Calculs de représentations sémantiques et syntaxe générative : les grammaires minimalistes catégorielles

Maxime Amblard

$N°$d'ordre : 3437

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Maxime AMBLARD

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : Informatique

Calculs de représentations sémantiques et syntaxe générative : les grammaires minimalistes catégorielles

---

Soutenue le : 21 Septembre 2007

Après avis de :

Mme	Isabelle TELLIER	Maître de conférence,	habilitée
M.	Uwe MÖNNICH	Professeur	**Rapporteurs**

Devant la commission d'examen formée de :

Mme TELLIER Isabelle....	MdC	univ. Lille 3	**Rapporteur**
M. MÖNNICH Uwe.....	Professeur	univ. Tübingen	**Rapporteur**
M. RETORÉ Christian	Professeur	univ. Bordeaux 1	**Directeur de thèse**
M. LECOMTE Alain.....	Professeur	univ. Paris 8	**Directeur de thèse**
M. SÉNIZERGUES Géraud	Professeur	univ. Bordeaux 1	**Examinateur**
M. ASHER Nicholas	DR	CNRS / Irit	**Examinateur**

Membres invités

M. KOBELE Gregory.....	Docteur	UCLA	**Invité**

– 2007 –

Remerciements

Je tiens à remercier chaleureusement Isabelle Tellier et Uwe Mönich qui m'ont fait l'honneur de rapporter mon manuscrit de thèse. Les échanges que nous avons pu avoir m'ont grandement enrichi et je leur exprime toute ma gratitude pour l'intérêt qu'ils ont porté à mes travaux.

Je tiens tout autant à remercier Géraud Sénizergue, Nicholas Asher et Gregory Kobele d'avoir accepté de participer à mon jury de thèse.

Mes remerciements vont également à mes deux directeurs de thèse Christian Retoré et Alain Lecomte qui m'encadrent depuis mon DEA. Christian a su me convaincre de ne jamais baisser les bras, tout en ayant de cesse de me m'ouvrir à de nouvelles questions et perspectives. La grande liberté qu'il m'a laissé pendant ces années font la grande diversité des thèmes abordés ici. Alain m'a apporté beaucoup dans la compréhension de la langue et de sa diversité. Il a également su me remettre sur les rails quand les chemins prenaient des directions trop multiples. Le couple qu'ils ont formé pour m'encadrer a su trouver un équilibre enthousiasmant quand, parfois, la recherche conduit à des moments de doutes profonds.

Je tiens également à remercier les actuels et anciens membres de l'équipe SIGNES avec qui les conversations ont toujours fait avancer ma réflexion, en particulier Richard Moot (merci pour les multiples relectures en anglais), Patrick Henry, Irène Durand et Lionel Clément. Je ne peux que remercier, très chaleureusement, pour leur confiance et leur soutien permanent, nos collègues de Bordeaux 3 dont Christian Bassac, Henri Portine et Joan Busquets. Mes remerciements vont également à Renaud Marlet auquel je réitère mes excuses pour lui avoir presque cassé le nez avec une raquette de badmington.

Surtout, ce manuscrit n'aurait pu trouver sa forme sans l'aide précieuse, efficace et généreuse de deux collègues et amis de qui j'ai beaucoup appris Sylvain Salvati et Greg Kobele. Je leur exprime très sincèrement toute ma gratitude.

Je souhaite aussi remercier les membres de la communauté que j'ai eu l'opportunité de rencontrer et dont les travaux ont poussé les miens : Willemijn Vermaat, Michael Morgaat, Jens Michaelis, Edward Stabler, John Hale, Matteo Capelletti, Hans-Martin Gartner.

Enfin, mes pensées vont vers mes "aînés", les anciens thésards de Christian : Roberto Bonato et Yannick Le Nir.

Mes remerciements scientifiques ne sauraient être complets sans remercier l'équipe administrative et les membres du laboratoire, ainsi que mes collègues de l'ENSEIRB, avec qui j'ai passé de précieux moments ces dernières années.

Je profite également de l'occasion pour remercier mes collègues de thèse Bilel Derbel, Olivier Bernardi et Jérémie Chalopin avec qui nous avons oeuvré (passionnément) pour la cause locale, Aïda Ouangraoua et Anthony Don, Jocelyn Fréchot, Julien Bernet. La liste serait bien longue et tout ceux qui ne sont pas cités sont, pour moi, également remerciés.

Merci à ma *team* de relecteurs, au premier rang desquels je me dois de citer helene (au moins 3, voire 4 fois le manuscrit en entier !), Amélie, Sébastien, Marie, Marie-Laure, etc.

Merci à mes amis et soutien pour les soirées de support moral, les évasions intellectuelles, les recentrages politiques, les coups de téléphone, pour ce qui fait aussi le reste de la vie. Merci helene, Sébastien, Laurent et Aliénor, Sandrine et Benjamin, Magali et Cyriaque, Marie et Agnès, Marie-Laure et Patrick, Antoine, Marc, Leslie, Sarah, Benoit, Alexandra, Vincent, François, Nathalie, Flo et Pascal, Philippe, Jules, Philippe D. (qui savait depuis si longtemps) (dans tous les ordres) + Socrate.

Merci à Francine Delmer pour les divagations culturelles sur le chemin de la science. Merci à tous les autres que je ne peux pas citer ici, qu'ils sachent que je les remercie profondément de continuer à m'accompagner.

Merci à ma famille pour son soutien au combien nécessaire durant ces années.

Merci à mon helene d'avoir supporté l'intrusion dans le quotidien du travail de recherche, de la non-disponibilité, de la longue période de rédaction. Merci de m'avoir soutenu et porté jusqu'ici.

À ma famille toute entière et par dessus tout mon helene.

Résumé / Abstract

Les travaux de cette thèse se situent dans le cadre de la linguistique computationnelle. La problématique est de définir une interface syntaxe / sémantique basée sur les théories de la grammaire générative.

Une première partie, concernant le problème de l'analyse syntaxique, présente tout d'abord, la syntaxe générative, puis un formalisme la réalisant : les grammaires minimalistes de Stabler. À partir de ces grammaires, nous réalisons une étude sur les propriétés de l'opération de fusion pour laquelle nous définissons des notions d'équivalence, ainsi qu'une modélisation abstraite des lexiques.

Une seconde partie revient sur le problème de l'interface. Pour cela, nous proposons un formalisme de type logique, basé sur la logique mixte (possédant des connecteurs commutatifs et non-commutatifs), qui équivaut, sous certaines conditions, aux grammaires de Stabler. Dans ce but, nous introduisons une normalisation des preuves de cette logique, normalisation permettant de vérifier la propriété de la sous-formule. Ces propriétés sont également étendues au calcul de Lambek avec produit.

À partir de l'isomorphisme de Curry-Howard, nous synchronisons un calcul sémantique avec les preuves réalisant l'analyse syntaxique. Les termes de notre calcul font appel aux propriétés du $\lambda\mu$-calcul, ainsi qu'à celles de la DRT (*Discourse Representative Theory*).

Une dernière partie applique ces formalismes à des cas concrets. Nous établissons des fragments d'une grammaire du français autour du problème des clitiques.

mots cléfs : grammaires génératives, interface syntaxe/sémantique, isomomrphisme de Curry-Howard, λ-calcul, $\lambda\mu$-calcul, logique linéaire, langages formels, grammaires catégorielles, grammaires minimalistes, types de Montague.

These works are parts of the framework of computational linguistic. We focus on the syntax/semantic interface based on generative grammars.

The first part treats of syntax analysis. We introduce Generative Grammars, which are a theoretical approach of syntax, and, then, Stabler minimalist grammars which are a formalism based on generative theory. We analyse the merge operation and we define equivalence for merge and an abstract modelling of lexicon.

The aim of the second part is now the syntax/semantic interface. We introduce a new logic formalism based on mixed logic (both commutative and non-commutative connectors). This formalism is equivalent to GM without SMC. We propose a normalisation (a non canonical one) for mixed logic which checks the sub-formula property. Then, these properties are both checked for Lambek calculus with product.

Syntactic analysis is performed by proof. On this proof, we synchronise a semantic calculus, using Curry-Howard isomorphism. Terms are based on $\lambda\mu$-calculus, and include DRT properties (Discourse Representative Theory).

The last part treats some linguistic phenomena. We focus on the cliticisation in french and propose a categorial minimalist grammar for french.

key words : generative syntax, syntax/semantic interface, Curry-Howard isomorphism, λ-calculus, $\lambda\mu$-calculus, linear logic, formal languages, categorial grammars, minimalist grammars, Montague semantic.

Table des matières

Remerciements i

Résumé / Abstract iii

Abréviations xi

Introduction 1

I Modélisation Syntaxique 11

1 Le programme minimaliste 15
1.1 Fondamentaux de la théorie générative . . . 15
1.2 Pour une histoire . . . 17
1.2.1 La phase combinatoire . . . 18
1.2.2 La phase cognitive . . . 20
1.2.3 Le Minimalisme . . . 23
1.3 Le programme minimaliste . . . 24
1.3.1 Système computationnel . . . 24
1.3.2 Le schéma de dérivation . . . 26
1.3.3 Règles du système calculatoire . . . 27
1.4 Sur la liberté . . . 30

2 Les grammaires minimalistes 31
2.1 Préliminaires . . . 31
2.1.1 Alphabets gradués et arbres . . . 32
2.1.2 Arbres minimalistes . . . 33
2.2 Relations entre sous-arbres . . . 33
2.3 Structures linguistiques . . . 39
2.3.1 Tête . . . 40
2.3.2 Projection maximale . . . 41
2.3.3 Complément et Spécifieur . . . 45
2.4 Grammaires minimalistes . . . 46
2.4.1 Définition . . . 46
2.4.2 Les traits . . . 46
2.4.3 Opérations . . . 48

2.4.4 Autres opérations possibles . . . 50
2.5 Exemples d'utilisation des GMs . . . 53
2.5.1 Phrase standard . . . 53
2.5.2 Phrase question . . . 56

3 Grammaires 59
3.1 Grammaires et Hiérarchie de Chomsky . . . 59
3.2 Les dérivations comme des files . . . 62
3.3 Transcription des MGs vers les MCFGs . . . 69
3.4 Arbre de fusion . . . 72
3.4.1 Arbres de fusion . . . 73
3.4.2 Équivalence entre GMs pour la fusion . . . 74
3.4.3 Équivalence des GMs et des CFGs . . . 77
3.5 Représentation abstraite d'un lexique . . . 83
3.6 conclusion . . . 86

4 Exemple de lexiques et étude de la capacité générative 87
4.1 Représentation simplifiée des GMs . . . 88
4.2 Phrase sur un compteur . . . 88
4.3 Compteurs enchâssés . . . 96
4.4 Duplications . . . 101
4.5 Duplication multiple . . . 105
4.6 Phrase de Fibonacci . . . 112
4.7 Conclusion . . . 118

II Formalismes logiques et Interface syntaxe/sémantique 119

5 Normalisation de la logique mixte 123
5.1 Présentation . . . 123
5.1.1 Logique linéaire non commutative . . . 124
5.1.2 Motivation pour ce type de calcul . . . 125
5.2 La logique mixte - Partially Commutative Linear Logic . . . 126
5.2.1 Formules et ordre . . . 126
5.2.2 Définitions générales . . . 128
5.3 Normalisation du Lambek avec produit . . . 129
5.3.1 Propriétés de $L_{\odot}$ . . . 129
5.3.2 Normalisation de $L_{\odot}$ . . . 132
5.3.3 Propriété de la sous-formule pour $L_{\odot}$ . . . 134
5.4 Normalisation des preuves de la logique mixte . . . 138
5.4.1 Propriétés de la logique mixte . . . 138
5.4.2 Normalisation de la logique mixte . . . 144
5.4.3 Propriété de la sous-formule pour la logique mixte . . . 149
5.5 Conclusion . . . 151

6 Les grammaires minimalistes catégorielles **153**
6.1 Grammaires Minimalistes Catégorielles . . . 154
6.1.1 Définitions . . . 154
6.1.2 Dérivations . . . 155
6.2 GM et GMCs . . . 160
6.2.1 Traduction de lexiques . . . 161
6.2.2 Inclusion GM/GMC . . . 161
6.3 Exemples . . . 166
6.3.1 Questions . . . 166
6.3.2 Forme passive . . . 168
6.3.3 Remnant Movement . . . 170

7 Interface syntaxe et sémantique **173**
7.1 Prérequis pour la sémantique computationnelle . . . 175
7.1.1 Logique des prédicats et théorie des modèles . . . 175
7.1.2 Types de Montague . . . 176
7.1.3 λ-calcul . . . 177
7.1.4 $\lambda\mu$-calcul . . . 179
7.2 Interface syntaxe/sémantique . . . 181
7.2.1 Morphisme de types . . . 182
7.2.2 Décomposition de la sémantique . . . 183
7.2.3 Règles de l'interface . . . 186
7.3 Exemples . . . 187
7.3.1 phrase Sujet-Verbe-Objet . . . 188
7.3.2 Questions . . . 191
7.3.3 Forme passive . . . 194
7.4 Conclusion . . . 197

8 Fragments d'une grammaire du français **199**
8.1 Définitions linguistiques . . . 200
8.1.1 Définition et propriétés des clitiques . . . 200
8.1.2 Typologie des pronoms clitiques faibles . . . 201
8.1.3 Approches syntaxiques . . . 203
8.1.4 Interprétation sémantique . . . 203
8.2 Analyse syntaxique des clitiques . . . 204
8.2.1 Version de Ed Stabler . . . 204
8.2.2 Extension : clitiques génitif, oblique et nominatif . . . 208
8.2.3 Négation . . . 211
8.2.4 Exemple d'analyse . . . 214
8.3 contrepartie sémantique . . . 218
8.4 Autour du groupe verbal - cas simple . . . 222
8.4.1 Montée sur les auxiliaires . . . 222
8.4.2 Impératif . . . 223
8.5 Verbes enchâssés . . . 225
8.5.1 Infinitif . . . 226
8.5.2 Verbe à contrôle . . . 227

8.5.3 Subordonnées relatives . . . 230
8.5.4 Sémantique des verbes enchâssés . . . 231
8.6 Autour du groupe nominal . . . 232
8.6.1 Dislocation . . . 233
8.6.2 Extraction d'un groupe nominal . . . 233
8.7 Conclusion . . . 234

Conclusion **237**
9.1 Contributions . . . 237
9.2 Perspectives . . . 240

A Définition des GM sous forme de chaînes **257**

B Implémentation pour les GMs **259**
B.1 Analyseurs syntaxiques . . . 259
B.1.1 Parser de John Hale . . . 259
B.1.2 Guillaumin et al. . . . 262
B.2 Autres implémentations . . . 263
B.2.1 Représentations sous forme de circuit . . . 263
B.2.2 Module de sémantique simple pour le parser de J. Hale . . . 266
B.3 Conclusion . . . 267

C Extension des dérivations par file et compteur en double fonction : le cas de $a^n b^{2^n}$ **269**

D Systèmes de règles **273**
D.1 Grammaires Minimalistes . . . 273
D.2 Logique mixte . . . 274
D.2.1 Calcul de Lambek avec produit . . . 274
D.2.2 Logique mixte . . . 275
D.3 Grammaires Minimalistes Catégorielles . . . 275
D.4 Interface syntaxe-sémantique . . . 276
D.4.1 $\lambda\mu$-calcul . . . 276
D.4.2 $\lambda\mu$-DRS . . . 277

E Analyse avec une GM de la phrase "Je ne la lui donne pas" **279**

F Une modélisation de la Langue des Signes Française par les GMs **289**
F.1 Présentation de la LSF . . . 289
F.1.1 Historique . . . 289
F.1.2 Particularités . . . 290
F.1.3 Ordre des signes en LSF . . . 291
F.2 Cadre théorique . . . 293
F.2.1 Les grammaires minimalistes . . . 293
F.2.2 Signème . . . 298
F.3 Modélisation . . . 299
F.4 Exemple de dérivation . . . 301

Table des figures

1 Modélisation générale des travaux. 6

2 Structure du programme minimaliste de Chomsky. 25
3 Relations spécifieur-tête-complément. 26
4 Fusion simple. 28
5 Structure d'une phrase simple. 28
6 Exemple de transformation de phrase par déplacement d'un constituant. . . 29

7 Arbre et relation internes. 33
8 Relation de dominance. 34
9 Relation de précédence. 35
10 Un arbre minimaliste t, les termes le composant et des exemples de relations entre les termes. 40
11 Structure de la liste de traits d'un item lexical. 48
12 Représentation sous forme d'arbre de la fusion. 49
13 Représentation sous forme d'arbre du déplacement. 50
14 Représentation graphique du déplacement faible. 51

15 Hiérarchie de Chomsky. 61
16 Hiérarchie de Chomsky et classe des langues naturelles. 62
17 Représentation du lexique pour le langage a^{2^n}. 64
18 Premier cas échouant lors de la dérivation de a^{2^n}. 67
19 Deuxième cas échouant lors de la dérivation de a^{2^n}. 68
20 Capacité générative des GMs. 69
21 Arbres de dérivation et de fusion de "Quel train prend Pierre". 75
22 Forme normale de fusion. 77
23 CFG $\longrightarrow$ GM. 81
24 Équivalence pour la fusion. 82
25 Exemple de circuit avec étiquette sur les arcs. 85
26 Exemple de circuit abstrayant un lexique. 85

27 Représentation abstraite du lexique pour deux terminaux sur un compteur. 90
28 Premier cas d'échec dans l'analyse de phrase sur un compteur. 95
29 Représentation du lexique pour un compteur et deux terminaux. 98
30 Lexique pour la duplication inverse. 102
31 Lexique pour la duplication inverse. 106
32 Dérivation pour les phrases de Fibonacci. 113

33 Lexique pour $a^{F(n)}$ où $F(n)$ la fonction de Fibonacci. 114

34 Règles de la logique mixte. 127
35 Structure des preuves autorisant la montée du produit dans $L_{\odot}$. 129
36 Exemples de preuve qui ne sont pas sous forme normale 133
37 Structure des preuves autorisant la montée du produit dans la logique mixte. 139
38 Exemples de preuve de la logique mixte qui ne sont pas sous forme normale 145
39 Descente du produit sur la règle $\backslash_e$ dans la logique mixte. 147

40 Premières étapes de l'analyse de la phrase : "Les enfants lisent un livre". . 159

41 Exemples de types. 177
42 calculs dans les GMCs. 187

43 Première version du traitement des clitiques. 205
44 Cluster de cliticisation. 210
45 Traitement des clitiques nominatifs. 211
46 Traitement standard de la négation. 213
47 Inflexion impérative. 225
48 Verbes pronominaux. 225
49 Infinitif. 227
50 Semi-auxiliaires. 230
51 Subordonnées relatives. 231
52 Autours des DPs. 234

53 Processus vers un calcul sémantique 266

54 Représentation du lexique pour a^{2^n}. 269

55 Structure de la liste de traits d'un item lexical 295
56 Représentation sous forme d'arbre de la fusion 296
57 Représentation sous forme d'arbre du déplacement 297
58 Structure d'une analyse en langue des signes française. 301

Abréviations

AIC	Adjunct Island Condition
CFG	Grammaires hors-contexte (context-free)
CYK	Cocke, Younger et Kasami
DAG	Graphe Acyclique Dirigé
DRS	Discourse Representation Structure
DRT	Discourse Representation Theory
DP	Determinal phrase - groupe déterminant
FL	Forme Logique
FP	Forme Phonologique
GB	Government and Binding
GC	Grammaires Catégorielles
GM	Grammaires Minimalistes
GMC	Grammaires Minimalistes Catégorielles
GU	Grammaire Universelle
HPSG	Head-Driven Phrase Structure Grammar
LCFRS	Linear Context-free Rewriting System
LFG	Lexico-Functional Grammars
MCFG	Multiple Context-Free Grammars
MCFG	Multiple Context-Free Grammars
MCS	Midly Context Sensitive
PCIMLL	Partially Commutative Intuitionnistic Multiple Linear Logic
PP	Principes et Paramètres
RTG	Regular Tree Grammar
SMC	Shortest move condition
SPIC	Specifier Island Condition
TAG	Tree Adjoining Grammars
TAG	Traitement Automatique des Langues
UTAH	Uniform Theta Assignement Hypothesis

Il veut, mais il ne sait pas quoi. Alors, il crie, pour dire qu'il ne sait pas ce qu'il veut. Et il crie aussi pour savoir, pour que, dans le flot de ses paroles, il sorte de lui-même ce renseignement sur ce qu'il veut.

Marguerite Duras

Introduction

Au croisement de plusieurs disciplines.

La linguistique computationnelle est au croisement de plusieurs disciplines : les mathématiques, l'informatique et la linguistique. L'apport simultané de ces trois champs d'investigations, avec leurs contraintes et leurs nécessités, fait de la linguistique computationnelle une thématique riche, contenant de nombreuses branches (qui ont parfois du mal à se rencontrer), dont le but est la résolution de problèmes autour des langues que nous utilisons. Les objectifs sont variés et font appel à des pans différents des apports possibles de chacune de ces disciplines.

La linguistique est le sujet d'étude de ces travaux. Elle a pour objet la description de phénomènes apparaissant dans les langues humaines (on préférera la terminologie de *langue naturelle*). À partir de ces théories nous proposons des modélisations formelles et automatisées. La linguistique se subdivise encore, donnant des angles d'approche distincts. En effet, les études synchroniques et diachroniques ont pour point de vue une langue unique (son état à un moment donné et ses évolutions structurelles dans le temps). Les études théoriques, quant à elles, se proposent de comprendre la création des structures permettant la description de ces langues, ainsi que la conceptualisation de ce qui est commun à toutes les langues (formalisation d'une grammaire universelle). Enfin la linguistique contextuelle étudie les relations des langues par rapport au monde.

Les mathématiques apportent des outils de modélisation formelle à ces théories. Les principales contributions viennent de la logique, de la théorie des langages et des probabilités. Notre volonté de communiquer se base sur un système que nous voulons cohérent et la logique permet de vérifier la conservation de cette cohérence. La logique est une discipline difficile à classifier, elle fait à la fois partie des mathématiques et de la philosophie (sa position dans le système universitaire n'est pas la même en fonction des pays). De plus, son apparition se situe exactement à la jonction de ces trois disciplines, représentant notre capacité à raisonner.

Tout comme la logique, la théorie des langages nous donne des modélisations formelles de la capacité générative des grammaires, ce qui permet d'identifier la classe de langage des langues naturelles et de vérifier la possibilité de reconnaître ces structures par nos formalismes.

Ces concepts font également partie de l'informatique, qui est nécessaire par exemple lorsque l'on veut réaliser des implémentations fonctionnelles pour une utilisation grand public (prenant alors de nombreuses situations en considération). Nous devons laisser de côté ce qui s'apparente trop à la modélisation pour utiliser les techniques d'algorithmique et les probabilités afin d'optimiser l'obtention des résultats. A contrario, la formalisation nécessite à la fois l'usage de théories linguistiques et de théories à la frontière des

mathématiques et de l'informatique, dans ce cadre, on citera la validation de modèles syntaxiques par la possibilité d'utiliser des algorithmes d'apprentissage à partir d'exemples positifs. Dans le même temps, et comme ces travaux le montrent, mettre en place des formalismes de linguistique computationnelle induit le développement des théories informatiques.

Nous dissocions la possibilité de réaliser des analyses (à partir d'un énoncé, trouver l'ensemble des modélisations associées) de celle de faire de la génération (à partir d'un ensemble de modélisations, trouver l'énoncé associé). Les problématiques sont très différentes puisque d'un côté, on collecte de l'information traduite en structure alors que de l'autre on cherche les parties de structures signifiantes devant être interprétées. On peut les considérer comme opposées bien qu'elles soient fortement corrélées et complémentaires. L'idée de traduction automatisée nécessite à la fois d'être capable d'analyser un nouvel énoncé en une structure donnée puis de produire, à partir de la structure, un énoncé dans un autre système de représentation. Ces transformations se font à plusieurs niveaux que l'on regroupe en une organisation que nous allons présenter.

Une caractérisation des niveaux d'analyses linguistiques.

Traditionnellement, on classe les différents niveaux d'analyse sur lesquels on peut intervenir. Nous n'essayons pas d'identifier une hiérarchie entre les domaines, mais plutôt de montrer les limites posées par une question spécifique. Ce classement permet également d'identifier les relations qui peuvent être mises à jour entre les différents niveaux.

Notre sujet est proche de la grammaire mais pas dans le sens de celle enseignée à l'école élémentaire qui n'appartient pas au domaine linguistique. Cette dernière a pour objet de donner une norme d'une langue alors que la linguistique en étudie les pratiques. Nous appelons *grammaire* un système (formel) permettant de décrire le fonctionnement d'un phénomène.

On identifie les différents niveaux de la langue par le classement suivant :

- phonétique : grammaire des sons (concrets) d'une langue.
- phonologie : grammaire du son (abstrait) - phonème.
- prosodie : grammaire du rythme, de l'accent, du ton.
- morphologie : grammaire du mot - morphème. On distingue deux types de morphologie :
  - morphologie dérivationnelle : formation des mots (préfixe, suffixes, *etc.*) avec modification de catégorie possible.
  - morphologie flexionnelle : déclinaison ou conjugaison des mots, généralement sans changement de catégorie.
- syntaxe : grammaire de la phrase - syntagme.
- sémantique : grammaire du sens. On distingue également deux types de sémantique.
  - sémantique vériconditionnelle : déterminer les conditions de vérité d'un énoncé.
  - sémantique compositionnelle : calcul du sens d'un énoncé.
- pragmatique : intention de communication.

Plus le sujet d'étude se situe en début de liste, plus il s'approche de la notion de *signifiant*. À l'inverse, plus il est en fin de liste, plus le sujet renvoie au *signifié*. On définit comme *interface* chaque point de jonction de la liste.

On a pour habitude de considérer que chacun de ces niveaux d'analyse est un module autonome car composé de problèmes homogènes. Cependant, certains formalismes tentent de traiter simultanément plusieurs d'entre eux. S'il est vrai que des phénomènes appartiennent clairement à un niveau plutôt qu'à un autre (par exemple la portée des quantificateurs à la sémantique), il n'est pas évident de proposer les frontières de la répercussion de ces phénomènes : l'influence de la portée de quantification peut-elle ou non apparaître en phonologie ? Nous sommes tentés de répondre par la positive. La vision modulaire trouve des arguments mais des cas d'étude montrent la nécessité d'importantes communications entre chaque niveau. Ainsi, nous préférons la vision où les phénomènes étudiés sont transversaux aux différentes classes.

Applications pour la linguistique computationnelle.

La question de l'intégration des outils issus de la linguistique computationnelle dans le monde réel, permet de définir ce vers quoi nos travaux tendent. Le Graal est certainement la traduction automatique. Le problème principal pour obtenir un tel outil est qu'il faut maîtriser toutes les strates d'analyse évoquées précédemment. Pour le moment, l'avancée des travaux est loin de laisser présager qu'une proposition de traduction automatique performante puisse être développée à moyen terme. A contrario, si l'on restreint le domaine d'application et les structures utilisées, il est possible de proposer des outils pertinents. Certains d'entre eux sont accessibles sur l'Internet. Il y a encore quelques années, ces systèmes n'utilisaient que de la traduction mot-à-mot, nous assistons actuellement à l'apparition de nouvelles propositions qui utilisent des relations syntaxiques et des prémices de modélisations sémantiques.

Cependant, sans atteindre ce niveau de performance, des outils basés sur les recherches modernes sont largement développés, en particulier, des restrictions de la traduction à des domaines spécifiques, repérage d'expressions idiomatiques ou de propositions de traduction assistée. Toutes les extractions de sémantique partielle permettent de modéliser une partie de la logique sous-jacente à l'information, ce qui peut se traduire par des systèmes d'interrogations de bases de données en langue naturelle (plutôt qu'en utilisant un langage spécifique) ou l'obtention de résumés de texte. Du côté de la génération automatique de texte, un exemple réussi et connu est la génération de bulletin météorologique, [CKPR72]. L'un des grands enjeux actuels de la sémantique est la recherche d'informations dans une grande masse de données qui n'a pas de cohérence interne, ni de véracité globale. Enfin, de nombreux outils autour de la génération et de la reconnaissance de la parole sont développés afin d'assister au mieux les utilisateurs (téléphone portable, personnes déficientes, etc.).

Toutes ces applications réalistes ou rêvées pour le moment trouvent pleinement leur place dans l'évolution du monde. Nos échanges entre pays, la possibilité de communiquer directement avec le reste de la planète fait que la question du traitement des langues et de la bonne compréhension émerge, et ce de toutes parts, tant pour les communications personnelles que professionnelles (pour lesquelles les enjeux sont différents). L'institution la plus en demande d'outils multilingues est certainement la Communauté Européenne. On comprendra facilement la problématique posée par une telle structure et les différences d'approche avec nos communications personnelles, tout en relevant la nécessité de l'exactitude du résultat.

Théories et modèles.

En linguistique computationnelle, une distinction importante est faite entre la théorie qui est modélisée et le modèle qui permet la modélisation. On identifie deux grandes théories en syntaxe : la grammaire générative et les grammaires de dépendance. Nous nous concentrons sur la théorie générative sur laquelle nous reviendrons dans le chapitre suivant. La dépendance propose des systèmes formels basés exclusivement sur les dépendances entre les mots des phrases. Cette théorie a été largement développée par Mel' čuk, [MP87], puis reprise récemment dans le travaux de Kahane et Gerdès, [Kah06a] ou [GK06]. Dans l'ensemble de nos travaux, Nous nous sommes focalisé sur la théorie générative, dont les grandes lignes sont reprises dans [Jac95].

Parallèlement, de nombreux modèles ont été proposés. Citons les plus utilisés ainsi que leurs particularités principales (on rappellera qu'actuellement la classe des langues naturelles est considérée comme peu sensible au contexte). On retrouvera dans [AB00] et [Ret00] des présentations détaillées de tous ces formalismes :

- Grammaires d'Unification, [PS87], [SKP84]. Ces grammaires ont de bonnes propriétés calculatoires, leur formalisation permet une implémentation robuste, notamment par le langage de programmation PROLOG, qui a été entre autre développé à ces fins. Un grande partie des développements sont repris par Abeillé dans [Abe93]. Cependant, ces grammaires sont récursivement énumérables.
- Lexico-Functional Grammars - LFG, introduites par Bresnan et Kaplan, [KB82b]. Les divers niveaux de représentation permettent d'obtenir la structure en constituant, l'ordre des mots dans la phrase et les relations fonctionnelles (sujet-prédicat). Ces grammaires sont une restriction des grammaires d'unification dont la classe de langage est *context-sensitive*, donc théoriquement plus adaptée à celle des langues naturelles.
- Head Grammars - HPSG, introduites par Pollard et Sag [SP87], sont l'un des formalismes les plus étudiés et développés ces dernières années. Les analyses sont basées sur les relations structurelles entre les éléments. Il n'y a qu'un seul niveau de représentation où l'information est partagée (morphologie, syntaxe, sémantique et pragmatique). Les propriétés formelles sont décrites dans un cadre homogène ce qui permet d'aborder globalement les problèmes théoriques et d'implémentation. C'est grammaires sont Turing-complet.
- Tree Adjoining Grammars - TAG, introduites par Joshi, [JLT75]. Une présentation détaillée à ensuite été publiée sous le même titre, [JS97]. Ce modèle est très utilisé pour son élégance mathématique et l'efficacité de l'analyse syntaxique. De plus, c'est une grammaire d'arbres, ce qui permet de dériver directement des analyses en constituants des énoncés. Il utilise l'opération d'adjonction qui permet d'interpréter aisément les dépendances à longue distance. Ce formalisme est l'un des plus utilisés. De nombreux travaux le présentent et le développent. Les derniers travaux qui ont adoptés des questions similaires aux nôtres sont présentés dans [Par07].
- Categorial Grammars - GC, introduites par Bar-Hillel, [BH53] - mais dont les prémices remontent à Ajdukiewicz [Ajd35b]. Ces grammaires ont alors pris leur forme actuelle sous l'influence des travaux de Lambek, [Lam58]. Les différentes évolutions sont présentées dans [Ret00]La syntaxe est dérivable des propriétés combinatoires

des unités. Les entrées encodent exactement leur comportement syntaxique, cependant la classe de langage engendrée est trop petite au regard de celle des langues naturelles. Ce type de grammaires possède un avantage important, c'est d'être basé sur un cadre logique, permettant d'obtenir facilement la structure prédicative de l'énoncé analysé grace à l'isomorphisme de Curry-Howard. Plusieurs enrichissements ont été proposés comme les MMCG - MultiModal Catégorial Grammars, [Moo96] (qui utilisent des modalités pour augmenter la capacité générative ou les CCG - Combinatory Catégorial Grammars, [Ste81]).

Plus récemment, une multitude de formalismes ont été proposés, nous souhaitons citer entre autres : les grammaires de propriétés (GP), [BB01] basées sur la résolution de contraintes, les grammaires d'intéractions, [Per00], qui utilisent l'unification de polarités tout comme les GUP - Grammaires d'Unification Polarisées, [Kah06b]. La liste devrait être bien plus longue et elle ne prétend nullement à l'exhaustivité ni à proposer une hiérarchie entre les formalismes mais plutôt montrer le grand nombre de points de vue autour des questions de modélisation de la langue.

L'adéquation entre théorie et formalisme reste souvent une question ouverte. Si des esquisses de réponses peuvent être proposées, il n'y a jamais de réponse définitive. Les objets d'étude étant les mêmes, il est normal de considérer que les réponses puissent être multiples.

Notre choix porte sur la grammaire générative du point de vue de l'analyse. Les critiques usuelles stigmatisent la confusion entre les différents niveaux de représentation ainsi que les problèmes de transformation qui semblent être ingérables. Comme nous l'avons discuté, la non-modularité n'est pas en soit un problème catégorique. Le but clairement affiché par cette théorie est d'associer du sens à un son, en utilisant la syntaxe comme fil conducteur de l'analyse. À partir de ce postulat, il est aisé de dissocier ce qui a trait à chaque niveau afin de clarifier les résultats.

D'un autre côté, les points importants de cette théorie sont les liens directs entre langues, et même entre phrases d'une langue. Les liens entre langues viennent du fait qu'on ne focalise pas l'étude sur un type de langue donnée et que l'on considère la variation entre deux langues comme pouvant être modélisée par une suite de descripteurs (principes et paramètres). Ces représentations permettent d'envisager le multilinguisme : pour toutes les langues, il existe des transformations permettant de passer de l'une à l'autre. D'autre part, ces transformations expliquent les liens qui apparaissent entre une phrase question (ne portant que l'objet) et sa version affirmative (qui peut être sa réponse). On peut ainsi passer de l'une à l'autre en modifiant uniquement la position de l'un des complément (celui d'objet).

Notre point de vue étant de l'informatique, nous ne considérons pas la grammaire générative en elle-même, mais travaillons sur une formalisation de celle-ci. De fait, l'introduction d'un cadre formel strict permet de lever la critique sur les problèmes liés à la calculabilité algorithmique de la théorie. En effet, la formalisation étant dérivationnelle, nous donnons dans les définitions intrinsèques les conventions autorisant ou non les transformations.

Dès à présent nous allons revenir en détail sur la thématique de nos travaux.

Présentation de la thématique des travaux de thèse.

La première question ouverte a été celle de proposer une interface entre calcul syntaxique et représentation de la sémantique des énoncés. Notre but est, à partir d'un formalisme permettant la reconnaissance syntaxique d'énoncés d'une langue naturelle basée sur la grammaire générative, de proposer un système calculatoire permettant d'obtenir automatiquement une représentation de son sens.

Pour cela, nous sommes partis de la proposition de formalisation introduite par Ed Stabler en 1997, [Sta97] du programme minimaliste de Noam Chomsky, [Cho95]. Ce programme donne une définition non plus représentationnelle mais dérivationnelle de la syntaxe. Il s'agit alors de modéliser chaque propriété et de faire en sorte que le résultat d'une analyse ne puisse être obtenu que par composition, selon des règles données, de ces propriétés. Ce postulat rend une formalisation en linguistique computationnelle possible (les spécifications de ce programme sont regroupées dans les GMs).

Nous appelons *sémantique des énoncés*, une formule en logique d'ordre supérieur représentant la structure prédicative de l'énoncé. L'interprétation de ces formules étant le contenu sémantique (ou informationnel) de l'énoncé lui-même. La procédure classique à partir des GCs pour obtenir ces formules est d'utiliser le λ-calcul et l'isomorphisme de Curry-Howard, [How80]. En effet, le λ-calcul, [Chu40], est un système calculatoire efficace pour obtenir des représentations des structures prédicatives et l'isomorphisme permet d'associer à un calcul logique le λ-calcul linéaire, ce qui assure que chacune des composantes n'est utilisée qu'une fois. La figure 1 modélise les concepts utilisés pour obtenir les formules logiques. L'utilisation de cet isomorphisme est la première flèche de ce schéma. Une fois cette correspondance établie, nous passons en argument de ces λ-termes d'autres λ-termes, qui eux, peuvent ne pas être linéaires. Ils représentent la sémantique des parties, c'est-à-dire la sémantique des mots. Cette procédure est largement utilisée par les grammaires catégorielles [Lam58]. Le développement du λ-calcul doit principalement à J.Y. Girard, [Gir87a], notamment pour la version typée de ce calcul que l'on peut retrouver dans [Kri90].

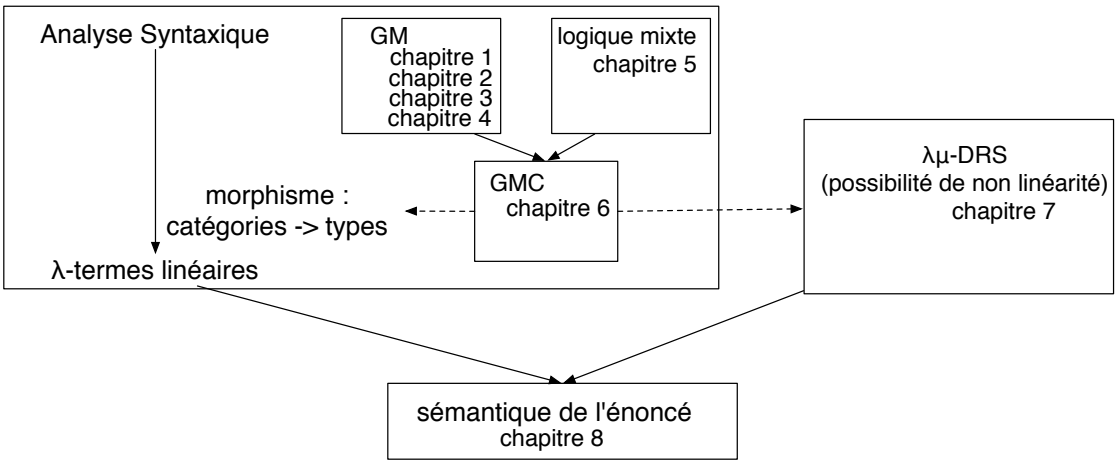


FIG. 1 – Modélisation générale des travaux.

Afin de construire les formules sémantiques nous ne pouvons pas directement proposer un système calculatoire. Plusieurs étapes sont nécessaires, d'abord nous avons travaillé sur

les GMs elles-mêmes, pour étudier leurs mécanismes internes et vérifier leur adéquation aux langues naturelles, puis nous avons été amenés à proposer l'utilisation d'un système logique dont un fragment définit un formalisme dans lequel les GMs sont incluses. Dans ce calcul, pour une analyse syntaxique, on associe à chaque mot une formule décrivant leur comportement syntaxique puis un calcul logique permet d'associer un type sémantique à la catégorie syntaxique des constituants. Si le résultat d'une analyse est une catégorie donnée, on considère la séquence comme étant une phrase d'une langue. Du côté de la sémantique, à chaque mot correspond un type et un terme qui modélise son apport sémantique. Chaque assemblage syntaxique entraîne dans le calcul sémantique une opération sur les termes sémantiques et, en fin d'analyse, on obtient une formule associée à la phrase de départ.

Maintenant que nous avons présenté les différentes notions permettant de situer nos travaux dans un cadre général, nous nous attachons à présenter la structure de ce manuscrit.

Plan de la thèse.
Le premier chapitre présente le programme minimaliste de Noam Chomsky (cette théorie linguistique est à la base du formalisme utilisé pour ces travaux). Pour cela, nous inscrivons dans une perspective historique l'évolution de la théorie générative. Cette démarche nous permet de donner les principaux arguments en sa faveur. Une seconde partie détaille les éléments qui composent le système, chacun d'entre eux trouvant sa correspondance dans les grammaires minimalistes (GMs), [Sta97].

Dans le second chapitre nous revenons sur l'ensemble des définitions mises en œuvre dans la formalisation des grammaires minimalistes - GMs. Ce chapitre préparatoire à la présentation des travaux de cette thèse expose ces grammaires sous la forme de termes algébriques. Grâce à cette caractérisation, nous définissons les propriétés sur les arbres minimalistes qui nous permettent de prouver des lemmes fréquemment admis et de donner une cohérence globale à ce système. Cependant, les perspectives de travail ont été nombreuses et nous ne conservons que celles utilisant une modélisation à partir des structures d'arbres et non sur les chaînes. Les deux présentations étant intrinsèquement liées, travailler sur l'une permet de comprendre comment les constructions sont réalisées dans l'autre. Afin de ne pas perdre le lecteur nous nous attachons à proposer des exemples complets d'analyses réalisées grâce à ce formalisme.

Dans le troisième chapitre, nous revenons sur quelques résultats connus autour des GMs (en particulier la transcription des GMs vers les *multiple context-free grammars*), puis nous présentons des travaux d'étude sur ce formalisme. Pour mettre en exergue les mécanismes des GMs, nous proposons une transcription d'une partie des GMs vers les grammaires hors-contexte (CFGs). Pour ce faire, nous introduisons la notion d'équivalence pour la fusion. Cette transcription permet non pas de reconnaître les mêmes langages que les GMs mais une sous-partie importante du calcul : la manière dont les éléments sont combinés lors d'une dérivation. À partir de ces sous-dérivations, nous envisageons deux usages différents :

- une modélisation abstraite des lexiques en utilisant les circuits, qui se révélera être un outil performant d'aide à la rédaction de lexique.

◦ un prétraitement pour l'analyse.

Revenons en un mot sur ce dernier point. Les GMs ont la particularité d'utiliser des entrées à phonologie vide, ce qui complexifie les analyseurs syntaxiques. Ils doivent tester tous les sous-ensembles contenant les entrées lexicales de l'énoncé avec l'union des parties du sous-ensemble des entrées à phonologie vide. Or, un grand nombre de ces sous-ensembles ne peuvent pas fournir d'analyse acceptante car les éléments ne sont pas combinables. Nous proposons un moyen pour les repérer facilement.

Dans le chapitre suivant, nous proposons plusieurs algorithmes permettant la rédaction automatique de lexique étant la base des GMs reconnaissant des langages formels. Grâce à ces derniers, nous pouvons analyser la capacité générative de ces grammaires, ainsi qu'identifier les mécanismes utilisés lors de la reconnaissance de phénomènes complexes. Nous exposons les GMs acceptant les phrases sur un compteur pour un nombre déterminé mais non borné de terminaux, puis sur un croisement de compteurs - appelé compteurs enchâssés. Nous poursuivons par la présentation de langage de copies. Enfin, nous présentons le cas des phrases de Fibonacci (un terminal à la puissance la fonction de Fibonacci).

Après avoir réalisé cette étude générale des GMs, nous reviendrons sur le problème de l'utilisation d'un cadre logique pour modéliser ces phénomènes syntaxiques. La première partie du travail consiste à trouver un cadre logique cohérent à même de modéliser les GMs. Ces idées ont vu le jour à partir de [BE96] puis ont commencées à prendre forme. La préface de [RS04] propose une synthèse sur cette question. La première tentative a été l'utilisation des grammaires de Lambek avec produit, [LR99] et [LR01] : les règles d'implication servant à modéliser la partie qui combine les éléments entre eux et l'élimination du produit servant à représenter les différents déplacements à l'aide d'hypothèses de la preuve vues comme des ressources.

Pour ce calcul, nous montrons dans un premier temps la possibilité d'utiliser une forme normale dans laquelle les produits sont *le plus haut possible* (ce qui permet de supprimer les parties inutiles du calcul) en introduisant la notion de k-redex-étendu, $k \in \mathbb{N}$ (redex classiques contenant d'autres règles). À partir de cette normalisation, nous prouvons la propriété de la sous-formule, qui nous assure une cohérence interne au calcul.

Mais un problème structurel se pose dans ce calcul : la non-commutativité du produit oblige les hypothèses à être introduites et utilisées dans un ordre donné, or les GMs ne sont pas aussi restreintes. Nous avons réalisé la même étude pour un cadre logique plus souple (ou plus complet) : la logique mixte, proposé par Philippe de Groote [dG98], qui possède à la fois des connecteurs commutatifs et non-commutatifs. On introduit une restriction d'ordre pour gérer simultanément les hypothèses. Nous proposons donc une normalisation pour cette logique. La dissociation des restrictions d'ordre n'étant pas assez fine, nous ne sommes pas en mesure de proposer une forme normale unique. Cependant, celle qui est proposée assure la minimisation du nombre de règles et vérifie la propriété de la sous-formule.

À partir de ce cadre logique, nous en utilisons un fragment pour modéliser les GMs. Les première propositions qui vont dans cette direction sont dues à C. Retoré et A. Lecomte, [LR99], [LR01]. La partie combinant les éléments est non-commutative et l'élimination

des hypothèses est, quant à elle, commutative (la restriction d'ordre étant appliquée systématiquement après chaque règle commutative). Nous proposons l'ensemble des règles de ce fragment qui définit un nouveau formalisme : les *grammaires minimalistes catégorielles* (GMCs). Nous prouverons que les langages des GMs sont inclus dans les langages des GMCs. Nous illustrerons l'utilisation de ce formalisme par des exemples.

Cette correspondance entre analyse syntaxique et cadre logique est la base nous permettant de proposer une interface syntaxe sémantique. À partir de ce nouveau formalisme nous pouvons utiliser l'isomorphisme de Curry-Howard qui associe un λ-calcul linéaire à nos énoncés de départ.

C'est la composante principale du chapitre suivant. En effet, comme nous l'avons mentionné, nous souhaitons obtenir un calcul automatisé nous fournissant des formules logiques représentant la sémantique des énoncés analysés (leur structure prédicative). Cependant, la mise en place d'un tel calcul nécessite la présentation d'autres concepts pour ces travaux : $\lambda\mu$-calcul, DRT et type à la Montague, [Mon73]. La théorie de Montague a été largement développée dans [Gam91a] et [Car96]. Pour sa part, la DRT a été introduite par H. Kamp, [KR93], et l'on trouvera une présenation détaillée dans [Cor02].

À partir des GMCs, on introduit un morphisme de types des formules vers les types à la Montague (enrichis par des variables d'événements servant à réifier les formules). L'utilisation d'une extension du λ-calcul est nécessaire : le $\lambda\mu$-calcul, proposé par Parigot, [Par92], qui permet de modéliser, dans le λ-calcul, les continuations. Elles offrent la possibilité d'utiliser une variable dans son contexte. C'est une façon de dissocier l'apport d'un élément, à partir du moment où cet élément est inclus dans le calcul. Les premières utilisations de ce calcul pour la sémantique sont dues à [dG01]. Sur ces termes, on ajoute des concepts issus de la DRT, par le marquage effectif de la portée d'un quantificateur, ce qui nous permet de proposer les $\lambda\mu$-DRS.

Le problème majeur provient de l'interprétation sémantique de l'opération de déplacement. Nous proposons pour cela que ce soit la substitution de variables. Comme elle est réalisée uniquement lorsque tous les traits syntaxiques sont traités, on suppose que chacun d'entre eux introduit une nouvelle part de la sémantique du constituant, ce qui nous permet de rendre compte des Uniform Theta Assignement Hypothesis (UTAHs) (assignation des rôles thématiques). Cette vision du fonctionnement est capitale dans la réussite de la mise en place d'une interface syntaxe-sémantique.

L'interface est dirigée par le calcul syntaxique jusqu'au point de *Spell-Out* (acceptation de la dérivation). Après ce stade, le calcul sémantique est alors autonome et se poursuit par une phase d'internalisation des prédicats, puis de résolution des μ-abstractions. À la fin du calcul, on obtient alors plusieurs formules représentant les différentes lectures de quantification.

L'utilisation de ce système permet de se rapprocher de travaux sur le calcul de la sémantique à partir des GMs qui proposaient une modélisation du déplacement comme abstraction dans le calcul ([Ver99], [Kob06]). Mais l'impossibilité d'utiliser un système simple vient, selon nous, du mélange des opérations qui ont lieu en même temps dans ces calculs, d'où la nécessité de dissocier sur chaque opération et de substituer.

Enfin, nous mettons en perspective les GMCs et l'interface syntaxe-sémantique dans des analyses concrètes pour des phénomènes linguistiques réels. Pour cela, nous proposons

un fragment d'une grammaire du français. Nous concentrons la problématique sur le cas des clitiques. Très présents en français (et plus largement dans les langues romanes), les clitiques ont la particularité de posséder des comportements différents en fonction du cadre dans lequel ils sont utilisés.

En se basant, sur une première analyse proposée par Stabler, [Sta99], nous modélisons l'ensemble des clitiques du français en dissociant cette procédure en deux parties :

– l'une à phonologie vide portant la sémantique et prenant une position argumentale du verbe.
– la seconde à sémantique vide mais portant la partie phonologique du clitique.

Seul le traitement du clitique négatif nécessite l'introduction d'une nouvelle construction autour de l'inflexion, les autres se positionnant de manière homogène.

La cliticisation a cependant des particularités lorsqu'elle se situe dans des constructions complexes, en particulier à l'intérieur du groupe verbal. Nous avons donc été amenés à, dans un premier temps, ajouter le traitement de la négation et de l'impératif. Puis nous étudions les phénomènes de verbes enchâssés dans lesquels la cliticisation est partagée par plusieurs verbes. La reconnaissance de ces phénomènes est rendue possible par la création d'un statut particulier pour les verbes à l'infinitif. Nous nous attachons également à montrer comment les constructions sémantiques sont rendues possibles dans de tels cas. Enfin, d'autres phénomènes analogues se manifestent dans des constructions complexes de groupe nominaux, nous montrons comment utiliser la procédure de cliticisation pour en rendre compte. Une suite d'exemples est exposée dans ce chapitre.

À partir de ces résultats qui approfondissent l'étude des GMs et proposent un nouveau formalisme pour lequel nous proposons une interface syntaxe/sémantique, nous concluons en évoquant les différentes perspectives issues de nos travaux, par exemple vers une interface dont la couverture des phénomènes est plus grande et pouvant être utilisée de manière effective à plus grande échelle.

Première partie

Modélisation Syntaxique

Avant d'entreprendre le premier chapitre, une question naturelle se pose, pourquoi consacrer un chapitre entier à la présentation du programme minimaliste, et non à d'autres thèmes tout autant fondateurs de ces travaux. Certainement parce que le programme minimaliste, proposé par Chomsky dans les années cinquante, [Cho57], est rarement présenté simplement et souvent méconnu des personnes pouvant s'intéresser à ces travaux. De plus, il est malaisé de renvoyer le lecteur aux ouvrages le présentant, et qui nous ont permis de mieux appréhender les questions posées, comme [Pol97] ou [Rad97], tant l'éventail de perspectives ouvertes est large et viendrait parasiter la lecture de ce manuscrit. Nous avons donc dévolu un chapitre à sa présentation.

Cependant, d'autres thèmes abordés plus après dans ce manuscrit auraient également mérités une présentation, par exemple les grammaires catégorielles, l'isomorphisme de Curry-Howard, etc. Mais le temps et l'espace impartis à l'exercice étant limités, nous renverrons le lecteur à d'autres ouvrages, plus nombreux et spécifiques tels [Ajd35a], [BH53], pour l'introduction historique des grammaires catégorielles et [Ret00] et [Moo97] pour en avoir une vision plus générale ; et [GLT88] pour l'isomorphisme de Curry-Howard.

C'est aussi la démarche que nous avons suivie dans l'introduction précédente pour les autres formalismes d'analyse syntaxique, pour lesquels, étant plus classiques, nous avons cités les références fondatrices.

Nous commencerons donc cette partie sur la modélisation syntaxique par placer dans une perspective historique la théorie générative qui porte les concepts linguistiques que nous avons choisi d'utiliser. La version la plus récente de cette théorie qui débute par [Cho57] est connue sous la terminologie de *programme minimaliste*, présenté dans [Cho95]. Afin de simplifier, autant que faire se peut, la compréhension de ces travaux, les concepts présentés dans ce chapitre seront repris sous les mêmes noms dans le formalisme que nous utiliserons par la suite.

Le chapitre suivant présente un formalisme implémentant le programme minimaliste qui a été introduit par Edward Stabler dans [Sta97], puis enrichi dans [Sta99], [Sta01]. Ce formalisme a l'intérêt de proposer un cadre formel simple à l'analyse syntaxique dans la perspective chomskyenne. En effet, il n'utilise que deux opérations (qui sont chacune raffinée) et des structures de listes. Ces grammaires ont été largement utilisées par exemple par J. Hale, [Hal03], et G. Kobele, [Kob06]. Nous reviendrons sur certaines propriétés fréquemment admises et nous verrons qu'elles sont parfois non triviales.

Puis, nous présentons plusieurs résultats connus sur ce formalisme, afin de le placer dans une perspective de théorie des langages. Pour cela nous revenons brièvement sur la hiérarchie de Chomsky [SC63], pour les grammaires puis nous exhibons un type particulier de dérivation par le langage $\{a^{2^n} | n \in \mathbb{N}\}$, qui a été proposé dans [MG05], ainsi qu'une transformation des GMs vers les LCFRS, [MMM00], [Mer06]. Nous avons poursuivi par une analyse des GMs sans l'opération de déplacement, qui montre que la fusion permet de construire des grammaires algébriques et le déplacement, d'aller au-delà. Le problème est alors de limiter la sur-génération potentielle. À partir de cette étude nous proposons une modélisation générique des lexiques.

Enfin, dans le but de mieux appréhender l'impact de l'opération de déplacement et

dans la même perspective que les travaux de Michaelis, [MG05], nous proposons différents exemples de grammaires pour des langages théoriques connus.

Chapitre 1

Le programme minimaliste

Sommaire

1.1 Fondamentaux de la théorie générative 15
1.2 Pour une histoire 17
1.3 Le programme minimaliste 24
1.4 Sur la liberté 30

Dans ce chapitre, nous présentons sous un aspect historique l'évolution de la grammaire générative qui a été introduite par N. Chomsky, [Cho55]. Dès cette époque, les modélisations syntaxiques ont été réalisées à partir de la notion d'arbre, au sens traditionnel de l'informatique moderne, [GS97]. L'évolution de la théorie est très proche de celle de l'homme. Nous noterons une première période de collaboration avec P. Schützenberger, [SC63] qui ouvre la partie de théorie des langages formels. Puis un retour à la linguistique, avec notamment [Cho65] puis l'introduction de la notion de transformations, qui donnera l'autre nom de ces grammaires, les grammaires transformationnelles, [Cho73]. Les deux étapes majeures suivantes de la théorie sont la proposition de GB, [Cho81] et le minimalisme qui est à la base de nos travaux, [Cho95].

1.1 Fondamentaux de la théorie générative

Chomsky[1] s'est inscrit en rupture avec la pensée *empirique* dès les années cinquante, pensée dominante depuis le $XVIII^{ème}$ siècle dans les sciences humaines. Il réintroduit le débat entre "empirisme" et "rationalisme". Les philosophes empiristes postulaient que l'être humain arrive au monde tel une surface vierge et que son expérience lui permet d'"écrire" sur cette surface. Ainsi, la connaissance qu'il acquiert au cours de sa vie serait la somme de ses expériences, marquées sur cette surface. Chomsky, dans le prolongement de Descartes, suppose lui qu'il est impossible que l'homme soit seulement réduit à cette somme et préfère expliquer le passage de l'expérience à la connaissance par l'admission de structures innées dans lesquelles s'inscrit l'expérience.

[1] Cette présentation doit beaucoup à l'article de C. Boeckx et N. Hornstein [BHar], ainsi qu'à celui de T. Reinhart [Rei07], dont les traductions françaises ont été rassemblées dans un numéro spécial des cahiers de l'Herne sur Noam Chomsky à l'occasion des cinquante ans de "Syntactic Structures", article fondateur de la théorie générative, [cdl07].

À partir de ces postulats se dégagent deux visions de l'être humain. Les empiristes considèrent que l'homme et l'animal viennent au monde avec la capacité d'apprendre de l'expérience, alors que les rationalistes différencient l'homme de l'animal comme étant un organisme doté de facultés particulières. La prégnance de la thèse empiriste peut s'expliquer par le contexte socio-culturel qui englobe cette période, notamment par le fait que cette thèse donne une impression très scientifique et distingue la science de la notion de religion et, par extrapolation, du contrôle de l'église. Dans le même temps, l'empirisme établit un parallèle fort avec le dogme capitaliste qui s'est développé à travers le monde à cette période. En effet, l'idée de "table rase" à partir de laquelle l'homme se construit, laisse des possibilités d'évolutions infinies à un être dans la société.

Dans les années cinquante, l'empirisme est repris par le comportementalisme (behaviorisme), porté par Skinner [Ski74]. Selon lui, l'apprentissage de la langue est un processus obtenu uniquement à partir d'exemples (positifs et négatifs) accompagnés de stimuli. Ainsi, un enfant qui réussit une phrase complexe et reçoit une récompense, met cette structure langagière dans la catégorie des structures utiles. La capacité langagière est alors structurée par analogie avec les exemples positifs et négatifs.

Se poser la question de formaliser le processus langagier suppose l'existence de règles le régissant, et formuler des règles entraîne déjà une abstraction sur la question du langage, induisant son caractère infini. Les concepts du comportementalisme, se basant sur une liste de possibilités et une règle de formation par analogie, excluent la possibilité de productions infinies (à partir d'un ensemble fini d'éléments et de la règle d'analogie, l'ensemble des productions est fini).

Il est aisé de prouver le contraire. Dans [Rei07], T. Reinhart propose la définition du "français optimiste" (hébreux dans la version originale) dans lequel il n'y a que les mots "tout, ira, bien, il, elle, pense, que". À partir de ce fragment du français, on propose les premières réalisations de phrase :

(1) Tout ira bien

(2) Il pense que tout ira bien.

(3) Elle pense qu'il pense que tout ira bien.

(4) Il pense qu'elle pense qu'il pense que tout ira bien.

On peut rapidement identifier la logique qui permet de passer de l'un des énoncés au suivant et de fait, on est capable de produire une infinité d'énoncés. Il faut malgré tout admettre que l'utilisation d'un grand nombre de relatives produit des énoncés difficilement compréhensibles, mais il est tout à fait possible d'extraire une idée de telles phrases.

Pour décrire le processus langagier, il est important, selon Chomsky, de différencier *compétence* et *performance*. La compétence englobe la capacité langagière alors que la performance décrit ce que nous en faisons effectivement. Les problèmes de compréhension dus au nombre de relatives dans les exemples précédents proviennent de notre performance langagière, mais le véritable sujet d'étude du linguiste reste la compétence langagière.

La pensée empiriste de Skinner, selon laquelle il n'y a pas de structure sous-jacente à la langue, est ébranlée par l'argument précédent. En réalité, il fait entrer la notion d'infini dans le concept d'analogie, mais il est par exemple difficile d'argumenter sur l'analogie entre la phrase 2 et la phrase 3 : elles n'ont pas le même nombre de mots, bien qu'elles aient une structure similaire. Cette approche est malgré tout utilisée en intelligence artificielle pour la production de systèmes '*intelligents*' capables d'apprendre par analogie.

Actuellement, ces derniers ne sont pas en mesure de reconnaître les exemples 2 et 3. On peut également illustrer cette possibilité par un autre argument où les analogies sont effectivement présentes, à partir des exemples suivants (extraits de [Rei07]) :

(5) L'investigateur a reçu deux livres d'Oxford

(6) L'investigateur a perdu deux livres d'Oxford

La notion d'analogie est clairement mise en évidence avec ces deux phrases, en effet elles sont identiques sauf en ce qui concerne le verbe. Il est donc possible de produire l'exemple 7 et par analogie, l'apprenant acceptera la phrase 8 (les phrases précédées d'un $*$ ne sont pas considérées comme valides) :

(7) C'est d'Oxford que l'investigateur a reçu deux livres.

(8) *C'est d'Oxford que l'investigateur a perdu deux livres.

Le cadre de la pensée empiriste ne résiste pas à ce nouvel argument. En conséquence, on peut difficilement justifier de suivre ce courant pour obtenir une modélisation pertinente de la langue naturelle. Selon Chomsky, il existe un modèle abstrait sous-jacent à la langue, qui offre de meilleures perspectives et nous reviendrons ultérieurement sur celles-ci.

De ces premiers commentaires, on retiendra le caractère infini de la langue, la structure sous-jacente, la distinction entre compétence et performance, le terme de *génération* qui provient de l'adéquation du modèle avec la notion de compétence du locuteur (ce qui peut être généré).

1.2 Pour une histoire

La linguistique générative est issue des travaux de Noam Chomsky. Elle date de plus de cinquante ans et a beaucoup évolué entre son introduction et sa version actuelle. Ces modifications sont en grande partie dues au sujet de l'étude lui-même, qui est un phénomène naturel, donc la perception que l'on en a, est influencée par le contexte socio-culturel. Dans sa courte histoire, la linguistique générative a connu trois grandes périodes visant chacune ses objectifs propres. Actuellement, ces trois axes sont encore fortement présents et inspirent des travaux nouveaux, certes voisins mais néanmoins différents. L'identification de ces périodes permet de comprendre comment, et en quoi, les enjeux ont évolué. Cette présentation en trois grandes périodes a été proposée par Boeckx et Hornstein dans [BHar]. Ils distinguent les phases *combinatoire*, *cognitive* et *minimaliste*. Pour chacune, les travaux ont été influencés par des parallèles avec des sciences plus anciennes et plus développées. Ainsi la phase combinatoire serait plus proche de l'ingénierie, la cognitive de la biologie et la phase minimaliste de la physique.

À Chacune de ces périodes est associées un texte fondateur de Noam Chomsky qui propose les axes principaux de ses recherches. Le texte central de la phase combinatoire est *Syntactic Structures*, [Cho57]. La période cognitive peut être dissociée en deux moments : le premier est ouvert par *Aspects of the Theory of Syntax*, [Cho65], et le second par *Lectures on Government and Binding*, [Cho81]. Cette période est la plus longue, ce qui explique la distinction en deux parties. Enfin, la dernière, est issue du *Minimalist Programm*, [Cho95]. Bien que le sujet d'étude de tous ces textes soit le même, chacun se focalise sur un aspect particulier de la langue. Ils peuvent être vus comme des balises dans la courte histoire de

la linguistique générative. Nous allons revenir sur chacune de ces périodes pour dégager et expliciter les enjeux du minimalisme, sujet de ce manuscrit.

1.2.1 La phase combinatoire

Comme nous l'avons évoqué, la phase combinatoire s'ouvre avec *Syntactic Structures*. Ce texte d'une centaine de page annonce une nouvelle théorie linguistique. Le principe fondamental en est de développer un formalisme explicite ("génératif") qui puisse rendre compte de phénomènes linguistiques. Chomsky y pose le problème de la façon suivante :

> *"Le but fondamental de l'analyse linguistique d'une langue L est de séparer les séquences grammaticales qui sont des phrases de L des séquences non grammaticales qui ne sont pas des phrases de L. La grammaire de L sera donc un dispositif qui engendre toutes les phrases grammaticales de L, et aucune des phrases non grammaticales"*

De ce point de vue, l'enjeu est donc de trouver une formalisation rendant compte précisément d'une langue naturelle, c'est-à-dire de ses productions acceptables (les productions en langue naturelle étant infinies). L'analyse est alors vue de façon combinatoire puisque toutes les productions sont envisageables. Le but est de caractériser l'ensemble des productions possibles par des règles (une grammaire) permettant de dissocier les deux ensembles infinis de productions possibles/impossibles. On remarquera aussi que le terme de générativiste provient de cette perspective de générer uniquement les phrases grammaticalement correctes.

Cette théorie présente deux arguments importants, qui sont développés dans la comparaison des avantages et inconvénients des formalisations alternatives de la grammaire. D'abord, l'impossibilité pour des grammaires à états finis d'être un modèle correct. En effet, les processus d'états finis (dits *markoviens*) sont incapables de modéliser les langues ayant des dépendances entre éléments non-contigus : par exemple, une chaîne qui possède n occurrences d'un a puis autant d'un b. Ce type de dépendance est extrêmement présent dans les langues naturelles, comme le montre Chomsky :

> Soient $S_1, S_2, S_3, \ldots$, des phrases déclaratives du français. On peut avoir les phrases :
>
> (9) Si S_1, alors S_2.
>
> (10) Soit S_3, soit S_4.
>
> (11) La personne qui a dit S_5, va arriver aujourd'hui
>
> Dans (9) on ne peut pas avoir *ou* à la place de *alors* ; dans (10) on ne peut pas avoir *alors* à la place de *soit* et dans (11) on ne peut avoir *vont* à la place de *va*. Dans chacune de ces trois phrases il y a une dépendance entre des mots de part et d'autre de la virgule, mais pour chacune de ces dépendances, on peut insérer une phrase déclarative (voire l'un des trois exemples).
>
> Il est clair qu'en français on peut construire une phrase $a + S_1 + b$ avec une dépendance entre a et b, et l'on peut choisir pour S_1 une phrase contenant une nouvelle dépendance, par exemple $c + S_2 + d$, puis choisir pour S_2 une autre phrase de la même forme, etc. Un ensemble de telles phrases aura

> toutes les propriétés qui les excluent de l'ensemble des langages à états finis.

Dans cet exemple on voit le caractère infini des productions langagières réalisables et celui de l'utilisation de dépendances non locales. Ces deux aspects ne peuvent être pris en compte par les grammaires d'états finis. On montre aisément la présence de ce type de construction pour un grand nombre de langues naturelles, ce qui rend de telles grammaires inaptes à rendre compte de la structure langagière.

Chomsky aborde un autre type de grammaires : les grammaires à *structure de phrases*. Elles échouent à rendre compte de la langue naturelle mais pour un argument bien différent. Elles peuvent tout à fait produire une infinité d'énoncés, cependant, elles se contentent de "coller" à la langue. Chomsky les qualifie alors "d'extrêmement complexes, *ad hoc*, et peu transparentes". L'idée sous-jacente est que, bien qu'il soit possible d'utiliser la description des phénomènes pour les reconnaître, le grand nombre de règles nécessaires pour en rendre compte, ainsi que tous les cas particuliers, se rapproche de la tentative de décrire une infinité de règles. L'argument selon lequel un grand nombre de vies finies ne suffisent pas à décrire une infinité d'éléments est suffisant pour remettre en cause ce type d'approche.

La problématique est donc de trouver les généralisations significatives évidentes. Chomsky exhibe l'exemple de la formation des phrases passives qui a ouvert une large discussion. Nous nous contenterons de présenter l'exemple et de commenter brièvement la comparaison entre les grammaires à structure de phrases et la perspective chomskyenne. Soient les deux énoncés suivants en relation actif/passif :

(12) John ate a bagel [Jean mangea un beignet]

(13) A bagel was eaten by John [Un beignet a été mangé par Jean]

Nous pouvons constater que les relations verbe/sujet et verbe/objet de (12) ne trouvent pas de restriction dans les mêmes relations de (13). Les phrases suivantes sont des représentations abstraites de structures de phrases où NP désigne un groupe nominal et V un verbe. Donc si (14) est correct alors (15) l'est aussi. Et sinon, (15) est également faux.

(14) NP_1 V NP_2

(15) NP_2 be V+en by NP_1

Nous allons discuter la suite d'exemples qu'il propose :

(16)
1. John drinks wine [Jean boit du vin]
2. Wine is drunk by John [Du vin est bu par Jean]
3. Sincerity frightens John [La sincérité effraie Jean]
4. John is frightened by sincerity [Jean est effrayé par la sincérité]

(17)
1. *Wine drinks John [Du vin boit Jean]
2. *John is drunk by wine [Jean est bu par du vin]
3. *John frightens by sincerity [Jean effraie la sincérité]
4. *Sincerity is frightened by John [La sincérité est effrayée par Jean]

Les restrictions pour les phrases à l'actif semblent être en général les mêmes que pour celles au passif. Les grammaires à structure de phrases doivent donc rendre compte de toutes les restrictions encodées pour les verbes transitifs à l'actif dans leur version au passif. Chomsky soutient que ce type de grammaire ne peut rendre compte d'une généralisation pour ces phénomènes, ce qui les rend impropres à être un outil formel pour la reconnaissance des langues naturelles.

