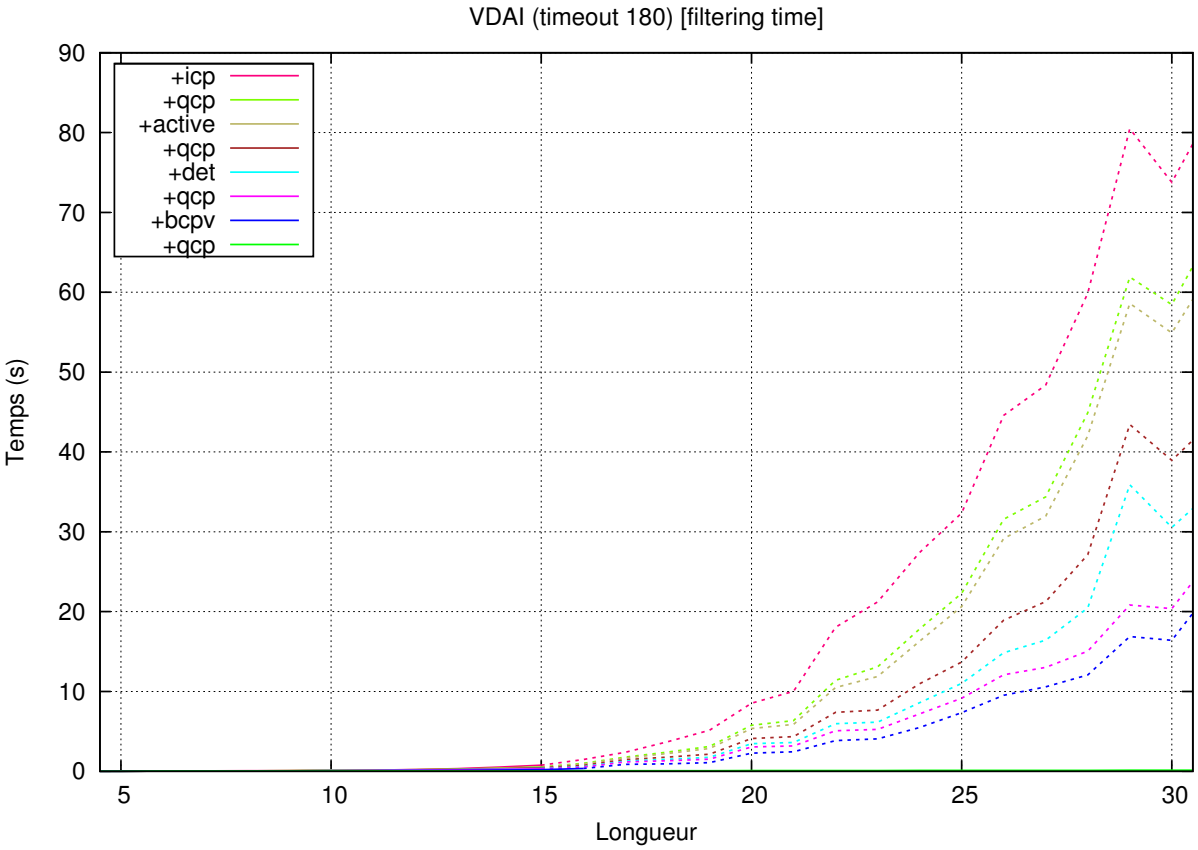



FIGURE 7.2 – Coût en temps des méthodes de filtrage selon la longueur des phrases

La première application du *QCP* a lieu sur l'automate initial : le temps est quasi linéaire et inférieur à 0.2 secondes même pour les phrases longues. Les autres méthodes sont plus coûteuses : *BCP* prend, uniquement sur les polarités virtuelles (*BCPV*), jusqu'à 16 secondes en moyenne pour les phrases de longueur 29 ; *DET* prend jusqu'à 15 secondes en moyenne pour les phrases de longueur 29 ; *ACTIVE* prend jusqu'à 16 secondes en moyenne pour les phrases de longueur 30 ; *ICP* prend jusqu'à 19 secondes en moyenne sur les phrases de longueur 29. Les applications successives du *QCP* après ces méthodes prennent entre 2 et 8 secondes chacune.

### 7.2.3 Efficacité des méthodes de filtrage

La figure 7.3 montre l'efficacité des méthodes de filtrage selon la longueur des phrases. Les courbes représentent le logarithme du nombre moyen d'étiquetages grammaticaux par phrase en fonction de la longueur des phrases, au fur et à mesure de l'application successive des méthodes de filtrage. Chaque courbe correspond au résultat de l'application d'une méthode de filtrage

supplémentaire.

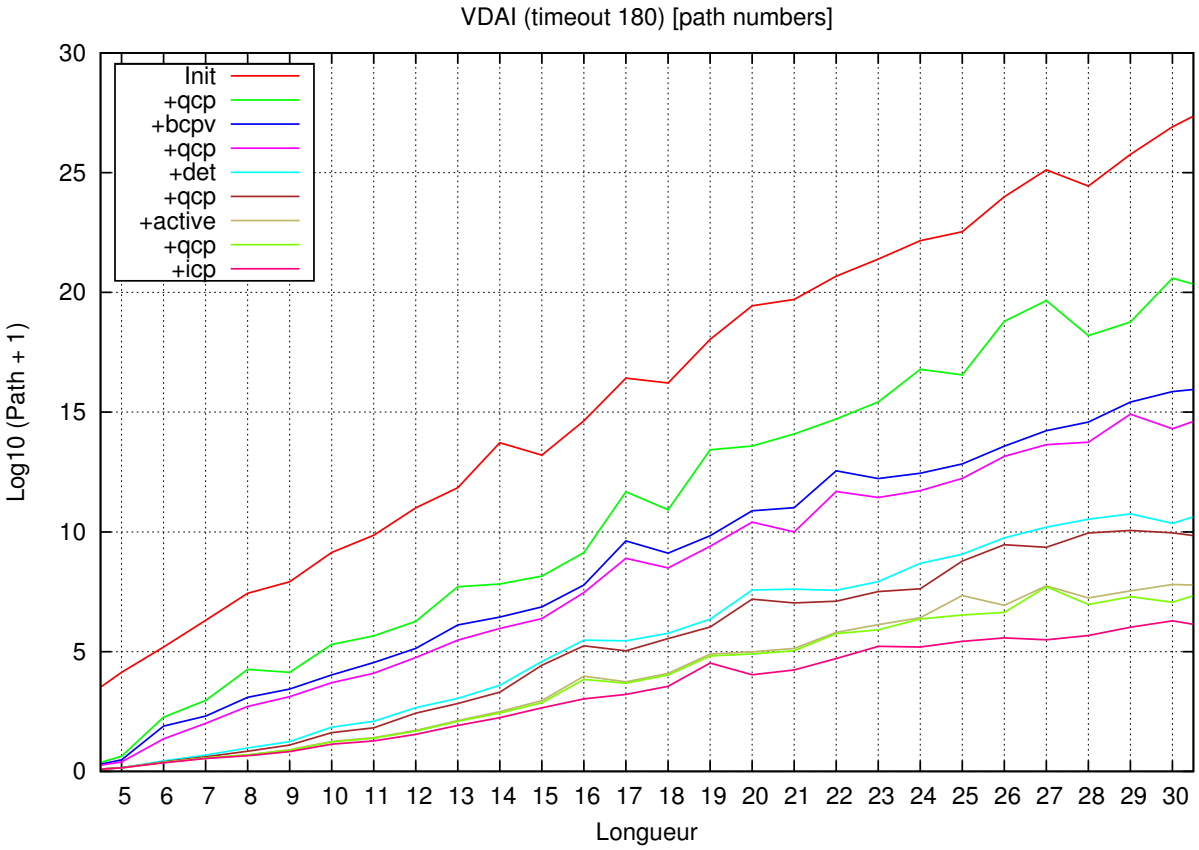



FIGURE 7.3 – Nombre d'étiquetages après application des méthodes de filtrage

Prenons comme exemple les phrases de longueur 27 :

1. la courbe supérieure, en rouge, correspond au nombre initial d'étiquetages grammaticaux ; les phrases de longueur 27 ont initialement en moyenne $10^{25}$ étiquetages grammaticaux possibles ;
2. la deuxième courbe, en vert vif, correspond au résultat de l'application de la méthode *QCP* sur l'automate d'étiquetages initial ; ces phrases ont alors environ $10^{19}$ étiquetages grammaticaux possibles en moyenne ;
3. la troisième courbe, en bleu, correspond au résultat de l'application de la méthode *BCP* pour les seules polarités virtuelles sur l'automate précédent ; il reste alors environ $10^{14}$ étiquetages en moyenne ;
4. la quatrième courbe, en mauve, est produite par l'application de la méthode *QCP* sur l'automate précédent ; il reste un peu moins de $10^{14}$ étiquetages ;
5. la courbe bleue claire est produite par application de la méthode *DET*, toujours sur l'automate précédent, qui correspond au filtrage déterministe par bilan de polarités ; il reste alors $10^{10}$ étiquetages ;

6. la courbe marron est produite par une nouvelle application de la méthode *QCP* ; il reste légèrement plus de $10^9$ étiquetages ;
7. la courbe kaki résulte de l'application de la méthode *ACTIVE*, qui correspond au filtrage par bilan de polarités ; après application de cette méthode, il reste un peu moins de $10^8$ étiquetages ;
8. la courbe vert clair correspond à l'application de *QCP* à l'automate précédent ; il reste $10^7$ étiquetages ;
9. la courbe magenta est obtenue par application de la méthode *ICP* ; il reste en moyenne pour les phrases de longueur 27 entre $10^5$ et $10^6$ étiquetages.

Au final, la combinaison des différentes méthodes de filtrage symbolique pour les formalismes polarisés apporte un gain extrêmement important : le nombre moyen d'étiquetages passe de $10^{25}$ à $10^6$ pour les phrases de longueur 27.

Ces courbes amènent deux commentaires généraux. Premièrement, les trajectoires des courbes sont linéaires. Nous retrouvons ainsi l'augmentation exponentielle du nombre d'étiquetages possibles, suivant l'augmentation de la longueur des phrases. Deuxièmement, chaque application d'une méthode de filtrage supplémentaire produit une courbe dont la pente est plus faible que la courbe précédente. Or, la pente des courbes correspond à l'ambiguïté moyenne par mot : si un mot peut être associé en moyenne à $a$ descriptions élémentaires, alors le nombre d'étiquetages pour une phrase de longueur $n$ est environ $a^n$ et donc $log(a^n) = n \cdot log(a)$. Par conséquent, l'ambiguïté moyenne peut être lue comme $10^s$ où $s$ est la pente des courbes de la figure 7.3.

Une illustration est donnée par la figure 7.4, qui contient l'ambiguïté moyenne par mot au fur et à mesure des applications de méthodes de filtrage. L'ambiguïté moyenne par mot passe de 8.18 initialement à 6.12 après application du *QCP* sur l'automate initial, puis 4.02 après application du *BCPV*, puis diminue régulièrement jusqu'à atteindre 1.83 après application de tous les filtres.

| init | *QCP* | *+BCPV* | *+QCP* | *+DET* | *+QCP* | *+ACTIVE* | *+QCP* | *+ICP* |
|---|---|---|---|---|---|---|---|---|
| 8.18 | 6.12 | 4.02 | 3.80 | 2.91 | 2.76 | 2.20 | 2.12 | 1.83 |

FIGURE 7.4 – Ambiguïté moyenne par mot au fur et à mesure des méthodes appliquées

## 7.3 Apport à l'existant

Les nouvelles méthodes de filtrage que nous avons présentées dans cette partie, *BCP*, *QCP* et *ICP*, sont complémentaires des méthodes de filtrage par bilan de polarités que nous avons présentées au chapitre 4. Cette complémentarité a deux origines.

Premièrement, les méthodes de filtrage fondées sur les compagnons s'appliquent à un plus grand nombre de polarités que les méthodes de filtrage par bilan de polarités. Les méthodes

de filtrage par bilan de polarités réduisent en effet les descriptions syntaxiques élémentaires à l'ensemble de leurs polarités positives et négatives. Elles ignorent donc les autres polarités insaturées, comme les polarités virtuelles ($\sim$) en IG ou les polarités blanches ($\square$) des GUP [Kah06], qui doivent être composées avec une polarité saturée, positive ou négative. Par contraste, les méthodes de filtrage fondées sur les compagnons s'appliquent à toutes les polarités insaturées, y compris les polarités virtuelles et blanches. Intuitivement, les polarités virtuelles et les polarités blanches modélisent un contexte obligatoire. Dans les Grammaires de Dépendances, ces polarités sont notamment utilisées pour représenter la relation entre un modifieur et son gouverneur : le modifieur requiert la présence de son gouverneur mais ne change pas l'état de saturation de ce dernier.

Deuxièmement, les méthodes de filtrage par bilan de polarités oublient les structures qui organisent l'information syntaxique : structures de traits, arbres, connecteurs logiques orientés... Les informations structurelles regroupent les polarités et les orientent les unes par rapport aux autres et par rapport à l'ancre. Elles permettent ainsi de spécifier des informations sur le contexte de la polarité. En IG, ce contexte est le nœud auquel appartient la polarité mais également ses nœuds pères et frères. Ce contexte permet de spécifier le syntagme auquel un mot peut appartenir ou les mots qui doivent le précéder ou le suivre. Il permet également de contrôler la directionnalité des compléments d'un mot et l'ordre de ces compléments. A contrario, les méthodes de filtrage fondées sur les compagnons prennent en compte l'information structurelle.

Les deux points précédents impliquent que les méthodes fondées sur le bilan de polarités sont incapables de filtrer un étiquetage qui ne respecte pas les contraintes de contexte au sens large. Ce n'est pas le cas des méthodes de filtrage fondées sur les compagnons.

Les résultats expérimentaux présentés dans la section précédente montrent que nos méthodes, utilisées en complément des méthodes par bilan de polarités, améliorent grandement l'efficacité de la phase d'étiquetage grammatical.

## 7.4 Limites

Les méthodes de filtrage que nous avons présentées dans cette partie présentent trois grandes limites : le coût, l'inadaptation à l'analyse robuste et l'efficacité.

Tout d'abord, les méthodes *BCP*, *QCP* et surtout *ICP*, comme les méthodes *DET* et *ACTIVE*, sont des méthodes de filtrage coûteuses, qui nécessitent de calculer et de faire l'intersection d'un grand nombre d'automates. Dans les formalismes dans lesquels le problème de l'analyse syntaxique est NP-difficile, comme en IG, le coût du filtrage reste toutefois nettement inférieur à celui de l'analyse.

Ensuite, ces méthodes utilisent toutes des critères de filtrage globaux, à l'échelle de la phrase, et de façon stricte. Ces caractéristiques rendent ces méthodes peu adaptées à l'analyse robuste : elles filtreront en effet la plupart des phrases qui ne sont pas strictement grammaticales au sens de la grammaire utilisée.

Enfin, ces méthodes ne permettent pas de détecter des contraintes tripartites, comme l'ordre relatif de deux dépendants d'un mot. Par exemple, si une description de verbe impose que son complément d'objet direct précède (largement) son complément d'objet indirect, tous deux étant à droite du verbe, le filtrage ne fera pas de distinction entre deux étiquetages, l'un contenant l'ordre licite V OBJ IOBJ et l'autre l'ordre illicite V IOBJ OBJ. En effet, l'ordre relatif des compléments est dans la description du verbe, alors qu'elle concerne les deux compléments. La distinction entre ces deux configurations est laissée au processus d'analyse syntaxique.

## 7.5 Perspectives

Ces travaux ouvrent plusieurs perspectives d'approfondissement et de raffinement.

### 7.5.1 Application à d'autres formalismes

Jusqu'à présent, nous avons testé nos méthodes, avec succès, sur les IG dans l'analyseur LEOPAR. Nos méthodes étant applicables à tout formalisme grammatical lexicalisé polarisé, il faudrait maintenant que nous les expérimentions sur d'autres formalismes et d'autres chaînes de traitement. Un candidat naturel est le système GenI de génération de phrases en LTAG [Kow07], qui utilise déjà le filtrage par bilan de polarités. Les méthodes de filtrage fondées sur le principe du compagnonnage pourraient être également utilisées dans ce système de génération, en particulier car elles donnent des réponses efficaces à deux problèmes identifiés de GenI : la gestion des modificateurs (adjectifs, adverbes...) et les contraintes de directionnalité entre les mots.

### 7.5.2 Affinement des contraintes à la volée

Nous avons vu que les ensembles de compagnons hypothétiques ne pouvaient être calculés sur la grammaire ancrée, à cause de la taille de cette dernière. Nous avons donc utilisé la grammaire non ancrée, qui est une approximation surensembliste de la grammaire ancrée. Il serait intéressant d'affiner à la volée, pour une phrase, les contraintes de la grammaire non ancrée en ajoutant les informations de l'ancrage, comme le genre et le nombre. Ces informations supplémentaires devraient permettre d'augmenter l'efficacité de nos méthodes de filtrage. Dans les méthodes *BCP* et *ICP*, l'efficacité du filtrage et le temps de calcul sont corrélés. Il faudrait donc trouver une méthode intermédiaire entre le *QCP* et le *BCP* ou l'*ICP*, soit par des automates d'approximation, soit en isolant un sous-ensemble d'éléments sur lesquels nous sommes assurés de l'efficacité du *BCP* ou de l'*ICP*.

### 7.5.3 Intégration à l'analyse profonde

Une autre perspective de ce travail concerne l'intégration des méthodes de filtrage fondées sur les compagnons pour guider l'analyse profonde. Au cours de nos expérimentations, nous avons observé que, pour au moins 20% des mots, les polarités non saturées de la description qui leur

est associée ont un unique compagnon hypothétique dans l'étiquetage. Les méthodes de filtrage déterminent donc complètement le comportement de ces mots pendant l'analyse. Les algorithmes d'analyse de l'analyseur LEOPAR ne sont à ce jour pas capables d'utiliser ces informations.

### 7.5.4 Unification et généralisation des méthodes de filtrage

Enfin, le filtrage *ICP* ouvre la voie à un cadre unique qui permet de concilier et de comparer les méthodes de filtrage par invariant de comptage (bilan de polarités) et par principe du compagnonnage.

En effet, il est possible d'utiliser le morphisme d'abstraction de destructuration (noté *destr*) de [BGP04], afin d'oublier toute la structure qui entoure les traits polarisés avant de leur appliquer la méthode *ICP*. Cela revient alors à calculer les automates du filtrage par invariant de comptage, avant que ces automates soient émondés. Il semble donc que, sur certains ensembles de polarités que nous qualifions de « clos », comme le sont tous les traits polarisés de même nom et de même valeur, nous puissions imposer la contrainte supplémentaire du bilan de polarité nul. Dans le langage LXR, cela signifie que seuls les mots dans lesquels tous les X, mais également tous les L et tous les R, sont accrochés, appartiennent au langage.

La situation générale est donnée par le tableau de la figure 7.5, où POL représente les méthodes de filtrage par bilan de polarités.

| | Polarités non saturées | Polarités non saturées et concurrentes | Polarités non saturées, concurrentes et en ensembles clos |
|---|---|---|---|
| non orienté | *BCP* non orienté | *ICP* non orienté | POL généralisé |
| orienté | *BCP* | *ICP* | POL généralisé orienté |

FIGURE 7.5 – Résumé des méthodes de filtrage symbolique pour les formalismes polarisés

### 7.5.5 Approximations par morphismes d'abstraction

Nous venons d'esquisser un lien entre *ICP* et POL, qui repose sur l'utilisation d'un morphisme d'abstraction. De nombreux morphismes d'abstraction peuvent être définis, qui oublient plus ou moins d'information dans les structures syntaxiques. Le morphisme de destructuration de POL est un exemple extrême car toute l'information structurelle est oubliée. Il serait intéressant de définir différents morphismes et de les tester expérimentalement, afin de garder ceux qui présentent le meilleur compromis entre rapidité et efficacité.

### 7.5.6 Caractérisations des langages LXR

Les langages LXR que nous avons introduits et étudiés dans ce chapitre doivent pouvoir être caractérisés de deux façons supplémentaires. La première caractérisation est en terme de langages formels : les langages LXR semblent constituer une sous-classe des *shuffle languages* [Gis81, Jed99,

BBH11]. Nous reprenons ici l'intuition qui en est donnée par [BBH11]. Les shuffle languages sont les langages générés par des shuffle expressions, qui sont des expressions rationnelles augmentées par deux opérateurs : le shuffle (en français brassage, mélange, battage (de cartes)) et la clôture du shuffle. L'opérateur de shuffle, noté $\odot$, est défini au départ sur deux chaînes $u$ et $v$ et retourne l'ensemble de tous les entrelacements possibles des symboles de $u$ et $v$. Par exemple, si $u = ab$ et $v = cd$, alors $u \odot v = ab \odot cd = \{abcd, acbd, acdb, cabd, cadb, cdab\}$. La définition de cet opérateur est étendue aux langages réguliers, le résultat du shuffle de deux langages réguliers $\mathcal{L}_1$ et $\mathcal{L}_2$ étant l'ensemble des mots générés par entrelacement entre un mot de $\mathcal{L}_1$ et un mot de $\mathcal{L}_2$ : $\mathcal{L}_1 \odot \mathcal{L}_2 = \bigcup\{u \odot v | u \in \mathcal{L}_1, v \in \mathcal{L}_2\}$.

La deuxième caractérisation est en terme de logique : le type de non-commutativité à l'œuvre dans les langages LXR appelle un lien avec certaines logiques non-commutatives, comme la logique NL [AR00].

## Conclusion

Nous avons évalué et comparé dans ce chapitre les méthodes de filtrage des étiquetages grammaticaux pour les formalismes polarisés. La combinaison des méthodes existantes et des trois méthodes que nous avons introduites dans cette partie donne de très bons résultats expérimentaux. Si ces méthodes ont un coût en temps non négligeable, elles réduisent fortement l'ambiguïté moyenne par mot, sans éliminer aucun étiquetage correct. Ce dernier point est garanti par le fait que nos méthodes de filtrage reposent sur des contraintes statiques de la grammaire et non sur un modèle statistique. Ces méthodes peuvent encore être améliorées à la fois en temps et en efficacité et leur formalisation ouvre des perspectives théoriques intéressantes.

Ce chapitre clôt la deuxième partie de ce manuscrit qui portait sur la phase d'étiquetage grammatical dans l'analyse syntaxique des formalismes lexicalisés polarisés. Dans la troisième partie, nous abordons la question du calcul d'une représentation sémantique de la phrase à partir d'une analyse syntaxique produite par un formalisme grammatical lexicalisé. Ces deux parties abordent des problèmes différents et emploient pour les résoudre des méthodes différentes. Toutes deux concourent cependant à la création d'une chaîne de traitement linguistique autour d'un analyseur syntaxique pour un formalisme grammatical lexicalisé polarisé, comme LEOPAR pour les IG.

# Troisième partie

# Interface syntaxe-sémantique pour les formalismes lexicalisés polarisés

# Chapitre 8

# Interface syntaxe-sémantique générique pour les formalismes lexicalisés

Nous ouvrons maintenant la troisième partie de ce travail, qui porte sur une autre étape importante des chaînes de traitement linguistique : l'analyse sémantique de la phrase. Concrètement, nous nous intéressons à la façon de calculer une représentation sémantique pour une phrase à partir de son analyse syntaxique. Dans ce chapitre, nous déterminons notre approche de cette question. Nous présentons le type de structures sémantiques que nous voulons produire, la perspective que nous adoptons sur le lien entre analyse syntaxique et analyse sémantique et le format des structures syntaxiques sur lesquelles nous nous appuyons.

Dans la section 8.1, nous définissons notre objectif, qui est la construction de représentations sémantiques sous-spécifiées. Nous examinons ensuite, dans la section 8.2, les trois types d'architectures — intégrées, parallèles et séquentielles — qui sont couramment employées pour relier l'analyse syntaxique et l'analyse sémantique. Nous nous plaçons finalement dans une perspective séquentielle, ce qui requiert de déterminer le format des structures syntaxiques à partir desquelles calculer la sémantique. Dans la section 8.3, nous dressons la liste des informations syntaxiques nécessaires au calcul de la sémantique. Nous examinons ensuite la façon dont ces informations sont représentées dans les structures syntaxiques produites par les analyseurs d'une part et dans les structures syntaxiques utilisées pour annoter les corpus d'autre part. Cet état des lieux nous amène à définir dans la section 8.4 une extension du format d'annotation en dépendances du *French Treebank*. C'est à partir de structures syntaxiques dans ce format étendu que, dans le prochain chapitre, nous montrons comment calculer des représentations sémantiques.

## 8.1 Représentation sémantique visée

### 8.1.1 Niveau de détail de la représentation sémantique

On distingue actuellement deux types d'analyses sémantiques en TAL qui répondent à des besoins applicatifs différents et correspondent à deux niveaux de détail. L'**analyse sémantique superficielle** consiste à repérer et identifier les arguments des prédicats d'une phrase, selon la formule « qui a fait quoi à qui, où, quand et comment ? ». Cette tâche est couramment appelée « étiquetage par des rôles sémantiques », ou SRL pour *Semantic Role Labeling* [MCLS08]. L'enjeu est de déterminer quels syntagmes, dans une phrase, contiennent les arguments sémantiques d'un prédicat. Par exemple, la phrase *« Tout homme gentil aime une femme »* est analysée comme suit :

[$_{Arg0}$ Tout homme gentil] *aime* [$_{Arg1}$ une femme].

*« Tout homme »* est identifié comme l'argument 0 du verbe *« aime »* et *« une femme »* comme son argument 1, en utilisant le jeu d'étiquettes du corpus annoté par des rôles sémantiques PropBank [PGK05]. Les arguments 0 et 1 sont attribués aux arguments syntaxiques qui correspondent respectivement à un agent et à un patient ou thème prototypiques. Les éventuels arguments supplémentaires du prédicat sont numérotés par 2 et plus. Aucune généralisation sémantique ne peut par contre être faite sur les étiquettes PropBank ; leur interprétation est particulière à chaque verbe. Cette analyse sémantique est qualifiée de superficielle car elle ne porte pas sur le sens détaillé des syntagmes, ignorant par exemple la contribution sémantique des adjectifs épithètes ou des déterminants.

Nous nous intéressons dans la suite de ce manuscrit au problème de l'**analyse sémantique profonde**, qui associe à une phrase une ou plusieurs structures qui représentent le sens de la phrase, à partir des contributions sémantiques de chacun des mots qui la composent. En TAL, l'analyse sémantique profonde s'inscrit la plupart du temps dans un le cadre particulier de la **sémantique vériconditionnelle compositionelle**. La *sémantique vériconditionnelle* caractérise formellement les phrases du point de vue de leurs conditions de vérité logique. Les représentations sémantiques que nous cherchons à construire sont donc des formules logiques qui représentent le ou les sens possibles d'une phrase. L'exemple *« Tout homme gentil aime une femme »* peut décrire deux situations du monde différentes, qui sont décrites par les deux formules logiques non équivalentes de la figure 8.1.

$$\forall x.homme(x) \wedge gentil(x) \rightarrow (\exists y.femme(y) \wedge aime(x,y))$$

$$\exists y.femme(y) \wedge (\forall x.homme(x) \wedge gentil(x) \rightarrow aime(x,y))$$

FIGURE 8.1 – Formules logiques pour les deux lectures de *« Tout homme gentil aime une femme »*

La *sémantique compositionelle* calcule le sens d'une phrase à partir du sens de ses parties, suivant le principe de compositionalité : « le sens d'une expression complexe est une fonction du sens de ses parties et de leur mode de combinaison ». Depuis les travaux de Montague [Mon70, Mon74],

l'approche classique en sémantique vériconditionnelle compositionelle consiste à calculer la représentation sémantique d'une phrase via un morphisme de l'analyse syntaxique vers l'analyse sémantique. L'analyse syntaxique fournit en effet le mode de combinaison des parties d'une expression composée. Dans notre exemple, l'analyse syntaxique permet (notamment) de lier les arguments du prédicat *aime* : le premier argument est la variable $x$ introduite par le sujet *« tout homme gentil »*, le deuxième argument est la variable $y$ introduite par l'objet *« une femme »*.

### 8.1.2 Structures de représentation sémantique

#### Formules logiques avec quantificateurs généralisés

Les structures que nous utilisons pour représenter la sémantique sont des formules de logique des prédicats étendue par des quantificateurs généralisés [KW11]. Les quantificateurs généralisés permettent d'augmenter l'expressivité de la logique des prédicats, ce qui est utile pour les expressions de quantification rencontrées en langue naturelle comme « la plupart », qui ne peuvent pas être exprimées en logique des prédicats. Dans la théorie des quantificateurs généralisés, un quantificateur dénote une relation entre deux ensembles, appelés *restriction* et *corps* de la portée de ce quantificateur. Ainsi, le quantificateur « tout » correspond à la relation d'inclusion de l'ensemble dénoté par la restriction dans l'ensemble dénoté par le corps de sa portée ; le quantificateur « une » correspond à la relation d'intersection non vide entre l'ensemble dénoté par la restriction et celui dénoté par le corps de sa portée. Dans la première formule logique de notre exemple,

$$\forall x.homme(x) \wedge gentil(x) \rightarrow (\exists y.femme(y) \wedge aime(x,y))$$

- $\forall x$ a, dans la restriction de sa portée $homme(x) \wedge gentil(x)$ et dans le corps de sa portée $\exists y.(femme(y) \wedge aime(x,y))$ ;
- $\exists y$ a, dans la restriction de sa portée $femme(y)$ et dans le corps de sa portée $aime(x,y)$.

#### Formules logiques sous-spécifiées

Dans les travaux de Montague [Mon70, Mon74], l'interface syntaxe - sémantique, que nous notons ISS, est une *fonction* des structures syntaxiques vers les formules logiques : à une analyse syntaxique de phrase est associée exactement une formule logique. Cette approche pose problème pour les phrases qui sont à la fois sémantiquement ambiguës et syntaxiquement dénuées d'ambiguïté, comme *« Tout homme gentil aime une femme »*. Pour produire les deux formules logiques qui correspondent aux deux lectures de cette phrase, il faut en effet produire deux analyses syntaxiques distinctes. Cela revient à créer une ambiguïté syntaxique fallacieuse pour gérer une ambiguïté purement sémantique.