Il propose alors une règle de la forme (*Aux* désigne la position de l'auxiliaire) :

(18) Si S_1 est une phrase grammaticale de la forme
$NP_1 - Aux - V - NP_2$
Alors la chaîne correspondante de la forme
$NP_2 - Aux + be + en - V - by - NP_1$
est aussi une phrase grammaticale.

Ce type de règle évite certaines redondances inélégantes qui consistent en la description exhaustive des phrases acceptables.

La plupart des travaux de cette phase de la linguistique générative ont consisté à examiner diverses combinaisons de règles, de structures de phrase et d'opérations transformationnelles, dont l'objectif était d'engendrer toutes les phrases, et seulement les phrases grammaticales d'un langage L donné, et, par là, de refléter les relations perçues par les locuteurs natifs. Les types de transformations pour rendre compte d'un phénomène pouvaient soit produire des phrases agrammaticales, soit échouer pour d'autres cas, ce qui était un problème récurrent. Ces transformations n'étaient donc pas nécessairement la généralisation opérée par un locuteur natif. Montrer l'adéquation entre généralisation et perception d'un locuteur natif était un problème particulièrement difficile.

L'identification des axiomes était également la démarche choisie par d'autres disciplines comme l'informatique ou la philosophie. Cette perspective combinatoire a eu de grandes répercussions en informatique théorique, notamment dans la branche des langages formels, nous y reviendrons au cours du chapitre 3.

1.2.2 La phase cognitive

Aspect d'une théorie de la syntaxe

Dans cette phase, le générativisme se place dans une perspective cognitive et biologique. C'est certainement la déclaration la plus claire de la théorie générative. Chomsky soutient que le problème central de la linguistique est d'expliquer la capacité des enfants à acquérir leur langue maternelle. Il expose deux normes d'évaluation :

1. Une grammaire est descriptivement adéquate *dans la mesure où elle décrit correctement la compétence intrinsèque du locuteur natif idéalisé.*
2. Une théorie de la grammaire est descriptivement adéquate *si elle fournit une grammaire descriptivement adéquate pour chaque langue naturelle.*

La description des grammaires perd le caractère combinatoire de la possibilité de générer exactement une langue, mais tend vers une perspective cognitive plus abstraite qui peut se résumer par *qu'est-ce que la connaissance possédée par un locuteur natif.* Les grammaires ne sont donc pas évaluées pour elles-mêmes mais pour leur correspondance avec la théorie de l'adéquation à la langue.

La notion d'*adéquation* devient plus perceptible si on la regarde par le prisme de la question de la proposition d'un *modèle d'acquisition* de la langue. Le problème qui se pose à un enfant peut être résumé ainsi :

> "[Un] enfant qui a appris une langue a développé une représentation interne d'un système de règles... Il l'a fait sur la base d'observations de ce que nous pouvons appeler des *données linguistiquement primaires*. À partir de telles données, l'enfant construit une grammaire, c'est-à-dire une théorie du langage dont les phrases bien formées des données linguistiques primaires constituent un petit échantillon."

L'objet de l'acquisition est donc de formuler des hypothèses sur le modèle qui permet à un locuteur d'apprendre la grammaire de sa langue naturelle. Ainsi, l'une des hypothèses fondatrices est que seules les hypothèses "valides" soient retenues, c'est-à-dire que dans la phase d'apprentissage, l'être humain ne cherche pas le bon degré de granularité dans la formation des règles, mais de manière *innée*, possède la connaissance de cette granularité. Ce postulat peut paraître singulier, mais est confirmé par un grand nombre d'études psycholinguistiques.

La non-remise en cause de ce niveau de formalisation interne répond au principe de *pauvreté du stimulus*. Ce principe suggère que tous les stimuli reçus par un enfant ne peuvent recouvrir l'ensemble des énoncés corrects. Tout comme la formation des données primaires, le principe de pauvreté du stimulus est la base d'un pan entier de la linguistique générativiste, dont le résultat principal est que la langue ne peut être apprise par une suite finie de stimuli chez l'enfant. Il est toujours possible de trouver des contre-exemples dans le processus d'acquisition, ou des particularités d'une langue, pour remettre en cause la vision empiriste de l'apprentissage. Nous reviendrons sur la distinction entre rationalisme et empirisme dans la section suivante. Chomsky affirme donc la nécessité de mécanismes biologiques spécifiques à l'être humain pour expliquer le développement de la langue, d'où la proposition d'une structure sous-jacente, probablement encodée dans nos gènes.

Dans *Aspect of the theory of syntax*, le projet de développer des théories adéquates a apporté plusieurs conditions abstraites qui sont résumées ainsi :

> "... nous devons exiger d'une telle linguistique qu'elle fournisse
>
> 1. une énumération de la classe $s_1, s_2, \ldots$ des phrase possibles ;
> 2. une énumération de la classe $SD_1, SD_2, \ldots$ des descriptions structurelles possibles ;
> 3. une énumération de la classe $G_1, G_2, \ldots$ des grammaires génératives possibles ;
> 4. la spécification d'une fonction f telle que $SD_{f(i,j)}$ soit la description structurelle assignée à la phrase s_i par la grammaire G_j (pour i et j arbitraires) ;
> 5. la spécification d'une fonction m telle que $m(i)$ est un entier associé à la grammaire G_i comme étant *sa valeur* (avec des valeurs inférieures indiquées par des nombres plus grands).

Un dispositif qui rend compte de ces exigences pourrait employer les descriptions linguistiques primaires pour former des grammaires adéquates aux phrases entendues par

l'enfant. La dernière condition dicte l'introduction d'une métrique entre les grammaires biologiquement accessibles. Elle permet de distinguer les différents stades d'acquisition de la langue.

Même si cette caractérisation est correcte, elle est particulièrement difficile à mettre en œuvre. En particulier, la métrique pose de véritables problèmes de formalisation. Il a fallu attendre les années quatre-vingts avec *Lectures on Government and Binding* (GB) [Cho81] pour voir l'apparition d'un élément permettant d'évaluer des niveaux de descriptions avec les notions de *principes et paramètres*.

Principes et Paramètres

Comme nous venons de le voir, après *Aspect of the theory of syntax*, le problème central de la linguistique générative a été identique à celui d'une branche de la biologie, connue sous le nom de *morphologie théorique*. Elle peut se résumer (beaucoup trop succinctement) par l'identification du concept pour lequel les espèces actuellement présentes ne sont qu'un sous-ensemble des espèces théoriques potentielles. Il est donc central de comprendre pourquoi les formes qui existent, existent, et pourquoi celles qui n'existent pas, n'existent pas.

Lectures on Government and Binding reformule le problème de la façon suivante : les enfants naissent dotés d'un ensemble de principes de constructions grammaticales (introduction de la notion de grammaire universelle - GU). Les principes de la GU ont des paramètres non spécifiés. Les grammaires spécifiées se forment par attribution de valeurs à des paramètres. Ces valeurs sont apportées par analyse des données linguistiques primaires auxquelles est soumis l'apprenant.

Ainsi, le milieu environnant d'un enfant lui permet bien d'apprendre une langue, tout en laissant des paramètres ouverts un grand moment. Le degré de connaissance atteint par un enfant n'est pas en contradiction avec le potentiel inné qu'il possède.

Un grand nombre des travaux des années soixante dix et en particulier ceux de Kayne [Kay75], peuvent être vus par ce prisme. Un exemple de Pollock [Pol89] met en avant ce type de recherche et une réalisation de la notion de principes et paramètres : le cas des adverbes en français et en anglais. En anglais, l'adverbe ne peut pas se positionner entre le verbe et l'objet direct au contraire du français.

(19)
1. *John eats quickly an apple
2. Jean mange rapidement une pomme
3. John quickly eats an apple
4. *Jean rapidement mange une pomme

Le paradigme induit semble être le résultat d'une variation paramétrique entre la grammaire de l'anglais et celle du français. Dans les deux langues, la structure de la phrase est proche de

$$[_s\text{Sujet [Inflexion [Adverbe [}_{VP}\text{ Verbe Objet]]]]}$$

C'est l'ajout de l'inflexion qui transforme cette structure en véritable phrase. L'inflexion se caractérise par l'ajout de ses marqueurs au verbe dans les deux langues. On l'appelle "principe de l'ajout de l'inflexion". La façon d'apporter l'inflexion à chacune des deux langues est différente (paramètre de la langue). En anglais, l'inflexion descend vers le verbe, tandis qu'en français le verbe monte vers l'inflexion. On a alors les structures :

français : [$_s$Sujet [Inflexion + Verbe [Adverbe [$_{VP}$ ⟨~~V~~⟩ Objet]]]]
anglais : [$_s$ Sujet [⟨~~Infl~~⟩ [Adverbe [$_{VP}$ Verbe + Inflexion Objet]]]]

Cette différence de paramètre pour le principe d'ajout de l'inflexion, explique la variation de position de l'adverbe dans les deux langues étudiées, où sa position est structurellement la même (à côté du verbe). Mais cette théorie ne dit pas en elle-même qu'elle est la meilleure théorie. Simplement, elle présente la structure générale à laquelle la théorie doit savoir répondre. À nouveau, dans cette description on voit l'enjeu de la restructuration des énoncés entre une forme donnée et la forme finale. La notion de *principes et paramètres* (PP) alors proposée, a su faire un très large consensus dans la communauté de la linguistique générative et plus largement encore. Elle a ouvert la voie à la dernière phase que nous allons aborder maintenant : le *minimalisme*.

1.2.3 Le Minimalisme

Dans cette partie, nous allons traiter les rapports entre le minimalisme et l'histoire dans laquelle il se place. Comme ce manuscrit s'inscrit directement dans cette perspective, nous n'aborderons sa présentation que dans la section suivante qui lui est entièrement consacrée.

Les périodes historiques que nous venons de parcourir ont chacune un apport principal. La première période pose le problème de fournir des outils formels pour la description du langage. La seconde se place dans une perspective cognitive (et par extension biologique). Elle pose un cadre général au problème de l'acquisition. La proposition de PP présente trois qualités majeures : d'une part, elle fait le lien entre la langue parlée d'un individu et celle à laquelle il est exposé, d'autre part elle est cohérente avec le principe de pauvreté du stimulus et enfin elle est applicable à la recherche linguistique.

Si on admet l'idée de PP, la question naturelle qui se pose est celle de la meilleure proposition. À cette question, Chomsky répond par le *programme minimaliste*. Il faut s'accorder sur l'acceptation de minimalité dans les enjeux proposés sur lesquels nous allons à présent revenir. Pour cela, Chomsky pose le problème sous un jour différent et cherche à lier le *son* et le *sens* de la langue en utilisant la syntaxe comme pivot du système calculatoire. De ce fait, il abandonne les différents niveaux de représentation anciennement utilisés si ces derniers ne sont que des problèmes apparents pouvant être dérivés de propriétés plus profondes. En fait, le Minimalisme répond à une question plus large posée aux sciences : au-delà de savoir comment les choses sont, il faut aussi poser la question de pourquoi ces choses sont ainsi.

Cette idée d'élégance de la solution proposée a une conséquence directe dès les premiers articles. Le Minimalisme est fortement influencé par *Government and Binding* (GB) qui est particulièrement développé avec une large couverture expérimentale. C'est l'un des PPs retenus les plus évolués. GB distingue quatre niveaux de représentation : la *structure profonde*, la *structure de surface*, la *forme phonologique* et la *forme logique*. Les deux derniers sont les interfaces avec d'autres niveaux d'interprétation de la langue. Chomsky affirme que n'importe quelle théorie réaliste d'analyse de la langue ferait le lien entre ces deux niveaux de représentation (Aristote avait déjà proposé de faire un lien entre son et sens). Ils répondent donc à un niveau d'abstraction plus important que le cadre proposé.

Par contre, les structures profondes et de surface ne semblent pas répondre aux mêmes exigences et semblent être plus utiles à l'explication que théoriquement justifiées. Si cela

est vrai, alors une meilleure théorie du point de vue conceptuel est une théorie qui n'utilise pas ces niveaux de représentation.

Ces formes de factorisations théoriques peuvent avoir deux directions. L'une horizontale qui cherche à réunir des ensembles de phénomènes et de lois. Par exemple, GB possède différents modules tels le liage, le contrôle, etc. Le but est alors de minimiser le nombre de modules tout en gardant le même niveau de couverture des phénomènes. La seconde direction est, elle, plus verticale. On cherche à extraire les incohérences en isolant les erreurs et en essayant de les corriger et ce en conservant le principe d'optimalité qui repose sur l'idée que les grammaires tentent de fournir des représentations utilisables par les interfaces.

Évidemment, ces perspectives de factorisation ne sont pas mutuellement exclusives. Elles sont d'ailleurs issues de remarques très générales qui peuvent s'appliquer à l'ensemble des sciences. L'honnêteté intellectuelle est d'ailleurs de dire que ce n'est que l'application de concepts génériques aux cas particuliers de la linguistique théorique.

Contrairement à l'acceptation courante autour du minimalisme, le but n'est pas d'obtenir une représentation minimale, mais que l'ensemble du système du point de vue conceptuel et du point de vue calculatoire soit minimal.

Ainsi Chomsky fait le lien entre les deux premières phases et place l'enjeu de la linguistique théorique dans une perspective pragmatique (au sens de sa réalisation et non de son sujet d'étude). À présent, nous allons revenir sur les différentes composantes du programme minimaliste.

1.3 Le programme minimaliste

1.3.1 Système computationnel

Le *Minimalist Program* (programme minimaliste) [Cho95] est un court ouvrage technique qui condense toute la théorie chomskyenne. Le but est d'avoir un système capable de rendre compte de la connaissance d'une langue, au sens où connaître une langue est la faculté d'associer un sens à du son. Le système computationnel est basé sur l'analyse syntaxique d'un énoncé. À partir des phrases, il produit une structure linguistique (basée sur la notion d'arbre) en utilisant le lexique mental qui contient les mots et autres morphèmes de la langue (les plus petites unités de son porteuse de sens qu'il soit possible d'isoler dans un énoncé).

Revenons sur le lexique. Chacune de ses entrées (item lexical) porte un encodage de son comportement syntaxique, une forme associée dans la perspective sonore de l'analyse, et une forme pour la perspective du sens (logique). Du point de vue du programme minimaliste, le lexique est entendu au sens fort, c'est-à-dire que les mots sont, pour la morphologie flexionnelle, fléchis. Il est présenté comme une collecte non-structurée des données.

La production des arbres d'analyse n'utilise pas de niveau intermédiaire dans le calcul, elle s'opère donc en une seule étape. La dérivation aboutit à deux interfaces avec d'une part le système articulo-perceptuel qui concerne les aspects physiques de la langue (les sons ; dans la suite, nous appellerons forme phonologique (PF) la part de l'item lexical destinée à ce système et par extension au système lui-même) et d'autre part le système intentionnel-conceptuel qui envisage les phrases dans le but de penser ou de communiquer. Cette interface est appelée interface syntaxe/sémantique et sera l'un des objets d'étude

important de la suite de ce manuscrit. On appellera forme logique (LF) la part de l'item lexical destinée à ce système et, par extension, au système lui-même. Nous reviendrons sur les formes que prennent les PF et LF et sur leur utilisation.

Comme nous l'avons vu, Chomsky suppose que ce système est universel au sens où il est commun à la production de toute langue (GU). Les différences entre les langues se situent aux interfaces PF et LF. La question de savoir à quel moment la dérivation doit être transformée en production sonore est dépendante des caractéristiques de la langue, donc du lexique. Ainsi, la GU admet des productions très différentes (et donc des langues différentes).

À partir du lexique, le système calculatoire utilise plusieurs étapes en vue de fournir une analyse. Pour un énoncé donné, il effectue une *énumération* des items lexicaux qui entrent nécessairement ou peuvent entrer en considération dans l'analyse. Il s'agit ensuite de récupérer chacun des mots par une *sélection* sur ce sous-ensemble du lexique. Puis, le système utilise ses propres règles de composition, appelées *merge* et *move*. Nous reviendrons ultérieurement sur leur fonctionnement. Chomsky distingue un point particulier à partir duquel la dérivation syntaxique à proprement parler est terminée, mais après lequel des opérations peuvent/doivent être effectuées pour fournir des données utilisables aux interfaces. Ce point est appelé le "*Spell-Out*".

Les principes présentés ici sont repris dans la figure 2. Le système computationnel doit donc être considéré pour sa capacité à rendre compte d'une langue (et d'elle seule) mais aussi sur son adéquation entre les productions son/sens et les attentes des interfaces.

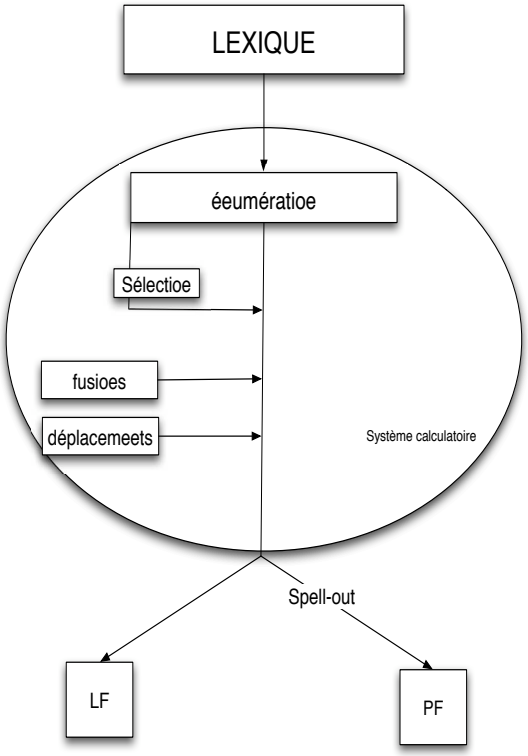


Fig. 2 – Structure du programme minimaliste de Chomsky.

1.3.2 Le schéma de dérivation

À partir d'un exemple de phrase, nous allons présenter les relations entre les différents éléments et les interprétations faites dans le système computationnel.

(20) Les employés ont justifié leur déplacement à Nancy.

Le mot *déplacement* est un nom que nous marquerons par la catégorie N qui est une catégorie de base. Elle représente tous les mots de la langue ayant la propriété de référer à des entités abstraites (dans le cas présent qui sont des *déplacements*). Dans la phrase 20, un sous-ensemble a été sélectionné. La représentation syntaxique de la catégorie N crée un nom de niveau supérieur N^1 pour le groupe *déplacement à Nancy* (qui domine des catégories et non des mots). "À Nancy" se place à sa droite et est appelé *complément* de la catégorie N car il la caractérise (lieu où s'est déroulé le déplacement). L'élément principal de ce groupe est toujours le premier nom. On dit que c'est la *tête* du groupe. Le déterminant possessif *leur* joue ici un rôle de quantification sur le déplacement. Il a pour fonction de sélectionner dans l'ensemble des *déplacements à Nancy* celui qui est le leur. Il se place à gauche du groupe qu'il détermine et est appelé *spécifieur*. À nouveau il est nécessaire d'introduire un niveau supplémentaire d'abstraction N^2. On a ainsi analysé les relations entre les différents éléments d'un groupe nominal (groupe dont la tête est un nom). On résume ces relations par le schéma de la figure 3

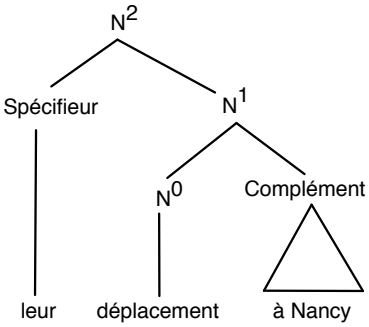


FIG. 3 – Relations spécifieur-tête-complément.

Il s'avère que ce schéma de relations syntaxiques n'est pas dépendant de la catégorie de base, et ce phénomène est vérifié dans toutes les langues étudiées. Ainsi, on peut lui substituer la catégorie d'un verbe V. De plus, on remarque que les notions de spécifieur et de complément peuvent être optionnelles et que seule celle de complément est multiple. Si on abstrait des remarques précédentes la catégorie utilisée par X, on obtient le motif suivant, connu sous le nom de schéma X-barre :

$$X^2 \rightarrow \text{(Spécificateur) } X^1$$
$$X^1 \rightarrow X^0 \text{ (Compléments)}$$

L'utilisation d'un schéma d'analyse peut faire apparaître des ambiguïtés, même dans des cas simples comme la phrase 20. En effet, si l'on suppose que Nancy est l'un des actants de la scène, on peut considérer que c'est la justification qui se fait auprès du personnage Nancy (et que nous ne connaissons pas le lieu du déplacement en question). Dans certains

cas, ces ambiguïtés sont bien réelles dans la langue et ne peuvent pas être résolues par la syntaxe. Cependant, pour certaines analyses, elles n'existent pas par exemple pour la phrase

(21) les employés ont annulé leur déplacement à Nancy.

Dans l'exemple précédent, il est impossible que Nancy soit l'un des personnages de l'action et c'est donc nécessairement la ville de destination du déplacement. Cette information est en fait déductible des rôles répartis par le verbe entre les groupes nominaux de la phrase. Ainsi, *justifier* doit pouvoir attribuer le rôle d'Agent (qui justifie), le rôle de Thème (ce qui est justifié) et le rôle Receveur (à qui cela est justifié). On supposera que tous les verbes peuvent attribuer des rôles de Lieu ou de temps de manière optionnelle. Dans le cas du verbe *justifier* on peut avoir une ambiguïté sur le rôle de *à Nancy* qui peut être un receveur optionnel ou un lieu. Dans le cas de *annuler*, il est impossible de lui attribuer le rôle de receveur qui n'est pas disponible pour ce verbe. L'attribution des rôles thématiques portés par le verbe est lexicalement encodé et on formule le principe suivant :

"Chaque complément d'une tête lexicale doit recevoir un rôle thématique".

Nous reviendrons sur la distribution des rôles thématiques dans le chapitre 7. Maintenant que nous avons défini les relations entre éléments dans le système calculatoire, nous allons revenir sur les règles utilisées et sur leur fonctionnement.

1.3.3 Règles du système calculatoire

Comme nous l'avons évoqué, le système calculatoire compose des éléments entre eux pour obtenir une structure d'arbre représentant l'analyse des relations syntaxiques entre éléments de la phrase. Pour cela, Chomsky introduit une opération de composition de deux items pour former un morceau de dérivation, appelée *merge* (fusion), et une opération qui permet de transformer une structure en une nouvelle, comme pour le cas de mise à la forme passive d'un énoncé à l'actif. Il faut donc que les propriétés soient vérifiées à un certain niveau puis transformer l'analyse pour obtenir la forme adéquate. On appelle *move* (déplacement) cette opération.

Le déplacement est l'élément capital de ce système parce que son utilisation ajoute une grande souplesse à l'analyse et permet de reconnaître un grand nombre de phénomènes syntaxiques. On obtient un cadre qui possède uniquement deux règles qui imposent la vérification d'une structure avant de produire l'analyse complète qui est nécessairement plus grande. La taille des dérivations peut sembler un contre-argument à ce système, cependant l'opération de déplacement provient d'arguments linguistiques forts. De plus, la complexité apparente n'est pas reportée pour des analyses plus élaborées.

Nous allons revenir sur chacune de ces opérations pour les présenter en détail.

Merge

La fusion est une opération qui construit un nouvel élément syntaxique à partir de deux objets syntaxiques. L'un des deux objets devient la tête de la nouvelle struture. Chomsky suggère que la manière de déterminer qui est la tête doit être encodée dans le lexique. Le rôle de l'opération est de construire une structure plus élaborée à partir de plusieurs des éléments en les regroupant en une structure syntaxiquement cohérente.

La figure 4 présente graphiquement le résultat d'une fusion. Dans cet exemple nous revenons sur la structure des groupes nominaux. Nous avons utilisé le fait que la tête de

tels groupes était le nom, mais dans la théorie générative, on considère que la tête est le déterminant pour ses propriétés sémantiques fortes (quantificateur), d'où la construction d'un groupe DP plutôt que d'un groupe NP.

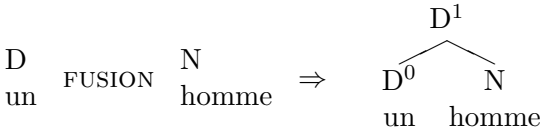


FIG. 4 – Fusion simple.

À partir de cette opération, nous pouvons dériver des exemples simples comme dans la figure 5.

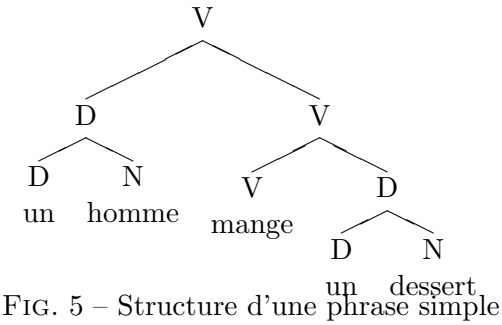


FIG. 5 – Structure d'une phrase simple.

Move

L'opération de déplacement est le mécanisme qui décrit les phénomènes de mouvement. Chomsky définit cette opération comme celle permettant de dissocier le niveau d'interprétation de ce qu'on entend. Ainsi chaque constituant d'une phrase peut prendre des positions différentes pour son interprétation.

Des nombreuses études linguistiques montrent que ce phénomène apparaît dans beaucoup de langues. Par exemple, pour les questions en anglais (Stabler [Sta97]) :

(22) Which tortillas does Maria make ?
* Maria makes which tortilla ?

La phrase acceptée dans cet exemple passe d'abord par une phase où l'ordre entre les constituants est *normalisé*. En anglais, comme en français, le sujet précède le verbe qui précède l'objet. Ainsi, bien qu'elle ne soit pas acceptée, la seconde phrase est en fait une étape de l'analyse. Pour certaines raisons, ici parce que la phrase est une question, un déplacement est réalisé, permettant à l'objet du verbe de passer en première position de l'énoncé. Cette transformation est présentée dans la figure 6.

Dans le programme minimaliste, Chomsky utilise la notion de traits qui correspondent à la dénotation de propriétés des éléments. Ainsi, chaque élément lexical possède une description de ses propriétés par des traits. Les *traits formels* sont regroupés sous forme de séquence pour chaque entrée lexicale. C'est à partir de ces traits que les dérivations s'opèrent, en particulier, la possibilité/nécessité d'opérer un déplacement. Certaines entrées

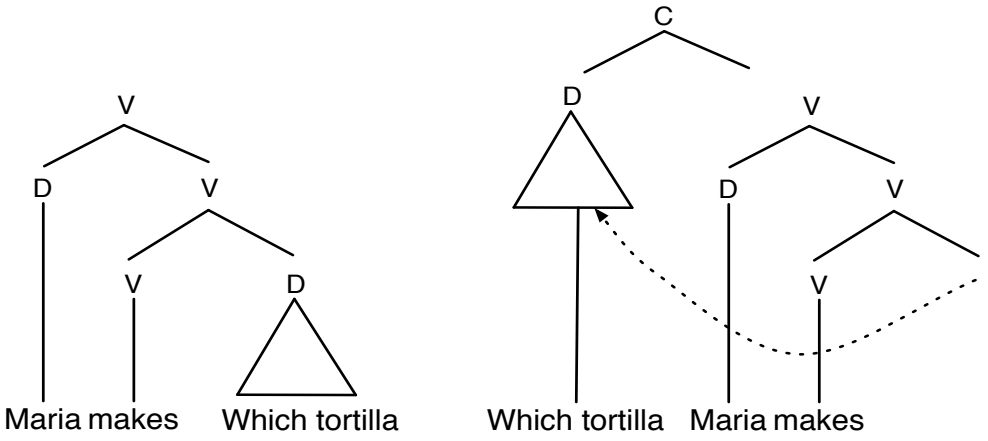


Fig. 6 – Exemple de transformation de phrase par déplacement d'un constituant.

lexicales n'ont pas de trait sémantique ou phonologique, elles ne sont présentes que pour assurer la validité de la dérivation.

Grâce à cette structure d'arbre, construit à partir des règles du système et des propriétés des éléments de base issus du lexique, on peut effectuer un grand nombre d'analyses syntaxiques.

Afin d'affiner la capacité générative du système, on introduit des conditions sur les dérivations. Le lexique et le fonctionnement du système ne peuvent pas contenir toute l'information. Chomsky propose de poser la *Condition d'adéquation aux interfaces* et la *Condition d'Économie* pour mémoriser d'autres informations.

Condition d'adéquation aux interfaces (Bare Output Condition - BOC) : Certaines conditions de dérivation sont induites par l'extérieur et ce, pour les deux niveaux LF et PF. Ces conditions restreignent le résultat en sortie du système calculatoire. En fait, seul un sous-ensemble des dérivations produites converge sous l'influence de la BOC. Grâce à cette condition, en PF, on évite de trouver une séquence non-prononçable de mots et en LF une séquence non-compréhensible (mais qui peut être grammaticalement acceptable). Cette condition prévoit une communication entre les différents niveaux d'interprétation que l'énoncé peut avoir.

Conditions d'Economie : Les déplacements doivent être limités par une condition d'économie pour ne pas faire exploser la complexité du système calculatoire. Elle sert à définir quand et comment un trait peut être utilisé pour faire un déplacement. Par exemple, elle suppose que si un déplacement peut être réalisé et que plusieurs candidats sont présents dans la dérivation, alors c'est l'élément pour lequel la condition d'économie est minimale qui réalise ce déplacement. Elle s'inscrit entièrement dans le cadre *minimaliste* que nous utilisons.

Enfin, dans la théorie minimaliste, on ajoute la notion d'*Island* qui sont des clôtures pour le déplacement. Ainsi, un constituant pris dans un island aura sa position finale dans la dérivation. Il existe de nombreux travaux sur la formalisation des conditions restreignant le déplacement. On sent intuitivement que l'ajout de contraintes sur le déplacement permet

d'affiner la capacité langagière du système.

Nous avons présenté les principales composantes du programme minimaliste.

1.4 Sur la liberté

À présent que nous avons décrit le programme minimaliste de Chomsky, il semble nécessaire de revenir sur un point important. Cet exposé ainsi que les chapitres qui vont suivre sont très fortement inspirés par les travaux de Chomsky. Il est important de préciser qu'il n'y a aucunement d'adoration de la personne ou de ses théories dans ce travail mais la présence récurrente de Chomsky dans ce manuscrit est principalement due à l'influence de cet homme sur la communauté scientifique et l'utilisation de ses théories pour leur grande cohérence. Les arguments présentés dans ce chapitre ne servent qu'à expliquer pourquoi certains concepts sont apparus et comment ils ont trouvé leur place dans ces travaux.

Avant de conclure, revenons sur le système proposé. La problématique de la modélisation de la langue s'inscrit évidemment dans un contexte plus large. Les perspectives biologiques apportées par Chomsky sont vues également comme un élément de réponse à la question des rapports entre êtres humains. Si l'on élargit cette question, on perçoit que l'engagement de Chomsky dans la cause politique provient de la même démarche.

De plus, la perspective rationaliste de description de la fonction langagière permet de situer l'humain comme doté d'une capacité immense de pensée, de création et de perfectionnement. Les évolutions sont à la fois le fruit de réflexions et de mises en perspective dans la connaissance acquise aux cours des générations. Ainsi, l'homme est vu comme une créature libre, même si le concept de liberté ne découle pas directement de la modélisation de la langue dont nous parlons, il dénote d'une dynamique dans ce sens.

Et c'est dans ce concept de liberté que nous souhaitons inscrire les travaux présentés dans la suite de ce manuscrit, dans la continuité de la pensée chomskyenne mais en gardant un recul nécessaire à la clairvoyance scientifique. Si le cadre d'étude est cohérent, il n'est pas la seule proposition actuelle, et s'inscrit, plus généralement, dans une histoire de la linguistique moderne. Le but n'est pas de justifier en soi cette théorie, mais de dégager les pistes toujours valides à mettre en perspective avec d'autres évolutions de la linguistique.

Ce programme a été formalisé en un type de grammaire appelée *grammaires minimalistes*. Avant de passer aux définitions des GMs, nous allons donner la structure des représentations utilisées pour expliciter les dérivations.

Chapitre 2

Les grammaires minimalistes

Sommaire

2.1 Préliminaires 31
2.2 Relations entre sous-arbres 33
2.3 Structures linguistiques 39
2.4 Grammaires minimalistes 46
2.5 Exemples d'utilisation des GMs 53

La perspective ouverte par Noam Chomsky avec son programme minimaliste pose la question de la proposition d'un système formel reprenant les enjeux précédemment exposés. Le but est double : introduire des systèmes automatisés de reconnaissance d'énoncés en langue naturelle et dans le même temps, comme Chomsky l'a affirmé lui-même, proposer une validation des analyses avancées par les linguistes. C'est sur cela que nous nous concentrons. Le premier système formel a été proposé par Ed Stabler, [Sta97] sous le nom de grammaire minimaliste - GM et qui a évolué dans [Sta99]. Elles ont été reprises sous différents angles d'approche par H. Harkema, [Har01], par J. Michaelis par une description formelle des langages reconnus, [Mic05], ou [Mic98], et même en collaboration avec G. Kobele, [MK05]. Ce dernier a également développé un système d'interface avec la sémantique, [Kob06]. Un parser a été implémenté pour ces grammaires par J. Hale, [Hal03].

Nous donnerons ici une présentation de ce formalisme en l'adaptant à la description algébrique des arbres, vus comme des contextes, ou dit à la "Huet", [Hue97], pour lesquels les descriptions des outils théoriques peuvent être retrouvées dans [LC]. Grâce à cette description, nous prouverons des propriétés du formalisme fréquemment admises. Nous nous attacherons à montrer comment les concepts mis en jeu dans le programme minimaliste trouvent leur interprétation ici. Une fois les GMs présentées, nous proposerons deux exemples de dérivation pour rendre compte de leur capacité à modéliser des phénomènes syntaxiques. Avant cela, revenons sur quelques notions élémentaires.

2.1 Préliminaires

L'ensemble $\{0, 1, \cdots\}$ des entiers naturels est noté $\mathbb{N}$. L'ensemble vide est noté $\emptyset$. Pour $k \in \mathbb{N}$, $[k]$ représente l'ensemble $\{1, \cdots, k\}$ si $k > 0$ et l'ensemble vide sinon (*i.e.* $[0] = \emptyset$).

Pour un ensemble A, $\mathcal{P}(A)$ est l'ensemble des parties de A, $|A|$ sa cardinalité, et A^* est l'ensemble des chaînes sur A. La chaîne vide est notée ϵ et la longueur d'une chaîne ω est notée $|\omega|$. Enfin, nous noterons A^+ l'ensemble des chaînes non vide ($A^+ = A^\star - \{\epsilon\}$).

2.1.1 Alphabets gradués et arbres

Étant donnés un ensemble A et une fonction $rang : A \to \mathbb{N}$, $\langle A, rang\rangle$ forme un **ensemble gradué** dont A est le support et **rang** est la fonction d'arité. Par abus de notation, nous confondrons un ensemble gradué $\Sigma = \langle A, rang\rangle$ et son ensemble support A. Nous écrirons par exemple $\sigma \in \Sigma$ pour $\sigma \in A$, $|\Sigma|$ pour $|A|$, ... Un **alphabet gradué** est un ensemble gradué de support fini.

Pour $k \geqslant 0$, $\Sigma^{(k)}$ est l'ensemble $\{\sigma \in \Sigma | rang(\sigma) = k\}$. On notera $\sigma^{(k)}$ pour indiquer le rang de l'élément σ ($rang(\sigma) = k$).

Soit Σ un ensemble gradué, l'ensemble des arbres construits sur Σ, noté T_Σ, est le plus petit ensemble de chaînes $T_\Sigma \subseteq (\Sigma \cup \{(;);,\})^*$ tel que si $\sigma \in \Sigma^{(k)}$, $k \geqslant 0$, et $t_1, \ldots, t_k \in T_\Sigma$, alors $\sigma(t_1, \ldots, t_k) \in T_\Sigma$. Pour $\alpha \in \Sigma^{(0)}$, on note l'arbre $\alpha()$ également α. Si $t = \sigma(t_1, \ldots, t_k)$, σ est appelé la **racine** de t.

Pour $\Sigma = \langle A, rang\rangle$ un ensemble gradué et B un ensemble, on note $\Sigma \cup B$ l'ensemble gradué $\langle A \cup B, rang'\rangle$ où pour $\sigma \in A$, $rang'(\sigma) = rang(\sigma)$ et pour $\sigma \in B$ $rang(\sigma) = 0$. On note $T_\Sigma(B)$ l'ensemble $T_{\Sigma \cup B}$.

On se dote d'un ensemble de variables $X = \{x_1, x_2, \cdots\}$, l'ensemble $\{x_1, \cdots, x_k\}$ est noté X_k. Etant donné t un élément de $T_\Sigma(X_k)$ et $t_1, \ldots, t_k \in T_\Sigma(X_l)$, on écrit $t[t_1, \ldots, t_k]$ l'élément de $T_\Sigma(X_l)$ obtenu par **substitution simultanée** des occurrences de x_1 par t_1, ..., et des occurrences de x_k par t_k. La substitution est associative, c'est-à-dire que pour $t \in T_\Sigma(X_k)$, $t_1, \ldots, t_k \in T_\Sigma(X_l)$ et $s_1, \ldots, s_l \in T_\Sigma(X_m)$ nous avons l'égalité :

$$t[t_1[s_1, \ldots, s_l], \ldots, t_k[s_1, \ldots, s_l]] = t[t_1, \ldots, t_k][s_1, \ldots, s_l]$$

Pour un alphabet gradué Σ, l'ensemble des **k-contextes** construits sur Σ est le sous-ensemble, noté $\mathcal{C}^k_\Sigma$, des arbres de $T_\Sigma(X_k)$ qui contiennent exactement une occurrence de chacun des éléments de X_k. On peut voir un k-contexte comme un arbre avec k trous.

Pour modéliser la notion de sous-arbre, nous utilisons les contextes dans une approche inspirée par [Hue97]. En effet, pour désigner sans ambiguïté une occurrence d'un arbre u dans un arbre t, il suffit de donner le contexte $C \in \mathcal{C}^1_\Sigma$ tel que $C[u] = t$. Par exemple, dans un alphabet gradué approprié, supposons que $t = b(a, b(a, a))$, t contient alors trois occurrences de a, et pour parler de l'occurrence la plus à gauche, il suffit de considérer le contexte $b(x_1, b(a, a))$. La figure 7 présente de manière intuitive le rapport entre contexte et sous-arbre. Le sous-arbre t_1 est dénoté par son contexte (présenté par l'arbre I de la figure 7) et symétriquement, le sous-arbre t_2 par le sien (l'arbre III). La figure présente aussi le 2-contexte par l'arbre II qui est la partie commune aux contextes de t_1 et t_2. Il se limite à l'arbre privé de t_1 et t_2. Aussi, pour $t \in T_\Sigma$, l'ensemble des sous-arbres de t, noté $\mathcal{S}_t$, sera représenté par $\{C \in \mathcal{C}^1_\Sigma | \exists u \in T_\Sigma, C[u] = t\}$. Il est clair que pour chaque élément C de $\mathcal{S}_t$ il existe un unique u qui vérifie $C[u] = t$. Par ailleurs, pour $C \in \mathcal{S}_t$, l'élément u de T_Σ est appelé **feuille** de t, tel que $C[u] = t$, s'il a pour racine un élément de $\Sigma^{(0)}$. Nous serons amenés par la suite à utiliser des k-contextes sur certains termes, nous notons donc pour $k > 0$, $\mathcal{S}^k_t = \{C \in \mathcal{C}^k_\Sigma | \exists t_1, \ldots, t_k \in T_\Sigma . C[t_1, \ldots, t_k] = t\}$. On notera que $\mathcal{S}^1_t = \mathcal{S}_t$ et

que pour $C \in \mathcal{S}_t^k$ les termes $t_1, \ldots, t_k$ tels que $C[t_1, \ldots, t_k] = t$ sont déterminés de façon unique.

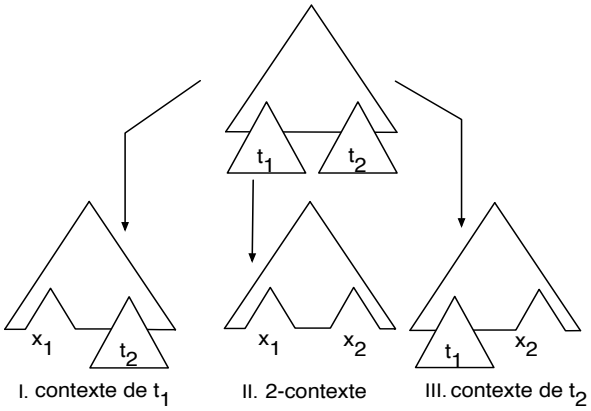


Fig. 7 – Arbre et relation internes.

2.1.2 Arbres minimalistes

On note Σ_{MG} l'alphabet gradué $\langle \{<;>\}, rang_{MG} \rangle$ tel que :

$$rang_{MG}(<) = 2 \text{ et } rang_{MG}(>) = 2$$

Étant donné un ensemble fini A, on appelle **arbres minimalistes** construits sur A l'ensemble $T_{\Sigma_{MG}}(A)$. Nous parlons d'arbres minimalistes lorsque la mention de A n'a pas d'importance, ou que A est évident à partir du contexte. On a ainsi défini les arbres binaires dont les nœuds sont étiquetés par $<$ ou $>$.

2.2 Relations entre sous-arbres

Nous formalisons les relations entre les positions relatives des éléments de $\mathcal{S}_t$. Il s'agit de définir les notions informelles *d'être au-dessus de*, *d'être à gauche* ou *à droite de*. Nous définissons également une relation spécifique aux arbres minimalistes, la relation de projection induisant la notion d'être l'élément principal de l'arbre.

Dans ce qui suit on suppose la donnée d'un alphabet gradué quelconque Σ.

Définition 1. *Soit $t \in T_\Sigma$, et $C_1, C_2 \in \mathcal{S}_t$, on dit que C_1 **domine** C_2 (noté $C_1 \lhd^* C_2$) s'il existe $C' \in \mathcal{S}_t$ tel que $C_1[C'] = C_2$.*

*Soit la **dominance stricte**, notée $\lhd^+$ telle que $C_1 \lhd^+ C_2$ si $C_1 \lhd^* C_2$ et $C_1 \neq C_2$.*

*Soit la **dominance immédiate**, notée $\lhd$, telle que $C_1 \lhd C_2$ si $C_1 \lhd^+ C_2$ et pour tout C si $C_1 \lhd^+ C$ et $C \lhd^* C_2$ alors $C = C_2$.*

La relation $\lhd^+$ est bien fondée puisque pour tout ensemble de sous-termes d'un arbre, il existe un élément racine des autres, donc minimal pour la relation. La conséquence immédiate est qu'il n'existe pas de suite infinie de sous-termes d'un arbre pour la dominance.

Une illustration de la relation de dominance est donnée dans la figure 8 où $C_1 \lhd^* C_2$.

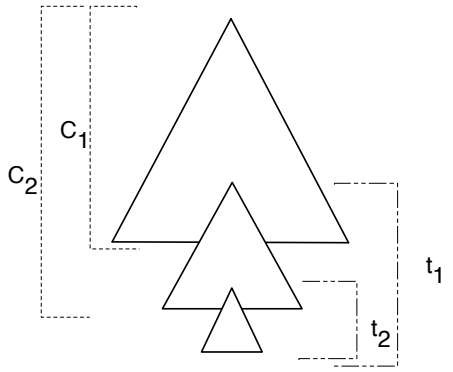


FIG. 8 – Relation de dominance.

De plus, on remarque aisément que la relation de dominance est antisymétrique puisque deux éléments ne peuvent pas mutuellement induire une construction où ils se dominent en même temps.

À présent, nous allons proposer des propriétés pour cette relation de dominance. Mais avant cela, remarquons l'intérêt de la présentation par contexte. Dans un arbre $t \in T_\Sigma$, on distingue deux nœuds n_1 et n_2 qui induisent deux contextes C_1 et C_2, en relation de dominance $C_1 \lhd C_2$. Le fait que ces deux contextes appartiennent à t implique que la racine de ces deux contextes est alors la même, c'est-à-dire celle de t. À partir d'un 1-contexte C, on peut construire un nouvel arbre $t' = C[t]$. On a alors substitué à la variable x_1 de C la racine de t. Il est donc possible de construire $C[C_1]$ et $C[C_2]$ en substituant leur racine (celle de t) à x_1 qui appartiennent tous deux à $C[t]$. Les notations sont alors grandement simplifiées.

Lemme 2. *Soit $t, t' \in T_\Sigma$, $C_1, C_2 \in \mathcal{S}_t$ et $C \in \mathcal{S}_{t'}$.*

La relation de dominance passe au contexte et réciproquement les relations de dominances vérifiées dans un contexte sont vraies localement :

$$C_1 \lhd^* C_2 \Leftrightarrow C[C_1] \lhd^* C[C_2]$$

$$C_1 \lhd C_2 \Leftrightarrow C[C_1] \lhd C[C_2]$$

$$C_1 \lhd^+ C_2 \Leftrightarrow C[C_1] \lhd^+ C[C_2]$$

Démonstration. On notera que C_1 et C_2 étant des contextes du même arbre t, ils contiennent tous les deux la racine de t. Dans les deux constructions $C[C_1]$ et $C[C_2]$, la variable libre de C est substituée par la racine de t, et les résultats sont des contextes du même arbre t'.

La preuve découle de la définition de la dominance immédiate. Si $C_1 \lhd^* C_2$ alors il existe C' tel que $C_1[C'] = C_2$. Or $C[C_1] \lhd^* C[C_1[C']]$ d'où $C[C_1] \lhd^* C[C_2]$. Réciproquement, si $C[C_1] \lhd^* C[C_2]$, il existe C' tel que $C[C_1[C']] = C[C_2]$ d'où $C_1[C'] = C_2$. Avec $C_1 \lhd^* C_1[C']$ on déduit que $C_1 \lhd^* C_2$.

$\lhd$ et $\lhd^+$ sont des cas particuliers de $\lhd^*$. □

La relation de dominance permet de définir les notions de père/fils entre éléments d'un arbre. Si $C_1 \lhd C_2$ on dit que C_1 est le père de C_2 et C_2 le fils de C_1. Cette vision des relations est une version verticale des éléments des arbres.

Définition 3. *Soit $t \in T_\Sigma$, $C_1, C_2 \in \mathcal{S}_t$, on dit que C_1 **précède immédiatement** C_2 (noté $C_1 \prec C_2$) s'il existe $C \in \mathcal{S}_t$ tel que*

1. $C_1 = C[\sigma(t_1, \ldots, t_j, x_1, t_{j+2}, \ldots, t_k)]$ *et*
2. $C_2 = C[\sigma(t_1, \ldots, t_j, t_{j+1}, x_1, \ldots, t_k)]$.

*La **précédence**, notée $\prec^\sim$, est la plus petite relation définie par le système de règles suivant :*

$$\frac{C_1 \prec^\sim C_2 \quad C_2 \prec^\sim C_3}{C_1 \prec^\sim C_3}\,[prec_{trans}] \qquad \frac{C_1 \prec C_2}{C_1 \prec^\sim C_2}\,[prec_*] \qquad \frac{C_1 \lhd^* C_2}{C_2 \prec^\sim C_1}\,[prec_{dom}]$$

*$\prec^+$ est la relation de **précédence stricte**. On dit que C_1 précède strictement C_2, noté $C_1 \prec^+ C_2$, si $C_1 \prec^\sim C_2$ et $C_1 \neq C_2$.*

Une illustration de la relation de précédence est donnée dans la figure 9 où $C_1 \prec^\sim C_2$ à partir du constructeur $op(C_1, C_2)$.

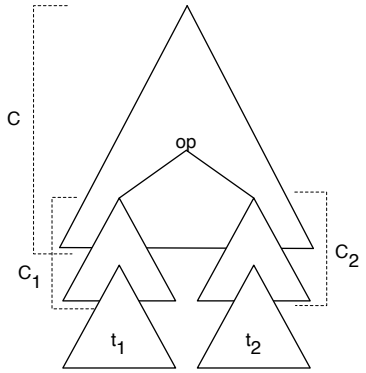


Fig. 9 – Relation de précédence.

On appelle $[prec_{fils}]$ la règle combinant la relation de dominance et la transitivité pour la précédence :

$$\frac{C_1 \prec C_2 \quad C_2 \lhd^* C_3}{C_1 \prec^\sim C_3}\,[prec_{fils}]$$

Lemme 4. *Soit $t, t' \in T_\Sigma$, $C_1, C_2 \in \mathcal{S}_t$ et $C \in \mathcal{S}_{t'}$.*

La relation de précédence passe au contexte et réciproquement les relations de précédences vérifiées dans un contexte sont vraies localement :

$$C_1 \prec C_2 \Leftrightarrow C[C_1] \prec C[C_2]$$

$$C_1 \prec^\sim C_2 \Leftrightarrow C[C_1] \prec^\sim C[C_2]$$

$$C_1 \prec^+ C_2 \Leftrightarrow C[C_1] \prec^+ C[C_2]$$

Démonstration. Par définition, $C_1 \prec C_2$ implique qu'il existe C' tel que :
$C_1 = C'[\sigma(t_1, \ldots, t_j, x_1, t_{j+2}, \ldots, t_k)]$ et $C_2 = C'[\sigma(t_1, \ldots, t_j, t_{j+1}, x_1, \ldots, t_k)]$.
Donc, pour tout C :
$C[C_1] = C[C'[\sigma(t_1, \ldots, t_j, x_1, t_{j+2}, \ldots, t_k)]]$ et $C[C_2] = C[C'[\sigma(t_1, \ldots, t_j, t_{j+1}, x_1, \ldots, t_k)]]$,
d'où, par définition, $C[C_1] \prec C[C_2]$.
Réciproquement, par induction sur C, si $C[C_1] \prec C[C_2]$:
– ou bien $C = x_1$ et donc trivialement on a $C_1 \prec C_2$.
– ou bien $C = \sigma(t_1, \cdots, t_i, C', \cdots, t_n)$ tel que $C'[C_1] \prec C'[C_2]$, par hypothèse d'induction on obtient $C_1 \prec C_2$.

On montre l'implication pour la précédence par induction sur la dérivation de $C_1 \prec^\sim C_2$.

1. règle $[prec_*]$: si $C_1 \prec C_2$, d'après le résultat précédent, $C[C_1] \prec C[C_2]$ d'où $C[C_1] \prec^\sim C[C_2]$.
2. règle $[prec_{trans}]$: si $C_1 \prec^\sim C_3$ et $C_3 \prec^\sim C_2$, par hypothèse d'induction, $C[C_1] \prec^\sim C[C_3]$ et $C[C_3] \prec^\sim C[C_2]$ et par transitivité $C[C_1] \prec^\sim C[C_2]$.
3. règle $[prec_{dom}]$: si $C_2 \lhd^* C_1$, d'après le lemme 2 $C[C_2] \lhd^* C[C_1]$ d'où $C[C_1] \prec^\sim C[C_2]$.

La précédence passe au contexte.
En utilisant les mêmes schémas de preuves et définitions, on obtient la réciproque.

1. règle $[prec_*]$: si $C[C_1] \prec C[C_2]$, d'après le résultat précédent, $C_1 \prec C_2$ d'où $C_1 \prec^\sim C_2$.
2. règle $[prec_{trans}]$: si $C[C_1] \prec^\sim C[C_3]$ et $C[C_3] \prec^\sim C[C_2]$, par hypothèse d'induction, $C_1 \prec^\sim C_3$ et $C_3 \prec^\sim C_2$ donc par transitivité $C_1 \prec^\sim C_2$.
3. règle $[prec_{dom}]$: si $C[C_2] \lhd^* C[C_1]$, d'après le lemme 2 $C_2 \lhd^* C_1$ d'où $C_1 \prec^\sim C_2$.

Pour la précédence stricte, l'équivalence est une conséquence de la preuve sur la précédence. □

La relation de précédence permet de définir les notions de filiation entre nœuds (vision horizontale des relations entre éléments d'un arbre). Si $C_1 \prec C_2$ on dira que C_1 et C_2 sont *frères*.

Définition 5. *Soit $t \in T_{\Sigma_{MG}}(A)$, et $C_1, C_2 \in \mathcal{S}_t$, on dit que C_1 se **projette immédiatement** sur C_2 (noté $C_1 < C_2$) s'il existe $C \in \mathcal{S}_t$ tel que l'une des deux propriétés suivantes est vérifiée :*

1. $C_1 = C[<(x_1, t_2)]$ *et* $C_2 = C[<(t_1, x_1)]$
2. $C_1 = C[>(t_2, x_1)]$ *et* $C_2 = C[>(x_1, t_1)]$

On remarque que dans ce cas $C \lhd C_1$ et $C \lhd C_2$. Donc si $C_1 < C_2$ ou $C_2 < C_1$ alors il existe C tel que $C \lhd C_1$ et $C \lhd C_2$.

On note $<^\sim$ la plus petite relation définie par le système de règles suivant :

$$\frac{C \in \mathcal{S}_t}{C <^\sim C}\,[0] \qquad \frac{C_1 <^\sim C_2 \quad C_2 <^\sim C_3}{C_1 <^\sim C_3}\,[trans] \qquad \frac{C_1 < C_2}{C_1 <^\sim C_2}\,[\sim]$$

$$\frac{C_1 \lhd^* C_2 \quad C_3 \lhd^* C_4 \quad C_2 < C_3}{C_1 <^\sim C_4}\,[A] \qquad \frac{C_1 \lhd C_2 \quad C_2 < C_3}{C_2 <^\sim C_1}\,[B]$$

On note $C_1 <^+ C_2$ la relation définie par $C_1 <^\sim C_2$ et $C_1 \neq C_2$.

*On appelle $<^\sim$ la relation de **projection** et $<^+$ la relation de **projection stricte**. On dit que C_1 se projette (resp. se projette strictement) sur C_2 si $C_1 <^\sim C_2$ (resp. $C_1 <^+ C_2$).*

Les trois premières règles sont analogues à celles utilisées pour la précédence. La règle $[B]$ indique que si un contexte se projette sur son frère, il se projette sur son père. La règle $[A]$ peut se résumer par : la relation de projection passe au travers de la relation de dominance.

Lemme 6. *À partir des règles $[A]$, $[0]$ et $[\sim]$ on peut dériver les deux règles suivantes :*

$$\frac{C_1 \lhd C_2 \quad C_2 < C_3}{C_1 <^\sim C_3}\,[A_{bis}] \qquad \frac{C_1 < C_2 \quad C_2 \lhd^* C_3}{C_1 <^\sim C_3}\,[A_{ter}]$$

qui séparent les relations de projections entre relation verticale et horizontale. A_{bis} est le passage de la projection d'un fils sur l'autre à celle du père sur un des fils. A_{ter} est la descente de la projection dans l'arborescence.

Démonstration.

1. A_{bis} est obtenue à partir des règles $[0]$ et $[A]$:
$$\frac{\dfrac{C_1 \lhd C_2}{C_1 \lhd^* C_2} \qquad \dfrac{}{C_3 \lhd^* C_3}\,[0] \qquad C_2 < C_3}{C_1 <^\sim C_3}\,[A]$$
2. De même pour A_{ter} : $\dfrac{\dfrac{}{C_1 \lhd^* C_1}\,[0] \qquad C_2 \lhd^* C_3 \quad C_1 < C_2}{C_1 <^\sim C_3}\,[A]$

□

On remarquera que par définition la relation de projection est réflexive et transitive. Nous montrerons dans la suite de ce chapitre l'antisymétrie de cette relation. Cette relation est en fait un ordre sur les contextes.

Lemme 7. *La relation de projection passe au contexte :*

$$C_1 < C_2 \Rightarrow C[C_1] < C[C_2]$$

$$C_1 <^\sim C_2 \Rightarrow C[C_1] <^\sim C[C_2]$$

$$C_1 <^+ C_2 \Rightarrow C[C_1] <^+ C[C_2]$$

Démonstration. Par définition de la projection immédiate, pour $C_1, C_2 \in S_t$ si $C_1 < C_2$, il existe $C' \in S_t$ tel que :

1. ou bien $C_1 = C'[<(x_1, t_2)]$ et $C_2 = C'[<(t_1, x_1)]$ donc pour tout $C \in \mathcal{C}^1$ on a :
   (a) $C[C_1] = C[C'[<(x_1, t_2)]]$ et
   (b) $C[C_2] = C[C'[<(t_1, x_1)]]$

   et donc $C[C_1] < C[C_2]$.
2. ou bien $C_1 = C'[>(t_2, x_1)]$ et $C_2 = C'[>(x_1, t_1)]$donc pour tout $C \in \mathcal{C}^1$ on a :

(a) $C[C_1] = C[C'[>(t_2, x_1)]]$ et

(b) $C[C_2] = C[C'[>(x_1, t_1)]]$

et donc $C[C_1] < C[C_2]$.

Dans tous les cas, la projection immédiate passe au contexte.

Pour montrer la propriété pour la projection, on utilise une induction sur la dérivation de $C_1 <^{\sim} C_2$:

1. règle [0] : si $C_1 = C_2$ alors $C[C_1] = C[C_2]$, on obtient $C[C_1] <^{\sim} C[C_2]$.
2. règle [$\sim$] : si $C_1 < C_2$, d'après ce qui précède $C[C_1] < C[C_2]$, on obtient $C[C_1] <^{\sim} C[C_2]$.
3. règle [*trans*] : si $C_1 <^{\sim} C_3$ et $C_3 <^{\sim} C_2$, par hypothèse d'induction $C[C_1] <^{\sim} C[C_3]$ et $C[C_3] <^{\sim} C[C_2]$. Par transitivité, $C[C_1] <^{\sim} C[C_2]$.
4. règle [A] : si $C_1 \lhd^* C_4$, $C_3 \lhd^* C_2$ et $C_4 < C_3$, d'après le lemme 2 $C[C_1] \lhd^* C[C_4]$, $C[C_3] \lhd^* C[C_2]$ et d'après ce qui précède $C[C_4] < C[C_3]$. En utilisant la règle [A] : $C[C_1] <^{\sim} C[C_2]$.
5. règle [B] : si $C_2 \lhd C_1$ et $C_1 <^{\sim} C_3$, d'après le lemme 2 $C[C_2] \lhd C[C_1]$ et par hypothèse d'induction $C[C_1] <^{\sim} C[C_3]$. La règle [B] implique $C[C_1] <^{\sim} C[C_2]$.

Dans tous les cas possibles, la relation de projection passe au contexte.

La relation de projection stricte est un cas particulier de la relation de projection. □

On définit alors la notion de projection maximale d'un sous-terme comme étant le plus grand sous-arbre dont il est la tête.

Définition 8. *Soit $t \in T_{MG}$, $C \in S_t$. On appelle* ***projection maximale*** *de C (notée $proj_{max}(C)$) le sous-terme défini par :*

- *si $C = x_1$, $proj_{max}(C) = x_1$*
- *si $C = C'[< (x_1, t)]$ ou $C = C'[> (t, x_1)]$, $proj_{max}(C) = proj_{max}(C')$*
- *si $C = C'[< (t, x_1)]$ ou $C = C'[> (x_1, t)]$, $proj_{max}(C) = C$*

Par transposition directe de la définition de la projection maximale, on obtient les trois propriétés suivantes :

1. s'il n'existe aucun $C' \in \mathcal{C}_t^1$ tel que $C < C'$ alors $proj_{max}(C) = C$.
2. si $C' \lhd C$ et il existe $C'' \in S_t$ tel que $C < C''$ alors $proj_{max}(C) = proj_{max}(C')$.
3. si $proj_{max}(C) = proj_{max}(C')$ alors $C \lhd^* C'$ ou $C' \lhd^* C$.

À partir de la définition de la projection maximale, on peut prouver certaines propriétés que nous utiliserons pour montrer l'antisymétrie de la relation de projection.

Lemme 9. *Soit $t \in T_\Sigma$, $C_1, C_2 \in S_t$. Si $C_1 < C_2$, il existe $C \in S_t$ tel que $C \lhd C_1$, $C \lhd C_2$ et $proj_{max}(C_2) = C_2$.*

Lemme 10. *Soit $t \in T_{MG}$, $C \in S_t$: $proj_{max}(C) \lhd^* C$*

Démonstration. Par induction sur la structure de $\lhd^*$:

- soit $proj_{max}(C) = C$ d'après la réflexivité de la dominance $C \lhd^* C$, on en déduit $proj_{max}(C) \lhd^* C$.
- soit il existe $C_1, C_2 \in S_t$ tels que $C < C_2$ et $C_1 \lhd C, C_1 \lhd C_2$. Par définition de la projection maximale , $proj_{max}(C) = proj_{max}(C_1)$. Par induction on a $proj_{max}(C_1) \lhd^* C_1$ d'où $proj_{max}(C) = proj_{max}(C_1) \lhd^* C_1 \lhd C$.

donc $proj_{max}(C) \lhd^* C$. □

Lemme 11. *Soit $t \in T_{MG}$, $C_1, C_2 \in S_t$: si $C_1 \lhd^* C_2$ alors $proj_{max}(C_1) \lhd^* proj_{max}(C_2)$*

Démonstration. Par induction sur la structure de $\lhd_*$.

1. ou bien $C_1 = C_2$, on a alors $proj_{max}(C_1) = proj_{max}(C_2)$, d'où $proj_{max}(C_1) \lhd^* proj_{max}(C_1) = proj_{max}(C_2)$.
2. ou bien il existe C_3 tel que $C_1 \lhd C_3 \lhd^* C_2$. Par hypothèse d'induction, $C_3 \lhd^* C_2$ implique que $proj_{max}(C_3) \lhd^* proj_{max}(C_2)$.
   De plus, si $C_1 \lhd C_3$ on a :
   - ou bien $C_3 <^\sim C_1$ qui par définition implique que $proj_{max}(C_3) = proj_{max}(C_1)$ donc $proj_{max}(C_1) = proj_{max}(C_3) \lhd^* proj_{max}(C_2)$.
   - ou bien $C_1 <^\sim C_3$ qui par définition implique que $proj_{max}(C_3) = C_3$. On obtient donc $C_1 \lhd C_3 = proj_{max}(C_3) \lhd^* proj_{max}(C_2)$. D'après le lemme 10 $proj_{max}(C_1) \lhd^* C_1$ d'où $proj_{max}(C_1) \lhd^* proj_{max}(C_2)$.

□

Lemme 12. *Soit $t \in T_{MG}$, $C_1, C_2 \in S_t$: si $C_1 < C_2$ alors $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$*

Démonstration. d'après le lemme 9, si $C_1 < C_2$, alors il existe $C \in S_t$ tel que $C \lhd C_1$, $C \lhd C_2$ et $proj_{max}(C_2) = C_2$. D'après la définition de la projection maximale, $proj_{max}(C) = proj_{max}(C_1)$. Or d'après le lemme 10 $proj_{max}(C) \lhd^* C$.

On a donc $proj_{max}(C_1) = proj_{max}(C) \lhd^* C \lhd C_2 = proj_{max}(C_2)$, d'où

$$proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$$

□

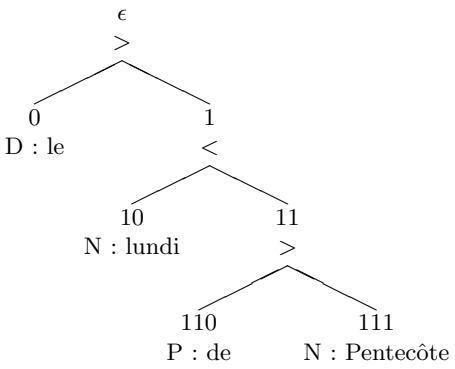
2.3 Structures linguistiques

Du point de vue de la linguistique, les arbres permettent de représenter des relations grammaticales entre les éléments d'un énoncé. Des concepts linguistiques sont associés aux structures particulières introduites dans les arbres minimalistes. Ces relations ont été proposées par analyse des analogies structurelles entre groupes verbaux et nominaux, par exemple *les enfants ont voyagé en avion* et *le voyage des enfants en avion.*

Ainsi, les groupes de mots cohérents dans un énoncé (syntagmes), quel que soit leur nature, ont une structure similaire. Celle-ci est supposée la même pour toutes les langues, indépendamment de l'ordre des sous-termes. Ces hypothèses sont à la base de la théorie X-barre introduite dans les années soixante-dix, [Cho73] et de celle du programme minimaliste de Chomsky, [Cho95], comme nous l'avons évoqué dans le chapitre précédent.

Nous utiliserons l'exemple du groupe nominal : *le lundi de Pentecôte* pour mettre en évidence les différentes définitions des structures linguistiques. L'arbre minimaliste de *le lundi de Pentecôte* est présenté dans la figure 10.

Dans l'arbre ci-dessous, les noeuds sont étiquetés par leur position dans l'arbre en accolant un 0 à la valeur du père pour le fils gauche et un 1 pour le fils droit. La racine est alors étiquetée ϵ.



$$
\begin{array}{lcl}
C_\epsilon & = & [] \\
C_0 & = & >(x_1, <(N:\text{lundi}, >(P:\text{de}, N:\text{Pentecôte}))) \\
C_1 & = & >(D:\text{le}, x_1) \\
C_{10} & = & >(D:\text{le}, <(x_1, >(P:\text{de}, N:\text{Pentecôte}))) \\
C_{11} & = & >(D:\text{le}, <(N:\text{lundi}, x_1))) \\
C_{110} & = & >(D:\text{le}, <(N:\text{lundi}, >(x_1, N:\text{Pentecôte})))) \\
C_{111} & = & >(D:\text{le}, <(N:\text{lundi}, >(P:\text{de}, x_1))))
\end{array}
$$

tête : C_{10} est la tête de t. C_{111} est la tête de C_{11}
Spécifieur : C_0 est le spécifieur de C_{10}
Compléments : C_{11}, C_{110} et C_{111} sont compléments de C_{10}
Projections maximales : La projection maximale de C_{10} est C_ϵ. La projection maximale de C_{110} est C_{11}.

FIG. 10 – Un arbre minimaliste t, les termes le composant et des exemples de relations entre les termes.

2.3.1 Tête

Linguistiquement la **tête** est l'élément autour duquel se constitue un groupe de mots. Dans le cas d'une phrase, les différents groupes nominaux (sujet, compléments, ...) se distribuent relativement au *verbe*.

Définition 13. *Soit $t \in T_{MG}$. Si pour tout $C' \in \mathcal{S}_t, C <^\sim C'$ alors C est appelé **tête** de t.*

Par extension, on dira de toute feuille qu'elle est au moins la tête du sous-arbre réduit à elle-même.

Lemme 14. *Pour tout sous-arbre d'un arbre minimaliste, il existe un unique élément minimal pour la relation de projection et celui-ci est une feuille.*

Démonstration.
- Existence : par induction sur la taille de t.

1. si $t = a, a \in A$. Par définition, l'arbre réduit à un seul élément est une feuille et elle est sa propre tête.
2. si $t = <(t_1, t_2)$: par hypothèse d'induction on suppose que C_1 est la tête de t_1 et C_2 celle de t_2. Pour $C' \in S_{t_1}, C_1 <^\sim C'$.
   Soit $C' \in S_t$, on a alors :

– ou bien $C' = <(C'', t_2)$. Donc $C'' \in S_{t_1}$ et on a $C_1 <^{\sim} C''$ et d'après le lemme 7 on obtient $<(C_1, t_2) <^{\sim} <(C'', t_2)$.
– ou bien $C' = <(t_1, C'')$. Donc $C'' \in S_{t_2}$ et on a $<(t_1, x_1) \lhd^* <(t_1, C'')$.
Or, on a $C_1 <^{\sim} x_1$ et par le lemme 7 on obtient $<(C_1, t_2) <^{\sim} <(x_1, t_2)$.
Par définition $<(x_1, t_2) < <(t_1, x_1)$. Pour $C' = x_1$ et en utilisant la règle $[A_{ter}]$ avec $<(t_1, x_1) \lhd^* <(t_1, C'')$, on obtient $<(x_1, t_2) <^{\sim} <(t_1, C'')$. La transitivité nous permet d'obtenir $<(C_1, t_2) <^{\sim} <(t_1, C'')$.

3. si $t = >(t_2, t_1)$: ce cas est symétrique par rapport au précédent.

∘ L'unicité est une conséquence de l'antisymétrie de la relation de projection qui est démontrée dans le lemme 16 (pour la notion de projection maximale). □

Pour un arbre $t \in T_{MG}$, on note $H_t[x] \in S_t$ un arbre de t dont x est la tête et $head(t)$ la feuille tête de t. On obtient alors $t = H_t[head(t)]$.

La démonstration du lemme ci-dessus montre que l'on peut aisément trouver la tête d'un arbre en suivant le sens de projection marqué sur le nœud. Dans l'exemple de la figure 10, on calcule les têtes. Pour trouver la tête du sous-arbre C_ϵ, le premier nœud nous indique qu'elle est vers la droite, le nœud suivant qu'elle est vers la gauche ; la feuille C_{10} est alors la tête de t. De plus, C_{10} est aussi la tête du sous-terme C_1 et C_{111} celle du sous-terme C_{11}.

2.3.2 Projection maximale

On souhaite donner une caractérisation logique de la relation de projection. Pour cela, nous allons revenir sur plusieurs propriétés des relations précédemment introduites. Soit $t \in T_{MG}, C \in S_t$, on a la proposition suivante :

Lemme 15. *Soit $t \in T_{MG}$ et $C_1, C_2 \in S_t$, tels que $C_1 <^{\sim} C_2$ on a alors soit $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ soit $(C_2 \lhd^* C_1$ et $proj_{max}(C_1) = proj_{max}(C_2))$.*

Démonstration. Par induction sur la dérivation de $C_1 <^{\sim} C_2$:

1. Si $C_1 = C_2$ on a explicitement $proj_{max}(C_2) = proj_{max}(C_1)$ et $C_2 \lhd^* C_2 = C_1$
2. Si $C_1 < C_2$, le lemme 12 permet de conclure que $proj_{max}(C1) \lhd^+ proj_{max}(C_2)$.
3. À partir de la règle de transitivité :

$$\frac{C_1 <^{\sim} C_3 \quad C_3 <^{\sim} C_2}{C_1 <^{\sim} C_2}\,[trans]$$

Quatre cas sont à étudier :

(a) Si $proj_{max}(C_1) \lhd^+ proj_{max}(C_3)$ et
   i. $proj_{max}(C_3) \lhd^+ proj_{max}(C_2)$: par transitivité $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$.
   ii. $C_2 \lhd^* C_3 \wedge proj_{max}(C_2) = proj_{max}(C_3)$:
   on a $proj_{max}(C_1) \lhd^+ proj_{max}(C_3) = proj_{max}(C_2)$,
   d'où $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$.

(b) si $C_3 \lhd^* C_1 \wedge proj_{max}(C_3) = proj_{max}(C_1)$ et
   i. $proj_{max}(C_3) \lhd^+ proj_{max}(C_2)$:
   on a $proj_{max}(C_1) = proj_{max}(C_3) \lhd^+ proj_{max}(C_2)$.

ii. $C_2 \lhd^* C_3 \wedge proj_{max}(C_2) = proj_{max}(C_3)$:
on a $C_2 \lhd^* C_3 \lhd^* C_1$ et $proj_{max}(C_1) = proj_{max}(C_3) = proj_{max}(C_2)$.

4. à partir de la règle $[A]$:

$$\frac{C_1 \lhd^* C_3 \quad C_4 \lhd^* C_2 \quad C_3 < C_4}{C_1 <^{\sim} C_2}\,[A]$$

le lemme 11 nous permet d'obtenir :
   (a) $C_1 \lhd^* C_3 \Rightarrow proj_{max}(C_1) \lhd^* proj_{max}(C_3)$
   (b) $C_4 \lhd^* C_2 \Rightarrow proj_{max}(C_4) \lhd^* proj_{max}(C_2)$
   D'après le lemme 12, pour $C_3 < C_4$ on a alors $proj_{max}(C_3) \lhd^+ proj_{max}(C_4)$. Par transitivité, nous obtenons $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$.

5. à partir de la règle $[B]$:

$$\frac{C_2 \lhd C_1 \quad C_1 < C_3}{C_1 <^{\sim} C_2}\,[B]$$

Par définition de la projection maximale, si $C_2 \lhd C_1$ et $C_1 < C_3$, $proj_{max}(C_1) = proj_{max}(C_2)$, de plus, si $C_2 \lhd C_1$ alors $C_2 \lhd^* C_1$.

□

À partir de ce lemme, on peut montrer l'antisymétrie de la relation de projection.

Lemme 16. *Soit $t \in T_{MG}$ et $C_1, C_2 \in S_t$, si $C_1 <^{\sim} C_2$ et $C_2 <^{\sim} C_1$ alors $C_1 = C_2$.*

Démonstration. On utilise le lemme 15 sur chacune des projections :
Soit $C_1 <^{\sim} C_2$ et $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ et

1. ou bien $C_2 <^{\sim} C_1$ et $proj_{max}(C_2) \lhd^+ proj_{max}(C_1)$. La relation de dominance stricte n'étant pas réflexive, deux éléments ne peuvent pas se dominer strictement simultanément. Ce cas n'est pas possible.
2. ou bien $C_2 <^{\sim} C_1$ et $C_2 \lhd^* C_1 \wedge proj_{max}(C_1) = proj_{max}(C_2)$. Or $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ implique que $proj_{max}(C_1) \neq proj_{max}(C_2)$. On obtient alors une contradiction donc ce cas n'est pas possible.

Soit $C_1 <^{\sim} C_2$ et $C_1 \lhd^* C_2 \wedge proj_{max}(C_1) = proj_{max}(C_2)$ et

1. ou bien $C_2 <^{\sim} C_1$ et $proj_{max}(C_2) \lhd^+ proj_{max}(C_1)$: situation symétrique par rapport à la précédente. Cette configuration n'est pas possible.
2. ou bien $C_2 <^{\sim} C_1$ et $C_2 \lhd^* C_1 \wedge proj_{max}(C_1) = proj_{max}(C_2)$. On a donc $C_1 \lhd^* C_2$ et $C_2 \lhd^* C_1$ ce qui, par antisymétrie de la dominance, implique que $C_1 = C_2$,

□

La conséquence directe de ce lemme est que la relation de projection est bien un ordre sur les contextes des arbres minimalistes.

Pour montrer la réciproque du lemme 15, nous séparons chacune des parties de la disjonction.

Lemme 17. *Soit $t \in T_{MG}$ et $C_1, C_2 \in S_t$, si $C_2 \lhd^* C_1$ et $proj_{max}(C_1) = proj_{max}(C_2)$ alors $C_1 <^{\sim} C_2$.*

Démonstration. Par induction sur la relation de dominance $\lhd^*$:
- Si $C_1 = C_2$, on a bien $C_2 \lhd^* C_1$ et $proj_{max}(C_1) = proj_{max}(C_2)$.
- Sinon, il existe $C_3 \in S_t$ tel que $C_2 \lhd^* C_3 \lhd C_1$. On a alors
  - ou bien $C_3 <^{\sim} C_1$: donc il existe C_4 tel que $C_4 < C_1$ et $C_3 \lhd C_4$.

    On sait que $C_2 \lhd^* C_3$ et d'après le lemme 10 $proj_{max}(C_2) \lhd^* proj_{max}(C_3) = proj_{max}(C_4)$.

    De plus, on sait que $C_4 < C_1$ et d'après le lemme 12 $proj_{max}(C_4) \lhd^+ proj_{max}(C_1)$.

    D'où $proj_{max}(C_2) \lhd^+ proj_{max}(C_1)$. Ce qui est en contradiction avec l'hypothèse selon laquelle $proj_{max}(C_1) = proj_{max}(C_2)$,
  - ou alors $C_1 <^{\sim} C_3$ implique par définition $proj_{max}(C_1) = proj_{max}(C_3)$.
    On a donc $C_2 \lhd^* C_3$ et $proj_{max}(C_2) = proj_{max}(C_1) = proj_{max}(C_3)$. Par hypothèse d'induction, on en déduit $C_3 <^{\sim} C_2$.
    Ce qui nous permet de conclure par transitivité avec $C_1 <^{\sim} C_3$ que $C_1 <^{\sim} C_2$.

□

Afin de montrer la deuxième partie de la réciproque du lemme 15, on introduit deux autres lemmes portant sur les conséquences de la différence entre deux projections maximales pour deux contextes différents.

Lemme 18. *Si $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ alors C_2 ne peut pas dominer C_1.*

Démonstration. Par induction sur la structure de la dominance :

1. Si $C_2 \lhd C_1$. D'après $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ on sait que $proj_{max}(C_1) \neq proj_{max}(C_2)$, d'où d'après la définition de la projection maximale $C_1 = proj_{max}(C_1)$. On obtient donc $proj_{max}(C_2) \lhd^* C_2 \lhd C_1 = proj_{max}(C_1)$ ce qui est en contradiction avec l'hypothèse de départ.
2. Si $C_2 \lhd^+ C_1$, alors il existe C_3 tel que $C_2 \lhd C_3 \lhd^* C_1$. On a alors :
   (a) ou bien $proj_{max}(C_3) = proj_{max}(C_1)$, d'où $proj_{max}(C_3) \lhd^+ proj_{max}(C_2)$. En utilisant $C_2 \lhd C_3$, d'après la preuve précédente, cela est en contradiction avec les hypothèses de départ.
   (b) ou bien $proj_{max}(C_3) \lhd^+ proj_{max}(C_1)$. On a donc $proj_{max}(C_3) \lhd^+ proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ et $C_2 \lhd C_3$. À nouveau, la preuve précédente implique que cette configuration est en contradiction avec les hypothèses de départ.
   (c) ou bien $proj_{max}(C_1) \lhd^+ proj_{max}(C_3)$ et $C_3 \lhd^* C_1$, par hypothèse d'induction, cette configuration est en contradiction avec les hypothèses de départ.

On obtient alors que $C_2 \lhd^* C_1$ n'est pas possible. □

Lemme 19. *Soit $t \in T_{MG}$ et $C_1, C_2 \in S_t$, si $proj_{max}(C_1) \lhd^+ proj_{max}(C_2)$ et $C_1 \lhd^* C_2$ alors il existe $C_3, C_4 \in S_t$ tels que $C_3 < C_4$, $C_4 \lhd^* proj_{max}(C_2)$, $proj_{max}(C_1) = proj_{max}(C_3)$ et $C_1 \lhd^* C_3$, $C_1 \lhd^* C_4$.*

Démonstration. Par induction sur la structure de la dominance :

1. ou bien $C_1 \lhd C_2$. Par définition, il existe C_3 tel que $C_1 \lhd C_3$.

– ou bien $C_2 < C_3$. Dans ce cas, on sait que $proj_{max}(C_3) = C_3$ et $proj_{max}(C_1) = proj_{max}(C_2)$. Or d'après les hypothèses, $proj_{max}(C_1) \neq proj_{max}(C_2)$. Ce cas est donc absurde.
– ou bien $C_3 < C_2$. Par définition, on sait que $proj_{max}(C_2) = C_2$ et $proj_{max}(C_1) = proj_{max}(C_3)$. Si on pose $C_4 = C_2$, on a alors $C_3 < C_4$, de plus $C_4 \lhd^* C_4 = C_2 = proj_{max}(C_2)$. On rappelle que par définition $C_1 \lhd C_3$ et $proj_{max}(C_1) = proj_{max}(C_3)$.

Les propriétés sont donc vérifiées.

2. ou bien $C_1 \lhd^* C_2$, ce qui implique qu'il existe C_5 tel que $C_1 \lhd C_5 \lhd^* C_2$.
   (a) ou bien $proj_{max}(C_1) = proj_{max}(C_5)$. On a donc $proj_{max}(C_5) \lhd^+ proj_{max}(C_2)$. Par hypothèse d'induction sur $C_5 \lhd^* C_2$, il existe C_3, C_4 tels que $C_3 < C_4$, $C_4 \lhd^* proj_{max}(C_2)$, $proj_{max}(C_5) = proj_{max}(C_3)$ et $C_5 \lhd^* C_3$, $C_5 \lhd^* C_4$.
   Or $proj_{max}(C_5) = proj_{max}(C_1)$, d'où $proj_{max}(C_1) = proj_{max}(C_3)$ et $C_1 \lhd C_5$. Les propriétés sont donc vérifiées.
   (b) ou bien $proj_{max}(C_1) \neq proj_{max}(C_5)$. Dans ce cas, il existe C_6 tel que $C_6 < C_5$, $C_1 \lhd C_6$, $proj_{max}(C_1) = proj_{max}(C_6)$ et $proj_{max}(C_5) = C_5$.
   On sait que $C_5 \lhd_* C_2$, d'après le lemme 11 $proj_{max}(C_5) \lhd^* proj_{max}(C_2)$. Or $C_5 = proj_{max}(C_5) \lhd^* proj_{max}(C_2)$. Pour $C_3 = C_6$ et $C_4 = C_5$ les propriétés sont vérifiées.

□

Lemme 20. *Soit $t \in T_{MG}$ et $C_1, C_2 \in S_t$, $proj_{max}(C_1) \lhd^+ proj_{max}(C_2) \Rightarrow C_1 <^{\sim} C_2$*

Démonstration. Par énumération des cas possibles :

1. Soit $C_1 \lhd^* C_2$ En utilisant le lemme 19, on obtient qu'il existe C_3 et C_4 tels que $C_3 < C_4$ et $C_4 \lhd^* proj_{max}(C_2)$ et $proj_{max}(C_1) = proj_{max}(C_3)$.
   Puisque $proj_{max}(C_1) = proj_{max}(C_3)$ alors :
   (a) ou bien $C_1 \lhd^* C_3$: qui permet de conclure par l'utilisation de la règle $[A]$.
   $$\frac{C_1 \lhd^* C_3 \quad C_4 \lhd^* proj_{max}(C_2) \lhd^* C_2 \quad C_3 < C_4}{C_1 <^{\sim} C_2}\,[A]$$
   (b) ou bien $C_3 \lhd^* C_1$. En utilisant l'égalité des projections maximales, le lemme 17 permet de conclure $C_1 <^{\sim} C_2$.
2. Soit $C_2 \lhd^* C_1$, d'après le lemme 18, ce cas n'est pas possible.
3. Soit il existe C_3, C_4, C_5 tels que $C_3 \lhd C_4$, $C_3 \lhd C_5$, $C_4 \lhd^* C_1$, $C_5 \lhd^* C_2$ et pour que l'hypothèse de départ soit vérifiée $proj_{max}(C_1) = proj_{max}(C_5) = proj_{max}(C_3)$ et $C_5 < C_6$. D'après le lemme 17, $C_1 <^{\sim} C_5$.
   De plus
   $$\frac{C_1 <^{\sim} C_5 \qquad \dfrac{C_5 < C_6 \quad C_6 \lhd^* C_2}{C_5 <^{\sim} C_2}\,[A_{ter}]}{C_1 <^{\sim} C_2}\,[trans]$$

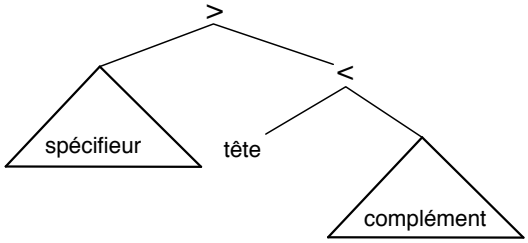
□

Par abus de langage et souci de simplification, nous dirons qu'un nœud est la projection maximale d'une feuille, si ce nœud est la racine de la projection maximale.

Dans l'exemple de la figure 10 on obtient les projections maximales suivantes : le nœud C_{10} est la tête des nœuds C_1 et C_ϵ, or $C_\epsilon \lhd C_1$, donc $proj_{max}(C_{10}) = C_\epsilon$. De la même manière, $proj_{max}(C_{110}) = C_{11}$.

2.3.3 Complément et Spécifieur

On introduit une terminologie sur les sous-arbres par rapport à la tête. Les éléments venant après la tête ont pour rôle de lui apporter de l'information. Dans une phrase en français, la tête de l'arbre d'analyse est le verbe et c'est l'objet de ce verbe qui prend cette position (cela peut aussi être une autre phrase, ...). Dans ce cas, ces éléments sont en position de **complément** avec la tête. A contrario, les éléments placés devant la tête ont pour rôle de déterminer qui (ou quoi) est dans la relation (pour une phrase, ce sera le sujet de la phrase). Ces éléments sont dits en relation de **spécifieur** de la tête. Nous représentons graphiquement ces notions de tête, complément et spécifieur par l'arbre :



Les définitions de complément et de spécifieur données ici sont une réalisation dans ce formalisme des notions de complément et spécifieur définies dans le chapitre précédent.

La notion de **complément** est un raffinement de position entre un sous-terme d'un arbre et la tête par la relation de précédence qui les relie. Soit $t \in S_{MG}$, C_1 est en position de complément de $head(t) = C$, si $proj_{max}(C) \lhd^* C_1$ et $C \prec^+ C_1$. On note alors C_1 *comp* C.

Sur l'exemple de la figure 10, d'après les définitions, la tête de l'arbre est $head(t) = C'[10]$. Le sous-terme $C''[11]$ est la plus grande projection maximale qui succède à la tête de l'arbre. On a donc $C''[11]$ *comp* $C'[10]$.

La notion de **spécifieur** est symétrique par rapport à celle de *complément*. Soit, $t \in S_{MG}$, C_1 est un spécifieur de $head(t) = C$, si $proj_{max}(C) \lhd^* C_1$ et $C_1 \prec^+ C$. On note alors C_1 *spec* C la relation de spécifieur entre deux sous-termes d'un arbre minimaliste.

Sur l'exemple de la figure 10, d'après les définitions, la tête de l'arbre est C_{10}. Le sous-terme C_0 précède la tête. On a donc C_0 *spec* C_{10}.

Pour finir l'analyse des rapports entre les constituants de cet exemple, du point de vue linguistique, nous pouvons dire que : $(D : le)$ est un spécifieur de la tête $(N : lundi)$ et $((P : de)(N : Pentecôte))$ est un complément de cette tête. $((D : le)(N : lundi)((P : de)(N : Pentecôte)))$ est la projection maximale de la tête $(N : lundi)$.

2.4 Grammaires minimalistes

Nous abordons à présent la définition des GMs proposée par Ed Stabler, [Sta97]. Le système calculatoire est entièrement basé sur la notion de traits. À partir de ces derniers, des règles de composition sont définies afin d'établir les structures grammaticales sur les arbres minimalistes. Les traits sont associés à des items lexicaux (nous reviendrons sur la rédaction et l'utilisation des lexiques). Les différents types de traits marquent l'utilisation linguistique qui est faite des items lexicaux dans ces grammaires.

2.4.1 Définition

Les étapes des dérivations réalisées par les GMs sont déclenchées par les traits des entrées lexicales. Ces grammaires sont entièrement lexicalisées et de fait définies par la donnée de leur lexique. Les règles de composition des expressions formées sont quant à elles toujours les mêmes.

Une **grammaire minimaliste** est définie par un quintuplet $\langle V, Traits, Lex, \Phi, c\rangle$ où :

$V = \{P \cup I\}$ est l'ensemble fini des traits non-syntaxique où,

P est l'ensemble des formes phonologiques et
I est l'ensemble des formes logiques

$Traits = \{B \cup S \cup L_a \cup L_e\}$ est l'ensemble fini des traits syntaxiques,
Lex est l'ensemble des expressions construites à partir de P et de $Traits^*$,
$\Phi = \{merge, move\}$ est l'ensemble des fonctions génératrices,
$c \in Traits$ est le trait acceptant.

Dans ces grammaires, chaque forme phonologique est utilisée comme entrée du lexique et comme forme associée à la liste de traits. Ces dernières constituent les "terminaux" de la grammaire. Une lecture gauche-droite des formes phonologiques sur les structures dérivées et acceptées permet de reconnaître la séquence de terminaux reconnue.

Le langage $L(G)$ reconnu par G, une grammaire minimaliste, est la clôture du lexique par les fonctions génératrices ϕ. À tout énoncé accepté par une GM correspondra un arbre minimaliste obtenu à partir des règles de composition sur lesquelles nous reviendrons. Leur fonctionnement permet, pour une phrase d'une langue naturelle, d'obtenir l'arbre d'analyse générativiste traditionnel.

2.4.2 Les traits

Comme nous l'avons dit, une GM est définie par son lexique qui stocke les ressources. Chaque entrée du lexique est décrite par une liste de traits qui encode son comportement lors d'une dérivation. Une GM contient des traits de deux sortes : **les traits syntaxiques** et **les traits non-syntaxiques**.

On note V l'ensemble des traits non syntaxiques composé :

- des **traits phonologiques** (forme phonologique - FP) notés entre barres obliques - / /.
- des **traits sémantiques** (forme logique/sémantique - FL) notés entre parenthèses - ( ).

L'ensemble des traits syntaxiques est construit à partir de deux sous-ensembles : l'ensemble des catégories de base, noté B et l'ensemble des traits de déplacement, noté D. En utilisant ces sous-ensembles, on définit les différents types de traits utilisés dans les listes des items lexicaux de la grammaire :

- soit $B = \{v,\ dp,\ c, \cdots\}$ l'ensemble des **catégories de base**,
- soit $S = \{=d \mid d \in B\}$ l'ensemble des **sélecteurs**,
- soit $L_a = \{+k \mid k \in D\}$ l'ensemble des **assignateurs**,
- soit $L_e = \{-k \mid k \in D\}$ l'ensemble des **assignés**.

Revenons sur le rôle et la signification de chacun de ces éléments.

Lorsque l'on modélise une langue naturelle, les éléments de B dénotent des concepts linguistiques standards, par exemple v pour un verbe, n pour un nom, p pour une préposition, d pour un déterminant... Dans cet ensemble, on distingue un type particulier appelé **trait acceptant**, qui sera le symbole acceptant de la grammaire. Généralement on utilise le trait c qui représente la position *complementizer* de la dérivation, c'est-à-dire l'état auquel on vérifie la terminaison de la phrase. Dans ce cas, c est l'unique trait syntaxique présent dans l'arbre d'analyse et il est sur la tête.

Les *sélecteurs* expriment une demande par rapport à une autre expression possédant le trait de base *équivalent*. Si α est un trait de base, $=\alpha$ est un sélecteur. Il exprime la demande d'une expression possédant le même trait α.

Les *assignateurs* sont les traits qui assignent une propriété à une expression et qui sont dans une relation spécifieur-tête par rapport à celle qui les porte. À nouveau, lors de la modélisation d'une langue naturelle, les assignateurs sont utilisés pour apporter une propriété à une autre expression (c'est par exemple le cas pour les langues naturelles). Cette dernière vient occuper une nouvelle place en relation spécifieur par rapport à la tête.

Parallèlement, l'expression recevant le trait assigné doit être appropriée, autrement dit elle demande à recevoir ce trait. On traduit cela par le fait qu'elle possède le trait complémentaire de $+f$ qui est noté $-f$. Pour le traitement d'une langue naturelle, le correspondant du trait $+cas$ sera $-cas$, possédé uniquement par les groupes nominaux qui *doivent nécessairement recevoir un cas.*

L'union de ces ensembles forme l'ensemble des traits syntaxiques de la grammaire utilisés pour modéliser le comportement syntaxique des items lexicaux $Traits = \{B \cup S \cup L_e \cup L_a\}$.

À partir de ces ensembles, on définit la structure d'une *entrée lexicale* comme un élément de :

$$(S(S \cup L_a)^*)^* B(L_e)^*/FP/(FL)$$
$$\text{ou bien } B(L_e)^*/FP/(FL)$$

Les FL n'étant pas réellement prises en compte lors des dérivations, pour présenter les lexiques nous utiliserons la notation pour laquelle la liste de traits est séparée de la forme phonologique associée par deux points :

$$\text{FP : liste de traits}$$

Ces listes peuvent être reconnues par un automate régulier donné en figure 55, extension de la proposition de [Ver99]. Dans cette structure, on distingue deux parties, la première

contenant des sélecteurs et des assignateurs, traits déclenchant les règles comme nous allons le voir, puis un trait de base et des assignés, traits attendant d'être composés dans la suite de la dérivation.

De manière analogue, on peut les définir par une grammaire régulière (nous utiliserons cette définition dans un chapitre ultérieur) : soit b quelconque appartenant à B et d quelconque appartenant à D, la liste de traits associée à un item lexical est reconnue par la grammaire :

$$
\begin{aligned}
L &::= {=}b\ S_1 \mid B \\
S_1 &::= {=}b\ S_1 \mid {+}d\ S_1 \mid B \\
B &::= b\ S_2 \mid b \\
S_2 &::= -d\ S_2 \mid -d
\end{aligned}
$$

Pour la suite, nous adopterons les notations suivantes : e est un trait d'un type arbitraire et E une suite de traits éventuellement vide. Une entrée lexicale est composée d'au moins une suite de traits et d'une forme phonologique associée notée entre barres obliques : $e_1E/\zeta_1/$.

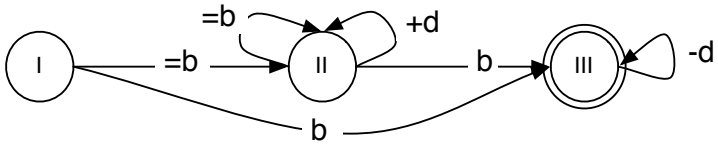


FIG. 11 – Structure de la liste de traits d'un item lexical.

2.4.3 Opérations

Soit Φ l'ensemble des **fonctions génératrices**. Φ contient deux types d'opération sur les arbres minimalistes : la **fusion** (*merge*) et le **déplacement** (*move*), chacune de ces opérations pouvant être raffinée. Les déclenchements de la fusion et du déplacement sont conditionnés par le premier trait de la liste sur la tête de l'arbre.

La suite des opérations qui interviennent dans une dérivation permet de générer plusieurs types de représentations : l'arbre d'analyse (arbre minimaliste obtenu après acceptation de l'énoncé) et l'ensemble des arbres dérivés (arbres intermédiaires vers l'arbre d'analyse). La suite des arbres dérivés peut être recalculée à partir de l'arbre d'analyse et de la succession des règles utilisées pour l'obtenir. En général, nous nous efforcerons de donner les différentes étapes produites lors d'une analyse en se basant sur ces arbres intermédiaires.

Pour A une grammaire minimaliste, on note $\mathcal{T}_\mathcal{G} = T_{MG}(A)$.

La fusion

La *fusion (merge)* est une opération qui unit deux arbres pour en former un troisième. Cette opération est déclenchée par la présence d'un sélecteur et d'un trait de base correspondant. $merge : T_{MG} \times T_{MG} \rightarrow T_{MG}$. Les traits utilisés pour cette opération sont alors effacés.

Soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l :: =h\ E]$ et $t' = H_{t'}[l' : h\ E']$ avec $h \in B$:

$$merge(t, t') = \begin{cases} < (l :: E, H_{t'}[l' :: E']) & \text{si } t \in Lex, \\ > (H_{t'}[l' :: E'], H_t[l :: E]) & \text{sinon.} \end{cases}$$

La représentation graphique de cette règle est donnée par la figure 56. La fusion est l'opération qui met en relation les différentes expressions construites au fur et à mesure de la dérivation. La tête du nouvel arbre pointe vers l'expression portant le sélecteur.

$\forall t, t' \in T_{MG}$ tels que $t = H_t[l :: =h\ E], t' = H_{t'}[l : h\ E']$ avec $h \in B$:

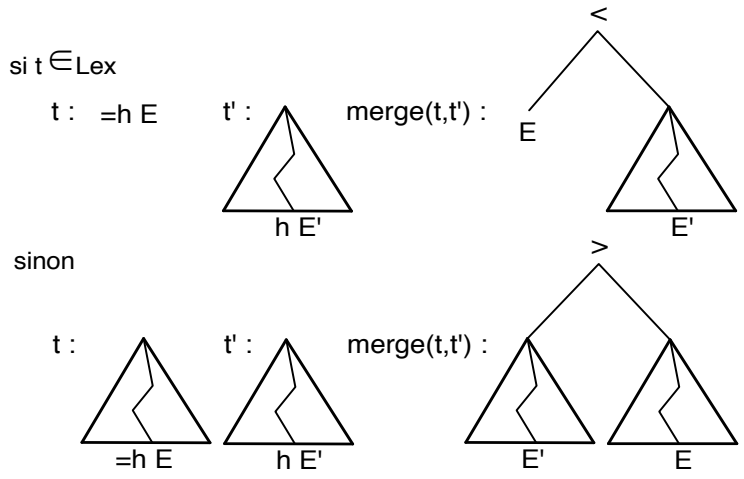


Fig. 12 – Représentation sous forme d'arbre de la fusion.

Le déplacement

Comme nous l'avons présenté dans le chapitre précédent, cette seconde opération est primordiale dans la définition de la grammaire générative. Elle correspond au déplacement effectif d'un constituant en première position de la dérivation, c'est-à-dire en haut de l'arbre dérivé. Elle réalise une restructuration d'un arbre minimaliste. La présence simultanée d'un élément de L_a en première position de la liste de traits de la tête et d'un élément de L_e équivalent en première position d'une liste de traits d'une des occurrences du même arbre la déclenche.

De manière intuitive la procédure est la suivante : lorsque le premier trait de la tête d'une dérivation est un assignateur $(+)$, on cherche dans le reste de la dérivation une feuille dont le premier trait est l'assigné $(-)$ équivalent. Si on en trouve un, on déclenche un déplacement en faisant passer en haut de l'arbre la projection maximale de la feuille portant l'assigné. La tête de la nouvelle expression reste celle de l'expression avant déplacement. Il se peut qu'il y ait plusieurs feuilles dont le premier trait est l'assigné $(-)$ équivalent. Le choix d'une feuille particulière rend l'opération de déplacement non-déterministe.

$move : T_{MG} \rightarrow T_{MG}$

pour tout arbre $t = C[l : +g\ E, l' : -g\ E']$, tel que $t = H_t[l : +g\ E]$.

Il existe $C_1, C_2 \in S_t$ tels que : C_2 est la projection maximale de la feuille l' et C_1 est t privé de C_2. On a alors $t = C_1[l : +g\ E, C_2[l' : -g\ E']]$.

– $C_2[l' : -g\ E'] = proj_{max}(C[l' : -g\ E])$
– $C_1[l : +g\ E, x_1] = proj_{max}(C[l : +g\ E, x_1])$

$$move(t) = >(C_2[l' : E'], C_1[l : E, \epsilon])$$

où ϵ est la feuille vide.

Si t' est obtenu par déplacement à partir de t, nous noterons que $t' \in move(t)$.

Le sous-arbre est alors en relation spécifieur-tête. Les deux traits ayant permis le déplacement sont alors supprimés. La représentation graphique des règles est présentée dans la figure 57.

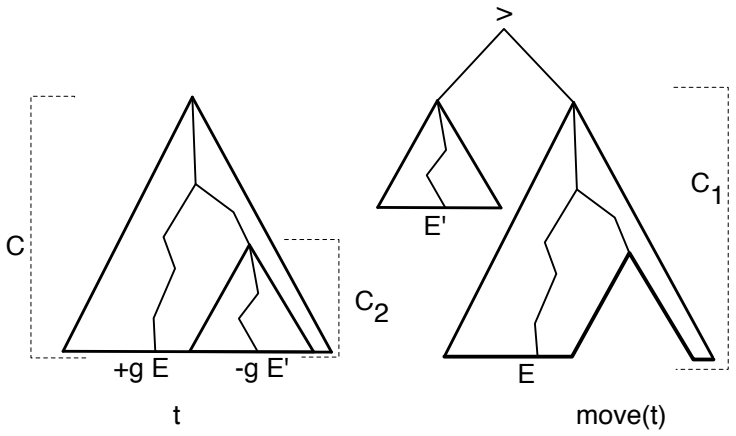


FIG. 13 – Représentation sous forme d'arbre du déplacement.

Comme nous l'avons vu dans le chapitre précédent, tous les déplacements ne sont pas envisageables. On ajoute (ou non) des conditions sur les déplacements possibles. Une condition majeure dans la définition même du programme minimaliste est la condition d'économie. Ainsi, en cas d'ambiguïté, le déplacement doit avoir lieu sur l'élément *le plus proche* de la tête. Cependant, les linguistes sont très partagés sur la définition d'une notion de proximité entre constituants. Pour ne pas trancher, nous suivons [Sta97] et nous utilisons une condition forte sur le principe d'économie, qui l'englobe, pour laquelle il ne doit pas y avoir d'ambiguïté dans le déclenchement d'un déplacement - *Shortest Move Condition* (SMC). Sous cette condition, l'opération de déplacement devient déterministe. Cela se traduit par la propriété si : $t = C[l : +g\ E, l' : -g\ E']$ alors pour tout $C', t = C'[l_1 :\ E_1, l_2 :\ E_2]$ et $E_1 = -h_1\ E'_1$ et $E_2 = -h_2\ E'_2$ et $h_2 \neq h_1$.

Une condition de localité peut aussi être utilisée, la *Specifier Island Condition* - SPIC. Les "islands" définissent les domaines qui interdisent des extractions. La SPIC impose que pour être déplacé, un élément ne doit pas être en position de spécifieur à l'intérieur d'un sous-arbre. Cette condition a été introduite par Ed Stabler dans [Sta99] en s'inspirant des travaux de [KS00] et [Kay98] qui proposent que les éléments déplacés soient uniquement en position de complément.

2.4.4 Autres opérations possibles

Stabler propose également un raffinement du déplacement. On distingue les déplacements dits **forts** des déplacements dits **faibles**. C'est alors une gestion particulière des formes phonologiques. Le *déplacement fort* déplacera toutes les composantes de l'élément, alors que le *déplacement faible* laissera en position initiale la forme phonologique.

On note les traits de déplacement fort en majuscules et les traits de déplacement faible en minuscules et on utilise cette notation tant pour les assignés que pour les assignateurs.

Le déplacement fort est le déplacement présenté dans la section déplacement. Le déplacement faible est alors, dans les mêmes conditions :

$$move(t) = >(C_2[\epsilon : E'], C_1[l : E, l'])$$

Graphiquement, les déplacements sont les mêmes, sauf pour la partie phonologique, comme le montre la figure 14 où $/l'/$ n'est pas déplacé dans sa position d'origine.

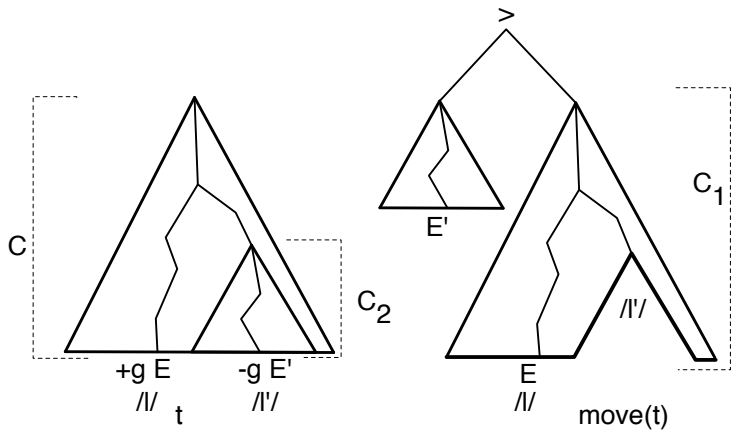


Fig. 14 – Représentation graphique du déplacement faible.

C'est en faisant varier la valeur *fort/faible* sur certains traits que l'on obtient des variations dans l'ordre des mots. C'est notamment ce qui permet d'analyser, à partir des mêmes règles, des langues SOV - Sujet-Objet-Verbe, comme le Japonais, ou SVO - Sujet-Verbe-Objet, comme le français. On donne de cette manière une réalisation de la notion de **paramètres** d'une langue naturelle, comme présentée dans le chapitre précédent.

On peut étendre les définitions précédentes pour obtenir un système plus performant dans la reconnaissance d'énoncés en langue naturelle, comme Stabler l'établit dans [Sta01].

En effet, plusieurs études linguistiques montrent que parfois seule la tête d'un constituant doit changer de position. C'est ce qu'on appelle **"Head Movement"**. Ceci se présente notamment dans le cas d'inversion du sujet et du verbe avec son inflexion dans les questions. Dans ce cas, on parle de transformation *T-to-C* (car le verbe quitte sa position de "verbe ayant reçu son inflexion" T pour monter en fin de dérivation C). Un exemple de question avec inversion du sujet sera donné comme illustration dans la section 2.5.

La partie phonologique déplacée peut avoir été construite par d'autres fusions de tête. De plus, son positionnement à droite ou à gauche de la nouvelle tête est déterminé par un marqueur spécifique sur le sélecteur.

Cette opération, contrairement à ce que son nom pourrait laisser supposer, n'est pas un nouveau type de déplacement, mais une réelle fusion, comme le montre [Sta01]. L'identification systématique de la tête est alors nécessaire lors de l'analyse.

La fusion de tête est déclenchée par un trait de sélection spécial.

On définit les deux ensembles de traits suivants :

Soit $S_g = \{=>x \mid x \in B\}$ l'ensemble des traits déclenchant une fusion de tête où la tête est adjointe à gauche.

Soit $S_d = \{x<= \mid x \in B\}$ l'ensemble des traits déclenchant une fusion de tête où la tête est adjointe à droite.

On définit la fusion de tête avec adjonction à gauche, sous les mêmes conditions que la fusion, par :

soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : =>h\ E]$ et $t' = H_{t'}[l' : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[l'l :\ E], H_{t'}[\epsilon :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[\epsilon :\ E'], (H_t[l'l :\ E]) & \text{sinon.} \end{cases}$$

où seules les positions des formes phonologiques sont différentes. Et la fusion de tête avec adjonction à droite, sous les mêmes conditions que la fusion, par :

soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : =>h\ E]$ et $t' = H_{t'}[l' : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[ll' :\ E], H_{t'}[\epsilon :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[\epsilon :\ E'], (H_t[ll' :\ E]) & \text{sinon.} \end{cases}$$

On peut remarquer que quelle que soit la GM utilisant les fusions de tête, il existe une GM sans fusion de tête équivalente sur les chaînes. Cependant, l'ajout de cette opération nous permet de conserver des analyses équivalentes à celles données classiquement en linguistique générative.

Revenons sur le cas de l'inflexion. Une étude de la position des adverbes en anglais montre que les énoncés sont de la forme Sujet-adverbe-inflexion-verbe-objet, contrairement au français où l'adverbe se positionne entre l'inflexion et le verbe. Les linguistes génératives expliquent cette variation par une descente de l'inflexion vers le verbe principal, comme nous l'avons exposé dans le chapitre précédent. Cette opération est appelée *Affix Hopping* et est modélisée par extension de *Head Movement*.

On définit les deux ensembles de traits suivants :

Soit $S_{ahg} = \{x=> \mid x \in B\}$ l'ensemble des traits déclenchant un Affix Hopping où la tête est adjointe à gauche.

Soit $S_{ahd} = \{<=x \mid x \in B\}$ l'ensemble des traits déclenchant un Affix Hopping où la tête est adjointe à droite.

On définit l'Affix-Hopping avec adjonction de la forme phonologique à gauche par :

soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : <=h\ E]$ et $t' = H_{t'}[l' : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[\epsilon :\ E], H_{t'}[l'l :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[l'l :\ E'], (H_t[\epsilon :\ E]) & \text{sinon.} \end{cases}$$

et l'Affix-Hopping avec adjonction de la forme phonologique à droite par :

soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : h=>\ E]$ et $t' = H_{t'}[l' : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[\epsilon :: E], H_{t'}[ll' :: E']) & \text{si } t \in Lex, \\ < (H_{t'}[ll' :: E'], (H_t[\epsilon :: E]) & \text{sinon.} \end{cases}$$

L'opération d'Affix-Hopping est donc une opération inverse de la fusion de tête par rapport à Head Movement, les deux ne différant de la fusion standard que pour la gestion des formes phonologiques.

Une dernière opération est parfois ajoutée dans l'utilisation des GMs. Nous ne ferons que l'évoquer ici car elle ne sera pas utilisée dans la suite. Il s'agit de l'**adjonction**. Une proposition de formalisation dans le cadre des GMs a été donnée dans [MG03]. Elle se fait par introduction d'un nouveau marqueur $\approx$ définissant un nouvel ensemble de traits. Cette opération est analogue à la fusion mais sa particularité réside dans la non-consommation du trait de base avec lequel il y a combinaison.

Enfin, une autre définition des GMs a été introduite à partir non plus des structures d'arbres mais des chaînes [Sta99]. Cette seconde version présente des avantages pour la partie calculatoire mais perd en qualité de représentation. Nous présentons à titre indicatif cette définition dans l'annexe A. Nous nous attacherons dans la suite de ce manuscrit à conserver la version des GMs sur les arbres minimalistes.

2.5 Exemples d'utilisation des GMs

Nous allons présenter les étapes d'analyses réalisées par une GM pour deux énoncés du français. Pour cela, nous présenterons les différents arbres dérivés qui permettent d'aboutir à l'arbre d'analyse. Chaque grammaire utilise les règles définies précédemment. Afin de simplifier la présentation, nous donnons uniquement le lexique nécessaire à la réalisation de la dérivation. Ainsi, nous supposons que les ensembles de traits sont induits par ces lexiques. Le trait acceptant est toujours le trait c.

Lors d'une analyse standard, on suppose que chaque groupe nominal attend une attribution de cas, ce qui se fera par une vérification de la présence simultanée dans une dérivation, d'une demande de cas et d'une donation de cas, (*i.e.* par un déplacement).

De plus, nous faisons entrer dans l'analyse l'inflexion du verbe se basant sur l'observation commune que seul un verbe conjugué peut prendre un sujet. Enfin, l'analyse se terminera par une vérification du fait que la phrase n'est pas une relative enchâssée ou une question, donc qu'en tant que telle, elle est une phrase simple.

2.5.1 Phrase standard

Les phrases standard en français respectent un ordre donné où le sujet est le premier élément de la phrase, suivi du verbe, lui-même suivi de son objet. Par exemple :

(23) *Pierre* *prend* *un* *train.*
S V O

Lexique 21.

Pierre :	*d -case*	$infl$:	$<=v$ *+case* V
un :	$=n$ *d -case*	*comp* :	$=V$ *c*
train :	*n*		
prendre :	$=d$ *+case* $=d$ *v*		

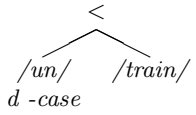
Nous revenons sur la composition de ce lexique. Dans ce lexique, nous introduisons un nom propre qui fonctionne alors comme un groupe nominal. Dans ce cas, il est de catégorie *d* - pour *determinal phrase* - et il attend une attribution de cas. De manière analogue, on trouve un nom commun et un déterminant. C'est le déterminant qui construit le groupe nominal, d'où l'appellation *determinal phrase*. Le verbe attend deux groupes nominaux et peut attribuer un cas.

Les deux autres entrées correspondent à deux étapes que nous introduisons systématiquement dans les analyses : d'une part l'inflexion, et d'autre part la partie *complementizer*. L'inflexion porte sur le verbe et introduit le cas nominatif (position T dans la dérivation). Ainsi seuls les verbes ayant reçu leur inflexion pourront recevoir pleinement leur sujet. La dernière entrée correspond à la phase de *complementizer* vérifiant la bonne terminaison de l'analyse (position *c* dans la dérivation). Ces deux entrées sont en réalité à phonologie vide (ϵ). Pour des raisons de compréhension nous les appellerons $infl$ et *comp*.

Dérivation 22.

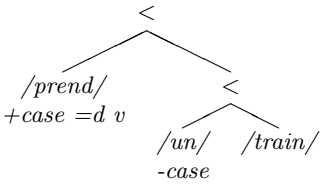
1. *Énumération* = {*Pierre, un, train, prendre, infl, comp*}
2. *Entrée lexicale "un" :* $=n$ *d* $-$*case*
   Entrée lexicale "train" : n

   Fusion entre les deux pour construire un groupe nominal :

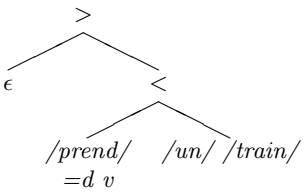


3. *entrée lexicale "prendre"* $=d$ *+case* $=d$ *v*

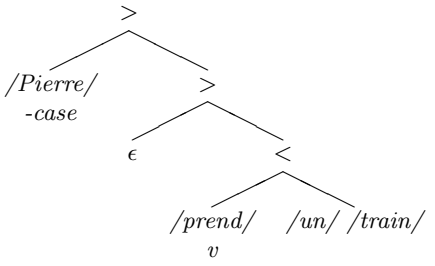
   Fusion : le verbe prend un groupe nominal qui sera l'objet du verbe de la phrase.



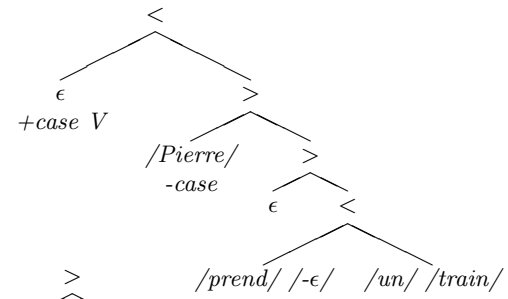
4. *Premier trait sur la tête de la structure : +case, donc on déclenche un déplacement faible. C'est la vérification du cas pour un groupe nominal. Cette position dans la dérivation est supposée être celle correspondant à l'accusatif (le verbe commence par recevoir son objet).*



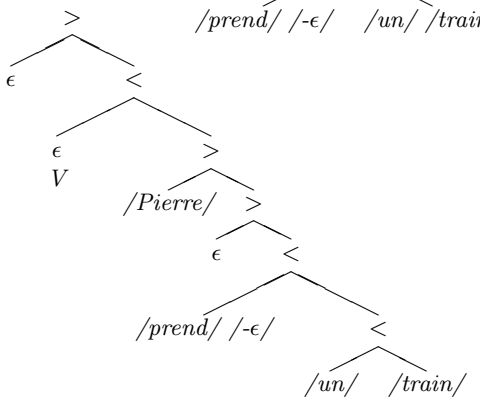
5. *Entrée lexicale "Pierre" : d −case*
   Fusion : entrée dans la dérivation du groupe nominal sujet.



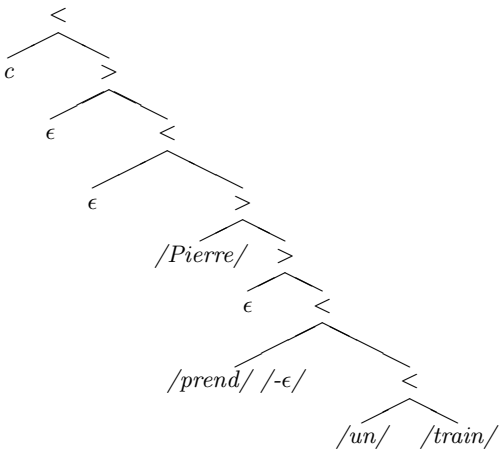
6. *Entrée lexicale "-ϵ" : <=v +case V*
   Affix hopping - descente de la forme phonologique de l'inflexion. Le verbe est maintenant considéré comme ayant reçu un temps, il peut donc recevoir un sujet. C'est l'inflexion qui apporte la donation de cas pour le nominatif dans la dérivation.



Déplacement : vérification du cas nominatif.



7. *Entrée lexicale "comp" : =V c*
8. *Fusion : vérification de l'unité de la phrase.*
9. *Symbole de la dérivation : 'c' . Acceptation et fin de la dérivation car c'est le seul trait présent dans la dérivation.*
10. *Une lecture gauche-droite des éléments dans l'arbre d'analyse donne la séquence analysée : "Pierre prend un train" - résultat de la lecture gauche droite des formes phonologiques.*



À partir de cet exemple d'analyse, on peut étendre à d'autres types de structures syntaxiques. Notamment les questions ou les relatives.

2.5.2 Phrase question

Dans les questions, l'analyse se déroule de la même manière et on obtient les constituants dans l'ordre Sujet-Verbe-Objet (SVO). C'est lors de la dernière étape (***comp***) que sera vérifiée la présence d'un trait de question. S'il y en a, le constituant sera déplacé en première position, et, pour le cas des questions sur l'objet, ce dernier sera déplacé pour obtenir l'ordre Objet-Sujet-Verbe. Il reste le problème de l'inversion du verbe et du sujet qui est réalisé par Head-Movement sur l'entrée *comp*. Présentons maintenant une dérivation de ce type.

(24) Quel train prend Pierre ?

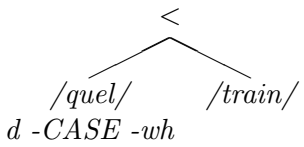
Lexique 23.

$Pierre :$	$d\ -case$	$infl :$	$=>v\ +case\ V_{inv}$
$quel :$	$=n\ d\ -CASE\ \text{-}\ wh$	$comp :$	$=>V_{inv}\ +WH\ c$
$train :$	n		
$prendre :$	$=d\ +case\ =d\ v$		

Pour inverser le verbe et le sujet, on utilise un trait particulier sur le verbe ayant reçu son inflexion. L'inversion entre le verbe et le sujet est assurée par une fusion de tête comme nous l'avons exposé dans la section 2.4.4, ce qui permet de remonter la forme verbale en haut de l'analyse, puis il y aura déplacement de l'objet qui passera en première position.

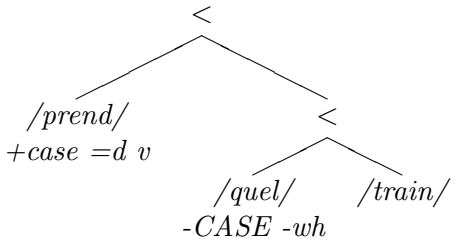
Dérivation 24. *Énumération* = {*Pierre, quel, train, prendre, infl, comp*}

1. *Entrée lexicale "quel" : =n d −CASE −wh*
   Entrée lexicale "train" : n
   Fusion : construction du groupe nominal auquel il manque un cas et une position de question.



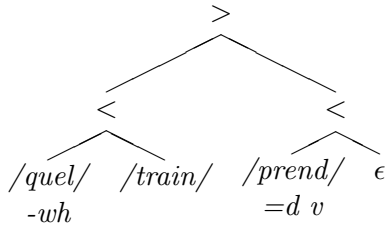
/quel/ /train/
d -CASE -wh

2. *Entrée lexicale "prendre" =d +case =d v*
   Fusion : mise en relation du verbe et d'un groupe nominal.



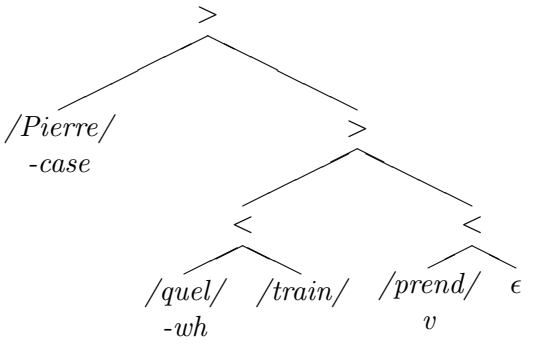
/prend/
+case =d v
/quel/ /train/
-CASE -wh

3. *Premier trait sur la tête de la structure : +case, donc on déclenche un déplacement fort : vérification du cas accusatif.*

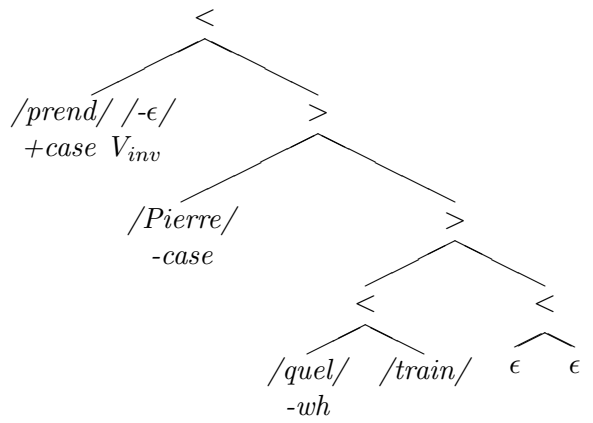


/quel/ /train/
-wh
/prend/ ϵ
=d v

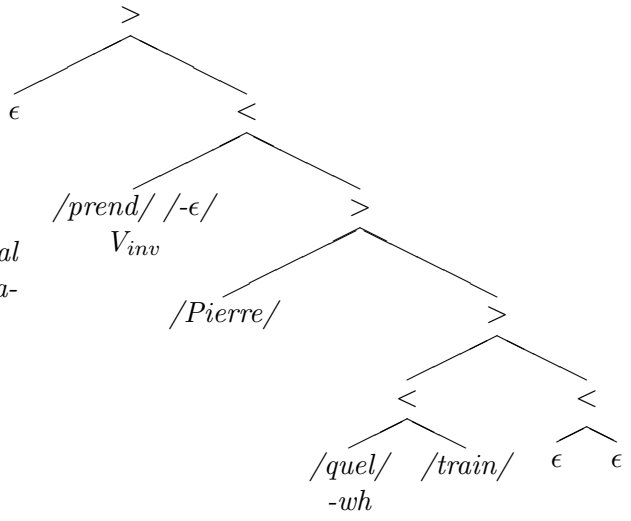
4. *Entrée lexicale "Pierre" : d −case*
   Fusion : mise en relation du second groupe nominal (un nom propre ici).

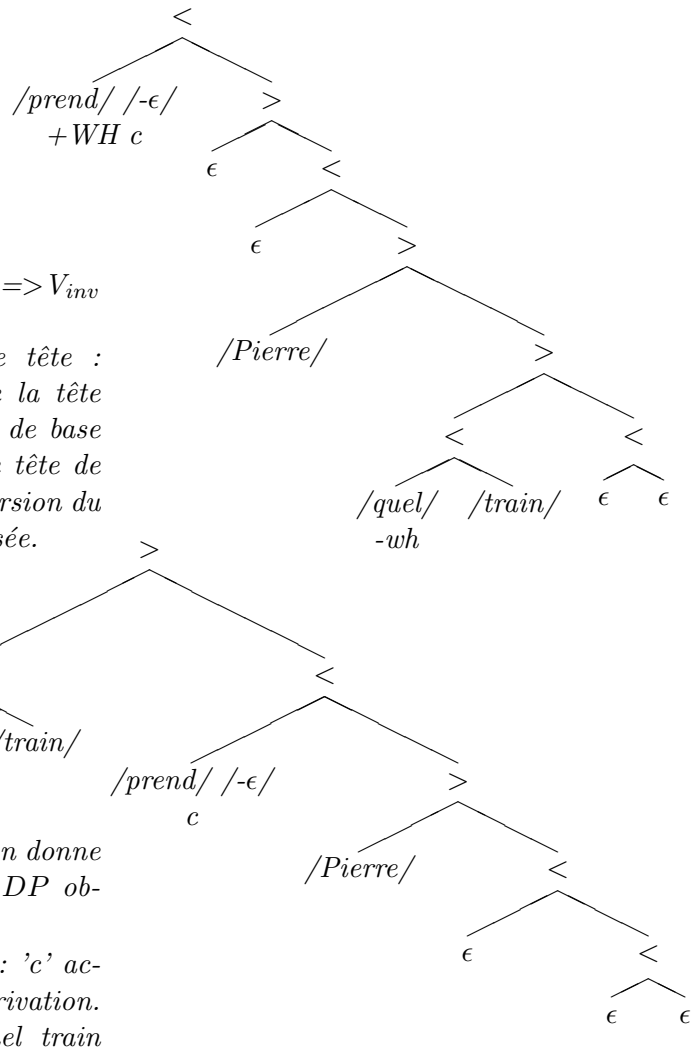


5. *Entrée lexicale "-ϵ" : =>v +case V_{inv}*
   Fusion de tête - montée de la forme phonologique de la tête :



Déplacement : du groupe nominal sujet : vérification du cas nominatif.





6. *Entrée lexicale "comp" : =>* V_{inv} *+WH c*
   Fusion : déplacement de tête : la forme phonologique de la tête de l'arbre portant le trait de base monte jusqu'à la nouvelle tête de l'arbre dérivé. Ainsi l'inversion du verbe et du sujet est réalisée.
   Déplacement : la dérivation donne une place de question au DP objet.
7. *Symbole de la dérivation : 'c' acceptation et fin de la dérivation. projection en FP : "Quel train prend Pierre ?"*

On assure que l'inversion du verbe et du sujet n'est demandée que pour les entrées de type *comp* portant un trait de résolution de question "+WH" ou "+wh". Dans ce cas, cette configuration sera refusée pour les questions portant sur le sujet de la phrase.

L'arbre d'analyse ainsi obtenu correspond à la structure syntaxique de la phrase. Ces deux exemples permettent de voir comment la version question d'un énoncé peut être mise en relation avec la forme affirmative. Les deux analyses divergent d'une transformation induite par le questionneur.

Chapitre 3

Grammaires

Sommaire

3.1 Grammaires et Hiérarchie de Chomsky 59
3.2 Les dérivations comme des files 62
3.3 Transcription des MGs vers les MCFGs 69
3.4 Arbre de fusion 72
3.5 Représentation abstraite d'un lexique 83
3.6 conclusion 86

Nous avons établi les concepts linguistiques théoriques utilisés dans le programme minimaliste, ainsi qu'une formalisation de ces concepts, nous souhaitons maintenant revenir sur certains résultats connus pour ces grammaires. Pour cela, nous définirons le cadre formel dans lequel elles se placent, nous commencerons par donner les définitions classiques permettant d'identifier les classes de langages reconnues par ces grammaires. Puis nous présenterons un résultat sur la capacité générative des GMs. Enfin, ces résultats nous ont conduit à proposer des relations d'équivalence entre GMs et, par extension, une modélisation abstraite des lexiques. Ces travaux sont basés sur la théorie des langages formels dont les pionniers furent M-P. Schützenberger et N. Chomsky, [SC63]. Les évolutions récentes ont été rassemblées dans [RS97], où l'on trouvera une description de l'analyse des langages hors-contexte, [SN97] ou encore celle des langages d'arbres, [GS97]. La vision formelle des grammaires minimalistes a été récemment développée dans [MMM00] et [MMK01].

3.1 Grammaires et Hiérarchie de Chomsky

Une des propositions qui découle de la formalisation des grammaires, du point de vue de la génération par Noam Chomsky, est de les regrouper en fonction de leurs propriétés. On peut ainsi différencier des types de grammaires et donner une définition de la classe de langage engendrée. La question posée aux langages théoriques est également étendue à la classe à laquelle appartiennent les langues naturelles. Les différentes propriétés issues des contraintes imposées aux grammaires permettent de proposer une hiérarchie entre elles, qui sert de référence pour tous les formalismes.

Une grammaire est définie par un ensemble N de non-terminaux (notés A, B, ...),

un ensemble T de terminaux représentant les unités que l'on veut reconnaître (notés a, b, ...), un ensemble de règles de réécriture de la forme $\alpha \rightarrow \beta$ où $\alpha, \beta \in (N \cup T)^*$ et enfin un axiome principal $S \in N$. La clôture transitive de la relation de dérivation est notée $\rightarrow^*$. Le langage engendré par la grammaire est l'ensemble des séquences $\omega \in T^*$ telles que $S \rightarrow^* \omega$. Pour $\iota \in (N \cup T)^*$, on note $|\iota|$ la somme du nombre d'éléments appartenant à N et du nombre d'élément appartenant à T.

Chomsky propose une classification en quatre classes distinctes appelées de type-0,1,2,3, avec :

$$\text{types-3} \subsetneq \text{type-2} \subsetneq \text{type-1} \subsetneq \text{type-0}$$

Chaque classe de grammaire est définie comme suit :

type-0 on ne pose pas de contrainte particulière à la rédaction des règles. Les langages sont dits récursivement énumérables. Ils sont reconnus par des machines de Turing. Le problème de l'analyse pour de tels langages est indécidable.

type-1 est l'ensemble des grammaires contextuels (Context Sensitive). Les règles sont de la forme $\alpha \rightarrow \beta$ où $|\alpha| \leq |\beta|$. On remarque aisément l'inclusion de ces langages dans les langages de type-0. Il y a une relation de contexte entre les éléments de α qui peuvent se réécrire en une nouvelle séquence. Une règle contextuelle pour une grammaire du français serait de la forme *si un verbe transitif est suivi d'un complément d'objet*, il peut être réécrit en *la séquence analogue augmentée de l'inflexion*.

type-2 est l'ensemble des grammaires algébriques ou hors-contextes (Context-Free Grammars - CFG). Elles sont particulièrement étudiées. Les règles $\alpha \rightarrow \beta$ sont caractérisées par $\alpha \in N$ et $\beta \in (N \cup T)^*$. Ces langages sont reconnus par des automates à pile non-déterministes. L'analyse de ces langages est polynomiale.

type-3 est l'ensemble des grammaires régulières. Ces grammaires peuvent être de deux formes :
- linéaire à gauche : les règles sont de la forme $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta \in T^*N$
- linéaire à droite : les règles sont de la forme $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta \in NT^*$

Les langages sont construits et reconnus par des automates d'états finis. L'analyse de ces langages est polynomiale.

L'inclusion de ces classes de langages les unes dans les autres permet de les représenter sous forme de hiérarchie, appelée hiérarchie de Chomsky et présentée dans la figure 15.

Les études sur les capacités génératives ont permis d'identifier des types de langages clés permettant de classifier un langage engendré par un formalisme donné. Les constructions non-régulières suivantes peuvent être exprimées par des CFG :
- $a^n b^n, n \in \mathbb{N}$
- $\omega\tilde{\omega}, \omega \in T^*$ où $\tilde{\omega}$ la copie inverse de ω.

D'après le Lemme de pompage [JJ79], les langages suivants ne sont pas reconnaissables à partir des CFG et appartiennent aux grammaires contextuelles :
- multiple agreement : $a^n b^n c^n, n \in \mathbb{N}$,
- crossed agreement : $a^n b^m c^n d^m, n, m \in \mathbb{N}$
- duplication : ww.

Intuitivement, ils correspondent à des dépendances devant être mémorisées au cours de la dérivation. Cependant, [KB82a] montre que les structures $a^n b^n c^n$ sont présentes en Néerlandais. La structure sous-jacente aux langues naturelles, hypothèse des théorie de

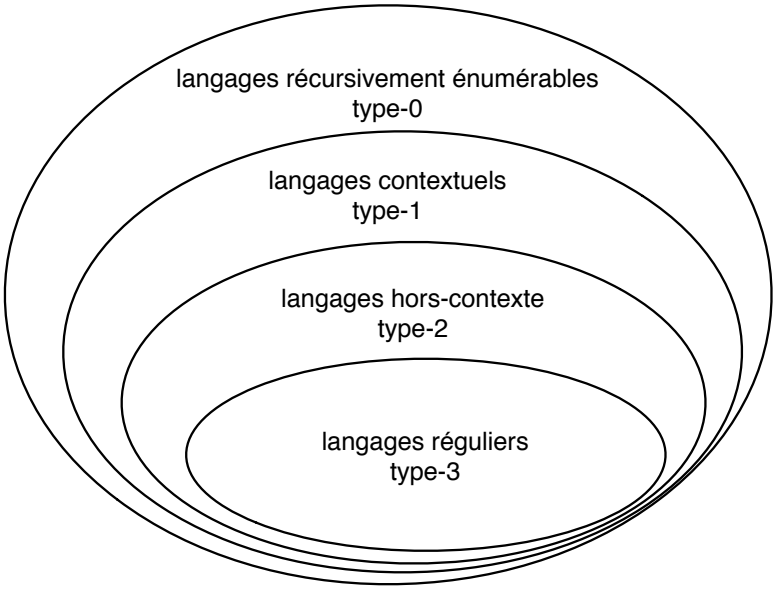


Fig. 15 – Hiérarchie de Chomsky.

Chomsky, la classe de langages à laquelle appartiennent les langues naturelles est au delà des CFGs.

De plus, on généralise les trois langages ci-dessus qui ne sont pas hors-contextes, pour tout entier naturel k : k-multiple agreement : $a_1^n \cdots a_k^n$, k-crossed agreement : $a_1^{n_1} \cdots a_k^{n_k} b_1^{n_1} \cdots b_k^{n_k} \cdots$, k-duplication : w^k.

Joshi [Jos85] et Schieber [Sch85] montrent que les constructions non généralisées peuvent apparaître dans les langues naturelles. Cependant la classe des langages contextuels est trop importante pour en rendre compte. L'analyse pour cette classe n'est pas nécessairement polynomiale. À la vue du temps de réaction de notre cerveau à recevoir et comprendre des énoncés en langue naturelle, on suppose que l'analyse est au pire polynomiale. Joshi [Jos85] définit une nouvelle classe intermédiaire entre les langages hors-contextes et contextuels. Cette classe représentant la classe des langues naturelles est dite faiblement sensible au contexte - (Midly Context Sensitive - MCS).

Un langage L est MCS s'il vérifie les propriétés suivantes :

1. si L est hors-contexte alors L appartient à la classe des langages MCS,
2. L peut être analysé en temps polynomial selon la taille de l'entrée de l'analyse,
3. La classe MCS peut exprimer les trois langages suivants : pour T un alphabet, $a, b, c, d \in T$, $\omega \in T^*$ et $n, m \in \mathbb{N}$:
   (a) $a^n b^n c^n$,
   (b) $a^n b^m c^n d^m$,
   (c) $(\omega\omega)$,

4. L a la propriété de croissance constante.

Cette dernière propriété exprime l'idée que les relations entre éléments dépendent des éléments eux-mêmes et non de leur nombre. Elle est souvent vérifiée par la preuve du lemme de la pompe. Par exemple le langage a^{2^n}, $n \in \mathbb{N}$ n'a pas cette propriété. Jens Michaelis a montré que ce langage est engendré par les GMs sous certaines conditions. Nous reviendrons sur ce résultat dans la section suivante.

Cependant, si cette définition de la classe des langages MCSs semble faire un large consensus dans la communauté, de récentes études menées par G. Kobele montrent que des phénomènes de copies avec dépendances apparaissent dans certaines langues africaines. La question reste de savoir si c'est la réalisation d'un phénomène linguistique n'utilisant pas la propriété de croissance constante ou si c'est le résultat d'une copie réutilisée dans d'autres positions de l'énoncé.

Les langages faiblement sensibles au contexte sont donc un sous-ensemble des langages contextuels. Une version de la hiérarchie de Chomsky avec ces langages est présentée dans la figure 16. À présent que la classe de langages des langues naturelles est énoncée, revenons sur certaines propriétés des GMs.

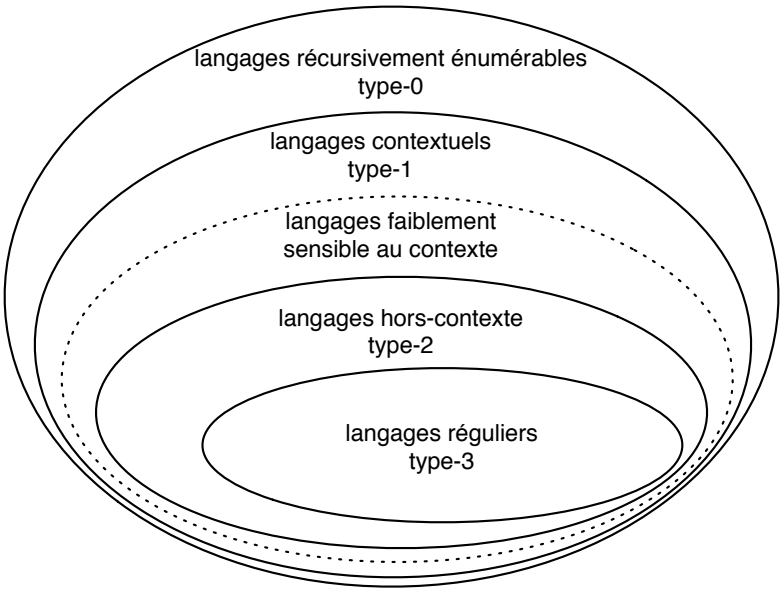


FIG. 16 – Hiérarchie de Chomsky et classe des langues naturelles.

3.2 Les dérivations comme des files

Dans [MK05], une version de l'analyse de a^{2^n} est présentée par J. Michaelis. Nous ferons à nouveau la preuve de l'appartenance à la classe des langages minimalistes de ce langage car les mécanismes mis en œuvre dans ce type de dérivation sont utilisés dans le chapitre suivant. Comme le langage $\{a^{2^n} | n \in \mathbb{N}\}$ est de type-0, cela implique que la capacité générative des GMs sous certaines conditions est de type-0, donc supérieure à la classe des langages MCS. De plus, les automates à file ont l'expressivité des machines de Turing.

Mais la remarque précédente sur les phénomènes de copie avec dépendances semble montrer que ce n'est pas un échec pour ce formalisme. Au contraire, la présence ou non de conditions serait alors une réalisation de la notion de paramètre.

Présentation générale

Pour parvenir à ce résultat, l'arbre d'analyse doit être envisagé comme une file où les différentes parties sont séparées par des marqueurs spéciaux. Le premier élément de la file est alors celui qui est "le plus bas" dans la dérivation, et les nouveaux éléments sont introduits "en haut" de la dérivation. L'opération *move* nous permet de déplacer un élément en première position de la file pour le mettre en dernière position s'il doit être conservé, ou éliminé sinon.