Pour éviter ce problème, une solution possible consiste à voir l'interface syntaxe - sémantique comme une *relation* entre des structures syntaxiques et des formules logiques. Une telle relation peut être réalisée de plusieurs façons. Formellement, cette relation peut être réalisée

par la composition de deux ACG qui partagent un langage abstrait [Pog07b], par le bimorphisme sous-jacent à une grammaire synchrone [Shi06] ou par l'énumération des démonstrations d'un séquent en logique linéaire [Dal99]. En pratique, la solution la plus répandue est d'associer à une analyse syntaxique une structure qui représente un ensemble de formules logiques. Ces structures sont appelées des *représentations sémantiques sous-spécifiées*, notées USR pour *Underspecified Semantic Representations* [BB03]. Au chapitre 2, nous avons utilisé des descriptions d'arbres pour décrire, à l'aide d'une seule structure, un ensemble d'arbres. De la même façon, une représentation sémantique sous-spécifiée représente un ensemble de structures sémantiques qui correspondent à des formules logiques. Les structures sémantiques représentées par une USR sont les modèles de cette dernière. Les trois formalismes de sémantique sous-spécifiée les plus répandus sont la sémantique à trous, notée HS pour *Hole Semantics* [Bos01], le langage de contraintes pour les lambda structures, notée CLLS pour Constraint Language for Lambda Structures [EKN01] et la sémantique à récursion minimale, notée MRS pour Minimal Recursion Semantics [CFPS05]. L'expressivité de ces trois formalismes est très similaire ; aucun d'entre eux n'est expressivement complet, c'est-à-dire capable de représenter n'importe quel sous-ensemble de lectures possibles d'une phrase [Ebe05]. Néanmoins, les USR sont utilisées avec succès autant pour des travaux en sémantique formelle que pour des tâches autour de l'inférence et du raisonnement [BB05, Bun07, Egg10].

Le principe qui sous-tend les représentations sémantiques sous-spécifiées est simple. La composition de deux structures sémantiques se fait en remplissant un trou de l'une des deux structures (la tête sémantique) par le crochet de l'autre structure (l'argument). Le remplissage du trou par le crochet se traduit par l'identification de la variable du trou avec celle du crochet. Afin de construire des représentations sémantiques sous-spécifiées, il faut ajouter la possibilité de composer deux structures sémantiques sans remplir directement le trou de l'une par le crochet de l'autre. Les structures correspondant aux quantificateurs, aux négations et aux modaux (entre autres) contiennent alors des trous qui ne peuvent pas être remplis directement. Leur portée a ainsi la liberté de « flotter ». Le flottement de cette structure par rapport à celle avec laquelle elle est composée est contraint par une relation. La contrainte peut être résolue de deux façons : soit le trou de la structure flottante est finalement rempli par le crochet de l'autre structure, soit les deux structures sont finalement reliées par une chaîne de structures telles que chacune de ces structures a un trou qui est rempli par le crochet de la suivante. La figure 8.2 correspond à la structure MRS simplifiée qui représente les deux lectures de *« Tout homme gentil aime une femme »*. Une structure sémantique sous-spécifiée peut en effet être représentée sous la forme d'une description d'arbre, dont les modèles sont des arbres équivalents à des formules logiques. La conjonction entre deux prédicats est implicite ; lorsque deux prédicats sont obligatoirement conjoints, ils sont notés côte à côte, simplement séparés par une virgule. Le flottement est indiqué par les traits pointillés. Les quantificateurs généralisés sont représentés par des prédicats ternaires dont les arguments sont respectivement la variable liée, la restriction de portée et le corps de portée. Les adjectifs, verbes et adverbes ont une variable d'événement afin de pouvoir

être modifiées, comme en sémantique Davidsonienne.

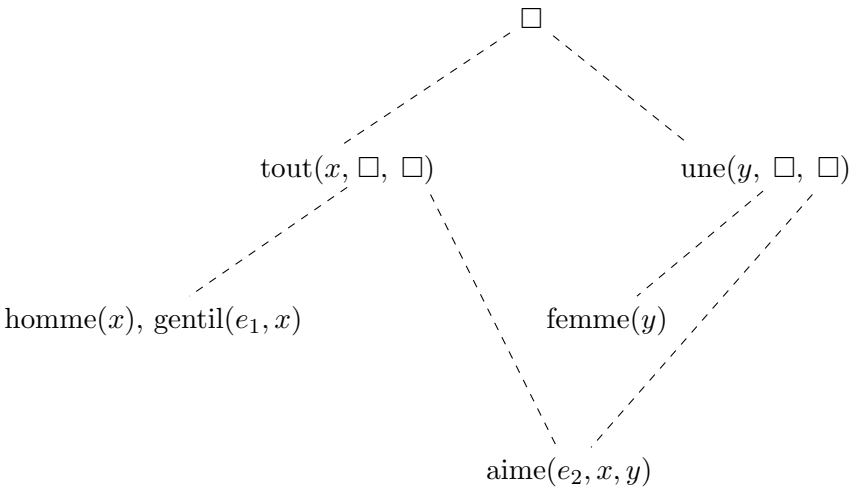



FIGURE 8.2 – Structure MRS pour *« Tout homme gentil aime une femme »*

## 8.2 Nature du lien entre syntaxe et sémantique

Dans cette section, nous passons en revue les trois principales façons de relier l'analyse syntaxique et l'analyse sémantique, qui sont : (i) l'**intégration** des processus d'analyse syntaxique et sémantique, qui est caractérisée par la présence, au sein d'une même structure, d'objets syntaxiques et sémantiques, (ii) la **parallélisation** des deux processus d'analyse, qui est caractérisée par la séparation des structures syntaxiques et sémantiques et par la synchronisation entre les opérations de composition syntaxique et sémantique, (iii) la **séquentialisation** des deux processus, analyse syntaxique puis sémantique, qui est caractérisée par l'absence de synchronisation explicite entre les processus de composition syntaxique et sémantique. Nous évaluons finalement ces trois options au regard du contexte dans lequel s'inscrit notre travail.

### 8.2.1 Architectures intégrées

La première possibilité consiste à intégrer les deux processus d'analyse syntaxique et sémantique. Les exemples les plus typiques de cette approche sont fournis par les grammaires du formalisme HPSG [PS94], comme la grammaire ERG [CF00] de l'anglais. Chaque structure de la grammaire contient des objets syntaxiques et sémantiques. Un unique processus d'analyse compose ces structures et construit ainsi à la fois l'analyse syntaxique et l'analyse sémantique de la phrase.

Une version légèrement plus faible de cette approche est utilisée dans la grammaire FTAG SEMFRAG [GK03, Par07] du français. À chaque structure syntaxique de la grammaire est associée une structure sémantique. La synchronisation entre la composition syntaxique et la composition sémantique est réalisée par l'ajout, dans les structures syntaxiques, de variables partagées avec

les structures sémantiques. La composition de deux structures syntaxiques a pour effet d'unifier deux variables partagées, une dans chaque structure, ce qui simule une composition des structures sémantiques associées. Dans les formalismes génératifs-énumératifs, cette possibilité correspond au calcul de la sémantique à l'aide de la structure dérivée.

#### Problèmes des architectures intégrées

Les architectures intégrées posent trois problèmes. Premièrement, ajouter des objets sémantiques ou des variables partagées aux structures syntaxiques peut avoir pour effet négatif d'augmenter l'expressivité du formalisme et donc la complexité de l'analyse syntaxique [KR04]. Deuxièmement, cette solution requiert d'ajouter des structures ou des informations sémantiques dans toutes les structures syntaxiques de la grammaire. Dans une grammaire de taille réelle, cette tâche est coûteuse et complexe [Par07]. Troisièmement, cette solution rend très difficile la comparaison et la réutilisation de propositions d'ISS d'un formalisme à un autre et d'une grammaire à une autre. Les formalismes existants utilisent en effet des modèles de calcul différents aussi bien pour l'analyse syntaxique que pour l'analyse sémantique : unification, réécriture d'arbres, $\lambda$-calcul... Au sein d'un même formalisme, les grammaires varient selon leurs présupposés linguistiques. On retrouve ici les problèmes posés par la comparaison d'analyses syntaxiques produites par des grammaires différentes dans des formalismes différents [CMB03, Kal06].

### 8.2.2 Architectures parallèles

La deuxième option consiste à construire en parallèle les structures syntaxique et sémantique et à synchroniser les deux processus d'analyse. À chaque structure syntaxique de la grammaire est associée une structure sémantique. L'analyse syntaxique et l'analyse sémantique se déroulent en parallèle, la synchronisation entre les deux processus d'analyse se faisant en associant à chaque opération (ou règle) de composition syntaxique une opération (ou règle) de composition sémantique. C'est l'*hypothèse règle à règle* [HK98].

La première proposition d'ISS de ce type est la Grammaire de Montague [Mon70] : la syntaxe et la sémantique sont des algèbres et l'ISS est un homomorphisme de l'algèbre syntaxique vers l'algèbre sémantique. Ainsi, à chaque dérivation syntaxique correspond une dérivation sémantique. Cet homomorphisme réalise de façon rigoureuse le principe de compositionalité. À la suite des travaux de Montague, les CG, CCG et TLG reprennent la vision de l'ISS comme un homomorphisme entre syntaxe et sémantique [Mor94, SB09].

L'homomorphisme entre syntaxe et sémantique est cependant souvent considéré comme trop rigide. Une réponse possible est apportée par le cadre formel des Grammaires Catégorielles Abstraites (ACG) [dG01], dans lequel il est possible de coder les CG mais également d'autres formalismes linguistiques [dGP04]. Les ACG permettent en effet de simuler des architectures parallèles et d'autres architectures plus complexes comprenant des niveaux profonds [Pog07b, Pog07a, PP10]. Les Grammaires Convergentes (CVG) [Pol11] sont un exemple

de formalisme récent, codable en ACG, qui postule des processus d'analyse syntaxique et sémantique en parallèle. La synchronisation entre les deux processus est assurée par un calcul d'interface explicitement dédié à l'ISS [dGPP11].

En LTAG, la plupart des propositions d'ISS adoptent une architecture parallèle en utilisant l'arbre de dérivation syntaxique pour faire le lien entre syntaxe et sémantique. Le point de départ de ces travaux est le constat que la structure de dérivation associée à une analyse LTAG est très proche d'une structure de dépendance syntaxique profonde [RJ97, RVSW95], voire d'un graphe sémantique de relations prédicat - argument [CK98]. De nombreuses propositions d'ISS pour les LTAG et les Grammaires d'Arbres Adjoints Ensemblistes (MC-TAG) ont donc été faites, qui utilisent l'arbre de dérivation [JVS99, KJ03, KR04, KR08, Pog04] ou un arbre de dérivation enrichi afin de récupérer certaines informations manquantes pour les dépendances à longue distance [Kal02].

L'arbre de dérivation est également au cœur de l'architecture des Grammaires d'Arbres Adjoints synchrones (S-TAG), qui peut être utilisée pour l'interface syntaxe - sémantique [SS90, SS94, Shi06, NS06]. Afin de gérer les ambiguïtés de portée, les S-TAG utilisent une version modifiée des TAG, qui permet des adjonctions multiples au même nœud. La conséquence de cette modification est qu'un arbre de dérivation est alors associé à plusieurs structures sémantiques.

L'architecture de correspondance des Grammaires Lexicales Fonctionnelles (LFG) [KB95] constitue une variante des architectures parallèles présentées jusqu'ici. Cette architecture préfère à l'hypothèse règle à règle, un système de fonctions de correspondance qui contraignent les analyses qui se déroulent parallèlement au sein des différents niveaux linguistiques. L'analyse syntaxique se déroule sur deux niveaux et génère deux structures syntaxiques : une structure de constituants (*c-structure*) et une structure de fonctions grammaticales (*f-structure*). C'est à partir de cette deuxième structure, plus abstraite que la première, qu'est définie la correspondance avec la sémantique de la phrase [DLPS96]. Chaque élément de la grammaire comprend donc une règle de dérivation de la structure en constituants, une spécification partielle de la structure de fonctions grammaticales et une spécification partielle de la structure sémantique. Le lien entre ces niveaux est assuré par deux fonctions de correspondance, la première allant des objets de la c-structure vers les objets de la f-structure, la deuxième allant des objets de la f-structure vers les objets de la structure sémantique. Les processus de construction de la c-structure, de la f-structure et de la structure sémantique sont distincts, mais les descriptions que ces processus manipulent sont contraintes par les fonctions de correspondance.

Les Grammaires d'Unification Sens-Texte (GUST) [Kah02] sont un formalisme grammatical qui produit des structures de dépendance aux niveaux syntaxique et sémantique. Les GUST adoptent un modèle articulé, qui implique que les différents niveaux linguistiques peuvent être vus comme co-existant (architecture parallèle) ou se succédant (architecture séquentielle). Des règles de correspondance font le lien entre les structures des différents niveaux. Ces règles mettent en relation deux fragments de structures appartenant à deux niveaux de représentation adjacents. Si ces règles sont lexicalisées, la grammaire entre parfaitement dans une architecture parallèle.

Les Grammaires de Dépendances eXtensibles (XDG) [Deb06] sont un autre exemple de formalisme qui produit des structures de dépendance à l'aide d'une architecture parallèle. Ce formalisme repose sur l'utilisation d'un langage de contraintes, qui permet à la fois la composition des structures d'un même niveau et la communication entre les différents niveaux.

#### Avantages des architectures parallèles

Par rapport aux architectures intégrées, les architectures parallèles présentent deux avantages. Premièrement, les différents niveaux linguistiques sont bien distincts. Ainsi, les formalismes utilisés pour l'analyse syntaxique et sémantique restent identiques, au sein d'une architecture parallèle, à ce qu'ils sont en isolation. En particulier, l'expressivité de chacun de ces formalismes et la complexité de leur analyse ne changent pas. De plus, la séparation des niveaux rend les différentes parties de la grammaire relativement indépendantes du point de vue de leur conception et de leur maintenance. Il est ainsi possible de modifier les structures sémantiques d'une grammaire sans modifier les structures syntaxiques ; seul leur lien doit être mis à jour. Il est également plus facile de proposer plusieurs modélisations sémantiques alternatives pour une même modélisation syntaxique.

Deuxièmement, établir un lien explicite entre la syntaxe et la sémantique permet de s'abstraire, dans l'ISS, de certains détails de la syntaxe. Il est par exemple courant, dans une grammaire, de distinguer plusieurs projections successives d'une même ancre. Dans la grammaire TAG de l'anglais XTAG [XTA95], un verbe a trois projections : le verbe lui-même $V$, le syntagme verbal $VP$ ou le noyau verbal $VN$ et la proposition $S$. Lors de la spécification du lien entre syntaxe et sémantique, il est possible d'attribuer à ces trois projections syntaxiques la même contribution sémantique. De fait, sémantiquement, la contribution de ces trois projections est une même variable d'événement et tous les modifieurs du verbe, quelle que soit la projection à laquelle ils sont rattachés, portent sémantiquement sur cette variable d'événement. L'abstraction ainsi obtenue par rapport à la syntaxe améliore grandement la lisibilité de l'ISS, ce qui facilite d'une part la conception et la maintenance d'une ISS dans une grammaire, d'autre part la comparaison entre différentes propositions d'ISS.

#### Problèmes des architectures parallèles

Les architectures parallèles sont très intéressantes sur le plan conceptuel et apportent certaines garanties, de complexité et de modularité, sur le plan pratique. Cependant, comme dans les architectures intégrées, il est nécessaire d'associer à chaque structure syntaxique de la grammaire une structure sémantique, ce qui reste coûteux.

Ensuite, dans de nombreux formalismes, les structures utilisées pour faire le lien entre la syntaxe et la sémantique sont idiosyncratiques au formalisme utilisé. Cette idiosyncrasie est un obstacle à la comparaison et à la réutilisation des propositions d'ISS entre formalismes. Un exemple est fourni par les CCG et les LTAG. Ces formalismes sont faiblement équivalents, c'est-à-dire qu'ils sont capables de dériver les mêmes langages de chaînes, mais ils ne sont pas fortement

équivalents, car ils ne sont pas capables de construire les mêmes structures de dérivation [HY08, KK09]. Or, dans ces deux formalismes, la solution la plus répandue consiste à utiliser l'arbre de dérivation syntaxique pour faire le lien entre la syntaxe et la sémantique. Si, pour analyser une même phrase, les deux formalismes ne sont pas capables de produire le même arbre de dérivation syntaxique, cela signifie que leurs grammaires doivent être sensiblement différentes, au niveau syntaxique comme au niveau sémantique. Par conséquent, il n'est pas possible d'utiliser ou de comparer directement des propositions d'ISS dans ces deux formalismes.

Enfin, le type de lien choisi dans certains formalismes pour assurer la synchronisation entre l'analyse syntaxique et l'analyse sémantique est trop limité pour transmettre l'information syntaxique nécessaire au calcul de la sémantique. C'est le cas des formalismes qui, comme les CCG et les LTAG, utilisent l'arbre de dérivation syntaxique. Ce dernier ne garde pas trace de certaines informations syntaxiques qui sont contenues dans la structure dérivée et qui sont nécessaires à l'ISS. Par exemple, les interactions entre verbes à montée et verbes d'attitude ainsi que les dépendances à longue distance provoquées par l'extraction des syntagmes en *qu-*, sont modélisées dans l'arbre dérivé LTAG, mais ne laissent pas de trace dans l'arbre de dérivation [KR04]. Gérer ces phénomènes nécessite de compliquer l'ISS, par exemple en enrichissant [Kal02] ou en décorant [KR08] l'arbre de dérivation. De tels contournements techniques rendent ces propositions moins générales et lisibles et renforcent leur caractère idiosyncratique.

### 8.2.3 Architectures séquentielles

La troisième option consiste à adopter une architecture séquentielle, dans laquelle l'analyse sémantique est calculée à partir du résultat de l'analyse syntaxique, sans faire référence au déroulement de cette dernière. On retrouve ce type d'architecture dans les premières théories de Chomsky [Cho65] comprenant plusieurs niveaux (syntaxe profonde, syntaxe de surface, phonologie, forme logique...) reliés par des systèmes de réécriture réalisant des transformations. Les nombreux débats qui ont entouré les architectures formelles proposées successivement par Chomsky ont eu pour effet de marginaliser les architectures séquentielles sur un plan théorique.

Les architectures séquentielles continuent cependant d'être utilisées pour des raisons pratiques dans des chaînes de traitement linguistique. C'est le cas par exemple dans l'analyseur XLE [MK96] pour les grammaires LFG [CK06]. L'architecture de correspondance est abandonnée pour l'ISS au profit d'une architecture séquentielle. La représentation sémantique de la phrase est calculée à partir de la structure de fonctions grammaticales (la *f-structure*) de la phrase, par application d'un système de réécriture de termes. Les règles de réécriture s'appliquent à la f-structure finale, sans faire référence à la grammaire qui a produit cette f-structure. Cette solution rend l'ISS totalement indépendante de l'analyse syntaxique et même de la grammaire. En effet, il n'est alors pas nécessaire d'associer une structure sémantique à chaque structure syntaxique de la grammaire. La même ISS peut ainsi être utilisée avec plusieurs grammaires LFG qui produisent des c-structures différentes mais des f-structures identiques.

#### Avantages des architectures séquentielles

Calculer la sémantique à partir du résultat de l'analyse syntaxique présente trois avantages concrets. Premièrement, cela permet de développer efficacement et rapidement une ISS, car il n'est pas nécessaire d'associer une structure sémantique à chaque structure syntaxique de la grammaire. Deuxièmement, si la structure syntaxique utilisée est suffisamment abstraite et neutre, l'ISS est alors indépendante de la grammaire mais également du formalisme utilisés pour l'analyse syntaxique. Une même ISS peut donc être utilisée dans plusieurs analyseurs ou avec plusieurs grammaires. Troisièmement, une telle ISS peut être appliquée à un corpus annoté syntaxiquement, afin de lui ajouter des annotations sémantiques. Cette possibilité s'avère particulièrement utile pour créer des corpus de référence servant à l'évaluation d'analyseurs ou de grammaires [SF05].

#### Inconvénients des architectures séquentielles

De nombreuses études concluent que les architectures séquentielles sont cognitivement peu réalistes [Jac07]. En particulier, l'approche séquentielle présente l'inconvénient d'empêcher ou du moins de retarder les réactions de la sémantique sur la syntaxe. Les architectures séquentielles sont donc, sur un plan purement théorique, d'un intérêt linguistique relativement faible.

### 8.2.4 Bilan

La plupart des modèles d'interface syntaxe - sémantique adoptent une approche parallèle : la représentation sémantique d'une phrase est construite pas à pas parallèlement à sa structure syntaxique. Selon le formalisme syntaxique choisi, cette approche est implantée de différentes façons. Dans une approche parallèle, l'interface syntaxe - sémantique est étroitement liée à la grammaire et au formalisme grammatical. La construction d'une telle interface est donc très coûteuse, surtout pour une grammaire à grande couverture.

Certains formalismes adoptent une approche intégrée. Un seul processus d'analyse construit conjointement les structures syntaxique et sémantique. Cette approche est minoritaire en raison notamment du coût de la construction de l'ISS et du manque de lisibilité de l'ISS.

Dans la suite de ce manuscrit, nous adoptons une approche séquentielle : nous utilisons une analyse syntaxique de phrase déjà effectuée pour produire une représentation sémantique. Par rapport à l'approche parallèle, l'approche séquentielle est psycholinguistiquement peu plausible. D'un autre côté, le calcul de la sémantique est rendu relativement indépendant du formalisme et de la grammaire employés pour l'analyse syntaxique. La seule contrainte est la forme de la structure produite par l'analyse syntaxique.

Deux facteurs particuliers expliquent notre choix. Le premier facteur est global : il n'existe pas à ce jour de corpus de référence annoté sémantiquement pour le français. L'absence de corpus de référence rend impossible l'évaluation d'une ISS pour une grammaire du français à grande couverture. L'incertitude qui en découle rend d'autant plus importante la question du

coût de l'écriture d'une ISS. Dans ce contexte, il paraît plus sûr de privilégier la solution la moins coûteuse, c'est-à-dire d'adopter une approche séquentielle de l'ISS, ce qui permet de ne pas annoter sémantiquement chaque structure de la grammaire.

Le deuxième facteur tient au contexte dans lequel s'inscrivent nos travaux, qui est le développement d'une chaîne de traitement linguistique autour de l'analyseur LEOPAR des IG, avec la grammaire du français FRIGRAM. La taille conséquente de la grammaire FRIGRAM, de l'ordre de 3 000 à 4 000 structures (3 788 actuellement), rend le développement d'une ISS très coûteux. De plus, contrairement à la grammaire XTAG [XTA95] qui a été figée et documentée, la grammaire FRIGRAM, plus récente, est encore en développement constant et les principes généraux de modélisation linguistique qui la sous-tendent ne sont pas encore totalement fixés. Là aussi, l'incertitude induite renchérit le coût de la création et de la maintenance de l'ISS, ce qui favorise le choix d'une approche séquentielle de l'ISS par rapport aux approches intégrée et parallèle.

## 8.3 Éléments pour le choix du format des structures syntaxiques

Maintenant que nous avons adopté une perspective séquentielle sur la relation entre analyse syntaxique et analyse sémantique, il nous faut déterminer le type des structures syntaxiques à partir desquelles calculer des représentations sémantiques. Nous abordons cette question en quatre temps. La première étape consiste à dresser la liste des informations syntaxiques utiles au calcul d'une représentation sémantique. La deuxième étape consiste à examiner la façon dont cette information est représentée dans les structures d'analyse syntaxique que les différents formalismes produisent. La troisième étape consiste à examiner les structures syntaxiques utilisées pour comparer les structures produites par les analyseurs. Enfin, la quatrième étape consiste à examiner les structures syntaxiques qui sont utilisées pour l'annotation de corpus.

### 8.3.1 Information syntaxique nécessaire au calcul de la sémantique

Les représentations sémantiques que nous voulons construire sont constituées de relations prédicat - arguments et d'informations sur le mode de composition sémantique entre les prédicats. Les informations syntaxiques nécessaires à la détermination du mode de composition sémantique sont fournies par tout format de structure syntaxique exhaustif. La plupart des relations prédicat - arguments coïncident au niveau syntaxique avec les relations entre une tête et ses compléments et entre une tête et ses modificateurs. Dans la terminologie des grammaires de dépendance, les relations prédicat - arguments coïncident avec les relations de dépendance syntaxique.

Parmi ces relations syntaxiques, les plus difficiles à modéliser pour les grammaires et à capturer pour les analyseurs sont les *constructions de dépendances non bornées*, également appelées *dépendances à longue distance* [Cho77, LH06]. Les dépendances à longue distance sont à l'œuvre notamment dans les questions directes (*« Qui Jean a-t-il **vu** ? »*), les questions indirectes (*« Je me demande qui Jean a **vu** »*) et les subordonnées relatives (*« L'homme que Jean a **vu** »*, *« Le*

*film dont Jean connaît la **fin** »*) et les factorisations à droite ou *right node raising* (*« Jean ne **peut** ni ne **veut** venir demain »*). Les mots en gras sont les gouverneurs et les éléments soulignés les dépendants. À ces constructions, il faut ajouter les constructions dites de *tough movement* (*« Ce livre est facile à **lire** »*) qui sont traditionnellement considérées dans la littérature comme induisant des dépendances à longue distance.

Des relations prédicat - arguments supplémentaires sont déterminées par les relations de *contrôle grammatical* [KB95]. Ces relations sont créées par les verbes à contrôle (*« Marie autorise Jean à **partir** »*), les verbes à montée (*« Jean semble **manger** »*) et les gérondifs (*« En **mangeant**, Jean fit un signe de la main »*).

Un élément important de la construction d'une représentation sémantique concerne l'identification, pour chaque pronom présent dans la phrase, de son antécédent. La contribution sémantique du pronom reprend en effet celle de son antécédent. Dans sa globalité, ce problème est considéré comme très difficile [Mit02]. L'analyse syntaxique permet de résoudre les cas les plus simples. Les pronoms concernés sont les pronoms clitiques répétitions de sujet comme dans *« Jean viendra-t-**il** ? »*, les pronoms réfléchis comme dans *« Jean **se** lave »* et les pronoms relatifs comme dans *« Jean **que** Marie connaît »*.

### 8.3.2 Structures syntaxiques produites par les formalismes grammaticaux

Dans la sous-section précédente, nous avons dressé la liste des informations syntaxiques qui sont nécessaires au calcul d'une représentation sémantique de la phrase. Dans les structures produites par les formalismes grammaticaux, la représentation des informations syntaxiques prend des formes variées :

- Le résultat d'une analyse en LTAG est un couple formé d'un arbre de constituants et d'un arbre de dérivation. Les dépendances locales sont codées dans la structure de constituants. Les dépendances à longue distance, les relations de contrôle et les relations d'antécédent sont indiquées par une co-référence entre un nœud et sa trace.
- Le résultat d'une analyse en HPSG est une structure de traits synthétisant l'information contenue dans la phrase, accompagnée de l'arbre des structures de traits utilisées et construites au cours de l'analyse. Toutes les dépendances sont codées dans la structure de traits finale.
- Le résultat d'une analyse en LFG est un couple formé d'un arbre de constituants et d'une structure de fonctions grammaticales. Toutes les dépendances sont codées dans la structure de fonctions grammaticales.
- Le résultat d'une analyse en CG, TLG et CCG, est une formule qui représente la contribution globale de la phrase, accompagnée d'un arbre de dérivation qui représente la démonstration de cette formule. Les dépendances locales et les relations d'antécédent simples que nous considérons ici sont réalisées par simple composition. Les dépendances à longue distance sont codées par des traits co-indexés attachés à certaines catégories, de même que les relations de contrôle.

- Les formalismes qui s'inscrivent dans le courant des Grammaires de Dépendances (DG) [Mel88], comme les GUST [Kah02, Lis06] et les XDG [Deb06], produisent des structures de dépendance. Les dépendances à longue distance sont indiquées dans la structure de dépendance syntaxique de surface, les relations de contrôle et les relations d'antécédent sont indiquées dans la structure de dépendance syntaxique profonde.
- Le résultat d'une analyse en IG est un triplet composé d'un ensemble de structures initiales, d'un arbre syntaxique final et d'une fonction d'interprétation qui les relie. Les dépendances locales sont codées dans la structure en constituants, les dépendances à longue distance, les relations de contrôle et les relations d'antécédent sont codées par des co-références de traits.

La façon dont les LTAG, les CCG et les IG codent certaines dépendances dans leurs structures syntaxiques est compréhensible par le linguiste mais requiert d'interpréter les structures syntaxiques. Ces dépendances sont donc codées d'une façon suffisamment indirecte pour que leur procédure d'extraction soit non triviale. Les procédures d'extraction proposées pour les CCG [CHS02, HS07], les LTAG [SG05] et les IG [MGP10] utilisent toutes l'historique détaillé de l'analyse syntaxique pour retrouver les dépendances codées indirectement. En LTAG et CCG, la structure de dérivation ne suffit pas. Par conséquent, les méthodes d'extraction proposées pour les CCG et les LTAG sont intrusives car elles nécessitent de modifier l'algorithme d'analyse. La méthode d'extraction pour les IG repose sur l'analyse du graphe d'interprétation, qui représente les structures initiales, la structure finale et la fonction d'interprétation. L'explicitation de la modélisation linguistique implicitement adoptée dans la grammaire permet de décrire la réalisation des différentes relations grammaticales par des motifs de graphe. Dans ces motifs, les polarités contenues dans les structures initiales jouent un rôle central pour déterminer les mots qui sont reliés par une relation grammaticale.

La diversité des structures syntaxiques produites par les formalismes grammaticaux lexicalisés nous incite à rechercher un format de structures syntaxiques plus consensuel. Cette question s'est déjà posée lors des campagnes d'évaluation des analyseurs syntaxiques pour lesquels des formats de structures syntaxiques ont été proposés [CMB03].

### 8.3.3 Structures syntaxiques utilisées pour la comparaison et l'évaluation d'analyses

De façon générale, il est très difficile de comparer et d'évaluer précisément les structures générées par différents formalismes grammaticaux. Les premières tentatives de comparaison, assimilant les formalismes aux conceptions linguistiques qui avaient motivé leur définition, ont été faites en termes formels. Le débat se poursuit sur ce terrain mais il a été éclipsé sur le plan concret par le besoin pragmatique d'évaluer et de comparer les analyses produites par différents analyseurs. Des formats de structures syntaxiques ont été conçus pour les campagnes d'évaluation, avec pour objectif d'être neutres par rapport aux formalismes et aux grammaires tout en représentant une information linguistique précise.

### Comparaison formelle d'analyses

Sur un plan purement formel, les formalismes grammaticaux sont traditionnellement comparés sous deux aspects, qui correspondent aux deux langages que ces formalismes génèrent [Cho65] :

1. les langages de chaînes, ce qui constitue leur *capacité générative faible* ;
2. les langages de structures d'analyse, par exemple les arbres d'analyse ou les arbres de démonstration, ce qui constitue leur *capacité générative forte*.

Vijay-Shanker et Weir ont par exemple montré que les TAG, les Grammaires Indexées Linéaires (LIG) [Gaz85], les Grammaires guidée par les Têtes (HG) [Pol84] et les CCG sont faiblement équivalentes [VSW94]. Cependant, alors que la capacité générative faible est une notion clairement établie, la définition précise de la *capacité générative forte* reste floue [Mil99].