Cette utilisation des dérivations comme des files implique de pouvoir faire passer le premier élément de la file en dernier élément de la file, autant de fois que nécessaire. Nous avons deux possibilités : soit tous les déplacements sont prévus dans le lexique, mais il faudra autant de lexiques que de mots à reconnaître, soit on utilise la notion de *copie*. Cette copie est réalisée par les assignés vus comme des marqueurs. Ainsi le passage d'un élément de la première à la dernière position se fait à l'aide d'une entrée portant à la fois l'assignateur correspondant (effacement) et un autre assigné correspondant (copie). Cependant la conséquence directe est qu'aucune forme phonologique ne peut être utilisée car celles-ci ne peuvent pas être effacées. Dans ce cas, la SMC doit être relaxée. Les opérations autorisées sont alors la fusion et le déplacement.

Voici le lexique proposé pour dériver de tels mots, engeandrant la GM $G_{puis} = \{a^{2^n} | n \in \mathbb{N}\}$.

Lexique 25. *Dans ce lexique, noté Lex, les lettres du mot sont "marquées" par l'assigné '−l' et la file par l'assigné '−m'. Lorsque la forme phonologique est est ϵ, nous ne la marquons pas.*

	phase d'initialisation		
type 1 :	*w -m*	*type* 2 :	*=w x -l*
	phase d'itération		
type 3 :	*= x +M y -m*	*type* 4 :	*=y +L z -l*
type 5 :	*=z y -l*	*type* 6 :	*=z x -l*
	phase de conclusion		
type 7 :	*=x +M c*	*type* 8 :	*=c +L c /a/*

À la fin de ce chapitre nous proposerons une représentation abstraite des lexiques. À titre indicatif, nous donnons cette représentation pour ce lexique dans la figure 17.

Synopsis de Dérivation 26.
Les dérivations se décomposent en trois phases :

1. ***initialisation :*** *cette phase est déterministe. Lors de celle-ci, la dérivation accumule le marqueur de fin d'itération $'-m'$ et un marqueur de lettre $'-l'$. Un passage à la phase de conclusion permet de reconnaître le mot pour $n = 0$, a^{2^0} ; c'est-à-dire /a/.*
2. ***itération :*** *Lors d'une itération, la file est réinitialisée par un déplacement/copie du marqueur de file $'-m'$ (entrée de type 3). Ensuite, chaque marqueur du symbole*

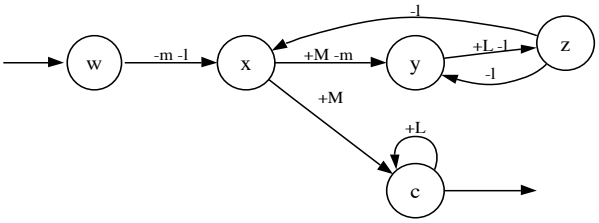


FIG. 17 – Représentation du lexique pour le langage a^{2^n}.

terminal est déplacé/copié (entrée de type 4) puis dupliqué (soit par une entrée de type 5 si on est dans la phase d'itération, soit par une entrée de type 6 si on passe à la fin de l'itération). Ainsi on passera de a^{2^n} *à* $a^{2^n * 2} = a^{2^{n+1}}$ *marqueurs du terminal.*

3. ***conclusion :*** *elle consiste en l'effacement du marqueur de pile* $'-m'$ *et le remplacement de chaque marqueur de terminal* $'-l'$ *par une forme phonologique* $/a/$.

Pour différencier ces trois étapes, on notera ϕ_{init} (*resp.* ϕ_{iter} et ϕ_{conc}) la fonction qui, pour le lexique, renvoie l'ensemble des entrées considérées dans la *phase d'initialisation* (*resp.* itération et conclusion). On appelle phase d'initialisation l'utilisation des deux entrées de ϕ_{init}. On appelle phase d'itération la partie d'une dérivation commençant par l'utilisation de l'entrée de type 3 et se terminant par l'utilisation de l'entrée de type 6, appartenant toutes deux à ϕ_{iter}. On appelle phase de conclusion la partie d'une dérivation n'utilisant que des entrées de ϕ_{conc}. Nous commencerons par présenter un exemple de dérivation. Pour des commodités de lecture, cet exemple utilise une représentation aplanie des arbres minimailstes qui sera présentée et largement utilisée dans le chapitre 4.

Exemple de dérivation a^{2^2}

Dérivation 27.

Une dérivation commence par une fusion entre les deux premières entrées lexicales.

1. *entrée type 1 :* $w\ -m$

   entrée type 2 : $=w\ x\ -l$

   fusion : $\underbrace{\mathbf{x}\ \ -\mathbf{l},\ -m}$

Nous pouvons maintenant soit passer dans la phase de conclusion et obtenir $a^{2^0} = a$ *(présence d'un* $'-l')$ *soit itérer :*

2. *entrée type 3 : copie/déplacement du marqueur m* $= x\ +M\ y\ -m$

   fusion : $\underbrace{+\mathbf{M}\ \ \mathbf{y}\ \ -\mathbf{m},\ \ -l, -m}$

   déplacement : on efface l'ancien marqueur m $\underbrace{\epsilon,\ \mathbf{y}\ \ -\mathbf{m},\ -l}$

3. *entrée de type 4 :* $=y\ +L\ z\ -l$

   fusion : introduction d'un nouveau marqueur l.

$\underline{\mathbf{+L\ z}\ -\mathbf{l},\ -m,\ -l}$

déplacement : effacement de l'ancien marqueur. $\underline{\epsilon,\ \mathbf{z}\ -\mathbf{l},\ -m}$

La phase d'itération se poursuit par une duplication de l'assigné que l'on vient d'introduire.

4. *entrée type 6 :* $=z\ x\ -l$

   fusion : $\underline{\mathbf{x}\ -\mathbf{l},\ -l,\ -m}$

Fin de la phase d'itération. Si nous passons à la phase de conclusion on obtient $a^{2^1} = a^2$ *(présence de deux* $-l$*). Nous recommençons une autre itération pour passer à* a^{2^2}*.*

5. *entrée de type 3 :* $=x\ +M\ y\ -m$

   fusion : $\underline{\mathbf{+M\ y}\ -\mathbf{m},\ -l,\ -l,\ -m}$

   déplacement : effacement du marqueur de pile. $\underline{\mathbf{y}\ -\mathbf{m},\ -l,\ -l}$

6. *entrée de type 4 :* $=y\ +L\ z\ -l$

   fusion : introduction d'un nouveau marqueur de terminal.

   $\underline{\mathbf{+L\ z}\ -\mathbf{l},\ -m,\ -l,\ -l}$

   déplacement : effacement d'un marqueur de terminal. $\underline{\mathbf{z}\ -\mathbf{l},\ -m,\ -l}$

7. *entrée lexicale de type 5 :* $=z\ y\ -l$

   fusion : duplication du marqueur que l'on vient d'introduire. $\underline{\mathbf{y}\ -\mathbf{l},\ -l,\ -m,\ -l}$

8. *entrée lexicale de type 4 :* $=y\ +L\ z\ -l$

   fusion : $\underline{\mathbf{+L\ z}\ -\mathbf{l},\ -l,\ -l,\ -m,\ -l}$

   déplacement : effacement du marqueur. $\underline{\mathbf{z}\ -\mathbf{l},\ -l,\ -l,\ -m}$

Puis duplication de l'assigné.

9. *entrée type 6 :* $=z\ x\ -l$

   fusion : duplication. $\underline{\mathbf{x}\ -\mathbf{l},\ -l,\ -l,\ -l,\ -m}$

Le nombre de $-l$ *est bien égal à* a^{2^2}*. Passage à la phase de conclusion.*

10. *entrée type 7 :* $=x\ +M\ c$

    fusion : $\underline{\mathbf{+M\ c},\ -l,\ -l,\ -l,\ -l,\ -m}$

    déplacement : effacement définitif du marqueur de file. $\underline{\mathbf{c},\ -l,\ -l,\ -l,\ -l}$

À présent, l'utilisation de l'entrée lexicale de type 8 permet d'effacer chaque marqueur $-l$ *et de le remplacer par un /a/. En itérant quatre fois une fusion/déplacement :*

13. *résultat des trois fusions/déplacements précédents :* $\underline{\mathbf{c\ /a/},\ /a/,\ /a/,\ -l}$

14. *entrée lexicale de type 8 :* $=c\ +L\ c\ /a/$

fusion : $\underbrace{+L \;\; c \;\; /a/, \;\; /a/, \;\; /a/, \;\; /a/, \;\; -l}$

déplacement : $\underbrace{c \;\; /a/, \;\; /a/, \;\; /a/, \;\; /a/}$

Théorème 28. *Le langage* $\{a^{2^n}, \forall n \in \mathbb{N}\}$ *est un langage minimaliste.*

Pour prouver ce théorème, on utilise une propriété et deux lemmes.

Propriété 29. *Pour* $i \in \mathbb{N}$*, Ssoit* P_i *la propriété pour* $t \in T_{MG}$ *définie par :*

1. $head(t) = \langle x \; -l \; \rangle$
2. *t contient* $2^i - 1$ *feuilles* $\langle -l \rangle$,
3. *t contient une unique feuille* $\langle -m \rangle$
4. *toutes les autres feuilles ne portent aucun trait syntaxique.*

Lemme 30. *Si* t_i *l'arbre produit après i itérations vérifie* P_i *(propriété 29), alors* t_{i+1} *l'arbre obtenu après* $i + 1$ *itérations vérifie* P_{i+1}.

Démonstration. Soit t_i qui vérifie P_i. On a donc $head(t_i) = \langle x \;\; -l \rangle$. Dans ϕ_{iter}, seule l'entrée de type 3 commence par un sélecteur $=x$. Le passage dans une phase d'itération se fait par une fusion qui efface le x de la tête de t_i. On note t'_i le résultat de cette fusion. t'_i contient alors 2^i feuilles $\langle -l \rangle$. La dérivation se poursuit par un déplacement du marqueur de file, donc l'effacement du $-m$ de t_i. La dérivation ne contient plus alors qu'une unique feuille portant le trait $-m$.

On a donc $head(t'_i) = \langle y \; -m \rangle$. Seule l'entrée $\langle =y \; +L \; z \; -l \rangle$ qui appartient à ϕ_{iter} commence par un sélecteur $=y$. On fusionne donc la dérivation avec cette entrée. La dérivation efface par un move le $-l$ le plus bas et construit t''_i. $head(t''_i) = \langle z \; -l \rangle$. La suite n'est pas déterministe car ϕ_{iter} possède deux entrées commençant par un sélecteur $=z$.

On suppose que la dérivation se poursuit par une fusion avec l'entrée de type 5. Dans ce cas, le premier trait de la tête est à nouveau y. Ce processus a donc introduit, à partir d'un $-l$ dans la file, deux nouveaux $-l$.

Si on utilise ce processus pour les 2^{i-1} feuilles $\langle -l \rangle$, on a alors $2 \times 2^{i-1}$ nouvelles feuilles $\langle -l \rangle$. De plus, il reste une feuille $\langle -l \rangle$ dans la file et la tête de la dérivation est $\langle y \; -l \rangle$.

La dérivation se poursuit avec une fusion/déplacement effaçant l'un des deux $-l$ puis une fusion avec une entrée de fin de l'itération (type 6). On a alors $2 \times (2^i - 1) + 1$ nouvelles feuilles $\langle -l \rangle$ et $head(t_{i+1}) = \langle x - \; l \rangle$. Cette procédure n'ajoute pas d'autres feuilles portant des traits syntaxiques.

Enfin, si P_i ne contient aucun autre trait syntaxique, t_{i+1} n'en contient pas non plus. On a donc t_{i+1} qui vérifie P_{i+1}.

La propriété est donc vraie s'il y a exactement 2^i copies/duplications pour chaque $-l$.

Revenons sur l'hypothèse faite précédemment : utilisation d'une entrée de type 6 au lieu de type 5 dans le processus (*i.e.* pas assez de copies/duplications avant de sortir de la phase d'itération). Nous représentons les différentes étapes de ces dérivations dans la figure 18. La dérivation commence par l'arbre (a). Le résultat de la fusion avec une entrée de type 6 est l'arbre (b) pour lequel le premier trait de la tête est alors x (au lieu d'un y). Deux entrées du lexique peuvent être utilisées : soit $\langle =x \; +M \; y \; -m \rangle$ qui appartient à ϕ_{iter}, soit $\langle =x \; +M \; c \rangle$ qui appartient à ϕ_{conc}. Quel que soit le choix fait, le trait suivant

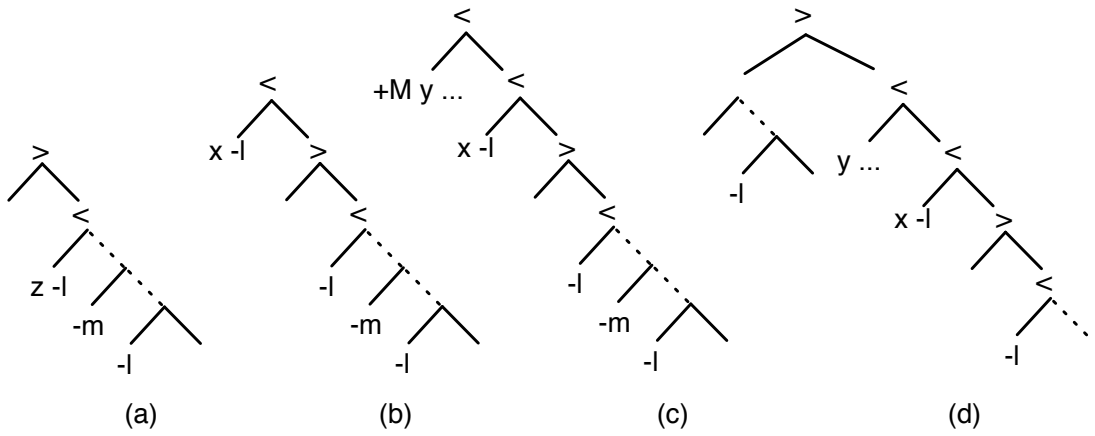


Fig. 18 – Premier cas échouant lors de la dérivation de a^{2^n}.

implique un déplacement du marqueur de la file (arbre (c)). Or la feuille qui porte le $-m$ se projette sur toutes les feuilles $\langle -l \rangle$ non encore traitées. Dans ce cas elles sont déplacées et passent en position de spécifieur de la nouvelle tête (arbre (d)). La dérivation ne pouvant être acceptée que si elle ne contient plus de trait, ils devront nécessairement être effacés dans la suite de la dérivation. Leur position de spécifieur ne permettra pas d'y parvenir. Donc la dérivation échouera.

L'autre possibilité d'erreur dans ce type de dérivation provient de la situation symétrique, lorsque l'entrée de type 6 doit être utilisée pour sortir de la phase d'itération. Dans ce cas, la dérivation utilise à la place une entrée de type 5. La figure 19 reprend les étapes de ce type de dérivation. On commence avec l'arbre (a). Le résultat de la fusion avec une entrée de type 5 est présenté par l'arbre (b) ayant comme tête $\langle y\ -l \rangle$. La seule entrée du lexique commençant par un sélecteur $=y$ est l'entrée de type 4. L'arbre (c) est le résultat de la fusion avec cette dernière. La tête implique alors un déplacement d'une feuille $-l$ qui entraîne avec elle le marqueur $-m$ en position de spécifieur. Il y a donc trop de copies/duplications. Lorsque la dérivation sortira de cette phase d'itération en effaçant le marqueur $-m$, elle ne pourra plus y parvenir car ce dernier a alors pris une position de spécifeur. La dérivation ne pourra pas se terminer.

□

Lemme 31. *Si t_i vérifie P_i, alors la phase de conclusion accepte exactement le mot a^{2^i}.*

Démonstration. Si $head(t_i) = \langle x\ -l\ \rangle$, t_i est fusionnée avec une entrée de type 7 pour donner t'_i. On a alors $head(t'_i) = \langle +Mc\ \rangle$ et $head(t_i) = \langle -l\ \rangle$. Donc t'_i contient 2^i feuilles $\langle -l \rangle$. La tête portant un assignateur $+M$ et t_i contenant une unique feuille $\langle -m \rangle$, un déplacement est réalisé. On a donc t''_i qui contient 2^i feuilles $\langle -l \rangle$, $head(t''_i) = \langle c \rangle$ et toutes les autres feuilles ne portent aucun trait.

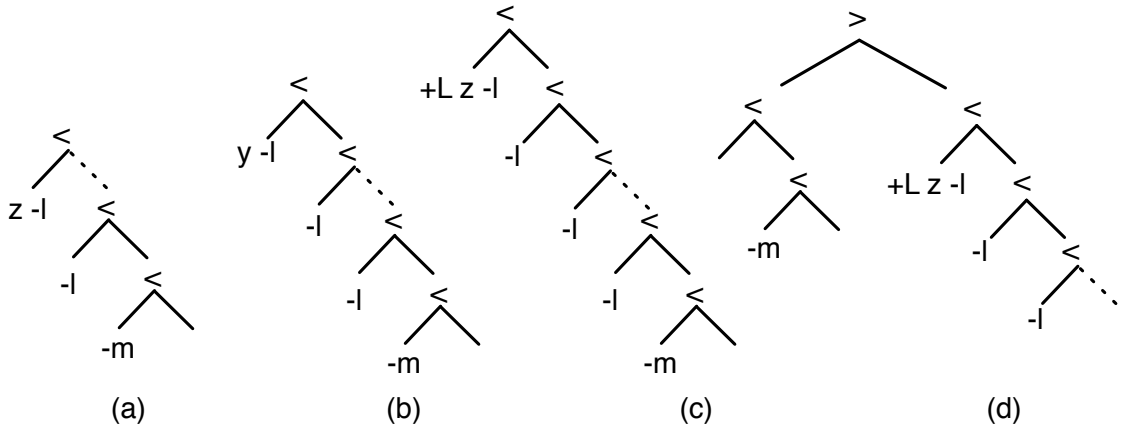


FIG. 19 – Deuxième cas échouant lors de la dérivation de a^{2^n}.

La suite n'est pas déterministe, mais ne permet que l'utilisation de l'unique entrée lexicale (type 8). Chaque utilisation nécessite une fusion effaçant le trait c sur la tête, puis un déplacement effaçant l'un des $-l$.

Cette entrée introduit également une forme phonologique $/a/$. Le trait en première position sur la tête est à nouveau c. La dérivation ne pouvant être acceptée que lorsqu'elle ne contient plus que le trait c, cette procédure doit être utilisée 2^i fois pour chacun des 2^i $\langle -l \rangle$. Comme chaque passage introduit un a, lorsque ces 2^i fusions/déplacements sont réalisés, l'analyse reconnaît a^{2^i}. □

On remarquera que chaque fusion/déplacement doit nécessairement avoir lieu sur la feuille la plus basse dans l'arbre (en première position de la file) sinon cette dernière sera déplacée en position de spécifieur, ne pouvant plus être effacée.

Démonstration du théorème 28. La phase d'initialisation étant déterministe, on obtient nécessairement $t_0 =< (x\ -l, -m)$ qui vérifie P_0 la propriété 29.

On utilise alors une induction sur $t_i \in T_{MG}$.

- t_0 vérifie P_0, d'après le lemme 31, la phase de conclusion accepte le mot a.
- Par hypothèse d'induction, le passage par i itérations permet de reconnaître le mot a^{2^i}. À partir de la phase d'initialisation, puis des i premières itérations, on obtient t_i. D'après le lemme 30, on obtient t_{i+1} qui vérifie P_{i+1}. D'après le lemme 31, la phase de conclusion permet alors de reconnaître $a^{2^{i+1}}$.

$\forall n \in \mathbb{N}$, G_{puis} reconnaît exactement a^{2^n}. □

On peut aisément utiliser la notion de file et de compteur en même temps. L'annexe C présente un lexique et une dérivation permettant de reconnaître un langage où les compteurs sont enchassés : $a^n b^{2^n}$. Les langages de compteurs sont exposés dans le chapitre 4.

La particularité des grammaires sur les compteurs est qu'elles relaxent la SMC. Michaelis a donc montré que si on conserve la SPIC, ces grammaires étaient de type-0, alors que dans l'autre sens - −SPIC et +SMC - les grammaires étaient équivalentes aux LCFRS qui sont MCS.

Ces résultats sont présentés dans le diagramme de la figure 20.

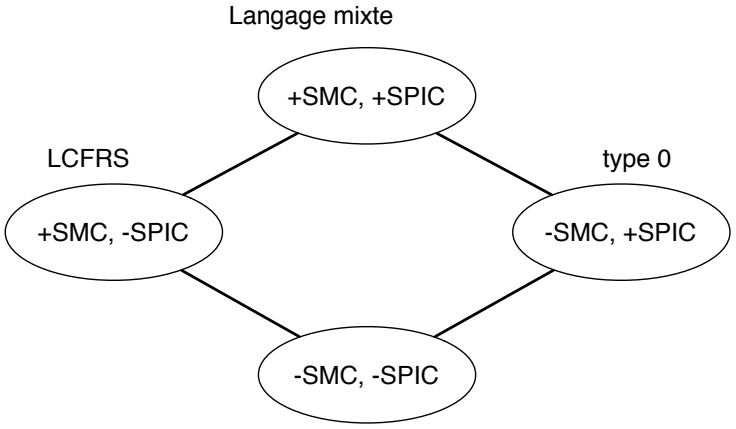


Fig. 20 – Capacité générative des GMs.

On notera que la capacité générative des GMs augmente en faisant varier les restrictions posées sur ces grammaires.

3.3 Transcription des MGs vers les MCFGs

La transcription des GMs en Multiple Context Free Grammars - MCFGs - est une partie importante pour la mise en œuvre d'algorithmes d'analyse plus efficaces mais également pour la compréhension du fonctionnement interne du formalisme. Les GMs sont composées de deux parties importantes : l'une permettant de connecter des éléments ensemble et pouvant se traduire par une CFG (comme nous le montrerons dans la suite de ce chapitre) et l'autre reconstruisant la structure de l'analyse. Les CFGs sont inaptes à rendre compte de l'opération *move*.

Pour cela, il faut utiliser les MCFGs. Cette transcription a été détaillée dans [MMM00] comme étape de la transcription des MGs vers les RTGs - Regular Tree Grammars - et dans [Gui04]. Un rapport détaillé avec amélioration a été proposé dans [Coh06] avec un exemple basé sur les lexiques de compteurs présentés dans le chapitre suivant. Nous commencerons par définir les MCFGs, puis nous exhiberons la transformation. Ces travaux sont basés sur la définition des GMs sous forme de chaînes (évoquées dans le chapitre précédent et présentées dans l'annexe A) pour laquelle seules les définitions des règles de composition changent. Les concepts et les définitions des autres éléments de la grammaire sont les mêmes.

Définition 32. *Une k-MCFG G est définie par G = (N, O, F, P, S) où*

– *N est un ensemble fini de symboles non-terminaux* $(A_0, A_1, \ldots)$,

- $O = \bigcup_{i=1}^{m} (T^*)^i$, *où* T *est un ensemble de symboles terminaux, disjoint de* N. O *est l'ensemble des tuples de chaînes de symboles de* T, *d'arité au plus* m,
- F *est un ensemble de fonctions partielles* f *de* $O^q \rightarrow O$ *(pour un entier* q *donné) ; on pose* $a(f)$ *le nombre d'argument de* $f \in F$ *ainsi que* $r(f)$ *et* $d_i(f)$ *pour* $1 \leqslant i \leqslant a(f)$ *les entiers tels que* $f : (T^*)^{d_1(f)} x (T^*)^{d_2(f)} x \dots x (T^*)^{d_{a(f)}(f)} \rightarrow (T^*)^{r(f)}$.
  Ainsi, chaque argument de f *est un tuple d'arité connue.*
- P *est l'ensemble des règles de la forme :*
  $$A_0 \rightarrow f[A_1, \dots A_q]$$
  avec $A, A_1, \cdots, A_n \in N$ *et* $f : (T^*)^{d(A_1)} x \cdots x (T^*)^{d(A_n)} \rightarrow (T^*)^{d(A)}$ *; les symboles non-terminaux ont donc une arité égale à celle des arguments des fonctions qui peuvent être appliquées,*
- $S \in N$ *le symbole initial.*

Si $q = 0$, la règle est terminale ($A \rightarrow f[]$ équivaut à $A \rightarrow \alpha$).

Le langage de G pour un non-terminal donné A, $L_G(A)$, est défini comme le plus petit ensemble pour lequel :

- Si $A \rightarrow \theta \in P$ (cas d'une règle terminale), alors $\theta \in L_G(A)$.
- Si on a :
  - $\theta_i \in L_G(A_i)$ pour $1 \leq i \leq q$,
  - $A \rightarrow f[A_1, \dots, A_q]$ et
  - $f[\theta_1, \dots, \theta_q]$ est défini,

  alors $f[\theta_1, \dots, \theta_q] \in L_G(A)$.

On définit alors $L_G = L_G(S)$ le langage produit par G.

Une k-MCFG impose une arité à chacun des non-terminaux - dont la valeur maximale est k. Par exemple, pour la règle suivante, où l'on nomme g la fonction sous-jacente :

$$A \rightarrow BC[0,1;1,0][0,0][1,1]$$

$g[(a,b),(c,d)]$ produit (bc,a,d). Chaque variable est donc accessible par sa position relative dans les tuples.

La conversion des MGs vers les MCFGs se fait à partir de la version des GMs pour des chaînes. On rappelle qu'on ajoute à la suite de la chaîne un symbole spécifiant :: pour une entrée lexicale, : une expression complexe et . l'une ou l'autre de ces constructions. Par exemple pour un type de fusion :

$$\frac{s. = f\gamma, \alpha_1, \cdots, \alpha_k \quad t.f\delta, \beta_1, \cdots, \beta_l}{s : \gamma, \alpha_1, \cdots, \alpha_k, t : \delta, \beta_1, \cdots, \beta_l}$$

$\sigma(\gamma, \alpha_1, \cdots, \alpha_k, \delta, \beta_1, \cdots, \beta_l) \rightarrow$
$\sigma(= f\gamma, \alpha_1, \cdots, \alpha_k)\sigma(f\delta, \beta_1, \cdots, \beta_l)[0,0] \cdots [0,k][1,0] \cdots [1,l]$

Dans cette règle, les arguments sont concaténés en fonction de leur type. Voici une transformation de règle sur un type de déplacement :

$$\frac{s : +f\gamma, \alpha_1, \cdots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \cdots, \alpha_k}{ts : \gamma, \alpha_1, \cdots, \alpha_{i-1}, \alpha_{i+1}, \cdots, \alpha_k}$$

$\sigma(\gamma, \alpha_1, \cdots, \alpha_{i-1}, \alpha i + 1, \cdots, \alpha_k) \rightarrow$
$\sigma(+f\gamma, \alpha_1, \cdots, \alpha_{i-1}, -f, \alpha_{i+1}, \cdots, \alpha_k)[0,i;0,0][0,1] \cdots [0,i-1][0,i+1] \cdots [0,k]$

On voit clairement la réalisation du "déplacement" dans la modification de l'ordre des arguments.

Exemple 33. *Conversion de la grammaire pour un mot de deux terminaux sur un compteur (présentée dans le chapitre 4).*

Lexique MG :

$0 \quad D\ -D\ /b/ \quad 1 \quad =V\ +D\ D\ -D\ /b/ \quad 2 \quad =D\ V\ -V\ /a/$
$4 \quad c/\epsilon/ \quad 3 \quad =D\ +V\ V\ -V\ /a/ \quad 5 \quad =V\ +D\ +V\ C\ /\epsilon/$

Initialisation

On ajoute à la MCFG :

$t_0 \to b$
$t_1 \to b$
$t_2 \to a$
$t_3 \to a$ *ainsi que* $S \to t_4[0,0]$
$t_4 \to \epsilon$
$t_5 \to \epsilon$

Puis on cherche toutes les combinaisons possibles :

– *fusion 0 et 3 :*

$$t_6 \to t_3 t_0[0,0][1,0]$$

les arguments sont alors [−D][+V V −V]. La dérivation échoue puisqu'elle n'est pas en mesure de réaliser le déplacement.

– *fusion 0 et 2 :*

$$t_7 \to t_2 t_0[0,0][1,0]$$

les arguments correspondent à [V −V] et [−D]

– *puis fusion avec 5 - fin dela dérivation pour "ab"*

$$t_8 \to t_5 t_7[0,0][1,0][1,1]$$

les arguments correspondent à [+D +V C], [−V] et [−D]

– *fusion entre t_7 et 1- entrée dans la phase d'itération*

$$t_9 \to t_1 t_7[0,0][1,0][1,1]$$

les arguments correspondent à [+D D −D], [−V] et [−D]

– *déplacement sur t_9*

$$t_{10} \to t_9[0,2;0,0][0,1]$$

les arguments correspondent à [D −D], et [−V]

– *déplacement sur t_8*

$$t_{11} \to t_8[0,2;0,0][0,1]$$

les arguments correspondent à [+V C], et [−V]

– *déplacement sur t_{11}*

$$t_{12} \to t_{11}[0,1;0,0]$$

l'argument correspond à [C]

On termine donc une dérivation après cette étape

$$S \to t_{12}[0,0]$$

– *fusion entre 3 et t_{10} - fin de la phase d'itération*

$$t_1 3 \to t_3 t_1 0[0,0][1,0][1,1]$$

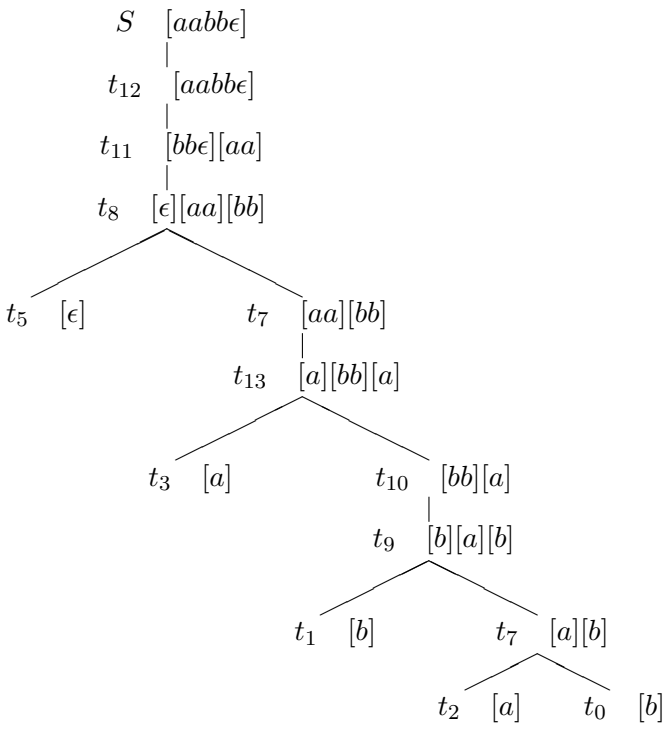
les arguments correspondent à [+V V −V], [-D] et [−V]

– *déplacement sur t_{13} retour à la fin de la phase d'itération ie t_7*

$$t_7 \to t_{13}[0,2;0,0][0,1]$$

les arguments correspondent à [V −V] et [−D]

On représente ce calcul sous forme d'arbre représentant les étapes de la dérivation :



L'analyse que l'on obtient ne correspond pas exactement à l'arbre de la GM. Cependant il est isomorphe à celui obtenu si on utilise des GMs sur les chaînes et non plus les arbres. Si on applique un transducteur à une conversion de cette MCFG en une RTG, on peut recalculer l'arbre d'analyse de la GM de départ. Il reste cependant évident que la taille de la MCFG correspondant à la GM est beaucoup plus importante. Ce que l'on gagne en temps d'analyse, on le perd en espace. Le but de cette transformation est d'utiliser des algorithmes d'analyse plus efficaces. Cependant, on identifie ici un problème plus conceptuel apporté par cette traduction : la démarche qui consiste à définir toutes les règles de réécriture dans la MCFG se rapproche davantage de la perspective basée sur la description des structures de phrases que de la vision généralisée proposée par Chomsky, comme il en a été question dans la section 1.2.1.

Nous arrêtons ici la présentation de l'état de l'art pour les GMs et revenons sur des travaux personnels à propos de la notion de grammaire pour les GMs.

3.4 Arbre de fusion

La plupart des formalismes grammaticaux se servent de règles de constructions ou de descriptions souvent homogènes pour décrire leur système (grammaire de réécriture, CFG, grammaires AB). Les grammaires minimalistes ont, elles, deux opérations de types très différents.

La *fusion* est une opération qui, à partir de deux arbres en construit un troisième.

$$fusion : T_{MG} \times T_{MG} \longrightarrow T_{MG} :$$

Les *déplacements* quant à eux prennent une dérivation et la réordonnent :

$$\textit{déplacement} : T_{MG} \longrightarrow T_{MG}.$$

Au cours d'une analyse, ces deux opérations ont des influences différentes. La fusion permet d'assembler les éléments de l'analyse vus comme des arbres, alors que le déplacement fait varier les relations qui existent entre les parties de ces arbres. La suite des fusions est donc primordiale car elle construit la dérivation (mais non suffisante pour aboutir aux analyses acceptantes).

Dans les premières versions de la théorie du minimalisme, Chomsky proposait de faire toutes les fusions puis la suite des déplacements. Cette idée avait l'avantage de construire l'arbre d'analyse en deux temps, d'abord l'arbre des fusions, puis l'arbre d'analyse à proprement parler. Cette procédure ne s'avère pas performante tant les parties de l'analyse sont imbriquées les unes dans les autres. De plus, cette façon d'envisager les analyses se posait en contradiction directe de la SMC : *la présence simultanée de constituants de type similaire entraîne celle de traits de déplacement identiques.* Il faut donc complètement traiter l'un d'eux avant d'introduire le second.

Nous proposons de faire une analyse des productions à partir de l'une d'elles. Nous allons donc isoler dans une dérivation la suite de fusions utilisées, et construire à partir de celle-ci une structure appelée *arbre de fusion.* Nous montrerons que ces arbres sont équivalents à des CFGs. Puis en nous inspirant de ces travaux, nous proposerons une modélisation abstraite des lexiques des GMs qui se révèle être un outil important lors de leur phase de rédaction.

3.4.1 Arbres de fusion

La structure d'arbre minimaliste de cette opération et l'enchaînement de plusieurs de ces opérations permettent de construire un arbre minimaliste dont les feuilles sont des entrées lexicales.

Définition 34. *On définit l'ensemble des* ***arbres de fusion****, noté* T_{MG_f} *comme des arbres minimalistes dont les nœuds ne sont pas étiquetés. Soit* Σ_{MG_f} *l'alphabet gradué* $\langle \{.\}, rang_{MG} \rangle$ *tel que :* $rang_{MG_f}(.) = 2$*, servant à définir* T_{MG_f}*.*

La fonction qui permet d'associer un arbre de fusion à un arbre obtenu par une GM est défini comme suit.

Définition 35. *Soit la fonction* $f : T_{MG_f} \longrightarrow T_{MG_f}$ *permettant de construire l'arbre de fusion image d'un arbre de dérivation d'une GM. Pour cela, on définit l'image des arbres obtenus par les règles des GMs.*

- *pour une fusion : introduction d'un noeud non étiqueté, dont les fils sont les deux arguments de la fusion, se positionnant comme dans une analyse usuelle.*
  Soient t *et* $t' \in T_{MG}(A)$ *tels que* $t = H_t[l :=_h E]$ *et* $t' = H_{t'}[l' :_h E']$ *avec* $h \in B$*, et f(t) et f(t') leur arbre de fusion :*

$$merge(t, t') = \begin{cases} < (H_t[l :\ E], H_{t'}[l' :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[l' :\ E'], H_t[l :\ E]) & \text{sinon.} \end{cases}$$

  et

$$f(merge(t,t')) = \begin{cases} .(H_{f(t)}[l :: E], H_{f(t')}[l' :: E']) & \text{si } t \in Lex, \\ .(H_{f(t')}[l' :: E'], H_{f(t)}[l :: E]) & \text{sinon.} \end{cases}$$

- *pour un déplacement : effacement des traits de déplacement :*
  pour tout arbre $t = C[l : +g\ E, l' : -g\ E']$ *et* $f(t)$ *son arbre de fusion* $= C'[l : +g\ E, l' : -g\ E']$,
  Il existe $C_1, C_2 \in S_t$ *tels que :*
  - $t = C_1[l : +g\ E, C_2[l' : -g\ E']]$,
  - $C_1[l : +g\ E, x_1] = proj_{max}(C[l : +g\ E, x_1])$
  - $C_2[l' : -g\ E'] = proj_{max}(C[l' : -g\ E])$

  $$move(t) = >(C_2[l' : E'], C_1[l : E, \epsilon])$$

  et

  $$f(move(t)) = C'[l : E, l' : E']$$
- *sur l'ensemble des feuilles,* f *est l'identité.*
- *pour* t *un arbre accepté par une GM, on efface le trait acceptant (i.e.* $head(f(t))$ *ne contient pas de trait syntaxique).*

Par abus de notation, pour une GM G *nous écrirons* $f(G)$ *pour désigner l'ensemble des arbres de fusion images des arbres d'analyse de* G *(i.e. les arbres de dérivation de* $L(G)$*).*

On note dans les feuilles de l'arbre les formes phonologiques. Les noeuds de l'*arbre de fusion* sont vides car nous n'utilisons pas la notion de projection maximale pour les opérations de fusion. De plus, les transformations que nous allons opérer sur les arbres minimalistes se vident de la notion de projection. Cette structure représente l'ordre dans lequel les éléments de l'énoncé sont associés. L'arbre de fusion est un historique des fusions nécessaires dans la dérivation.

Exemple 36. *Soit le lexique suivant :*

Pierre :	$d\ -case$	*quel* :	$=n\ d\ -case\ -wh$	*train* :	n
prendre :	$=d\ +case\ =d\ v$				
$infl$:	$<=v\ +case\ V$	*comp* :	$=V\ +WH\ c$		

À partir de celui-ci, nous pouvons calculer les arbres d'analyse et de fusion de l'exemple du chapitre précédent, présentés dans la figure 21.

3.4.2 Équivalence entre GMs pour la fusion

Les GMs sont, en général, écrites par des linguistes qui attribuent des listes de traits linguistiquement cohérentes. Cependant, il est nécessaire de normaliser les lexiques pour une meilleure efficacité des analyseurs syntaxiques. Pour cela, on introduit une notion d'équivalence pour la fusion entre les GMs basée sur des contrainte pour la rédaction des entrées lexicales.

Propriété 37. *On dit que deux GMs* G_1 *et* G_2 *sont* ***équivalentes pour la fusion*** *si pour tout arbre de dérivation* t_1 *de* G_1*, il existe* t_2 *un arbre de dérivation de* G_2 *tel que* $f(t_1) = f(t_2)$.

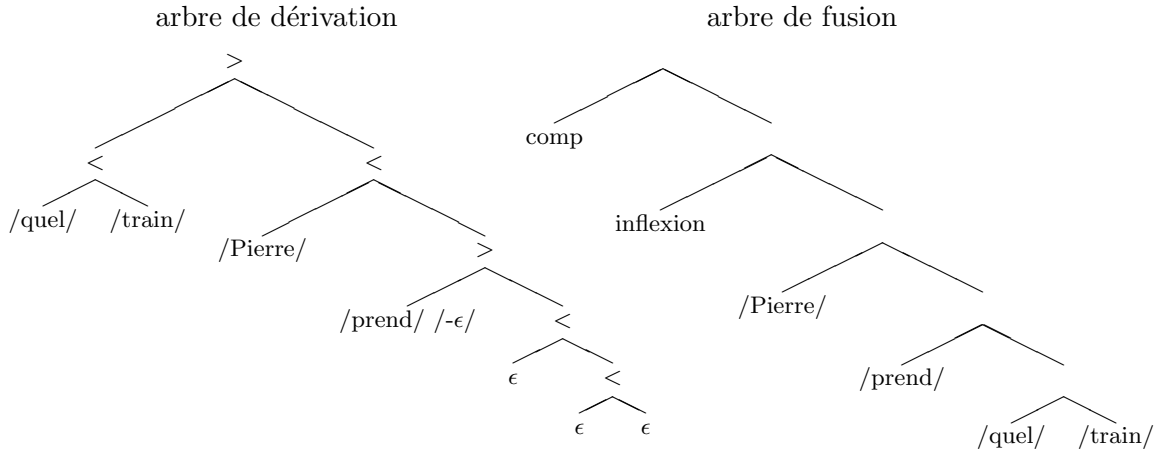
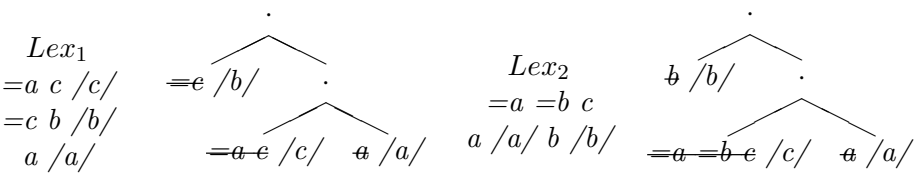


Fig. 21 – Arbres de dérivation et de fusion de "Quel train prend Pierre".

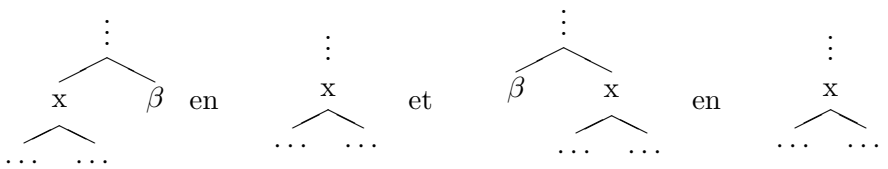
Exemple 38. *Soient deux lexiques Lex_1 et Lex_2 induisant resp. G_1 et G_2 deux GMs telles que les arbres de fusion issus de $L(G_1)$ soient identiques aux arbres de fusion issus de $L(G_2)$. (ici $L(G_1)$ et $L(G_2)$ sont réduits à 'bca').*



À partir d'une GM, il est possible de construire une GM équivalente pour la fusion. En s'inspirant de la mise sous forme normale de Chomsky pour les CFGs, on introduit la notion de mise sous **forme normale pour la fusion** des GMs. On appelle fnf la fonction qui à Lex, le lexique d'une GM G, associe le lexique Lex_{fnf} tel que la GM induite ait la propriété : $f(G) = L(G_{fnf})$.

Cependant, s'inspirer de la mise sous forme normale de Chomsky implique de dissocier complètement les **terminaux** (les formes phonologiques). Pour cela nous introduisons un élément neutre β pour les terminaux.

On appelle *contraction des éléments neutres* la réécriture de l'arbre de fusion qui consiste à, lorsque qu'un noeud possède un fils qui est une feuille β, supprimer ce noeud et le remplacer par le sous-arbre dont l'autre fils est la racine. Ainsi, on transforme : $C[C_1, \beta]$ en $C[C_1]$ et $C[\beta, C_1]$ en $C[C_1]$.



Une grammaire et sa mise sous forme normale de fusion sont équivalentes pour la fusion, à contraction des éléments neutres près.

On dit qu'une GM est sous **forme normale de fusion** si pour chaque entrée de son lexique, soit cette dernière possède deux sélecteurs et une forme phonologique neutre, soit elle contient un trait de base et une forme phonologique non-neutre :

pour x, x_1, x_2 des traits de la grammaire, toute entrée est de la forme :

$x\ /z/$

$=x_1\ =x_2\ x\ /\beta/$

Comme dans les grammaires sous forme normale de Chomsky, il n'y a que deux types d'entrées dans le lexique.

En se basant sur la structure des entrées lexicales, on définit la fonction fnf.

Définition 39. *À partir du Lex_1, on construit le nouveau lexique Lex_{fnf} dans lequel :*

- *on efface les traits de déplacement.*
- *on transforme les entrées obtenues par : soit h une entrée de Lex_1, h est de la forme : $=x_n \cdots =x_1\ y\ /z/$*
  1. *si le nombre de sélecteurs est zéro, on produit l'entrée $y/z/$,*
  2. *si le nombre de sélecteurs est 1, on produit les entrées $Z/z/$ et $= x_1 = Zy/\beta/$,*
  3. *si le nombre de sélecteurs est n, $n \in \mathbb{N}$ et $n \geqslant 2$ on produit les entrées $Z/z/$, $= a_1 = x_1y/\beta/$, $=x_n\ =Z\ a_{n-1}\ /\beta/$ et pour i de 1 à $(n-2)$: $=a_{i+1}\ =x_{i+1}\ a_i\ /\beta/$*

où les traits a_i et Z n'appartiennent pas aux traits de Lex_1. Cette condition implique qu'il n'y a pas d'interférence avec le reste du lexique et donc aucune augmentation de la capacité générative de la grammaire.

Cette nouvelle grammaire est beaucoup plus importante en terme de nombre d'entrées que celle d'origine mais elle dissocie ce qui, dans une entrée, construit la dérivation de son apport phonologique.

Lemme 40. *Soit G une GM, $f(G) = f(L(fnf(Lex_G)))$*

Démonstration. Par induction sur le nombre de sélecteurs :

Si l'entrée ne possède pas de sélecteur les fonctions fnf et f sont l'identité.

Si l'entrée a un unique sélecteur : $\langle =x\ y\ /z/\rangle$ devient $\langle z\ /z/\rangle$ et $\langle =x\ =z\ /\beta/\rangle$

On suppose que y est distinct de c. Dans ce cas, dans une dérivation acceptante dans G, l'entrée est fusionnée avec une entrée portant un x sur sa tête. On notera C_1 cette dérivation. Puis elle est fusionnée avec une entrée commençant par $=y$ sur sa tête, notée C_2. On obtient le contexte : $< (y\ /z/, C_1')$ tel que $head(C_1) = x\ head(C_1')$. Puis, la dérivation se poursuit par : $< (C_2', < (/z/, C_1'))$ tel que $head(C_2) = x\ head(C_2')$. La forme normale de fusion est donc : $.(C_2'', .(/z/, C_1''))$ tels que C_1'' est la forme normale de C_1' et C_2'' celle de C_2'.

Dans la grammaire sous forme normale de fusion, on utilise l'entrée $\langle =x\ =zy/\beta/\rangle$ fusionnée avec C_1. On obtient $< (=z\ y/\beta/, C_1')$. Cette dérivation est alors fusionnée avec l'entrée $\langle z/z/\rangle$. On obtient $t\ => (/z/, < (y/\beta/, C_1'))$ avec $head(t) = y$. Comme dans la grammaire de départ, elle est utilisée dans une fusion avec C_2 : $< (C_2', > (/z/, < (/\beta/, C_1')))$. Le calcul de la forme normale de fusion entraîne la contraction des β-formes

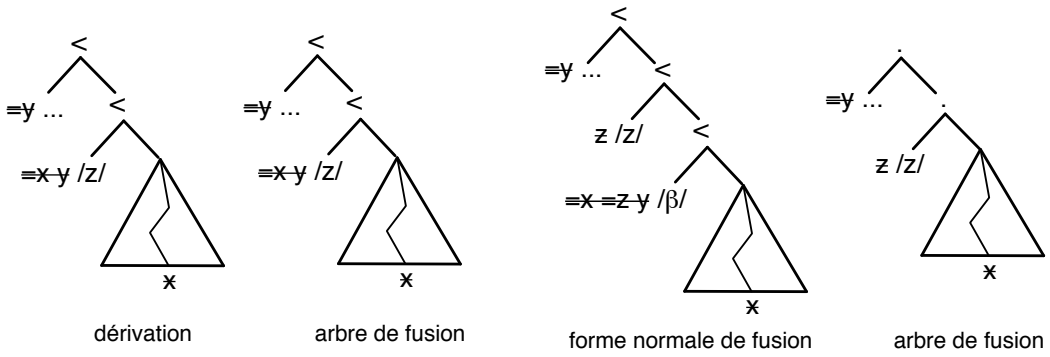


Fig. 22 – Forme normale de fusion.

phonologiques. On obtient la structure : $.(C''_2, .(/z/, C''_1)))$ tels que C''_1 est la forme normale de C'_1 et C''_2 celle de C'_2, ce qui est la même structure que celle obtenue dans G. La figure 22 présente ces deux transformations.

Si $y = c$, le trait est effacé en fin de dérivation. Les deux structures sont bien égales.

On suppose l'égalité de structure induite pour une entrée à k sélecteurs. Les entrées de la grammaire sous forme normale de fusion correspondante sont appelés Lex_k, soit une entrée à $k+1$ sélecteurs.

$Lex_{k+1} = Lex_k - \langle = a_k \; = x_k \; y \; /\beta/\rangle \cup \{\langle = a_{k+1} \; = x_{k+1} \; y \; /\beta/\rangle, \langle = a_k \; = x_k \; a_{k+1} \; /\beta/\rangle\}$

Dans Lex_{k+1} l'entrée $\langle = x_{k+1} \; = z \; a_k/\beta/\rangle$ est fusionnée avec un contexte C_1 dont le premier élément de la tête est x_{k+1}. On obtient $< (= z \; a_k/\beta/, C'_1)$ qui est fusionné avec $\langle z \; /z/\rangle$, c'est-à-dire : $> (/z/, < (a_k/\beta/, C'_1))$. Seule l'entrée $= a_k \; = x_k a_{k-1} \; /\beta/$ peut être utilisée pour obtenir : $< (= x_k a_{k-1} \; /\beta/, > (/z/, < (/\beta/, C'_1)))$. À nouveau cette dérivation sera fusionnée avec l'élément x_k. On a donc positionné dans cette partie de la dérivation x_k et x_{k+1}. Les entrées lexicales restant sont communes à Lex_k et Lex_{k+1}. D'après l'hypothèse d'induction, on conclut que l'utilisation de ces entrées permet de positionner correctement les items x_1 à x_{k-1}.

On a ainsi l'arbre de fusion : $.(C'_2, .(/z/, C'_1)))$, où C'_2 correspond à x_k et C'_1 correspond à x_{k+1}. L'arbre de fusion obtenu à partir de l'entrée de G est identique.

Quelles que soit les entrées utilisées, les constituants se positionnent correctement les uns par rapport aux autres dans la grammaire de départ et dans sa version sous forme normale de fusion. □

3.4.3 Équivalence des GMs et des CFGs

Les structures que nous venons d'introduire trouvent une traduction simple dans un formalisme bien connu et étudié, pour lequel il existe des analyseurs très efficaces : les grammaires hors-contexte - CFG.

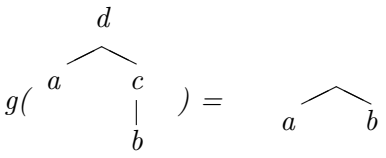
Cependant pour conserver la structure d'arbre minimaliste déjà présente dans ces analyses, la grammaire équivalente devra être sous forme normale de Chomsky.

De plus, nous avons besoin de réécrire les arbres de dérivation des CFGs. Pour les CFGs sous forme normale de Chomsky, nous introduisons une fonction de réécriture sur les arbres de dérivation.

Définition 41. *Soit Arbre l'ensemble des arbres d'analyse obtenus par les CFGs sous forme normale de Chomsky. Ces arbres possèdent des nœuds binaires et unaires. Les feuilles sont étiquetées par les terminaux de la grammaire et les nœuds par les non-terminaux.*

Soit la fonction g : Arbre $\longrightarrow T_{MG_f}$ qui, à un arbre associe cet arbre auquel est effacée l'étiquette de chaque noeud et sont contractés tous les noeuds unaires.

Par exemple $g(d(a, c(b))) = .(a, b)$. Ce qui est équivalent à la représentation graphique :



Par abus de notation nous écrirons, pour G une grammaire, g(G) pour désigner g(arbres de dérivation de G).

On définit la transformation d'une MG en une CFG.

Procédure 42. *Soit G une GM et soit G′ la CFG obtenue par les transformations suivantes :*

Dans G′ :

- *l'ensemble des traits syntaxiques forme V_N.*
- *l'ensemble des formes phonologiques forme V_T.*
- *la transformation de c, le trait acceptant est l'axiome de départ.*

L'ensemble des règles est construit à partir des items lexicaux :

Pour chaque item l :

- *on supprime les traits de déplacement.*
- *l est de la forme $=x_i \cdots =x_1$ y $/z/$, où i est le nombre de sélecteurs. Si :*
  1. *$i = 0$ on ajoute la règle $x \rightarrow /x/$*
  2. *$i = 1$ on ajoute la règle $y \rightarrow /z/x$*
  3. *$i \geqslant 2$ on ajoute la règle $y \rightarrow x_1 \cdots x_{i-1}/z/x_i$*
- *la grammaire est mise sous forme normale de Chomsky.*

Le premier élément se place à droite de la forme phonologique car c'est une entrée lexicale dans une fusion. Les autres terminaux se placent à gauche car c'est une expression qui est utilisée.

Nous appelons GMtoCFG cette procédure.

Lemme 43. *Pour toute GM, il existe une CFG sous forme normale de Chomsky telle que la contraction de l'arbre de dérivation de la CFG est équivalent à l'arbre de fusion de la GM :*

$$g(\text{arbres de dérivation de } CFG_{FNC}) = f(\text{arbres de dérivation de GM}).$$

Démonstration. La procédure 42 permet, à partir d'une GM, d'engendrer une grammaire dont la partie gauche est toujours réduite à un seul élément et dont la partie droite contient au moins un élément. Cette grammaire est donc une CFG qui peut être mise sous forme normale de Chomsky - FNC - selon la procédure standard.

L'arbre d'analyse résultant de cette CFG est alors équivalent. Par induction sur la taille de la dérivation :

1. si l'entrée lexicale n'a qu'un trait de base et une forme phonologique. Elle est transformée en une règle réécrivant un non-terminal (le trait de base) en un terminal (la forme phonologique). Le trait x sera nécessairement effacé dans la dérivation, pour introduire cet élément dans la dérivation ou effacé si c'est le symbole acceptant. La structure d'arbre induite par cette entrée est une feuille portant la forme phonologique.
   Dans la CFG, la règle transformera un non-terminal en un terminal, c'est-à-dire sur la structure d'arbre induite un nœud unaire et un fils portant la forme phonologique. Lors de la contraction le nœud unaire sera remplacé par la feuille portant la forme phonologique.
   On obtient bien les mêmes structures d'arbres.
2. soit l'entrée lexicale ayant un sélecteur : $=x\ y\ /z/$ devient y→ /z/ x dans la CFG, qui, mise sous forme normale de Chomsky, donnera les règles : $y \to Z\ x$ et $Z \to /z/$. Soit C un contexte tel que $head(C) = f$ et le premier trait de f est x. L'arbre de fusion de la GM sera : $.(y/z/, C')$, où C' est C dont le premier trait de la tête est effacé. Dans la suite de la dérivation, y est effacé, on obtient alors $.(/z/, C')$.
   Par la suite, soit y est consommé pour continuer la dérivation, soit c'est le trait acceptant et il est effacé. On obtient donc le sous-arbre de l'arbre de fusion $.(/z/, C')$. L'arbre de dérivation de la CFG sera : $y(Z(/z/), x)$. Par hypothèse d'induction, la dérivation obtenue à partir de x est la même que celle de C'. La contraction de l'arbre de dérivation de la CFG donne : $.(/z/, C')$.
3. soit l'entrée lexicale a plusieurs sélecteurs.
   Les entrées lexicales ayant plusieurs sélecteurs, se réécrivent en plusieurs règles. Chacune introduira une branche binaire dans l'arbre de dérivation. De manière analogue dans la GM, la dérivation utilisera plusieurs fusions pour décharger cette entrée de tous ses sélecteurs. Il y aura exactement les mêmes noeuds construits dans les deux représentations et par hypothèse d'induction, chaque sous-dérivation est de taille inférieure.

GM		CFG		CFG sous FNC
$= x_i\ = x_1\ y\ /z/$	$\Rightarrow$	$y \to x_1 \cdots x_{i-1}/z/x_i$	$\Rightarrow$	$y \to x_1\ a_1$
				$a_1 \to x_2\ a_2$
				$\vdots$
				$a_{i-1} \to x_{i-1}\ a_i$
				$a_i \to Z\ x_i$
				$Z \to /z/$

   Dans la GM le premier élément se placera à droite, ce que reprend l'avant-dernière régle de la CFG, alors que tous les autres se positionneront à gauche dans les deux grammaires.

On note C_k pour $i \in [i]$ les arbres obtenus par des sous-dérivations telles que le premier trait de $head(C_k)$ est x_k. Arbre de fusion de la GM :

$$f(> (C_1, > (C_2, (\cdots, < (y/z/, C_i))))) = \\ .(f(C_1), .(f(C_2), (\cdots .(/z/, f(C_i)))))$$

où y sera soit consommé dans la dérivation, soit effacé si c'est le symbole acceptant.

Par hypothèse d'induction, chaque dérivation produisant les C_k est de taille inférieure à celle produit, donc chacun des C_k fournit un arbre d'analyse équivalent dans la CFG, noté C_k. Dans la CFG, à partir de y on dérive l'arbre :

$$y(C_1, a_1(C_2, a_2(\cdot a_{i_1}(C_{i-1}, a_i(Z(/z/), C_i)))))$$

La contraction de cet arbre donne :

$$.(g(C_1), .(g(C_2), .(\cdots .(g(C_{i-1}), .(/z/, g(C_i))))))$$

Par hypothèse d'induction on a : $f(C_k) = g(C_k)$ et donc la structure induite par la GM est la même.

Quelle que soit l'opération utilisée, nous obtenons bien les mêmes structures d'arbres dans les deux grammaires. Les deux structures sont donc isomorphes et $f(GM) = g(CFG_{fnc})$.

On remarquera que dans les CFGs transformées, il existe plusieurs dérivations mais une seule permet d'aboutir à l'énoncé de départ. □

De manière analogue, on donne la procédure réciproque qui permet, à partir d'une CFG de construire une GM sans trait de déplacement.

Procédure 44. *Soit G une CFG sous forme normale de Chomsky, il n'y a donc que deux types de règle possibles dans G :*

- $V \rightarrow UW$
- $W \rightarrow /x/$

Pour chaque règle de la CFG, on ajoute au lexique de la GM G' :

1. *si la règle est de la forme $V \rightarrow /x/$ où x est un termial et V un non-terminal, l'entrée : $V/x/$.*
2. *si la règle est de la forme $V \rightarrow UW$ où U, V et W sont des non-terminaux, l'entrée : $=W=UV/\beta/$.*

Nous appelons $CFGtoGM$ cette procédure.

Lemme 45. *Pour toute CFG contractée sur les feuilles et non étiquetée sur les noeuds, il existe une GM sans trait de déplacement dont les arbres d'analyse sont équivalents.*

Démonstration. Lemme 45

La structure des arbres induits est beaucoup plus intuitive à aborder car, à chaque règle de la CFG correspond une unique règle dans la GM. Les arbres équivalents sont les arbres contractés sur les formes phonologiques des arbres d'analyse obtenus dans les GMs. Ces dernières sont des GMs sous forme normale de fusion. Pour retrouver la même structure d'arbre, pour les noeuds unaires sans forme phonologique sur le noeud, on contracte l'arbre, ce qui fait remonter la forme phonologique.

Par induction sur la taille de la dérivation :

– Pour les règles de type $V \rightarrow /x/$, nous aurons l'arbre de dérivation : $V(/x/)$, $g(V(/x/)) = /x/$.
La traduction dans la GM est l'entrée $V/x/$. À partir de cette entrée, on peut construire l'arbre de dérivation $V/x/$, enfin, $f(F(V/x/)) = F(/x/)$ car V sera consommé pour introduire l'élément dans la dérivation ou effacé si c'est le symbole acceptant, ce qui montre l'égalité des structures.
– Pour les règles du type : $V \rightarrow UW$ où U, V et W sont des non-terminaux, nous aurons l'arbre de dérivation : $V(C_U, C_W)$ où C_U et C_W sont les arbres de dérivation induits par U et W.

$$g(V(C_U, C_W)) = .(g(C_U), g(C_W))$$

Sa traduction dans la GM est $=W{=}UV/\beta/$. Nous aurons l'arbre de dérivation $>((C'_U), <(V/\beta/, C'_W))$ où C'_U et C'_W sont les arbres de dérivation induits par U et W. On obtient alors :

$$f(>((C'_U), <(V/\beta/, C'_W))) = .(f(C'_U), f(C'_W))$$

Or, d'après l'hypothèse d'induction on sait que $g(C_U) = f(C'_U)$ et $g(C_W) = f(C'_W)$. On obtient donc les mêmes dérivations.

□

Ces transformations sont résumées dans la figure 23.

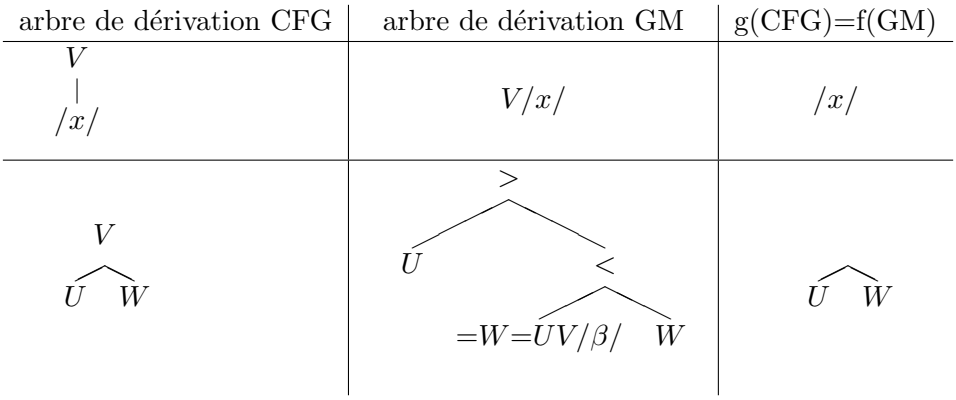


Fig. 23 – CFG $\longrightarrow$ GM

Théorème 46. *Les arbres de fusion des GMs sont équivalents aux arbres d'analyse des CFGs sous forme normale de Chomsky, à normalisation de fusion près.*

Démonstration. D'après le lemme 43, il y a équivalence entre une GM et une CFG pour les arbres de fusion et de dérivation.

En utilisant le lemme 45, on construit à partir d'une CFG une GM dont les arbres de dérivation contractés sur les formes phonologiques sont équivalents aux arbres d'analyse de la CFG.

De plus, il y a mise sous forme normale de fusion entre la GM de départ et le résultat de la double transformation : soit G une GM, pour les entrées de types :

- $y/z/$
  $CFGtoGM(GMtoCFG(y/z/)) = CFGtoGM(y \to /z/)) = y/z/$
  $fnf(y/z/) = y/z/$
- $= xy/z/$

$$CFGtoGM(GMtoCFG(= xy/z/)) = CFGtoGM\begin{pmatrix} y \to Zx \\ Z \to /z/ \end{pmatrix} = \begin{pmatrix} =x & =Z & y/\beta/ \\ & & Z/z/ \end{pmatrix}$$

$$\text{fnf(=x y /z/)} = \begin{pmatrix} =x & =Z & y/\beta/ \\ & & Z/z/ \end{pmatrix}$$

- $= x_i \cdots x_1\ y/z/$

$$CFGtoGM(GMtoCFG(G)) = CFGtoGM\begin{pmatrix} y \to x_1\ a_1 \\ a_1 \to x_2\ a_2 \\ \vdots \\ a_{i-1} \to Z\ x_i \\ Z \to /z/ \end{pmatrix} = \begin{pmatrix} =a_1 & =x_1 & y/\beta/ \\ =a_2 & =x_2 & a_1/\beta/ \\ & \vdots & \\ =Z & =x_i & a_{i-1}/\beta/ \\ & Z/z/ & \end{pmatrix}$$

$$fnf(= x_i \ldots x_1 y/z/) = \begin{pmatrix} =a_1\ =x_1\ y/\beta/ \\ =a_2\ =x_2\ a_1/\beta/ \\ \vdots \\ =Z\ =x_i\ a_{i-1}/\beta/ \\ Z/z/ \end{pmatrix}$$

Donc quel que soit le type d'entrée, et donc de dérivation induite, les transformations des grammaires redonnent la grammaire de départ sous forme normale de fusion :

$$CFGtoGM(GMtoCFG(G)) = fnf(G).$$

□

La figure 24 résume ces équivalences. La conséquence directe de ce théorème est que pour toute CFG sous forme normale de Chomsky, il existe une GM sans trait de déplacement faiblement équivalente, et une GM sous forme normale de fusion fortement équivalente à contraction des arbres de dérivation près. C'est-à-dire qu'un simple transducteur permet de passer de l'une à l'autre de ces représentations.

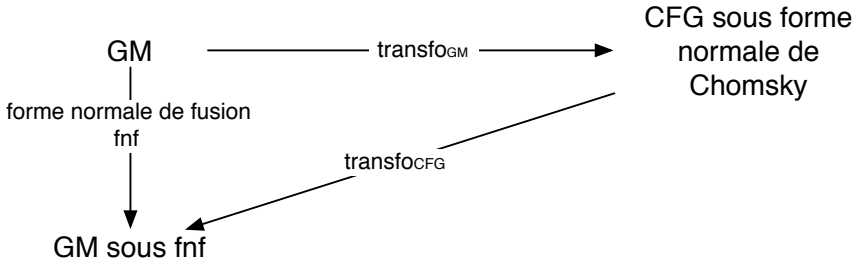


FIG. 24 – Équivalence pour la fusion.

Exemple 47. *Afin d'exhiber ces transformations nous les appliquons sur le lexique de l'exemple 36*

CFG avant mise sous FNC :	*transfo$_{GM}(L)$:*	
$C \rightarrow /\epsilon/\ V$	$C \rightarrow C'\ t$	$C' \rightarrow /comp/$
$V \rightarrow /inf/\ v$	$t \rightarrow I\ v$	$I \rightarrow /inf/$
$v \rightarrow d\ /prendre/\ d$	$v \rightarrow d\ v'$	$d \rightarrow /Pierre/$
$n \rightarrow /train/$	$v' \rightarrow v''\ d$	$v'' \rightarrow /prendre/$
$d \rightarrow /quel/\ n$	$d \rightarrow d'\ n$	$n \rightarrow /train/$
$d \rightarrow /Pierre/$		$d' \rightarrow /quel/$

Dans cette grammaire on obtient la dérivation suivante de la phrase "quel train Pierre prend" :

$C \rightarrow C'\ t \rightarrow /comp/\ t \rightarrow /comp/\ I\ v \rightarrow /comp/\ /inf/\ v \rightarrow /comp/\ /inf/\ d\ v' \rightarrow /comp/\ /inf/\ /Pierre/\ v' \rightarrow /comp/\ /inf/\ /Pierre/\ v''\ d \rightarrow /comp/\ /inf/\ /Pierre/\ /prendre/\ d \rightarrow /comp/\ /inf/\ /Pierre/\ /prendre/\ d'\ n \rightarrow /comp/\ /inf/\ /Pierre/\ /prendre/\ d'\ /train/ \rightarrow /comp/\ /inf/\ /Pierre/\ /prendre/\ /quel/\ /train/$

On peut représenter cette phrase sous forme d'arbre minimaliste, cela donne exactement le même arbre que celui de droite dans la figure 21.

Résultat de la transformation de la CFG sous forme normale de Chomsky en GM selon la procédure 44, transfo$_{CFG}$(transfo$_{GM}(L)$) :

CFG	*GM*	*CFG*	*GM*
$C \rightarrow C'\ t$	$=t\ =C'\ C\ /\beta/$	$v' \rightarrow v''\ d$	$=d\ =v''\ v'\ /\beta/$
$C' \rightarrow /comp/$	$C'\ /comp/$	$v'' \rightarrow /prendre/$	$v''\ /prendre/$
$t \rightarrow I\ v$	$=v\ =I\ t\ /\beta/$	$d \rightarrow d'\ n$	$=n\ =d'\ d\ /\beta/$
$I \rightarrow /inf/$	$I\ /infl/$	$n \rightarrow /train/$	$n\ /train/$
$v \rightarrow d\ v'$	$=v'\ =d\ v\ /\beta/$	$d' \rightarrow /quel/$	$d'\ /quel/$
$d \rightarrow /Pierre/$	$d\ /Pierre/$		

Ce lexique est le même que celui de l'exemple 36 sous forme normale de fusion présenté ci-dessous.

$=t\ =C'\ C\ /\beta/$	$I\ /infl/$	$=d\ =v''\ v'\ /\beta/$	$n\ /train/$
$C'\ /comp/$	$=v'\ =d\ v\ /\beta/$	$v''\ /prendre/$	$d'\ /quel/$
$=v\ =I\ t\ /\beta/$	$d\ /Pierre/$	$=n\ =d'\ d\ /\beta/$	

3.5 Représentation abstraite des suites possibles de fusions d'un lexique

À partir de la suite de fusions, que nous venons d'étudier, pour une analyse donnée, nous avons élargi la modélisation à la question de toutes les fusions réalisables pour un lexique. La structure obtenue ne représente plus rien de syntaxiquement compréhensible puisque pour chaque type différent il peut exister un grand nombre de terminaux, tous les noms d'un lexique sont alors représentés par autant de représentants.