Le problème est que, comme les différents formalismes grammaticaux ont été conçus pour modéliser des théories linguistiques différentes, ils n'ont pas été conçus pour générer les mêmes structures. L'impossibilité pour un formalisme de générer une structure particulière n'est donc pas un critère valable pour rejeter sa pertinence pour l'analyse de la langue naturelle. Plus généralement, les structures d'analyse sont trop propres à un formalisme pour permettre une réelle comparaison des analyses entre formalismes différents. La pertinence de la capacité générative forte pour comparer les formalismes grammaticaux est par conséquent mise en doute [Kal06, Kuh10].

### Comparaison linguistique d'analyses

Le besoin d'un format de structures syntaxiques neutre, permettant de représenter une information linguistique précise tout en restant neutre par rapport aux formalismes et aux grammaires, s'est fait ressentir concrètement pour la comparaison et l'évaluation des analyseurs syntaxiques.

Les premières campagnes évaluaient les analyseurs syntaxiques sur leur capacité à reconnaître les constituants d'une phrase. L'examen critique des résultats de ces campagnes a amené à abandonner les constituants au profit des relations grammaticales, notées GR pour Grammatical Relations, considérées comme fournissant une métrique d'évaluation plus fidèle [CBS98, CFL+02, KCR+03]. Les relations grammaticales englobent les relations de dépendance (locales et à longue distance), les relations de contrôle et les relations d'antécédents. Les relations grammaticales sont considérées comme moins idiosyncratiques et vérifiables plus rapidement et intuitivement par des humains que les structures de constituants.

L'émergence des analyseurs en dépendances statistiques a récemment relancé l'intérêt pour des corpus d'évaluation, au format GR, qui testent la capacité des analyseurs à retrouver les dépendances difficiles comme les dépendances à longue distance [RCS09, NRMGR10], mais également d'autres relations grammaticales dont le tough-movement et le contrôle [BFOZ11].

**Format de la campagne** PASSAGE Pour le français, un format de structures syntaxiques a été défini pour la campagne d'évaluation des analyseurs syntaxiques PASSAGE [13]. Ce format [14] consiste en un découpage de la phrase en groupes syntaxiques continus non récursifs, similaires à des *chunks* [15] (groupes nominaux notés GN, groupes prépositionnels notés GP, noyaux verbaux notés NV...), et de relations syntaxiques. Les relations sont étiquetées à l'aide d'un jeu de 14 étiquettes, dont chacune représente une fonction syntaxique et éventuellement la catégorie grammaticale du gouverneur de la relation. Par exemple, on distingue la relation *modifieur de nom* (MOD_N) de celle de *modifieur de verbe* (MOD_V)

Dans le guide d'annotation, les relations sont annotées entre deux mots, entre un groupe syntaxique et un mot ou entre deux groupes syntaxiques. La présence d'une tête dans chaque groupe syntaxique permet cependant de faire systématiquement porter les relations sur les mots et de se ramener ainsi à une structure plus proche d'une structure de dépendance. La structure formée par les relations dans une phrase est libre ; rien n'impose par exemple que ce soit un arbre. En particulier, le sujet des infinitifs et participes est annoté dès que possible : infinitifs introduits par un verbe à contrôle, participes têtes de participiales, gérondifs...

L'annotation en groupes et dépendances syntaxiques au format PASSAGE de la phrase *« Jean lit un livre que Marie connaît »* est donnée par la figure 8.3. Pour se conformer à l'usage en grammaires de dépendance, le sens des dépendances est inversé par rapport au format officiel PASSAGE. Le déterminant *« un »* reste isolé car PASSAGE ignore les relations déterminant-nom.

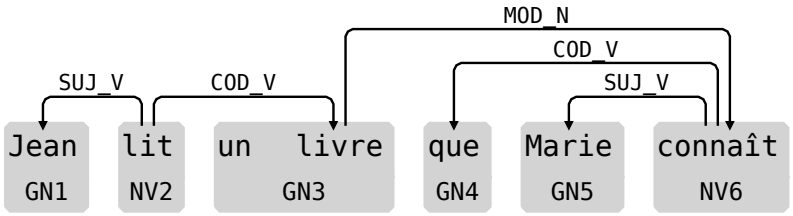



FIGURE 8.3 – Phrase annotée au format PASSAGE

Ce format a plusieurs points communs avec le format d'annotation fonctionnelle du FTB en constituants. Ainsi, ce format ne contient pas les dépendances internes aux constituants, telles que le lien entre un déterminant et le nom qu'il détermine. D'autre part, les constituants doivent être continus. Ce qui pourrait être modélisé dans un autre format par un constituant discontinu est ici annoté comme deux constituants distincts qui ont la même fonction par rapport au verbe.

Les structures hybrides du format PASSAGE sont le fruit d'un compromis entre différentes métriques et ne forment pas des structures linguistiques totalement exploitables en tant que telles. Ce format ne permet pas de construire des structures syntaxiques complètes, contrairement aux formats de relations grammaticales qui ont été définis pour l'anglais. Il ne nous est donc pas

13. http://atoll.inria.fr/passage/
14. http://www.limsi.fr/Individu/anne/Guide/PEAS_reference_annotations_v2.2.html
15. Ce ne sont donc pas nécessairement des syntagmes complets.

possible d'adopter ce format en l'état comme point de départ au calcul de la sémantique.

### 8.3.4 Structures utilisées pour l'annotation de corpus

Nous nous tournons maintenant vers les formats utilisés pour annoter des corpus. Si nous voulons pouvoir faire des expérimentations à grande échelle, il est en effet primordial d'utiliser pour les structures syntaxiques un schéma qui soit au moins compatible, sinon très similaire à un schéma d'annotation d'un corpus du français. Les premiers corpus annotés de taille significative ont été annotés en constituants. Le principal d'entre eux, le *Penn Treebank* de l'anglais constitué d'articles du *Wall Street Journal* [MMS93], a amorcé un mouvement qui a été suivi dans d'autres langues, dont le français avec le *French Treebank* constitué d'articles du journal *Le Monde* [ACT03]. Les structures en constituants sont toutefois moins adaptées aux langues à ordre des mots libre [Shi85], ce qui a favorisé la constitution de corpus annotés en dépendances, par exemple en tchèque [BHHH03]. La popularité croissante des représentations en dépendances a ensuite fait émerger le besoin de disposer de corpus annotés par des structures de dépendance syntaxique, y compris en anglais. La solution a alors consisté à définir une procédure de conversion des structures en constituants vers des structures en dépendances et à appliquer cette procédure aux corpus annotés en constituants [MMM06, CCDG09].

Nous nous concentrons maintenant sur les schémas d'annotation utilisés pour le *French Treebank*, qui est le corpus du français le plus important et le plus utilisé.

#### Schéma d'annotation fonctionnelle du FTB en constituants

Le French Treebank (ci-après FTB) [ACT03, AB04] est constitué de 32 000 phrases issues du journal *Le Monde*, découpées en constituants continus. Ces constituants sont annotés par huit fonctions grammaticales, qui réfèrent à leur rôle par rapport au verbe principal. Ces fonctions sont listées en figure 8.4.

| **Fonctions grammaticales** | |
|---|---|
| SUJ | sujet |
| OBJ | objet |
| A_OBJ | objet indirect introduit par *« à »* |
| DE_OBJ | objet indirect introduit par *« de »* |
| P_OBJ | objet indirect introduit par une autre préposition que *« à »* et *« de »* |
| ATS | attribut du sujet |
| ATO | attribut de l'objet |
| MOD | modifieur |

FIGURE 8.4 – Fonctions grammaticales du FTB

### Schéma d'annotation du FTB en dépendances de surface

Un schéma d'annotation en dépendances syntaxiques de surface pour le français [CCF11] a été dérivé de l'annotation en constituants du FTB [CCDG09, CCD10]. Dans ce schéma, la structure de dépendance associée à une phrase est un arbre orienté dont l'ensemble des nœuds correspond à l'ensemble des formes fléchies de la phrase (même celles qui sont sémantiquement vides) et les arêtes sont étiquetées par des relations de dépendance. Les relations de dépendance utilisées dans ce schéma sont listées en figure 8.5.

| | |
|---|---|
| **Relations pour gouverneurs verbaux** | |
| SUJ | sujet |
| SUJ_IMPERS | sujet impersonnel |
| OBJ | objet |
| A-OBJ | argument prépositionnel introduit par *« à »*, non locatif |
| DE-OBJ | argument prépositionnel introduit par *« de »*, non locatif |
| P-OBJ | autre argument prépositionnel |
| P-OBJ_AGT | argument prépositionnel complément d'agent (passif et causatif) |
| P-OBJ_LOC | argument prépositionnel locatif |
| ATS | attribut du sujet |
| ATO | attribut de l'objet |
| MOD | modificateur |
| MOD_CLEFT | subordonnée dans une construction clivée |
| MOD_LOC | modificateur sémantiquement locatif |
| AUX_TPS | auxiliaire de temps |
| AUX_PASS | auxiliaire du passif |
| AUX_CAUS | auxiliaire de causation |
| AFF | clitique figé (affixe) |
| AFF_MOYEN | clitique *« se »* pour la voix moyenne |
| **Relations pour gouverneurs non verbaux** | |
| OBJ | relation entre une préposition et le syntagme qu'elle introduit |
| ARG | relation entre une préposition et une autre préposition liée (ex : *« de Charybde en Scylla »*) |
| ARG_COMP | relation entre une comparative et son gouverneur |
| ARG_CONS | relation entre une consécutive et son gouverneur adverbial |
| MOD | modifieur repéré structuralement, sauf subordonnée relative |
| MOD_REL | subordonnée relative adnominale |
| DEP | dépendant ambigü entre argument et modifieur |
| DET | relation entre un élément déterminé et son déterminant |
| **Relations pour la coordination** | |
| COORD | relation entre un élément coordonné et le coordonnant suivant |
| DEP_COORD | relation entre un coordonnant et l'élément coordonné suivant |
| **Relations pour la ponctuation** | |
| PONCT | relation entre une tête de proposition et une ponctuation |

FIGURE 8.5 – Étiquettes de relations de dépendance du schéma du FTB

Sur la figure 8.5, toutes les relations ne sont pas sur le même alignement vertical. Le schéma d'annotation en dépendances comporte en effet deux niveaux de précision. Les étiquettes alignées le plus à gauche, comme SUJ et OBJ, forment le jeu de relations de base. C'est ce jeu qui est utilisé dans l'annotation en dépendances du FTB obtenue par conversion automatique de l'annotation en constituants. Les étiquettes du deuxième alignement sont des raffinements des étiquettes précédentes, pour certains cas particuliers : un sujet impersonnel est noté SUJ_IMPERS plutôt que SUJ, un argument prépositionnel complément d'agent est noté P-OBJ_AGT plutôt que P-OBJ... Ces étiquettes plus précises ne sont actuellement utilisées que pour l'annotation manuelle du FTB.

La conversion entre le schéma d'annotation en constituants et le schéma d'annotation en dépendances n'est pas immédiate pour deux raisons principales :

- Premièrement, dans le schéma d'annotation en dépendances, chaque forme fléchie de la phrase est la source ou la destination d'au moins une relation de dépendance, y compris à l'intérieur des constituants. Dans le schéma d'annotation en constituants, seuls certains constituants ont une fonction : ce sont les constituants qui ont une fonction grammaticale par rapport au verbe principal de la proposition. Les huit fonctions grammaticales utilisées pour les gouverneurs verbaux dans le schéma en constituants sont utilisées dans le schéma en dépendances comme étiquettes de relations. Dans le schéma en dépendances, l'usage de ces étiquettes est cependant étendu et des relations supplémentaires sont introduites pour les auxiliaires, les déterminants, les compléments et modifieurs de noms et d'adjectifs, les comparatifs, les coordinations...
- Deuxièmement, le schéma d'annotation en dépendances retenu n'impose pas la projectivité de l'arbre de dépendances. Cela signifie que la projection d'un nœud, c'est-à-dire le sous-arbre des nœuds qu'il gouverne, ne doit pas nécessairement correspondre à un segment continu de la phrase. La non-projectivité permet une modélisation simple des constituants discontinus et des dépendances à longue distance.

### 8.3.5 Bilan

Dans cette section, nous avons dressé la liste des informations syntaxiques nécessaires au calcul de la sémantique. Nous avons ensuite constaté que les structures syntaxiques produites par les différents formalismes grammaticaux présentaient un degré élevé de variabilité. Ces structures utilisent des moyens différents pour représenter les mêmes informations syntaxiques. En quête d'un format de structures qui soit moins idiosyncratique, nous nous sommes tourné vers les structures qui ont été conçues pour l'évaluation et la comparaison des analyseurs syntaxiques. Les formats les plus intéressants ont été proposés pour l'anglais ; ils utilisent des relations grammaticales. Il n'y a pas de format strictement équivalent pour le français. Le format le plus proche est celui de la campagne PASSAGE, mais c'est un format hybride qui mêle un découpage en groupes syntaxiques avec des relations grammaticales. Nous avons enfin examiné les schémas qui ont été utilisés pour les structures syntaxiques qui annotent le *French Treebank*. Le schéma d'annotation en dépendances de surface décrit des structures qui ne représentent pas toute l'information dont

nous avons besoin, mais ces structures sont suffisamment complètes pour être enrichies. Dans la prochaine section, nous définissons une extension du schéma d'annotation du FTB qui permet d'enrichir l'annotation en dépendances de surface par les relations de contrôle et d'antécédents.

## 8.4 Proposition de format : extension du schéma d'annotation en dépendances de surface du FTB

Nous présentons enfin dans cette section le format des structures syntaxiques que nous utilisons pour calculer la sémantique. Ce format étend le schéma d'annotation en dépendances de surface du FTB, par l'ajout de relations grammaticales dont les étiquettes sont, dès que possible, très proches des dépendances de surface équivalentes du FTB.

Les relations grammaticales que nous ajoutons désignent :

- les arguments profonds, lexicalement ou grammaticalement déterminés, des verbes à l'infinitif ou au participe, lorsqu'ils sont présents dans la phrase ;
- les antécédents d'anaphores syntaxiquement déterminés ;
- les sujets des syntagmes adjectivaux ;
- les arguments manquant aux conjoints d'une coordination.

Le premier point regroupe les relations de contrôle au sens large (verbes à contrôle, à montée et modaux, gérondifs) et le tough-movement. Nous reviendrons au chapitre 10 sur l'attribution de sujets aux syntagmes adjectivaux.

Au final, les graphes syntaxiques que nous manipulons sont définis sur une signature de graphes composée pour les nœuds, de la signature de traits de la figure 8.6, et pour les arêtes, de l'ensemble d'étiquettes de relations de dépendance de la figure 8.5 augmenté de l'ensemble d'étiquettes de relations grammaticales de la figure 8.7.

## Conclusion

Nous avons construit progressivement dans ce chapitre la démarche que nous adoptons pour calculer une représentation sémantique à partir d'une analyse syntaxique. Nous avons d'abord présenté le type de structures sémantiques que nous voulons construire. Nous avons ensuite examiné les grandes approches du lien entre analyse syntaxique et sémantique et avons évalué leur pertinence dans le contexte de notre travail, qui est le développement d'une chaîne de traitement linguistique autour de l'analyseur LEOPAR des IG, avec la grammaire FRIGRAM du français. Cela nous a amenés à adopter une perspective séquentielle dans laquelle le calcul de la sémantique se fait à partir d'une structure syntaxique déjà construite. Nous avons ensuite cherché à déterminer les structures syntaxiques que nous pouvions utiliser. Pour cela, nous avons listé les informations syntaxiques dont nous avons besoin. Nous avons ensuite observé la façon dont elles étaient représentées dans les analyses produites dans les différents formalismes, dans les structures utilisées pour l'évaluation et la comparaison des analyseurs, et dans les

| **Nom du trait** | **Valeurs possibles** | **Signification** |
|---|---|---|
| cat | | catégorie syntaxique |
| | adj | adjectif |
| | adv | adverbe |
| | det | déterminant |
| | n | nom commun |
| | np | nom propre |
| | prep | préposition |
| | pro | pronom |
| | prorel | pronom relatif |
| | v | verbe |
| frame | | cadre verbal |
| | * | toute chaîne de caractères |
| gen | | genre |
| | m | masculin |
| | f | féminin |
| lemma | | lemme |
| | * | toute chaîne de caractères |
| mood | | mode |
| | ind | indicatif |
| | imp | impératif |
| | inf | infinitif |
| | cond | conditionnel |
| | presp | participe présent |
| | pastp | participe passé |
| num | | nombre |
| | sg | singulier |
| | pl | pluriel |
| phon | | forme phonologique |
| | * | toute chaîne de caractères |
| prep | | préposition |
| | * | toute chaîne de caractères |
| tense | | temps verbal |
| | cond | conditionnel présent |
| | cond_past | conditionnel passé |
| | futur | futur |
| | futur_ant | futur antérieur |
| | imparfait | imparfait |
| | passe_ant | passé antérieur |
| | passe_comp | passé composé |
| | past | passé simple |
| | plus_que_parfait | plus que parfait |
| | prst | présent |
| voice | | voix |
| | active | active |
| | middle | moyenne |
| | passive | passive |

FIGURE 8.6 – Traits utilisés pour les informations morphosyntaxiques

| **Relations profondes pour gouverneurs verbaux** | |
|---|---|
| SUJP | sujet |
| OBJP | objet |
| A-OBJP | argument prépositionnel introduit par *« à »*, non locatif |
| DE-OBJP | argument prépositionnel introduit par *« de »*, non locatif |
| P-OBJP | autre argument prépositionnel |
| ATSP | attribut du sujet |
| ATOP | attribut de l'objet |
| **Relations pour gouverneurs non verbaux** | |
| SUJP | sujet d'un adjectif |
| **Relations d'antécédents** | |
| ANT_APP | antécédent d'un pronom rélatif dans une relative en apposition |
| ANT_REFL | antécédent d'un pronom réfléchi |
| ANT_REL | antécédent d'un pronom relatif dans une relative adnominale |
| ANT_REP | antécédent d'un pronom sujet répété |

FIGURE 8.7 – Étiquettes des relations grammaticales du schéma étendu

structures d'annotation des corpus. Nous avons finalement proposé un format de structures syntaxiques pour le français qui étend le format en dépendances de surface du FTB par des relations grammaticales de contrôle et d'antécédent.

Dans les chapitres suivants, nous montrons comment construire une interface syntaxe - sémantique générique pour les formalismes lexicalisés, à partir d'une telle structure en dépendances. Dans le chapitre 9, nous proposons un calcul de réécriture de graphes adapté à la transformation de structures de dépendances et un programme de réécriture qui produit une structure sémantique sous-spécifiée à partir d'une structure dans notre format étendu. Au chapitre 10, nous proposons un programme de réécriture pour ajouter aux structures de dépendance syntaxique de surface du FTB les relations grammaticales supplémentaires de notre format étendu. Ce programme garantit la généralité de notre démarche, en la rendant compatible avec tous les formalismes capables de produire une structure de dépendance syntaxique de surface. La composition de ces deux programmes nous permet de produire des structures sémantiques sous-spécifiées à partir des structures de dépendance de surface du FTB.

# Chapitre 9

# Réécriture de graphes modulaire pour l'interface syntaxe - sémantique

Dans le chapitre précédent, nous avons proposé de calculer des représentations sémantiques sous-spécifiées à partir de structures de dépendance syntaxique enrichies avec des relations grammaticales supplémentaires. Dans ce chapitre, nous montrons comment réaliser concrètement ce calcul à l'aide de la réécriture de graphes. Dans la section 9.1, nous motivons notre utilisation de la réécriture de graphes et nous donnons l'intuition du type de réécriture de graphes que nous utilisons. Dans la section 9.2, nous présentons le format des structures sémantiques que nous voulons produire. Dans la section 9.3, nous définissons le calcul de réécriture de graphes modulaire que nous utilisons. Dans la section 9.4, nous proposons un programme de réécriture de graphes qui calcule une représentation sémantique sous-spécifiée à partir d'une structure de dépendance syntaxique enrichie. Dans la section 9.5, nous appliquons ce programme de réécriture aux phrases du guide d'annotation du FTB en dépendances. Dans la section 9.6, nous discutons des limites et des perspectives immédiates ouvertes par l'utilisation de ce programme de réécriture, et nous le situons par rapport aux travaux les plus proches.

## 9.1 Motivation

Nous avons proposé au chapitre précédent de considérer des structures syntaxiques qui sont des arbres de dépendances de surface, auxquels sont ajoutées d'autres relations grammaticales qui dénotent une information plus profonde : relations de contrôle, arguments d'infinitifs déterminés par les constructions de tough-movement, sujets des syntagmes adjectivaux, antécédents d'anaphores déterminés par la syntaxe et arguments manquants aux conjoints d'une coordination. L'ajout de ces relations grammaticales a pour effet de transformer l'arbre de dépendance syntaxique de surface d'une phrase en un graphe syntaxique. Notre objectif étant de calculer des représentations sémantiques sous-spécifiées à partir de ces structures syntaxiques, nous devons maintenant choisir un modèle de calcul adapté aux structures de graphes.

### 9.1.1 Réécriture de graphes

Le modèle de calcul le plus utilisé en sémantique formelle est le $\lambda$-calcul. Ce modèle est adapté au calcul sur des arbres mais moins sur des graphes. Nos structures de départ étant des graphes syntaxiques, le $\lambda$-calcul n'est donc pas un bon candidat pour calculer la sémantique dans notre approche. Le modèle de calcul le plus naturel sur ces structures semble être la *réécriture de graphes* [Roz97]. Un système de réécriture de graphes est défini par un ensemble de *règles de réécriture de graphes*. L'application d'une règle à un graphe est déclenchée par le filtrage (*pattern matching* en anglais) d'un motif de graphe. Le sous-graphe correspondant au motif est alors isolé de son contexte pour être modifié localement, puis recollé dans le graphe. Appliquer une règle de réécriture a donc pour effet de transformer le graphe. Un calcul est une séquence d'applications de règles de réécriture sur un graphe donné.

Alors que la réécriture de mots, la réécriture d'arbres et la réécriture de termes peuvent être définies de façon directe et canonique [BN99], la réécriture de graphes est bien plus problématique [Roz97]. Deux points en particulier posent problème : la définition de la fonction de filtrage, qui reconnaît un motif dans un graphe, et la définition de la fonction de recollement, qui recolle le sous-graphe modifié dans son contexte. Ainsi, il n'existe pas de définition canonique d'un système de réécriture de graphes. La réécriture de graphes peut être définie dans un cadre catégorique, comme dans les approches par *single pushout* (SPO) et *double pushout* (DPO) [Roz97]. En pratique, il est cependant bien plus facile d'adopter une vision plus opérationnelle de la réécriture, dans laquelle la modification du graphe (la *partie droite* de la règle de réécriture) est définie par une séquence de commandes [Ech08] ; le contrôle de la façon dont les règles sont appliquées (la *partie gauche* de la règle) utilise, comme dans les approches classiques en réécriture de graphes, le filtrage de motifs. Nous nous inscrivons dans cette deuxième perspective et nous définissons, dans la section 9.3, un calcul de réécriture de graphes à base de commandes.

### 9.1.2 Réécriture de graphes modulaire

Dans un système de traitement linguistique, la localité des règles de réécriture permet de traiter chaque phénomène linguistique séparément. La réécriture de graphes est cependant très peu utilisée en TAL, parmi les exceptions figurent [Hyv84, BW01, Cro05, JdR07, BG09, CK10]. Cela peut s'expliquer par la difficulté de gérer de grands ensembles de règles. Lors de l'application d'un système de réécriture, l'utilisateur a peu de contrôle sur la séquence de pas de réécriture. Chaque pas de calcul est déclenché par des conditions locales sur le graphe entier et donc, plus il y a de règles, plus l'interaction entre les règles et la cohérence du système tout entier deviennent difficiles à maintenir. Pour résoudre ce problème, nous proposons d'organiser les règles en *modules*.

Un module est un ensemble de règles cohérent d'un point de vue linguistique, et qui représente une étape donnée de la transformation globale. Par exemple, le programme de réécriture que nous proposons à la section 9.4, comporte un module dont les règles ont pour effet de remplacer

un verbe avec ses compléments syntaxiques par le prédicat correspondant au verbe avec ses arguments sémantiques. Un autre module résout les liens d'anaphore qui sont internes à la phrase et qui sont déterminés par la syntaxe.

Le regroupement des règles en modules présente plusieurs avantages. Les trois principaux sont, pour nous, les suivants. Premièrement, les modules facilitent la conception d'une stratégie d'évaluation. Les modules constituent en effet un niveau intermédiaire entre le niveau des règles et le niveau du système global. Il est alors possible de définir des stratégies d'évaluation sur les modules, au lieu de les définir sur l'ensemble des règles. Nous ne considérons ici qu'une seule stratégie, qui consiste à définir un ordre séquentiel pour l'application des modules ; des stratégies plus complexes sont cependant possibles.

Deuxièmement, les modules facilitent le contrôle du bon déroulement de la réécriture. Il est en effet beaucoup plus facile de vérifier le résultat de l'application d'un petit nombre de règles. Il est également possible d'inférer automatiquement certaines propriétés invariantes des graphes au cours du calcul dans un module particulier. Cela permet de simplifier l'écriture des règles pour les modules suivants.

Troisièmement, le regroupement des règles en modules permet d'améliorer l'efficacité du calcul. Comme nous l'avons vu au chapitre précédent, le lien entre la syntaxe et la sémantique n'est pas fonctionnel mais relationnel et donc notre programme de réécriture ne peut pas être globalement confluent. Or, la confluence de la réécriture est une donnée critique pour la performance du programme, car elle permet de ne calculer qu'une seule forme normale au lieu de les calculer toutes. Les modules apportent une réponse partielle à ce problème, en offrant la possibilité d'isoler des sous-ensembles confluents de règles pour former des modules confluents.

Maintenant que nous avons choisi un modèle de calcul, il nous reste à déterminer le format exact des structures sémantiques que nous allons calculer.

## 9.2 Structure sémantique produite

Au chapitre précédent, nous nous sommes fixé pour objectif de produire des structures sémantiques sous-spécifiées, qui sont des descriptions d'ensembles de formules logiques. Jusqu'ici, nous n'avons cependant pas adopté de format précis pour ces structures sémantiques.

### 9.2.1 Choix du format

Les principaux formalismes de sémantique sous-spécifiée (MRS, Hole Semantics, CLLS) ayant des caractéristiques et une expressivité très semblables, le choix du formalisme semble avoir peu de conséquences formelles. Nous pouvons donc prendre en compte d'autres facteurs. Ainsi, sur le plan linguistique, le format syntaxique que nous utilisons repose sur la notion de dépendance syntaxique et sur celle, plus générale, de relation grammaticale. Or, les dépendances syntaxiques et les relations grammaticales coïncident largement avec les dépendances sémantiques [Mel88]. La façon la plus directe de produire des structures sémantiques semble donc de calculer, à partir

de ces structures de dépendance syntaxique enrichies, des structures de dépendance sémantique.

Actuellement, le seul format de structures sémantiques sous-spécifiées qui repose sur la notion de dépendance sémantique est la Dependency MRS, notée DMRS [Cop09]. Ce format a été conçu pour fournir des représentations alternatives, plus lisibles et plus faciles à comparer et à factoriser que les structures MRS et leurs équivalents pour l'analyse robuste, les structures RMRS [Cop07]. Une motivation supplémentaire est d'obtenir un format plus facile à utiliser pour la sémantique distributionnelle [PL07]. Par conception, une structure DMRS contient donc une information équivalente à une structure RMRS ou MRS. Une structure DMRS peut être produite de façon déterministe à partir d'une structure MRS et, inversement, une structure MRS peut être produite de façon déterministe à partir d'une structure DMRS.

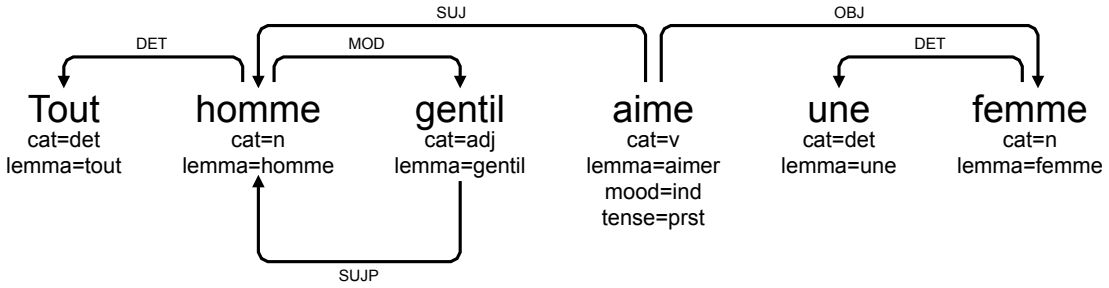
### 9.2.2 Description du format

Une structure DMRS, comme la structure du bas dans la figure 9.1, est un graphe orienté dont les nœuds représentent des prédicats. Chaque prédicat, sauf ceux qui représentent des quantificateurs généralisés, a une *variable caractéristique*, comme en sémantique Davidsonienne. Pour les verbes, les adjectifs, les adverbes et les prépositions sémantiquement pleines, la variable caractéristique est une variable d'événement ; pour les noms, c'est une variable d'individu. Dans une structure DMRS, chaque prédicat a une variable caractéristique distincte.

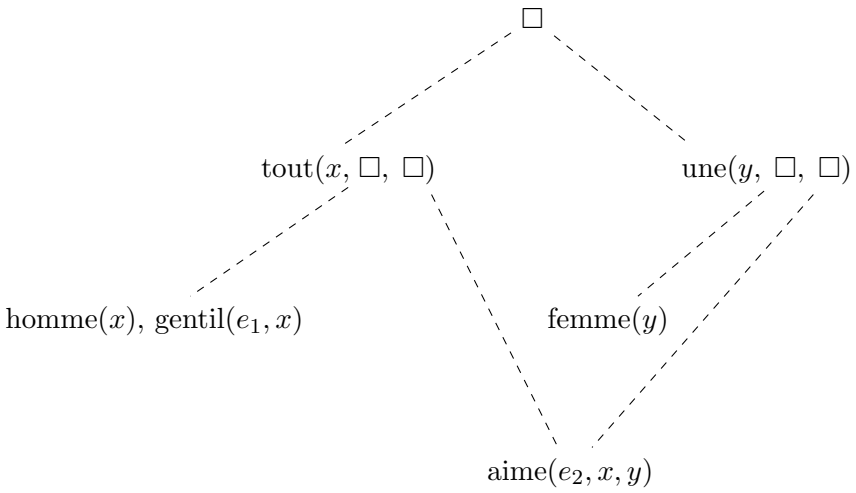
Chaque arête du graphe est étiquetée par une information, qui appartient à l'un des trois types suivants :

- les relations prédicat - argument, notées `ARG`$i$, où $i$ est un entier fixé par convention selon l'ordre d'oblicité SUJ, OBJ, ATS, ATO, A-OBJ, DE-OBJ... ; $i$ est supérieur ou égal à 1, car l'`ARG0` est réservé par convention à la variable caractéristique du prédicat ;
- les restrictions de portée des quantificateurs généralisés, qui sont notées `RSTR` ; les corps de portée ne sont pas marqués explicitement mais ils peuvent être calculés à partir du graphe ;
- le mode de combinaison sémantique entre les différents éléments de la structure, au choix parmi :
  - `EQ` quand deux prédicats sont directement conjoints et font partie d'un même nœud dans la structure MRS correspondante ;
  - `H` quand un prédicat est dans la portée d'un autre prédicat, ce dernier étant flottant ; la variable caractéristique du premier prédicat n'est pas obligatoirement un argument direct du deuxième car des quantificateurs peuvent s'intercaler entre ces deux prédicats ;
  - `NEQ` quand la variable caractéristique d'un prédicat est un argument d'un autre prédicat, mais que ces deux prédicats ne sont pas directement liés, de façon structurelle, dans la représentation MRS équivalente.