Cependant, on constate que la structure obtenue modélise le squelette de l'ensemble des dérivations réalisables. En abstrayant les terminaux sur des types uniques (un seul représentant pour tous les noms), on perd la structure d'arbre minimaliste, mais on obtient une nouvelle structure qui permet d'avoir une vision globale des dérivations possibles à partir d'un lexique.

Nous cherchons donc à représenter les différentes compositions d'éléments que l'on peut obtenir. Pour cela, nous utiliserons une représentation sous forme de circuit.

Définition 48. *Un **circuit** est une représentation de relations spécifiées par un ensemble de sommets S et un ensemble d'arcs orientés A.*

- *les **sommets** sont des atomes simples.*
- *les **arcs** sont des fonctions qui à un ensemble de sommets associent un sommet $S^* \rightarrow S$. On définit le **demi-arc entrant** comme la partie de l'arc atteignant le sommet final et le **demi-arc sortant** la partie de l'arc partant du sommet d'origine.*

Nous ne posons pas de limite aux réalisations de circuits. Les arcs peuvent être multiples : un même sous-ensemble de sommets peut posséder des arcs pointant vers des sommets distincts. Le demi-arc entrant étant unique, le circuit possédera plusieurs ensembles de demi-arcs sortants identiques.

Ce formalisme simple est très proche des graphes utilisés en informatique. La différence réside dans la formation des arcs qui ici utilise la notion de *conjonction*. De plus, dans la composition d'un arc, il n'y a pas d'ordre spécifique de l'ensemble des arcs sortants. Cette propriété nous fait perdre la correspondance avec la structure de liste utilisée dans la définition des entrées lexicales des GMs.

Nous donnons une procédure permettant de passer d'un lexique des GMs vers un circuit.

Procédure 49.

1. *l'ensemble S est composé des catégories de base.*
2. *La première étape consiste à abstraire sur l'ensemble du lexique l'ensemble des séquences de traits utilisées. Nous ne prenons donc qu'une seule fois en considération chaque séquence de traits spécifiques, sans la relier à une forme phonologique donnée.*
3. *Pour chaque séquence de traits dans cet ensemble, on efface tous les traits de déplacement.*

Pour chaque type d'entrée lexicale différent, on associe :

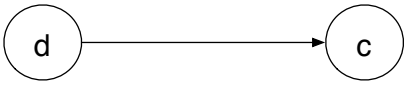
- *si l'entrée commence par un trait de base : un sommet.*

  L'entrée lexicale : [] : :[a]. devient :

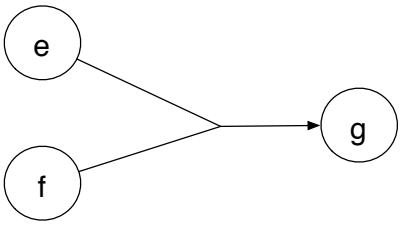
- *si l'entrée commence par un sélecteur : un demi-arc sortant du sommet représentant le sélecteur et entrant dans le sommet du trait de base.*

  L'entrée lexicale : [] : :[=d ; +f ; c]. devient : ⇒

- *Si l'entrée contient plusieurs sélecteurs, on place un demi-arc sortant de chaque sommet représentant le sélecteur et toutes se rejoignent sur le demi-arc entrant dans le sommet du trait de base.*

  Les entrées lexicales :
  [] : :[=e ; =f ; g].
  [] : :[f]. deviennent :



Une version de ces circuits peut être proposée en prenant en compte les différents traits de déplacement. Ils servent à étiqueter les demi-arcs des circuits. Ainsi, on placera sur les demi-arcs sortants les assignés qui suivent le trait de base, représenté dans le sommet, et sur le demi-arc entrant les assignateurs possibles.

Les circuits émanant de cette seconde version sont plus complexes car une séquence de sélecteurs donnée peut être suivie de différents assignés. Cependant, pour garder une structure similaire, on composera sous forme de | (ou exclusif) ces différentes séquences possibles. La reconstruction du lexique à partir de cette structure devient alors ambiguë, mais pour une analyse donnée, les entrées nécessaires sont alors désignées. La figure 25 présente un exemple de ce type de représentation.

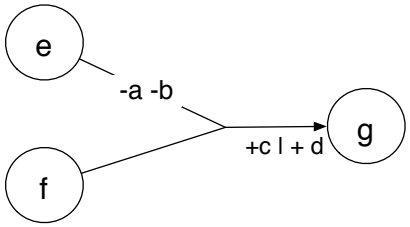


Fig. 25 – Exemple de circuit avec étiquette sur les arcs.

Un exemple de circuit à partir du lexique de l'exemple 36 est donné en figure 26.

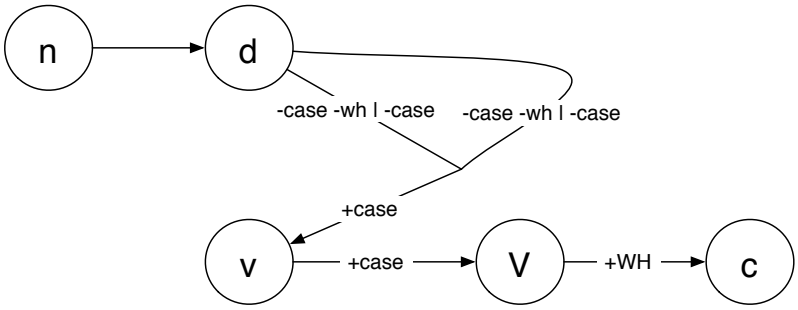


Fig. 26 – Exemple de circuit abstrayant un lexique.

Les *arbres de fusion* que nous avons étudiés dans ce chapitre, sont en fait des arbres couvrants de cette structure (permettant de lier tous les éléments de la sélection entre eux) où la racine principale est la catégorie acceptant et les feuilles sont des atomes unitaires non abstraits (pour lesquelles on différencie chaque noeud par sa forme phonologique).

Cependant, tous les arbres que l'on peut obtenir à partir d'un circuit, ne seront pas acceptés par la GM puisqu'ils dépendent des déplacements qui seront à effectuer dans la dérivation.

Cette structure ne semble pas, pour le moment, avoir un intérêt théorique, mais se révèle être un outil performant par exemple lors de la rédaction d'un lexique. Cette phase est primordiale pour la production de grammaires à large couverture pour extraire des représentations syntaxiques et sémantiques mais aussi pour valider les hypothèses faites pour la grammaire. Les circuits permettent notamment de vérifier l'intégrité d'un lexique lors de sa rédaction (l'ajout d'un nom ou d'un verbe transitif ne doit pas modifier structurellement le circuit présenté dans la figure 26) ou encore, de vérifier que chaque morceau du circuit est bien connecté au symbole acceptant. À partir de ces structures, on peut également vérifier qu'il existe une unique manière de composer des entrées lexicales pour rendre compte de divers phénomènes syntaxiques comme nous le verrons dans le chapitre 8, ou que la présence de plusieurs chemins trouve une explication linguistique pour l'analyse d'un certain type d'énoncé (ambiguïté des analyses syntaxiques).

Un autre rôle de cette structure est de permettre de filtrer les analyses possibles où impossibles. La particularité des GMs est d'introduire des éléments à phonologie vide sur les chaînes à analyser, comme nous l'avons fait pour l'inflexion et l'élément complementizer. Cela implique qu'à partir d'une sélection, il faut construire tous les ensembles contenant au moins la sélection, plus toutes les combinaisons possibles d'entrées à phonologie vide, pour identifier la sélection correcte. Cependant, une infime minorité de ces ensembles permet d'aboutir à une analyse acceptante, due à l'impossibilité de les combiner. Ces circuits permettent rapidement de savoir si une sélection est *couvrante*. Une implémentation de l'extraction de représentation abstraite des lexiques en Ocaml est présentée dans l'annexe B. L'implémentation de ce filtre dans l'analyseur des GMs n'est pas encore réalisée.

L'utilisation de circuits permet d'obtenir une représentation connexe du lexique. On peut également utiliser la notion d'automates récursifs pour modéliser les transitions, ce formalisme est développé dans [Tel05]. Le passage d'un sommet à un autre est alors conditionné au succès de l'analyse d'un autre automate. Ainsi, une transition dans la représentation est une partie d'analyse plus élaborée que la reconnaissance d'une chaîne. Le problème subsistant est alors de factoriser les automates, par exemple, comment encoder pour un verbe la nécessité de le réarranger avec deux groupes nominaux.

3.6 conclusion

À partir de la hiérarchie de Chomsky qui permet de classer les grammaires les une par rapport aux autres, nous avons abordé quelques résultats sur la capacité générative des GMs en présentant la vue de Michaelis des dérivations en les associants aux files. La capacité générative des files étant connues (type-0), les GMs sous certaines conditions ont alors une expressivité trop importante.

Nous avons ensuite présenté deux perspectives, l'une basée sur la fusion et le déplacement, l'autre se concentrant exclusivement sur la fusion. Le résultat montre que l'opération de fusion dans les GMs permet uniquement d'exprimer des grammaires algébriques. C'est l'utilisation de l'opération de déplacement qui permet d'augmenter l'expressivité de la grammaire. Ainsi, sans aller jusqu'à une classe de langage bien trop importante, il faut alors déterminer les critères permettant d'obtenir un formalisme exactement MCS.

Chapitre 4

Exemple de lexiques et étude de la capacité générative

Sommaire

4.1 Représentation simplifiée des GMs 88
4.2 Phrase sur un compteur 88
4.3 Compteurs enchâssés 96
4.4 Duplications 101
4.5 Duplication multiple 105
4.6 Phrase de Fibonacci 112
4.7 Conclusion 118

Dans ce chapitre, nous revenons sur la capacité générative des GMs. Une des questions ouvertes pour ces travaux était d'identifier clairement les mécanismes mis en oeuvre dans ces grammaires. Le but était d'identifier d'une part les conditions à utiliser pour limiter/augmenter la capacité générative de ces grammaires et d'autre part d'essayer d'identifier des conditions sur la rédaction des lexiques permettant de proposer des analyseurs syntaxiques plus performants. Le résultat de Michaelis, [MK05], présenté dans la section 3.2 a clos une partie des questions pour lesquelles nous n'avons donc pas eu besoin de persévérer. Cependant, nous exposons ici des GMs reconnaissant un grand nombre de langages théoriques connus.

Nous commencerons par présenter les langages de compteurs généralisés : $1^n 2^n \dots m^n$ publiés dans [Amb05a]. Puis, nous montrerons, comme extension des précédents compteurs, comment générer un lexique permettant l'analyse des compteurs enchâssés : $1^{a_1} 2^{a_2} 3^{a_1} 4^{a_2}$, $\forall a_i, i \in \mathbb{N}$. Enfin, dans une dernière section nous présenterons les lexiques et les scénarii pour l'obtention des duplications inverses, des duplications multiples et des phrases de Fibonacci - soit $\omega = a^{F(n)}$ pour $n \in \mathbb{N}$ et $F(n)$ la fonction de Fibonacci. L'utilisation de la reconnaissance des phrases de Fibonacci a été utilisée pour rapprocher les mécanismes mis en œuvre dans la correspondance entre les GMs et les automates à piles de piles. Cependant, ces travaux ne sont pas présentés dans ce manuscrit.

4.1 Représentation simplifiée des GMs

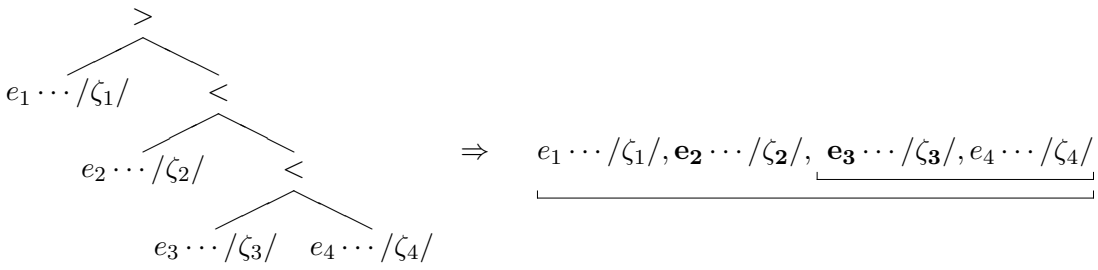
Ce chapitre contient de nombreux exemples de dérivations. Afin d'en simplifier la lecture nous représentons plus simplement les dérivations.

Une entrée lexicale est composée d'une suite de traits ainsi que de la forme phonologique associée, notée entre barres obliques : $e_1 E/\zeta_1/$. Une lecture gauche-droite des formes phonologiques de l'analyse permet d'identifier la phrase reconnue par cette dernière.

Traditionnellement, les analyses sont des arbres minimalistes. Dans cette section, nous utiliserons des listes ordonnées gauche-droite. Un constituant sera délimité par un crochet en-dessous et la tête de ce dernier sera marquée en gras.

Par souci de simplification de la représentation graphique, les groupes ne contenant plus qu'un élément et ceux ne contenant que des formes phonologiques ne seront plus soulignés par un crochet, et la tête reprendra une *fonte* normale.

Exemple 50. *Exemple de transformation d'un arbre minimaliste en représentation plane :*



$e_1 \cdots /\zeta_1/$ $<$

$e_2 \cdots /\zeta_2/$ $<$

$e_3 \cdots /\zeta_3/$ $e_4 \cdots /\zeta_4/$

$\Rightarrow \quad e_1 \cdots /\zeta_1/, \mathbf{e_2} \cdots /\zeta_2/, \underbrace{\mathbf{e_3} \cdots /\zeta_3/, e_4 \cdots /\zeta_4/}$

4.2 Phrase sur un compteur

Cette section présente un algorithme fournissant un lexique engendrant une GM reconnaissant le langage $1^n 2^n \ldots N^n, \forall n \in \mathbb{N}$, et ce pour tout $N \in \mathbb{N}$. Elle contient aussi un théorème sur l'appartenance aux langages minimalistes des phrases sur un compteur.

On note les traits de base et les sélecteurs par la lettre s et les traits de déplacement par la lettre m. Ces derniers sont indicés pour leur relation avec l'une des formes phonologiques.

Algorithme 51. *On supposera les formes phonologiques (ou terminaux de la grammaire) ordonnés :* $1 < \ldots < N-1 < N$ *selon leur ordre d'apparition dans la phrase et on distingue un trait particulier supplémentaire c, le symbole acceptant de la grammaire. Le lexique - noté* Lex_N *- est l'ensemble des entrées produites par :*

type 1 : $\langle s_N \text{ -}m_N \ /N/\rangle$
type 2 : pour i de 1 à (N-1)
$\langle {=}s_{i+1}\ s_i \text{ -}m_i\ /i/\rangle$
type 3 : pour j de 1 à (N-1)
$\langle {=}s_{j+1} \ {+}m_j\ s_j \text{ -}m_j\ /j/\rangle$
type 4 : $\langle {=}s_1 \ {+}m_N\ s_N \text{ -}m_N\ /N/\rangle$
type 5 : $\langle c\rangle$
type 6 : $\langle {=}s_1 \ {+}m_N \ \ldots \ {+}m_1\ c\rangle$

Synopsis de Dérivation 52. *Les dérivations se décomposent en trois phases :*

1. ***Initialisation :*** *les entrées de type 1 et 2 permettent d'entamer une dérivation en faisant entrer une fois chaque terminal dans la dérivation. À la fin de cette phase, à chaque terminal est associé un groupe constitué d'une seule occurrence de chacun d'eux et un assigné du même trait :* $-m_i\ /i/$.
2. ***Itération :*** *la phase d'itération est analogue à la phase d'initialisation. Elle est réalisée par les entrées de type 3 et 4 dans l'ordre induit par leur séquence de traits. Chacune d'entre elles ajoute un terminal et déplace le groupe associé à ce terminal pour former un nouveau groupe homogène contenant ce terminal supplémentaire. L'enchaînement des entrées de ce type garantit l'ajout d'une occurrence de tous les terminaux.*

$$\cdots, \underbrace{/i/, \cdots, -m_i\ /i/}$$
$$+m_i\ s_i\ -m_i\ /i/, \cdots, \underbrace{/i/, \cdots, -m_i\ /i/}$$
$$\underbrace{/i/, \cdots, -m_i\ /i/}, s_i\ -m_i\ /i/, \cdots,$$

3. ***Conclusion :*** *l'entrée de type 6 correspond à la dernière phase qui efface les assignés de chaque groupe de terminaux qui sont utilisés dans la phase 2.*

L'entrée de type 5 permet d'accepter le cas $n = 0$.

On note $\phi_{init}(Lex_N)$ (*resp.* ϕ_{iter} et ϕ_{conc}) l'ensemble des entrées utilisées dans la phase d'initialisation (*resp.* d'itération et de conclusion). On dit d'un arbre de dérivation pour un tel lexique qu'il est dans la phase d'initialisation (*resp.* itération ou conclusion) si la dernière entrée utilisée est dans $\phi_{init}(Lex_N)$ (*resp.* $\phi_{iter}(Lex_N)$ ou $\phi_{conc}(Lex_N)$). On appelle phase d'initialisation (*resp.* itération ou conclusion) une séquence de la dérivation utilisant toutes les entrées de ϕ_{init} (*resp.* ϕ_{iter} et ϕ_{conc}).

Un exemple de compteurs : 1^n2^n

Pour reconnaître une phrase de deux lettres sur un compteur 1^n2^n l'algorithme 51 fournit le lexique suivant :

Lexique 53.

phase d'initialisation			
type 1 ::	$s_2\ -m_2\ /2/$	*type* 2 ::	$=s_2\ s_1\ -m_1\ /1/$
phase d'itération			
type 4 ::	$=s_1\ +m_2\ s_2\ -m_2\ /2/$	*type* 5 ::	c
phase de conclusion			
type 3 ::	$=s_2\ +m_1\ s_1\ -m_1\ /1/$	*type* 6 ::	$=s_1\ +m_2\ +m_1\ c$

La représentation abstraite de ce lexique par un circuit est donnée dans la figure 27.

Dérivation 54.

1. *entrée lexicale de type 1 :* $s_2\ -m_2\ /2/$
   et de type 2 : $= s_2\ s_1\ -m_1\ /1/$

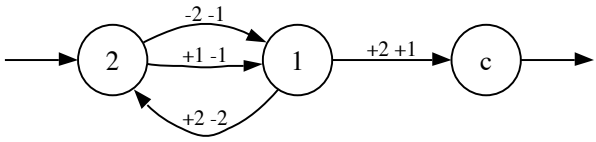


FIG. 27 – Représentation abstraite du lexique pour deux terminaux sur un compteur.

2. *fusion :* $\underline{\mathbf{s_1}\ -\mathbf{m_1}\ /\mathbf{1}/,\ -s_2\ /2/}$

3. *La phase d'initialisation est terminée. La dérivation contient une forme phonologique de chaque terminal. On peut soit terminer la dérivation et obtenir /1//2/, soit passer dans la phase d'itération : type 4 :* $= s_1\ + m_2\ s_2\ - m_2\ /2/$

   et une fusion avec la dérivation précédente :

   $\underline{+\mathbf{m_2}\ \mathbf{s_2}\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{-\mathbf{m_1}\ /\mathbf{1}/},\ -m_2\ /2/}$

4. *Le dernier /2/ qui vient d'être ajouté implique un déplacement du groupe de /2/ :*

   $\underline{/2/,\ \mathbf{s_2}\ -\mathbf{m_2}\ /\mathbf{2}/,\ -m_1\ /1/}$

5. *On a donc un /2/ de plus, il faut ajouter un /1/, ce qui se fait par une entrée lexicale de type 3 :* $= s_2\ + m_1\ s_1\ - m_1\ /1/$

   – deuxième partie de l'itération – et fusion :

   $\underline{+\mathbf{m_1}\ \mathbf{s_1}\ -\mathbf{m_1}\ /\mathbf{1}/,\ \underline{/2/},\ -\mathbf{m_2}\ /\mathbf{2}/,\ -m_1\ /1/}$

6. *Le groupe de /1/ s'homogénéise grâce à un déplacement :*

   $\underline{/1/,\ \mathbf{s_1}\ -\mathbf{m_1}\ /\mathbf{1}/,\ \underline{/2/},\ -\mathbf{m_2}\ /\mathbf{2}/}$

7. *On est alors revenu dans la même configuration qu'à l'étape 2, avec un /1/ et un /2/ de plus dans chaque groupe. Le même choix se présente : réitérer ou conclure. Passons à la conclusion.*

   Entrée lexicale de type 6 : $= s_1\ + m_2\ + m_1\ c$

   et fusion : $\underline{+\mathbf{m_2}\ + \mathbf{m_1}\ \mathbf{c},\ ,/1/,\ -\mathbf{m_1}\ /\mathbf{1}/,\ \underline{/2/},\ -\mathbf{m_2}\ /\mathbf{2}/}$

8. *Chaque groupe est maintenant déplacé dans l'ordre inverse d'apparition dans la phrase, on commence donc par le groupe de /2/.*

   Déplacement : $/2/,\ /2/,\ +\mathbf{m_1}\ \mathbf{c},\ \underline{/1/,\ -\mathbf{m_1}\ /\mathbf{1}/}$

9. *Enfin, on déplace le groupe de /1/ et acceptation :* $\underline{/1/,\ /1/,\ /2/,\ /2/,\ \mathbf{c}}$

Théorème 55. *Les langages de compteurs sont des langages minimalistes.*

Soit $p \in \mathbb{N}$ et G_p la GM engendrée par le lexique produit par l'algorithme 51 pour p : Soit $L_p = \{a_1^n \cdots a_p^n | n \in \mathbb{N}\}$.

$$L_p = L(G_p)$$

Pour prouver ce théorème, on définit la propriété suivante, qui est utilisée comme invariant entre les phases de la dérivation :

Propriété 56. *Soit $t_p \in T_{MG}$, la propriété $P(t_p)$ est vraie si : $\exists C_1, \ldots C_p \in S_{t_p}$ tels que :*

1. $head(t) = head(C_1)$,
2. $head(C_1) = \langle s_1\ -m_1\ /a_1/\rangle$, $head(C_2) = \langle -m_2\ /a_2/\rangle$, $\ldots$, $head(C_p) = \langle -m_p\ /a_p/\rangle$
3. $C_1 \lhd \ldots \lhd C_{p-1} \lhd C_p$,
4. *pour $i \in [p]$, pour toute feuille u de C_i n'appartenant pas à C_{i+1} et différente de la tête de C_i alors $u = \langle /a_i/\rangle$.*

Lemme 57. *L'arbre t_p obtenu par la phase d'initialisation est tel que $P(t_p)$ est vraie.*

Démonstration. Par induction sur p :

- Si $p = 0$, $\omega = \epsilon$. L'entrée de type 5 permet de le reconnaître sans phase d'initialisation.
- Si $p = 1$, l'entrée de type 1 nous donne l'arbre t_0 réduit à la feuille $\langle s_1\ -m_1\ /a_1/\rangle$. La phase d'initialisation est alors terminée. Alors, il existe $C_1 \in S_{t_0}$ tel que $C_1 = t_0$
  1. $head(t_0) = head(C_1)$
  2. $head(C_1) = \langle s_1\ -m_1\ /a_1/\rangle$
  3. C_1
  4. il n'existe pas $u \in C_1$ tel que $u \neq head(C_1)$.
- par hypothèse d'induction, pour $k < p$ la phase d'initialisation vérifie la propriété. Si $k = p$, les entrées de la phase d'initialisation - ϕ_{init}- des lexiques Lex_p et Lex_{p-1} sont tels que : $\phi_{init}(Lex_p) = \phi_{init}(Lex_{p-1}) - \{\langle s_{p-1}\ -m_{p-1}\ /a_{p-1}/\rangle\} \cup \{\langle s_p\ -m_p\ /a_p/\rangle \cup \langle =s_p\ s_{p-1} - m_{p-1}\ /a_{p-1}/\rangle\}$
  La phase d'initialisation commence alors par la fusion d'un item de type 1 et l'item de type 2 d'indice maximal. on obtient $t_0' =< (s_{p-1} - m_{p-1}/a_{p-1}/, -m_p/a_p/)$ dans lequel il existe C_p tel que $head(C_p) = -m_p\ /a_p/$.
  C_{p-1} est alors tel que $head(t_0') = head(C_{p-1}) = \langle s_{p-1}\ -m_{p-1}\ /a_{p-1}/\rangle$. Soit :
  1. $head(C_p) = \langle -m_p\ /a_p/\rangle$
  2. $C_{p-1} \lhd C_p$
  3. il n'existe pas $u \in C_p$ tel que $u \neq head(C_p)$

  Par hypothèse d'induction et en utilisant les entrées communes à $\phi_{init}(Lex_p)$ et $\phi_{init}(Lex_{p-1})$, à la fin de la phase d'initialisation on a :
  1. $head(t_p) = head(C_1)$,
  2. $head(C_1) = \langle s_1\ -m_1\ /a_1/\rangle$, $head(C_2) = \langle -m_2\ /a_2/\rangle$, $\cdots$, $head(C_{p-1}) = \langle -m_{p-1}\ /a_{p-1}/\rangle$

3. $C_1 \lhd \ldots \lhd C_{p-1}$,
4. pour $i \in [p-1]$, $\forall u \in C_i \wedge u \notin C_{i+1}$, $u \neq head(C_i)$ alors $u = \langle /a_i/ \rangle$

En composant ces propriétés avec celles exhibées pour C_p, la phase d'initialisation vérifie alors la propriété P (propriété 56).

□

Lemme 58. *Soit $t_p \in T_{MG}$ un arbre obtenu lors de la phase d'itération. Pour $k \in [p]$, on définit la propriété $Q_k(t_p)$, vérifiée s'il existe $C_1, \ldots, C_p$ tels que :*

1. $head(t_p) = head(C_k)$
2. $head(C_k) = \langle s_k \ -m_k \ /a_k/ \rangle$, $head(C_{k+1}) = \langle -m_{k+1} \ /a_{k+1}/ \rangle$, $\ldots$, $head(C_p) = \langle -m_p \ /a_p/ \rangle$, $head(C_1) = \langle -m_1 \ /a_1/ \rangle$, $head(C_2) = \langle -m_2 \ /a_2/ \rangle$, $\cdots$, $head(C_{k-1}) = \langle -m_{k-1} \ /a_{k-1}/ \rangle$
3. $C_k \lhd C_{k+1} \lhd \cdots \lhd C_p \lhd C_1 \lhd C_2 \lhd \cdots \lhd C_{k-1}$
4. *pour $i \in [p]$, pour tout u de C_i n'appartenant pas à C_{i+1} et u différent de $head(C_i)$ alors $u = \langle /a_i/ \rangle$*

Si $Q_k(t_p)$ est vérifiée, l'utilisation d'une entrée lexicale de ϕ_{iter} permet de vérifier $Q_{k-1}(t_p)$.

Démonstration. On suppose Q_k vérifiée par t. On a donc $head(t_p) = \langle s_k \ -m_k \ /a_k/ \rangle$, D'après la formation du lexique, la seule entrée lexicale de $\phi_{iter}(Lex_p)$ telle que le premier trait est un sélecteur $=s_k$ est : $\langle =s_k \ +m_{k-1} \ s_{k-1} \ -m_{k-1}/a_{k-1}/ \rangle$. On la fusionne avec t_p pour donner t'_p. On a alors $head(t'_p) = \langle +m_{k-1} \ s_{k-1} \ -m_{k-1}/a_{k-1}/ \rangle$ ce qui ajoute une forme phonologique $/a_{k-1}/$ et $head(C_k) = \langle -m_k/a_k/ \rangle$.

D'après Q_k, on sait qu'il existe C_{k-1} tel que $head(C_{k-1}) = \langle -m_{k-1}/a_{k-1}/ \rangle$. Un déplacement est donc réalisé, faisant passer en début de l'arbre C_{k-1}. On a alors $head(t''_p) = \langle s_{k-1} \ -m_{k-1}/a_{k-1}/ \rangle$ et l'ancienne tête de C_{k-1} ne contient plus de trait syntaxique.

Donc quelle que soit u une feuille de t''_p telle que u n'appartient pas à C_k et différente de $head(t''_p)$, u ne contient plus de trait syntaxique. On a alors un nouveau C_{k-1} qui est t''_p et qui vérifie :

1. $head(t''_p) = head(C_{k-1})$
2. $head(C_{k-1}) = \langle s_{k-1} -m_{k-1} \ /a_{k-1}/ \rangle$, $head(C_k) = \langle -m_k \ /a_k/ \rangle$, $\ldots$, $head(C_p) = \langle -m_p \ /a_p/ \rangle$, $head(C_1) = \langle -m_1 \ /a_1/ \rangle$, $head(C_2) = \langle -m_2 \ /a_2/ \rangle$, $\ldots$, $head(C_{k-2}) = \langle -m_{k-2} \ /a_{k-2}/ \rangle$
3. $C_{k-1} \lhd C_k \lhd C_{k+1} \lhd \cdots \lhd C_p \lhd C_1 \lhd C_2 \lhd \cdots \lhd C_{k-2}$
4. pour $i \in [p]$, pour tout u de C_i n'appartenant pas à C_{i+1} et différent de $head(C_i)$ alors $u = \langle /a_i/ \rangle$,

c'est-à-dire $Q_{k-1}(t''_p)$ dans lequel il y a une forme phonologique $/a_{k-1}/$ de plus que dans t_p. □

Lemme 59. *Soit t_p un arbre tel que $P(t_p)$ est vraie. Le passage de t_p par une phase d'itération construit t'_p tel que $P(t'_p)$ est vérifiée.*

Démonstration. Par induction sur p.

– si $p = 1$, d'après le lemme 57

1. $head(t) = head(C_1)$,
2. $head(C_1) = \langle s_1\ -m_1\ /a_1/\rangle$
3. C_1,
4. $\forall u \in C_1, u \neq head(C_1)$ alors $u = \langle /a_1/\rangle$

et la phase d'itération est réduite à une fusion avec une entrée de type 4 : : $\langle =s_1\ +m_1\ s_1\ -m_1\ /a_1/\rangle$. La nouvelle tête induit un déplacement déclenché par $+m_1$ et réalisé par l'entrée $-m_1/a_1/$, tête de C_1 obtenue après la fusion précédente. On obtient t'' tel que $head(t''_i) = \langle\ s_1\ -m_1\ /a_1/\rangle$ et les traits syntaxiques de l'ancienne tête de C_1 sont tous effacés. L'arbre obtenu conserve les propriétés de C_1 avec une forme phonologique supplémentaire. La propriété 56 est vérifiée pour t_{i+1}.

– si $p \neq 1$: si la propriété 56 est vérifiée sur t_i la $i^{ème}$ itération, on a :

1. $head(t) = head(C_1)$,
2. $head(C_1) = \langle s_1\ -m_1\ /a_1/\rangle$, $head(C_2) = \langle -m_2\ /a_2/\rangle$, $\cdots$, $head(C_p) = \langle -m_p\ /a_p/\rangle$
3. $C_1 \lhd \ldots \lhd C_{p-1} \lhd C_p$,
4. pour $j \in [p]$, pour tout u de C_j n'appartenant pas à C_{j+1} et différent de $head(C_j)$ alors $u = \langle /a_j/\rangle$

Dans ϕ_{iter}, seule l'entrée $\langle =s_1\ +m_p\ s_p\ -m_p\ /a_p/\rangle$ commence par un sélecteur $=s_1$. On fusionne donc cette entrée avec la dérivation précédente.
Puis la dérivation se poursuit par un déplacement de C_p. L'ancienne tête de C_p ne possède plus de trait syntaxique et $head(t''_i) = \langle s_p\ -m_p\ /a_p/\rangle$. On a alors un nouveau C_p possédant une forme phonologique $/a_p/$ de plus. Ainsi :

1. $head(t) = head(C_p)$
2. $head(C_p) = \langle s_p\ -m_p\ /a_p/\rangle$, $head(C_1) = \langle -m_1\ /a_1/\rangle$, $head(C_2) = \langle -m_2\ /a_2/\rangle$, $\cdots$, $head(C_{p-1}) = \langle -m_{p-1}\ /a_{p-1}/\rangle$
3. $C_p \lhd C_1 \lhd C_2 \lhd \cdots \lhd C_{p-1}$,
4. pour $j \in [p]$, pour tout u de C_j n'appartenant pas à C_{j+1} et différent de $head(C_j)$ alors $u = \langle /a_j/\rangle$ qui correspond à $Q_p(t''_i)$.

D'après le lemme 58 on peut donc passer à Q_{p-1}. Or par hypothèse d'induction à partir de Q_{p-1} on obtient t_{i+1} vérifiant la propriété 56 avec un C_p supplémentaire, minimal pour l'inclusion des C_i (car au-dessus de C_{p-1} avant l'induction).
On remarquera que l'induction est en réalité l'utilisation de $p-1$ fois le lemme 58 à partir de l'ensemble des entrées de type 3. Chacune d'elles introduisant une nouvelle forme phonologique correspondant à son indice, à la fin d'une phase d'itération, on obtient p nouvelles formes phonologiques réparties dans les p C_i, $i \in [p]$.

□

Lemme 60. *Soit t_i, un arbre obtenu après i itérations. La phase de conclusion accepte la phrase $a_1^{i+1} \ldots a_p^{i+1}$.*

Démonstration. Si t_i est obtenu après i itérations valides, on a alors $P(t_i)$ vraie. D'où $head(t_i) = \langle s_1\ -m_1\ /a_1/\rangle$.

Par induction sur p :

– si $p = 1$, on a alors :
  1. $head(t_i) = head(C_1)$
  2. $head(C_1) = \langle s_1\ {-m_1}\ /a_1/\rangle$
  3. C_1
  4. il n'existe pas $u \in C_1$ tel que $u \neq head(C_i)$.

  De plus, la phase de conclusion contient uniquement l'entrée de type 6 $\langle {=s_1}\ +m_1\ c\rangle$. On peut fusionner cette entrée avec t_i. On obtient $t_{conc} =< (+m_1\ c, t'_i)$ où $head(t'_i) = \langle -s_1\ /a_1/\rangle$. Un déplacement est alors réalisé entraînant $head(t'_i) = \langle /a_1/\rangle$ et $head(t''_i) = \langle c\rangle$. On a donc pour toute feuille u de C_1, $u = \langle /a_1/\rangle$. Tous les traits syntaxiques ont été annulés, à l'exception de la tête de l'arbre qui porte le trait acceptant. L'analyse est terminée et acceptée.
  De plus, la phase d'initialisation introduit un a_1 et lors de chaque phase d'itération, une et une seule entrée portant une forme phonologique $/a_1/$ est utilisée. Donc t_i contient exactement $i+1$ formes phonologiques $/a_1/$, d'où la phrase reconnue : a_1^{i+1}
– si $p \neq 1$. On suppose que Lex_{p-1} permet de reconnaître $a_1^{i+1} \cdots a_{p-1}^{i+1}$ après i itérations. La phase de conclusion est composée d'une unique entrée. Celle de Lex_p est la même que celle de Lex_{p-1} avec un assignateur de plus correspondant à a_p en première position de la suite des assignateurs. La dérivation commence donc par une fusion entre cette entrée et t'_i. On a alors :
  1. $head(t'_{conc}) = \langle +m_p \cdots +m_1\ c\rangle$,
  2. $head(C_1) = \langle -m_1\ /a_1/\rangle, head(C_2) = \langle -m_2\ /a_2/\rangle, \cdots, head(C_p) = \langle -m_p\ /a_p/\rangle$
  3. $C_1 \lhd \ldots \lhd C_p$,
  4. pour $i \in [p]$, pour tout u de C_i n'appartenant pas à C_{i+1} et différent de $head(C_i)$ alors $u = \langle /a_i/\rangle$

  On déplace alors C_p, ce qui supprime l'assigné de sa tête. On a donc quelle que soit une feuille u, $u \in C_p, u = \langle /a_p/\rangle$. Or chaque passage dans une phase d'itération utilise une et une seule entrée portant une forme phonologique $/a_p/$, donc C_p contient exactement $i+1$ $/a_p/$ qui sont positionnées en première position de la dérivation. On obtient alors les propriétés suivantes :
  1. $head(t) = \langle +m_{p-1} \ldots +m_1\ c\rangle$,
  2. $head(C_1) = \langle -m_1\ /a_1/\rangle$, $head(C_2) = \langle -m_2\ /a_2/\rangle$, $\ldots$, $head(C_{p-1}) = \langle -m_{p-1}\ /a_{p-1}/\rangle$
  3. $C_{p-1} \lhd \cdots \lhd C_1$,
  4. pour $i \in [p-1]$, pour tout u de C_i n'appartenant pas à C_{i+1} et différent de $head(C_i)$ alors $u = \langle /a_i/\rangle$

  Par hypothèse d'induction, chaque groupe de terminaux passe en première position pour donner une analyse acceptante. On obtient $a_1^{i+1} \cdots a_{p-1}^{i+1}$ qui vont se positionner devant a_p^{i+1}, d'où le résultat.

□

Lemme 61. *Dans une phase d'initialisation, l'utilisation d'une entrée de ϕ_{iter} fait immédiatement échouer l'analyse.*

Démonstration. Soit $t \in T_{GM}$.

La phase d'initialisation a pour but d'introduire des marqueurs pour chaque terminal. La phase d'itération utilise ces marqueurs pour créer des groupes de terminaux homogènes. L'utilisation d'une entrée de ϕ_{iter} dans la phase d'initialisation doit alors déplacer un groupe de terminaux dont le marqueur n'est pas encore introduit. La dérivation ne peut pas se poursuivre.

On suppose que la phase d'initialisation est correcte pour les k premières fusions. D'après la structure des entrées de ϕ_{init}, on a $head(t) = \langle s_{p-k}\ -m_{p-k}\ /a_{p-k}/\rangle$ et quelle que soit $u \in S_t$, $u = \langle -m_i/a_i/\rangle$ pour $i \in [p-k+1, p]$. L'arbre (a) de la figure 28 présente graphiquement cet arbre. Pour la $(k+1) - ième$ fusion on choisit d'utiliser un entrée de ϕ_{iter}. Seule $\langle = s_{p-k} + m_{p-k-1} s_{p-k-1}\ -m_{p-k-1}\ /a_{p-k-1}/\rangle$ permet de déclencher une fusion. L'arbre obtenu est l'arbre (b) de la figure 28 La tête du nouvel arbre obtenu t' est :

$$head(t') = \langle + m_{p-k-1} s_{p-k-1}\ -m_{p-k-1}\ /a_{p-k-1}/\rangle$$

Or $(p-k-1) \notin [p-k, k]$ donc il n'existe pas dans t' de feuille dont le premier trait soit l'assigné équivalent. Le déplacement demandé par $+m_{p-k-1}$ ne peut être réalisé et l'analyse échoue. □

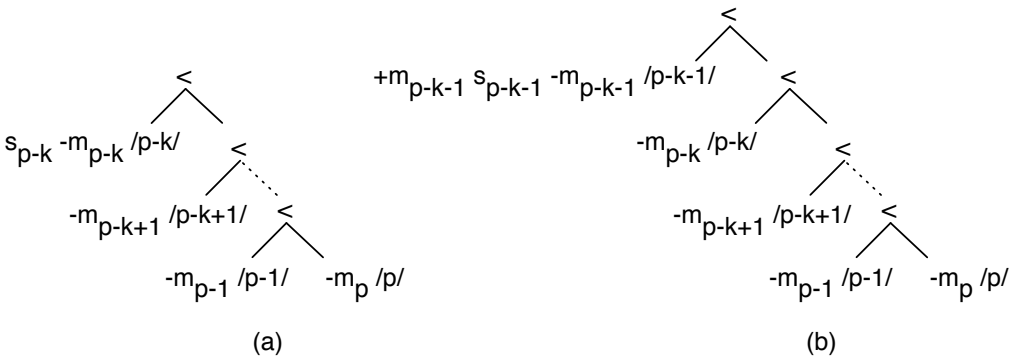


FIG. 28 – Premier cas d'échec dans l'analyse de phrase sur un compteur.

Lemme 62. *Dans une phase d'itération, l'utilisation d'une entrée de ϕ_{init} fait échouer la dérivation.*

Démonstration. Soit $t \in T_{MG}$. On note $|t|_h$ le nombre d'occurrences du trait h dans t.

Chaque passage par la phase d'initialisation au lieu de la phase d'itération implique qu'un assigné n'est pas effacé et un autre de même type est introduit. Or, seule la phase de conclusion permet d'effacer un assigné pour chaque marqueur. L'analyse ne sera pas acceptée car encore porteuse de traits syntaxiques autres que le trait acceptant.

On suppose que le passage par ϕ_{init} provient de l'utilisation de l'entrée de ϕ_{init} : $\langle = s_k\ s_{k-1}\ -m_{k-1}\ /a_{k-1}/\rangle$. On a alors $head(t) = \langle s_{k-1}\ -m_{k-1}\ /a_{k-1}/\rangle$. La dérivation se poursuit avec une fusion qui effacera s_{k-1} (que ce soit par l'utilisation de l'entrée correspondante de ϕ_{init} ou de ϕ_{iter}). Il existe donc deux feuilles $\langle -m_{k-1}\ /a_{k-1}/\rangle$, d'où

$|t_{-m_{k-1}}| \geq 2$. Le déplacement nécessaire n'étant pas réalisé, la propriété 56 n'est plus vérifiée.

Il existe deux entrées lexicales de cette grammaire portant l'assignateur $+m_{k-1}$:

1. $\langle =s_k \ +m_{k-1} \ s_{k-1} \ -m_{k-1} \ /a_{k-1}/ \rangle$ qui appartient à ϕ_{iter}
2. $\langle =s_1 \ +m_p \ \cdots \ +m_{k-1} \ \cdots c \rangle$ qui appartient à ϕ_{conc}

On note d (*resp.* d') l'arbre de dérivation obtenu avant (*resp.* après) l'utilisation de l'une de ces entrées.

La première entrée, en plus d'effacer un $-m_{k-1}$, en ajoute un nouveau à la dérivation. Donc pour toute utilisation de cette entrée $|d'|_{-m_{k-1}} = |d|_{-m_{k-1}}$ Cette entrée peut être utilisée un nombre indéfini de fois. Cependant, elle ne permet pas de faire diminuer le nombre de $-m_{k-1}$ dans la dérivation.

La seconde permet d'effacer un unique $-m_{k-1}$, donc $|d'|_{-m_{k-1}} = |d|_{-m_{k-1}} - 1$. Cette entrée ne peut être utilisée qu'une unique fois.

Nous avons vu que le passage par ϕ_{init} impliquait que le nombre de $-m_{k-1}$ était supérieur ou égal à 2. Or d'après les remarques ci-dessus, le nombre de $-m_{k-1}$ ne peut que diminuer de 1.

De plus, pour qu'une dérivation soit acceptante, elle ne doit pas contenir d'autre trait que le trait acceptant et $-m_{k-1}$ n'est pas le trait acceptant. Donc la dérivation est rejetée.

On remarque que si la SMC est utilisée, la phase de conclusion fera directement échouer la dérivation à cause de la présence de deux feuilles portant le même assigné en première position. □

Démonstration du théorème 55. D'après le lemme 57, la phase d'initialisation permet de vérifier la propriété 56. D'après le lemme 59, chaque phase d'itération conserve la propriété 56. D'après le lemme 60, après n itérations, on reconnaît la phrase $a_1^{n+1} \cdots a_p^{n+1}$.

Le langage $L_p = \{a_1^n \cdots a_p^n | n \in \mathbb{N}\}$ est reconnu par une GM G_p obtenue à partir de l'algorithme 51. De plus, G_p reconnaît exactement ce langage : en effet les seuls choix non déterministes lors de l'analyse sont :

- l'utilisation d'une entrée de la phase d'itération lors de l'initialisation, or le lemme 61 montre que ces analyses sont immédiatement rejetées.
- l'utilisation d'une entrée de la phase d'initialisation lors de l'itération, or le lemme 62 montre que ces analyses sont rejetées en fin de dérivation.

□

4.3 Compteurs enchâssés

Les langages de compteurs enchâssés sont des langages contextuels. Ils sont la généralisation des langages précédents. On montre ici qu'ils sont des langages minimalistes.

Définition 63. *On appelle compteurs enchâssés $CE_{(k,p)}$, l'ensemble des langages contenant k compteurs sur p terminaux ayant des dépendances croisées.*

Par exemple

$$CE_{(2,4)} = \{a_1^{k_1} a_2^{k_2} a_3^{k_1} a_4^{k_2}, \forall k_1, k_2 \in I\!N\}$$

Bien que ces constructions semblent peu probables en langues naturelles, Joshi [Jos85] et Shieber [Sch85] ont montré qu'elles existent.

Pour chaque compteur, noté k_i, $i \in [k]$, on suppose l'existence de p_i terminaux. On a alors $\sum_{i\in[k]} p_i = p$. On note pour $i \in [k]$ et $j \in [p_i]$, $a_{i,j}$ le $j^{ème}$ terminal du $i^{ème}$ compteur. Chaque terminal de la phrase possède une notation sur deux indices montrant sa dépendance à l'un des compteurs et une seconde notation sur un unique indice indiquant sa position dans la phrase. Cette distinction est nécessaire pour la présentation de l'algorithme suivant. De plus, pour dissocier les différents types de traits nous notons $s_{i,j}$ les traits de fusion relatifs à $a_{i,j}$ et $m_{i,j}$ les traits de déplacement relatifs à $a_{i,j}$.

On appelle *état pré-final de la dérivation* l'état de la dérivation avant le passage dans la phase de conclusion selon l'algorithme des compteurs sur N terminaux. On marquera l'arrivée dans l'état pré-final par un symbole représentant le compteur particulier, ici n_i.

Algorithme 64. *Construction du lexique pour les compteurs enchâssés.*
On suppose les p formes phonologiques ordonnées $a_1 < \ldots < a_p$.
pour i de 1 à k (pour chaque compteur)

- $\langle s_{i,p_i}\ -m_{i,p_i}\ /a_{i,p_i}/\rangle$
- *pour j de 1 à* (p_i - 1)
  $\langle =s_{i,j+1}\ s_{i,j}\ -m_{i,j}\ /a_{i,j}/\rangle$
- *pour k de 1 à* (p_i-1)
  $\langle =s_{i,k+1}\ +m_{i,k}\ s_{i,k}\ -m_{i,k}\ /a_{i,k}/\rangle$
- $\langle =s_{i1}\ +m_{i,p_i}\ s_{i,p_i}\ -m_{i,p_i}\ /a_{i,p_i}/\rangle$
- $\langle c\rangle$
- $\langle =s_{i,1}\ n_i\rangle$

$= n_1 \cdots = n_k + m_1 \cdots + m_p\ c$

Synopsis de Dérivation 65. *L'algorithme se divise en deux étapes :*

1. ***Initialisation :*** *L'algorithme 64 fait appel à l'algorithme 51 pour chacun des compteurs en modifiant la phase de conclusion. Elle se contente d'attribuer à l'analyse le trait représentant le compteur qu'elle vient de modéliser sans effectuer les déplacements de la phase de conclusion. Chaque dérivation est dans l'état pré-final.*
2. ***Conclusion :*** *après avoir utilisé k fois cette version modifiée de l'algorithme 51, l'analyse entre dans la phase de conclusion qui commence par fusionner dans un même arbre les k arbres précédents. Une fois toutes les sous-dérivations mises en commun, la tête possède une suite d'assignateurs correspondant à chaque terminal dans l'ordre supposé pour l'algorithme. Cette phase réordonne alors les groupes correspondant à chaque terminal et permet de reconnaître la phrase attendue.*

À nouveau, on note ϕ_{init} (*resp.* ϕ_{conc}) les entrées relatives à la phase d'initialisation (*resp.* la phase de conclusion).

Un exemple de compteurs enchâssés sur $CE_{(2,4)}$

Soit $CE_{(2,4)} = \{a_1^{k_1}a_2^{k_2}a_3^{k_1}a_4^{k_2}, \forall k_1, k_2 \in \mathbb{N}\}$.
Nous allons analyser la phrase $\omega = 1^1 2^2 3^1 4^2$.
Le lexique suivant permet de reconnaître les phrases de $CE_{(2,4)}$.

Lexique 66.

lexique construisant $1^{n_1}3^{n_1}$

$type1 \quad s_3 \ -m_3 \ /3/$	$type4 \quad = s_1 \ +m_3 \ s_3 \ -m_3/3/$
$type2 \quad = s_3 \ s_1 \ -m_1 \ /1/$	$type5 \quad n_1 \ c$
$type3 \quad = s_3 \ +m_1 \ s_1 \ -m_1 \ /1/$	$type6 \quad = s_1 \ n_1$

lexique construisant $2^{n_2}4^{n_2}$

$type7 \quad s_4 \ -m_4 \ /4/$	$type10 \quad = s_2 \ +m_4 \ s_4 \ -m_4 \ /4/$
$type8 \quad = s_4 \ s_2 \ -m_2 \ /2/$	$type11 \quad n_2 \ c$
$type9 \quad = s_4 \ +m_2 \ s_2 \ -m_2 \ /2/$	$type12 \quad = s_1 \ n_2$

phase de conclusion

$type13 \quad = n_1 \ = n_2 \ +m_4 \ +m_3 \ +m_2 \ +m_1 \ c$	

On peut représenter le lexique par un circuit, présenté par la figure 29.

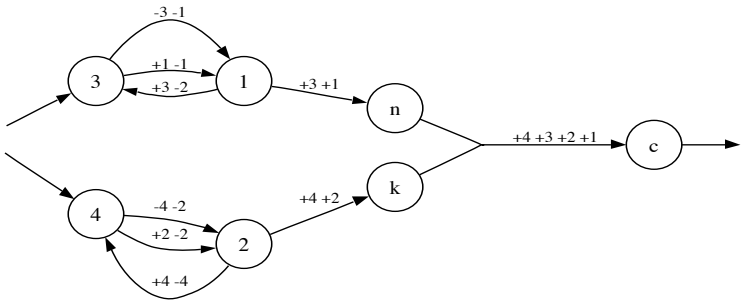


FIG. 29 – Représentation du lexique pour un compteur et deux terminaux.

Première partie de la dérivation, construction sur le premier compteur. On donne le même exemple que dans la partie 4.2, pour $n = 2$ et les terminaux $/1/$ et $/3/$ tels que $/1/ < /3/$.

1. entrée lexicale de type 1 : $s_3 \ -m_3 \ /3/$

   et de type 2 : $= s_3 \ s_1 \ -m_1 \ /1/$

2. fusion : $\underline{\mathbf{s_1} \ \mathbf{-m_1} \ \mathbf{/1/}, \ -m_3/3/}$

3. passage dans l'état pré-final : entrée lexicale de type 6 : $= s_1 n_1$

   et fusion : $\underline{\mathbf{n_1}, \underline{\mathbf{-m_1/1/}}, \ -m_3/3/}$

Deuxième partie de la dérivation, construction sur le second compteur.

1. entrée lexicale de type 7 : $s_4 \ -m_4 \ /4/$

   et de type 8 : $= s_4 \ s_2 \ -m_2 \ /2/$

2. fusion :

$$\underline{\mathbf{s_2}\ -\mathbf{m_2}\ /\mathbf{2}/,\ -m_4\ /4/}$$

3. Passage par une itération via une entrée lexicale de type 10 :

$$= s_2\ + m_4\ s_4\ - m_4\ /4/$$

et une fusion avec l'élément précédent : $\underline{+\mathbf{m_4}\ \mathbf{s_4}\ -\mathbf{m_4}\ /\mathbf{4}/,\ \underline{-\mathbf{m_2}\ /\mathbf{2}/,\ -m_4\ /4/}}$

4. Déplacement : $\underline{/4/,\ \mathbf{s_4}\ -\mathbf{m_4}\ /\mathbf{4}/,\ -m_2\ /2/}$

5. entrée lexicale de type 9 : $= s_4\ + m_2\ s_2\ - m_2\ /2/$

et une fusion $\underline{+\mathbf{m_2}\ \mathbf{s_2}\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{/4/,\ -\mathbf{m_4}\ /\mathbf{4}/,\ -m_2\ /2/}}$

6. Déplacement : $\underline{/2/,\ \mathbf{s_2}\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{/4/,\ -\mathbf{m_4}\ /\mathbf{4}/}}$

7. Passage dans la phase de conclusion, entrée lexicale de type 12 : $= s_2\ n_2$

et fusion : $\underline{\mathbf{n_2},\ \underline{/2/,\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{/4/,\ -\mathbf{m_4}\ /\mathbf{4}/}}}$

Mise en commun et fin de la dérivation.

Nous avons construit deux analyses distinctes sur chacun des compteurs. Il s'agit maintenant de les mettre en commun puis de réordonner chaque terminal.

1. Entrée lexicale de type 13 : $= n_1\ = n_2\ + m_4 + m_3 + m_2 + m_1 c$

Résultat de la première partie de la dérivation : $\underline{\mathbf{n_1},\ \underline{-\mathbf{m_1}\ /\mathbf{1}/,\ -m_3\ /3/}}$

fusion $\underline{= \mathbf{n_2}\ + \mathbf{m_4}\ + \mathbf{m_3}\ + \mathbf{m_2}\ + \mathbf{m_1}\ \mathbf{c}, \underline{-\mathbf{m_1}\ /\mathbf{1}/,\ /3/,\ -m_3/3/}}$

2. Résultat de la deuxième partie de la dérivation : $\underline{\mathbf{n_2},\ \underline{/2/,\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{/4/,\ -\mathbf{4}/\mathbf{4}/}}}$

fusion :

$$\underline{+\mathbf{m_4}\ + \mathbf{m_3}\ + \mathbf{m_2}\ + \mathbf{m_1}\ \mathbf{c}, \underline{-\mathbf{m_1}/\mathbf{1}/,\ -m_3\ /3/,\ \underline{/2/,\ -\mathbf{m_2}\ /\mathbf{2}/,\ \underline{/4/,\ -\mathbf{m_4}/\mathbf{4}/}}}}$$

3. Phase de réordonnancement : déplacements successifs de chaque terminal :

$$\underline{/4/,\ /4/,\ +\mathbf{m_3}\ + \mathbf{m_2}\ + \mathbf{m_1}\ \mathbf{c}, \underline{-\mathbf{m_1}\ /\mathbf{1}/,\ -m_3\ /3/,\ \underline{/2/,\ -\mathbf{m_2}\ /\mathbf{2}/}}}$$

4. Déplacement du groupe de /3/ :

$$\underline{/3/,\ /4/,\ /4/, +\mathbf{m_2}\ + \mathbf{m_1}\ \mathbf{c}, -m_1\ /1/, \underline{/2/,\ -\mathbf{m_2}\ /\mathbf{2}/}}$$

5. Déplacement du groupe de /2/ : $\underline{/2/,\ /2/,\ /3/,\ /4/,\ /4/, +\mathbf{m_1}\ \mathbf{c}, -m_1/1/}$

6. Enfin, déplacement du groupe de /1/ : $\underline{/1/,\ /2/,\ /2/,\ /3/,\ /4/,\ /4/, \mathbf{c}}$

Théorème 67. *Les compteurs enchâssés sont des langages minimalistes.*

Démonstration. La preuve est une conséquence du théorème 55.

Après chaque phase de conclusion intermédiaire, la propriété 56 est vérifiée. Les traits de chaque phase de conclusion sur chaque compteur sont reportés dans la phase finale de conclusion et les traits n_i ajoutés dans les phases de conclusion intermédiaires sont également effacés par la conclusion générale (présence de sélecteur $= n_i$).

Lors d'une dérivation, on commence par construire une sous-dérivation sur chaque compteur. Cette partie se fait en utilisant l'algorithme 51 avec une modification de la phase de conclusion qui se contente d'introduire un trait représentant le compteur, au lieu de réordonner tous les terminaux de la phrase.

$$\underbrace{=s_{i,1}\ +m_{i,1}\cdots+m_{i,p_i}\ c}\Rightarrow\ \underbrace{=s_{i,1}\ n_i}$$

À la fin de cette phase, la propriété 56 est vérifiée pour cette dérivation. Nous aurons ainsi n sous-phrases sur n compteurs, où chaque terminal peut être déplacé dans la suite de la dérivation - c'est-à-dire présence d'un assigné $-m_i$ en tête de chaque groupe d'un même terminal.

De plus, les traits supprimés de chaque phase de conclusion intermédiaire sont reportés dans la phase de conclusion générale. La phase de conclusion est constituée d'une seule entrée lexicale contenant tous les assignateurs que nous devions reporter, le trait acceptant c (factorisé de tous les reports) et une suite de sélecteurs permettant de rassembler toutes les sous-dérivations.

$$\forall i\ \in [k] : +m_{i,1}\cdots+m_{i,p_i}\ c \Rightarrow =n_1\cdots=s_k\ +m_1\cdots+m_p\ c$$

En effet, la phase de conclusion utilise sur chacune des dérivations le trait qui a remplacé les assignés et permet de rassembler la sous-dérivation particulière avec ce bloc.

Une fois cette étape terminée, chaque terminal est réordonné selon son ordre dans la phrase, de manière analogue au lemme 60.

Tous les traits intermédiaires ajoutés dans les sous-dérivations sont consommés et tous les traits disparus des sous-dérivations sont reportés en phase de conclusion. Avant la phase de conclusion générale, l'ensemble de la propriété 56 est vérifiée sur chaque sous-ensemble et comme la phase de conclusion du théorème 55 utilisait cette propriété, la conclusion les utilise. On a donc $CE_{(k,p)} \subseteq L(G)$.

De plus, d'après le théorème 55, la partie sur chaque compteur dérive exactement la sous-phrase attendue. Toute autre dérivation soit fait immédiatement échouer la dérivation (conséquence du lemme 61), soit introduit un assigné qui ne pourra pas être effacé (conséquence de lemme 62). Les étapes supplémentaires sont entièrement déterministes, donc seules ces dérivations sont acceptées. On a donc $L(G) \subseteq CE_{(k,p)}$, d'où $L(G) = CE_{(k,p)}$. □

4.4 Duplications

Nous avons étudié les principaux langages permettant de définir les langages faiblement sensibles au contexte et les langages hors-contextes, à l'exception des langages de copie. Nous nous intéressons maintenant aux langages de copie miroir, puis nous proposons une généralisation de la copie.

Duplication inverse

Les langages de copie inverse (ou palindrome), composés d'une phrase et de son image miroir, notée par un $\sim$, sont reconnus par une GM.

Algorithme 68. *Pour toute phrase finie ω sur un alphabet fini Σ de k éléments, on note a_i les éléments de Σ, pour $i \in [k]$.*

- *phase d'initialisation :*
  - *pour i de 1 à k :*
    - $\langle = w_i\ x\ /a_i/\rangle$
    - $\langle w_i$ *-l* $/a_i/\rangle$
  - $\langle = x\ y_2\ -tilde\rangle$
  - $\langle = y_2\ +L\ = x\ y_3\ -mot\rangle$
- *phase d'itération :*
  - $\langle = y_3\ +L\ +tilde\ y_4 - tilde\rangle$
  - $\langle = y_4\ +Mot\ = x\ y_3 - mot\rangle$
- *phase de conclusion :*
  - $\langle = y_3\ +L\ +Tilde\ +Mot\ c\rangle$
  - $\langle = x\ +L\ c\rangle$

Synopsis de Dérivation 69.

La phase d'initialisation *construit pour chaque terminal une sous-structure dont la tête contient une forme phonologique. Son complément contient un assigné de même type représentant la* lettre_copie*. On appelle* lettre/lettre_copie *cette construction. Cette phase se poursuit par la construction d'une dérivation dont la tête porte un marqueur tilde. Il servira à déplacer la copie inverse du mot dans la dérivation. Puis, on ajoute une nouvelle entrée qui déplace la* lettre_copie*, ce qui met en début de dérivation la version initiale du mot. Cette entrée porte le marqueur mot. De plus, elle introduit dans la dérivation le couple lettre/lettre_copie suivant.*

La phase d'itération *est composée de deux entrées. Elle débute par une structure contenant un mot, délimité dans la projection maximale du marqueur mot, d'une copie-miroir de ce mot délimité par la projection maximale du marqueur tilde et d'une construction lettre/lettre_copie. La première entrée fait passer en première position de la dérivation la* lettre_copie *puis la copie-miroir du mot. La* lettre_copie *est la nouvelle dernière lettre du mot miroir. Elle ajoute alors un nouveau marqueur tilde. Sa projection maximale contient alors la copie-miroir du mot augmentée d'une lettre. La seconde entrée déplace le mot. La lettre précédemment ajoutée devient alors la première lettre du mot (car dans la projection maximale du marqueur mot). Enfin, elle intègre une nouvelle construction* lettre/lettre_copie*. La phase d'itération peut alors recommencer avec la même structure.*

La phase de conclusion *positionne la dernière* lettre_copie*, puis le mot-miroir et enfin le mot de départ. On obtient alors un mot reconnu de droite à gauche suivi de son image-miroir. On remarque que l'entrée* $\langle =x\ +L\ c\rangle$ *permet de reconnaître les mots, et leur image-miroir, d'une lettre.*

Exemple de duplication inverse $a_1a_2a_3a_3a_2a_1$

Voici le lexique pour la construction de $\omega\tilde{\omega}$ pour $\omega = a_1a_2a_3$.

Lexique 70.

phase d'initialisation			
type : 1	$=w_1\ x\ /a_1/$	*type* : 2	$w_1\ -l/a_1/$
type : 3	$=w_2\ x\ /a_2/$	*type* : 4	$w_2\ -l\ /a_2/$
type : 5	$=w_3\ x\ /a_3/$	*type* : 6	$w_3\ -l/a_3/$
type : 7	$=x\ y_2-tilde$	*type* : 8	$=y_2\ +L\ =x\ y_3-mot$
phase d'itération			
type : 9	$=y_3\ +L\ +Tilde\ y_4-tilde$	*type* : 10	$=y_4\ +Mot\ =x\ y_3-mot$
phase de conclusion			
type : 11	$=y_3\ +L\ +Tilde+Mot\ c$	*type* : 12	$=x\ +L\ c$

Comme pour les autres présentations, nous donnons une représentation sous forme de circuit de ce lexique par la figure 30.

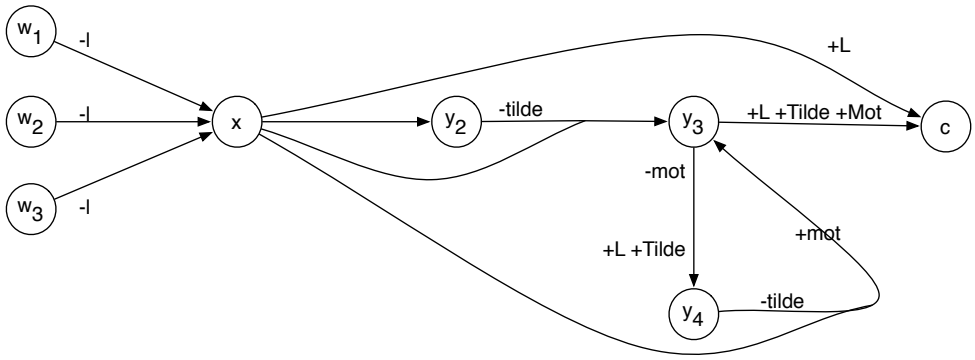


FIG. 30 – Lexique pour la duplication inverse.

Dérivation 71.

1. *entrée de type 1 :* $=w_1\ x\ /a_1/$

   entrée de type 2 : $w_1\ -l\ /a_1/$

   fusion : sous-structure a_1 $\underbrace{x\ /a_1/,\ -l\ /a_1/}$

2. *entrée de type 7 :* $=x\ y_2-tilde$

   fusion :

$y_2\ {-}tilde,\ /a_1/,\ {-}l\ /a_1/$

3. *entrée de type 8 :* $={y_2}\ {+}L\ {=}x\ y_3{-}mot$

   fusion : ${+}L\ {=}x\ y_3\ {-}mot\ ,\ {-}tilde,\ /a_1/,\ {-}l\ /a_1/$

4. *déplacement :* $/a_1/,\ {=}x\ y_3\ {-}mot\ ,\ {-}tilde,\ /a_1/$

5. *entrée de type 3 :* $={w_2}\ x\ /a_2/$

   entrée de type 4 : $w_2\ {-}l\ /a_2/$

   fusion : sous-structure a_2 $x\ /a_2/,\ {-}l\ /a_2/$

6. *fusion item 4 et item 5 :* $/a_2/,\ {-}l\ /a_2/,\ /a_1/,\ y_3\ {-}mot\ ,\ {-}tilde,\ /a_1/$

7. *entrée de type 9 :* $={y_3}\ {+}L\ {+}Tilde\ y_4{-}tilde$

   fusion : ${+}L\ {+}Tilde\ y_4{-}tilde,\ /a_2/,\ {-}l\ /a_2/,\ /a_1/,\ {-}mot\ ,\ {-}tilde,\ /a_1/$

   déplacement : $/a_2/,\ {+}Tilde\ y_4{-}tilde,\ /a_2/,\ /a_1/,\ {-}mot\ ,\ {-}tilde,\ /a_1/$

   déplacement : $/a_1/,\ /a_2/,\ y_4{-}tilde,\ /a_2/,\ /a_1/,\ {-}mot$

8. *entrée de type 10 :* $={y_4}\ {+}Mot\ {=}x\ y_3{-}mot$

   fusion : ${+}Mot\ {=}x\ y_3{-}mot/a_1/,\ /a_2/,\ {-}tilde,\ /a_2/,\ /a_1/,\ {-}mot$

   déplacement : $/a_2/,\ /a_1/,\ {=}x\ y_3{-}mot/a_1/,\ /a_2/,\ {-}tilde,$

9. *entrée de type 5 :* $={w_3}\ x/a_3/$

   entrée de type 6 : $w_3\ {-}l/a_3/$

   fusion : sous-structure a_3 $x\ /a_3/,\ {-}l/a_3/$

10. *fusion item 8 et item 9* $/a_3/,\ {-}l/a_3/,\ /a_2/,\ /a_1/,\ y_3{-}mot/a_1/,\ /a_2/,\ {-}tilde,$

11. *entrée de type 11 :* $={y_3}\ {+}L\ {+}Tilde\ {+}Mot\ c$

    fusion : ${+}L\ {+}Tilde\ {+}Mot\ c,\ /a_3/,\ {-}l/a_3/,\ /a_2/,\ /a_1/,\ {-}mot/a_1/,\ /a_2/,\ {-}tilde,$

    déplacement : $/a_3/,\ {+}Tilde\ {+}Mot\ c,\ /a_3/,\ /a_2/,\ /a_1/,\ {-}mot/a_1/,\ /a_2/,\ {-}tilde,$

    déplacement :

$$\underbrace{/a_1/,\ /a_2/,\ /a_3/,\ +Mot\ c,\ \underbrace{/a_3/,\ /a_2/,\ /a_1/,\ -mot}}$$

déplacement : $\underbrace{/a_3/,\ /a_2/,\ /a_1/,\ /a_1/,\ /a_2/,\ /a_3/,\ c}$

On remarquera que l'obtention de $\tilde{\omega}\omega$ est possible en inversant l'ordre des terminaux de la phrase de départ.

Théorème 72. *Soit Σ un alphabet, L=$\{\omega\tilde{\omega}|\omega \in \Sigma^*\}$ est un langage minimaliste.*

Lemme 73. *Soit $t_p \in T_{MG}$ obtenu après la phase d'initialisation est tel que :*

1. *$head(t_p) = \langle y_3\ -mot\rangle$ tel que $proj_{max}(head(t_p))$ reconnaît $/l_2//l_2//\omega_1//\tilde{\omega}_1/$,*
2. *il existe une feuille $-tilde$ telle que $proj_{max}(-tilde)$ reconnaît $\tilde{\omega}_1$,*
3. *il existe une feuille $-l$ telle que $proj_{max}(-l)$ reconnaît l_2,*
4. *aucune autre feuille ne contient de trait syntaxique.*

Démonstration. La phase d'initialisation de ces dérivations est entièrement déterministe. La succession des opérations produit la structure suivante, dont on peut retrouver la construction dans l'exemple précédent :

$$\underbrace{\underbrace{/a_2/,\ -l\ /a_2/},\ /a_1/,\ y_3\ -mot\ ,\ \underbrace{-tilde,\ /a_1/}}$$

La vérification des propriétés est alors triviale. □

Lemme 74. *Soit $t_p \in T_{MG}$ obtenu après k phases d'itération est tel que :*

1. *$head(t_p) = \langle y_3\ -mot\rangle$ tel que $proj_{max}(head(t_p))$ reconnaît $/l_{k+2}//l_{k+2}//\omega_{k+1}//\tilde{\omega}_{k+1}/$,*
2. *il existe une feuille $-tilde$ telle que $proj_{max}(-tilde)$ reconnaît $\tilde{\omega}_{k+1}$,*
3. *il existe une feuille $-l$ telle que $proj_{max}(-l)$ reconnaît l_{k+2},*
4. *aucune autre feuille ne contient de trait syntaxique.*

Démonstration. On suppose que ces propriétés sont vérifiées pour les $(k-1)$ phases d'itération. Cette phase est déterministe et utilise les deux entrées : $\langle{=}y_3\ {+}L\ {+}tilde\ y_4\ {-}tilde\rangle$ et $\langle{=}y_4\ {+}Mot\ {=}x\ y_3\ {-}mot\rangle$.

La première entrée est introduite par une fusion. Elle devient la tête de la dérivation. L'ancienne tête ne contient alors que le trait $-mot$ dont la projection maximale reconnaît alors $/l_{k+1}//l_{k+1}//\omega_k//\tilde{\omega}_k/$. La dérivation se poursuit alors par deux déplacements successifs. Le premier positionne en haut de la dérivation la lettre suivante du mot, le second la version miroir du mot de départ. Le spécifieur de la tête de la dérivation reconnaît alors $/\tilde{\omega}_k//l_{k+1}/$.

Puis la dérivation est fusionnée avec l'autre entrée de la phase d'itération. L'ancienne tête ne contient alors que le trait $-tilde$. L'opération suivante est un déplacement à partir de $-mot$ qui, à ce moment là, entraîne avec lui $/l_{k+1}//\omega_k/$ qui fait passer en première position le sous-mot reconnu, précédé de la lettre supplémentaire. La projection maximale de la tête reconnaît alors $/l_{k+1}//\omega_k//\tilde{\omega}_k//l_{k+1}/$, c'est-à-dire :

$$/\omega_{k+1}//\tilde{\omega}_{k+1}/$$

La dérivation est ensuite fusionnée avec une sous-structure de la lettre suivante. Cette sous-structure introduit alors deux formes phonologiques de la lettre suivante dont l'une possède l'assigné $-l$ et est sa propre projection maximale, ce qui vérifie la troisième propriété. La tête de la dérivation est alors $\langle y_3\ -mot\rangle$ et reconnaît alors

$$/l_{k+2}//l_{k+2}//\omega_{k+1}//\tilde{\omega}_{k+1}/$$

Le complément de la feuille $-tilde$ est alors vide, sa projection maximale reconnaît donc $/\tilde{\omega}_k//l_{k+1}/$, c'est-à-dire $/\tilde{\omega}_{k+1}/$. Ce qui vérifie la deuxième propriété.

De plus, nous n'avons pas introduit d'autre trait syntaxique. La phase d'itération conserve les propriétés. □

Démonstration du théorème 72. Les propriétés de la phase d'initialisation, lemme 73, puis celles de la phase d'itération, lemme 74 permettent d'obtenir une dérivation reconnaissant : $/l_k//l_k//\omega_{k-1}//\tilde{\omega}_{k-1}/$.

La phase de conclusion se réduit à une fusion avec l'entrée $=y_3\ +L\ +Tilde\ +Mc$, qui est possible grâce à la première propriété. La dérivation se poursuit par un déplacement de la lettre supplémentaire, puis de la version miroir du mot (sans la dernière lettre) et enfin la version originale du mot et de la dernière lettre qui sont encore dans la projection maximale du trait $-mot$. Tous les traits syntaxiques présents à la fin de la phase d'itération sont effacés par la phase de conclusion qui introduit le trait accpetant et reconnaît alors le mot $/\omega//\tilde{\omega}/$.

□

4.5 Duplication multiple

À partir de la duplication inverse, on se pose alors la question de la duplication. On peut aisément proposer un lexique reconnaissant $\omega\omega$, pour une phrase ω sur un alphabet Σ, en inversant l'ordre de déplacement du mot tilde et de la lettre traitée.

De manière analogue, la k-copie peut être reconnue en construisant dans la phase d'initialisation des sous-structures pour chaque terminal contenant m formes phonologiques distinctes. Il s'agit alors de vérifier au fur et à mesure qu'autant de formes phonologiques sont utilisées. Cependant, pour que la procédure soit générique, nous avons décidé de relâcher la SMC comme présenté dans la section 3.2. Les arguments permettant de montrer la production de ce lexique, sont similaires à ceux utilisés dans cette partie.

Algorithme 75. *Généralisation pour toute phrase ω sur un alphabet Σ fini à n éléments. On note a_i, pour $i \in [n]$ les éléments de σ.*

- *phase d'initialisation :*
  - *pour i de 1 à n :*
    - $\langle =w_i\ x\ /a_i/\rangle$
    - $\langle =w_i\ w_i\ -l\ /a_i/\rangle$
    - $\langle w_i\ -l\ /a_i/\rangle$
  - $=x\ z\ -t$
  - $=z\ +L\ z\ -t$
  - $=z\ +L\ y$
- *phase d'itération :*
  - $\langle =y\ \ =x\ z_2\ -t\rangle$
  - $\langle =z_2\ +T\ +L\ z_2\ -t\rangle$
  - $\langle =z_2\ +T\ +L\ y\rangle$
- *phase de conclusion :*
  - $\langle =y\ +T\ y_2\rangle$
  - $\langle =y_2\ +T\ y_2\rangle$
  - $\langle =y_2\ c\rangle$

Synopsis de Dérivation 76.
Construction de ω^k, où $\omega \in \Sigma^*$.

1. *la **phase d'initialisation** permet, d'une part, de construire des sous-structures de k formes phonologiques et d'autre part d'introduire des marqueurs pour les sous-mots. Ainsi, chaque entrée permet d'introduire un marqueur $-t$ (chacun ayant l'un des sous-mots dans se projection maximale).*
2. *la **phase d'itération** est l'ajout d'une structure contenant k formes phonologiques puis le déplacement successif d'une lettre et d'un sous-mot. Ce qui permet d'augmenter chaque sous-mots d'une lettre.*
3. *après k passages dans la phase d'itération, la dérivation efface les marqueurs de sous-mots pour donner la dérivation acceptée.*

On rappelle qu'on note ϕ_{init} (*resp.* ϕ_{iter} et ϕ_{conc}) le sous-ensemble du lexique contenant toutes les entrées dans la phase d'initialisation (*resp.* d'itération et de conclusion)

Exemple de 3-copies : $(cba)^3$

Lexique 77.

phase d'initialisation		
$type : 1 \quad = w_1\ x\ /a/$	$type : 2 \quad = w_1\ w_1\ -l\ /a/$	$type : 3 \quad w_1\ -l/a/$
$type : 4 \quad = w_2\ x\ /b/$	$type : 5 \quad = w_2\ w_2\ -l\ /b/$	$type : 6 \quad w_2\ -l/b/$
$type : 7 \quad = w_3\ x\ /c/$	$type : 8 \quad = w_3\ w_3\ -l\ /c/$	$type : 9 \quad w_3\ -l/c/$
$type : 10 \quad = x\ z\ -t$	$type : 11 \quad = z\ +L\ z\ -t$	$type : 12 \quad = z\ +L\ y$
phase d'itération		
$type : 13 \quad = y = x\ z_2\ -t$	$type : 14 \quad = z_2\ +T\ +L\ z_2\ -t$	$type : 15 \quad = z_2\ +T\ +L\ y$
phase de conclusion		
$type : 16 \quad = y\ +T\ y_2$	$type : 17 \quad = y_2\ +T\ y_2$	$type : 18 \quad = y_2 c$

Nous donnons une représentation sous forme de circuit de ce lexique par la figure 31.

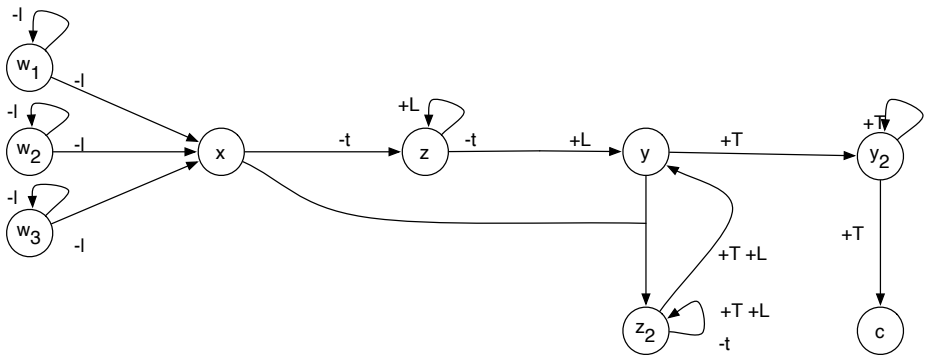


FIG. 31 – Lexique pour la duplication inverse.

Dérivation 78.

1. *entrée de type 2 :* $=w_1\ w_1\ -l\ /a/$
   entrée de type 3 : $w_1\ -l/a/$
   fusion : $\underbrace{w_1\ -l\ /a/,\ -l/a/}$
2. *entrée de type 1 :* $=w_1\ x\ /a/$
   fusion : $\underbrace{x\ /a/,\ -l\ /a/,\ -l/a/}$
3. *entrée de type 10 :* $=x\ z\ -t$
   fusion : $\underbrace{z\ -t,\ /a/,\ -l\ /a/,\ -l/a/}$
4. *entrée de type 11 :* $=z\quad +L\quad z\quad \text{-}t$
   fusion : $\underbrace{+L\quad z\quad \text{-}t,\ \underbrace{-t,\ /a/,\ -l\ /a/,\ -l/a/}}$
   déplacement : $\underbrace{/a/,\ z\quad \text{-}t,\ \underbrace{-t,\ /a/,\ -l\ /a/}}$
5. *entrée de type 12 :* $=z\ +L\ y$
   fusion : $\underbrace{+L\ y,\ \underbrace{/a/,\ \text{-}t,\ \underbrace{-t,\ /a/,\ -l\ /a/}}}$
   déplacement : $\underbrace{/a/,\ y,\ \underbrace{/a/,\ \text{-}t,\ \underbrace{-t,\ /a/}}}$
6. *entrée de type 13 :* $=y\ =x\ z_2\ -t$
   fusion : $\underbrace{=x\ z_2\ -t,/a/,\ \underbrace{/a/,\ \text{-}t,\ \underbrace{-t,\ /a/}}}$
7. *entrée de type 5 :* $=w_2\ w_2\ -l\ /b/$
   entrée de type 6 : $w_2\ -l/b/$
   fusion : $\underbrace{w_2\ -l\ /b/,\ -l/b/}$
8. *entrée de type 4 :* $=w_2\ x\ /b/$
   fusion : $\underbrace{x\ /b/,\ -l\ /b/,\ -l/b/}$
   fusion : $\underbrace{\underbrace{/b/,\ -l\ /b/,\ -l/b/}, z_2\ -t,\ /a/,\ \underbrace{/a/,\ \text{-}t,\ \underbrace{-t,\ /a/}}}$
9. *entrée de type 14 :* $=z_2\ +T\ +L\ z_2\ -t$
   fusion : $\underbrace{+T\ +L\ z_2\ -t,\ \underbrace{/b/,\ -l\ /b/,\ -l/b/}, -t,\ /a/,\ \underbrace{/a/,\ \text{-}t,\ \underbrace{-t,\ /a/}}}$
   déplacement :

$/a/, +L\ z_2\ -t, /b/, -l\ /b/, -l/b/, -t, /a/, /a/, \text{-}t$

déplacement : $/b/, /a/, z_2\ -t, /b/, -l\ /b/, -t, /a/, /a/, \text{-}t$

10. entrée de type 15 : $=z_2\ +T\ +L\ y$

fusion : $+T\ +L\ y, /b/, /a/, -t, /b/, -l\ /b/, -t, /a/, /a/, \text{-}t$

déplacement : $/a/, +L\ y, /b/, /a/, -t, /b/, -l\ /b/, -t, /a/$

déplacement : $/b/, /a/, y, /b/, /a/, -t, /b/, -t, /a/$

11. entrée de type 13 : $=y\ =x\ z_2\ -t$

fusion : $=x\ z_2\ -t, /b/, /a/, /b/, /a/, -t, /b/, z_2\ -t, /a/$

12. entrée de type 8 : $=w_3\ w_3\ -l\ /c/$

entrée de type 9 : $w_3\ -l/c/$

fusion : $w_3\ -l\ /c/, -l/c/$

13. entrée de type 7 : $=w_3\ x\ /c/$

fusion : $x\ /c/, -l\ /c/, -l/c/$

fusion : $/c/, -l\ /c/, -l/c/, z_2\ -t, /b/, /a/, /b/, /a/, -t, /b/, -t, /a/$

14. entrée de type 14 : $=z_2\ +T\ +L\ z_2\ -t$

fusion : $+T\ +L\ z_2\ -t, /c/, -l\ /c/, -l/c/, -t, /b/, /a/, /b/, /a/, -t, /b/, -t, /a/$

déplacement : $/b/, /a/, +L\ z_2\ -t, /c/, -l\ /c/, -l/c/, -t, /b/, /a/, /b/, /a/, -t$

déplacement : $/c/, /b/, /a/, z_2\ -t, /c/, -l\ /c/, -t, /b/, /a/, /b/, /a/, -t$

15. entrée de type 15 : $=z_2\ +T\ +L\ y$

fusion : $+T\ +L\ y, /c/, /b/, /a/, -t, /c/, -l\ /c/, -t, /b/, /a/, /b/, /a/, -t$

déplacement :

$/b/, /a/, +L\ y, /c/, /b/, /a/, -t, /c/, -l\ /c/, -t, /b/, /a/$

déplacement : $/c/, /b/, /a/, y, /c/, /b/, /a/, -t, /c/, -t, /b/, /a/$

16. *entrée de type 16 :* $=y\ +T\ y_2$

fusion : $+T\ y_2, /c/, /b/, /a/, /c/, /b/, /a/, -t, /c/, -t, /b/, /a/$

déplacement : $/c/, /b/, /a/, y_2, /c/, /b/, /a/, /c/, /b/, /a/, -t$

17. *entrée de type 17 :* $=y_2\ +T\ y_2$

fusion : $+T\ y_2, /c/, /b/, /a/, /c/, /b/, /a/, /c/, /b/, /a/, -t$

déplacement : $/c/, /b/, /a/, y_2, /c/, /b/, /a/, /c/, /b/, /a/$

18. *entrée de type 18 :* $=y_2 c$

fusion : $c, /c/, /b/, /a/, /c/, /b/, /a/, /c/, /b/, /a/$

La dérivation reconnaît la phrase $(cba)^3$.

Théorème 79. *Le langage de k-duplication $L = \{\omega^p | \omega \in \Sigma^*, p \in I\!N\}$ est un langage minimaliste.*

Lemme 80. *Soit $t_p \in T_{MG}$ l'arbre de dérivation obtenu après la phase d'initialisation, pour la reconnaissance de $(\omega)^p$, $p \in I\!N$. On note ω_i, $i \in [k]$, le sous-mot de ω de k lettres. t_p est tel que :*

1. *$head(t) = \langle y \rangle$*
2. *t_p contient $(p-1)$ feuilles $\langle -t \rangle$.*
3. *il existe une feuille $u = \langle -t \rangle$ telle que $proj_{max}(u)$ reconnaît $\omega_1 \wedge spec(u) = \epsilon$ et pour toutes les autres feuilles $u = \langle -t \rangle$, $spec(u)$ reconnaît ω_1*
4. *$proj_{max}(y)$ reconnaît ω_1.*

Démonstration. La première partie de la phase d'initialisation est non-déterministe. Elle construit une sous-structure contenant les formes phonologiques de la première lettre. On suppose qu'elle en introduit p. Nous reviendrons dans les preuves suivantes sur le cas où le nombre de formes phonologiques est différent de p. Dans ce cas, la dérivation est déterministe et on obtient :

- $head(t) = \langle x\ /a/ \rangle$
- t_p contient $(p-1)$ feuilles $\langle -l\ /a/ \rangle$.

La dérivation contient alors p formes phonologiques de la première lettre traitée (la dernière du mot). Elle se poursuit alors par une fusion avec la feuille $\langle =x\ z\ -t\rangle$. La tête de la dérivation, que nous appelons v, est alors $\langle z\ -t\rangle$. Nous avons donc introduit un $-t$ pour lequel $spec(\langle z\ -t\rangle) = \epsilon$ et $proj_{max}(\langle z\ -t\rangle)$ reconnaît ω_1^p.

Un choix est alors possible, soit la dérivation est fusionnée avec $\langle =z\ +L\ z\ -t\rangle$, soit avec $\langle =z\ +L\ y\rangle$.

Nous supposons que la dérivation est fusionnée avec l'entrée $\langle =z\ +L\ z\ -t\rangle$. Cette fusion est suivie d'un déplacement de la lettre la plus basse dans la dérivation. De plus le spécifieur de la dérivation est alors ω_1. v devient alors $\langle -t\rangle$ dont le spécifieur est vide et la projection maximale reconnaît ω_1^{p-1}. Si la feuille déplacée n'est pas la plus basse, elle entraîne avec elle une feuille contenant un assigné qui passe en position de spécifieur. À cause de la SPIC, ce trait ne pourra plus être effacé et la dérivation ne pourra pas se terminer.

Dans le cas où la dérivation est d'abord fusionnée avec $\langle =z\ +L\ y\rangle$, nous passons directement à la fin de la phase d'initialisation. La projection maximale de v contient alors au moins une feuille $-l\ /\omega_1/$. Or quelle que soit la suite de la dérivation, la structure des entrées du lexique implique qu'il faille réaliser un déplacement sur le trait $-t$. Soit celui de v est utilisé et dans ce cas, le trait $-l$ passe en position de spécifieur et ne pourra plus être effacé, soit ce n'est pas celui de v mais v est alors déplacé, et, dans le même temps, $-l$ avec lui. Plusieurs traits ne pourront plus être effacés. Dans tous ces cas, la dérivation échoue.

Grâce à cet argument, la dérivation est déterministe et doit utiliser $p-2$ fois l'entrée $\langle =z\ +L\ z\ -t\rangle$. Nous obtenons alors $p-3$ feuilles $u = \langle -t\rangle$ telles que $proj_{max}(u)$ reconnaît $/\omega_1/$ et la tête de la dérivation est alors $\langle z\ -t\rangle$ dont le spécifieur reconnaît $/\omega_1/$. De plus, la projection maximale de la feuille v a perdu $p-3$ forme phonologique $/\omega_1/$. Elle en contient alors deux.

La dérivation est alors non-déterministe. Soit elle utilise une fois de plus la même entrée et consomme le dernier trait $-l$. Dans ce cas, la dérivation ne peut plus sortir de la phase d'initialisation et échoue. Elle doit alors être fusionnée avec l'entrée $\langle =z\ +L\ y\rangle$ permettant de conclure la phase d'initialisation après un déplacement qui utilise le dernier trait $-l$. Dans ce cas la tête de la dérivation est $\langle u\rangle$. Elle contient $(p-1)$ feuilles $\langle -t\rangle$. v reconnaît $/\omega_1/$ et dont le spécifieur est vide. Comme nous l'avons vu, chaque feuille $\langle -t\rangle$ reconnaît dans son spécifieur $/\omega_1/$. Et enfin, le dernier déplacement fait passer une forme phonologique $/\omega_1/$ dans le spécifieur de la dérivation.

Nous avons ainsi vérifié les propriétés de ce lemme. □

Lemme 81. *Soit $t_p \in T_{MG}$ l'arbre de dérivation obtenu après k itérations, pour la reconnaissance de $(\omega)^p$, $p \in I\!N$. t_p est tel que :*

1. *$head(t) = \langle y\rangle$,*
2. *t_p contient $(p-1)$ feuilles $\langle -t\rangle$,*
3. *il existe une feuille $u = \langle -t\rangle$ telle que $proj_{max}(u)$ reconnaît $\omega_{k+1} \wedge spec(u) = \epsilon$ et pour toutes les autres feuilles $u = \langle -t\rangle$, $spec(u) = \omega_{k+1}$,*
4. *$proj_{max}(y)$ reconnaît ω_{k+1}.*

Démonstration. On suppose que les propriétés suivantes sont vérifiées après $k-1$ passages dans la phase d'itération :

1. $head(t) = \langle y \rangle$,
2. t_p contient $(p-1)$ feuilles $\langle -t \rangle$,
3. il existe une feuille $u = \langle -t \rangle$ telle que $proj_{max}(u)$ reconnaît $\omega_k \wedge spec(u) = \epsilon$ et pour toutes les autres feuilles $u = \langle -t \rangle$, $spec(u) = \omega_k$,
4. $proj_{max}(y)$ reconnaît ω_k.

La dérivation est alors fusionnée avec l'entrée $=y\ =x\ z_2\ -t$. La dérivation doit alors introduire une nouvelle sous-structure pour la lettre suivante du mot. Nous remarquons que la structure des entrées lexicales de ce lexique nécessite de déplacer successivement un trait $-t$ (correspondant aux sous-mots) puis un trait $-l$ (correspondant à la lettre suivante). Pour que la dérivation n'échoue pas, la sous-structure doit alors nécessairement contenir autant de formes phonologiques avec un trait $-l$ que de traits $-t$. Elle contient alors p formes phonologiques de la lettre suivante dans le mot. La tête de la dérivation contient dans sa projection maximale p fois $/\omega_k/$.

La dérivation peut être fusionnée :
– soit avec $=z_2\ +T\ +L\ z_2\ -t$,
– soit $=z_2\ +T\ +L\ y$.

La première entrée permet de traiter la phase d'itération. La seconde est celle permettant de sortir de la phase d'itération. Si la dérivation utilise d'abord la seconde phase, les traits $-l$ des formes phonologiques ne sont pas effacés et seront entraînés dans les déplacements suivants, se trouvant en position de spécifieur, ce qui fera échouer la dérivation. Nous devons d'abord utiliser la première entrée.

Dans ce cas, chaque fusion avec cette entrée introduit un trait $-t$ dont le spécifieur contient, par déplacements successifs, une forme $/\omega_k/$ suivie de la lettre suivante dans le mot, c'est-à-dire $/\omega_{k+1}/$. On remarquera que le choix des formes de $/\omega_k/$ ou de la lettre suivante n'est pas déterministe. Cependant, ne pas utiliser les formes les plus basses dans la dérivation entraîne avec avec elles d'autres assignés, qui se retrouvent en position de spécifieur, ce qui fera échouer la dérivation.

De plus, chaque utilisation de cette entrée enlève l'une des formes $/\omega_k/$ de la projection maximale de la première tête de la phase d'itération. Cette entrée doit être utilisée $p-2$ fois, pour laisser le dernier trait $-t$ à l'entrée permettant de conclure cette phase d'itération. Cette dernière entrée permet d'obtenir y comme tête de la dérivation. La première entrée de la phase a introduit un trait $-t$ dont la projection maximale contient $p-(p-2)-1$ formes $/\omega_{k+1}/$. Enfin, la dérivation contient $p-2$ traits $-t$ dont le spécifieur reconnaît $/\omega_{k+1}/$. Ce qui vérifie les propriétés du lemme avec une lettre du mot supplémentaire. □

Démonstration du théorème 79. La phase d'initialisation permet de construire une structure contenant la première lettre traitée. À partir de cette structure, la dérivation enchaîne plusieurs passages par la phase d'itération, permettant d'introduire et de traiter chacune des lettres du mot.

À la fin de la phase d'itération, la dérivation doit alors passer dans la phase de conclusion. Cette phase commence par une fusion avec l'entrée $\langle =y\ +T\ y_2 \rangle$, ce qui efface le trait y, puis par déplacement, la dérivation efface l'un des trait $-t$. À nouveau, le trait à utiliser

est celui qui est le plus bas, sinon, il entraîne avec lui un autre trait $-t$ qui se retrouve en position de spécifieur et, à nouveau, ne pouvant plus être effacé.

La dérivation se poursuit par une fusion :

– soit $\langle =y_2 \ +T \ y_2 \rangle$

– soit $\langle =y_2 \ c \rangle$.

L'utilisation de la dernière entrée fait échouer la dérivation car il reste encore $(p-1)$ traits $-t$ à traiter. Pour cela, l'utilisation de la première entrée permet d'effacer successivement les traits $-t$ restant dans la dérivation.

Une fusion avec $\langle =y_2 \ c \rangle$ permet d'accepter la dérivation qui reconnaît alors p fois le mot $/\omega_k/$. □

4.6 Phrase de Fibonacci

L'intérêt de présenter les dérivations sur les phrases de Fibonacci vient principalement de la notion de copie et d'effacement. Cette étude a servi de base pour la correspondance des GMs et des automates à piles de piles (voir [FS06] pour leur présentation).

Pour une dérivation t, on note $|t|_A$ le nombre de marqueur A dans t.

Synopsis de Dérivation 82.

Cette fois-ci, nous différencierons trois phases :

– ***phase de démarrage :*** *définition de F(0) et F(1), ainsi que leur mise en commun, c'est-à-dire F(2)*

– ***phase de récurrence :*** *On distingue deux parties. Chacune fonctionne sur un marqueur de terminal propre : la première à partir des $-a$ et la seconde des $-b$.*
La récursion est basée sur la décomposition de la suite de Fibonacci :
par définition $F(n+1) = F(n) + F(n-1)$, or $F(n)$ se décompose en somme de $F(n-1)$ et $F(n-2)$. On a donc $F(n+1) = 2 \times F(n-1) + F(n-2)$.
Le problème est donc de faire une copie de $F(n-1)$ à partir de $F(n)$, cette décomposition devant être unifiée par la suite. Ainsi on distingue les marqueurs de $F(n-1)$ de ceux de $F(n-2)$ par les $-a$ et les $-b$.
La figure 32 montre les transformations de la dérivation pour une phase de la récurrence.
La phase commence avec l'arbre (a). La première étape consiste à déplacer tous les marqueurs correspondant à $F(n-2)$, soit l'arbre (b).
Puis, pour chaque marqueur de $F(n-1)$ on les copie/duplique comme dans la procédure de a^{2^n} en utilisant un marqueur spécial pour le trait dupliqué, arbre (c).
Les marqueurs de $F(n-2)$ sont alors en bas de la dérivation, il faut à nouveau les déplacer, arbre (d). Puis, chacun des marqueurs de duplication est copié/déplacé en leur donnant l'appellation correspondant aux marqueurs de $F(n-2)$, soit l'arbre (e).
On obtient alors le rang suivant de la suite de Fibonacci et une inversion de l'ordre des marqueurs, arbre (f). À partir de ce point, on peut soit passer à la phase de conclusion, soit passer dans l'autre phase de la récurrence.
La procédure pour passer à $F(n+2)$ est alors la même avec inversion des marqueurs (deuxième partie de la récurrence).

– ***phase de conclusion :*** *On peut à la fin d'une phase de récurrence passer à la phase de conclusion qui remplace chaque marqueur par le terminal de la phrase.*

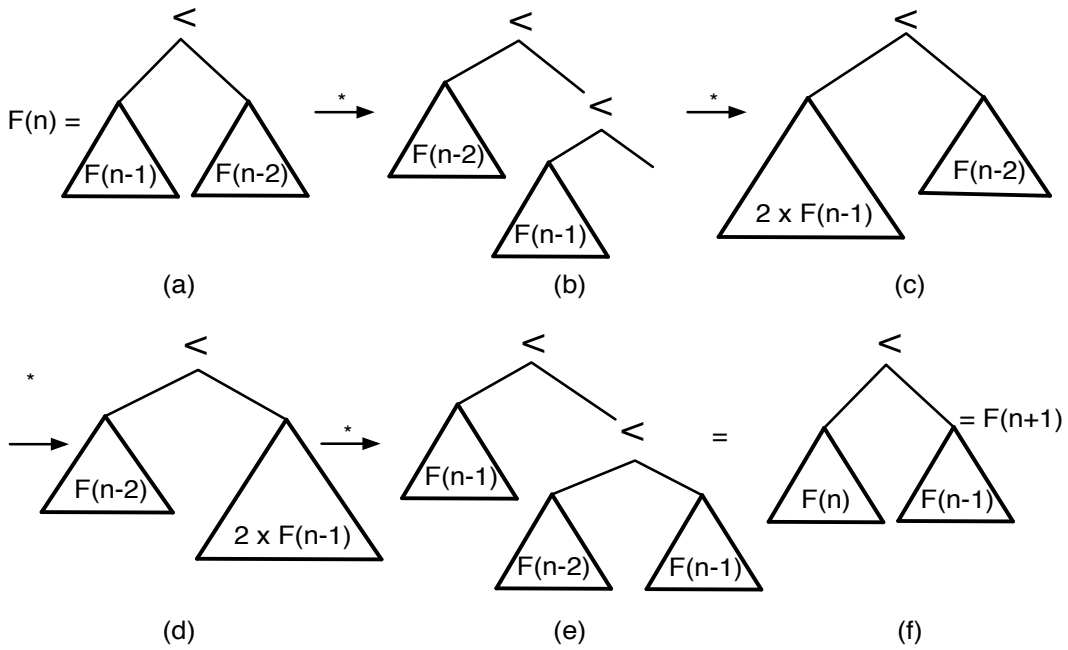


Fig. 32 – Dérivation pour les phrases de Fibonacci.

À présent, voici le lexique qui reconnaît les phrases de Fibonacci.

Lexique 83.

phase de démarrage :			
$type : 0$	$c\ /a/$		
$type : 1$	$= X\ s_1\ -m_a$	$type : 2$	$X\ -m_b$

Puis on distingue deux cas de récurrence en fonction de la parité de n :

récurrence paire :		
	$type : 3$	$= s_{3bis}\ 1$
déplacement des $-m_b$.	$type : 4$	$= 1\ +M_b\ 1\ -m_b$
duplication des A	$type : 5$	$= s_1\ s_2$
	$type : 6$	$= s_2\ +M_a = w_1\ s_2$
	$type : 7$	$= w\ w_1\ -m_a$
	$type : 8$	$w\ -m_{a_d}$
redéplacement des $-m_b$	$type : 9$	$= s_2\ s_1$
remplacement des m_{a_d} *par des* m_b	$type : 10$	$= s_1\ s_3$
	$type : 11$	$= s_3\ +M_{A_D}\ s_3\ -m_b$

récurrence impaire		
passage vers cette phase :		
	type : 12	$= s_3\ s_1bis$
déplacement des $-m_a$	*type* : 13	$= s_{1bis}\ +M_a\ s_{1bis}\ -m_a$
duplication des M_b	*type* : 14	$= s_{1bis}\ s_{2bis}$
	type : 15	$= s_{2bis}\ +M_b = w_2\ s_{2bis}$
	type : 16	$= w_3\ w_2\ -m_b$
	type : 17	$w_3\ -m_{b_d}$
redéplacement des $-m_a$	*type* : 18	$= s_{2bis}\ s_{1bis}$
remplacement des m_{b_d} *par des* m_a	*type* : 19	$= s_{1bis}\ s_{3bis}$
	type : 20	$= s_{3bis}\ +M_{B_D}\ s_{3bis}\ -m_a$

phase de conclusion			
type : 21	$= s_3\ c$	*type* : 22	$= s_{3bis}\ c$
type : 23	$= c\ +M_a\ c\ /a/$	*type* : 24	$= c\ +M_b\ c\ /a/$

Ce lexique est présenté sous forme de circuit dans la figure 33.

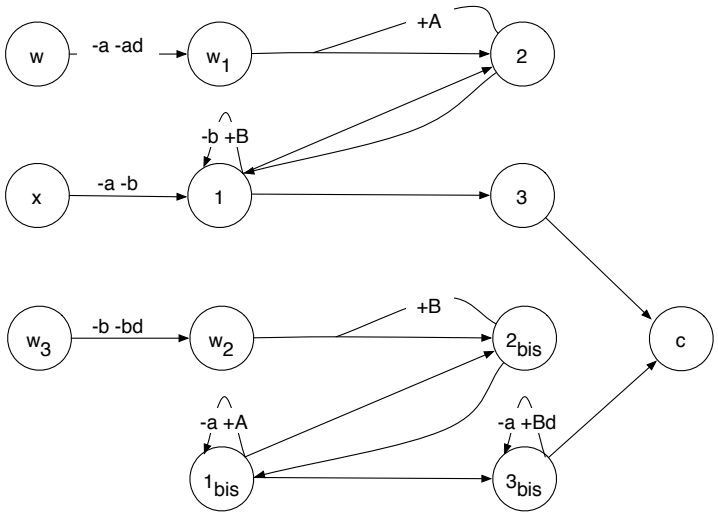


FIG. 33 – Lexique pour $a^{F(n)}$ où $F(n)$ la fonction de Fibonacci.

Exemple : $a^{F(4)} = a^5$.

Ce type de dérivation utilise un couplage entre dérivation par file et duplication. Les utilisations de marqueurs différents et/ou l'introduction de marqueurs spécifiques de file permettent de restreindre les duplications à des morceaux de dérivation.

Dérivation 84.

1. *entrée lexicale de type 1 :* $= X\ s_1\ -m_a$

   entrée lexicale de type 2 : $X\ -m_b$

   fusion :

$\underbrace{\mathbf{s_1} - \mathbf{m_a}, -m_b}$

2. *entrée lexicale de type 4 :* $= s_1 \ + M_b \ s_1 \ - m_b$

   fusion : $\underbrace{+\mathbf{M_b}\ \mathbf{s_1} \ - \mathbf{m_b}, -m_a, -m_b}$

   déplacement : $\underbrace{\mathbf{s_1} \ - \mathbf{m_b}, -m_a}$

3. *entrée lexicale de type 5 :* $= s_1 \ s_2$

   fusion : $\underbrace{\mathbf{s_2}, -m_b, -m_a}$

4. *entrée lexicale de type 6 :* $= s_2 \ + M_a \ = w_1 \ s_2$

   fusion : $\underbrace{+\mathbf{M_a} \ = \mathbf{w_1}\ \mathbf{s_2}, -m_b, -m_a}$

   déplacement : $\underbrace{= \mathbf{w_1}\ \mathbf{s_2}, -m_b}$

5. *entrée lexicale de type 7 :* $= w \ w_1 \ - m_a$

6. *entrée lexicale de type 8 :* $w_1 \ - m_{a_d}$

   fusion : $\underbrace{\mathbf{w_1} \ - \mathbf{m_a}, -a_d}$

7. *fusion item 4 et 6 :* $\underbrace{\mathbf{s_2}, -m_b, \underbrace{-m_a, -m_{a_d}}}$

8. *entrée lexicale de type 9 :* $= s_2 s_1$

   fusion : $\underbrace{\mathbf{s_1}, -m_b, \underbrace{-m_a, -m_{a_d}}}$

9. *entrée lexicale de type 10 :* $= s_1 s_3$

   fusion : $\underbrace{\mathbf{s_3}, -b, \underbrace{-m_a, -m_{a_d}}}$

10. *entrée lexicale de type 11 :* $= s_3 + M_{a_D} s_3 - m_b$

    fusion : $\underbrace{+\mathbf{M_{a_D}} \mathbf{s_3} - \mathbf{m_b}, -m_b, \underbrace{-m_a, -m_{a_d}}}$

    déplacement : $\underbrace{\mathbf{s_3} \ - \mathbf{m_b}, -m_b, -m_a}$

    qui représente $a^{F(3)}$. Dans la récurrence suivante les marqueurs sont inversés.

11. *entrée lexicale de type 12 :* $= s_3 \ s_{1bis}$

    fusion : $\underbrace{\mathbf{1bis}, -m_b, -m_b, -m_a}$

12. *entrée lexicale de type 13 :* $= s_{1bis} \ + M_a \ s_{1bis} - m_a$

    fusion : $\underbrace{+\mathbf{M_a}\ \mathbf{s_{1bis}} \ - \mathbf{m_a}, -m_b, -m_b, -m_a}$

    déplacement : $\underbrace{\mathbf{s_{1bis}} \ - \mathbf{m_a}, -m_b, -m_b}$

13. *entrée lexicale de type 14 :* $= s_{1bis} s_{2bis}$

    fusion :

$\mathbf{s_{2bis}}, -m_a, -m_b, -m_b$

14. *entrée lexicale de type 15 :* $= s_{2bis} \; + M_b \; = w_2 \; s_{2bis}$

    fusion : $+\mathbf{M_b} \; = \mathbf{w_2} \; \mathbf{2bis}, -m_a, -m_b, -m_b$

    déplacement : $= \mathbf{w_2} \; \mathbf{s_{2bis}}, -m_a, -m_b$

15. *entrée lexicale de type 16 :* $= w_3 \; w_2 \; - m_b$

16. *entrée lexicale de type 17 :* $w_3 \; - m_{b_d}$

    fusion : $\mathbf{w_2} \; - \mathbf{m_b}, -m_{b_d}$

    fusion item 14 et 16 : $\mathbf{s_{2bis}}, -m_a, -m_b, -m_b, -m_{b_d}$

    Puis on réitère les dernières étapes pour le deuxième $-m_b$

17. *entrée lexicale de type 15 :* $= s_{2bis} \; + M_b \; = w_2 \; s_{2bis}$

    fusion : $+\mathbf{M_b} \; = \mathbf{w_2} \; \mathbf{s_{2bis}}, -m_a, -m_b, -m_b, -m_{b_d}$

    déplacement : $= \mathbf{w_2} \; \mathbf{s_{2bis}}, -m_a, -m_b, -m_{b_d}$

    fusion item 17 et 16 - produit : $\mathbf{s_{2bis}}, -m_a, -\mathbf{m_b}, -m_{b_d}, \; -\mathbf{m_b}, -m_{b_d}$

    On repasse par 1bis pour déplacer tous les $-a$

18. *entrée lexicale de type 18 :* $= s_{2bis} s_{1bis}$

    fusion : $\mathbf{s_{1bis}}, -\mathbf{m_b}, -m_{b_d}, \; -\mathbf{m_b}, -m_{b_d}, \; -m_a$

19. *entrée lexicale de type 13 :* $= s_{1bis} \; + M_a \; s_{1bis} - m_a$

    fusion : $+\mathbf{M_a} \; \mathbf{s_{1bis}} - \mathbf{m_a}, -m_b, -m_{b_d}, \; -m_b, -m_{b_d}, \; -m_a$

    déplacement : $\mathbf{s_{1bis}} - \mathbf{m_a}, -\mathbf{m_b}, -m_{b_d}, \; -\mathbf{m_b}, -m_{b_d}$

    Passage au traitement des m_b *dupliqués :*

20. *entrée lexicale de type 19 :* $= s_{1bis} \; s_{3bis}$

    fusion : $\mathbf{s_{3bis}}, -m_a, -\mathbf{m_b}, -m_{b_d}, \; -\mathbf{m_b}, -m_{b_d}$

21. *entrée lexicale de type 20 :* $= s_{3bis} \; + M_{b_D} \; s_{3bis} \; - m_a$

    fusion : $+\mathbf{M_{b_D}} \; \mathbf{s_{3bis}} \; - \mathbf{m_a}, -m_a, -\mathbf{m_b}, -m_{b_d}, \; -\mathbf{m_b}, -m_{b_d}$

    déplacement : $\mathbf{s_{3bis}} \; - \mathbf{m_a}, -m_a, -\mathbf{m_{b_d}}, -m_b, \; -m_b$

22. *entrée lexicale de type 20 :* $= s_{3bis} \; + M_{b_D} \; s_{3bis} \; - m_a$

    fusion :

$$\underbrace{+\mathbf{M_{b_D}}\ \mathbf{s_{3bis}}\ -\mathbf{m_a}, -m_a, -m_a, \underbrace{-m_{b_d}, -m_b}, -m_b}$$

déplacement : $\underbrace{\mathbf{s_{3bis}}\ -\mathbf{m_a}, -m_a, -m_a, -m_b, -m_b}$

Cette récurrence terminée, la dérivation passe dans la phase de conclusion.

23. *entrée lexicale de type 22 :* $= s_{3bis}\ c$

    fusion : $\underbrace{\mathbf{c} - \mathbf{m_a}, -m_a, -m_a, -m_b, -m_b}$

24. *entrée lexicale de type 24 :* $= c\ + M_b\ c\ /a/$

    fusion : $\underbrace{+\mathbf{M_b}\ \mathbf{c/a/}, -m_a, -m_a, -m_a, -m_b, -m_b}$

    déplacement : $\underbrace{c\ /a/, -m_a, -m_a, -m_a, -m_b}$

    On réitére ces dernières étapes pour chacun des marqueurs et on obtient au bout de 12 *fusions/déplacements :* $\underbrace{c\ /a/, /a/, /a/, /a/, /a/}$

qui correspond à $a^5 = a^{F(4)}$.

Théorème 85. *$L = \{a^{F(n)} | n \in I\!N\}$, où F(n) est la fonction de Fibonnaci, est un langage minimaliste.*

Démonstration. La preuve de ce théorème repose exclusivement sur l'utilisation de la dérivation comme une file, ce qui implique que toute dérivation ne suivant pas le synopsis sera rejetée par la SPIC comme montré dans le théorème 28.

Par induction sur la dérivation.

D'après l'exemple, on a vu qu'on pouvait dériver $a^{F(4)}$. En suivant la même dérivation mais en passant directement dans la phase de conclusion après chaque itération, on peut également reconnaître $a^{F(3)}$ et $a^{F(2)}$. À partir de l'entrée de type 0 on peut reconnaître $a^{F(1)}$ et $a^{F(0)}$.

On suppose qu'on a obtenu la dérivation avant passage dans la phase de conclusion pour $a^{F(k)}$ tel que $k = 2n + 1, n \in \mathbb{N}$. La première récurrence déplace tous les marqueurs B en haut de la dérivation. Puis à chaque marqueur A est substitué une sous-structure : $< (w_1\ -m_a, -m_{a_d})$. Après cette partie de dérivation, tous les marqueurs A sont passés en haut de la dérivation. On déplace les marqueurs B, puis chaque marqueur a_d est déplacé (car en position de complément) et remplacé par un marqueur B. On a donc dans t' le résultat obtenu par une itération à partir de t : $|t|_B + 2 \times |t|_A$ marqueur du terminal, répartis $|t'|_A = |t|_A$ et $|t'|_B = |t|_B + |t|_A$. Le passage dans la phase de conclusion nous donnera $a^{2F(n-1)+F(n)} = a^{F(n+1)}$.

Dans le cas où $a^{F(k)}$ telle que $k = 2n, n \in \mathbb{N}$. On utilise une procédure analogue sur les marqueurs A et les marqueurs B car leurs positions dans la dérivation sont inversées.

Enfin la phase de conclusion substitue à chaque marqueur A et B une forme phonologique $/a/$

Seules les dérivations suivant le synopsis aboutissent, donnant $a^{F(n)}, n \in \mathbb{N}$. □

4.7 Conclusion

Tous ces lexiques se sont révélés importants dans l'analyse des correspondances des GMs avec d'autres formalismes. En particulier pour la traduction des MGs vers les MCFGs ou RTG - Regular Tree Grammars -, dont une première approche a été donné par Michaelis *et al* [MMM00].

La présentation de la section 4.2 est la généralisation des langages *k-multiple-agreement*, $\forall k \in \mathbb{N}$, $a_1^n \cdots a_k^n$. De la même manière la présentation de la section 4.3 est la généralisation des compteurs enchâssés aux k-crossed-agreement :
$\forall k \in \mathbb{N}$, $a_1^{n_1} \cdots a_k^{n_k} b_1^{n_1} \cdots b_2^{n_k}$. Ils ne nécessitent pas de relaxer la SMC.

La section 4.4 présente la k-duplication : $\omega^k, \forall k \in \mathbb{N}$, ω une phrase sur un alphabet Σ. Nous avons ainsi montré la possibilité de réaliser les généralisations des diverses constructions hors-contextes.

On peut facilement étendre ce lexique pour toutes les puissances de puissances sur un terminal. Cela fait simplement varier le nombre de duplications. Ici il n'y en avait qu'une seule à réaliser. On trouvera en annexe C une présentation d'un lexique qui utilise à la fois un compteur en exposant et un compteur en puissance de 2. On peut également simplement envisager de mélanger ce traitement des compteurs en exposant d'exposant à celui des compteurs enchâssés.

Le lexique pour les langages de compteur a été utilisé dans [Mer06] où la transformation de ces lexiques en LCFRS donne une grammaire de taille minimale. De plus, les phrases de Fibonacci sont utilisées pour la comparaison des GMs avec les automates à piles de piles, comme nous l'avons mentionné.

Cette étude conclut la partie de nos travaux consacrée aux GMs du point de vue de la syntaxe. Maintenant que les tenants et les aboutissants de la théorie et de sa modélisation sont posés, nous posons la question du passage au niveau suivant, c'est-à-dire la sémantique. Pour cela, comme nous allons longuement revenir dessus dans la deuxième partie, nous utilisons des systèmes de type logique.

Deuxième partie

Formalismes logiques et Interface syntaxe/sémantique

Dans cette nouvelle partie, nous souhaitons nous nous concentrons sur la question de l'interface syntaxe/sémantique. Comme nous venons de le développer, les GMs fournissent un formalisme efficace pour l'analyse qui permet une bonne représentation de la syntaxe. De plus, elles sont simples car n'utilisant que deux types de règles et des listes de traits. Ce qui tendrait à démontrer le caractère minimal du formalisme qui, pour autant, ne perd pas en expressivité.

Cependant, dès les années quatre-vingt-dix, on voit émerger des convergences dans l'évolution de la théorie syntaxique et les grammaires catégorielles les plus simples (celles dites de Adjuckiewicz-Bar-Hillel), [BE96]. Pour leur part, les dérivations obtenues à partir des GCs apportent beaucoup d'informations sur la structure des éléments utilisés dans la dérivation. Il est alors aisé de donner une correspondance règle à règle permettant de calculer une représentation sémantique à chaque étape de la dérivation.

D'un côté, nous sommes en mesure d'utiliser un système performant syntaxiquement, pour lequel il existe des analyseurs efficaces (polynômiaux) et d'un autre côté, nous avons un système pour lequel l'interface syntaxe-sémantique est simple. Ainsi, nous voulons tirer avantage de ces deux versants du problème. Les GCs n'ont malgré tout qu'un pouvoir expressif très restreint et il a alors été envisagé d'utiliser le calcul de Lambek avec produit, [Lam58] qui permet d'obtenir une plus large couverture, [LR01], tout en laissant des questions non résolues.

Nous proposons donc l'utilisation d'une extension de ce calcul qui gère à la fois des opérateurs commutatifs et non-commutatifs. La première version a été introduite par de Groote, [dG96], qui est à la base d'une extension introduite dans [BdGR97] que nous utilisons ici. Pour cela, nous commencerons par introduire ce système et, dans la perspective de le mettre en œuvre par la suite, nous proposons une forme normale qui permet une preuve de la propriété de la sous-formule. Ainsi, le formalisme permettra d'obtenir des dérivations cohérentes. Ces démonstrations ne s'inscrivant pas directement dans le cadre de l'interface syntaxe-sémantique, nous commençons par développer des questions.

Puis, nous revenons à la problématique de l'interface syntaxe/sémantique par la présentation d'un nouveau formalisme : les GMCs. Elles s'inscrivent directement dans le prolongement de R. Berwick et S. Epstein, [BE96], et A. Lecomte et C. Retoré[LR01]. D'autres évolutions des GCs pour l'analyse syntaxique ont parallèlement été développées dans l'école hollandaise dirigée par M. Moortgat, [Moo96], [Moo02b], [Ver05], qui leur ajoute des modalités. Nous ne inscrivons pas dans cette perspective, pour laquelle il nous semble que le caractère de minimalité est plus difficile à conserver. Nous montrons que le système proposé est alors similaire aux GMs sans la SMC. La question de l'intégration de la SMC et de ses conséquences sur l'équivalence des formalismes reste toutefois ouverte.

Puis, en ce basant sur ce nouveau formalisme nous définissons une interface syntaxe/sémantique, pour laquelle nous possédons une correspondance règle à règle et qui permet à tout moment de calculer une formule sémantique. Pour cela nous utilisons l'isomorphisme de Curry-Howard qui nous permet de passer au λ-calcul, [Chu40]. Nous verrons cependant que les simples λ-termes ne suffisent pas et nous introduirons des termes basés sur le $\lambda\mu$-calcul ,[Par92], et la DRT, [KR93].

Enfin, afin de donner corps à ces grammaires, nous développons un fragment de grammaire du français. Dans le but de conserver une cohérence globale à ce fragment, nous avons choisi de modéliser les clitiques. Les phénomènes les régissant dans les langues romanes sont largement reconnus comme non-triviaux et interviennent dans de nombreuses constructions. Cette étude est basée sur les travaux pionniers de Stabler sur cette question, [Sta97] qui utilisent une modélisation de D. Sportiche, [Spo92]

Chapitre 5

Normalisation de la logique mixte

Sommaire

5.1 Présentation 123
5.2 La logique mixte - Partially Commutative Linear Logic 126
5.3 Normalisation du Lambek avec produit 129
5.4 Normalisation des preuves de la logique mixte 138
5.5 Conclusion 151

Dans la première partie de cette présentation, nous avons adopté le point de vue des GMs. À présent, nous souhaitons étendre ce formalisme à une modélisation basée sur un système de type logique. Nous commencerons par introduire et étudier les propriétés de la logique mixte, introduite dans [dG96] et [BdGR97]. Ce type de logique découle des travaux de G. Gentzen, [Gen34a], [Gen36], selon l'orientation proposée par Lambek, [Lam58]. On trouvera une présentation détaillée de l'évolution de ces théories dans [Gir97]. Cette dernière nous servira à proposer un nouveau formalisme semblable à ces grammaires, à partir duquel nous définirons une interface syntaxe/sémantique dans les prochains chapitres. Ce chapitre, un peu en dehors des problématiques plus linguistiques, entame cette nouvelle partie car il pose les bases nécessaires à la compréhension de la suite de ce manuscrit.

5.1 Présentation

Les logiques non-commutatives proviennent naturellement de problématiques mathématiques et peuvent être utilisées pour la modélisation de phénomènes réels du monde. Mathématiquement, la non-commutativité est nécessaire du point de vue de la sémantique vériconditionnelle (sémantique des phrases basées sur des monoïdes qui peuvent être non-commutatifs) et de la syntaxe (calcul des séquents avec des ordres plutôt que des formules, réseaux de démonstration qui peuvent posséder des liens d'axiomes parenthésés). La non-commutativité apparaît également dans des applications comme la théorie de la concurrence, par exemple l'exécution de réseaux de Pétri ou pour la linguistique computationnelle. Ces notions datent des années cinquante et de l'apparition du calcul de Lambek.

On reviendra sur la logique non-commutative, puis sur son intérêt pour la théorie de la concurrence et la linguistique computationnelle.

5.1.1 Logique linéaire non commutative

La logique linéaire [Gir87a] offre un point de vue logique sur le calcul de Lambek [Lam58] et sur les calculs non-commutatifs. Pendant plusieurs années, la difficulté a été d'intégrer les connecteurs commutatifs et non-commutatifs ensemble. Une première solution, sans calcul de termes, a été *Pomset Logic*, maintenant étudiée dans une version étendue avec calcul des séquents appelée *Calculus of Structures* [Gug02].

Une autre sorte de calcul utilisant le calcul des séquents a été introduite par de Groote dans [dG96], basée sur la logique intuitionniste pour fonctionner correctement. Ce calcul consiste en la superposition du calcul de Lambek (non-commutatif) et de la logique linéaire intuitionniste (commutative). Pour marquer la distinction entre les deux types de connecteurs, il est nécessaire que les contextes incluent deux marqueurs différents représentant la conjonction d'hypothèses, l'un étant commutatif l'autre étant non-commutatif. Donc nous utilisons les ordres séries-parallèles, partiellement ordonnés sur des multi-ensembles de formules. Ils forment les parties droites de séquents. On note $(...,...)$ pour l'ordre parallèle et $\langle ...;... \rangle$ pour l'ordre série : donc $\langle (a,b);(c,d) \rangle$ représente l'ordre partiel fini $a{<}c, b{<}c, a{<}d, b{<}d$.

Bien sûr, les deux relations doivent être liées. Soit le produit commutatif est plus fort que le non-commutatif, soit leur relation est inversée. Les deux options fonctionnent de la même manière si une direction est fixée une fois pour toutes. Cette relation entre les ordres est le résultat d'une règle structurelle modifiant l'ordre.

Cependant, une différence doit être notée entre le calcul d'Abrusci-Ruet et le calcul intuitionniste de de Groote. Elle concerne précisément la règle d'ordre. Le calcul d'Abrusci-Ruet utilise une vision intuitionniste limitée à des séquents n'ayant qu'une seule formule dans leur partie droite et les connecteurs intuitionnistes (précisément l'implication et la conjonction). Mais il y a une différence importante avec le calcul de de Groote sur la formation de la règle d'ordre :

$$\frac{\Gamma \text{ordonné par } I \vdash C}{\Gamma \text{ordonné par } J \vdash C}$$

Dans le calcul de de Groote, J peut être n'importe quel ordre tant que $J \subset I$ (vu comme des ensembles de paires ordonnées de formules de Γ). Alors que chez Ruet J ne peut être obtenu que par transformation de relations commutatives en relations non-commutatives, ce qui ne permet pas de fournir comme résultat tous les sous-ordres J. D'ailleurs, Bechet, de Groote et Retoré ont montré que quatre règles de réécriture étaient nécessaires pour obtenir tous les sous-ordres partiels série-parallèles (sp) à partir d'ordres partiels série-parrallèles, voir [BdGR97]. Voici un exemple de dérivation que l'on peut obtenir dans le calcul de de Groote et pas dans le calcul de Ruet :

$$\frac{\langle (a,b);(c,d) \rangle \vdash (a \otimes b) \odot (c \otimes d)}{(\langle a;c \rangle, \langle b,d \rangle) \vdash (a \otimes b) \odot (c \otimes d)}$$

Le calcul d'Abrusci-Ruet admet une syntaxe sous forme de réseaux de preuve qui peuvent être restreints au cas intuitionniste. Le calcul de de Groote, bien que plus souple, ne possède pas ce type de représentation sous forme de réseau de preuve ou de déduction

naturelle : il n'existe que le calcul sous forme de séquents pour lequel l'élimination des coupures est vérifiée par une version sémantique dans [dG96] et par une méthode théorique dans [Ret04]. Dans ce chapitre nous montrons la normalisation de la logique mixte. D'abord nous obtenons un calcul plus adéquat pour la linguistique computationnelle grâce à l'isomorphisme de Curry-Howard et ensuite il peut être vu comme une première étape vers une syntaxe sous forme de réseau de preuve de la logique mixte.

5.1.2 Motivation pour ce type de calcul

La non-commutativité en logique est plus naturelle dans la perspective de la consommation des ressources. Une hypothèse est vue comme une ressource qui peut être utilisée mais il faut alors penser à la façon dont elles sont organisées et accessibles. Abrusci, [Abr91] et d'autres, justifient la présence de la linéarité pour rendre compte de la non-commutativité. Cependant, le premier calcul non-commutatif, le calcul de Lambek, qui a été introduit bien avant la logique linéaire, est un calcul linéaire, dont les relations avec les autres systèmes logiques, en particulier intuitionnistes, ont été comprises seulement après l'invention de la logique linéaire par Girard.

La théorie de la concurrence, basée sur l'ordre des calculs ou des ressources, est une application naturelle à ce type de logique. Dans ce cadre, les preuves sont vues comme des programmes, et la normalisation comme un processus calculatoire. La *Pomset logic* et le calcul des structures sont plus utiles parce que l'ordre qui s'applique aux coupures représente les calculs à exécuter [Ret97, Gug02]. Mais dans le cadre dans lequel les preuves sont vues comme des calculs, par exemple dans le style de la programmation logique de Miller, le calcul étudié ici est plus adéquat. De plus, les exécutions de calculs peuvent être encodées dans le calcul non-commutatif. Ce point est la motivation principale de la proposition de Ruet. Retoré a également fourni une description de l'exécution parallèle d'un réseau de Pétri dans ce calcul. C'est une véritable approche de simultanéité où $a||b$ n'est pas réduit à $a; b \oplus b; a$ (où $\oplus$ est le choix non-déterministe). Une exécution selon un ordre partiel série-parallèle correspond à une preuve dans le calcul partiellement commutatif de la logique mixte. Dans cette approche basée sur l'ordre de calculs parallèles, toutes les transitions de tout ensemble minimal peuvent être utilisées simultanément.

Notre motivation principale pour ce calcul est la linguistique computationnelle et les formalismes des grammaires et en particulier, la description de formalismes MCS. Notre perspective de travail est la description logique des classes de grammaires présentées par Lambek : d'une structure d'analyse, on peut calculer automatiquement la structure logique de la phrase. Ceci est particulièrement vrai si le calcul de Lambek, ou l'extension partiellement commutative que nous utilisons, est représenté sous forme de déduction naturelle. En effet, les catégories syntaxiques peuvent être projetées sur des catégories sémantiques basées sur deux types, individus ι et valeur de vérité t (les types à la Montague). Nous reviendrons sur cette transformation dans le chapitre 7. Dans cette perspective, la preuve du calcul de Lambek (l'analyse syntaxique) peut être transformée en une preuve de la logique intuitionniste, qui est un λ-terme décrivant une formule logique dans le style de Church.

Le calcul de Lambek est cependant trop restrictif pour être un formalisme d'analyse des langues naturelles, notamment parce qu'il ne décrit que les langages hors-contextes. C'est

la raison pour laquelle on utilise le calcul partiellement commutatif. C'est une extension du travail de Lecomte et Retoré [LR01] qui proposaient une représentation des GMs grâce au calcul de Lambek avec produit ($L_{\odot}$). En cela, le calcul de de Groote, présenté sous forme de déduction naturelle permet d'obtenir une représentation des phrases analysées. Dans la perspective de l'analyse et pour les autres applications, il est important de pouvoir proposer une normalisation des preuves (ainsi que l'unicité). En effet, la forme normale est la structure des phrases analysées et la normalisation assure la cohérence du calcul. L'algorithme de normalisation, qui est implicite dans la preuve, permet de définir les analyses correctes qui sont celles d'un séquent prouvable. De plus, les analyses syntaxiques et les formules sémantiques sont obtenues à partir de la forme normale. Enfin, on vérifie la propriété de la sous-formule dans ce calcul, ce qui assure de la cohérence intrinsèque des preuves obtenues.

5.2 La logique mixte - Partially Commutative Linear Logic

5.2.1 Formules et ordre

Le calcul des séquents pour la logique mixte - Partially Commutative Intuitionnistic Multiple Linear Logic (PCIMLL) - a été introduit par de Groote, dans [dG96]. Ce calcul utilise à la fois la logique linéaire multiplicative commutative intuitioniste et le calcul de Lambek avec produit ($L_{\odot}$) qui est la logique linéaire multiplicative non-commutative intuitionniste.

Les formules sont définies à partir d'un ensemble de variables de proposition P, par le produit commutatif ($\otimes$), le produit non-commutatif ($\odot$), l'implication commutative ($\multimap$), les deux implications non-commutatives ($/$ et $\backslash$). Leur syntaxe répond à la grammaire :

$$\mathrm{L} ::= \mathrm{P} \mid \mathrm{L} \odot \mathrm{L} \mid \mathrm{L} \otimes \mathrm{L} \mid \mathrm{L}/\mathrm{L} \mid \mathrm{L}\backslash\mathrm{L} \mid \mathrm{L} \multimap \mathrm{L}$$

La partie gauche des séquents est définie par des multi-ensembles de formules qui représentent l'ordre série-parallèle (sp). Ce dernier peut être défini par les deux opérations suivantes : l'union disjonctive, notée $(..., ...)$ et l'union conjonctive $\langle ...; ... \rangle$ pour lequel le domaine est l'union disjonctive des deux domaines, et chaque formule dans la première composante est *avant* toute autre formule de la deuxième. Ils respectent la syntaxe suivante :

$$\mathrm{CTX} ::= \mathrm{L} | \langle \mathrm{CTX}; \mathrm{CTX} \rangle \mid (\mathrm{CTX}, \mathrm{CTX})$$

Par exemple, le contexte $\langle \langle B; (A \multimap (B \setminus (D \,/\, C), A) \rangle; C \rangle$ représente de l'ordre série-parallèle, $Succ(B) = (A, A \multimap (B \setminus (D \,/\, C))$, $Succ(A) = Succ(A \multimap (B \setminus (D \,/\, C)) = C$ où $Succ(X)$ est le successeur immédiat de X et cette fonction du domaine vers des parties de ce domaine détermine complètement un ordre fini.

Le terme représentant un ordre *sp* est unique modulo la commutativité de $(\_, \_)$ et l'associativité de $(..., ...)$ et $\langle ...; ... \rangle$. La notation sous cette forme est en fait abrégée, une notation commode pour représenter les ordres série-parallèles. Cela est vrai même si les termes *sp* sont différents. La partie gauche de deux séquents est considérée comme égale à chaque fois qu'ils sont égaux en tant que multi-ensembles partiellement ordonnés.

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash C}{< \Gamma; \Delta > \vdash C}\,[\backslash_e] \qquad \frac{\Delta \vdash A/C \quad \Gamma \vdash A}{< \Delta; \Gamma > \vdash C}\,[/_e] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \multimap C}{(\Gamma, \Delta) \vdash C}\,[\multimap_e]$$

$$\frac{< A; \Gamma > \vdash C}{\Gamma \vdash A\backslash C}\,[\backslash_i] \qquad \frac{< \Gamma; A > \vdash C}{\Gamma \vdash C/A}\,[/_i] \qquad \frac{(A, \Gamma) \vdash C}{\Gamma \vdash A \multimap C}\,[\multimap_i]$$

$$\frac{\Delta \vdash A \odot B \quad \Gamma, < A; B >, \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[\odot_e] \qquad \frac{\Delta \vdash A \otimes B \quad \Gamma, (A, B), \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[\otimes_e]$$

$$\frac{\Delta \vdash A \quad \Gamma \vdash B}{< \Delta; \Gamma > \vdash A \odot B}\,[\odot_i] \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{(\Delta, \Gamma) \vdash A \otimes B}\,[\otimes_i]$$

$$\frac{}{A \vdash A}\,[axiom] \qquad \frac{\Gamma \vdash C}{\Gamma' \vdash C}\,[\text{entropy} — \text{whenever } \Gamma' \sqsubset \Gamma]$$

Fig. 34 – Règles de la logique mixte.

Les lettres grecques en majuscule sont utilisées pour les contextes. Une expression $\Gamma[]$ représente un contexte où l'on distingue un élément particulier $[*]$, où pour une expression $\Gamma[\Delta]$ l'élément $[*]$ est remplacé par le contexte Δ. Ces résultats sont présentés dans [BdGR97] et [Ret04].

On représente les règles de la logique mixte par les règles de la figure 34, celles-ci reprennent les règles classiques de la logique linéaire multiplicative commutative intuitioniste et de la logique linéaire multiplicative non-commutative intuitionniste, chacune apportant les règles d'introduction et d'élimination pour son(ses) connecteur(s) implicatif(s) et pour son connecteur produit. De plus, nous utilisons la règle d'axiome, ainsi qu'une règle d'entropie ($\sqsubset$), correspondant à l'inclusion des ordres (affaiblissement des relations d'ordre).

Revenons sur quelques explications et commentaires de ce calcul, en particulier en ce qui concerne la règle d'entropie :

$\Gamma' \sqsubset \Gamma$ toutes les fois que ces contextes qui sont des ordres *sp* partiellement ordonnés de formules ont les mêmes domaines de multi-ensembles $|\Gamma| = |\Gamma'|$, et qu'on considère chaque occurrence d'une formule comme distincte des autres (si $A < B$ dans Γ' alors $A < B$ dans Γ). L'inclusion $\sqsubset$ des ordres série-parallèles peut être vue comme une règle de réécriture (modulo commutativité et associativité) sur les termes *sp* les dénotant, comme montré dans [BdGR97] - se référer également à [Ret04] où la règle d'ordre est utilisée dans l'autre sens, mais cela ne change pas la normalisation.

On remarquera que pour l'application des règles $\otimes_e$ et $\odot_e$, A et B doivent être équivalents :

$$\forall X \neq A, B \begin{cases} X < A \Leftrightarrow X < B \\ X > A \Leftrightarrow X > B \end{cases}$$

Dans la règle $\otimes_e$, on a $A \not\leq B$ et $A \not\geq B$, alors que dans la règle $\odot_e$, on a $A < B$. Elles sont alors remplacées par le contexte ayant produit $A \otimes B$ (dans la figure 34, il s'agit de Δ)

Cette formulation dans le style du λ-calcul de l'élimination du produit linéaire est due à Abramsky dans [Abr93], du côté des termes, cela correspond au constructeur $let\ x = (u_1, v_1) in\ t(u, v)$.

De plus, une première version de la normalisation a été exposée dans [Neg02b] mais la résolution du problème a nécessité l'introduction de règles complexes (et potentiellement *ad hoc*) pour MLL (Multiplicative Linear Logic). On suppose que ces règles sont motivées par d'autres propriétés et fonctionnent pour le calcul linéaire complet.

Nous reviendrons sur l'utilisation de la logique mixte pour la reconnaissance d'énoncés des langues naturelles dans le chapitre 6. Cependant, afin de donner une intuition de l'utilisation future de ce calcul et justifier la nécessité de proposer une normalisation, on donne les contours de ce problème. Dans ces grammaires, comme dans les grammaires catégorielles, l'utilisation des implications non-commutatives permet d'unir des éléments entre eux tout en ayant une relation d'ordre sur ces éléments. Cette opération sera utile à la redéfinition de l'opération de *fusion*. Parallèlement, l'utilisation d'hypothèses, que l'élimination de produit permet de substituer dans la suite de la dérivation, permet de marquer une position dans la dérivation puis de les utiliser pour positionner une autre dérivation lorsque toutes les hypothèses (tous les traits) sont vérifiées. En cela nous nous rapprochons de la notion de *déplacement*. Les hypothèses sont exactement entendues comme des ressources disponibles.

Cependant, bien que des propositions aient été faites dans ce sens sans utiliser la non-commutativité, [LR99], [ALR04], elles ne permettent pas de rendre compte au mieux de certains phénomènes syntaxiques. C'est pour cela que nous souhaitons nous positionner dans un cadre logique offrant les connecteurs commutatifs et non-commutatifs pour l'implication et le produit. Ces propositions ont l'avantage de permettre de définir des isomorphismes des formules vers les types, à l'image de l'isomorphisme de Curry-Howard et donc d'utiliser le λ-calcul pour fournir des formules logiques "représentant le sens de l'énoncé" (soit un premier pas vers une interface syntaxe-sémantique).

Pour utiliser complètement cette logique, nous devons nécessairement vérifier la cohérence de nos preuves (c'est-à-dire vérifier la propriété de la sous-formule). Pour cela, nous commençons par proposer une normalisation dans cette logique. Les règles de ce calcul sont reprises dans l'annexe D.2.2, ainsi que celles de $\mathrm{L}_{\odot}$ dans l'annexe D.2.1.

5.2.2 Définitions générales

Pour une preuve δ, on note S_j une occurrence d'un séquent dans δ, $|S_j|$ pour le séquent correspondant, et $|S_j|^r$ la formule dans la partie droite de ce séquent.

Dans une preuve δ, $B(S_0)$ est la **branche principale** issue d'une occurrence S_0 d'un séquent, $|B(S_0)|$ est le plus petit chemin contenant S_0 et clos par les opérations suivantes :

1. Si $S \in B(S_0)$ est obtenue par une règle unaire R d'une occurrence S' d'un séquent, alors $S' \in B(S_0)$.
2. Si $S \in B(S_0)$ est obtenue par une élimination de produit $\odot_e$ (*resp.* $\otimes_e$), alors la prémisse portant le marqueur S', avec $|S'| = \Gamma[\langle A, B\rangle] \vdash C$ (resp. $|S'| = \Gamma[(A, B)] \vdash C$) est aussi dans $B(S_0)$.
3. Si $S \in B(S_0)$ est obtenue par une règle d'élimination d'implication $\backslash_e$ (*resp.* $/_e$, $\multimap_e$),

alors la prémisse portant le marqueur S', avec $|S'| = \Delta \vdash A \setminus C$ (*resp.* $|S'| = \Delta \vdash A \,/\, C$, $|S'| = \Delta \vdash A \multimap C$) est aussi dans $B(S_0)$.

Pour tout chemin d'une branche principale $B(S)$ de S à S_i tel que $|S|^r = |S_i|^r$, si $|S|$ est une règle d'élimination et $|S_i|$ une règle d'introduction elles le sont nécessairement sur une même formule et on dit que ces deux règles sont **conjointes**.

5.3 Normalisation du Lambek avec produit

5.3.1 Propriétés de $\mathrm{L}_\odot$

Le calcul de Lambek avec produit ($\mathrm{L}_\odot$) est la restriction de la logique mixte aux connecteurs $\setminus$, $/$ et $\odot$. De plus, il n'utilise que l'ordre $\langle ...; ... \rangle$, c'est-à-dire il traite de séquences de formules (ordre total). Dans ce cas, nous ne le marquerons pas. La règle d'entropie n'est donc pas utilisée.