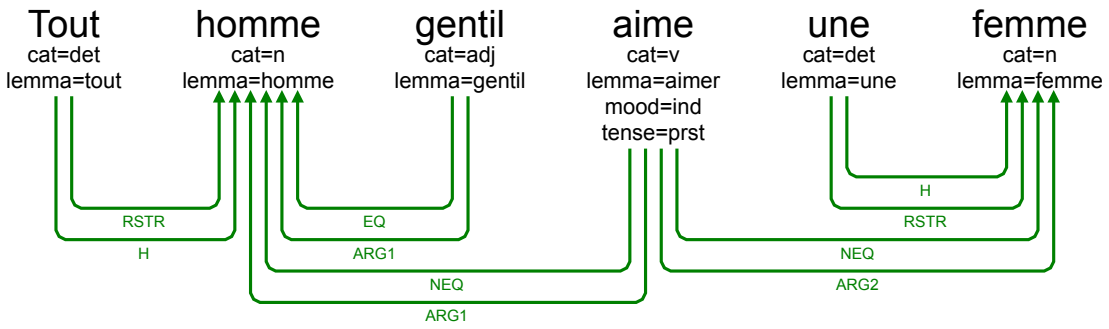
La figure 9.1 contient la structure syntaxique, la structure MRS et la structure DMRS associées à notre exemple. Le prédicat *« tout »* est un quantificateur flottant (`H`), qui a dans sa restriction (`RSTR`) *« homme »*. La variable caractéristique de *« homme »* est le premier (et unique) argu-




(a) Structure syntaxique




(b) Structure MRS




(c) Structure DMRS

FIGURE 9.1 – Structures syntaxique et sémantiques de *« Tout homme gentil aime une femme »*

ment (ARG1) du prédicat *« gentil »*, et le premier argument du prédicat *« aime »*. Les prédicats *« homme »* et *« gentil »* sont directement conjoints et ils font partie du même nœud dans la structure MRS (EQ). Le prédicat *« aime »* n'est pas structurellement lié au prédicat *« homme »* dans la MRS (NEQ). Le prédicat *« femme »* est dans la restriction (RSTR) de *« une »*, qui flotte (H). La variable caractéristique de ce prédicat est également le deuxième argument (ARG2) de *« aime »*, et ce deux prédicats ne sont pas structurellement liés dans la MRS (NEQ).

La MRS, la RMRS et la DMRS ont été conçues pour des expérimentations à grande échelle, sans prendre position sur des distinctions sémantiques fines. C'est la raison pour laquelle les numéros des arguments des prédicats suivent l'ordre d'oblicité syntaxique et ne cherchent aucunement à exprimer des généralités sémantiques.

### Différences avec les structures DMRS standard

Les structures DMRS que nous considérons dans ce manuscrit diffèrent des structures DMRS standard définies par Copestake sur deux points : nos structures sémantiques contiennent explicitement les arguments optionnels non réalisés syntaxiquement et tous les groupes nominaux sont quantifiés. Ces deux points se manifestent par la création de nœuds vides, dont la forme phonologique est notée « $\epsilon$ ».

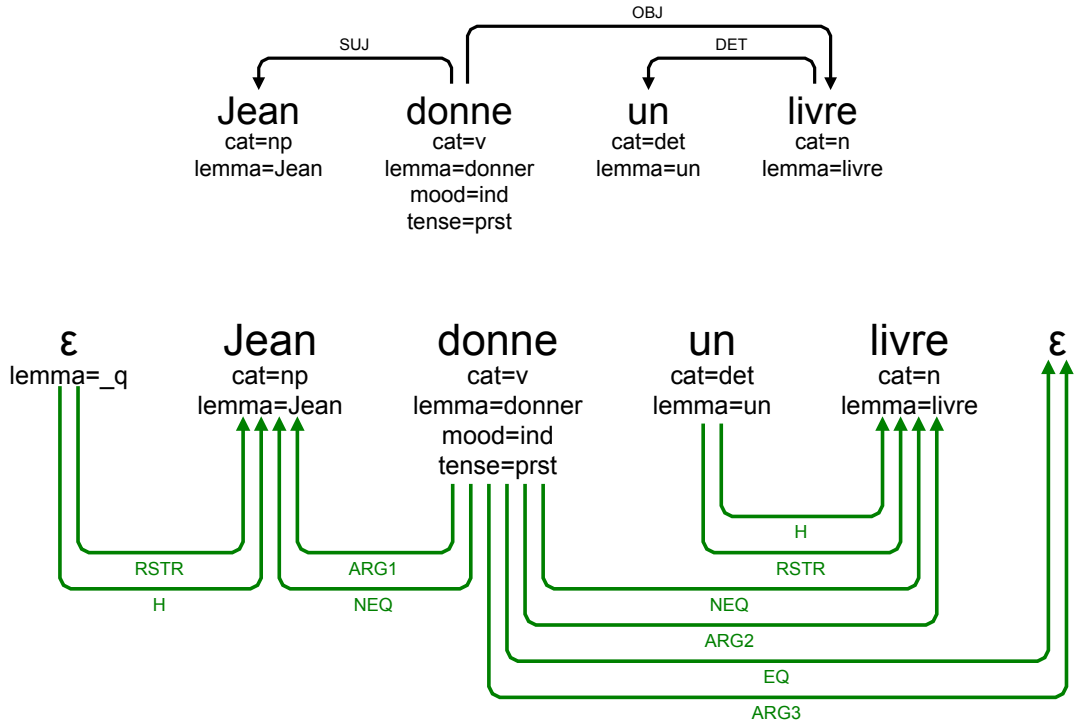



FIGURE 9.2 – Analyses syntaxique et sémantique de *« Jean donne un livre »*

Dans les structures DMRS standard, les arguments optionnels non réalisés des verbes n'ap-

paraissent pas, alors qu'ils sont traduits par des nœuds vides dans nos structures DMRS. Par exemple, dans la phrase *« Jean donne un livre »*, dont l'analyse syntaxique est donnée dans la partie supérieure de la figure 9.2, l'objet indirect, qui désigne à qui Jean donne le livre, n'est pas réalisé. Le prédicat sémantique correspondant à *« donner »* a trois arguments. Dans la structure MRS associée à cette phrase, l'argument sémantique correspondant à l'objet indirect est donc une variable libre : $donner(x, y, u)$. Or, les nœuds d'une structure DMRS n'ont pas d'arité explicite. À partir d'une structure DMRS standard, il est donc impossible de reconstruire la structure MRS correspondante sans avoir recours à une information lexicale externe, qui spécifie le nombre d'arguments du prédicat associé au verbe. Ce recours au lexique n'est pas nécessaire pour construire des structures MRS à partir des structures DMRS que nous considérons. Nos structures DMRS sont ainsi directement transformables en structures MRS et donc réellement équivalentes à celles-ci.

De plus, nous suivons la convention adoptée dans la grammaire ERG et considérons que tout groupe nominal est quantifié [Fli08]. Nous ajoutons donc un quantificateur par défaut à tout groupe nominal qui n'a pas de quantificateur explicite en syntaxe. Ces quantificateurs ont une forme phonologique vide et leur lemme est *« _q »*. Dans notre exemple, nous ajoutons deux quantificateurs par défaut, l'un pour *« Jean »* et un autre pour l'argument non réalisé.

Maintenant que nous avons présenté le format des structures sémantiques, nous devons définir le calcul de réécriture de graphes que nous utilisons.

## 9.3 Calcul de réécriture modulaire de graphes

Nous définissons dans cette section un calcul de réécriture de graphes modulaire à base de commandes. Plus précisément, la modification du graphe (la *partie droite* de la règle de réécriture) est définie par une séquence de commandes ; le contrôle de la façon dont les règles sont appliquées (la *partie gauche* de la règle) utilise, comme dans les approches classiques en réécriture de graphes, le filtrage de motifs (*pattern matching* en anglais).

Dans ce cadre, une *règle* est une paire formée d'un *motif* (*pattern* en anglais) et d'une séquence de *commandes*. Nous illustrons ces notions par des exemples de règles et de réécriture.

### 9.3.1 Graphes

#### Arêtes

Nous supposons donné un ensemble fini $\mathcal{E}$ d'étiquettes d'arêtes qui correspondent aux noms des relations de dépendances utilisés pour annoter les phrases. Ces étiquettes peuvent correspondre à des dépendances syntaxiques ou sémantiques. Dans ce chapitre, nous utilisons l'ensemble $\mathcal{E}$ construit par union des ensembles d'étiquettes des figures 8.5 et 8.7.

#### Nœuds

Pour décorer les nœuds, nous utilisons les structures de traits que nous avons déjà définies au chapitre 2, page 28. Seule la notation change : les traits sont dorénavant notés $f = v$, et non plus $f : v$. Nous rappelons ici les définitions utiles.

**Définition 9.3.1** (Signature de traits)**.** Une **signature de traits** $(\mathcal{F}, \mathcal{V})$ est définie par :
- un ensemble fini $\mathcal{F}$ de constantes appelées **noms de traits** ;
- un ensemble fini $\mathcal{V}$ contenant, pour chaque nom de trait $f$ dans $\mathcal{F}$, un ensemble fini $\mathcal{A}_f$ de constantes appelées **valeurs atomiques**.

Les traits sont construits relativement à une signature de traits.

**Définition 9.3.2** (Trait)**.** Un **trait** est une paire $(f, v)$ avec $f \in \mathcal{F}$ et $v \in \mathcal{A}_f$ ; dans ce qui suit, on écrit le trait $(f, v)$ comme $f = v$.

**Définition 9.3.3** (Structure de traits)**.** Une **structure de traits** sur une signature de traits $(\mathcal{F}, \mathcal{V})$ est un ensemble fini de traits $(f_i, v_i)$ de noms distincts : si $i \neq j$ alors $f_i \neq f_j$.

Dans ce chapitre, nous utilisons la signature de traits $(\mathcal{F}, \mathcal{V})$ de la figure 8.6.

#### Graphes

**Définition 9.3.4** (Graphe)**.** Un graphe $\mathcal{G}$ est défini par un 6-uplet $(\mathcal{N}, \mathbf{fs}, \mathcal{A}, \mathbf{lab}, \sigma, \tau)$ avec :
- un ensemble fini $\mathcal{N}$ de nœuds ;
- une fonction d'étiquetage **fs** de $\mathcal{N}$ dans les structures de traits définies sur $(\mathcal{F}, \mathcal{V})$ ;
- un ensemble fini $\mathcal{A}$ d'arêtes ;
- une fonction d'étiquetage **lab** de $\mathcal{A}$ dans $\mathcal{E}$ ;
- deux fonctions $\sigma$ et $\tau$ de $\mathcal{A}$ dans $\mathcal{N}$ qui donnent pour chaque arête, respectivement, sa source et sa destination.

De plus, nous interdisons qu'il existe, entre deux nœuds, deux arêtes de même orientation qui aient la même étiquette.

### 9.3.2 Motifs et filtres

#### Nœuds de motif

Les nœuds des graphes que nous manipulons contiennent des structures de traits telles que nous venons de les définir. Cependant, pour écrire des règles de réécriture suffisamment générales, les motifs en partie gauche de nos règles utilisent un langage plus riche de structures de traits sous-spécifiées.

**Définition 9.3.5** (Trait de motif)**.** Un **trait de motif** sur une signature de traits $(\mathcal{F}, \mathcal{V})$ est un couple $(f, V)$ tel que $f \in \mathcal{F}$ et $V \subset \mathcal{A}_f$. Le trait de motif $(f, \{v_1, \ldots, v_k\})$ est noté $f = v_1 | \ldots | v_k$ si $V$ n'est pas vide, $f = \perp$ sinon.

Le trait $cat = v|aux$ signifie que le nom de trait $cat$ doit être associé à l'une des deux valeurs $v$ ou $aux$.

**Définition 9.3.6** (Structure de traits de motif)**.** Une **structure de traits de motif** sur une signature de traits $(\mathcal{F}, \mathcal{V})$ est un ensemble fini de traits de motif $(f_i, V_i)$ de noms distincts.

### Filtrage de motif

**Définition 9.3.7** (Filtrage de structure de traits)**.** Soit $\rho$ une structure de traits de motif et $\sigma$ une structure de traits. Nous disons que $\rho$ **filtre** $\sigma$, noté $\rho \ltimes \sigma$ si et seulement si pour tout trait de motif $(f, V) \in \rho$, il existe un trait $(f, v) \in \sigma$ tel que $v \in V$.

Notons que le filtrage de structure de traits est, par définition, différent de l'unification de structures de traits. En effet, s'il existe un trait de motif $(f, V) \in \rho$ mais aucun trait $(f, v) \in \sigma$, alors $\rho$ ne filtre pas $\sigma$, alors que $\rho$ et $\sigma$ peuvent être unifiables.

### Filtre de motif

Formellement, un *motif basique* est un graphe.

**Définition 9.3.8** (Filtre pour un motif basique)**.** Un *filtre* $\phi$ pour un motif basique $\mu = (\mathcal{N}', \mathbf{fs}', \mathcal{A}', \mathbf{lab}', \sigma', \tau')$ dans un graphe $\mathcal{G} = (\mathcal{N}, \mathbf{fs}, \mathcal{A}, \mathbf{lab}, \sigma, \tau)$ est un couple de fonctions injectives $\phi$ et $\psi$, $\phi$ de $\mathcal{N}'$ dans $\mathcal{N}$ et $\psi$ de $\mathcal{A}'$ dans $\mathcal{A}$, tels que $(\phi, \psi)$ :

- respecte l'étiquetage des nœuds : $\mathbf{fs}'(n) \ltimes \mathbf{fs}(\phi(n))$ ;
- respecte l'étiquetage des arêtes : $\mathbf{lab}(\psi(A)) \subset \mathbf{lab}'(A)$ ;
- respecte les sources des arêtes : $\sigma(\psi(A)) = \phi(\sigma'(A))$ ;
- respecte les destinations des arêtes : $\tau(\psi(A)) = \phi(\tau'(A))$.

La condition sur l'étiquetage des arêtes permet de définir des motifs basiques contenant des disjonctions d'étiquettes sur les arêtes : l'étiquette de l'arête de l'instance doit être incluse dans l'ensemble des étiquettes de l'arête du motif basique.

Les motifs sont une extension des motifs basiques qui permettent à l'utilisateur de spécifier des conditions négatives.

**Définition 9.3.9** (Motif)**.** Un *motif* $\mathcal{M} = \langle \mu, \{\nu_1, \ldots, \nu_k\} \rangle$ est défini par :

- un motif basique $\mu$, appelé *motif basique positif* ;
- une séquence (éventuellement vide) de $k$ motifs basiques $\nu_1, \ldots, \nu_k$, appelés *motifs basiques négatifs*.

Chaque motif basique négatif est interprété indépendamment des autres.

**Définition 9.3.10** (Filtre pour un motif)**.** Un filtre pour $\langle \mu, \{\nu_1, \ldots, \nu_k\} \rangle$ est un filtre pour $\mu$ qui ne peut pas être étendu en un filtre pour un $\mu; \nu_i$, pour tout $i, 1 \leq i \leq k$.

### 9.3.3 Commandes

Les commandes sont des opérations de bas niveau sur les graphes qui sont utilisées pour décrire la réécriture du graphe dans l'application d'une règle. Dans la description ci-dessous, nous supposons donné un filtre de motif $(\phi, \psi) : \mathcal{M} \to \mathcal{G}$. Nous décrivons ici l'ensemble de commandes que nous avons utilisé dans nos expérimentations. Cet ensemble de commandes peut naturellement être étendu.

- **del_edge**$(\alpha, \beta, \ell)$ supprime l'arête étiquetée $\ell$ entre $\alpha$ et $\beta$. Plus formellement, nous supposons que $\alpha \in \mathcal{N}'$, $\beta \in \mathcal{N}'$ et $\mathcal{M}$ contient une arête $a$ de $\alpha$ vers $\beta$ étiquetée par $\ell \in \mathcal{E}$. Alors, **del_edge**$(\alpha, \beta, \ell)(\mathcal{G})$ est le graphe $\mathcal{G}$ sans l'arête $\phi(e)$ Dans ce qui suit, nous donnons seulement la définition intuitive de la commande : grâce à l'injectivité du filtrage $\phi$, nous oublions de façon implicite la distinction entre $x$ et $\phi(x)$.
- **add_edge**$(\alpha, \beta, \ell)$ ajoute une arête étiquetée $\ell$ entre $\alpha$ et $\beta$. On suppose qu'une telle arête n'existe pas déjà dans $\mathcal{G}$.
- **shift_edge**$(\alpha, \beta)$ redirige toutes les arêtes incidentes à $\alpha$ : chaque arête dont $\alpha$ est la source est déplacée pour partir de $\beta$ ; de façon similaire, chaque arête dont $\alpha$ est la destination est déplacée pour aboutir à $\beta$ ;
- **del_node**$(\alpha)$ supprime le nœud $\alpha$ dans $\mathcal{G}$. Si $\mathcal{G}$ contient des arêtes incidentes à $\alpha$, celles-ci sont supprimées silencieusement.
- **add_node**$(\beta)$ ajoute un nouveau nœud d'identifiant $\beta$, qui est un nouvel identifiant.
- **add_feat**$(\alpha, f = v)$ ajoute le trait $f = v$ au nœud $\alpha$. Si $\alpha$ contient déjà un trait de nom $f$, ce trait est remplacé par le nouveau trait.
- **copy_feat**$(\alpha, \beta, f)$ copie le trait nommé $f$ du $\alpha$ vers le nœud $\beta$. Si $\beta$ contient déjà un trait nommé $f$, ce trait est remplacé par le trait de $\alpha$. En particulier, si $\alpha$ ne contient pas de trait nommé $f$ et si $\beta$ contient un trait nommé $f$, le trait $f$ de $\beta$ est supprimé.

Notons que les commandes définissent une fonction partielle sur les graphes : la commande **add_edge**$(\alpha, \beta, \ell)$ n'est pas définie sur un graphe qui contient déjà une arête étiquetée $\ell$ de $\alpha$ vers $\beta$.

L'effet d'une séquence de commandes est la composition des effets de chaque commande. Les séquences de commandes sont supposées cohérentes avec le motif :

- **del_edge** réfère toujours à une arête décrite dans le motif et qui n'a pas été modifiée auparavant par une commande **del_edge** ou **shift_edge** ;
- chaque commande réfère seulement à des identifiants définis soit dans le motif, soit dans une commande **add_node** préalable ;
- aucune commande ne réfère à un nœud qui a été supprimé préalablement par une commande **del_node**.

Enfin, nous définissons une *règle de réécriture* comme un couple composé d'un motif et d'une séquence cohérente de commandes.

Un premier exemple de règle est donné figure 9.3, le motif étant à gauche et la séquence de commandes à droite. Cette règle appelée `PASSIVE_AUX` supprime le nœud correspondant à

l'auxiliaire de la construction passive et modifie les traits du verbe principal en conséquence. Ce dernier prend le temps et le mode de l'auxiliaire et il est marqué comme étant à la voix passive.

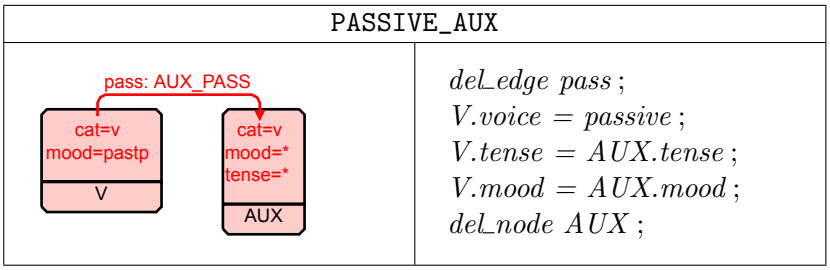



FIGURE 9.3 – Règle de l'auxiliaire passif

La règle `PASSIVE_ATS` de la figure 9.4 constitue un deuxième exemple, qui illustre la commande **add_node** et les motifs négatifs. Cette règle est utilisée pour les constructions passives dans lesquelles le sujet sémantique du verbe n'est pas réalisé. Le motif négatif écarte le cas où le verbe a un complément d'agent [16].

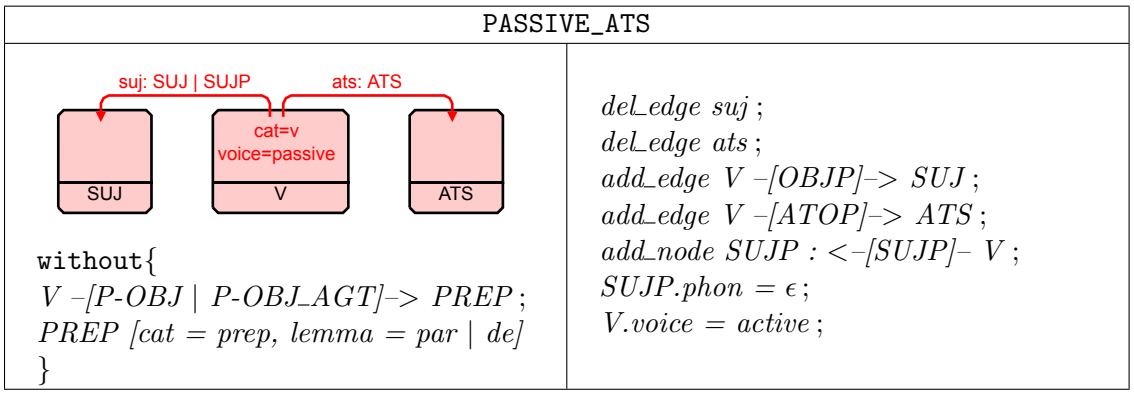



FIGURE 9.4 – Règle de reformulation du passif avec attribut du sujet sans agent

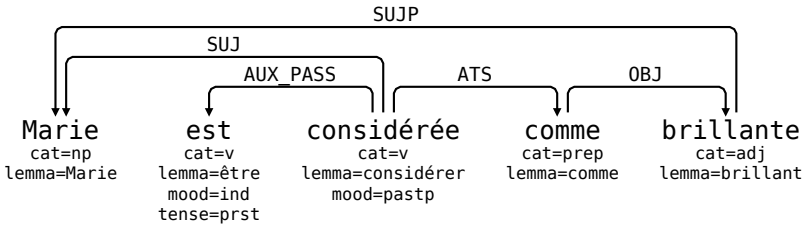
### 9.3.4 Réécriture

Nous considérons un graphe $\mathcal{G}$ et une règle de réécriture $r = (\mathcal{P}, [c_1, \ldots, c_k])$. Nous disons que $\mathcal{G}'$ est obtenu à partir de $\mathcal{G}$ en un pas de réécriture par la règle $r$, que nous notons $\mathcal{G} \longrightarrow_r \mathcal{G}'$, s'il existe un morphisme de filtrage $(\phi, \psi) : \mathcal{M} \to \mathcal{G}$ et que $\mathcal{G}'$ est obtenu à partir de $\mathcal{G}$ en appliquant la composition des commandes $c_k \circ \ldots \circ c_1$.

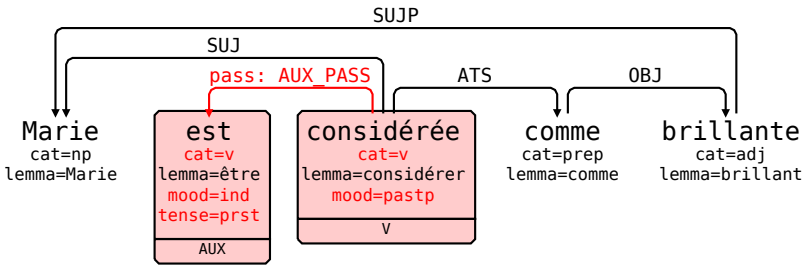
Illustrons maintenant deux pas de réécriture avec les deux règles précédentes. Considérons la figure 9.5. Le premier graphe est une structure de dépendance syntaxique pour la phrase *« Marie est considérée comme brillante »*. Le deuxième graphe montre l'identification du motif de la règle de réécriture `PASSIVE_AUX` et le troisième le résultat de l'application de cette règle. Le

16. Le motif négatif est plus complexe qu'il ne devrait être car le complément d'agent est noté *P-OBJ_AGT* dans le guide d'annotation mais marqué *P-OBJ* dans le corpus.

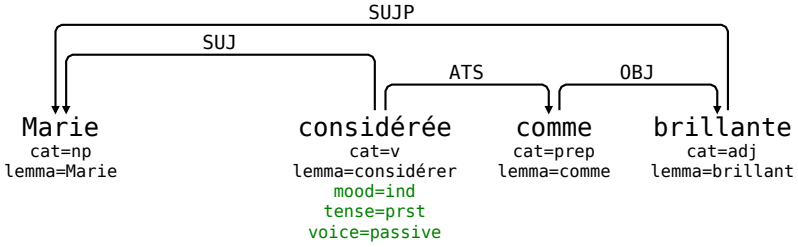
*structure initiale*



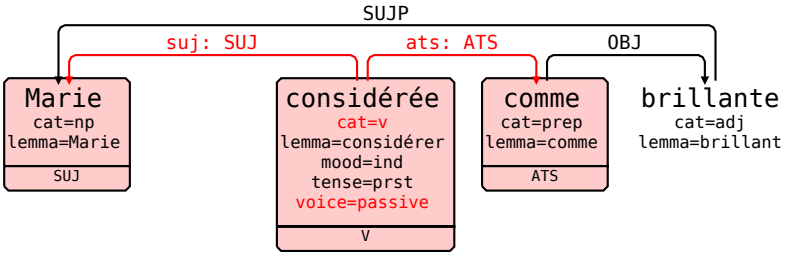

*identification du motif de la règle* ***PASSIVE_AUX***




*réécriture de la règle* ***PASSIVE_AUX***




*identification du motif de la règle* ***PASSIVE_ATS***




*réécriture de la règle* ***PASSIVE_ATS***

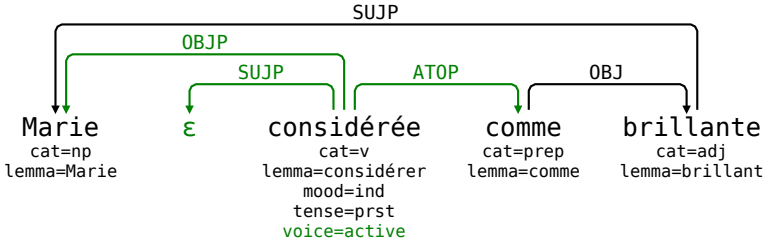



FIGURE 9.5 – Deux pas de réécriture pour *« Marie est considérée comme brillante. »*

quatrième graphe montre l'identification, sur le résultat de la réécriture précédente, du motif de la règle `PASSIVE_ATS`, et le cinquième graphe montre le résultat de l'application de cette règle.

### 9.3.5 Modules, formes normales et programmes

Un *module* est un ensemble de règles de réécriture.

Pour un module $\mathcal{D}$, nous disons que $\mathcal{G}'$ est une *$\mathcal{D}$-forme normale* du graphe $\mathcal{G}$ s'il existe une séquence de pas de réécriture par des règles $r_1, \ldots, r_k \in \mathcal{D}$ entre $\mathcal{G}$ et $\mathcal{G}'$ :

$$\mathcal{G} \longrightarrow_{r_1} \mathcal{G}_1 \longrightarrow_{r_2} \mathcal{G}_2 \ldots \longrightarrow_{r_k} \mathcal{G}'$$

et si aucune règle de $\mathcal{D}$ ne peut être appliquée à $\mathcal{G}'$.

Enfin, un *programme* de réécriture est une séquence de modules de réécriture.

## 9.4 Des graphes syntaxiques aux graphes de dépendance sémantique

La réécriture de graphes permet de procéder pas à pas à la transformation d'un graphe syntaxique en un graphe sémantique, en écrivant une règle de réécriture pour chaque règle linguistique. Alors que l'effet de chaque règle est local, le regroupement des règles en modules permet un meilleur contrôle sur l'effet global de l'ensemble des règles. Dans cette section, nous présentons à titre d'exemple un programme de réécriture de graphes, donc une séquence de modules, qui couvre une partie significative de la grammaire du français.

### 9.4.1 Un programme de réécriture pour le calcul de la sémantique

Nous laissons de côté les causatives, les propositions clivées, la coordination, les énumérations, les comparatives et les ellipses, mais ces phénomènes peuvent être traités en ajoutant d'autres modules de réécriture. Le programme de réécriture que nous utilisons est composé d'environ 850 règles au total (dont 757 pour les cadres verbaux), réparties en 21 modules qui sont appliqués dans l'ordre suivant :

1. *INIT* : applique des pré-traitements pour uniformiser les informations présentes dans le graphe syntaxique (ajout d'un temps aux infinitifs, suppression des complémenteurs pour les objets directs) ;
2. *TENSE* : calcule les temps composés des verbes et supprime les nœuds des auxiliaires de temps ;
3. *PASSIVE-VOICE* : calcule la voix passive ou moyenne et supprime les nœuds des auxiliaires de voix ; ce module contient la règle `PASSIVE_AUX` présentée précédemment ;
4. *ACTIVE-VOICE* : affecte la voix active aux verbes qui ne se sont pas vus affecter une voix passive ou moyenne ;

5. *SPLIT-PRO* : dissocie chaque pronom (personnels clitiques, relatifs, interrogatifs) représentant un complément indirect en deux nœuds, l'un pour la préposition et l'autre pour l'objet de cette préposition (ex : *« en »* est remplacé par *« de »* + $\epsilon$) ;
6. *REPET-IL* : supprime les pronoms personnels sujets répétitions de groupes nominaux ;
7. *REFORMUL* : remplace les reformulations, qui sont des transformations régulières (passif, moyen, impersonnel), par la forme active canonique qui leur correspond ; ce module contient la règle `PASSIVE_ATS` présentée précédemment ;
8. *EXTRA-NODES* : crée des nœuds vides pour attribuer un sujet aux adjectifs et verbes qui n'en ont pas et un déterminant aux syntagmes nominaux qui n'en ont pas ;
9. *DET* : traduit les déterminants en quantificateurs généralisés, dont la restriction contient au minimum la tête du syntagme nominal ;
10. *V-FRAME* : repère les cadres verbaux (un verbe, ses affixes et ses compléments), remplace le verbe par le prédicat correspondant et crée les liens prédicat - argument ; les nœuds correspondant aux arguments optionnels non réalisés syntaxiquement sont créés ; ces règles sont ancrées lexicalement ;
11. *AFFIXES* : pour les verbes qui n'ont pas été capturés dans un cadre verbal et réécrits par *V-FRAME*, intègre leurs affixes aux verbes ; ces règles ne sont pas ancrées, elles s'appliquent par défaut ;
12. *SEM-MODAL* : pour les verbes qui n'ont pas été capturés dans un cadre verbal et réécrits par *V-FRAME*, calcule leurs arguments sémantiques ;
13. *SEM-ARGV* : transforme le sujet des verbes [17] en leur premier argument sémantique ; pour les verbes qui n'ont pas été capturés dans un cadre verbal et réécrits par *V-FRAME*, calcule leurs autres arguments sémantiques en utilisant des règles par défaut ;
14. *SEM-ARGA* : calcule les arguments sémantiques des adjectifs, quelle que soit leur fonction syntaxique, sans fixer leur mode de combinaison ;
15. *MODIF-PART* : calcule les arguments sémantiques des modificateurs « particuliers » (adverbes non scopaux, négations, adjectifs non intersectifs) et fixe leur mode de combinaison avec ces arguments ;
16. *MODIF* : calcule les arguments sémantiques des modificateurs dans le cas général ;
17. *RESTR-APP* : marque les subordonnées relatives comme restrictives ou appositives ;
18. *SEM-CONJ* : traite les conjonctions de coordination et de subordination ;
19. *ANA* : fusionne les anaphores résolues avec leur antécédent ;
20. *SEM-COMB* : détermine le mode de combinaison syntaxique entre les prédicats, selon des règles par défaut ;
21. *CLEAN-SEM* : nettoie la structure sémantique finale (supprime les ponctuations).

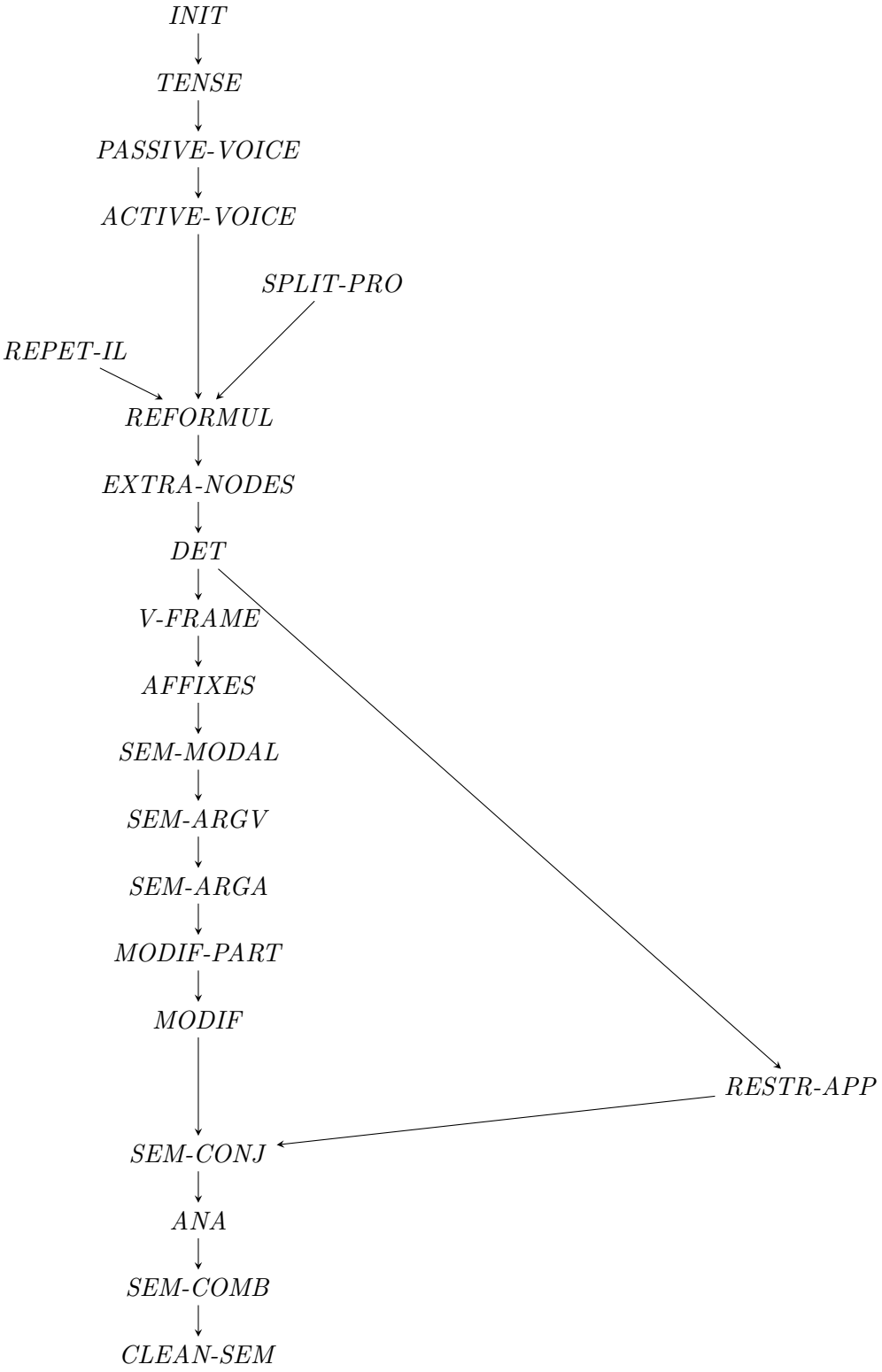


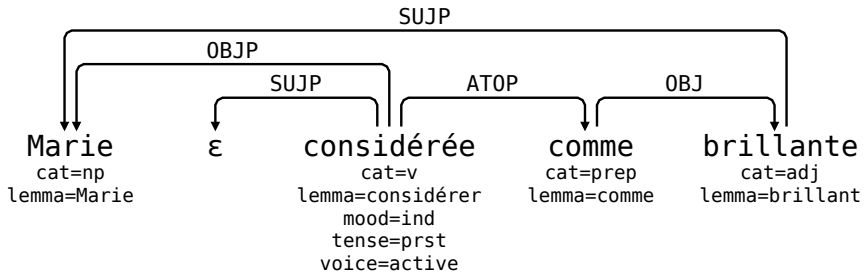

FIGURE 9.6 – Dépendances entre les modules

Les dépendances entre modules qui reflètent les hypothèses implicitement contenues dans les règles, sont représentées par le graphe de la figure 9.6. Plutôt que de détailler le contenu de chacun des 21 modules du programme précédent, nous allons donner une idée du déroulement du programme en poursuivant la réécriture de notre exemple.

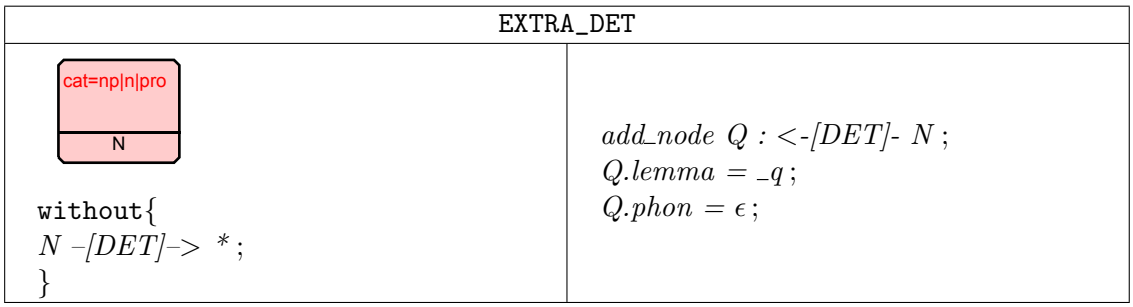
17. Sauf les sujets impersonnels qui sont capturés par les règles de *V-FRAME*.

### 9.4.2 Suite de l'application du programme à un exemple

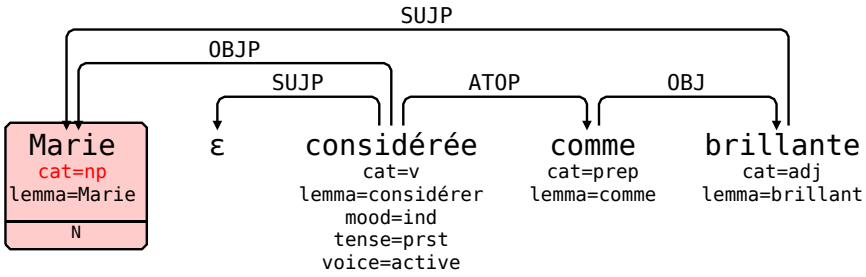
Reprenons l'exemple *« Marie est considérée comme brillante »*, que nous avons commencé à réécrire à la section précédente. Pendant l'application des 7 premiers modules, de *INIT* à *REFORMUL*, deux règles ont été appliquées : `PASSIVE_AUX`, dans le module *PASSIVE-VOICE*, et `PASSIVE_ATS`, dans le module *REFORMUL*. À cette étape du programme de réécriture, la structure associée à notre exemple est la suivante :

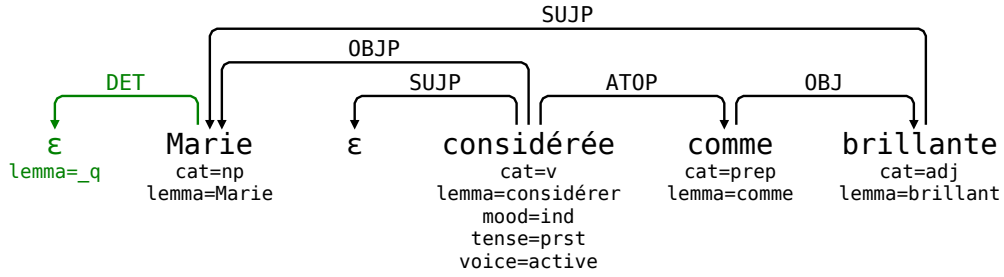



Ce graphe correspond approximativement à la paraphrase *« on considère Marie comme brillante »*. Le module suivant *REFORMUL* est *EXTRA-NODES*. Ce module contient une règle, `EXTRA_DET`, qui ajoute un déterminant par défaut aux syntagmes nominaux qui n'en ont pas, afin de garantir que tout syntagme nominal aura un quantificateur dans la structure sémantique.

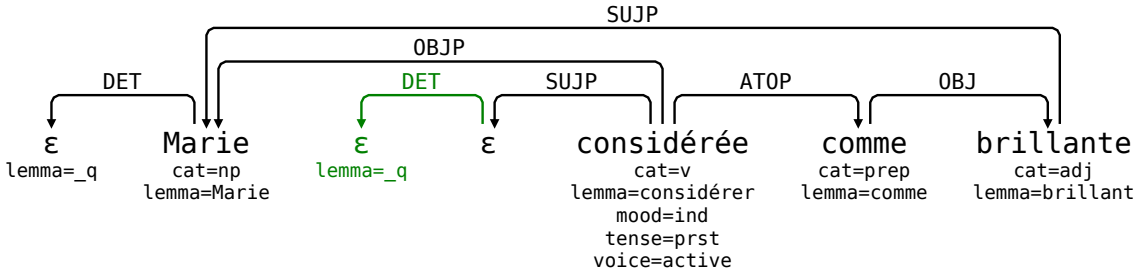



L'application de cette règle a l'effet suivant :

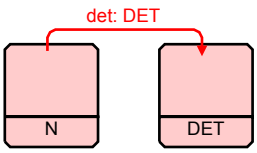




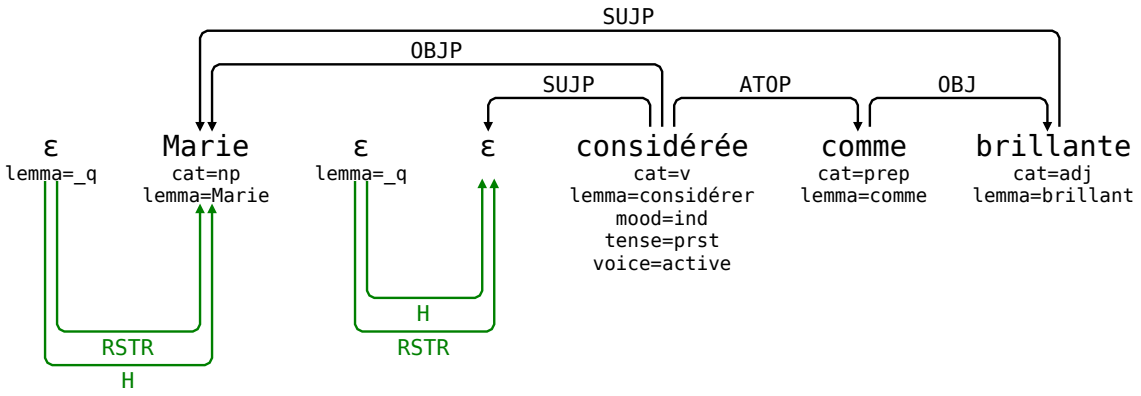

Dans ce même module, une règle similaire ajoute un déterminant par défaut aux nœuds vides, produisant la structure ci-dessous :




Le module suivant *EXTRA-NODES* est *DET*. Ce module remplace les déterminants par les quantificateurs généralisés correspondants. Il contient une règle générique, appelée `QUANT`, qui supprime la relation DET entre la tête du syntagme nominal et le déterminant, et désigne la tête du syntagme comme le cœur de la restriction du quantificateur généralisé correspondant au déterminant.

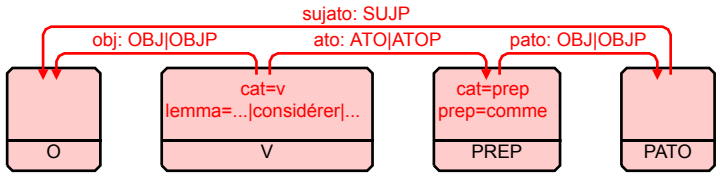
| QUANT | |
|---|---|
|   | *del_edge det* ;<br>*add_edge DET -[RSTR]-> N* ;<br>*add_edge DET -[H]-> N* ; |

Cette règle s'applique deux fois sur notre structure, ce qui produit le graphe suivant :




Le module suivant *DET* est *V-FRAME*, qui contient une règle pour chaque cadre verbal. Notre exemple contient un verbe avec un objet et un attribut de l'objet adjectival introduit par « comme ». La règle correspondante, `V-OBJ-ATO-comme`, est ancrée lexicalement : le nœud V du motif contient *lemma=...|considérer|...*. La règle correspondante est la suivante :

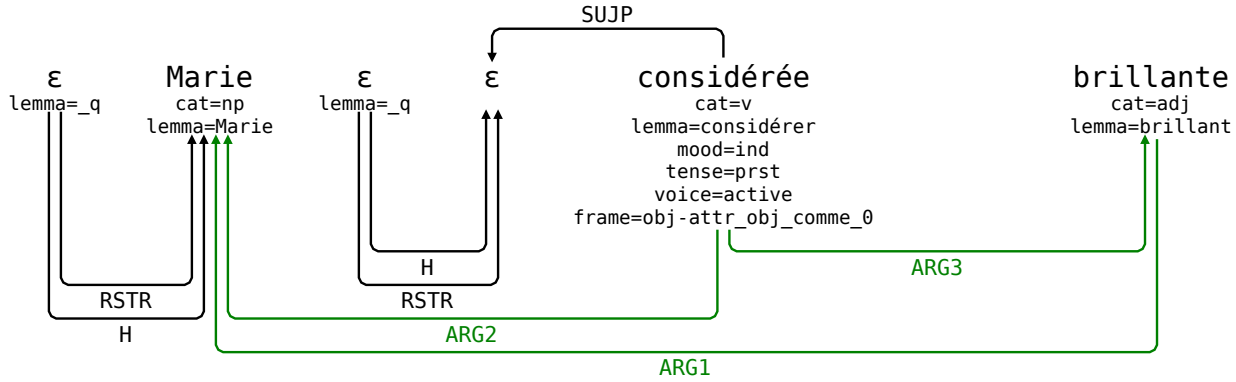
**V-OBJ-ATO-comme**



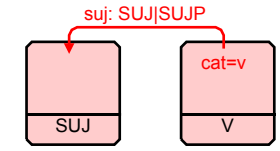

without{
*V [frame=*]* ;
}
without{
*V -[A-OBJ|DE-OBJ|P-OBJ|ATS|A-OBJP|DE-OBJP|P-OBJP]-> ** ;
}

*V.frame=obj-attr_obj_comme_0* ;
*del_edge ato* ;
*del_edge pato* ;
*del_node PREP* ;
*add_edge V -[ARG3]-> PATO* ;
*del_edge obj* ;
*add_edge V -[ARG2]-> OBJ* ;
*add_edge PATO -[ARG1]-> OBJ* ;

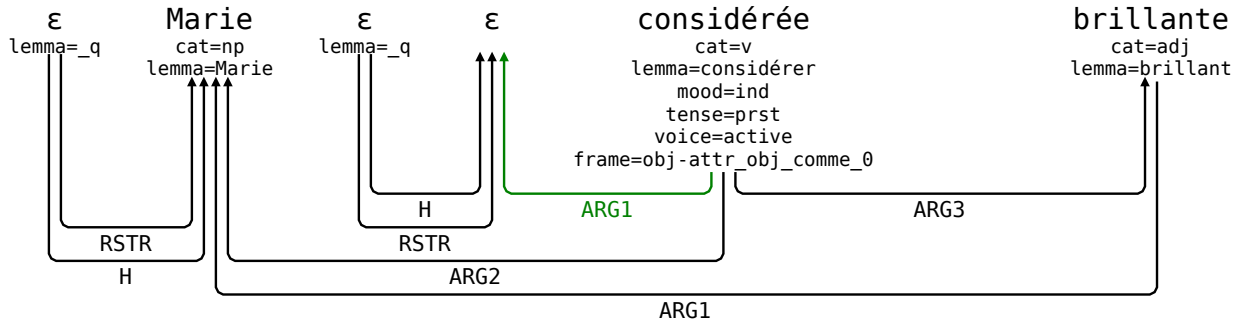
L'application de cette règle produit le graphe suivant :




Les modules suivants sont *AFFIXES* et *SEM-MODAL*. Notre exemple ne contenant ni affixe ni verbe modal, aucune règle de ces modules ne s'applique. Ensuite vient le module *SEM-ARGV*, qui contient notamment une règle, `SEMV_SUJ`, qui réécrit le sujet de tous les verbes en le premier argument sémantique du prédicat verbal.

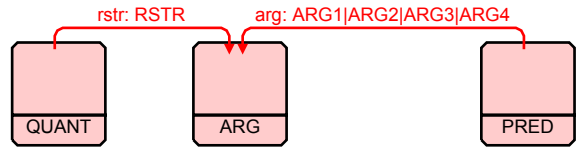
| SEMV_SUJ | |
|---|---|
|  <br>without{<br>*V -[ATS]-> * ;*<br>} | *del_edge suj ;*<br>*add_edge V -[ARG1]-> SUJ ;* |

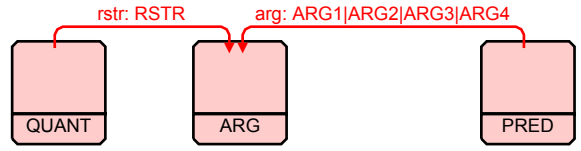
L'application de cette règle produit le graphe suivant :




Sur ce graphe, aucun des modules *SEM-ARGA*, *MODIF-PART*, *MODIF*, *RESTR-APP*, *SEM-CONJ* et *ANA* ne contient de règle qui peut être appliquée. Ensuite vient le module *SEM-COMB*, qui complète la structure avec les informations de combinaison sémantique manquantes. Les règles de ce module sont génériques. Elles font l'hypothèse que tous les liens qui dénotent une

relation flottante (H) ont été positionnés par les éléments scopaux correspondants, et que tous les modes de combinaison particuliers (comme les adjectifs non intersectifs, qui se combinent en H, ou les subordonnées relatives restrictives, qui se combinent en EQ) ont été fixés directement par les éléments correspondants. Le mode de combinaison est déterminé de la façon (résumée) suivante : si l'argument est le cœur de la restriction d'un quantificateur (c'est-à-dire la tête d'un syntagme nominal), le mode de combinaison est NEQ, sinon c'est EQ. Le module *SEM-COMB* contient donc deux règles, ADD_NEQ et ADD_EQ. L'application de ces deux règles produit la structure DMRS de

| ADD_NEQ | |
|---|---|
|  <br>without{<br>*PRED -[EQ\|NEQ\|H]-> ARG* ;<br>} | *add_edge PRED -[NEQ]-> ARG* ; |

| ADD_EQ | |
|---|---|
|  <br>without{<br>*PRED -[EQ\|NEQ\|H]-> ARG* ;<br>} | *add_edge PRED -[EQ]-> ARG* ; |

la figure 9.7.

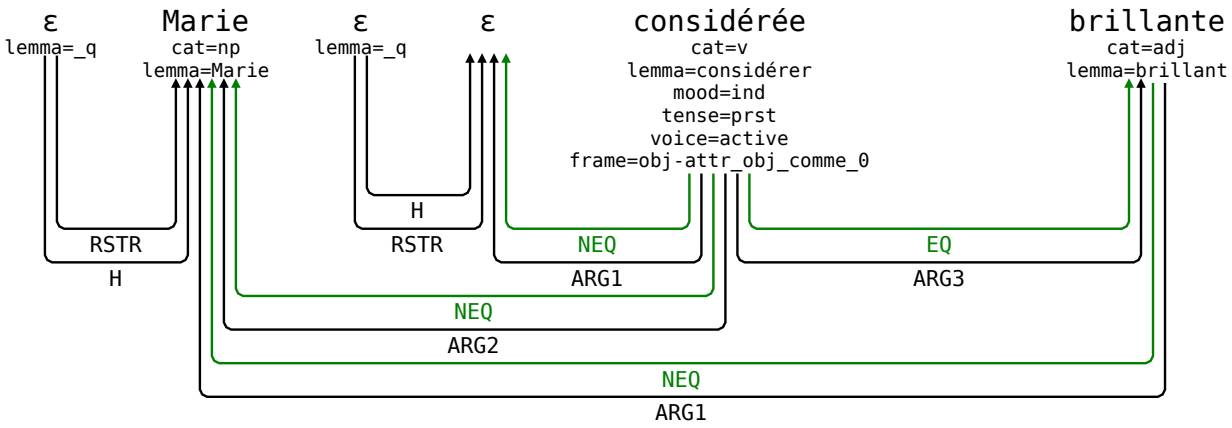



FIGURE 9.7 – Structure DMRS pour *« Marie est considérée comme brillante »*

### 9.4.3 Utilisation de Dicovalence pour les prédicats verbaux

Lorsque nous avons présenté le format DMRS, nous avons souligné le principal problème des structures DMRS standard, à savoir leur sous-détermination de l'arité des prédicats. Cette sous-détermination empêche de convertir directement une structure DMRS en une structure MRS. Nous avons donc proposé de construire des DMRS dans lesquelles les arguments optionnels d'un prédicat sont réalisés par des nœuds vides. Pour savoir quels nœuds vides il faut créer et dans quelles conditions, de l'information lexicale est nécessaire. Pour les verbes, nous proposons ici d'extraire cette information du lexique Dicovalence.

Le lexique des verbes du français *Dicovalence* [vdEM03] [18] contient les cadres valenciels de plus de 3 700 verbes simples du français, pour plus de 8 000 entrées au total. Dans Dicovalence, chaque verbe a autant d'entrées distinctes dans le lexique que de cadres valenciels distincts pour chaque sens de ce verbe. Par exemple, le verbe *« passer »* a 22 entrées de sens distincts (les arguments optionnels sont marqués entre parenthèses) :

1. montrer : passer un film (à des amis),
2. écouler du temps quelque part ou en faisant quelque chose : passer des vacances (en montagne) (à faire de la randonnée),
3. filtrer : passer le café,
4. traverser : passer (la porte),
5. sauter, ignorer : passer (un tour de jeu),
6. ...

18. Version 2.

Cela signifie que la phrase *« Jean passe le pont »* peut en théorie correspondre aux 5 sens de *« passer »* ci-dessus. Ces 5 entrées spécifient 4 cadres valenciels différents (les arguments optionnels sont marqués entre parenthèses) :

1. SUJ V OBJ (A-OBJ),
2. SUJ V OBJ (P-OBJ_LOC) (P-OBJ),
3. SUJ V OBJ,
4. SUJ V (OBJ) [x2].

Pour chacun de ces 4 cadres, il est possible d'écrire une règle de réécriture. Lorsqu'elle s'applique à *« Jean passe le pont »*, la règle correspondant au premier cadre fixe l'arité du prédicat *« passer »* à 3 arguments et crée un nœud vide pour l'A-OBJ non réalisé. La règle correspondant au deuxième cadre fixe l'arité du prédicat à 4 arguments et crée deux nœuds vides, un pour le P-OBJ_LOC et un autre pour le P-OBJ. Les règles correspondant aux troisième et quatrième cadres fixent l'arité du prédicat à 2 arguments. On obtient ainsi au moins 3 structures DMRS différentes pour *« Jean passe le pont »*.

Les plus de 8 000 entrées de Dicovalence décrivent actuellement 757 configurations syntaxiques possibles :

- verbe intransitif avec affixe *« en »* (ex : en découdre, en finir, en jeter...),
- verbe avec un OBJ avec affixe *« en »* (ex : en croire),
- verbe avec un A-OBJ avec affixe *« en »* (ex : en coûter, en remontrer...),
- verbe avec un A-OBJ, un DE-OBJ optionnel non réalisé et un affixe *« en »* (ex : en vouloir),
- verbe avec un A-OBJ, un DE-OBJ optionnel réalisé et un affixe *« en »* (ex : en vouloir),
- ...

Pour chaque configuration syntaxique, nous générons automatiquement une règle qui est ancrée par tous les verbes qui présentent cette configuration. Actuellement, nous générons donc automatiquement 757 règles ancrées lexicalement par des verbes. Chacune de ces règles transforme le verbe en prédicat d'arité fixe, et ses arguments syntaxiques en arguments sémantiques, à l'exception du sujet qui est géré de la même façon pour tous les verbes. Ces 757 règles constituent le module *V-FRAME*.

## 9.5 Expérimentation

Il est très difficile d'évaluer le résultat produit par notre programme de réécriture car il n'existe pas de corpus sémantique de référence pour le français. Nous avons donc réalisé une expérimentation, en utilisant une suite de tests qui est à la fois suffisamment restreinte pour être validée manuellement et suffisamment étendue pour couvrir une grande variété de phénomènes linguistiques. Comme nous utilisons une extension du format d'annotation en dépendances de surface du FTB, nous avons construit notre suite de tests à partir des exemples du guide d'annotation du FTB en dépendances [CCF11]. Par nature, un guide d'annotation essaie de couvrir une grande variété de phénomènes avec un ensemble restreint d'exemples.

La dernière version en date[19] de ce guide contient 186 phrases d'exemple. Dans notre programme de réécriture, nous laissons de côté les clivées, les coordinations et les comparatifs. Nous laissons également de côté un petit ensemble de phrases exotiques pour lesquelles nous ne parvenons pas à écrire de structure syntaxique satisfaisante. Au final, notre expérimentation concerne 116 phrases en français. Pour obtenir les structures syntaxiques au format FTB enrichi, nous avons utilisé la sortie en dépendances de l'analyseur LEOPAR, obtenue en appliquant la procédure d'extraction des dépendances de [MGP10]. La sortie en dépendances de LEOPAR ne correspond pas exactement au format que nous avons défini. Nous avons donc écrit un programme de réécriture de graphes pour convertir la sortie de LEOPAR en une structure syntaxique dans notre format. Ensuite, chaque structure de dépendances syntaxiques a été vérifiée manuellement et corrigée si nécessaire. Enfin, nous avons appliqué le programme de réécriture que nous avons présenté dans la section précédente.

Pour toutes ces phrases, notre système produit au moins une forme normale. Même si la DMRS est un formalisme de représentation sémantique sous-spécifiée, notre système peut produire plusieurs représentations sémantiques pour une même structure syntaxique. Ce cas de figure survient pour les verbes très ambigus, comme *« passer »* que nous avons évoqué dans la section précédente, mais également pour les subordonnées relatives dont la lecture est ambiguë, entre relative appositive et relative restrictive. Notre système surgénère pour certaines phrases car nous manquons d'information lexicale pour différencier, par exemple, les adverbes scopaux et non-scopaux.

Nous présentons ici le résultat de la réécriture pour deux (parties de) phrases.

## 9.6 Discussion

### 9.6.1 Limites et perspectives

Le programme de réécriture présenté dans ce chapitre pour calculer une représentation sémantique à partir d'une structure syntaxique peut être amélioré sous de nombreux aspects.

Le premier point qui peut être amélioré concerne la précision et la couverture de ce programme. Le programme actuel utilise essentiellement le lexique des verbes Dicovalence. Il manque des ressources équivalentes pour les autres catégories lexicales : adjectifs, adverbes, noms, déterminants...