Propriété 86. *Soit R une élimination de produit $\odot_e$ de $\Gamma[\Delta] \vdash C$ entre une preuve δ_0 de conclusion $\Delta \vdash A \odot B$ et une preuve de conclusion $\Gamma[A, B] \vdash C$ obtenue par une règle R' entre une preuve δ_1 d'un séquent $\Theta[A, B] \vdash X$ (et si R' est une règle binaire une seconde preuve δ_2 de conclusion $\Psi \vdash U$). Cette description de la structure de ces preuves est reprise dans la figure 35.*

On peut alors obtenir une preuve pour le même séquent $\Gamma[\Delta] \vdash C$ en appliquant d'abord la règle R entre la preuve de conclusion δ_0 de $\Delta \vdash A \odot B$ et la preuve δ_1 de conclusion $\Theta[A, B] \vdash X$ donnant le séquent $\Theta[\Delta] \vdash X$. En appliquant la règle R' à cette nouvelle preuve et potentiellement la preuve δ_2, on obtient le même séquent $\Gamma[\Delta] \vdash C$.

$$\dfrac{\dfrac{\vdots\,\delta_0}{\Delta \vdash A \odot B} \quad \dfrac{\vdots\,\delta_2 \quad \vdots\,\delta_1}{\dfrac{\Psi \vdash U \quad \Theta[A,B] \vdash X}{\Gamma[A,B] \vdash C}\,R'}}{\Gamma[\Delta] \vdash C}\,R \quad \Rightarrow \quad \dfrac{\dfrac{\vdots\,\delta_2}{\Psi \vdash U} \quad \dfrac{\vdots\,\delta_0 \quad \vdots\,\delta_1}{\dfrac{\Delta \vdash A \odot B \quad \Theta[A,B] \vdash X}{\Theta[\Delta] \vdash X}\,R}}{\Gamma[\Delta] \vdash C}\,R'$$

Fig. 35 – Structure des preuves autorisant la montée du produit dans $\mathrm{L}_\odot$.

Démonstration. La preuve de cette proprosition est une étude de cas en fonction du type de la règle présente au-dessus de la règle d'élimination de produit. Nous allons à présent présenter les différents cas :

- Montée sur $\setminus_e$:
  - hypothèses dans la prémisse gauche de $\setminus_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{A, B \vdash D \quad \Delta \vdash D \setminus C}{A, B, \Delta \vdash C}\,[\setminus_e]}{\Gamma, \Delta \vdash C}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \odot B \quad A, B \vdash D}{\Gamma \vdash D}\,[\odot_e] \quad \Delta \vdash D \setminus C}{\Gamma, \Delta \vdash C}\,[\setminus_e]$$

- hypothèses dans la prémisse droite de $\backslash_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta \vdash D \quad A, B \vdash D\backslash C}{\Delta, A, B \vdash C}\,[\backslash_e]}{\Delta, \Gamma \vdash C}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\Delta \vdash D \quad \dfrac{\Gamma \vdash A \odot B \quad A, B \vdash D\backslash C}{\Gamma \vdash D\backslash C}\,[\odot_e]}{\Delta, \Gamma \vdash C}\,[\backslash_e]$$

∘ Montée sur $\backslash_i$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{D, \Delta, A, B, \Delta' \vdash C}{\Delta, A, B, \Delta' \vdash D\backslash C}\,[\backslash_i]}{\Delta, \Gamma, \Delta' \vdash D\backslash C}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \odot B \quad D, \Delta, A, B, \Delta' \vdash C}{D, \Delta, \Gamma, \Delta' \vdash C}\,[\odot_e]}{\Delta, \Gamma, \Delta' \vdash D\backslash C}\,[\backslash_i]$$

∘ Montée sur $\odot_e$:

- hypothèses dans la prémisse gauche de $/_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta \vdash C/D \quad A, B \vdash D}{\Delta, A, B \vdash C}\,[/_e]}{\Delta, \Gamma \vdash C}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\Delta \vdash C/D \quad \dfrac{\Gamma \vdash A \odot B \quad A, B \vdash D}{\Gamma \vdash D}\,[\odot_e]}{\Delta, \Gamma \vdash C}\,[/_e]$$

- hypothèses dans la prémisse droite de $/_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{A, B \vdash C/D \quad \Delta \vdash D}{A, B, \Delta \vdash C}\,[/_e]}{\Gamma, \Delta \vdash C}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \odot B \quad A, B \vdash C/D}{\Gamma \vdash C/D}\,[\odot_e] \quad \Delta \vdash D}{\Gamma, \Delta \vdash C}\,[/_e]$$

∘ Montée sur $/_i$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta, A, B, \Delta', D \vdash C}{\Delta, A, B, \Delta', \vdash C/D}\,[/_i]}{\Delta, \Gamma, \Delta' \vdash C/D}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\Gamma \vdash A \odot B \quad \Delta, A, B, \Delta', D \vdash C}{\dfrac{\Delta, \Gamma, \Delta', D \vdash C}{\Delta, \Gamma, \Delta' \vdash C/D}\,[/_i]}\,[\odot_e]$$

∘ Montée sur $\odot_e$:

- hypothèses dans la prémisse gauche du premier $\odot_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta, A, B, \Delta' \vdash C \odot D \quad \Phi, C, D, \Phi' \vdash E}{\Phi, \Delta, A, B, \Delta', \Phi' \vdash E}\,[\odot_e]}{\Phi, \Delta, \Gamma, \Delta', \Phi' \vdash E}\,[\odot_e]$$

$$\Rightarrow \quad \dfrac{\dfrac{\Gamma \vdash A \odot B \quad \Delta, A, B, \Delta' \vdash C \odot D}{\Delta, \Gamma, \Delta' \vdash C \odot D}\,[\odot_e] \quad \Phi, C, D, \Phi' \vdash E}{\Phi, \Delta, \Gamma, \Delta, \Phi' \vdash E}\,[\odot_e]$$

- hypothèses dans la prémisse droite du premier $\odot_e$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta \vdash C \odot D \quad \Phi, A, B, C, D, \Phi' \vdash E}{\Phi, A, B, \Delta, \Phi' \vdash E}\,[\odot_e]}{\Phi, \Gamma, \Delta, \Phi' \vdash E}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\Delta \vdash C \odot D \quad \dfrac{\Gamma \vdash A \odot B \quad \Phi, A, B, C, D, \Phi' \vdash E}{\Phi, \Gamma, C, D, \Phi' \vdash E}\,[\odot_e]}{\Phi, \Gamma, \Delta, \Phi' \vdash E}\,[\odot_e]$$

- Montée sur $\odot_i$:
  - hypothèses dans la prémisse gauche du premier $\odot_i$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta, A, B, \Delta' \vdash C \quad \Phi \vdash D}{\Delta, A, B, \Delta', \Phi \vdash C \odot D}\,[\odot_i]}{\Delta, \Gamma, \Delta', \Phi \vdash C \odot D}\,[\odot_e] \Rightarrow \dfrac{\dfrac{\Gamma \vdash A \odot B \quad \Delta, A, B, \Delta' \vdash C}{\Delta, \Gamma, \Delta' \vdash C}\,[\odot_e] \quad \Phi \vdash D}{\Delta, \Gamma, \Delta', \Phi \vdash C \odot D}\,[\odot_i]$$

  - hypothèses dans la prémisse droite du premier $\odot_i$:

$$\dfrac{\Gamma \vdash A \odot B \quad \dfrac{\Delta \vdash C \quad \Phi, A, B, \Phi' \vdash D}{\Delta, \Phi, A, B, \Phi' \vdash C \odot D}\,[\odot_i]}{\Delta, \Phi, \Gamma, \Phi' \vdash C \odot D}\,[\odot_e] \quad \Rightarrow \quad \dfrac{\Delta \vdash C \quad \dfrac{\Gamma \vdash A \odot B \quad \Phi, A, B, \Phi' \vdash D}{\Phi, \Gamma, \Phi' \vdash D}\,[\odot_e]}{\Delta, \Phi, \Gamma, \Phi' \vdash C \odot D}\,[\odot_i]$$

On a ainsi étudié tous les cas possibles de combinaisons de règles. L'élimination du produit a donc la possibilité de monter au-dessus de toute règle si les hypothèses utilisées sont dans la même prémisse. □

Définition 87. *On dit d'une règle $\odot_e$ sur la formule du séquent $A \odot B$ qu'elle est **le plus haut possible** si dans les prémisses de la règle au-dessus, les hypothèses A et B n'appartiennent pas à la même formule.*

*On dira qu'une règle est **en bas** de la preuve si c'est la règle qui fournit le séquent conclusion de la preuve.*

On appelle **redex** dans une preuve la succession immédiate d'une règle d'introduction et de son élimination conjointe.

Dans ce calcul, il y a donc quatre redex possibles (en fonction de la position du $\odot_i$ pour les redex dépendant de ce connecteur) dont nous donnons les schémas de preuve :

- Redex$_/$: introduction $/_i$ et élimination immédiate de $/_e$.

$$\dfrac{\dfrac{\begin{array}{c}D\\\vdots\\C\end{array}}{C/D}\,[/_i] \qquad \begin{array}{c}\vdots\\\delta_1\\\vdots\\D\end{array}}{C}\,[/_e] \quad \Rightarrow \quad \begin{array}{c}\vdots\\\delta_1\\\vdots\\D\\\vdots\\C\end{array}$$

- Redex$_\backslash$: introduction $\backslash_i$ et élimination immédiate de $\backslash_e$.

$$\dfrac{\dfrac{\vdots\,\delta_1}{D} \qquad \dfrac{\begin{array}{c}D\\ \vdots\\ C\end{array}}{D\backslash C}\,[\backslash_i]}{C}\,[\backslash_e] \quad \Rightarrow \quad \begin{array}{c}\vdots\,\delta_1\\ D\\ \vdots\\ C\end{array}$$

- Redex$_\odot$: introduction $\odot_i$ et élimination immédiate de $\odot_e$ sur la gauche.

$$\dfrac{\dfrac{\dfrac{\vdots\,\delta_1}{A} \quad \dfrac{\vdots\,\delta_2}{B}}{A \odot B}\,[\odot_i] \qquad \begin{array}{c}A\quad B\\ \vdots\\ D\end{array}}{D}\,[\odot_e] \quad \Rightarrow \quad \begin{array}{ccc}\vdots\,\delta_1 & & \vdots\,\delta_1\\ A & & B\\ & \vdots & \\ & D & \end{array}$$

- Redex$_\odot$: introduction $\odot_i$ et élimination immédiate de $\odot_e$ sur la droite.

$$\dfrac{\dfrac{\vdots\,\delta_1}{A \odot B} \qquad \dfrac{A \quad B}{A \odot B}\,[\odot_i]}{A \odot B}\,[\odot_e] \quad \Rightarrow \quad \begin{array}{c}\vdots\,\delta_1\\ A \odot B\end{array}$$

À partir de la notion de redex, on définit les **k-redex-étendus** dans une preuve.

Tout chemin d'une branche principale $B(S_0)$ de longueur k de S_0 à S_n avec $|S_0|^r = |S_n|^r$, tel que $|S_0|$ est le marqueur d'une règle d'élimination R_e et S_n est la conclusion d'une règle d'introduction R_i, est appelé un **k-redex-étendu**. On remarquera que les *0-redex-étendus* sont en réalité les redex de la logique mixte, déjà présentés.

Proposition 88. *Un k-redex-étendu contient uniquement des règles $\odot_e$ ou alors contient un k'-redex-étendu, avec $k' < k$.*

Démonstration. Pour une instance de X qui devient un X/U, le X sera nécessairement produit par une élimination de U (pour le "même" X). Dans ce cas, le k-redex-étendu contient un plus petit k'-redex-étendu. Seule la règle $\odot_e$ permet de conserver en conclusion l'une des prémisses, elle peut donc être utilisée un nombre non déterminé de fois sans changer la notion de même X. □

5.3.2 Normalisation de $\mathbf{L}_\odot$

Une preuve est sous **forme normale** si elle ne contient pas de k-redex-étendu et que toutes les règles $\odot_e$ sont le plus haut possible.

Pour une preuve δ, on définit $PE(\delta)$ comme l'ensemble des occurrences des règles $\odot_e$ dans δ. Pour $R \in PE(\delta)$, on définit les deux entiers suivants :

1. l'entier $g(R)$ est le nombre de règles s'il y a un k-redex-étendu dans $B(S_0)$ et 0 sinon avec comme prémisse conclusion S_0 ;
2. l'entier $d_{conj}(R)$ est le nombre de règles n'appartenant pas à $PE(\delta)$ entre R et la règle qui lie les hypothèses A et B effacées par R.

On définit $h(\delta)$ comme $\sum_{R\in PE(\delta)} d_{conj}(R)$ et $g(\delta)$ comme le $\min_{R\in PE(\delta)}(g(R))$ égal à zéro si et seulement si δ ne possède plus de k-redex-étendu produit. On note $n(\delta)$ le nombre de règles de δ.

Dans le but de montrer la normalisation dans $L_{\odot}$, on définit la mesure sur les preuves δ formée d'un triplet d'entiers, respectant l'ordre lexicographique :

$$|\delta| = \langle n(\delta), h(\delta), g(\delta)\rangle$$

Propriété 89. *Une preuve δ est sous forme normale si elle ne contient pas de 0-redex-étendu et si $h(\delta) = 0$ et $g(\delta) = 0$.*

Démonstration. Soit δ une preuve de $L_{\odot}$, dans $|\delta|$:

- le premier entier est la mesure classique qui permet de normaliser le calcul de Lambek. Il est minimal si la preuve ne contient pas de 0-redex-étendu.
- le second entier donne le processus pour faire remonter le plus haut possible les $\odot_e$ Dans cette phase de normalisation, les k-redex-étendus $\backslash$ et $/$ apparaissent et peuvent être supprimés de la preuve. Si toutes les règles $\odot_e$ ont leur position le plus haut possible dans la preuve alors $h(\delta) = 0$. Seuls les k-redex-étendus$_{\odot}$ peuvent encore être présents dans δ. Ce cas est présenté dans l'exemple 1 de la figure 36.
- le troisième entier représente le nombre de règles dans un k-redex-étendu$_{\odot}$. Lorsqu'il est nul, c'est qu'il n'y a plus de k-redex-étendu $\odot$ dans δ. Ce cas est présenté dans l'exemple 2 de la figure 36.

□

exemple 1

$$\dfrac{\vdash C \qquad \dfrac{\vdash E\odot F \qquad \dfrac{C,E,F\vdash A/B}{E,F\vdash C\backslash(A/B)}\,[\backslash_i]}{\vdash C\backslash(A/B)}\,[\odot_e]}{\vdash A/B}\,[\backslash_e]$$

$g(\backslash_e) = 1 \Rightarrow$
présence d'un redex-caché

exemple 2

$$\dfrac{\vdash E\odot F \qquad \dfrac{\vdash C \qquad \dfrac{E\vdash (C\backslash A)/B \quad F\vdash B}{E,F\vdash C\backslash A}\,[/_e]}{E,F\vdash A}\,[\odot_e]}{\vdash A/B}\,[/_e]$$

$h(\odot_e) = 1 \Rightarrow$
$\odot_e$ n'est pas en position

Fig. 36 – Exemples de preuve qui ne sont pas sous forme normale

Théorème 90. *Toute preuve δ dans $L_{\odot}$ a une forme normale unique.*

Démonstration. On procède par induction sur $|\delta|$. Par hypothèse d'induction, toute preuve δ' de taille $|\delta'| < \langle r, d, g\rangle$ a une forme normale unique. Et, étant donnée une preuve δ de taille $= \langle r, d, g\rangle$, on montre que δ a également une forme normale unique.

• Si δ possède un redex, on peut alors l'éliminer de la preuve. Dans la preuve résultat δ', $n(\delta') < n(\delta)$, d'où $|\delta'| < \langle r, d, g\rangle$ et par hypothèse d'induction, δ' a une forme normale unique, donc δ également.

• Sinon :

Si $d \neq 0$: soit R la règle $\odot_e$ la plus basse différente de 0. Dans ce cas, il existe une règle $R' \neq \odot_e$ plus haute que R, et R peut monter au-dessus de toutes les règles $\odot_e$ comprises entre elle et R', puis R peut monter sur R'. La preuve résultat δ' est telle que $n(\delta') = n(\delta)$ et $h(\delta') = h(\delta) - 1$. Les valeurs de $d_{conj}(R_i)$, pour R_i les règles du type $\odot_e$ en dessous de R, restent nulles parce que R ne contribue pas à leur $d_{conj}(\_)$). On a donc $\delta' < \langle r, d, g \rangle$, d'après l'hypothèse d'induction, δ' possède une forme normale unique, donc δ également.

Sinon :

Si $g \neq 0$: soit R' telle que $g(R') = g$. Cette règle peut monter au-dessus de sa prémisse gauche. Le nombre de règles et la somme sont inchangés. Dans sa partie gauche, on intervertit un $\odot_e$ et un $\odot_e$ et g diminue alors de 1. Donc la preuve résultat δ' est telle que $|\delta'| < |\delta|$. D'après l'hypothèse d'induction, δ' possède une forme normale unique, donc δ également.

Sinon : d'après la propriété 89, la preuve est sous forme normale.

$\odot_e$ ne peut apparaître qu'après une règle avec deux prémisses liant ses deux hypothèses A et B. De plus un unique $\odot_e$ peut être le plus haut possible après l'utilisation d'une règle. En effet, une règle ne peut relier que deux hypothèses : l'hypothèse la plus à droite de la prémisse gauche et l'hypothèse la plus à gauche de la prémisse droite. Ainsi, on donne une position unique à chaque $\odot_e$, donnant l'unicité de la forme normale. □

Toutes les preuves ont une forme normale unique qui peut être calculée à partir de l'algorithme sous-jacent dans cette preuve. Les formes normales des deux exemples de la figure 36 sont alors obtenues par les preuves suivantes :

exemple 1

$$\dfrac{\vdash E \odot F \qquad \dfrac{\vdash C \qquad \dfrac{C, E, F \vdash A/B}{E, F \vdash C\backslash(A/B)}\,[\backslash_i]}{\vdash A/B}\,[\backslash_e]}{\vdash C\backslash(A/B)}\,[\odot_e]$$

$g(\backslash_e) = 0$

exemple 2

$$\dfrac{\vdash C \qquad \dfrac{\vdash E \odot F \qquad \dfrac{E \vdash (C\backslash A)/B \quad F \vdash B}{E, F \vdash C\backslash A}\,[/_e]}{\vdash A/B}\,[\odot_e]}{E, F \vdash A}\,[/_e]$$

$h(\odot_e) = 0$

5.3.3 Propriété de la sous-formule pour $\mathbf{L}_\odot$

Théorème 91. *Toute preuve du calcul de Lambek avec produit sous forme normale δ d'un séquent $\Gamma \vdash C$, vérifie la propriété de la sous-formule : toute formule dans une preuve sous forme normale est une sous-formule des hypothèses Γ ou de la conclusion de la preuve C.*

Démonstration. On procède par induction sur la preuve :

On utilisera une version plus forte de la propriété de la sous-formule : toute formule dans une preuve sous forme normale est une sous-formule des hypothèses ou de la conclusion de la preuve et si la dernière règle utilisée est une élimination de \ ou / , toute formule est sous-formule des hypothèses. On remarquera que la règle d'axiome vérifie clairement la propriété de la sous-formule car le séquent est alors l'une des hypothèses.

On vérifie donc la validité de l'hypothèse d'induction après l'utilisation de chaque règle :

1. $\backslash_e$: soit la preuve δ, où Γ_i représente l'ensemble des hypothèses utilisées par la sous-preuve δ_i, pour $i \in [2]$:

$$\dfrac{\Gamma_1 \vdots \delta_1 \quad C \qquad \dfrac{\Gamma_2 \vdots \delta_2 \quad \cdot}{C\backslash D}\,[R]}{D}\,[\backslash_e]$$

   D'après l'hypothèse d'induction :
   – dans δ_1 toute formule est une sous-formule de C ou de Γ_1 ;
   – dans δ_2 toute formule est une sous-formule de $C\backslash D$ ou de Γ_2.
   La conclusion D et la prémisse C sont des sous-formules directes de la prémisse $C\backslash D$.
   Nous devons alors vérifier la propriété pour la règle $[R]$ au-dessus de cette prémisse :
   - si R est $/_e$ ou $\backslash_e$: en utilisant l'hypothèse d'induction, on conclut que $C\backslash D$ est une sous-formule de Γ_2. Donc toute formule de δ est sous-formule de Γ_2.
   - si R est $\backslash_i$: ce cas est absurde car la règle serait l'introduction d'un 0-redex-étendu, or δ est sous forme normale.
   - si R est $/_i$: ce cas n'est structurellement pas possible car on ne peut pas dériver $C \setminus D$ à partir de cette règle.
   - si R est $\odot_i$: cet autre cas est également structurellement impossible car on ne peut pas dériver $C \setminus D$ à partie de cette règle.
   - si R est $\odot_e$. À nouveau, nous devons vérifier la règle R' qui se trouve au-dessus :

$$\dfrac{\Gamma_1 \vdots \delta_1 \quad C \qquad \dfrac{A \odot B \qquad \dfrac{\Gamma_2[A,B] \vdots \delta_2 \quad \cdot}{C\backslash D}\,[R']}{C\backslash D}\,[\odot_e]}{D}\,[\backslash_e]$$

   Si R' est $\backslash_e$ ou $/_e$, en utilisant l'hypothèse d'induction, $C\backslash D$ est une sous-formule des hypothèses.
   Si R' est $\backslash_i$: ce cas est absurde car il serait la règle d'introduction d'un 1-redex-étendu, or δ est sous forme normale, donc ce n'est pas possible.
   Si R' est l'une des autres règles d'introduction ($\backslash_i$ ou $\odot_i$) : ces cas sont structurellement impossibles. On ne peut pas dériver $C \setminus D$ à partir de cette règle.
   Si R' est $\odot_e$, à nouveau, nous devons vérifier la validité sur la règle au-dessus. On remarque alors que le nombre de règles au-dessus de cette règle est fini. Elles constituent alors une séquence telle que :

$$\dfrac{\dfrac{\Gamma_1}{\vdots\,\delta_1}\quad\dfrac{A_1\odot B_1\qquad\dfrac{\dfrac{\Gamma_2[A_1,\cdots,A_n,B1,\cdots,B_n]}{\vdots\,\delta_2}}{C\backslash D}\,[R]}{C\backslash D}\,[\odot_e]}{C}\qquad\dfrac{A_n\odot B_n\qquad\dfrac{\dfrac{\Gamma_2[A_1,\cdots,A_n,B1,\cdots,B_n]}{\vdots\,\delta_2}}{C\backslash D}\,[R]}{C\backslash D}\,[\odot_e]}{D}\,[\backslash_e]$$

Dans ce cas, on a :

- soit cette séquence contient uniquement des $\odot_e$ alors $C\backslash D$ est nécessairement l'une des hypothèses.
- soit il existe une règle R'' différente de $\odot_e$ dans cette séquence. Alors en utilisant l'argument qui lui correspond on en déduit que $C\backslash D$ est une sous-formule des hypothèses.

Dans tous les cas, la conclusion de $\backslash_e$ est une sous-formule des hypothèses. L'hypothèse d'induction est bien vérifiée.

2. $/_e$: soit la preuve δ, où Γ_i représente l'ensemble des hypothèses utilisées par la sous-preuve δ_i, pour $i \in [2]$:

$$\dfrac{\dfrac{\Gamma_2}{\vdots\,\delta_2}\qquad\dfrac{\Gamma_1}{\vdots\,\delta_1}}{\dfrac{D/C\qquad C}{D}\,[\backslash_e]}\,[R]$$

La conclusion D et la prémisse C sont des sous-formules directes de la prémisse D/C. On vérifie la propriété en fonction de la règle au-dessus de la prémisse dont la conclusion est D/C : ce cas est analogue au cas $\backslash_e$. On prouve donc de manière analogue que D/C est sous-formule de Γ_2.

3. $\backslash_i$: soit la preuve δ, où Γ_1 représente l'ensemble des hypothèses utilisées par la sous-preuve δ_1 :

$$\dfrac{\dfrac{C,\Gamma_1}{\vdots\,\delta_1}}{\dfrac{D}{C\backslash D}\,[\backslash_i]}$$

Dans δ_1 toute formule est une sous-formule de D ou de C et Γ_1. De plus, D est sous-formule de $C\backslash D$. Donc toute formule de δ est sous-formule de C,Γ_1 ou $C\backslash D$.

4. $/_i$: soit la preuve δ, où Γ_1 représente l'ensemble des hypothèses utilisées par la sous-preuve δ_1 :

$$\dfrac{\begin{array}{c} C,\Gamma_1 \\ \vdots\,\delta_1 \\ D \end{array}}{D/C}\,[/_i]$$

Ce cas est strictement symétrique à celui de $\backslash_i$: D est sous-formule de D/C, et toute formule de δ_1 est sous-formule de C,Γ_1 ou D. Par transitivité, toute formule de δ est sous-formule de C,Γ ou D/C.

5. $\odot_i$: soit la preuve δ, où Γ_i représente l'ensemble des hypothèses utilisées par la sous-preuve δ_i, pour $i \in [2]$:

$$\dfrac{\begin{array}{cc} \Gamma_1 & \Gamma_2 \\ \vdots\,\delta_1 & \vdots\,\delta_2 \\ C & D \end{array}}{C \odot D}\,[\odot_i]$$

   – Dans δ_1 toute formule est une sous-formule de C ou de Γ_1.
   – Dans δ_2 toute formule est une sous-formule de D ou de Γ_2.

   De plus , C et D sont sous-formules de $C \odot D$. Par transitivité, toute formule de δ est sous-formule de Γ_1, Γ_2 ou de $C \odot D$.

6. $\odot_e$: soit la preuve δ, où Γ_i représente l'ensemble des hypothèses utilisées par la sous-preuve δ_i, pour $i \in [2]$:

$$\dfrac{\begin{array}{cc} \Gamma_1 & \Gamma_2 \\ \vdots\,\delta_1 & \vdots\,\delta_2 \\ A \odot B & D \end{array}}{D}\,[\odot_e]$$

   – Dans δ_1, toute formule est une sous-formule de $A \odot B$ ou de Γ_1.
   – Dans δ_2, toute formule est une sous-formule de D ou de Γ_2.

   La conclusion de δ est la conclusion de l'une des prémisses, donc la propriété est vérifiée pour la partie de la preuve qui fournit cette conclusion, soit δ_2. On vérifie la propriété pour l'autre partie de la dérivation. Dans ce cas, on doit montrer que $A \odot B$ est sous-formule des hypothèses de δ_1. Pour cela, on analyse la règle R au-dessus :
   ◦ si R est $\backslash_e$ ou $/_e$, en utilisant l'hypothèse d'induction et le type de règle, $A \odot B$ est sous-formule de Γ_1
   ◦ si R est $\backslash_i$ ou $/_i$: ce cas est structurellement impossible car on ne peut pas produire $A \odot B$ après ces dernières.
   ◦ si R est $\odot_i$: ce cas est absurde car la succession de ces deux règles impliquerait un 0-redex-étendu, or la preuve est sous forme normale.
   ◦ si R est $\odot_e$. On vérifie la propriété pour la règle au-dessus.

$$\dfrac{\dfrac{E \odot F \qquad \begin{array}{c}\Gamma_1[E,F]\\ \vdots\ \delta_1\\ A \odot B\end{array}}{A \odot B}\,[\odot_e] \qquad \begin{array}{c}\Gamma_2[A,B]\\ \vdots\ \delta_2\\ D\end{array}}{D}\,[\odot_e]$$

- Si $A \odot B$ n'est pas une hypothèse : dans ce cas, il existe une règle $\odot_i$ conjointe avec la règle que nous analysons. Cela implique la présence d'un k-redex-étendu dans la dérivation or la preuve est sous forme normale.
- Si $A \odot B$ est l'une des hypothèses de Γ_1, alors $A \odot B$ est bien sous-formule des hypothèses.

Dans tous les cas possibles, $A \odot B$ est sous-formule des hypothèses. Ainsi, on a vérifié la propriété de la sous-formule pour $\odot_e$.

□

Dans $\mathrm{L}_{\odot}$, toutes les preuves ont donc une unique forme normale qui vérifie la propriété de la sous-formule. On observera que contrairement à Negri [Neg02b], les règles utilisées sont les règles usuelles pour ce calcul.

5.4 Normalisation des preuves de la logique mixte

Nous proposons à présent une normalisation des preuves de la logique mixte, à partir de laquelle on vérifie la propriété de la sous-formule. Dans la même perspective que pour $\mathrm{L}_{\odot}$, la normalisation positionne de manière unique les éliminations de produit non-commutatif, et forme des séquences d'éliminations de produit commutatif, la position relative d'une élimination de produit commutatif dans une séquence n'étant pas unique.

5.4.1 Propriétés de la logique mixte

Propriété 92 (Les éliminations de produits peuvent monter dans la preuve)**.** *Soit R une élimination de produit $\otimes_e$ (resp. une règle $\odot_e$) de $\Gamma[\Delta] \vdash C$ entre une preuve δ_0 de conclusion $\Delta \vdash A \otimes B$ et une preuve de conclusion $\Gamma[(A,B)] \vdash C$ (resp. $\Gamma[\langle A;B\rangle] \vdash C$) obtenue par une règle R' d'une preuve δ_1 d'un séquent $\Theta[(A,B)] \vdash X$ (resp. $\Theta[\langle A;B\rangle] \vdash X$) (et si R' est une règle binaire une seconde preuve δ_2 de conclusion $\Psi \vdash U$).*

On peut alors obtenir une preuve pour le même séquent $\Gamma[\Delta] \vdash C$ en appliquant d'abord la règle $\otimes_e$ (resp. la règle $\odot_e$) entre la preuve δ_0 de conclusion $\Delta \vdash A \otimes B$ et la preuve δ_1 de conclusion $\Theta[(A,B)] \vdash X$ (resp. $\Theta[\langle A;B\rangle] \vdash X$) donnant le séquent $\Theta[\Delta] \vdash X$. En appliquant la règle R' à cette nouvelle preuve et potentiellement la preuve δ_2, on obtient le même séquent $\Gamma[\delta] \vdash C$.

On représente cette dérivation par la figure 37

Démonstration. La preuve est similaire à celle de la propriété 86. C'est une étude de cas selon la règle au-dessus de l'élimination de produit. Cette élimination ne peut monter que lorsque les hypothèses qui doivent être effacées sont dans la même prémisse et occupent leurs places respectives selon l'ordre requis par l'élimination.

$$\dfrac{\dfrac{\vdots\,\delta_0}{\Delta \vdash A \otimes B} \quad \dfrac{\vdots\,\delta_2 \qquad \vdots\,\delta_1}{\dfrac{\Psi \vdash U \quad \Theta[(A,B)] \vdash X}{\Gamma[(A,B)] \vdash C}\, R'}}{\Gamma[\Delta] \vdash C}\, R \quad \Rightarrow \quad \dfrac{\dfrac{\vdots\,\delta_2}{\Psi \vdash U} \quad \dfrac{\vdots\,\delta_0 \qquad \vdots\,\delta_1}{\dfrac{\Delta \vdash A \otimes B \qquad \Theta[(A,B)] \vdash X}{\Theta[\Delta] \vdash X}\, R}}{\Gamma[\Delta] \vdash C}\, R'$$

Fig. 37 – Structure des preuves autorisant la montée du produit dans la logique mixte.

Nous reprenons donc chaque cas pour $\otimes_e$.

- Montée sur $\backslash_e$.
  - si les hypothèses sont dans la prémisse gauche de $\backslash_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Gamma[(A,B)] \vdash D \quad \Phi \vdash D\backslash C}{< \Gamma[(A,B)]; \Phi > \vdash C}\,[\backslash_e]}{< \Gamma[\Delta]; \Phi > \vdash C}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad \Gamma[(A,B)] \vdash D}{\Gamma[\Delta] \vdash D}\,[\otimes_e] \qquad \Phi \vdash D\backslash C}{< \Gamma[\Delta]; \Phi > \vdash C}\,[\backslash_e]$$

  - si les hypothèses sont dans la prémisse droite de $\backslash_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Gamma \vdash D \quad \Phi[(A,B)] \vdash D\backslash C}{< \Gamma; \Phi[(A,B)] > \vdash C}\,[\backslash_e]}{< \Gamma; \Phi[\Delta] > \vdash C}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Gamma \vdash D \qquad \dfrac{\Delta \vdash A \otimes B \quad \Phi[(A,B)] \vdash D\backslash C}{\Phi[\Delta] \vdash D\backslash C}\,[\otimes_e]}{< \Gamma; \Phi[\Delta] > \vdash C}\,[\backslash_e]$$

- Montée sur $/_e$.
  - si les hypothèses sont dans la prémisse droite de $/_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Phi \vdash C/D \quad \Gamma[(A,B)] \vdash D}{< \Phi; \Gamma[(A,B)] > \vdash C}\,[/_e]}{< \Phi; \Gamma[\Delta] > \vdash C}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Phi \vdash C/D \qquad \dfrac{\Delta \vdash A \otimes B \quad \Gamma[(A,B)] \vdash D}{\Gamma[\Delta] \vdash D}\,[\otimes_e]}{< \Phi; \Gamma[\Delta] > \vdash C}\,[/_e]$$

  - si les hypothèses sont dans la prémisse gauche de $/_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Phi[(A,B)] \vdash C/D \quad \Gamma \vdash D}{< \Phi[(A,B)]; \Gamma > \vdash C}\,[/_e]}{< \Phi[\Delta]; \Gamma > \vdash C}\,[\otimes_e] \Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad \Phi[(A,B)] \vdash C/D}{\Phi[\Delta] \vdash C/D}\,[\otimes_e] \qquad \Gamma \vdash D}{< \Phi[\Delta]; \Gamma > \vdash C}\,[/_e]$$

- Montée sur $\multimap_e$
  - si les hypothèses sont dans la prémisse gauche de $\multimap_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Gamma[(A,B)] \vdash D \quad \Phi \vdash D \multimap C}{(\Gamma[(A,B)], \Phi) \vdash C}\,[\multimap_e]}{(\Gamma[\Delta]; \Phi) \vdash C}\,[\otimes_e] \Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad \Gamma[(A,B)] \vdash D}{\Gamma[\Delta] \vdash D}\,[\otimes_e] \qquad \Phi \vdash D \multimap C}{(\Gamma[\Delta], \Phi) \vdash C}\,[\multimap_e]$$

  - si les hypothèses sont dans la prémisse droite de $\multimap_e$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{\Gamma \vdash D \quad \Phi[(A,B)] \vdash D \multimap C}{(\Gamma, \Phi[(A,B)]) \vdash C}\,[\multimap_e]}{(\Gamma, \Phi[\Delta]) \vdash C}\,[\otimes_e] \Rightarrow \dfrac{\Gamma \vdash D \qquad \dfrac{\Delta \vdash A \otimes B \quad \Phi[(A,B)] \vdash D \multimap C}{\Phi[\Delta] \vdash D \multimap C}\,[\otimes_e]}{(\Gamma, \Phi[\Delta]) \vdash C}\,[\multimap_e]$$

- Montée sur $/_i$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{< \Gamma[(A,B)]; D > \vdash C}{\Gamma[(A,B)] \vdash C/D}\,[/_i]}{\Gamma[\Delta] \vdash C/D}\,[\otimes_e] \Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad < \Gamma[(A,B)]; D > \vdash C}{< \Gamma[\Delta]; D > \vdash C}\,[\otimes_e]}{\Gamma[\Delta] \vdash C/D}\,[/_i]$$

- Montée sur $\backslash_i$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{< D; \Gamma[(A,B)] > \vdash C}{\Gamma[(A,B)] \vdash D \backslash C}\,[\backslash_i]}{\Gamma[\Delta] \vdash D \backslash C}\,[\otimes_e] \Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad < D; \Gamma[(A,B)] > \vdash C}{< D; \Gamma[\Delta] > \vdash C}\,[\backslash_i]}{\Gamma[\Delta] \vdash D \backslash C}\,[\backslash_i]$$

- Montée sur $\multimap_i$:

$$\dfrac{\Delta \vdash A \otimes B \qquad \dfrac{(\Gamma[(A,B)], D) \vdash C}{\Gamma[(A,B)] \vdash D \multimap C}\,[\multimap_i]}{\Gamma[\Delta] \vdash D \multimap C}\,[\otimes_e] \Rightarrow \dfrac{\dfrac{\Delta \vdash A \otimes B \quad (\Gamma[(A,B)], D) \vdash C}{(\Gamma[\Delta], D) \vdash C}\,[\otimes_e]}{\Gamma[\Delta] \vdash D \multimap C}\,[\multimap_i]$$

- Montée sur $\otimes_e$:
  - si les hypothèses sont dans la prémisse droite de $\otimes_e$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{\Delta \vdash C \otimes D \quad (\Phi,(A,B),(C,D),\Phi') \vdash E}{(\Phi,(A,B),\Delta,\Phi') \vdash E}\,[\otimes_e]}{(\Phi,\Gamma,\Delta,\Phi') \vdash E}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Delta \vdash C \otimes D \quad \dfrac{\Gamma \vdash A \otimes B \quad (\Phi,(A,B),(C,D),\Phi') \vdash E}{(\Phi,\Gamma,(C,D),\Phi') \vdash E}\,[\otimes_e]}{(\Phi,\Gamma,\Delta,\Phi') \vdash E}\,[\otimes_e]$$

  - si les hypothèses sont dans la prémisse gauche de $\otimes_e$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{(\Delta,(A,B),\Delta') \vdash C \otimes D \quad (\Phi,(C,D),\Phi') \vdash E}{(\Phi,\Delta,(A,B),\Delta',\Phi') \vdash E}\,[\otimes_e]}{(\Phi,\Delta,\Gamma,\Delta',\Phi') \vdash E}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\dfrac{\Gamma \vdash A \otimes B \quad (\Delta,(A,B),\Delta') \vdash C \otimes D}{(\Delta,\Gamma,\Delta') \vdash C \otimes D}\,[\otimes_e] \quad (\Phi,(C,D),\Phi') \vdash E}{(\Phi,\Delta,\Gamma,\Delta,\Phi') \vdash E}\,[\otimes_e]$$

- Montée sur $\otimes_i$:
  - si les hypothèses sont dans la prémisse gauche de $\otimes_i$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{(\Delta,(A,B),\Delta') \vdash C \quad \Phi \vdash D}{(\Delta,(A,B),\Delta',\Phi) \vdash C \otimes D}\,[\otimes_i]}{(\Delta,\Gamma,\Delta',\Phi) \vdash C \otimes D}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\dfrac{\Gamma \vdash A \otimes B \quad (\Delta,(A,B),\Delta') \vdash C}{(\Delta,\Gamma,\Delta') \vdash C}\,[\otimes_e] \quad \Phi \vdash D}{(\Delta,\Gamma,\Delta',\Phi) \vdash C \otimes D}\,[\otimes_i]$$

  - si les hypothèses sont dans la prémisse droite de $\otimes_i$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{\Delta \vdash C \quad (\Phi,(A,B),\Phi') \vdash D}{(\Delta,\Phi,(A,B),\Phi') \vdash C \otimes D}\,[\otimes_i]}{(\Delta,\Phi,\Gamma,\Phi') \vdash C \otimes D}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Delta \vdash C \quad \dfrac{\Gamma \vdash A \otimes B \quad (\Phi,(A,B),\Phi') \vdash D}{(\Phi,\Gamma,\Phi') \vdash D}\,[\otimes_e]}{(\Delta,\Phi,\Gamma,\Phi') \vdash C \otimes D}\,[\otimes_i]$$

- Montée sur $\odot_e$:

- si les hypothèses sont dans la prémisse droite de $\odot_e$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{\Delta \vdash C \odot D \quad (\Phi, (A, B), \Psi, < C; D >, \Psi', \Phi') \vdash E}{(\Phi, (A, B), \Psi, \Delta, \Psi', \Phi') \vdash E}\,[\odot_e]}{(\Phi, \Gamma, \Psi, \Delta, \Psi', \Phi') \vdash E}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Delta \vdash C \odot D \quad \dfrac{\Gamma \vdash A \otimes B \quad (\Phi, (A, B), \Psi, < C; D >, \Psi', \Phi') \vdash E}{(\Phi, \Gamma, \Psi, < C; D >, \Psi', \Phi') \vdash E}\,[\otimes_e]}{(\Phi, \Gamma, \Psi, \Delta, \Psi'\Phi') \vdash E}\,[\odot_e]$$

- si les hypothèses sont dans la prémisse gauche de $\odot_e$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{(\Delta, (A, B), \Delta') \vdash C \odot D \quad (\Phi, \Psi, < C; D >, \Psi', \Phi') \vdash E}{(\Phi, \Psi, \Delta, (A, B), \Delta', \Psi', \Phi') \vdash E}\,[\odot_e]}{(\Phi, \Psi, \Delta, \Gamma, \Delta', \Psi', \Phi') \vdash E}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\dfrac{\Gamma \vdash A \otimes B \quad (\Delta, (A, B), \Delta') \vdash C \odot D}{(\Delta, \Gamma, \Delta') \vdash C \odot D}\,[\otimes_e] \quad (\Phi, \Psi, < C; D >, \Psi', \Phi') \vdash E}{(\Phi, \Psi, \Delta, \Gamma, \Delta', \Psi', \Phi') \vdash E}\,[\odot_e]$$

◦ Montée sur $\odot_i$:

- si les hypothèses sont dans la prémisse gauche de $\odot_i$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{(\Delta, (A, B), \Delta') \vdash C \quad \Phi \vdash D}{< (\Delta, (A, B), \Delta'); \Phi > \vdash C \odot D}\,[\odot_i]}{< (\Delta, \Gamma, \Delta'); \Phi > \vdash C \odot D}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\dfrac{\Gamma \vdash A \otimes B \quad (\Delta, (A, B), \Delta') \vdash C}{(\Delta, \Gamma, \Delta') \vdash C}\,[\otimes_e] \quad \Phi \vdash D}{< (\Delta, \Gamma, \Delta'); \Phi > \vdash C \odot D}\,[\odot_i]$$

- si les hypothèses sont dans la prémisse droite de $\odot_i$:

$$\dfrac{\Gamma \vdash A \otimes B \quad \dfrac{\Delta \vdash C \quad (\Phi, (A, B), \Phi') \vdash D}{< \Delta; (\Phi, (A, B), \Phi') > \vdash C \odot D}\,[\odot_i]}{< \Delta; (\Phi, \Gamma, \Phi') > \vdash C \odot D}\,[\otimes_e]$$

$$\Rightarrow \dfrac{\Delta \vdash C \quad \dfrac{\Gamma \vdash A \otimes B \quad (\Phi, (A, B), \Phi') \vdash D}{(\Phi, \Gamma, \Phi') \vdash D}\,[\otimes_e]}{< \Delta; (\Phi, \Gamma, \Phi') > \vdash C \odot D}\,[\odot_i]$$

◦ Montée sur $\sqsubset$:

$$\dfrac{\dfrac{\Gamma \vdash A \otimes B}{\Gamma' \vdash A \otimes B}\,[\sqsubset] \quad \Delta[A, B] \vdash D}{\Delta[\Gamma'] \vdash D}\,[\otimes_e] \quad \Rightarrow \quad \dfrac{\Gamma \vdash A \otimes B \quad \Delta[A, B] \vdash D}{\dfrac{\Delta[\Gamma] \vdash D}{\Delta[\Gamma'] \vdash D}\,[\sqsubset]}\,[\otimes_e]$$

(l'effacement d'hypothèses dans Γ n'affecte pas l'ordre)

La vérification de la propriété pour le cas non-commutatif est une extension de la propriété 86 qui est analogue au cas précédent. □

On procède de manière analogue au cas étudié pour $\mathrm{L}_{\odot}$. Pour cela on introduit les **redex** du calcul. La logique mixte possède sept redex : un pour chaque connecteur implicatif et deux pour chaque connecteur produit, l'introduction conjointe peut être soit dans la prémisse gauche, soit dans la prémisse droite.

On présente les sept redex :

- Redex$_{/}$: introduction $/_i$ et élimination directe de $/_e$.

$$\dfrac{\dfrac{\vdots}{\langle \Gamma; D\rangle \vdash C}}{\dfrac{\Gamma \vdash C/D}{\langle \Gamma;\Delta\rangle \vdash C}\,[/_i] \qquad \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\end{matrix}}\,[/_e] \quad \Rightarrow \quad \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\\ \vdots\\ \langle \Gamma;\Delta\rangle \vdash C\end{matrix}$$

- Redex$_{\backslash}$: introduction $\backslash_i$ et élimination directe de $\backslash_e$.

$$\dfrac{\begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\end{matrix} \qquad \dfrac{\begin{matrix}\vdots\\ \langle D;\Gamma\rangle \vdash C\end{matrix}}{\Gamma \vdash D\backslash C}\,[\backslash_i]}{\langle \Delta;\Gamma\rangle \vdash C}\,[\backslash_e] \quad \Rightarrow \quad \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\\ \vdots\\ \langle \Delta;\Gamma\rangle \vdash C\end{matrix}$$

- Redex$_{\multimap}$: introduction $\multimap_i$ et élimination directe de $\multimap_e$.

$$\dfrac{\begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\end{matrix} \qquad \dfrac{\begin{matrix}\vdots\\ (D,\Gamma) \vdash C\end{matrix}}{\Gamma \vdash D \multimap C}\,[\multimap_i]}{(\Delta,\Gamma) \vdash C}\,[\multimap_e] \quad \Rightarrow \quad \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta \vdash D\\ \vdots\\ (\Delta,\Gamma) \vdash C\end{matrix}$$

- Redex$_{\odot}$: introduction $\odot_i$ et élimination directe de $\odot_e$ à gauche.

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Delta_1 \vdash A\end{matrix} \quad \begin{matrix}\vdots\\ \delta_2\\ \vdots\\ \Delta_2 \vdash B\end{matrix}}{\langle \Delta_1;\Delta_2\rangle \vdash A \odot B}\,[\odot_i] \qquad \begin{matrix}\vdots\\ \Gamma[\langle A; B\rangle] \vdash D\end{matrix}}{\Gamma[\langle \Delta_1;\Delta_2\rangle] \vdash D}\,[\odot_e] \quad \Rightarrow \quad \Gamma[\langle \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ A\end{matrix} ; \begin{matrix}\vdots\\ \delta_2\\ \vdots\\ B\end{matrix} \rangle] \vdash D$$

- Redex$_{\odot}$: introduction $\odot_i$ et élimination directe de $\odot_e$ à droite.

$$\dfrac{\begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Gamma \vdash A \odot B\end{matrix} \qquad \dfrac{A \vdash A \quad B \vdash B}{\langle A; B\rangle \vdash A \odot B}\,[\odot_i]}{\Gamma \vdash A \odot B}\,[\odot_e] \quad \Rightarrow \quad \begin{matrix}\vdots\\ \delta_1\\ \vdots\\ \Gamma \vdash A \odot B\end{matrix}$$

- Redex$_{\otimes}$: introduction $\otimes_i$ et élimination directe de $\otimes_e$ à gauche.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \delta_1\\ \vdots\\ A\end{array} \quad \begin{array}{c}\vdots\\ \delta_2\\ \vdots\\ B\end{array}}{A \otimes B}\,[\otimes_i] \qquad \begin{array}{c} A \quad B \\ \vdots \\ D \end{array}}{D}\,[\otimes_e] \quad \Rightarrow \quad \Gamma[(\ \begin{array}{c}\vdots\\ \delta_1\\ \vdots\\ A\end{array} \ , \ \begin{array}{c}\vdots\\ \delta_2\\ \vdots\\ B\end{array} \ )] \vdash D$$

- Redex$_\otimes$: introduction $\otimes_i$ et élimination directe de $\otimes_e$ à droite.

$$\dfrac{\begin{array}{c}\vdots\\ \delta_1\\ \vdots\\ \Gamma \vdash A \otimes B\end{array} \qquad \dfrac{A \vdash A \quad B \vdash B}{(A, B) \vdash A \otimes B}\,[\otimes_i]}{\Gamma \vdash A \otimes B}\,[\otimes_e] \quad \Rightarrow \quad \begin{array}{c}\vdots\\ \delta_1\\ \vdots\\ \Gamma \vdash A \otimes B\end{array}$$

À nouveau, on utilise la notion de k-redex-étendu, que nous redonnons pour ce calcul. Tout chemin d'une branche principale $B(S_0)$ de longueur k de S_0 à S_n avec $|S_0|^r = |S_n|^r$, tel que $|S_0|$ est le marqueur d'une règle d'élimination R_e et S_n est la conclusion d'une règle d'introduction R_i est appelé un **k-redex-étendu**. On remarquera que les *0-redex-étendus* sont en réalité les redex de la logique mixte, présentés ci-dessus.

Une preuve est sous **forme normale** si elle ne contient pas de k-redex-étendu, pour tout $k \in \mathbb{N}$.

5.4.2 Normalisation de la logique mixte

Une preuve est sous **forme normale** si elle ne contient pas de k-redex-étendu. On définit ensuite les composantes de la mesure utiles à l'induction pour la normalisation.

1. Pour une règle R, *élimination d'implication* ($\backslash_e$, $/_e$ ou $\multimap_e$), avec comme prémisse conclusion S_0, l'entier $e(R)$ est k s'il y a un k-redex-étendu dans $B(S_0)$ appelé le k-redex-étendu au-dessus de R et 0 sinon.

   Dans cette version, nous comptons toutes les règles y compris les règles d'entropie et les éliminations de produit (c'est donc exactement le nombre k pour les k-redex-étendus concernant l'élimination d'implications).

2. Pour une règle R, *élimination de produit*, avec comme prémisse conclusion S_0, l'entier $g(R)$ est k s'il y a un k-redex-étendu dans $B(S_0)$ appelé le k-redex-étendu au-dessus de R et 0 sinon. (c'est donc exactement le nombre k pour les k-redex-étendus concernant l'élimination de produit).

Soit δ une preuve de la logique mixte, on définit $IER(\delta)$ (*resp.* $PER(\delta)$) comme étant l'ensemble des occurrences des règles d'élimination d'implication (*resp.* de produit) dans δ.

On définit $e(\delta)$ comme le $\min_{R \in IER(\delta)}(e(R))$ et $g(\delta)$ comme le $\min_{R \in PER(\delta)}(g(R))$ égal à 0 si et seulement si δ ne contient plus de k-redex-étendu implicatif (*resp.* produit). De plus, on définit $r(\delta)$ le nombre de règles dans δ. On introduit alors la mesure d'une preuve δ, notée $|\delta|$, comme le triplet d'entiers naturels, relatif à l'ordre lexicographique :

$$< r(\delta), e(\delta), g(\delta) >$$

Propriété 93. *Une preuve dont la mesure est $\langle n, 0, 0\rangle$ est sous forme normale.*

Démonstration. Soit δ une preuve de la logique mixte, dans la mesure de δ :

- le premier entier correspond au nombre de règles qui est minimal si δ ne contient plus de k-redex-étendu.
- le second entier calcule la distance entre chaque partie d'un k-redex-étendu implicatif ($\backslash$, / ou $\multimap$). Si sa valeur est nulle, δ n'en contient plus.
- le troisième entier est la distance entre chaque partie d'un k-redex-étendu produit ($\odot$ ou $\otimes$). Si cette valeur est nulle, δ ne contient plus de k-redex-étendu de ce type.

On remarquera que les deux autres redex ne peuvent être que des 0-redex-étendus. On trouvera deux exemple de preuves qui ne sont pas sous forme normale dans la figure 38. □

exemple 1

$$\dfrac{\vdash C \qquad \dfrac{\vdash E \odot F \qquad \dfrac{(C, \langle E; F\rangle) \vdash A/B}{\langle E; F\rangle \vdash C\backslash(A/B)}\,[\backslash_i]}{\vdash C\backslash(A/B)}\,[\odot_e]}{\vdash A/B}\,[\backslash_e]$$

$e(\backslash_e) = 1 \Rightarrow$
présence d'un redex-caché

exemple 2

$$\dfrac{\dfrac{\vdash A \otimes B \qquad \dfrac{A \vdash E \quad B \vdash F}{(A, B) \vdash E \otimes F}\,[\otimes_i]}{\vdash E \otimes F}\,[\otimes_e] \qquad (E, F) \vdash D}{\vdash D}\,[\otimes_e]$$

$g(\odot_e) = 1 \Rightarrow$
présence d'un redex-caché

Fig. 38 – Exemples de preuve de la logique mixte qui ne sont pas sous forme normale

Propriété 94. *Un k-redex-étendu $S_0 \cdots S_k$, contenant une règle d'élimination d'implication contient un k'-redex-étendu, avec $k > k'$.*

Démonstration. Soit δ une preuve sous forme normale. On remonte au travers d'une branche principale à partir de la conclusion et on exhibe les différentes possibilités rencontrées pour cette dérivation :

L'un des k-redex-étendus minimaux a la structure suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X}\,[introduction]}{\vdots\,\delta_3}}{U}}{U/A}\,[/_i]}{\vdots\,\delta_2}$$
$$\dfrac{U/A \qquad\qquad A}{U}\,[/_e]$$
$$\vdots\,\delta_1$$
$$X$$
$$\dfrac{}{}\,[elimination]$$

On définit alors :

– δ_1 comme une séquence de règles d'élimination et de règles d'entropie ;
– δ_2 comme une séquence d'élimination de produit et d'entropie.

U est la formule produite par la règle d'élimination d'implication la plus haute. Pour cette dérivation, le nombre de symboles dans U est supérieur au nombre de symboles dans X. Puis, dans la branche principale, δ_2 est une séquence d'élimination de produit et de règles d'entropie.

La seule règle au-dessus qui puisse fournir la formule U/A est alors une règle d'introduction, car c'est le seul type de règles qui puisse diminuer le nombre de symboles dans la formule.

Sur l'exemple donné, la seule règle d'introduction que l'on puisse structurellement utiliser est $/_i$ sur la formule A. Cette introduction étant dans la branche principale, elle est nécessairement la règle conjointe de l'introduction précédente. On a donc trouvé un nouveau k'-redex-étendu à l'intérieur du k-redex-étendu. k' est alors le nombre de règles de δ_2 et comme δ_2 est une sous-partie du calcul général on a bien $k > k'$. □

On peut alors proposer une conséquence de cette propriété.

Lemme 95. *Dans une preuve δ, un k-redex-étendu qui minimise $e(\delta)$ différent de zéro ne contient que des règles d'élimination de produit et d'entropie.*

Démonstration. Si le k-redex-étendu est minimal, d'après la propriété 94, il ne contient pas de règles d'élimination. De plus, si l'on n'utilise pas de règles d'élimination, le nombre de symboles dans la formule ne peut pas diminuer. Il doit donc être constant dans le k-redex-étendu.

Dans ce cas, les seules règles que l'on puisse utiliser sont les règles dont la conclusion est l'une des prémisses. La séquence de règles ne peut qu'être composée d'éliminations de produits et de règles d'entropie : $\odot_e, \otimes_e$ ou $\sqsubset$. □

Propriété 96. *Les éliminations de produit et les règles d'entropie peuvent descendre sous les règles d'éliminations d'implication.*

Soit R une élimination de produit $\otimes_e$ (resp. une règle $\odot_e$) de $\Gamma[\Delta] \vdash C$ entre une preuve δ_0 de conclusion $\Delta \vdash A \otimes B$ et une preuve δ_1 de conclusion $\Gamma[(A, B)] \vdash C$ (resp. $\Gamma[\langle A; B\rangle] \vdash C$). Cette preuve est composée avec une preuve δ_2 de conclusion $\Theta \vdash U$ par une règle R' d'élimination d'implication. La conclusion est alors $\langle\Theta; \Gamma[\Delta]\rangle \vdash V$ si R' est $\backslash_e$ (resp. $\langle\Gamma[\Delta]; \Theta\rangle \vdash V$ si R' est $/_e$ et $(\Gamma[\Delta], \Theta) \vdash V$ si R' est $\multimap_e$). La figure 39 présente le cas où R' est $\backslash_e$.

On peut alors obtenir une preuve pour le même séquent qui dépend de R' en appliquant d'abord la règle R' entre la preuve δ_2 de conclusion $\Theta \vdash U$ et la preuve δ_1 de conclusion $\Gamma[(A, B)] \vdash X$ (resp. $\Gamma[\langle A; B\rangle] \vdash X$) donnant le séquent $\langle\Theta; \Gamma[(A, B)]\rangle \vdash V$ (resp. $\langle\Gamma[\langle A; B\rangle]; \Theta\rangle \vdash V$ et $(\Theta, \Gamma[(A, B)]) \vdash V$). En appliquant la règle R à cette nouvelle preuve, on obtient le même séquent $\langle\Theta; \Gamma[\Delta]\rangle \vdash V$ (resp. $\langle\Gamma[\Delta]; \Theta\rangle \vdash V$ et $(\Theta, \Gamma[\Delta]) \vdash V$).

Démonstration. Les éliminations d'implication ne modifient pas l'ordre entre les formules d'une même prémisse et ne les utilisent pas. Les éliminations de produits et les règles d'entropie n'utilisent pas les formules mais uniquement les hypothèses. Donc ces règles peuvent être utilisées dans n'importe quel ordre. □

$$\dfrac{\dfrac{\vdots\,\delta_2}{\Theta \vdash U} \qquad \dfrac{\dfrac{\vdots\,\delta_0}{\Delta \vdash A \otimes B} \quad \dfrac{\vdots\,\delta_1}{\Gamma[(A;B)] \vdash C}}{\Gamma[\Delta] \vdash C}\, R}{\langle \Theta; \Gamma[\Delta] \rangle \vdash V}\, [\backslash_e] \quad \Rightarrow \quad \dfrac{\dfrac{\vdots\,\delta_0}{\Delta \vdash A \otimes B} \quad \dfrac{\dfrac{\vdots\,\delta_2}{\Theta \vdash U} \qquad \dfrac{\vdots\,\delta_1}{\Gamma[(A;B)] \vdash C}}{\langle \Theta; \Gamma[(A;B)] \rangle \vdash V}\, [\backslash_e]}{\langle \Theta; \Gamma[\Delta] \rangle \vdash V}\, R$$

Fig. 39 – Descente du produit sur la règle $\backslash_e$ dans la logique mixte.

Théorème 97. *Toute preuve δ de la logique mixte possède une forme normale.*

Démonstration. Soit δ une preuve telle que $|\delta| = \langle n, e, g \rangle$.

On procède par induction sur la mesure de la preuve.

Par hypothèse d'induction, toute preuve δ' de mesure $|\delta'| < \langle n, e, g \rangle$ a une forme normale.

Si δ possède un redex : le fait de réduire ce redex fait diminuer le nombre de règles dans δ donc la preuve δ' obtenue est telle que $n(\delta') < n(\delta)$, d'où $|\delta'| < |\delta|$. D'après l'hypothèse d'induction δ' possède une forme normale, donc δ aussi.

Sinon :

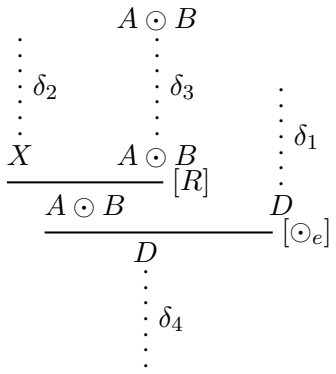
Si $e(\delta) \neq 0$, alors il existe une règle d'élimination d'implication S telle que S fait partie d'un $e(\delta)$-redex-étendu. Cet $e(\delta)$-redex-étendu étant minimal, la propriété 95 implique qu'il contient uniquement des éliminations de produit et des règles d'entropie. De plus la propriété 96, permet de faire monter S au-dessus de la règle au-dessus d'elle (ce qui correspond à faire descendre une élimination de produit ou une règle d'entropie en dessous d'une élimination d'implication). La preuve ainsi obtenue δ' est alors telle que $n(\delta') = n(\delta)$ et $e(\delta') = e(\delta) - 1$. La mesure diminuant, l'hypothèse d'induction permet de conclure que δ' a une forme normale et donc δ.

Sinon $e(\delta) = 0$:

Si $g(\delta) \neq 0$: alors il existe une règle d'élimination de produit R telle que R fait partie d'un $g(\delta)$-redex-étendu. Dans ce cas, δ ne contient plus de redex-étendu implicatif, et s'il est minimal, il ne contient que des règles d'élimination de produit et des règles d'entropie. R peut alors monter sur sa prémisse gauche – la règle $\otimes_i$ conjointe est nécessairement dans cette partie de la dérivation. Or pour ces règles, les éliminations de produit peuvent toujours monter au-dessus comme le montre la propriété 92.

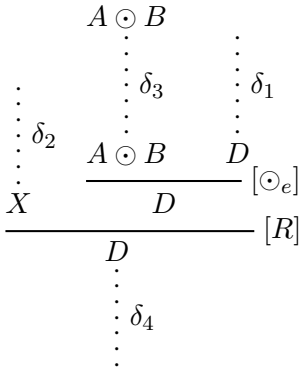
On obtient alors la preuve δ' telle que $n(\delta') = n(\delta)$. On vérifie qu'aucun nouveau k-redex-étendu basé sur une règle de $IEP(\delta)$ n'apparaît :

δ est de la forme :



- toute branche principale de δ_3 vers δ_4 ne contient pas de redex-étendu parce que δ_3 est dans la partie gauche du $\odot_e$
- toute branche principale de δ_1 vers δ_4 peut contenir des redex-étendus.
- toute branche principale de δ_2 vers δ_4 ne contient pas de redex-étendu car δ_2 est dans la partie gauche du $\odot_e$ (seulement pour les éliminations de produit).

Le schéma de réduction du redex donne alors la nouvelle structure de preuve de δ' :



Dans cette nouvelle preuve :

- toute branche principale de δ_3 vers δ_4 ne contient pas de redex-étendu car δ_3 est dans la partie gauche du $\odot_e$
- toute branche principale de δ_1 vers δ_4 ne contient pas de nouveau redex-étendu, et la mesure de ces redex-étendus diminue de 1.
- toute branche principale de δ_2 vers δ_4 ne contient pas de redex-étendu parce que δ_2 est dans la partie gauche de R (qui est nécessairement une élimination de produit).

  La preuve ne contient donc pas de nouveau k-redex-étendu ; donc la dérivation ne contient pas de nouveau redex-étendu implicatif. On a donc, $e(\delta') = e(\delta)$ et $g(\delta') = g(\delta) - 1$. Donc $|\delta'| < |\delta|$ et d'après l'hypothèse d'induction δ' a une forme normale et donc δ en a aussi une.

Sinon : on a $e(\delta) = g(\delta) = 0$, d'après la propriété 93, δ est sous forme normale.

□

Les formes normales des exemples de la figure 38 sont alors obtenues à partir des preuves transformés suivantes :

exemple 1

$$\dfrac{\vdash E \odot F \qquad \dfrac{\vdash C \qquad \dfrac{(C, \langle E; F\rangle) \vdash A/B}{\langle E; F\rangle \vdash C\backslash(A/B)}\,[\backslash_i]}{\vdash A/B}\,[\backslash_e]}{\vdash C\backslash(A/B)}\,[\odot_e]$$

$e(\backslash_e) = 0$

exemple 2

$$\dfrac{\vdash A \otimes B \qquad \dfrac{\dfrac{A \vdash E \quad B \vdash F}{(A, B) \vdash E \otimes F}\,[\otimes_i] \qquad (E, F) \vdash D}{(A, B) \vdash D}\,[\otimes_e]}{\vdash D}\,[\otimes_e]$$

$g(\odot_e) = 0$

Pour les preuves sous forme normale, on peut vérifier la propriété de la sous-formule.

5.4.3 Propriété de la sous-formule pour la logique mixte

Théorème 98. *La propriété de la sous-formule est vérifiée pour les preuves de la logique mixte sous forme normale : dans une preuve de la logique mixte sous forme normale δ d'un séquent $\Gamma \vdash C$, toute formule d'un séquent est sous-formule des hypothèses (Γ) ou de la conclusion (C).*

Démonstration. On procède par induction sur le nombre de règles de la preuve.

À nouveau on utilise un pas d'induction plus fort que la propriété de la sous-formule : toute formule d'une preuve de la logique mixte sous forme normale est sous-formule des hypothèses ou de la conclusion de la preuve et si la dernière règle utilisée est une élimination d'implication $\backslash_e$, $/_e$ ou $-\!\circ_e$, toute formule est sous-formule des hypothèses.

On remarquera que la règle d'entropie conserve la prémisse comme conclusion, donc dans tous les cas elle vérifie le pas d'induction. De plus, la règle d'axiome vérifie cette propriété puisque le séquent est alors l'une des hypothèses.

Reprenons l'induction en dissociant sur chaque cas de règle :

1. pour la règle $\backslash_e$.

$$\dfrac{\begin{array}{c}\Delta_1 \\ \vdots\,\delta_1 \\ \Delta \vdash C\end{array} \qquad \begin{array}{c}\Gamma_2 \\ \vdots\,\delta_2 \\ \Gamma \vdash C\backslash D\end{array}}{< \Delta; \Gamma > \vdash D}\,[\backslash_e]$$

   En utilisant la règle d'induction, toute formule de δ_1 est sous-formule des hypothèses Δ_1 ou de la conclusion C. De plus, toute formule de δ_2 est sous-formule des hypothèses Γ_2 ou de la conclusion $C \setminus D$. Enfin, C est sous-formule de $C\backslash D$ et D aussi.

   Nous devons donc vérifier que $C\backslash D$ est sous-formule des hypothèses de δ_2.

   On analyse la règle R qui produit $C\backslash D$

   - si R est $\backslash_i$: ce cas est absurde car il s'agirait d'un 0-redex-étendu or la preuve est sous forme normale.

- si R est $/_i$ ou $\multimap_i$ ou $\otimes_i$ ou $\odot_i$: ces cas sont structurellement impossibles car ces règles ne peuvent pas produire $C\backslash D$.
- si R est $\backslash_e$, $/_e$ ou $\multimap_e$: d'après l'hypothèse d'induction $C\backslash D$ est sous-formule de Γ_2.
- si R est $\otimes_e$, $\odot_e$ ou $\sqsubset$: la conclusion étant l'une des prémisses, il nous faut regarder la règle au-dessus.
  Si elle correspond à l'un des cas précédents, on peut utiliser le même argument.
  Sinon, la preuve est une séquence finie de $\otimes_e$, $\odot_e$ ou $\sqsubset$ qui font passer l'une des prémisses en conclusion. Dans ce cas, $C\backslash D$ est nécessairement une hypothèse de Γ_2

2. pour les règles $/_e$ et $\multimap_e$, l'analyse est strictement symétrique.

$$\dfrac{\begin{array}{cc}\Gamma_2 & \Delta_1\\ \vdots\,\delta_2 & \vdots\,\delta_1\\ \Gamma\vdash D/C & \Delta\vdash C\end{array}}{<\Gamma;\Delta>\vdash D}\,[/_e] \qquad \dfrac{\begin{array}{cc}\Delta_1 & \Gamma_2\\ \vdots\,\delta_1 & \vdots\,\delta_2\\ \Delta\vdash C & \Gamma\vdash C\multimap D\end{array}}{<\Delta;\Gamma>\vdash D}\,[\multimap_e]$$

   Toute formule est sous-formule de D/C (*resp.* $C\multimap D$) ou des hypothèses et D/C (*resp.* $C\multimap D$) est sous-formule de Γ_2.

3. Pour les règles d'introduction, la conclusion des sous-preuves est une sous-formule de la conclusion de la preuve.
   Si la règle R est $\backslash_i$, soit la preuve δ :

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\,\delta_1\\ <\Gamma;C>\vdash D\end{array}}{\Gamma\vdash C\backslash D}\,[\backslash_i]$$

   En utilisant l'hypothèse d'induction, toute formule de δ_1 est sous-formule des hypothèses Γ ou de la conclusion D. Or D est sous-formule de $C\backslash D$, donc toute formule de δ est sous-formule de Γ ou $C\backslash D$.

4. si la règle R est $/_i$, soit la preuve δ :

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\,\delta_1\\ <\Gamma;C>\vdash D\end{array}}{\Gamma\vdash D/C}\,[/_i]$$

   Toute formule de δ_1 est sous-formule des hypothèses Γ ou de la conclusion D. Or D est sous-formule de D/C, donc toute formule de δ est sous-formule des hypothèses Γ ou de la conclusion D/C.

5. si la règle R est $\multimap_i$, soit la preuve δ :

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\,\delta_1\\ <\Gamma;C>\vdash D\end{array}}{\Gamma\vdash C\multimap D}\,[\multimap_i]$$

Toute formule de δ_1 est sous-formule des hypothèses Γ ou de la conclusion D. Or D est sous-formule de $C \multimap D$, donc toute formule de δ est sous-formule des hypothèses Γ ou de la conclusion $C \multimap D$.

6. si la règle R est $\otimes_i$, soit la preuve δ :

$$\dfrac{\begin{array}{cc}\Delta_1 & \Gamma_2 \\ \vdots\,\delta_1 & \vdots\,\delta_2 \\ \Delta \vdash C & \Gamma \vdash D\end{array}}{(\Delta,\Gamma) \vdash C \otimes D}\,[\otimes_i]$$

   - toute formule de δ_1 est sous-formule des hypothèses Δ_1 ou de la conclusion C.
   - toute formule de δ_2 est sous-formule des hypothèses Γ_2 ou de la conclusion D.
   - de plus C et D sont des sous-formules de $C \otimes D$, alors dans Γ,, toute formule est sous-formule des hypothèses Δ, Γ ou de la conclusion $C \otimes D$.