Ensuite, notre traitement de certains phénomènes pourrait être affiné. Nos règles ne tiennent par exemple pas compte de l'ordre des mots, ce qui est problématique pour déterminer la portée respective de certains adverbes [BG08] et des quantificateurs. Ce problème est amplifié par notre traitement des reformulations. Les reformulations sont des formes syntaxiques différentes qui partagent la même sémantique vériconditionnelle et définissent donc une classe d'équivalence [vdEM03]. Dans notre système, nous choisissons un représentant de cette classe d'équivalente (la forme canonique active) et réécrivons toutes les autres formes de la classe d'équivalence

19. Version 1.2, mai 2011.

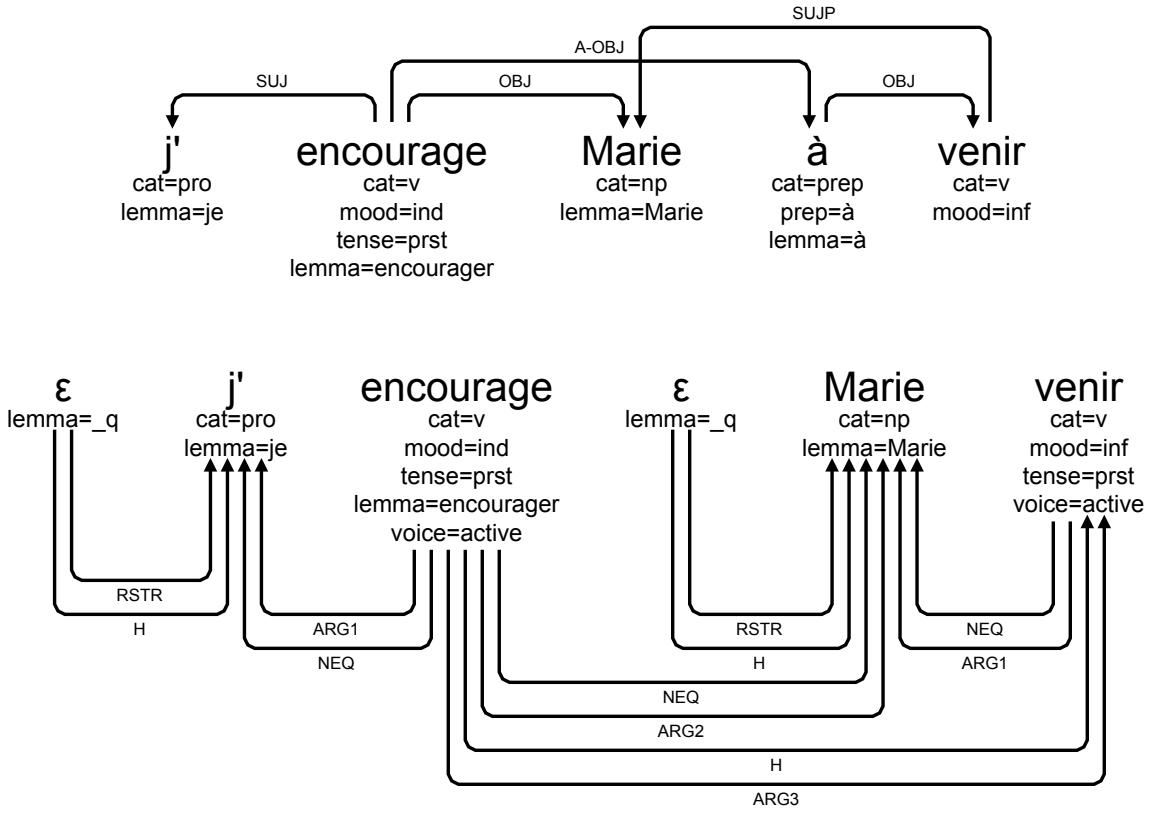



FIGURE 9.8 – [057] *« J'encourage Marie à venir. »*

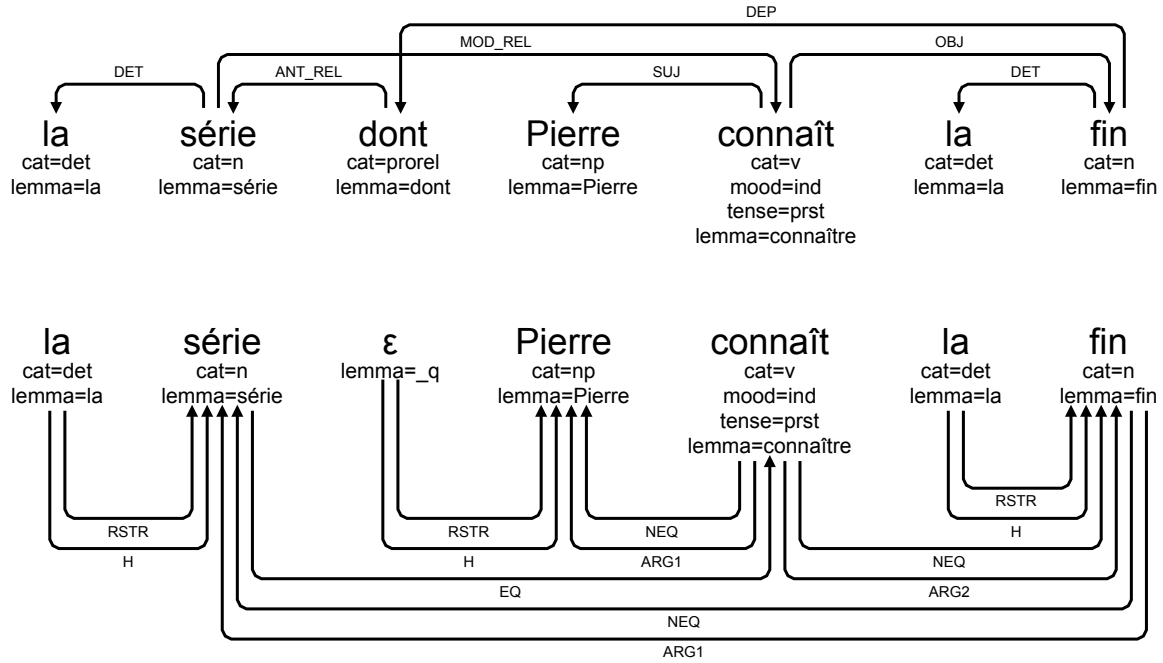



FIGURE 9.9 – [106] *« La série dont Pierre connaît la fin »*

vers ce représentant. L'équivalence sémantique réelle des reformulations est cependant controversée, entre autres parce que les reformulations ne semblent pas induire les mêmes lectures préférentielles en présence d'arguments quantifiés [Cho65].

Enfin, les formalismes de représentation sémantique que nous avons adoptés, la MRS et la DMRS, présentent eux-mêmes des limites. Leur algèbre de composition ne permet pas de modéliser l'intensionalité. La numérotation des arguments des prédicats reste très syntaxique et ne permet aucune généralisation sémantique. Les structures sémantiques produites restent pauvres par rapport, par exemple, aux graphes sémantiques de la TST qui utilisent les fonctions lexicales [Mel96].

D'un point de vue calculatoire, de nombreux points restent à étudier concernant le calcul de réécriture de graphes que nous avons présenté. Un point particulièrement intéressant concerne l'inférence automatique d'invariants sur la structure du graphe à différentes étapes de l'application d'un programme de réécriture.

### 9.6.2 Travaux apparentés

Nous comparons maintenant notre approche aux propositions antérieures d'utilisation de la réécriture dans des systèmes de TAL, à l'interface entre syntaxe et sémantique. Nous commentons ici quelques propositions qui semblent les plus proches de nos travaux.

- Crouch et King [CK06] calculent des structures sémantiques à partir de f-structures LFG à l'aide d'un calcul de réécriture de termes, dont les règles sont strictement ordonnées. Le calcul employé ne permet pas de gérer les ambiguïtés de portée de quantificateurs.
- Spreyer et Frank [SF05] utilisent le même calcul de réécriture de termes que Crouch et King pour produire des structures MRS à partir du corpus annoté en dépendances *TIGER Dependency Bank* pour l'allemand.
- Bedaride et Gardent [Bed10] proposent un système de réécriture de graphes appliqué selon une stratégie d'ordre total sur les règles de réécriture. Cet ordre total est construit de façon à appliquer les règles de motif plus spécifique avant les règles de motif plus général. Ainsi, la règle du passif long (avec agent) doit être appliquée avant la règle du passif court (sans agent). L'utilisation conjointe des motifs négatifs et des modules nous permet d'éviter une telle rigidité. De plus, le système de Bedaride et Gardent est confluent et ne produit donc qu'une seule structure sémantique par phrase.
- Chaumartin et Kahane [CK10] proposent d'apprendre automatiquement, à partir d'exemples, des règles d'interface syntaxe - sémantique qui mettent en correspondance un fragment d'arbre de dépendance et un fragment de graphe sémantique de la TST. Cette approche est plus automatisée que la nôtre et l'intégration de ressources, notamment de lexiques et d'analyseurs, est plus aisée. Cette proposition diffère de la nôtre par le modèle de réécriture de graphes utilisé et par les structures sémantiques produites (graphes sémantiques de la TST d'un côté, structures DMRS de l'autre).

## Conclusion

Dans ce chapitre, nous avons défini un calcul de réécriture de graphes modulaire à base de commandes et nous avons proposé un programme de réécriture qui produit des structures sémantiques sous-spécifiées au format DMRS à partir d'arbres de dépendance syntaxique de surface enrichis de relations grammaticales. Nos expérimentations sur les phrases du guide d'annotation du FTB sont concluantes et montrent en tout cas la viabilité de notre approche.

La vocation principale de ce programme de réécriture de graphes était cependant d'illustrer la flexibilité de notre calcul. Ce calcul peut être utilisé pour d'autres formats d'entrée et de sortie que ceux que nous avons utilisés ; il est d'ailleurs adapté à la conversion entre différents formats. L'approche modulaire facilite le développement et la maintenance de grands ensembles de règles. Il est par exemple facile d'étendre un programme par l'ajout de nouveaux modules ou par le raffinement de modules déjà présents, pour obtenir une analyse sémantique plus précise.

Nous exploitons cette flexibilité au prochain chapitre. Nous proposons en effet un programme de réécriture qui peut être appliqué avant le programme présenté dans ce chapitre. Ce programe permet d'enrichir automatiquement des arbres de dépendance syntaxique de surface au format FTB avec les relations grammaticales définies dans notre format étendu. La composition des deux programmes de réécriture permet de produire une structure DMRS à partir d'un arbre de

dépendance syntaxique de surface.

# Chapitre 10

# Enrichissement de structures de dépendance syntaxique de surface

Dans le chapitre précédent, nous avons présenté un calcul de réécriture de graphes et nous avons montré comment calculer, au moyen de ce calcul, une représentation sémantique d'une phrase au format DMRS à partir d'une structure de dépendance syntaxique de surface au format FTB enrichie de relations grammaticales. Nous poursuivons cette ligne de recherche dans ce chapitre en proposant une méthode qui ajoute aux structures de dépendance syntaxique de surface du FTB les relations grammaticales nécessaires et nous appliquons cette méthode au French Treebank. Ces règles, en combinaison avec celles du chapitre 9, permettent d'obtenir automatiquement une structure sémantique DMRS à partir d'une structure de dépendance de surface annotée selon le schéma du FTB.

Les relations grammaticales que nous ajoutons permettent de représenter les antécédents d'anaphores syntaxiquement déterminés et les arguments profonds syntaxiquement déterminés des verbes à l'infinitif, des participes et des adjectifs. Dans la section 10.1, nous revenons sur cette extension du schéma d'annotation à l'aide d'un exemple. Dans la section 10.2, nous donnons l'intuition de l'interdépendance de ces relations grammaticales, encore une fois sur un exemple. Nous décrivons ensuite, dans la section 10.3, le programme de réécriture de graphes que nous utilisons pour produire, à partir d'une structure de dépendance dans le schéma d'annotation du FTB, une structure dans le schéma étendu. Ce programme de réécriture de graphes est constitué d'un ensemble de règles grammaticales et lexicales. Les règles lexicales utilisent une information de contrôle extraite du lexique des verbes français *Dicovalence*. Les différents modules sont présentés en section 10.4. Dans la section 10.5, nous validons expérimentalement notre programme de réécriture en l'appliquant au FTB annoté en dépendances syntaxiques de surface. Enfin, dans la section 10.6, nous discutons des limites et perspectives de ce programme de réécriture.

## 10.1 Position du problème

Les analyseurs syntaxiques profonds sont capables de fournir deux types de relations qui ne font pas partie du schéma d'annotation en dépendances de surface du FTB mais qui sont utiles au calcul de la sémantique, profonde ou superficielle [CFL+02, CHS02, RCS09, BFOZ11]. Le premier type de relations concerne les antécédents d'anaphores syntaxiquement déterminés : pronoms relatifs, pronoms personnels réfléchis et pronoms personnels répétitions de sujets. Le deuxième type de relations porte sur les arguments profonds des adjectifs et des verbes à l'infinitif, au participe passé ou au participe présent, lorsque ces arguments se trouvent dans la phrase et qu'ils sont déterminés par la syntaxe.

L'exemple donné figure 10.1 contient de telles relations manquantes. La phrase est annotée par, en noir, les dépendances syntaxiques du guide d'annotation en dépendances de surface du FTB [CCDG09, CCF11] [20] et, en vert, les relations à ajouter.

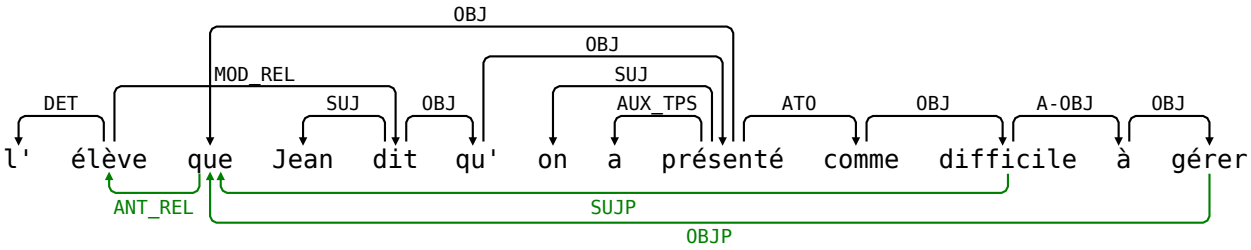



FIGURE 10.1 – Exemple d'enrichissement des dépendances de surface

### 10.1.1 Antécédents d'anaphores syntaxiquement déterminés

Trois types de pronoms présentent la particularité d'être des anaphores dont l'antécédent est présent dans la phrase et syntaxiquement déterminé. Ce sont les pronoms relatifs, les pronoms personnels réfléchis et les pronoms personnels répétitions de sujets.

Si les pronoms personnels réfléchis et répétitions de sujets ne posent pas de problème, retrouver l'antécédent d'un pronom relatif dans la structure de dépendance de surface s'avère parfois complexe. Sur notre exemple, il faut remonter une chaîne de trois dépendances OBJ, qui va du pronom relatif *« que »* jusqu'à la tête de la relative *« dit »* (en passant par *« présenté »* et *« qu' »*). De là, on retrouve l'antécédent *« élève »* en remontant la dépendance MOD_REL. Plus généralement, les phénomènes d'extraction et d'enchâssement créent, entre les pronoms relatifs et leurs antécédents, des chaînes de dépendances dont la longueur n'est pas bornée. La composition de ces chaînes de dépendances obéit en outre à certaines contraintes portant à la fois sur le contenu des nœuds et sur les relations de dépendances de la chaîne. Au final, bien que l'antécédent des pronoms relatifs soit syntaxiquement déterminé, le calcul qui permet de retrouver cet antécédent n'est pas immédiat.

20. http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html

### 10.1.2 Arguments profonds

Dans une structure de dépendance syntaxique de surface, les verbes à l'infinitif n'ont pas de sujet. Les verbes à contrôle comme *« permettre »* fournissent cependant un sujet profond à leur infinitif complément, comme le montre l'accord de l'infinitif passé dans la phrase *« Jean permet à Marie d'**être entendue** »*.

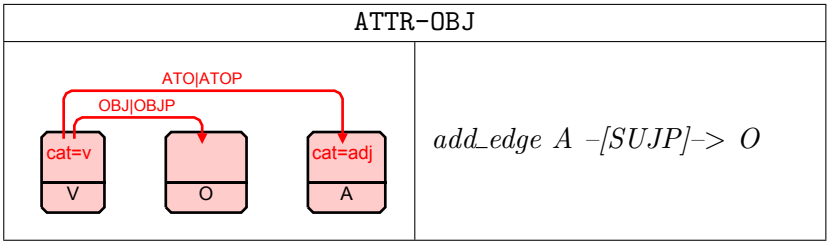
De la même façon, les participes présents et participes passés dans leur utilisation adjectivale, ainsi que les gérondifs, ont un sujet profond syntaxiquement déterminé, comme dans *« **En mangeant**, Jean fit un signe de la main »*. Remarquons que deux syntagmes adjectivaux peuvent d'ailleurs être coordonnés sans que leur tête soit de la même catégorie, comme dans *« **Livide** et **effrayé** par la situation, Jean frémit »*. Afin de traiter de façon uniforme les syntagmes adjectivaux, nous étendons donc la fonction de sujet aux adjectifs. Nous associons donc systématiquement un sujet aux syntagmes adjectivaux, que leur tête soit un participe ou un adjectif.

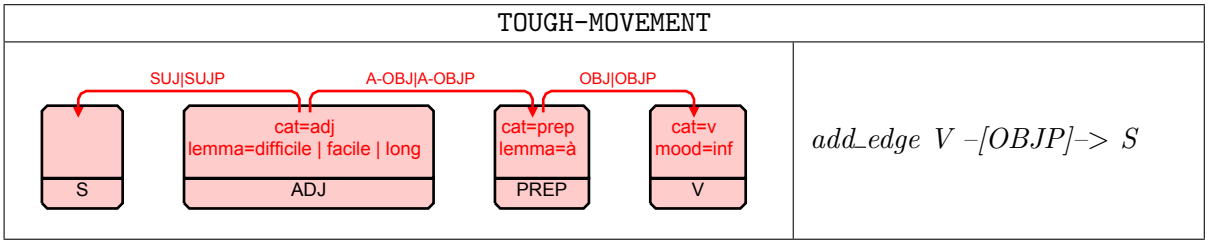
Certains adjectifs attribuent un sujet profond à leur complément infinitif, comme *« lent »* dans *« Jean est lent à **comprendre** »*. D'autres adjectifs fournissent enfin un objet profond à leurs infinitifs compléments : c'est le phénomène du *tough movement* [Rez06]. Ainsi dans l'expression *« un livre difficile à **lire** »*, l'objet de *« lire »* est *« livre »*.

Cette dernière construction est présente dans l'exemple de la figure 10.1 : *« gérer »* a pour objet profond *« que »*. Le calcul de cette relation nécessite plusieurs étapes. Tout d'abord, il faut déterminer le sujet profond de *« difficile »* ; *« difficile »* étant l'attribut de l'objet de *« présenté »*, son sujet profond est l'objet direct de ce verbe, c'est-à-dire *« que »*. Partant de cette nouvelle relation, nous pouvons marquer la relation d'objet profond entre le verbe *« gérer »* et le sujet profond de *« difficile »*.

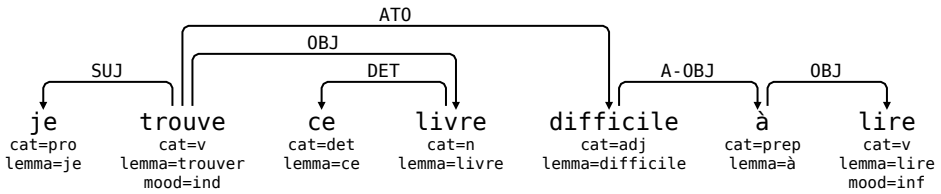
## 10.2 Exemple de réécriture

Voyons plus précisément l'interaction que nous venons d'évoquer entre les règles de calcul des arguments profonds. La figure 10.2, montre le déroulement de la réécriture sur la phrase *« Je trouve ce livre difficile à lire »*. Les deux règles de réécriture utilisées dans cet exemple sont `ATTR-OBJ` et `TOUGH-MOVEMENT`. Comme notre objectif ici est simplement d'enrichir des structures de dépendance sans les modifier, les séquences de commandes en partie droite des règles sont très restreintes. Pour la règle `ATTR-OBJ` , la séquence se réduit à la commande *add_edge A –[SUJP]–> O* qui ajoute une relation SUJP de l'attribut de l'objet à l'objet du verbe.

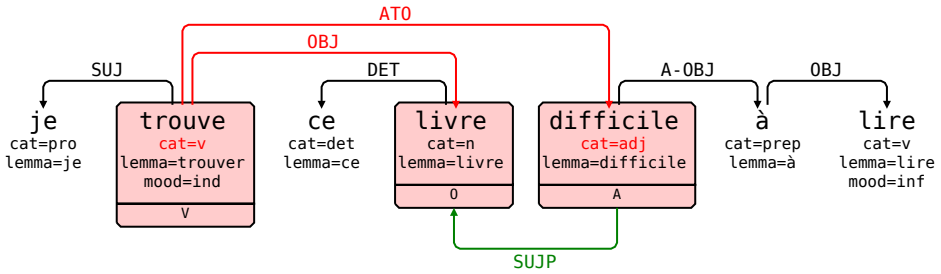




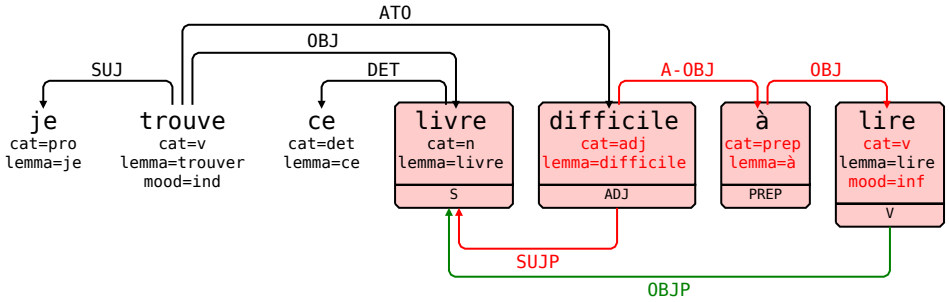

*structure initiale*




*identification du motif de la règle* ***ATTR-OBJ*** *puis réécriture*




*identification du motif de la règle* ***TOUGH-MOVEMENT*** *puis réécriture*




*structure finale*

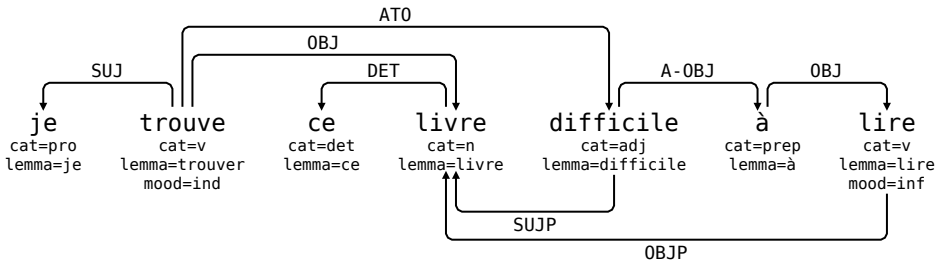



FIGURE 10.2 – *« Je trouve ce livre difficile à lire. »*

L'interaction entre les règles induit un ordre dans l'application de celles-ci, dont nous devrons tenir compte lors de la répartition des règles en modules.

## 10.3 Règles utilisées pour l'enrichissement

L'enrichissement de l'annotation de surface se fait au moyen de quatre types de règles :

- des *règles grammaticales d'actants* déterminent, sur la base d'information purement syntaxique, certains sujets d'infinitifs, de participes et d'adjectifs,
- des *règles lexicales d'actants*, extraites d'un lexique, déterminent les sujets ou objets d'infinitifs compléments de verbes ou d'adjectifs à contrôle,
- des *règles d'antécédents* déterminent les antécédents des anaphores déterminées par la syntaxe,
- des *règles de coordination* ajoutent les actants syntaxiques manquants aux conjoints.

### 10.3.1 Règles grammaticales d'actants

Les participes têtes de syntagmes adjectivaux n'ont pas de sujets explicites dans le FTB car celui-ci ne contient que des arbres de dépendances. Le sujet d'un participe a déjà une autre fonction syntaxique ; le désigner reviendrait donc à lui attribuer un deuxième gouverneur et à violer la contrainte d'arbre. Voyons quelques exemples (les participes sont mis en gras) :

*Jean arrive en* **chantant**.
**Abandonnée** *de tous, elle ne sait plus que faire.*
*Pierre sait Marie* **délaissée** *par son mari.*
*Un homme se* **présentant** *comme son frère vient d'arriver.*

La configuration grammaticale dans laquelle se trouvent ces participes permet de déterminer leur sujet.

Les adjectifs peuvent se trouver dans des constructions variées : épithète, attribut du sujet, attribut de l'objet, dislocation. Toutes ces constructions sont compatibles avec le *tough movement* [Kah03, Rez06], comme le montrent les exemples suivants :

*Je connais un livre* **difficile** *à lire.*
*Ce livre passe pour* **difficile** *à lire.*
*Je trouve ce livre* **difficile** *à lire.*
**Difficile** *à lire, ce livre n'est pas à conseiller à tout le monde.*

Dans ces phrases, la relation entre *« livre »* et *« difficile »* implique que *« livre »* est objet de *« lire »*. Attribuer un sujet profond aux adjectifs permet de créer les relations d'objet profond induites par le *tough movement* avec une seule règle, qui est indépendante de la configuration dans laquelle ces adjectifs apparaissent.

Il y a autant de règles que de constructions intégrant des syntagmes adjectivaux. Ici, nous en considérons sept : épithète, attribut du sujet (avec ou sans préposition), attribut de l'objet (avec ou sans préposition), dislocation, gérondif. La règle `ATTR-OBJ` de la figure 10.2 s'applique aux attributs de l'objet sans préposition.

### 10.3.2 Règles lexicales d'actants

#### Verbes à contrôle, à montée et modaux

Lorsqu'un infinitif est complément d'un verbe à contrôle ou à montée ou d'un verbe modal, son sujet, s'il est exprimé dans la phrase, est déterminé par ce verbe. Voyons quelques exemples où les infinitifs sont mis en gras et leur sujet profond est souligné :

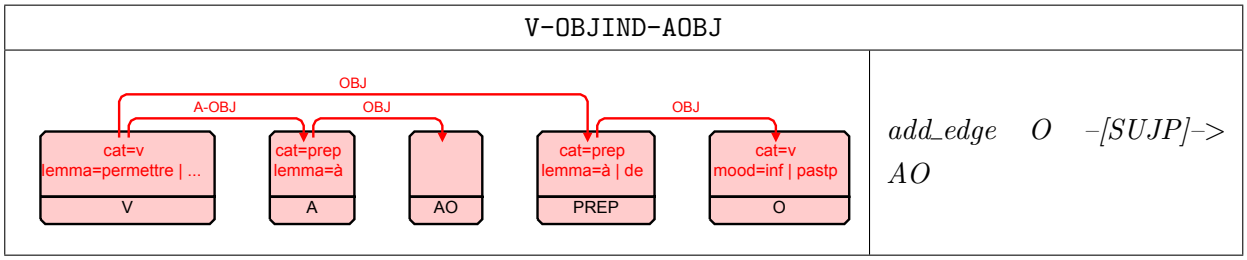
*Jean permet à Marie de* **venir**.
*Jean promet à Marie de* **venir**.
*Jean propose à Marie de* **venir**.
*Jean semble* **changer** *d'avis.*

Dans les deux premières phrases, les verbes *« permettre »* et *« promettre »* déterminent le sujet de *« venir »* : *« Marie »* pour la première et *« Jean »* pour la deuxième. La troisième phrase est ambiguë, le sujet de *« venir »* peut être *« Jean »* ou *« Marie »*. Dans la dernière phrase, *« sembler »* est un verbe à montée, l'infinitif qui le suit partage son sujet.

Les verbes à contrôle peuvent être groupés en différentes classes, selon la fonction grammaticale de l'infinitif contrôlé et la fonction grammaticale du syntagme sujet de cet infinitif. Il y a trois possibilités pour chacun de ces deux facteurs : sujet, objet direct et objet indirect, donc six classes possibles. Pour chaque classe, il y a plusieurs règles, afin de gérer l'éventuelle présence d'un complémenteur *« à »* ou *« de »* introduisant l'infinitif.

Les verbes à contrôle possèdent, dans le lexique des verbes du français *Dicovalence* [vdEM03], un ou plusieurs champs *PIVOT* qui contiennent leurs informations de contrôle. Ces informations sont extraites automatiquement de *Dicovalence* pour ancrer lexicalement les règles des verbes à contrôle, comme *« permettre »* dans l'exemple suivant :

```
VAL$    permettre: P0 P1 (P2)
NUM$    60700
FRAME$  subj:pron|n:[hum], obj:pron|n|compl|de_inf:[abs,mood:subj], ?objà:pron|n:[hum]
PIVOT$  P2/P0 [below de_inf in P1]
```




`P2/P0 [below de_inf in P1]` se lit ainsi : l'objet indirect de *« permettre »* introduit par *« à »*, noté P2 dans *Dicovalence*, est le sujet, noté P0, d'une infinitive introduite par *« de »* (de_inf) qui est l'objet direct , noté P1, de *« permettre »*. Cela se traduit dans la règle `V-OBJIND-AOBJ`

par la commande *add_edge O –[SUJP]–> AO* qui ajoute une relation SUJ de l'objet direct O vers le groupe nominal objet indirect AO.

#### Adjectifs à contrôle

Lorsqu'un infinitif est complément d'un adjectif à contrôle, son sujet ou son objet — quand il est présent dans la phrase — est déterminé lexicalement. Le *tough movement* est l'une de ces configurations. Voici quelques exemples où les infinitifs sont en gras :

*Jean est lent à* **comprendre**.
*Le livre est difficile à* **comprendre**.

Comme pour les verbes, c'est une information lexicale sur les adjectifs *« lent »* et *« difficile »* qui détermine que le sujet du premier est également le sujet de *« comprendre »*, alors que le sujet du second est l'objet direct de *« comprendre »*.

Les adjectifs à contrôle forment deux classes, selon que leur sujet est le sujet ou l'objet direct de l'infinitif contrôlé. Ces deux cas se traduisent chacun par une règle ; la règle `TOUGH-MOVEMENT` de la figure 10.2 correspond au cas où c'est l'objet de l'infinitif qui est contrôlé.

Il n'existe pas, pour les adjectifs, l'équivalent de Dicovalence pour les verbes. Nous avons donc relevé les adjectifs du FTB qui ont un argument infinitif et nous les avons classés manuellement pour ancrer les règles.

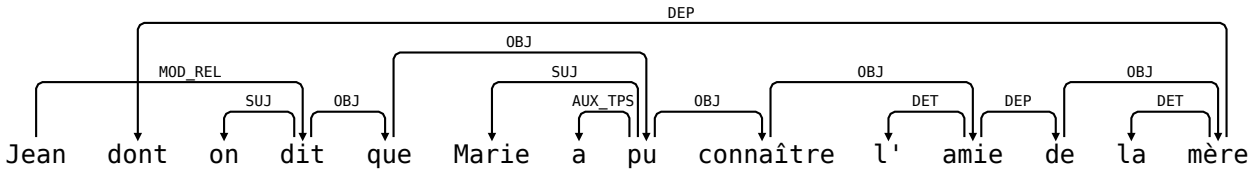
### 10.3.3 Les antécédents d'anaphores syntaxiquement déterminés

Connaître les liens anaphoriques qui sont totalement déterminés par la syntaxe permet de calculer certaines co-références sémantiques. Dans ce travail, nous en distinguons trois types.

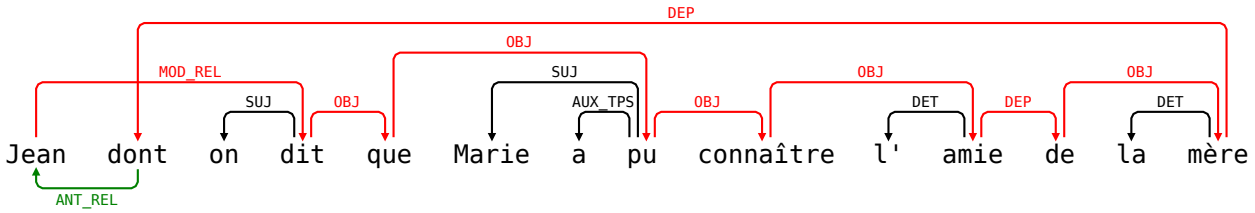
Le premier est le pronom réfléchi complément réel d'un verbe : son antécédent est le sujet de ce verbe. Par exemple, dans *« Jean ne veut pas* **se** *laver »*, l'objet direct de *« laver »* est *« se »*, qui co-réfère sémantiquement avec *« Jean »*. Une règle permet de détecter ce cas et une relation notée ANT_REFL entre le pronom et son antécédent est ajoutée.

Le second est le pronom personnel répétition d'un sujet : son antécédent est l'autre sujet du verbe. Dans *« Jean veut-***il** *manger ? »*, *« veut »* a deux sujets, dont *« il »* qui co-réfère sémantiquement avec *« Jean »*. Une relation ANT_REP est alors ajoutée.

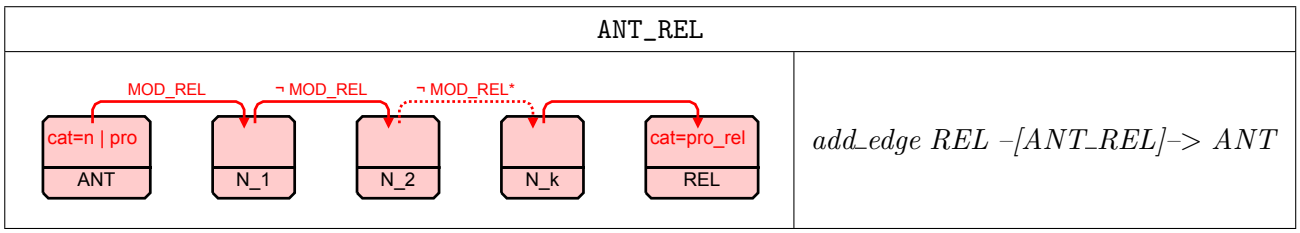
Le troisième type de lien anaphorique que nous considérons relie un pronom relatif à son antécédent. La création de ce lien, noté ANT_REL, est un problème plus complexe que les précédents. Dans le FTB, il faut pour cela remonter les dépendances depuis le pronom relatif jusqu'à la tête de la relative, laquelle est reliée à l'antécédent par une dépendance MOD_REL. Considérons l'expression suivante annotée selon le guide du FTB :



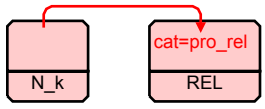

Dans cet exemple, on passe du pronom relatif à son antécédent en remontant successivement les dépendances DEP, OBJ, DEP, OBJ, OBJ, OBJ, OBJ, MOD_REL (en rouges dans la figure ci-dessous). Finalement, la nouvelle dépendance ANT_REL (en vert et en bas sur la figure), est produite.

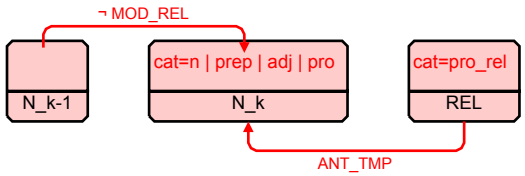


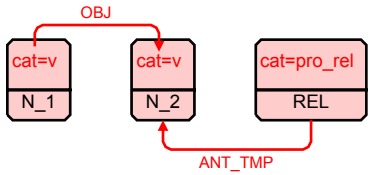

La transformation peut être décrite à l'aide du motif généralisé qui suit :

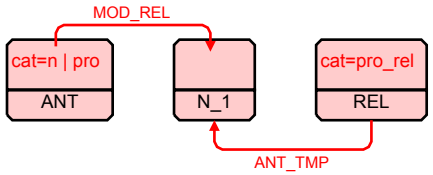



Toutefois, d'une part, ce type de motif ne permet pas d'exprimer les contraintes (comme les contraintes d'îlots) sur les structures de trait des nœuds. D'autre part, un tel motif n'est plus local, or la localité du calcul est une propriété importante pour garantir une implantation efficace. La solution proposée ici emploie un lien temporaire ANT_TMP entre le pronom relatif et le point de la chaîne où on se situe dans sa remontée. Les quatre règles ci-dessous implantent le motif généralisé en prennant en compte les contraintes d'îlots.

<table>
<tr><th>INIT</th><th>REC</th></tr>
<tr><td> 

<br>add_edge REL –[ANT_TMP]–> N<sub>k</sub></td><td> 

<br>del_edge REL –[ANT_TMP]–> N<sub>k</sub><br>add_edge REL –[ANT_TMP]–> N<sub>k−1</sub></td></tr>
<tr><th>RECV</th><th>STOP</th></tr>
<tr><td> 

<br>del_edge REL –[ANT_TMP]–> N<sub>2</sub><br>add_edge REL –[ANT_TMP]–> N<sub>1</sub></td><td> 

<br>del_edge REL –[ANT_TMP]–> N<sub>1</sub><br>add_edge REL –[ANT_REL]–> ANT</td></tr>
</table>

Les relatives qui sont en apposition ou les constructions clivées peuvent être repérées par le même mécanisme de règle généralisée décomposée en règles atomiques (en changeant seulement la dernière règle). Les exemples suivants sont issus du corpus :

C'est une novice inspirée qui redressa [...]
Une mutation est en marche, qui ne s'arrêtera sans doute pas [...]

De façon intéressante, la procédure de remontée de la chaîne de dépendances que nous venons de proposer pour retrouver l'antécédent d'un pronom relatif correspond au post-traitement appliqué par [NRMGR10] pour extraire les dépendances non-bornées complexes à partir d'une analyse en dépendances de surface.

### 10.3.4 La coordination

La coordination est annotée dans le FTB de façon à respecter la contrainte d'arbre : un lien COORD va de la tête du premier conjoint jusqu'à la conjonction de coordination et un autre lien DEP_COORD va de la conjonction jusqu'à la tête du second conjoint. Si la tête du second conjoint est un verbe ou un adjectif, il est nécessaire de déterminer ses actants syntaxiques pour calculer sa représentation sémantique ; or les actants syntaxiques du second conjoint ne sont pas annotés dans le FTB quand ils sont partagés avec le premier conjoint. Considérons quelques exemples de coordination de syntagmes verbaux ou adjectivaux, où la tête du second conjoint est en gras et les actants partagés soulignés :

(10.1) Jean déballe et **mange** son casse-croûte.

(10.2) Abandonnée de son mari et **dégoûtée**, Marie ne sort plus.

(10.3) Jean pense partir aujourd'hui et **pouvoir** rentrer dans un mois.

Nous nous limitons ici à l'annotation du sujet du second conjoint. L'annotation des autres arguments peut être ajoutée, en utilisant des règles ancrées lexicalement. Le principe est simple : dans le cas de deux verbes conjoints, si le premier conjoint a un sujet mais pas le second, alors le second conjoint a également pour sujet celui du premier conjoint. Deux règles différentes créent cette dépendance, selon que les syntagmes verbaux coordonnés sont introduits ou non par une préposition ou un complémenteur.

## 10.4 Modules

Nous avons montré au chapitre 9 qu'une organisation modulaire permet d'avoir un contrôle global sur le calcul et de simplifier le développement d'un système de règles. Les modules sont déterminés linguistiquement : un module est un ensemble de règles contribuant à une transformation linguistique particulière qu'il est possible d'isoler relativement aux autres règles.

Par rapport à notre objectif général de construire une représentation sémantique à partir d'une représentation syntaxique en dépendances, les modules jouent un rôle essentiel. Par rapport à la tâche limitée qui est celle présentée dans ce chapitre et qui consiste à enrichir les dépendances du FTB, ils ont une portée plus restreinte. Les interactions possibles entre les règles permettent difficilement de les isoler en modules.

Ainsi, les règles grammaticales d'actants et les règles lexicales d'actants peuvent se combiner de plusieurs façons. Prenons deux exemples. Dans la phrase *« Jean boit en* **essayant** *d'***oublier** *»*, une règle grammaticale établit que *« Jean »* est sujet d'*« essayant »*, puis une règle lexicale ajoute que *« Jean »* est aussi sujet d'*« oublier »*. Dans la phrase *« Marie interdit à Jean de* **boire** *en* **conduisant** *»*, une règle lexicale établit *« Jean »* comme sujet de *« boire »*, puis une règle grammaticale établit que *« Jean »* est aussi sujet de *« conduisant »*.

Séparer les règles lexicales et grammaticales dans des modules différents ne permettrait pas de capturer toutes leurs interactions. C'est la raison pour laquelle nous répartissons les règles, au nombre de 38, présentées dans les sous-sections précédentes en quatre modules ordonnés comme suit :

- *ANT-REL* : ajoute les liens ANT_REL et ANT_APP des pronoms relatifs vers leur antécédent ; il contient 6 règles qui décomposent le motif généralisé ;
- *ANT-PRO* : ajoute les liens ANT_REFL et ANT_REP des pronoms réfléchis et des pronoms personnels répétitions de sujets vers leur antécédent ; il contient 3 règles, 2 pour les pronoms répétitions de sujets et 1 pour les pronoms réfléchis ;
- *SUBJ-REAL* : ajoute le sujet des infinitifs, participes passés et adjectifs quand il est réalisé (dont règles grammaticales et règles lexicales pour les verbes à montée) ; il contient 12 règles, dont 4 pour les verbes à montée et les modaux ;

– *ARG-REAL-CONTROL* : ajoute les arguments syntaxiques profonds des infinitifs introduits par un verbe ou un adjectif à contrôle ; il contient 17 règles : 4 pour les adjectifs et 13 pour les verbes à contrôle.

Le fait d'ajouter des sujets aux adjectifs permet de gérer le *tough movement* avec une seule règle. Cela impose l'ordre d'application des modules concernés : le module *SUBJ-REAL* doit être appliqué avant le module *ARG-REAL-CONTROL*.

Enfin, la coordination intervient dans tous les modules et nécessite un traitement particulier. Reprenons un exemple déjà introduit pour la coordination :

Jean pense partir aujourd'hui et **pouvoir** rentrer dans un mois.

Dans cet exemple, il faut déterminer le sujet de *« partir »* avant d'en déduire que *« pouvoir »* partage ce sujet. Cela incite donc à appliquer les règles de coordination après le module *SUBJ-REAL*. Cependant, pour déterminer le sujet de *« rentrer »*, il est nécessaire de connaître le sujet de *« pouvoir »*. De ce point de vue, il faudrait donc appliquer les règles relatives à la coordination avant le module *SUBJ-REAL*. Un moyen de résoudre cette contradiction est de particulariser chaque règle de coordination en autant de règles qu'il existe de modules concernés.

## 10.5 Expérimentation

Nous avons appliqué notre système aux 12 351 phrases annotées en dépendances que contient le FTB. Les relations ANT_X, SUJP et OBJP que nous ajoutons sont très courantes dans le corpus : on en observe dans 87% des phrases. En outre, notre programme ajoute, sur chaque phrase, trois nouvelles relations en moyenne.

Au total, nous ajoutons :

– 3 691 relations ANT_X (3 152 pour les pronoms relatifs, 99 pour les pronoms réfléchis et 270 pour les pronoms personnels répétitions de sujets) ;
– 33 605 relations SUJP (23 940 pour les adjectifs et 9 665 pour les verbes) ;
– 19 relations OBJP.

Pour avoir une idée précise de la pertinence de ce système, il convient également d'essayer de détecter les cas problématiques, c'est-à-dire les cas où des relations sont susceptibles de manquer dans les structures produites par notre programme de réécriture. Pour cela, nous avons observé les trois configurations suivantes :

– les verbes (sauf impératifs et auxiliaires) qui n'ont toujours pas de sujet (de surface ou profond) après réécriture : il y en a 3 548 ;
– les adjectifs sans sujet profond : 955 ;
– les pronoms relatifs sans antécédent : 242 ;

Pour chacune de ces configurations, nous avons étudié un échantillon de 100 phrases que nous avons classées manuellement. Le résultat de ce classement figure dans le tableau 10.3 ci-dessous.

Les erreurs d'annotation du FTB que nous avons relevées sont systématiques et probablement liées à la conversion du FTB des constituants aux dépendances [CCDG09]. L'annotation

| | Pas d'erreur | Source de l'erreur | | |
|---|---|---|---|---|
| | | Annotation FTB | Lexique | Règles |
| V sans sujet | 51 | 29 | 13 | 7 |
| Adj sans sujet | 32 | 45 | 1 | 22 |
| Pro_rel sans antécédent | 5 | 72 | 0 | 23 |

FIGURE 10.3 – Analyse des configurations suspectes

du FTB est donc la première source d'erreur dans notre processus. Les problèmes de lexique apparaissent dans des constructions nominales, comme dans *« la volonté de Jean de convaincre Marie »*. Ces constructions ne sont pas décrites dans Dicovalence. Enfin, nos règles ne couvrent pas certains phénomènes linguistiques : par exemple, un adjectif en emploi substantivé peut être antécédent d'une relative. Nous pourrions relâcher les contraintes des motifs pour augmenter la robustesse des règles et permettre notamment qu'un adjectif soit l'antécédent d'une relative, mais notre programme serait alors moins intéressant pour la fouille d'erreurs dans l'annotation en dépendances du FTB.

Comme au chapitre précédent, il n'existe pas de corpus de référence du français qui soit annoté avec ces relations grammaticales. Il serait peut-être possible, toutefois, d'extraire une partie de l'information contenue dans le MFTDB (Modified French Treebank Dependency Bank) [Sch11]. Le MFTDB correspond à une partie du FTB dont l'annotation a été modifiée puis enrichie avec des f-structures LFG.

## 10.6 Discussion

### 10.6.1 Travaux apparentés

Gardent et Cerisara [GC10, Gar10] ont également utilisé la réécriture de graphes sur le FTB. Leur objectif était toutefois différent du nôtre, puisqu'il consistait à ajouter une annotation en rôles sémantiques pour les verbes. C'est pourquoi la plupart des règles qu'ils ont conçues vise à reformuler les constructions passives et causatives en constructions actives canoniques. Quelques règles supplémentaires traitent de la coordination et des sujets des infinitifs. Ces dernières règles, comme elles n'utilisent pas d'information lexicale, ont le défaut de choisir systématiquement pour sujet des infinitifs compléments celui du verbe dont ils dépendent. Enfin, l'étude n'aborde pas la question de la dépendance entre les règles et de leur ordre d'application.

Le système de Cahill, McCarthy, van Genabith et Way [CMGW02] ont proposé un algorithme d'annotation automatique du *Penn Treebank* par des f-structures LFG. Ce système est relativement proche du nôtre, puisque les f-structures contiennent les relations grammaticales profondes que nous ajoutons. Cependant, l'algorithme qu'ils emploient pour générer les f-structures est

sensiblement différent. De plus, cet algorithme repose, pour déterminer les sujets des gérondifs, infinitifs et participes, sur les annotations de traces du Penn Treebank. Dans notre approche, l'information qui permet de déterminer ces sujets n'est pas contenue dans la structure de départ.

### 10.6.2 Limites

Nous avons montré au chapitre 9 qu'il est possible de prendre en compte les reformulations dans toute leur diversité (passif, moyen, causatif, impersonnel ...) à l'aide de la réécriture de graphes. Nous avons cependant écarté celles-ci du travail présenté dans ce chapitre, car nous nous étions fixé comme cadre d'enrichir les structures en dépendances du FTB sans modifier les relations existantes. Intégrer les reformulations nécessiterait d'ajouter des règles au système qui vient d'être présenté. Les interactions complexes entre ces règles nouvelles et les règles existantes font que cela ne peut pas se faire simplement par l'ajout d'un module avant ou après ceux qui ont été définis dans ce chapitre. Par exemple, pour la phrase *« Jean est autorisé à* **partir** *»*, la reformulation du passif doit être effectuée avant la détermination du sujet profond de *« partir »*. C'est le contraire pour la phrase *« Jean demande à Marie d'être* **aidée** *de Pierre. »* En conséquence, l'intégration des reformulations devrait nous amener à restructurer le système de modules ou à augmenter sensiblement le nombre de règles.

Par ailleurs, nos expérimentations ont mis en évidence certaines limites du format d'annotation du FTB, qui sont dues essentiellement au choix des annotateurs de n'avoir que des arbres comme structures de dépendance. Premièrement, l'annotation des constructions causatives ne distingue pas le sujet et l'objet du verbe causé. Dans *« Jean fait* **manger** *un lapin »*, *« lapin »* est annoté OBJ de *« manger »*, que le lapin mange ou soit mangé. Deuxièmement, lorsque deux verbes sont coordonnés, l'annotation ne permet pas de retrouver les compléments partagés du second conjoint. Ainsi *« Jean* **déballe** *et* **mange** *son sandwich »* et *« Jean* **déballe** *son sandwich et* **mange** *»* ont la même annotation. Rien ne permet de distinguer que dans la première phrase, *« sandwich »* est également objet de *« déballe »* alors que dans la deuxième, *« sandwich »* n'est pas objet de *« mange »*.

### 10.6.3 Perspectives

Ce programme de réécriture peut être utilisé pour deux tâches distinctes.

#### Fouille d'erreurs

La première tâche concerne la fouille d'erreurs sur les structures du FTB. En effet, les problèmes que nous avons rencontrés au cours de nos expérimentations (échec de la réécriture, structures produites malformées) relèvent souvent d'erreurs d'annotations ou d'incohérences dans les structures du FTB. Notre système peut donc servir, par effet de bord, d'outil de fouilles d'erreurs et d'aide à la correction de corpus annotés.

**Affinement d'étiquettes de dépendances**

La deuxième tâche est l'affinement des étiquettes des relations de dépendance. Actuellement, la conversion automatique du schéma d'annotation fonctionnelle en constituants du FTB ne produit pas les étiquettes plus précises définies pour certaines relations dans le guide d'annotation en dépendances. Les étiquettes précises définies dans le guide sont :

- SUJ_IMPERS pour les clitiques sujets explétifs *« il »* ;
- P-OBJ_AGT pour les compléments d'agent des constructions passives et causatives ;
- P-OBJ_LOC pour les arguments prépositionnels locatifs ;
- MOD_CLEFT pour les modificateurs qui sont des subordonnées dans les constructions clivées ;
- MOD_LOC pour les modificateurs sémantiquement locatifs ;
- AFF_MOYEN pour les clitiques *« se »* qui marquent la voix moyenne ;
- ARG_COMP entre une comparative et son gouverneur ;
- ARG_CONS entre une consécutive et son gouverneur adverbial.

Les étiquettes génériques sont remplacées lors de l'annotation manuelle par les étiquettes précises, lorsque ces dernières sont pertinentes.

Les outils et les ressources que nous avons utilisés dans ce chapitre peuvent être utilisés pour accélérer la phase d'annotation manuelle. Dans chaque entrée Dicovalence, il est spécifié en particulier :

- si le sujet de ce cadre verbal est un clitique sujet explétif, noté `pseudo_il` ;
- si ce cadre verbal a un argument prépositionnel locatif, noté `PL` pour les locatifs et `PDL` pour les délocatifs ;
- si ce cadre verbal admet la voix moyenne, qui correspond à la reformulation passive notée `se passif`.

Nous pouvons, par le même mécanisme que dans le chapitre précédent et ce chapitre, générer des règles de réécriture qui reconnaissent les configurations correspondantes et remplacent l'étiquette générique par l'étiquette précise. Dans la plupart des cas, la configuration syntaxique ne permet pas de déterminer automatiquement si une règle doit s'appliquer. Les règles de réécriture peuvent cependant être appliquées de façon interactive, pour détecter toutes les dépendances qui pourraient être précisées dans le corpus et les proposer à un annotateur. Celui-ci n'aurait alors qu'à décider pour chaque dépendance candidate si la réécriture doit s'appliquer.

## Conclusion

Dans ce chapitre, nous avons présenté un programme de réécriture qui ajoute des relations grammaticales à une structure de dépendance syntaxique de surface au format du FTB. Ces relations supplémentaires sont calculées par les analyseurs syntaxiques profonds et utilisées pour le calcul de la sémantique. Il n'existe actuellement pas pour le français de corpus annoté avec ces relations grammaticales. Les expérimentations qui requièrent ces informations, comme celle que nous avons conduite au chapitre précédent, demandent par conséquent une phase manuelle

de préparation des données pour ajouter les annotations manquantes. Nous pallions ici à ce manque en proposant un enrichissement automatique du FTB par des règles purement grammaticales et par d'autres qui utilisent de l'information lexicale extraite de Dicovalence [vdEM03]. Cet enrichissement automatique repose sur le calcul de réécriture de graphes modulaire que nous avons présenté au chapitre précédent. La composition des deux programmes de réécriture présentés dans ce manuscrit permet de calculer une représentation DMRS à partir d'un arbre de dépendance syntaxique de surface du FTB. Le développement et l'amélioration conjoints de ces deux programmes permet d'envisager, à terme, leur utilisation pour produire une annotation de référence du FTB, exploitable pour l'évaluation des analyseurs syntaxiques profonds et des chaînes de traitement linguistique qui produisent des structures sémantiques.

# Conclusion

L'objectif général qui motive les travaux présentés dans ce manuscrit est de faire de l'analyse syntaxique et sémantique à grande échelle sur des corpus écrits. De nombreux formalismes grammaticaux, de nombreuses grammaires et de nombreux analyseurs syntaxiques ont été conçus et développés dans ce but.

Le point de départ concret de ce travail est un triplet formé par le formalisme des IG, l'analyseur syntaxique LEOPAR pour les IG, et la grammaire d'interaction du français FRIGRAM. Comment, autour de ce triplet, développer une chaîne d'analyse linguistique qui produise des représentations sémantiques ?

Les IG sont un formalisme grammatical lexicalisé, qui appartient au courant de l'analyse syntaxique fondé sur la théorie des modèles, et dans lequel la notion de polarité joue un rôle central. Les IG présentent de nombreux points communs avec les principaux formalismes grammaticaux, dont LTAG, HPSG et les CG, mais sans leur ressembler suffisamment pour que les méthodes couramment utilisées dans ces formalismes soient directement transposables. La grammaire FRIGRAM résiste tout autant à la classification, empruntant autant aux grammaires de constituants qu'aux grammaires de dépendance et étant en perpétuelle évolution.

Pour bâtir une chaîne de traitement linguistique autour de ces éléments, il faut donc s'appuyer sur les propriétés fondamentales du formalisme : la lexicalisation et la polarisation, et sur les propriétés fondamentales de la grammaire : la réalisation de relations grammaticales entre mots.

## 1 Contributions

Dans la première partie de la thèse, nous exploitons les propriétés de lexicalisation et de polarisation d'un formalisme grammatical pour extraire des informations d'une grammaire. Pour cela, nous définissons d'abord au chapitre 1 le cadre général des formalismes grammaticaux lexicalisés polarisés. Nous présentons ensuite, au chapitre 2, les IG comme un exemple de formalisme grammatical lexicalisé polarisé. Dans les formalismes lexicalisés, analyser une phrase consiste à associer aux mots de cette phrase des structures syntaxiques de la grammaire et à composer ces structures. La polarisation de ces formalismes permet de rendre explicites les possibilités et les besoins de composition des structures de la grammaire.

Dans le chapitre 3, nous exploitons les besoins exprimés par les polarités. Nous partons du constat qu'il est possible de calculer, pour chaque structure de la grammaire, l'ensemble des

structures de la grammaire qui sont capables de répondre à l'un de ses besoins de composition, dans une analyse de phrase. Nous définissons ainsi la notion de *compagnon hypothétique* d'une instance de polarité dans une grammaire. Si une structure a un besoin de composition qui ne peut être satisfait par aucune structure de la grammaire, alors cette structure ne peut être utilisée dans aucune analyse syntaxique de phrase. Ces structures inutiles peuvent être retirées de la grammaire. Nous définissons ainsi la notion de *quasi-réduction d'une grammaire polarisée*, qui est applicable à toutes les grammaires polarisées quel que soit le formalisme. La notion traditionnelle, plus forte, de réduction d'une grammaire n'est à notre connaissance définie que pour les formalismes grammaticaux génératifs-énumératifs. La définition des compagnons hypothétiques et de la quasi-réduction des grammaires polarisées, ainsi que leur utilisation pour le développement et la maintenance de grammaires, constituent la première contribution de cette thèse.

Dans la deuxième partie de la thèse, nous utilisons les propriétés de la grammaire que nous avons mises en évidence dans la première partie pour améliorer la première phase de l'analyse syntaxique des formalismes lexicalisés : l'étiquetage grammatical. Cette phase est cruciale pour les formalismes qui, comme les IG, combinent une analyse syntaxique NP-difficile et des grammaires induisant une forte ambiguïté lexicale initiale. Il existe peu de méthodes symboliques pour l'étiquetage grammatical. Nous les présentons au chapitre 4. Nous proposons ensuite, aux chapitres 5 et 6, trois méthodes symboliques de filtrage des étiquetages grammaticaux pour les formalismes polarisés. Ces méthodes utilisent la notion de compagnon hypothétique introduite dans la première partie de la thèse. Nous implantons ces méthodes de filtrage sur automates et nous montrons, au chapitre 7 leur efficacité et leur applicabilité par des expérimentations sur corpus, en les combinant aux méthodes de filtrage des étiquetages grammaticaux pour les formalismes polarisés qui existaient auparavant. La définition et l'évaluation de ces méthodes pour l'analyse syntaxique constituent la deuxième contribution de cette thèse.

Dans la troisième partie de la thèse, nous exploitons les structures de relations grammaticales entre mots qui sont construites par les grammaires lexicalisées pour proposer une interface syntaxe - sémantique générique. Dans le chapitre 8, nous constatons d'abord que, dans la plupart des formalismes lexicalisés, les propositions d'interface syntaxe - sémantique n'utilisent qu'une partie de l'information produite par l'analyse syntaxique. De plus, les architectures utilisées induisent un lien très fort entre l'interface syntaxe - sémantique et le formalisme ou la grammaire. Nous proposons de remédier à cela en découplant clairement le calcul de la sémantique de celui de la syntaxe, et en calculant la sémantique à partir de structures de dépendance syntaxique enrichies par des relations grammaticales. Dans les formalismes lexicalisés, l'historique de l'analyse syntaxique permet de construire des structures de dépendances ; les relations grammaticales supplémentaires peuvent quant à elles être extraites de l'historique et du résultat de l'analyse syntaxique. Dans les formalismes polarisés, l'historique de l'analyse syntaxique correspond essentiellement à l'historique des interactions entre les polarités. Les polarités occupent ainsi une place centrale dans la réalisation des relations de dépendance syntaxique et des autres relations

grammaticales, comme les arguments profonds des infinitifs et des participes.

Les structures de dépendance syntaxique sont des arbres, que l'ajout d'autres relations grammaticales transforme en graphes dans toute leur généralité. Le mécanisme de calcul le plus adapté pour travailler sur de telles structures est la réécriture de graphes. Nous définissons au chapitre 9, un calcul de réécriture de graphes qui présente deux particularités : l'effet de chaque règle est décrit par une séquence de commandes et les règles sont regroupées en modules. Le premier point permet d'avoir une vision opérationnelle du calcul, alors que le second permet de contrôler l'exécution du calcul. Nous présentons ensuite, aux chapitres 9 et 10, deux programmes de réécriture. Le premier produit, à partir d'une structure de dépendance enrichie extraite d'une analyse syntaxique, des structures sémantiques sous-spécifiées au format DMRS. Le deuxième enrichit des structures de dépendance syntaxique avec des relations grammaticales, en utilisant de l'information syntaxique et de l'information lexicale extraite de Dicovalence. Nous montrons la faisabilité de notre démarche par des expérimentations sur le corpus journalistique du *French Treebank*. La composition de ces deux programmes permet de calculer des structures sémantiques sous-spécifiées au format DMRS à partir d'une structure de dépendance syntaxique de surface au format du FTB. La démonstration concrète de la pertinence et de la faisabilité d'utiliser des structures syntaxiques riches, des graphes, et un modèle de calcul très expressif, la réécriture de graphes modulaire à base de commandes, pour l'interface syntaxe - sémantique, constituent la troisième contribution de cette thèse.

Les travaux des trois parties ont en commun d'exploiter, dans les formalismes grammaticaux lexicalisés polarisés, la même information : les interactions entre polarités qui constituent l'essentiel de l'historique du processus d'analyse syntaxique. De plus, les méthodes que nous avons proposées ne font aucune hypothèse forte, ni sur le formalisme grammatical, ni sur la grammaire. La procédure de quasi-réduction de grammaire, les méthodes de filtrage des étiquetages grammaticaux et l'interface syntaxe-sémantique présentées dans ce manuscrit sont donc applicables à tout formalisme grammatical lexicalisé polarisé.

Nous avons ainsi abordé trois des étapes d'une chaîne de traitement linguistique bâtie autour d'un analyseur syntaxique : le développement de grammaires, l'étiquetage grammatical et l'interface syntaxe - sémantique.

## 2 Perspectives

Les perspectives ouvertes par ce travail concernent essentiellement la deuxième et la troisième partie. Les contributions de la première partie peuvent toutefois être exploitées avec profit dans un environnement de développement de grammaires, notamment pour mettre en place des tests unitaires.

Les travaux sur les méthodes de filtrage ouvrent plus de questions qu'elles n'en résolvent. Les performances des nouvelles méthodes de filtrage peuvent encore être améliorées en prenant mieux en compte l'information contenue dans la phrase et en utilisant ces méthodes pour guider

les algorithmes d'analyse. De plus, ces nouvelles méthodes de filtrage semblent pour le moment très différentes, dans leur principe et leur fonctionnement, des méthodes existantes. Plusieurs éléments indiquent cependant qu'il doit exister un cadre général qui offrirait une perspective unifiée sur ces différentes méthodes. Ensuite, sur le plan théorique, la caractérisation de ces méthodes en termes de langage et de logique n'est pas triviale. Enfin, il serait intéressant d'appliquer ces nouvelles méthodes aux autres formalismes polarisés ou polarisables, comme les LTAG.

Les résultats de la troisième partie ouvrent la voie à la production de structures de dépendances sémantiques sous-spécifiées à partir de corpus annotés par des dépendances syntaxiques et, plus généralement, à une utilisation plus répandue des systèmes modulaires de réécriture en linguistique informatique. De manière générale, il est cependant impératif d'améliorer profondément nos programmes de réécriture pour avoir une modélisation linguistique beaucoup plus fine. Cette amélioration requiert d'intégrer des ressources linguistiques. La réécriture de graphes est un modèle de calcul très souple, qui pourrait être utilisé avec profit pour annoter les expressions multi-mots en corpus et pour les intégrer au calcul de la sémantique [BK10]. Dans un premier temps, les programmes de réécriture que nous avons présentés peuvent servir d'outil de fouille d'erreurs pour corriger l'annotation en dépendance du FTB. À long terme, il est envisageable d'utiliser des programmes de réécriture de graphe pour créer une annotation syntaxique enrichie et une annotation sémantique qui puissent servir de références à l'évaluation des analyseurs syntaxiques profonds et à l'entraînement d'analyseurs statistiques.

# Bibliographie


[AB04] Anne Abeillé and Nicolas Barrier. Enriching a french treebank. In *Proceedings of Language Ressources and Evaluation Conference*, 2004. 138

[Abr96] V. Michele Abrusci. Lambek calculus, cyclic multiplicative-additive linear logic, noncommutative multiplicative-additive linear logic : language and sequent calculus. In *Proceedings of 1996 Roma Workshop on Proofs and Linguistic Categoriss*, pages 21–48, 1996. 20

[ACT03] Anne Abeillé, Lionel Clément, and François Toussenel. Building a Treebank for French. In Anne Abeillé, editor, *Treebanks. Building and Using Parsed Corpora*, chapter 10. Dordrecht : Kluwer Academic Publishers, 2003. 4, 43, 138

[Ajd35] Kazimierz Ajdukiewicz. Die syntaktische Konnexität. *Studia philosophica*, 1(1) :27, 1935. 9, 13

[AR00] V. Michele Abrusci and Paul Ruet. Non-commutative logic i : the multiplicative fragment. *Annals of pure and applied logic*, 101(1) :29–64, 2000. 120

[BB03] Patrick Blackburn and Johan Bos. Computational semantics. *Theoria*, 18(1) :27–45, 2003. 126

[BB05] Patrick Blackburn and Johan Bos. *Representation and inference for natural language : A first course in computational semantics*. CSLI, 2005. 126

[BBH11] Martin Berglund, Henrik Björklund, and Johanna Högberg. Recognizing shuffled languages. *Language and Automata Theory and Applications*, pages 142–154, 2011. 120

[BBN+09] Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. Mica : A probabilistic dependency parser based on tree insertion grammars. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 185–188, Boulder, Colorado, 2009. 68

[Bed10] Paul Bedaride. *Implication Textuelle et Réécriture*. PhD thesis, Université Henri Poincaré - Nancy 1, 2010. 171

[BFOZ11] Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of*

*the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 397–408, Edinburgh, Scotland, 2011. 136, 174

[BG08] Olivier Bonami and Danièle Godard. Lexical semantics and pragmatics of evaluative adverbs. In Louise McNally and Christopher Kennedy, editors, *Adjectives and Adverbs : Syntax, Semantics, and Discourse*, volume 19 of *Oxford Studies in Theoretical Linguistics*, pages 274–304. Oxford University Press, 2008. 168

[BG09] Paul Bédaride and Claire Gardent. Semantic Normalisation : a Framework and an Experiment. In *Proceedings of IWCS*, Tilburg, Netherlands, 2009. 146

[BGM03] Patrick Blackburn, Bertrand Gaiffe, and Maarten Marx. Variable free reasoning on finite trees. *Mathematics of Language*, 8, 2003. 10

[BGP03] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. Analyse syntaxique électostatique. *Traitement Automatique des Langues*, 44 :3 Évolutions en analyse syntaxique, 2003. 3, 37

[BGP04] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *Proceedings of CoLing 2004*, 2004. 2, 3, 20, 72, 74, 119

[BH53] Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29(1) :47–58, 1953. 9

[BHHH03] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. *The Prague Dependency Treebank : A Three-Level Annotation Scenario*, pages 103–127. Amsterdam : Kluwer, 2003. 138

[BJ99] Srinivas Bangalore and Aravind K. Joshi. Supertagging : An approach to almost parsing. *Computational linguistics*, 25(2) :237–265, 1999. 67, 69

[BJ10] Srinivas Bangalore and Aravind K. Joshi. *Supertagging : Using Complex Lexical Descriptions in Natural Language Processing*. The MIT Press, 2010. 2, 11, 67, 68, 69

[BK10] Timothy Baldwin and Su Nam Kim. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010. ISBN 978-1420085921. 192

[Bla05] Philippe Blache. Property grammars : A fully constraint-based theory. *Constraint Solving and Language Processing*, pages 119–142, 2005. 10

[Bla07] Philippe Blache. Pour une représentation décentralisée de l'information syntaxique. In *Actes de TALN*, 2007. 10, 25

[BLRP06] Guillaume Bonfante, Joseph Le Roux, and Guy Perrier. Lexical disambiguation with polarities and automata. In *Procedings of 11th International Conference on Implementation and Application of Automata*, Taipei Taiwan, 2006. 3, 72, 74

[BN99] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1999. 146

[Bos01] Johan Bos. *Underspecification and Resolution in Discourse Semantics*, volume 12 of *Saarbrücken dissertations in computational linguistics and language technology.* Universität des Saarlandes, 2001. 126

[Bou98] Pierre Boullier. Proposal for a natural language processing syntactic backbone. Technical report, INRIA, 1998. 48

[Bou03] Pierre Boullier. Guided earley parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03), Nancy, France*, pages 43–54, 2003. 55

[Bou10] Pierre Boullier. A nonstatistical parsing-based approach to supertagging. In Srinivas Bangalore and Aravind K. Joshi, editors, *Supertagging : Using Complex Lexical Descriptions in Natural Language Processing*, pages 173–191. The MIT Press, 2010. 2, 69, 83

[Bun07] Harry Bunt. Semantic underspecification : Which technique for what purpose ? *Computing Meaning*, pages 55–85, 2007. 126

[Bus01] Wojciech Buszkowski. Lambek grammars based on pregroups. *Logical aspects of computational linguistics*, pages 95–109, 2001. 99

[BW01] Bernd Bohnet and Leo Wanner. On using a parallel graph rewriting formalism in generation. In *Proceedings of EWNLG '01*, pages 1–11. Association for Computational Linguistics, 2001. 146

[Can96] Marie-Hélène Candito. A principle-based hierarchical representation of ltags. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 194–199. Association for Computational Linguistics, 1996. 58

[Car08] Olivier Carton. *Langages formels, calculabilité et complexité.* Vuibert, 2008. 48

[CBS98] John Carroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation : a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, 1998. 136

[CCD10] Marie-Hélène Candito, Benoît Crabbé, and Pascal Denis. Statistical french dependency parsing : Treebank conversion and first results. *Proceedings of LREC'2010*, 2010. 5, 139

[CCDG09] Marie-Hélène Candito, Benoît Crabbé, Pascal Denis, and François Guérin. Analyse syntaxique statistique du français : des constituants aux dépendances. In *Actes de TALN*, Senlis, France, 2009. 138, 139, 174, 183

[CCF11] Marie-Hélène Candito, Benoît Crabbé, and Mathieu Falco. *Dépendances syntaxiques de surface pour le français*, 2011. 139, 167, 174

[CCV06] James R. Curran, Stephen Clark, and David Vadas. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 697–704. Association for Computational Linguistics, 2006. 68

[CF00] Ann Copestake and Dan Flickinger. An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Conference on Language Resources and Evaluation*, 2000. 127

[CFL+02] John Carroll, Anette Frank, Dekang Lin, Detlef Prescher, and Hans Uszkoreit. — Beyond PARSEVAL — towards improved evaluation measures for parsing systems. In *Proceedings of the Beyond PARSEVAL Workshop at LREC*, 2002. 136, 174

[CFPS05] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics - an Introduction. *Research on Language and Computation*, 3 :281–332, 2005. 126

[Chi00] David Chiang. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 456–463. Association for Computational Linguistics, 2000. 17

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax*, volume 119. The MIT press, 1965. 131, 136, 170

[Cho77] Noam Chomsky. On wh-movement. In Thomas Wasow Peter Culicover and Adrian Admajian, editors, *Formal Syntax*, pages 71–132. Academic Press, New York, 1977. 133

[Cho95] Noam Chomsky. *The minimalist program*, volume 1765. Cambridge University Press, 1995. 10

[Cho02] Noam Chomsky. *Syntactic structures*. Walter de Gruyter, 2002. 10

[CHS02] Stephen Clark, Julia Hockenmaier, and Mark Steedman. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 327–334. Association for Computational Linguistics, 2002. 135, 174

[CK98] Marie-Hélène Candito and Sylvain Kahane. Can the tag derivation tree represent a semantic graph ? an answer in the light of meaning-text theory. In *Proceedings of the Fourth Workshop on Tree-Adjoining Grammars and Related Frameworks*, 1998. 129

[CK06] Richard Crouch and Tracy H. King. Semantics via f-structure rewriting. In *Proceedings of the LFG06 Conference*, pages 145–165, 2006. 131, 171

[CK10] François-Régis Chaumartin and Sylvain Kahane. Une approche paresseuse de l'analyse sémantique ou comment construire une interface syntaxe-sémantique à partir d'exemples. In *Actes de TALN*, Montréal, Canada, 2010. 146, 171

[CMB03] John Carroll, Guido Minnen, and Ted Briscoe. Parser evaluation : using a grammatical relation annotation scheme. In Anne Abeillé, editor, *Treebanks. Building and Using Parsed Corpora*, pages 299–316. Dordrecht : Kluwer Academic Publishers, 2003. 128, 135

[CMGW02] Aoife Cahill, Mairead McCarthy, Josef Van Genabith, and Andy Way. Automatic annotation of the penn treebank with lfg f-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation-Bootstrapping Annotated Language Data*, pages 8–15, 2002. 184

[Cop07] Ann Copestake. Semantic composition with (robust) minimal recursion semantics. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 73–80. Association for Computational Linguistics, 2007. 148

[Cop09] Ann Copestake. Slacker semantics : why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–9, Athens, Greece, 2009. Association for Computational Linguistics. 5, 148

[Cro05] Richard Crouch. Packed Rewriting for Mapping Semantics to KR. In *Proceedings of IWCS*, 2005. 146

[Dal99] Mary Dalrymple, editor. *Semantics and syntax in Lexical Functional Grammar : The resource logic approach*. The MIT Press, 1999. 126

[DDPL10] Denys Duchier, Thi-Bich-Hanh Dao, Yannick Parmentier, and Willy Lesaint. Une modélisation en CSP des Grammaires de Propriétés. In *JFPC 2010 - Sixièmes Journées Francophones de Programmation par Contraintes*, pages 123–132, Caen France, 2010. 10

[Deb06] Ralph Debusmann. *Extensible Dependency Grammar : a modular grammar formalism based on multigraph description*. PhD thesis, Universität des Saarlandes, 2006. 10, 130, 135

[dG01] Philippe de Groote. Towards abstract categorial grammars. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 252–259. Association for Computational Linguistics, 2001. 128

[dGP04] Philippe de Groote and Sylvain Pogodalla. On the expressive power of abstract categorial grammars : Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4) :421–438, 2004. 128

[dGPP11] Philippe de Groote, Sylvain Pogodalla, and Carl Pollard. About Parallel and Syntactocentric Formalisms : A Perspective from the Encoding of Convergent Gram-

mar into Abstract Categorial Grammar. *Fundamenta Informaticae*, 106(2-4) :211–231, 2011. 129

[DL10] Valmi Dufour-Lussier. *Parsing Punctuation and Coordination Extragrammatically.* PhD thesis, Université Nancy 2, 2010. Master's thesis. 25

[DLP05a] Denys Duchier, Joseph Le Roux, and Yannick Parmentier. The metagrammar compiler : An nlp application with a multi-paradigm architecture. *Multiparadigm Programming in Mozart/Oz*, pages 175–187, 2005. 14, 58

[DLP05b] Denys Duchier, Joseph Le Roux, and Yannick Parmentier. XMG : Un compilateur de méta-grammaires extensible. *Actes de TALN*, 2005. 43

[DLPS96] Mary Dalrymple, John Lamping, Fernando Pereira, and Vijay Saraswat. *A deductive account of quantification in LFG*, pages 33–57. CSLI Publications, Stanford, 1996. 129

[Ebe05] Christian Ebert. *Formal investigations of underspecified representations.* PhD thesis, Department of Computer Science, King's College, University of London, 2005. 126

[Ech08] Rachid Echahed. Inductively sequential term-graph rewrite systems. In *ICGT*, pages 84–98, 2008. 146

[Egg10] Markus Egg. Semantic underspecification. *Language and Linguistics Compass*, 4(3) :166–181, 2010. 126

[EKN01] Markus Egg, Alexander Koller, and Joachim Niehren. The constraint language for lambda structures. *Journal of Logic, Language and Information*, 10(4) :457–485, 2001. 126

[Fli08] Dan Flickinger. The english resource grammar. Technical Report Technical Report 2007-7, LOGON, 2008. Draft of 2008-11-30. 151

[Gar10] Claire Gardent. Extraction des cadres syntaxiques à partir de P7dep, December 2010. Notes transmises par l'auteur. 184

[Gaz85] Gerald Gazdar. *Applicability of indexed grammars to natural languages.* Number CSLI-85-34 in Technical Report. Center for the Study of Language and Information, Stanford University, 1985. 136

[GC10] Claire Gardent and Christophe Cerisara. Semi-Automatic Probanking for French. In *TLT9 – the ninth international workshop on Treebanks and Linguistic Theories*, Tartu, Estonia, 2010. 184

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1) :1–101, 1987. 2, 17

[Gis81] Jay Gischer. Shuffle languages, petri nets, and context-sensitive grammars. *Communications of the ACM*, 24(9) :597–605, 1981. 120

[GK03] Claire Gardent and Laura Kallmeyer. Semantic construction in feature-based tag. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 123–130. Association for Computational Linguistics, 2003. 127

[GP09] Bruno Guillaume and Guy Perrier. Interaction Grammars. *Research on Language and Computation*, 7(2-4) :171–208, 2009. 23, 47

[HK98] Irene Heim and Angelika Kratzer. *Semantics in generative grammar*. Wiley-Blackwell, 1998. 128

[HMU06] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. 98

[HS07] Julia Hockenmaier and Mark Steedman. CCGbank : a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3) :355–396, 2007. 135

[Hud84] Richard A. Hudson. *Word grammar*. B. Blackwell, 1984. 11

[HY08] Julia Hockenmaier and Peter Young. Non-local scrambling : the equivalence of TAG and CCG revisited. In *Proceedings of TAG*, volume 9, 2008. 131

[Hyv84] Eero Hyvönen. Semantic Parsing as Graph Language Transformation - a Multidimensional Approach to Parsing Highly Inflectional Languages. In *COLING*, pages 517–520, 1984. 146

[Jac07] Ray Jackendoff. A parallel architecture perspective on language processing. *Brain Research*, 1146 :2–22, 2007. 132

[JdR07] Valentin Jijkoun and Maarten de Rijke. Learning to transform linguistic graphs. In *Second Workshop on TextGraphs : Graph-Based Algorithms for Natural Language Processing*, Rochester, NY, USA, 2007. 146

[Jed99] Joanna Jedrzejowicz. Structural properties of shuffle automata. *Grammars*, 2(1) :35–51, 1999. 120

[Jes37] Otto Jespersen. *Analytic Syntax*. George Allen & Unwin Ltd, London, 1937. 13

[JS97] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. *Handbook of Formal Languages : Beyond Words*, 3 :69–123, 1997. 2, 10

[JVS99] Aravind K. Joshi and K. Vijay-Shanker. Compositional semantics with lexicalized tree-adjoining grammar (ltag) : How much underspecification is necessary ? In *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, Tilburg, 1999. 129

[Kah01] Sylvain Kahane. Grammaires de dépendance formelles et théorie sens-texte. *Actes de TALN*, 2001. 11

[Kah02] Sylvain Kahane. *Grammaire d'Unification Sens-Texte : Vers un modele mathématique articulé de la langue*. PhD thesis, Université Paris 7, 2002. 129, 135

[Kah03] Sylvain Kahane. On the status of the deep syntactic structure. In *Actes de la première conférence internationale sur la Théorie Sens-Texte (MTT 2003)*, Paris, 2003. 177

[Kah06] Sylvain Kahane. Polarized unification grammars. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING 2006) and the 44th annual meeting of the Association for Computational Linguistics (ACL 2006)*, pages 137–144, Sydney, 2006. Association for Computational Linguistics. 1, 2, 4, 11, 14, 15, 18, 20, 47, 117

[Kal02] Laura Kallmeyer. Using an enriched TAG derivation structure as basis for semantics. In *Proceedings of TAG+ 6 Workshop*, pages 127–136, 2002. 129, 131

[Kal06] Laura Kallmeyer. Comparing lexicalized grammar formalisms in an empirically adequate way : The notion of generative attachment capacity. In *International Conference on Linguistic Evidence*, pages 154–156, 2006. 128, 136

[KB95] Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar : A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar*, pages 29–130, 1995. 2, 10, 129, 134

[KCR+03] Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. The parc 700 dependency bank. In *Proceedings of the EACL03 : 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, 2003. 136

[Kin00] Alexandra Kinyon. Hypertags. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 446–452. Association for Computational Linguistics, 2000. 43

[KJ03] Laura Kallmeyer and Aravind K. Joshi. Factoring predicate argument and scope semantics : Underspecified semantics with LTAG. *Research on Language & Computation*, 1(1) :3–58, 2003. 129

[KK09] Alexander Koller and Marco Kuhlmann. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 460–468. Association for Computational Linguistics, 2009. 131

[Kni09] Marie Laurence Knittel. Le statut des compléments du nom en [de NP]. *The Canadian Journal of Linguistics/La revue canadienne de linguistique*, 54(2) :255–290, 2009. 93

[Kow07] Eric Kow. *Réalisation de surface : ambiguïté et déterminisme — Surface realisation : ambiguity and determinism*. PhD thesis, Université Henri Poincaré - Nancy 1, 2007. 2, 20, 118

[KR04] Laura Kallmeyer and Maribel Romero. LTAG semantics with semantic unification. In *Proceedings of TAG*, volume 7, pages 155–162, 2004. 128, 129, 131

[KR08] Laura Kallmeyer and Maribel Romero. Scope and situation binding in LTAG using semantic unification. *Research on Language & Computation*, 6(1) :3–52, 2008. 129, 131

[Kuh10] Marco Kuhlmann. *Dependency Structures and Lexicalized Grammars : An Algebraic Approach*, volume 6270. Springer-Verlag New York Inc, 2010. 136

[KW11] Edward Keenan and Dag Westerståhl. Generalized quantifiers in linguistics and logic. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of logic and language (Second Edition)*. Elsevier, 2011. 125

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3) :154–170, 1958. 9

[Lam99] Joachim Lambek. Type grammar revisited. *Logical aspects of computational linguistics*, pages 1–27, 1999. 9, 99

[Lar08] François Lareau. *Vers une grammaire d'unification Sens-Texte du français : le temps verbal dans l'interface sémantique-syntaxe*. PhD thesis, Université de Montréal, 2008. 15

[LH06] Robert D. Levine and Thomas E. Hukari. *The unity of unbounded dependency constructions*. Number 166 in CSLI Lecture Notes. CSLI, 2006. 133

[Lis06] Pierre Lison. Implémentation d'une interface sémantique-syntaxe basée sur des grammaires d'unification polarisées. Master's thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 2006. 135

[LORP+96] Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. TSNLP : Test Suites for Natural Language Processing. In *Proceedings of the 16th conference on Computational linguistics*, pages 711–716, 1996. 43, 57

[Mar90] Hiroshi Maruyama. Structural disambiguation with constraint propagation. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 31–38. Association for Computational Linguistics, 1990. 67

[MCLS08] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. Semantic role labeling : an introduction to the special issue. *Computational linguistics*, 34(2) :145–159, 2008. 124

[Mel88] Igor Mel'čuk. *Dependency Syntax : theory and practice*. State University of New York Press, Albany, 1988. 11, 135, 147

[Mel96] Igor Mel'čuk. Lexical functions : a tool for the description of lexical relations in a lexicon. *Lexical functions in lexicography and natural language processing*, 31 :37–102, 1996. 170

[MGP10] Jonathan Marchand, Bruno Guillaume, and Guy Perrier. Motifs de graphe pour le calcul de dépendances syntaxiques complètes. In *Actes de TALN*, Montréal, Canada, 2010. 27, 135, 168

[Mil99] Philip H. Miller. *Strong generative capacity*. Number 103 in CSLI Lecture Notes. CSLI Publications, Stanford, 1999. 136

[Mit02] Ruslan Mitkov. *Anaphora resolution*. Longman, 2002. 134

[MK96] John T. Maxwell and Ronald M. Kaplan. An efficient parser for lfg. In *Proceedings of LFG*, volume 96, 1996. 131

[MMM06] Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, 2006. 138

[MMS93] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english : The penn treebank. *Computational linguistics*, 19(2) :313–330, 1993. 138

[Mon70] Richard Montague. Universal grammar. *Theoria*, 36(3) :373–398, 1970. 9, 124, 125, 128

[Mon74] Richard Montague. *The proper treatment of quantification in ordinary English*, pages 247–270. Yale University Press, 1974. 124, 125

[Mön97] Uwe Mönnich. Adjunction As Substitution : An Algebraic Formulation of Regular, Context-Free and Tree Adjoining Languages. In *Formal Grammar Conference*, volume 1, page 7012, Aix-en-Provence, 1997. eprint arXiv : cmp-lg/9707012. 39

[Moo88] Michael Moortgat. *Categorial investigations : logical and linguistic aspects of the Lambek calculus*, volume 9. Foris Pubns USA, 1988. 74

[Mor94] Glyn Morrill. Type logical grammar : categorial logic of signs. *Computational Linguistics*, 23(4), 1994. 9, 67, 128

[Mor10] Glyn Morrill. *Categorial Grammar : Logical Syntax, Semantics, and Processing*. Oxford University Press, 2010. 13, 16

[Nas95] Alexis Nasr. A formalism and a parser for lexicalised dependency grammars. In *4th International Workshop on Parsing Technologies*, pages 186–195, Prague, 1995. 16, 19

[Nas96] Alexis Nasr. *Un modèle de reformulation automatique fondé sur la Théorie Sens Texte : Application aux langues contrôlées*. PhD thesis, Université Paris 7, 1996. 16

[Nas04] Alexis Nasr. Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement. Habilitation à diriger des recherches, Université Paris 7, décembre 2004. 67

[Ned00] Mark-Jan Nederhof. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1) :17–44, 2000. 55

[NRMGR10] Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez-Rodríguez. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841. Association for Computational Linguistics, 2010. 136, 181

[NS06] Rebecca Nesson and Stuart M. Shieber. Simpler tag semantics through synchronization. In Shuly Wintner, editor, *Formal Grammar 2006*, 2006. 129

[Par07] Yannick Parmentier. *SemTAG : une plate-forme pour le calcul sémantique à partir de Grammaires d'Arbres Adjoints*. PhD thesis, Université Henri Poincaré - Nancy 1, 2007. 127, 128

[Per03] Guy Perrier. *Les grammaires d'interaction*. PhD thesis, Université Nancy 2, 2003. Habilitation à diriger les recherches en informatique. 10

[Per07] Guy Perrier. A French Interaction Grammar. In *RANLP 2007*, pages 463–467, Borovets Bulgarie, 2007. 58, 109

[PGK05] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank : An annotated corpus of semantic roles. *Computational Linguistics*, 31(1) :71–106, 2005. 124

[PL07] Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2) :161–199, 2007. 148

[Pog04] Sylvain Pogodalla. Computing semantic representation : Towards acg abstract terms as derivation trees. In *Proceedings of the seventh international workshop on tree adjoining grammar and related formalisms (TAG+ 7)*, pages 64–71, 2004. 129

[Pog07a] Sylvain Pogodalla. Ambiguïté de portée et approche fonctionnelle des grammaires d'arbres adjoints. In *Actes de la 14e conférence sur le Traitement Automatique des Langues Naturelles (communications orales)*, page 325, 2007. 128

[Pog07b] Sylvain Pogodalla. Generalizing a Proof-Theoretic Account of Scope Ambiguity. In *7th International Workshop on Computational Semantics - IWCS-7*, Tilburg, Pays-Bas, January 2007. 126, 128

[Pol84] Carl J. Pollard. *Generalized phrase structure grammars, head grammars and natural language*. PhD thesis, Stanford University, 1984. 136

[Pol99] Carl J. Pollard. Strong generative capacity in hpsg. *Lexical and constructional aspects of linguistic explanation*, 1 :281–297, 1999. 10

[Pol11] Carl Pollard. Covert movement in logical grammar. *Logic and Grammar*, pages 17–40, 2011. 128

[PP10] Sylvain Pogodalla and Florent Pompigne. Controlling extraction in abstract categorial grammars. In *15th Conference on Formal Grammar - FG 2010*, 2010. 128

[PS94] Carl Pollard and Ivan A. Sag. *Head-driven phrase structure grammar*. University of Chicago Press, 1994. 2, 26, 127

[PS01] Geoffrey K. Pullum and Barbara C. Scholz. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. *Logical aspects of computational linguistics*, pages 17–43, 2001. 2, 10, 11, 23

[RCS09] Laura Rimell, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 2-Volume 2*, pages 813–821. Association for Computational Linguistics, 2009. 136, 174

[Ret96] Christian Retoré. Calcul de lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2) :39–70, 1996. 2

[Ret00] Christian Retoré. The logic of categorial grammars. *Lecture notes ESSLLI*, 2000. 2, 13, 14, 20

[Rez06] Milan Rezac. On tough-movement. In Cedric Boeckx, editor, *Minimalist Essays*, Linguistik Aktuell/Linguistics Today 91, pages 288–325. John Benjamins, 2006. 175, 177

[RJ97] Owen Rambow and Aravind K. Joshi. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. *Recent trends in Meaning-Text Theory*, 39 :167, 1997. 129

[Rog96] James Rogers. A model-theoretic framework for theories of syntax. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 10–16. Association for Computational Linguistics, 1996. 10

[Rog03] James Rogers. Syntactic structures as multi-dimensional trees. *Research on Language & Computation*, 1(3) :265–305, 2003. 20

[Roz97] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1 : Foundations*. World Scientific, 1997. 146

[RVSW95] Owen Rambow, K. Vijay-Shanker, and David Weir. D-tree grammars. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 151–158. Association for Computational Linguistics, 1995. 17, 129

[SAJ88] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars : application to tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 578–583. Association for Computational Linguistics, 1988. 1, 10, 11, 12, 67

[SB09] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. *Non-transformational Syntax : A Guide to Current Models. Blackwell, Oxford*, 2009. 9, 13, 67, 128

[SCdlCB06] Benoît Sagot, Lionel Clément, Éric Villemonte de la Clergerie, and Pierre Boullier. The Lefff 2 syntactic lexicon for French : architecture, acquisition, use. In *LREC*, Gênes, 2006. 43

[Sch11] Natalie Schluter. *Treebank-Based Deep Grammar Acquisition for French Probabilistic Parsing Resources*. PhD thesis, Dublin City University, 2011. 184

[SCJ08] Libin Shen, Lucas Champollion, and Aravind K. Joshi. Ltag-spinal and the treebank. *Language Resources and Evaluation*, 42(1) :1–19, 2008. 17

[SF05] Kathrin Spreyer and Anette Frank. The tiger 700 rmrs bank : Rmrs construction from dependencies. In *Proceedings of LINC 2005*, pages 1–10, 2005. 132, 171

[SG05] Djamé Seddah and Bertrand Gaiffe. Des arbres de dérivation aux forêts de dépendance : un chemin via les forêts partagées. In *Actes de TALN*, 2005. 135

[Shi85] Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3) :333–343, 1985. 138

[Shi06] Stuart M. Shieber. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 377–384, 2006. 126, 129

[SS90] Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th conference on Computational linguistics-Volume 3*, pages 253–258. Association for Computational Linguistics, 1990. 129

[SS94] Yves Schabes and Stuart M. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1) :91–124, 1994. 129

[Tes34] Lucien Tesnière. Comment construire une syntaxe. *Bulletin de la Faculté des Lettres de Strasbourg*, 7 :219–229, 1934. 13

[vB86] Johan van Benthem. *Categorial Grammar*, chapter 7, pages 123–150. Studies in linguistics and philosophy. D. Reidel Pub. Co., 1986. 73

[vB91] Johan van Benthem. *Language and Action : Categories, Lambdas and Dynamic Logic*. North-Holland, 1991. 73

[vdEM03] Karel van den Eynde and Piet Mertens. La valence : l'approche pronominale et son application au lexique verbal. *Journal of French Language Studies*, 13(01) :63–104, 2003. 43, 166, 168, 178, 187

[vdWH88] Ton van der Wouden and Dirk Heylen. Massive disambiguation of large text corpora with flexible categorial grammar. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 694–698. Association for Computational Linguistics, 1988. 73

[VS92] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4) :481–517, 1992. 2, 20

[VSJ88] K. Vijay-Shanker and Aravind K. Joshi. Feature structures based tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 714–719. Association for Computational Linguistics, 1988. 20, 26

[VSW94] K. Vijay-Shanker and David J. Weir. The equivalence of four extensions of context-free grammars. *Theory of Computing Systems*, 27(6) :511–546, 1994. 136

[XTA95] The XTAG Research Group. A lexicalized tree adjoining grammar for english. Technical Report IRCS Report 95-03, The Institute for Research in Cognitive Science, University of Pennsylvania, 1995. 43, 58, 130, 133

## Résumé

Les travaux de cette thèse portent sur l'analyse syntaxique et sémantique de la phrase, en utilisant pour l'analyse syntaxique un formalisme grammatical lexicalisé polarisé et en prenant comme exemple les grammaires d'interaction. Dans les formalismes grammaticaux lexicalisés, les polarités permettent de contrôler explicitement la composition des structures syntaxiques. Nous exploitons d'abord le besoin de composition exprimé par certaines polarités pour définir une notion faible de réduction de grammaire applicable à toute grammaire lexicalisée polarisée. Nous étudions ensuite la première phase de l'analyse syntaxique des formalismes lexicalisés : l'étiquetage grammatical. Nous exploitons là encore le besoin de composition de certaines polarités pour concevoir trois méthodes symboliques de filtrage des étiquetages grammaticaux que nous implantons sur automate. Nous abordons enfin l'interface syntaxe-sémantique des formalismes lexicalisés. Nous montrons comment l'utilisation de la réécriture de graphes comme modèle de calcul permet concrètement d'utiliser des structures syntaxiques riches pour calculer des représentations sémantiques sous-spécifiées.



**Mots-clés:** étiquetage grammatical, interface syntaxe-sémantique, réécriture de graphes, polarités, grammaires d'interaction, structures de dépendance.



## Abstract

This thesis focuses on the syntactic and semantic analysis of sentences, when using, for the syntactic analysis, a polarized lexicalized grammatical formalism and taking interaction grammars as an example. In lexicalized grammatical formalisms, polarities enable to explicitly control the composition of syntactic structures. First we use the need for composition expressed by some polarities to define a weak notion of grammar reduction that is applicable to any polarized lexicalized grammar. Then, we study the first step of the syntactic analysis in lexicalized formalisms : supertagging. We use once again the need for composition expressed by some polarities to design three symbolic methods for filtering supertaggings that we implement on automata. Finally, we look at the syntax-semantics interface of lexicalized formalisms. We show how the use of graph rewriting as the model of computation concretely enables to use rich syntactic structures to compute underspecified semantic representations.



**Keywords:** supertagging, syntax-semantics interface, graph rewriting, polarities, interaction grammars, dependency structures.