7. pour la règle $\otimes_e$:

$$\dfrac{\begin{array}{cc}\Delta_1 & \Gamma_2 \\ \vdots\,\delta_1 & \vdots\,\delta_2 \\ \Delta \vdash A \otimes B & \Gamma \vdash D\end{array}}{D}\,[\otimes_e]$$

   - toute formule de δ_1 est sous-formule des hypothèses Δ_1 ou de la conclusion $A \otimes B$.
   - toute formule de δ_2 est sous-formule des hypothèses Γ_2 ou de la conclusion D.
   - De plus, D est la conclusion de δ. Donc les formules de δ_2 sont sous-formules des hypothèses Γ_2 ou de la conclusion de la preuve $\delta : D$.

   Pour vérifier la propriété de la sous-formule, nous devons vérifier que $A \otimes B$ est sous-formule des hypothèses de δ_1.

   On regarde la règle R au-dessus :

   - si R est $\backslash_e$, $/_e$ ou $\multimap_e$, en utilisant l'hypothèse d'induction $A \otimes B$ est sous-formule des hypothèses Δ_1.
   - si R est $\otimes_i$: ce cas est absurde car il s'agirait d'un 0-redex-étendu or la preuve est sous forme normale.
   - si R est $\backslash_i$, $/_i$, $\multimap_i$ ou $\odot_i$: ces cas sont structurellement impossibles car ces règles ne peuvent pas produire $A \otimes B$.
   - $\otimes_e$, $\odot_e$ ou $\sqsubset$: la conclusion étant l'une des prémisses, il nous faut regarder la règle au-dessus :
     - si elle correspond à l'un des cas précédents, on peut utiliser le même argument.
     - sinon, la preuve contenant un nombre fini de règles, la séquence de règles est alors finie. De plus, elle ne contient que des $\otimes_e$, $\odot_e$ et $\sqsubset$, donc la formule est une hypothèse incluse dans Δ_1.

     Dans tous les cas possibles, $A \otimes B$ ou $A \odot B$ est sous-formule des hypothèses.

Dans la logique mixte, toute preuve sous forme normale vérifie la propriété de la sous-formule. □

5.5 Conclusion

En utilisant les motivations issues de la théorie de la concurrence et de la linguistique computationnelle, nous avons défini la logique mixte en déduction naturelle et prouvé la normalisation de ce calcul. Pour Lambek avec produit, un sous-calcul de la logique mixte, nous avons caractérisé une forme normale unique. L'unicité de la forme normale pour la logique mixte peut être réalisée car ce qui distingue deux preuves sous forme normale d'une même preuve, est la position relative des $\otimes_e$ dans les séquences les portant. En réalité, il existe une forme normale unique mais non canonique. L'algorithme de normalisation que nous avons proposé conserve l'ordre de départ entre les $\otimes_e$. On obtient alors une unique forme. Cependant, les preuves avec inversions des $\otimes_e$ dans les séquences de $\otimes_e$ fournissent des preuves équivalentes.

Une extension de ces travaux vers une version sous forme de réseau de preuve de la logique mixte (qui permet également de fournir des λ-termes et des représentations sémantiques) est une étape ultérieure qui poursuivra ces travaux.

Nous verrons dans les chapitres suivants, comment un fragment de ce calcul nous permet de définir un formalisme équivalent aux GMs et comment, à partir de celui-ci, nous obtenons des représentations sémantiques pour les énoncés analysés.

Chapitre 6

Les grammaires minimalistes catégorielles

Sommaire

6.1 Grammaires Minimalistes Catégorielles 154
6.2 GM et GMCs 160
6.3 Exemples 166

Dans ce chapitre nous proposons un formalisme qui simule les GMs en utilisant un fragment des règles de la logique mixte. Les premiers pas d'un rapprochement entre la théorie générative et les GCs remontent à S. Epstein, [BE96]. Un volume de *Language and Computation* rassemble plusieurs articles qui s'inscrivent dans cette perspective, [RS04], en particulier [Lec04b], ainsi que les travaux de Cornell sur les liens entre grammaires de Lambek et grammaires transformationnelles, [Cor04]. Nous allons voir comment les propriétés des GMs peuvent alors être interprétées dans les GMCs. Ce formalisme est inspiré des calculs proposés dans [LR99] et [LR01], eux-mêmes définis à partir du calcul de Lambek avec produit. Il a été utilisé dans [ALR03] et [ALR04] pour définir une interface syntaxe/sémantique pour les GMs et c'est d'ailleurs la motivation originelle de ce calcul. Il a été en effet introduit pour transposer aux GMs les techniques élégantes développées pour le calcul de Lambek et qui réalisent l'interface syntaxe/sémantique à l'aide de l'isomorphisme de Curry-Howard. Cette proposition de modélisation de la théorie générative par un système de type logique n'est pas la première. De nombreuses évolutions, portées par M. Moortgat à l'université d'Utrecht, ont vu le jour à partir de l'ajout de modalités aux GCs, [Moo96b], [Moo96b], [Moo88], [BM]. Un pont entre ce type de grammaires et les GMs a été proposé par W. Vermaat, [Ver99]. Cependant, nous n'avons pas réalisé d'étude comparative directe entre leur système et le nôtre. Il reste clair que les enjeux sont identiques. Cependant, nous préférons garder comme postulat le caractère de minimalité du système qui ne possède que deux types de règles et des structures simples.

À présent, nous en venons à la présentation des GMCs pour lesquelles nous montrerons l'équivalence avec les GMs sans la SMC. Ce chapitre se terminera par une série d'exemples. Remarquons d'abord que les notions introduites dans les GMs de tête, spécifieur, complément, mouvement, n'apparaissent pas explicitement dans la nomenclature des GMCs. Cependant, la grande similitude de ces grammaires avec les GMs permet à chaque étape de

retrouver leur correspondance. Ces concepts ne sont pas abandonnés, mais au contraire complètement intégrés aux GMCs.

6.1 Grammaires Minimalistes Catégorielles

6.1.1 Définitions

Nous donnons la définition des grammaires minimalistes catégorielles. Nous verrons par la suite les rapports que ces grammaires entretiennent avec les GMs.

Les GMCs sont des grammaires lexicalisées. Leurs dérivations sont dirigées par les formules associées aux entrées lexicales construites à partir des connecteurs de la logique mixte. Les dérivations des GMCs sont des preuves particulières de la logique mixte qui sont étiquetées de façon à réaliser les mêmes opérations que les GMs. Nous aborderons cette notion d'étiquetage des preuves de la logique mixte au moment où nous définirons les opérations des GMCs.

Une *grammaire minimaliste catégorielle* - GMC - est définie par un quintuplet $\langle N, \mathrm{P}, Lex, \Phi, C\rangle$ où :

N est l'union de deux ensembles finis disjoints Ph et I qui sont respectivement l'*ensemble des formes phonologiques* et celui des *formes logiques*.

P est l'union de deux ensembles finis disjoints P_1 et P_2 qui sont respectivement l'*ensemble des traits des constituants* et celui des *traits de déplacement*.

Lex est un sous-ensemble fini de $E \times F \times I$ l'ensemble des entrées lexicales[1].

$\Phi = \{merge, move\}$ est l'ensemble des fonctions génératrices,

$C \in \mathrm{P}$ est la formule acceptante.

L'ensemble E est égal à Ph^*, quant à l'ensemble F des formules utilisées pour construire Lex, il est défini à partir de l'ensemble P, du tenseur commutatif $\otimes$ et des deux implications non-commutatives / et \. Les formules contenues dans F sont celles reconnues par le non-terminal L de la grammaire suivante :

$$\begin{aligned}
&\mathrm{L} ::= (\mathrm{B}) \,/\, \mathrm{P}_1 \mid \mathrm{C}\\
&\mathrm{B} ::= \mathrm{P}_1 \setminus (\mathrm{B}) \mid \mathrm{P}_2 \setminus (\mathrm{B}) \mid \mathrm{C}\\
&\mathrm{C} ::= \mathrm{P}_2 \otimes (\mathrm{C}) \mid \mathrm{C}_1\\
&\mathrm{C}_1 ::= \mathrm{P}_1
\end{aligned}$$

Les formules des entrées lexicales des GMCs possèdent donc un / comme premier opérateur, puis une séquence de \. Intuitivement, cette séquence est la traduction de la séquence de sélecteurs et d'assignateurs des entrées des GMs modélisant ce avec quoi une entrée se compose. Les entrées se terminent par une série de $\otimes$. La formule suivante est potentiellement une formule d'une entrée lexicale d'une GMC : $(d \setminus h \setminus j \setminus k \setminus (a \otimes b \otimes c)) \,/\, m$, alors que celle-ci ne l'est pas : $(d \setminus h \setminus j \setminus k \setminus (a \otimes b \otimes c)) \,/\, m \,/\, p$, car elle possède deux /. En résumé, ces formules sont de la forme $(c_m \setminus \ldots \setminus c_1 \setminus (b_1 \otimes \ldots \otimes b_n \otimes a))/d$, avec $a \in \mathrm{P}_1$, $b_i \in \mathrm{P}_2$, $c_j \in \mathrm{P}$ et $d \in \mathrm{P}_1$.

La structure des formules des GMCs est le pendant de celle des listes de traits des GMs. Pour enrichir l'ensemble des opérations des GMs (ajouter *Affix-Hopping*, *Head-Movement*),

[1] Dans ce chapitre, nous considérerons que Lex est un sous-ensemble de $E \times F$. La composante sémantique des entrées lexicales étant définie et étudiée dans le chapitre suivant.

on introduit des décorations sur le symbole des sélecteurs (=) permettant de spécifier le type de fusion à réaliser. De la même façon, nous introduisons des décorations sur les opérateurs \ ou / à l'aide les marqueurs > et < pour obtenir $/_<$, $<\!/$, $/_>$, $>\!/$, $\backslash_<$, $<\!\backslash$, $\backslash_>$, $>\!\backslash$. Du point de vue de la logique mixte, ces nouveaux opérateurs se comportent de la même manière que s'il n'étaient pas marqués. Ils se distinguent des anciens opérateurs au niveau des opérations qu'ils réalisent sur les étiquetages des preuves de la logique mixte et permettent de définir une version enrichie des GMCs.

6.1.2 Dérivations

Les étiquettes

Nous avons déjà évoqué que les dérivations des GMCs étaient des preuves étiquetées de la logique mixte. Avant de définir les étiquetages, nous définissons les étiquettes et quelques opérations pour les manipuler.

Soit V un ensemble dénombrable et infini de variables tel que : $Ph \cap V = \emptyset$. On appelle T l'union de Ph et de V et on définit l'ensemble Σ, appelé *l'ensemble des étiquettes*, comme l'ensemble des triplets d'éléments de T^*.

Chaque composante des éléments de Σ, correspond aux relations spécifieur/tête/complément des arbres minimalistes. Pour une étiquette r, on considérera, en général et sauf mention contraire, que r est égal à $(r_{spec}, r_{tete}, r_{comp})$. Pour remplacer l'une des composantes d'une étiquette r par la chaîne vide, nous utiliserons les notations suivantes :

- $r_{-tete} = (r_{spec}, \epsilon, r_{comp})$
- $r_{-spec} = (\epsilon, r_{tete}, r_{comp})$
- $r_{-comp} = (r_{spec}, r_{tete}, \epsilon)$

L'ensemble des variables ayant au moins une occurrence dans l'étiquette r sera noté $Var(r)$. Le nombre des occurrences d'une variable x dans une chaîne $s \in T^*$ sera noté $|s|_x$, quant au nombre des occurrences de x dans une étiquette r, nous l'écrirons $\varphi_x(r)$. On dit qu'une étiquette est *linéaire* si pour tout x de V, $\varphi_x(r) \leqslant 1$.

Une *substitution* est une fonction partielle de V dans T^*. Si σ est une substitution, que s est une chaîne de T^* et que r est une étiquette, il sera noté $s.\sigma$ et $r.\sigma$ la chaîne et l'étiquette obtenues en remplaçant simultanément dans s et r les variables par la valeur que leur associe σ (les variables pour lesquelles σ n'est pas définie demeurent inchangées). En particulier, si le domaine de définition d'une substitution σ est fini et est égal à $x_1, \ldots, x_n$, et que $\sigma(x_i) = t_i$, nous écrirons $[t_1/x_1, \ldots, t_n/x_n]$ pour dénoter σ. De plus, pour une chaîne s et une étiquette r, nous écrirons respectivement $s[t_1/x_1, \ldots, t_n/x_n]$ et $r[t_1/x_1, \ldots, t_n/x_n]$ pour $s.\sigma$ et $r.\sigma$. On appelle renommage toute substitution qui est aussi une injection à valeur dans V. On dira que deux étiquettes r_1 et r_2 (*resp.* deux chaînes s_1 et s_2) sont égales au renommage près de leurs variables s'il existe un renommage σ tel que $r_1.\sigma = r_2$ (*resp.* $s_1.\sigma = s_2$).

On note $\bullet$ l'opération de *concaténation* sur les chaînes de T^*. On définit $Concat$ l'opération de concaténation sur les étiquettes qui concatène les trois composantes dans l'ordre linéaire gauche-droite : pour $r \in \Sigma$, $Concat(r) = r_{spec} \bullet r_{tete} \bullet r_{comp}$.

Les dérivations étiquetées

Dans cette section, nous commençons par introduire la notion d'étiquetage sur un sous-ensemble des preuves de la logique mixte n'utilisant que les règles logiques utiles à la définition des règles des GMCs. Nous montrons ensuite comment construire des dérivations pour les GMCs. La section D.3 de l'annexe D reprend toutes les règles présentées ici.

On appelle *logique minimaliste* le fragment de la logique mixte composé des règles d'axiome, $\backslash_e$, $/_e$, $\otimes_e$ et $\sqsubset$. Les règles de cette logique permettent de définir celles des GMCs.

Étant donné une GMC $G = \langle N, \text{P}, Lex, \Phi, C\rangle$, un *G-environnement* est ou bien $x : A$ avec $x \in V$ et $A \in F$, ou bien $\langle G_1; G_2\rangle$ ou encore (G_1, G_2) avec G_1 et G_2 des G-environnements qui utilisent des ensembles disjoints de variables. Les G-environnements sont des ordres série-parallèles sur des sous-ensembles de $V \times F$. Nous étendons naturellement aux G-environnements la relation d'entropie, notée comme dans le chapitre précédent $\sqsubset$. On appelle *G-séquents* les séquents de la forme $\Gamma \vdash_G (r_s, r_t, r_c) : B$ où Γ est un G-environnement, $B \in F$ et $(r_s, r_t, r_c) \in \Sigma$.

On appelle *G-étiquetage* une dérivation d'un G-séquent obtenue à partir des règles d'inférence suivantes :

$$\frac{\langle s, A\rangle \in Lex}{\vdash_G (\epsilon, s, \epsilon) : A}\,[Lex]$$

$$\frac{x \in V}{x : A \vdash_G (\epsilon, x, \epsilon) : A}\,[axiome]$$

$$\frac{\Gamma \vdash_G r_1 : A \,/\, B \quad \Delta \vdash_G r_2 : B \quad Var(r_1) \cap Var(r_2) = \emptyset}{\langle \Gamma; \Delta\rangle \vdash_G (r_{1s}, r_{1t}, r_{1c} \bullet Concat(r_2)) : A}\,[/_e]$$

$$\frac{\Delta \vdash_G r_2 : B \quad \Gamma \vdash_G r_1 : B \setminus A \quad Var(r_1) \cap Var(r_2) = \emptyset}{\langle \Gamma; \Delta\rangle \vdash_G (Concat(r_2) \bullet r_{1s}, r_{1t}, r_{1c}) : A}\,[\backslash_e]$$

$$\frac{\Gamma \vdash_G r_1 : A \otimes B \quad \Delta[x : A, y : B] \vdash_G r_2 : C \quad Var(r_1) \cap Var(r_2) = \emptyset \quad A \in \text{P}_2}{\Delta[\Gamma] \vdash_G r_2[Concat(r_1)/x, \epsilon/y] : C}\,[\otimes_e]$$

$$\frac{\Gamma \vdash_G r : A \quad \Gamma' \sqsubset \Gamma}{\Gamma' \vdash_G r : A}\,[\sqsubset]$$

On peut remarquer qu'un G-étiquetage est en fait un arbre de preuve de la logique minimaliste sur lequel les hypothèses des séquents sont décorées par des variables et les conclusions des séquents sont décorées par des étiquettes.

Si $\Gamma \vdash_G r : B$ est un G-séquent dérivable, alors r est linéaire, et $Var(r)$ est exactement l'ensemble des variables déclarées dans Γ, enfin, pour tout renommage σ, $\Gamma.\sigma \vdash_G r.\sigma : B$ est un G-séquent dérivable.

Les opérations *merge* et *move*

Dans les GMCs, les opérations de fusion et de déplacement sont simulées par des combinaisons de règles de la logique minimaliste produisant des G-étiquetages.

Merge est l'élimination de / (*resp.* \) suivie d'une règle *d'entropie*. Dans le calcul sur les étiquettes, cela correspond au passage en position de complément (*resp.* spécifieur) de l'une des étiquettes dans l'autre. On remarquera que les merges basées sur / sont nécessairement réalisées à partir d'entrée lexicale. Elles ne contiennent donc pas de contexte.

$$\frac{\dfrac{\vdash (r_{spec}, r_{tete}, r_{comp}) : A \,/\, B \quad \Delta \vdash s : B}{\Delta \vdash (r_{spec}, r_{tete}, r_{comp} \bullet Concat(s)) : A}\,[/_e]}{\Delta \vdash (r_{spec}, r_{tete}, r_{comp} \bullet Concat(s)) : A}\,[\sqsubset]$$

$$\frac{\dfrac{\Delta \vdash s : B \quad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus A}{\langle \Delta; \Gamma \rangle \vdash (Concat(s) \bullet r_{spec}, r_{tete}, r_{comp}) : A}\,[\setminus_e]}{\Delta, \Gamma \vdash (Concat(s) \bullet r_{spec}, r_{tete}, r_{comp}) : A}\,[\sqsubset]$$

Afin de simplifier les notations, on utilisera uniquement l'ordre commutatif (,), marqué seulement par la virgule ','. Cette combinaison de règles est notée $[mg]$.

Move est l'élimination d'un tenseur sur les formules et correspond à la substitution des variables associées pour le calcul phonologique.

$$\frac{\Gamma \vdash r_1 : A \otimes B \quad \Delta[u : A, v : B] \vdash r_2 : C}{\Delta[\Gamma] \vdash r_2[Concat(r_1)/u, \epsilon/v] : C}\,[\otimes_e]$$

Cette règle ne s'applique que dans les cas où $A \in \text{P}_2$ et B est de la forme $B_1 \times \ldots B_n \times D$ où $B_i \in \text{P}_2$ et $D \in \text{P}_1$.

On note $[mv]$ le déplacement. Il utilise pleinement la notion de ressources proposée par les hypothèses du calcul. Il s'agit de positionner ces ressources dans la preuve (c'est la contrepartie des assignateurs) puis d'y substituer le constituant qui s'y combine. Ainsi l'hypothèse de P_1 représente l'entrée dans la dérivation du constituant par sa catégorie de base et les hypothèses de P_2 représentent les positions où il est "déplacé". Cependant, un constituant déplacé n'intervient réellement dans la dérivation que lorsque toutes ses contributions prévues ont été marquées par des hypothèses.

Ainsi, le système de dérivation des GMCs est résumé par l'ensemble de règles suivant :

$$\frac{\langle s, A \rangle \in Lex}{\vdash_G (\epsilon, s, \epsilon) : A}\,[Lex] \qquad \frac{\vdash (r_{spec}, r_{tete}, r_{comp}) : A \,/\, B \quad \Delta \vdash s : B}{\Delta \vdash (r_{spec}, r_{tete}, r_{comp} \bullet Concat(s)) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \quad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus A}{\Delta, \Gamma \vdash (Concat(s) \bullet r_{spec}, r_{tete}, r_{comp}) : A}\,[mg]$$

$$\frac{\Gamma \vdash r_1 : A \otimes B \quad \Delta[u : A, v : B] \vdash r_2 : C}{\Delta[\Gamma] \vdash r_2[Concat(r_1)/u, \epsilon/v] : C}\,[mv]$$

L'ensemble $\mathbb{D}_G$ des dérivations reconnues par une GMC G est l'ensemble des preuves obtenues à l'aide de ces règles et dont le séquent conclusion est $\vdash r : C$. Le langage reconnu par G est $L(G) = \{Concat(r) | \vdash r : C \in \mathbb{D}_G\}$.

Utilisation et extensions des opérations *merge* et *move*

Dans cette section, nous exposons le fonctionnement des GMCs. Nous expliquons en particulier la nécessité d'utiliser systématiquement la règle d'*entropie*. Nous montrons comment utiliser les règles des GMCs pour rendre compte du mouvement cyclique. Nous illustrons sur un exemple l'utilisation de ces grammaires. Finalement, nous proposons des extensions des GMCs pour modéliser les opérations de déplacement fort/faible, *Head-Movement* et *Affix-Hopping*.

L'utilisation systématique de la règle d'entropie après la fusion permet d'obtenir l'ordre commutatif entre toutes les hypothèses de la preuve. Si plusieurs hypothèses correspondant à des constituants différents s'intercalent dans la preuve, elles n'empêcheront pas les éliminations de se produire. Par exemple, la preuve suivante n'utilise pas la règle d'entropie :

$$\dfrac{F \vdash F \qquad \dfrac{A \vdash A \qquad \dfrac{\vdash A \setminus F \setminus C \,/\, E \quad E \vdash E}{E \vdash A \setminus F \setminus C}\,[\setminus_e]}{A; E \vdash F \setminus C}\,[\setminus_e]}{F; A; E \vdash C}\,[\setminus_e]$$

Il n'est pas possible de déclencher une élimination de produit à partir de la formule $\vdash E \otimes F$. En effet, la non-commativité du contexte $F; A; E$ nous en empêche. Or, il serait désirable pour des raisons linguistiques de pouvoir le faire. Par exemple, pour l'analyse d'une phrase où $A \setminus F \setminus C \,/\, E$ est la formule que la GMC associe à un verbe, E correspond au DP objet, A au DP sujet et F est le cas de l'objet. Supposons qu'une dérivation nous donne un DP ayant le bon cas, la formule qui lui sera associée sera nécessairement de la forme $F \otimes E$. Pour combiner les deux dérivations ensemble, c'est-à-dire donner pour complément ce DP au verbe, il nous faut éliminer cette formule. Ce type de situation peut apparaître fréquemment dans les dérivations des GMCs car les hypothèses correspondant à un constituant ne sont pas systématiquement introduites à proximité les unes des autres. Cela explique pourquoi nous utilisons le produit commutatif et la règle d'entropie dans les GMCs.

Pour le cas de *déplacements cycliques* (où un constituant se déplace plusieurs fois), nous relions les hypothèses intervenant pour un même constituant dès leur introduction dans la preuve. Pour cela, lorsqu'une nouvelle hypothèse A est introduite, on utilise un $[mv]$ avec un séquent hypothèse $A \otimes B \vdash A \otimes B$ où A appartient à P_2 et B est de la forme $B_1 \otimes \ldots \otimes B_n \otimes D$ où $B_i \in \mathrm{P}_2$ et $D \in \mathrm{P}_1$.

$$\dfrac{x : A \otimes B \vdash (\epsilon, x, \epsilon) : A \otimes B \quad \Delta[u : A, v : B] \vdash r : C}{\Delta[A \otimes B] \vdash r[x/u, \epsilon/v] : C}\,[\otimes_e]$$

Un exemple des premières étapes de la dérivation de la phrase : “les enfants lisent un livre” est présenté dans la figure 40.

Avant de commenter la dérivation, il nous faut remarquer que les formules des DPs sont $k \otimes d$ où $k \in \mathrm{P}_2$ et $d \in \mathrm{P}_1$. On a bien l'une des sous-formules dénotant de la catégorie DP

$$\dfrac{\dfrac{\vdash (\epsilon, un, livre) : k \otimes d \qquad \dfrac{\dfrac{\vdash (\epsilon, li, \epsilon) : (k \setminus (d \setminus v)) \,/\, d \quad u : d \vdash (\epsilon, u, \epsilon) : d}{v : k \vdash (\epsilon, v, \epsilon) : k \qquad u : d \vdash (\epsilon, li, u) : k \setminus d \setminus v}\,[mg]}{v : k, u : d \vdash (v, li, u) : d \setminus v}\,[mg]}{\vdash (un\ livre, li, \epsilon) : d \setminus v}\,[mv]$$

$$\dfrac{\vdash (\epsilon, -sent, \epsilon) : (k \setminus t) /_< v \qquad \dfrac{w : d \vdash (\epsilon, w, \epsilon) : d \vdash (un\ livre, li, \epsilon) : d \setminus v}{w : d \vdash (w\ un\ livre, li, \epsilon) : v}\,[mg]}{w : d \vdash (\epsilon, lisent, w\ un\ livre) : k \setminus t}\,[mg]$$

Fig. 40 – Premières étapes de l'analyse de la phrase : "Les enfants lisent un livre".

et la seconde de ses propriétés (doit recevoir un cas). Dans la GM, le verbe se combine par fusion avec le *DP*, puis la dérivation déclenche un déplacement. Dans la GMC, le verbe se combine par fusion avec une hypothèse d (marquant la position d'entrée dans la dérivation du constituant) puis avec une hypothèse k (marquant la position de déplacement du constituant). La présence de ces deux hypothèses implique qu'elles doivent être déchargées. On les utilise immédiatement avec $[mv]$ sur le séquent $\vdash (\epsilon, un, livre) : (k \otimes d)$.

Puis, dans la GM, cette dérivation fusionne avec le *DP* qui prendra la position de sujet, soit dans la GMC un *merge* avec une nouvelle hypothèse de type d. Enfin, le verbe reçoit son inflexion par *Head-Movement*. La dérivation de la figure 40 s'arrête à ce point. La suite consiste en l'introduction d'une nouvelle hypothèse relative au cas (nominatif) qui permet le déchargement simultané de k et de d par introduction dans la dérivation du *DP* sujet. La dernière étape de *complementizer* est à nouveau utilisée via une entrée $c \,/\, t$, c'est-à-dire transformant la dérivation au stade t en une dérivation acceptante.

On remarque une grande similarité entre les deux formalismes, à ceci près que dans les GMCs, les constituants ne se déplacent pas mais utilisent des hypothèses marquant les places où ils laissent une trace. Ce formalisme utilise donc les propriétés de commutativité de la logique mixte et voit les hypothèses comme des ressources. La section 6.3 présente trois exemples complets de dérivations obtenues par des GMCs.

Les déplacements fort/faible peuvent être simulés par la localisation de la substitution qui est déterminée par l'appartenance des hypothèses à P_1 ou à P_2.

$$\dfrac{s : \Gamma \vdash A \otimes B \quad r[u, v] : \Delta[u : A, v : B] \vdash C}{r[Concat(s)/u, \epsilon/v] : \Delta[\Gamma] \vdash C}\,[move_{fort}]$$

$$\dfrac{s : \Gamma \vdash A \otimes B \quad r[u, v] : \Delta[u : A, v : B] \vdash C}{r[\epsilon/u, Concat(s)/v] : \Delta[\Gamma] \vdash C}\,[move_{faible}]$$

Cette version des déplacements fort/faible est un peu différente de celle de la première version des GMs proposée par Stabler [Sta97], mais se rapproche davantage de celle développée dans les extensions des GMs plus récentes comme [Kob06].

Comme nous l'avons vu pour les GMs, l'opération *merge* est raffinée par des décorations $<$ et $>$. Nous avons également introduit ces décorations dans les GMCs, ce qui permet de définir les opération de *Head-Movement* et *Affix-Hopping*.

Head-Movement :

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A /_{<} B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, r_{tete} \bullet s_{tete}, r_{comp} \bullet Concat(s_{-tete})) : A}\,[mg]$$

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A_{>} / B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, s_{tete} \bullet r_{tete}, r_{comp} \bullet Concat(s_{-tete})) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B_{>} \setminus A}{\Delta, \Gamma \vdash (Concat(s_{-tete}) \bullet r_{spec}, s_{tete} \bullet r_{tete}, r_{comp}) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus_{<} A}{\Delta, \Gamma \vdash (Concat(s_{-tete}) \bullet r_{spec}, r_{tete} \bullet s_{tete}, r_{comp}) : A}\,[mg]$$

Affix-Hopping :

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A /_{>} B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, \epsilon, r_{comp} \bullet Concat((s_{spec}, r_{tete} \bullet s_{tete}, s_{comp}))) : A}\,[mg]$$

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A_{<} / B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, \epsilon, r_{comp} \bullet Concat((s_{spec}, s_{tete} \bullet r_{tete}, s_{comp}))) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B_{<} \setminus A}{\Delta, \Gamma \vdash (Concat(s_{spec}, s_{tete} \bullet r_{tete}, s_{comp}) \bullet r_{spec}, \epsilon, r_{comp}) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus_{>} A}{\Delta, \Gamma \vdash (Concat(s_{spec}, r_{tete} \bullet s_{tete}, s_{comp}) \bullet r_{spec}, \epsilon, r_{comp}) : A}\,[mg]$$

Ces règles sont le pendant direct de celles de *Head-Movement* et d'*Affix-Hopping* des GMs. Il est aisé de remarquer qu'elles reconnaissent les mêmes chaînes et construisent les mêmes structures d'arbre.

Ce type de dérivation ne conserve pas la relation de projection (et donc de spécifieur, de tête et de complément), ce qui implique de réintroduire ces principes sur les chaînes pour ces dernières opérations (utilisation de triplets de chaînes). Cependant, on remarquera que la tête d'une dérivation est en fait la formule principale, et donc pas extension, la projection maximale est alors la preuve dont une entrée est la formule principale. Les spécifieurs sont simplement les éléments à la gauche d'une tête et les compléments ceux à sa droite.

6.2 GM et GMCs

Nous allons revenir sur les relations entre GMC et GM. Dans un premier temps, nous montrons comment transformer une GMC en GM et réciproquement. Puis nous montrerons que ces transformations préservent les langages engendrés par ces grammaires.

6.2.1 Traduction de lexiques

Les GMs et les GMCs sont lexicalisées, ce qui a pour conséquence que les grammaires sont entièrement définies par leur lexique. Le passage d'une GMC $\langle N,\text{P}, Lex, \{[mg], [mv]\}, C\rangle$ à une GM donne une grammaire de la forme $\langle N,\text{P}, Lex', \{merge, move\}, C\rangle$. Il suffit donc de définir les transformations entre Lex et Lex' pour spécifier comment transformer une GMC en GM et réciproquement.

Nous définissons d'abord comment passer du lexique d'une GM à celui d'une GMC. Dans cette définition, γ est une séquence de traits et f un trait de la GM. On distinguera son type par les marqueurs des GMs $(=, +, -)$.

On appelle $MtoC$ la fonction de passage d'un lexique d'une GM à celui d'une GMC.

$MtoC(Lex) = \{mtoc(\sigma :: \gamma) | \sigma :: \gamma \in Lex\}$

$mtoc(\sigma :: \gamma) = \langle \sigma, (\gamma)^{\alpha}\rangle$, où la transformation $(\cdot)^{\alpha}$ est définie par le système d'équations suivant :

$$\begin{array}{lcl}
(\gamma\ -f)^{\alpha} & = & (f \otimes (\gamma)^{\delta}) \\
(=f\ \gamma)^{\alpha} & = & (\gamma)^{\beta}\ /\ f \\
(f)^{\alpha} & = & f \\
(=f\ \gamma)^{\beta} & = & f \setminus (\gamma)^{\beta} \\
(+f\ \gamma)^{\beta} & = & f \setminus (\gamma)^{\beta} \\
(\gamma\ -f)^{\beta} & = & (f \otimes (\gamma)^{\delta}) \\
(f)^{\beta} & = & f \\
(\gamma\ -f)^{\delta} & = & (f \otimes (\gamma)^{\delta}) \\
(f)^{\delta} & = & f
\end{array}$$

De manière analogue on définit la transformation réciproque.

On appelle $CtoM$ la fonction de passage d'un lexique d'une GMC à celui d'une GM.

$CtoM(Lex) = \{ctom(\langle \sigma, F\rangle) | \langle \sigma, F\rangle \in Lex\}$

$ctom(\langle \sigma, F\rangle) = \sigma :: (\gamma)^{\alpha'}$, où la transformation $(\cdot)^{\alpha'}$ est définie par le système d'équations suivant :

$$\begin{array}{lcl}
(F\ /\ f)^{\alpha'} & = & =f\ (F)^{\beta'} \\
(f \otimes F)^{\alpha'} & = & (F)^{\delta'}\ -f \\
(f)^{\alpha'} & = & f \\
(f \setminus F)^{\beta'} & = & \left\{ \begin{array}{l} =f\ (F)^{\beta'} si\ f \in p_1 \\ +f\ (F)^{\beta'} si\ f \in p_2 \end{array} \right. \\
(f)^{\beta'} & = & f \\
(f \otimes F)^{\beta'} & = & (F)^{\delta'}\ -f \\
(f \otimes F)^{\delta'} & = & (F)^{\delta'}\ -f \\
(f)^{\delta'} & = & f
\end{array}$$

On remarquera que $CtoM$ et $MtoC$ sont des fonctions réciproques.

6.2.2 Inclusion GM/GMC

Nous montrons l'inclusion du langage engendré par G, une GM sans SMC, dans le langage engendré par G' la GMC telle que $G' = MtoC(G)$. Pour cela, nous introduisons les *structures dérivées alternatives* pour les GMs ainsi que des fonctions de fusion et déplacement pour ces nouvelles structures. Nous introduisons cette notion afin de donner

une représentation enrichie des arbres dérivés des GMs. Nous exhiberons un mécanisme qui permet de passer de ces structures aux arbres dérivés des GMs. Composées avec ce mécanisme les opérations de fusion et de déplacement sur les structures dérivées alternatives s'interprètent par des fusions et des déplacements sur les arbres dérivés qu'elles représentent et réciproquement. D'autre part, nous introduisons une notion de *preuves éclatées* pour les GMCs, notion qui est très similaire à celle de structures dérivées alternatives pour les GMs. Un mécanisme, similaire à celui qui permet de passer des structures dérivées alternatives aux arbres dérivés, permet de passer de preuves éclatées à des preuves. Cela nous permettra de conclure que le langage reconnu par la GM G est inclus dans le langage engendré par $MtoC(G)$.

Les *structures dérivées alternatives* sont de la forme $\langle t, S\rangle$ où $t \in T_{MG}(V)$, ($T_{MG}(V)$ est l'ensemble des arbres minimalistes dont les feuilles peuvent être également des variables de V) et $S \subseteq V^+ \times T_{MG}(V)$. Intuitivement t, l'*arbre principal* de la structure, représente un arbre dérivé duquel les constituants ont été enlevés pour être stockés dans S. S peut alors être interprétée comme une substitution qui permet de retrouver l'arbre dérivé représenté. Pour $S \subseteq V^+ \times T_{MG}$ (*resp.* $L \in V^+$), on note $Var(S)$ (*resp.* $Var(L)$) l'ensemble des variables de V ayant une occurrence dans S (*resp.* L). Par ailleurs, nous noterons $S_1 \uplus S_2$ pour dénoter l'union de S_1 et de S_2 lorsque S_1 et S_2 sont disjoints.

Pour t et t_1 appartenant à $T_{MG}(V)$ et $x \in Var(t)$, on note $t[x := t_1]$ la substitution dans t de la variable x par t_1. De plus, on notera σ_L^λ la substitution qui associe λ à chaque variable de la séquence de variables L.

Nous définissons comment les structures dérivées alternatives sont construites à partir des items lexicaux, puis les opérations de fusion et de déplacement, $merge'$ et $move'$, sur ces structures :

1. $\sigma :: \gamma \rightarrow \langle \sigma :: \gamma, \emptyset\rangle$, pour $\sigma :: \gamma \in Lex$
2. $merge'(\langle \sigma_1 :: {=}d\gamma_1, S_1\rangle, \langle C_2[\sigma_2 :: d], S_2\rangle) = \langle < (\sigma_1 :: \gamma_1, C_2[\sigma_2]), S_1 \uplus S_2\rangle$, si $Var(S_1) \cap Var(S_2) = \emptyset$.
3. $merge'(\langle \sigma_1 :: {=}d\gamma_1, S_1\rangle, \langle C_2[\sigma_2 :: d\gamma_2], S_2\rangle) = \langle < (\sigma_1 :: \gamma_1, x), S_1 \uplus S_2 \uplus \{x, C_2[\sigma_2 :: \gamma_2]\}\rangle$, si $Var(S_1) \cap Var(S_2) = \emptyset$, $x \notin Var(S_1) \cup Var(S_2)$ et $\gamma_2 \neq \epsilon$.
4. $merge'(\langle C_1[\sigma_1 :: {=}d\gamma_1], S_1\rangle, \langle C_2[\sigma_2 :: d], S_2\rangle) = \langle > (C_2[\sigma_2, C_1[\sigma_1 :: \gamma_1]], S_1 \uplus S_2\rangle$, si $Var(S_1) \cap Var(S_2) = \emptyset$, $C_1[x_1] \neq x_1$.
5. $merge'(\langle C_1[\sigma_1 :: {=}d\gamma_1], S_1\rangle, \langle C_2[\sigma_2 :: d\gamma_2], S_2\rangle) = \langle > (x, C_1[\sigma_1 :: \gamma_1]), S_1 \uplus S_2 \cup \{x, C_2[\sigma_2 :: \gamma_2]\}\rangle$, si $Var(S_1) \cap Var(S_2) = \emptyset$, $C_1[x_1] \neq x_1$ et $x \notin Var(S_1) \cup Var(S_2)$ et $\gamma_2 \neq \epsilon$.
6. $move'(\langle C_1[\sigma_1 :: +d\gamma_1], S \uplus \{\langle L, C_2[\sigma_2 :: -d\gamma_2]\rangle\}\rangle) = \langle > (x, C_1[\sigma_1 :: \gamma_1]), S \uplus \{\langle xL, C_2[\sigma_2 :: \gamma_2]\rangle\}\rangle)$, $x \notin Var(S)$ et $\langle L, C_2[\sigma_2 :: -d\gamma_2]\rangle \in S$.

Remarquons que $move'$ n'est pas déterministe, le résultat dépend du choix de l'élément déplacé. Si $\langle t', S'\rangle$ est obtenu par déplacement à partir de $\langle t, S\rangle$, nous noterons que $\langle t', S'\rangle \in move'(\langle t, S\rangle)$.

On introduit une relation notée $<_\tau$ sur les couples de liste de variables et d'arbre minimaliste avec variables. Si $\beta_1 = \langle L_1, t_1\rangle$ et $\beta = \langle L_2, t_2\rangle$ alors $\beta_1 <_\tau \beta_2$ s'il existe $x \in L_2$ $x \in Var(t_1)$.

Si $\langle t, S\rangle$ est une structure dérivée alternative obtenue seulement à partir d'entrées lexicales et des opérations $merge'$ et $move'$, alors on peut montrer facilement que la relation

$<_\tau$ sur $S' = S \cup \{\langle x, t\rangle\}$ (où $x \notin Var(S)$) définit un graphe acyclique dirigé (DAG) ; c'est-à-dire que $<_\tau$ est acyclique : il n'existe pas de suite d'éléments de S', $\alpha_1, \ldots, \alpha_n$ telle que $n > 1$, $\alpha_1 <_\tau \alpha_2 <_\tau \cdots <_\tau \alpha_n$ et $\alpha_n = \alpha_1$; et connexe : pour tout élément $\alpha \in S'$, il existe une suite d'éléments $\alpha_1, \ldots, \alpha_n$ telle que $\alpha_1 = \langle x, t\rangle$ et $\alpha_n = \alpha$. Cela nous garantit que si $Card(S') > 1$ alors il existe toujours $\langle L_1, t_1\rangle$ et $\langle L_2, t_2\rangle$ dans S' tels que $Var(L_2) \subseteq Var(t_1)$.

Nous nous servons de ces propriétés pour définir une relation de *reconstruction* de l'arbre dérivé des GMs à partir d'une structure dérivée alternative. Cette relation est notée $\rhd_\tau$ et elle est définie par :

$S \uplus \{\langle L_1, t_1\rangle; \langle xL_2, t_2\rangle\} \rhd_\tau (S \cup \{\langle L_1, t_1[x := t_2].\sigma^{\lambda}_{L2}\rangle\})$ si $Var(xL_2) \subseteq Var(t_1)$.

On note $\rhd^*_\tau$ la clôture transitive réflexive de $\rhd_\tau$. On peut voir facilement que si $S_1 \rhd_\tau S_2$ et que $<_\tau$ induit une structure de DAG sur S_1, alors il en est de même sur S_2. De plus, il est évident que $\rhd_\tau$ est confluente. En conséquence, si $\langle t, S\rangle$ est une structure dérivée alternative obtenue à partir du lexique et des opérations $merge'$ et $move'$, en appelant S' l'ensemble $S \cup \{\langle x, t\rangle\}$ avec $x \notin Var(S)$, on obtient alors que $S' \rhd^*_\tau \{\langle x, t'\rangle\}$ où t' est un arbre minimaliste sans variable. On appelle t' l'*arbre minimaliste*[2] *associé à la structure alternative* $\langle t, S\rangle$, que l'on note $T_{min}(\langle t, S\rangle)$. Il est facile de montrer par induction sur les arbres de dérivation que *merge* et *move* peuvent être obtenues par composition de T_{min} avec $merge'$ et $move'$, ce qui se traduit des deux propriétés suivantes :

1. $T_{min}(merge'(\langle t_1, C_1\rangle, \langle t_2, C_2\rangle)) = merge(T_{min}(\langle t_1, C_1\rangle, T_{min}(\langle t_2, C_2\rangle)))$,
2. si $\langle t', S'\rangle \in move'(\langle t, S\rangle)$ alors $T_{min}(\langle t', S'\rangle)) \in move(T_{min}(\langle t, S\rangle))$.

Réciproquement, on peut montrer que si $u_1 = T_{min}(\langle t_1, S_1\rangle)$ et $u_2 \in move(u_1)$ alors il existe $\langle t_2, S_2\rangle \in move'(\langle t_1, S_1\rangle)$ tel que $T_{min}(\langle t_2, S_2\rangle) = u_2$. Cela finit de montrer que les structures dérivées alternatives et leurs opérations de fusion et de déplacement simulent exactement les GMs. Ces structures alternatives sont très proches des *preuves éclatées* que nous allons à présent introduire. Nous pouvons ainsi déduire l'inclusion du langage de G dans celui de G' en montrant que les preuves éclatées et les structures alternatives sont similaires.

À présent, nous définissons les *preuves éclatées* pour les GMCs. On notera $\mathcal{P}$ l'ensemble des preuves de G'. Les preuves éclatées sont de la forme $\langle \mathcal{D}, R\rangle$ où $\mathcal{D} \in \mathcal{P}$ et $R \subseteq V^+ \times \mathcal{P}$. On appelle $\mathcal{D}$ la *preuve principale* de la dérivation. Les parties de preuve ayant trait au déplacement sont stockées dans l'ensemble R et associées aux variables qui marquent les hypothèses correspondant à leurs déplacements. On remarquera une très grande similarité entre les preuves éclatées et les structures dérivées alternatives. Pour $R \subseteq V^+ \times \mathcal{P}$, on note $Var(R)$ l'ensemble des variables de V appartenant à $\mathcal{P}$. Nous écrirons $\mathcal{D} :: \Gamma \vdash r : A$ pour référer à une dérivation $\mathcal{D}$ du séquent $\Gamma \vdash r : A$ dans une GMC[3].

Nous définissons comment les preuves éclatées sont construites à partir des items lexicaux, puis les opérations de fusion et de déplacement, $merge''$ et $move''$, sur ces structures :

1. $\sigma :: \gamma \rightarrow \left\langle \dfrac{mtoc(\sigma :: \gamma) \in Lex'}{\vdash (\epsilon, \sigma, \epsilon) : (\gamma)^\alpha}, \emptyset \right\rangle$
2. $merge''(\langle \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : C/A, R_1\rangle, \langle \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : A, R_2\rangle) = \left\langle \dfrac{\mathcal{D}_1 \quad \mathcal{D}_2}{\Gamma_1, \Gamma_2 \vdash r_1 \bullet r_2 : C}, R_1 \cup R_2 \right\rangle$,

[2] Cet arbre est défini de manière unique car la relation $\rhd_\tau$ est confluente.

[3] Attention : $\mathcal{D} :: \Gamma \vdash r : A$ et $\mathcal{D}$ dénotent du même objet

si A est une formule atomique, $Var(R_1) \cap Var(R_2) = \emptyset$ et $x \notin Var(R_1) \cup Var(R_2)$

3. $merge''(\langle \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : C/A, R_1\rangle, \langle \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : B \otimes A, R_2\rangle) =$
$$\left\langle \frac{\mathcal{D}_1 \quad x : A \vdash (\epsilon, x, \epsilon) : A}{\Gamma_1, x : A \vdash r_1 \bullet x : C}, R_1 \cup R_2 \cup \{\langle x, \mathcal{D}_2\rangle\} \right\rangle,$$
si $Var(R_1) \cap Var(R_2) = \emptyset$ et $x \notin Var(R_1) \cup Var(R_2)$

4. $merge''(\langle \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : A\backslash C, R_1\rangle, \langle \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : A, R_2\rangle) =$
$$\left\langle \frac{\mathcal{D}_2 \quad \mathcal{D}_1}{\Gamma_2, \Gamma_1 \vdash r_2 \bullet r_1 : C}, R_1 \cup R_2 \right\rangle,$$
si A est une formule atomique,$Var(R_1) \cap Var(R_2) = \emptyset$ et $x \notin Var(R_1) \cup Var(R_2)$

5. $merge''(\langle \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : A\backslash C, R_1\rangle, \langle \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : B \otimes A, R_2\rangle) =$
$$\left\langle \frac{x : A \vdash (\epsilon, x, \epsilon) : A \quad \mathcal{D}_1}{x : A, \Gamma_1 \vdash x \bullet r_1 : C}, R_1 \cup R_2 \cup \{\langle x, \mathcal{D}_2\rangle\} \right\rangle,$$
si $Var(R_1) \cap Var(R_2) = \emptyset$ et $x \notin Var(R_1) \cup Var(R_2)$

6. $move''(\langle \mathcal{D}_1 :: \Gamma_1 \vdash r : B_k\backslash C, L \uplus \{\langle x_k x_{k-1} \ldots x_1 y, \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : B_n \otimes \ldots \otimes B_k \otimes \ldots \otimes B_1 \otimes A\rangle\} = \left\langle \frac{x_k : B_k \vdash (\epsilon, x_k, \epsilon) : B_k \quad \mathcal{D}_1}{x_k : B_k, \Gamma_1 \vdash x_k \bullet r : C}, L \uplus \{\langle x_k x_{k-1} \ldots x_1 y, \mathcal{D}_2\rangle\} \right\rangle$

Remarquons que $move''$ n'est pas déterministe, le résultat dépend du choix de l'élément déplacé. Si $\langle \mathcal{D}', R'\rangle$ est obtenu par déplacement à partir de $\langle \mathcal{D}, R\rangle$, nous noterons que $\langle \mathcal{D}', R'\rangle \in move''(\langle \mathcal{D}, R\rangle)$.

On introduit une relation notée $<_\pi$ sur les couples listes de variables preuves des preuves éclatées. Si $\alpha_1 = (L_1, \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : a_1)$ et $\alpha_2 = (L_2, \mathcal{D}_2 :: \Gamma_2 \vdash r_2 : a_2)$ alors $\alpha_1 <_\pi \alpha_2$ s'il existe $x \in L_2$ et $x \in Var(r_1)$.

De même que pour les structures dérivées alternatives, si $\langle \mathcal{D}, R\rangle$ est une preuve éclatée obtenue seulement à partir d'entrées lexicales et des opérations $merge''$ et $move''$, alors on peut montrer facilement que la relation $<_\pi$ sur $R' = R \cup \{\langle x, p\rangle\}$ (où $x \notin Var(R)$) définit un DAG.

Pour définir qu'un déplacement relatif à une liste de variables $x_n \ldots x_1 y$ est réalisable, on définit l'opération $[mv; x_n \ldots x_1 y]$ qui est l'opération $move$ à partir d'un séquent qui est une séquence de $\otimes$ dont les formules de base sont associées à des variables de $x_n \ldots x_1 y$. On appelle $D = B_n \otimes \ldots \otimes B_1 \otimes A$ représentant l'ensemble des hypothèses utilisées dans la substitution.

$$\dfrac{\dfrac{\overline{z_1 : B_1 \otimes A \vdash z_1 : B_1 \otimes A} \quad \mathcal{D}_1 :: \Gamma, x_n : B_n, \ldots, x_1 : B_1, y : A \vdash r_1 : C}{\Gamma, x_n : B_n, \ldots, x_2 : B_2, z_1 : B_1 \otimes A \vdash z_1 \bullet r_1 : C} \\ \vdots \\ \dfrac{\overline{z_n : D \vdash z_n : D} \qquad \Gamma, x_n : B_n, z_{n-1} : B_{n-1} \otimes \ldots \otimes B_1 \otimes A \vdash r_1[z_{n-1}] : C}{\Gamma, z_n : D \vdash r_1[z_n/z_{n-1}] : C}$$

On définit une relation de reconstruction d'une preuve d'une GMC à partir de la structure de preuve éclatée. Cette relation est notée $\rhd_\pi$ et elle est définie par la règle, où $D = B_n \otimes \ldots \otimes B_1 \otimes A$:

$$S \uplus \{\langle x_1 \dots x_n y, \mathcal{D}_1 :: \Gamma_1 \vdash r_1 : D\rangle\} \uplus \{\langle L_2, \mathcal{D}_2 :: \Gamma_2, x_n : B_n, \dots, x_1 : B_1, y : A \vdash r_2 : C\rangle\} \rhd_\pi S \cup \left\{ \left(L_2, \quad \frac{\begin{array}{cc} \mathcal{D}_1 & \Gamma_2, x_n : B_n, x' : B_{n-1} \otimes \dots \otimes A \vdash r : C \end{array}}{\Gamma_1, \Gamma_2 \vdash r[Concat(r_1)/x_n, \epsilon/x'] : C} \begin{array}{c} \mathcal{D}_2 \\ \vdots \\ [mv; x_{n-1} \dots x_1 y] \\ [mv] \end{array} \right) \right\}$$

On note $\rhd_\pi^*$ la clôture transitive réflexive de $\rhd_\pi$.

Les preuves ainsi obtenues à partir de $\rhd_\pi^*$, sont les preuves des GMCs, qui ne sont pas sous forme normale. En effet, la nécessité de retarder les éliminations de produit en fin de dérivation fait qu'elles ne sont pas à leur place. Cependant, en utilisant la preuve de normalisation du chapitre précédent, on peut aisément calculer les preuves sous forme normale. On remarquera que pour toute preuve d'une GMC, il est aisé de construire la preuve éclatée lui correspondant. La différence se situe sur les constituants devant se déplacer qui ne seront connus qu'en cours de dérivation.

Étant donné $t \in T_{MG}(V)$ et une preuve $\mathcal{D}$ d'un séquent $\Gamma \vdash r : C$ d'une GMC, on dit que $t \approx_{str} \mathcal{D}$ si $Concat(r)$ est égal à la concaténation des feuilles de t prises de gauche à droite.

Étant donnée une structure dérivée alternative $\langle t, S\rangle$ et une preuve éclatée $\langle \mathcal{D}, R\rangle$, on dit qu'elles représentent la *même* dérivation, ce que l'on note $\langle t, S\rangle \approx \langle \mathcal{D}, R\rangle$, si

1. $t \approx_{str} \mathcal{D}$
2. pour tout $(L, t') \in S$, il existe $(L, \mathcal{D}') \in T$ tel que $t' \approx_{str} \mathcal{D}'$
3. pour tout $(L, \mathcal{D}') \in R$, il existe $(L, t') \in T$ tel que $t' \approx_{str} \mathcal{D}'$

Les propriétés suivantes peuvent être établies aisément :

1. $\langle \sigma :: \gamma, \emptyset\rangle \approx \left\langle \dfrac{mtoc(\sigma :: \gamma) \in Lex'}{\vdash (\epsilon, \sigma, \epsilon) : (\gamma)^\alpha}, \emptyset \right\rangle$
2. $Si \langle t_1, S_1\rangle \approx \langle \mathcal{D}_1, R_1\rangle$ et $\langle t_2, S_2\rangle \approx \langle \mathcal{D}_2, R_2\rangle$ alors
   $merge'(\langle t_1, S_1\rangle, \langle t_2, S_2\rangle) \approx merge''(\langle \mathcal{D}_1, R_1\rangle, \langle \mathcal{D}_2, R_2\rangle)$
3. $Si \langle t, S\rangle \approx \langle \mathcal{D}, R\rangle$ alors :
   (a) pour tout $\langle t', S'\rangle \in move'(\langle t, S\rangle)$, il existe $\langle \mathcal{D}', R'\rangle \in move''(\langle \mathcal{D}, R\rangle)$ tel que $\langle t', S'\rangle \approx \langle \mathcal{D}', R'\rangle$
   (b) pour tout $\langle \mathcal{D}', R'\rangle \in move''(\langle \mathcal{D}, R\rangle)$, il existe $\langle t', S'\rangle \in move'(\langle t, S\rangle)$ tel que $\langle t', S'\rangle \approx \langle \mathcal{D}', R'\rangle$

Pour une structure dérivée alternative $\langle t, S\rangle$ et une preuve éclatée $\langle \mathcal{D}, R\rangle$, construites simultanément (à partir des mêmes entrées lexicales et des mêmes séquences d'opérations), on obtient $\langle t, S\rangle \approx \langle \mathcal{D}, R\rangle$. Par extension à la grammaire, on a donc $L(G) \subseteq L(CtoM(G))$. En remarquant que toutes les preuves des $CtoM(G)$ peuvent être dérivées en une preuve étendue, et inversement, à partir de toute preuve étendue on peut reconstruire une preuve sous forme normale des GMCs, on obtient l'inclusion dans l'autre sens. On a donc $L(G) = L(MtoC(G))$, pour toute grammaire minimaliste G sans SMC. La question de la capacité générative des GMCs avec SMC reste ouverte.

L'approche que nous avons présentée ne tient pas compte des opérations de *Head-Movement* et d'*Affix-Hopping*. Cependant, on peut étendre ce résultat en observant que ces opérations dérivent les mêmes chaînes dans les deux types de structures.

6.3 Exemples

À présent, nous allons présenter trois exemples de dérivations réalisées à partir de GMCs

6.3.1 Questions

Analyse de la phrase :

(25) Quels livres lisent les enfants ?

L'analyse est similaire à celle de la phrase affirmative, à l'exception de la présence d'un trait de déplacement supplémentaire pour le *DP* objet. Au lieu de substituer le constituant dans sa position objet pour la phrase affirmative, on substitue une hypothèse $k \otimes d \vdash k \otimes d$, qui sera elle-même remplacée en fin de dérivation, lorsque le dernier trait de déplacement est introduit.

On dissocie la formule du verbe sur deux entrées. Ainsi un verbe transitif de type V nécessite un groupe nominal. Ce V est consommé pour introduire la suite des dépendances pour une phrase à l'actif. On utilise le lexique suivant :

Lexique 99.

articles	$\vdash (\epsilon, les, \epsilon) : \; k \otimes d \,/\, n$
	$\vdash (\epsilon, Quels, \epsilon) : \; (wh \otimes (k \otimes d)) \,/\, n$
nom	$\vdash (\epsilon, enfants, \epsilon) : \; n$
	$\vdash (\epsilon, livres, \epsilon) : \; n$
lire	$\vdash (\epsilon, li, \epsilon) : \; V \,/\, d$
	$\vdash (\epsilon, \epsilon, \epsilon) : \; (k \setminus (d \setminus v)) \,/_< V$
inflexion	$\vdash (\epsilon, -sent, \epsilon) : \; (k \setminus t) \,/_< v$
comp	$\vdash (\epsilon, \epsilon, \epsilon) : \; (wh \setminus c) \,/_< t$

La première phase de la dérivation est la saturation des différentes positions verbales. On commence par introduire une première hypothèse d. La formule du verbe est ensuite modifiée par une entrée à phonologie vide par un déplacement de tête. La forme phonologique du verbe reste donc sur la tête de la dérivation :

$$\dfrac{\dfrac{\vdash (\epsilon, \epsilon, \epsilon) : (k \setminus (d \setminus v)) \,/_< V \qquad \dfrac{\vdash (\epsilon, li, \epsilon) : V \,/\, d \quad u : d \vdash (\epsilon, u, \epsilon) : d}{u : d \vdash (\epsilon, li, u) : V}\,[mg]}{u : d \vdash (\epsilon, li, u) : k \setminus (d \setminus v)}\,[mg]$$

Cette dernière induit l'introduction d'une hypothèse de type k. La présence simultanée de ces deux hypothèses induit leur substitution par une nouvelle hypothèse composée :

$$\dfrac{\dfrac{x : k \otimes d \vdash (\epsilon, x, \epsilon) : k \otimes d \quad v : k, u : d \vdash (v, li, u) : d \setminus v}{x : k \otimes d \vdash (x, li, \epsilon) : d \setminus v}\,[mv]}{\dfrac{v : k \vdash (\epsilon, v, \epsilon) : k \quad u : d \vdash (\epsilon, li, u) : k \setminus (d \setminus v)}{x : k \otimes d \vdash (\epsilon, x, \epsilon) : k \otimes d \quad v : k, u : d \vdash (v, li, u) : d \setminus v}\,[mg]}\,[mg]$$

Enfin, on introduit la seconde hypothèse correspondant au *DP* sujet :

$$\frac{w : d \vdash (\epsilon, w, \epsilon) : d \quad x : k \otimes d \vdash (x, li, \epsilon) : d \setminus v}{w : d, x : k \otimes d \vdash (w\ x, li, \epsilon) : v}\ [mg]$$

Puis le verbe reçoit son inflexion par un nouveau déplacement de tête. Un verbe ayant reçu son inflexion peut alors attribuer le cas nominatif. La preuve se poursuit par un *merge* avec une seconde hypothèse k :

$$\frac{\vdash (\epsilon, -sent, \epsilon) : (k \setminus t) /_< v \quad w : d, x : k \otimes d \vdash (w\ x, li, \epsilon) : v}{w : d, x : k \otimes d \vdash (\epsilon, lisent, w\ x) : k \setminus t}\ [mg]$$

$$\frac{y : k \vdash (\epsilon, y, \epsilon) : k \quad w : d, x : k \otimes d \vdash (\epsilon, lisent, w\ x) : k \setminus t}{y : k, w : d, x : k \otimes d \vdash (y, lisent, w\ x) : t}\ [mg]$$

Parallèlement, on construit le *DP* sujet *les enfants* :

$$\frac{\vdash (\epsilon, les, \epsilon) : (k \otimes d) / n \quad \vdash (\epsilon, enfants, \epsilon) : n}{\vdash (\epsilon, les, enfants) : k \otimes d}\ [mg]$$

que l'on substitue dans la dérivation principale :

$$\frac{\vdash (\epsilon, les, enfants) : k \otimes d \quad y : k, w : d, x : k \otimes d \vdash (y, lisent, w\ x) : t}{x : k \otimes d \vdash (les\ enfants, lisent, x) : t}\ [mv]$$

À la fin de la preuve, l'entrée *comp* porte la formule permettant d'introduire le *DP* objet. Cela est réalisé par l'introduction d'une hypothèse wh :

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : (wh \setminus c) /_< t \quad x : k \otimes d \vdash (les\ enfants, lisent, x) : t}{x : k \otimes d \vdash wh \setminus (\epsilon, lisent, les\ enfants\ x) : c}\ [mg]$$

$$\frac{z : wh \vdash (\epsilon, z, \epsilon) : wh \quad x : k \otimes d \vdash wh \setminus (\epsilon, lisent, les\ enfants\ x) : c}{z : wh, x : k \otimes d \vdash (z, lisent, les\ enfants\ x) : c}\ [mg]$$

Parallèlement, on construit le *DP* objet *Quels livres* :

$$\frac{\vdash (\epsilon, Quels, \epsilon) : (wh \otimes (k \otimes d)) / n \quad \vdash (\epsilon, livres, \epsilon) : n}{\vdash (\epsilon, Quels, livres) : wh \otimes (k \otimes d)}\ [mg]$$

que l'on substitue dans la dérivation principale :

$$\frac{\vdash (\epsilon, Quels, livres) : wh \otimes (k \otimes d) \quad z : wh, x : k \otimes d \vdash (z, lisent, les\ enfants\ x) : c}{\vdash (Quels\ livres, lisent, les\ enfants\ x) : c}\ [mv]$$

La preuve reconnaît *Quels livres lisent les enfants ?*

6.3.2 Forme passive

Nous allons détailler l'analyse d'un exemple de phrase au passif.

(26) la présidente est élue par le peuple

Dans ce type de phrase, le verbe garde la même entrée lexicale. Ainsi, à un verbe transitif est associé la formule V / d. Le modifieur du verbe n'est pas le même que celui qui est utilisé pour les phrases affirmatives ou les interrogatives (voir l'exemple précédent), mais une entrée particulière donnant la forme passive à ce verbe. Grâce à cette entrée, l'hypothèse introduite en position objet recevra le cas nominatif. La forme de cette entrée induit aussi l'introduction d'un *DP* en position agentive dans cet énoncé.

Pour ce type de construction de *DP* complexe, on utilise généralement l'opération de *late adjonction*. Nous souhaitons proposer ces traitements dans les GMCs sans ajouter cette opération. Pour conserver l'ordre normal à l'intérieur de ces constituants, nous utilisons donc des entrées à phonologie vide assurant leur bonne formation.

Pour constuire le *DP* agentif, nous introduisons dans le lexique une entrée à phonologie vide qui gère le traitement d'un *DP* standard puis lui ajoute une préposition. Cette entrée impliquera que deux hypothèses soient utilisées dans la preuve, l'une de type d l'autre correspondant à k. On suppose que dans ces constructions, la préposition définit le cas du *DP*. Cette partie de la dérivation se termine par l'ajout de la préposition, donnant dans l'exemple un *DP* agentif (ayant donc déjà reçu son cas).

La preuve reconnaissant cet énoncé est réalisée à partir du lexique suivant :

Lexique 100.

article	$\vdash (\epsilon, le, \epsilon) :$	$(k \otimes d) \,/\, n$
	$\vdash (\epsilon, la, \epsilon) :$	$(k \otimes d) \,/\, n$
nom commun	$\vdash (\epsilon, peuple, \epsilon) :$	n
	$\vdash (\epsilon, présidente, \epsilon) :$	n
verbe	$\vdash (\epsilon, él, \epsilon) :$	$V \,/\, d$
modification du verbe	$\vdash (\epsilon, -u, \epsilon) :$	$(d_{agent} \setminus (ablatif \setminus v_{pass})) /_< V$
auxiliaire	$\vdash (\epsilon, est, \epsilon) :$	$(k \setminus t) \,/\, v_{pass}$
comp	$\vdash (\epsilon, \epsilon, \epsilon) :$	$c \,/\, t$
	$(\epsilon, \epsilon, \epsilon) :$	$\vdash (k \setminus (par \setminus (d_{agent} \otimes ablatif))) \,/\, d$
	$(\epsilon, par, \epsilon) :$	$\vdash par$

Lors de la première partie de la preuve, le verbe est saturé par une hypothèse de type d. Puis il reçoit son comportement syntaxique (ici sa forme passive).

$$\dfrac{\vdash (\epsilon, -ue, \epsilon) : (d_{agent} \setminus (ablatif \setminus v_{pass})) /_< V \qquad \dfrac{\vdash (\epsilon, él, \epsilon) : V \,/\, d \quad u : d \vdash (\epsilon, u, \epsilon) : d}{u : d \vdash (\epsilon, él, u) : V}\,[mg]}{u : d \vdash (\epsilon, élue, u) : d_{agent} \setminus (ablatif \setminus v_{pass})}\,[mg]$$

Puis, la dérivation sature deux positions par des variables, l'une de type d_{agent} et l'autre de type $ablatif$:

$$\frac{v : d_{agent} \vdash (\epsilon, v, \epsilon) : d_{agent} \quad u : d \vdash (\epsilon, élue, u) : d_{agent} \setminus (ablatif \setminus v_{pass})}{v : d_{agent}, u : d \vdash (v, élue, u) : ablatif \setminus v_{pass}} [mg]$$

$$\frac{w : ablatif \vdash (\epsilon, w, \epsilon) : ablatif \quad v : d_{agent}, u : d \vdash (v, élue, u) : ablatif \setminus v_{pass}}{w : ablatif, v : d_{agent}, u : d \vdash (w \bullet v, élue, u) : v_{pass}} [mg]$$

Parallèlement, le complément d'agent *par le peuple* est dérivé. D'abord par construction d'un DP :

$$\frac{\vdash (\epsilon, le, \epsilon) : (k \otimes d) \,/\, n \quad \vdash (\epsilon, peuple, \epsilon) : n}{\vdash (\epsilon, le, peuple) : k \otimes d} [mg]$$

Puis la dérivation sature la structure d'un complément d'agent :

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : (k \setminus (par \setminus (d_{agent} \otimes ablatif))) \,/\, d \quad x : d \vdash (\epsilon, x, \epsilon) : d}{x : d \vdash (\epsilon, \epsilon, x) : k \setminus (par \setminus (d_{agent} \otimes ablatif))} [mg]$$

$$\frac{y : k \vdash (\epsilon, y, \epsilon) : k \quad x : d \vdash (\epsilon, \epsilon, x) : k \setminus (par \setminus (d_{agent} \otimes ablatif))}{y : k, x : d \vdash (y, \epsilon, x) : par \setminus (d_{agent} \otimes ablatif)} [mg]$$

Ceci permet de réaliser la substitution du DP dans la structure :

$$\frac{\vdash (\epsilon, le, peuple) : k \otimes d \quad y : k, x : d \vdash (y, \epsilon, x) : par \setminus (d_{agent} \otimes ablatif)}{\vdash (le\ peuple, \epsilon, \epsilon) : par \setminus (d_{agent} \otimes ablatif)} [mv]$$

Le complément d'agent est finalement construit par l'ajout de la préposition :

$$\frac{\vdash (\epsilon, par, \epsilon) : par \quad \vdash (le\ peuple, \epsilon, \epsilon) : par \setminus (d_{agent} \otimes ablatif)}{\vdash (par\ le\ peuple, \epsilon, \epsilon) : d_{agent} \otimes ablatif} [mg]$$

La dérivation se poursuit en combinant ce DP agent avec la première partie dérivée.

$$\frac{\vdash (par\ le\ peuple, \epsilon, \epsilon) : d_{agent} \otimes ablatif \quad \begin{array}{l} w : ablatif, v : d_{agent}, u : d \\ \vdash (w \bullet v, élue, u) : v_{pass} \end{array}}{u : d \vdash (par\ le\ peuple, élue, u) : v_{pass}} [mv]$$

Cette structure verbale reçoit une inflexion ce qui implique une hypothèse de type nominatif (k).

$$\dfrac{y : k \vdash (\epsilon, y, \epsilon) : k \qquad \dfrac{u : d \vdash (par\ le\ peuple, élue, u) : v_{pass} \quad \vdash (\epsilon, est, \epsilon) : (k \setminus t) \,/\, v_{pass}}{u : d \vdash (\epsilon, est\ élue, par\ le\ peuple \bullet u) : k \setminus t}\,[mg]}{y : k, u : d \vdash (y, est\ élue, par\ le\ peuple \bullet u) : t}\,[mg]$$

L'étape suivante construit le *DP* sujet.

$$\dfrac{\vdash (\epsilon, la, \epsilon) : (k \otimes d) \,/\, n \quad \vdash (\epsilon, présidente, \epsilon) : n}{\vdash (\epsilon, la, présidente) : k \otimes d}\,[mg]$$

Ce dernier est substitué grâce à la présence simultanée des deux hypothèses k et d. La dérivation se termine par une vérification de la non-inclusion dans une autre dérivation via l'entrée *comp*.

$$\dfrac{\dfrac{\vdash (\epsilon, la, présidente) : k \otimes d \quad y : k, u : d \vdash (y, est\ élu, par\ le\ peuple \bullet u) : t}{\vdash (la\ présidente, est\ élue, par\ le\ peuple) : t}\,[mv] \quad \vdash (\epsilon, \epsilon, \epsilon) : c \,/\, t}{\vdash (\epsilon, \epsilon, la\ présidente\ est\ élue\ par\ le\ peuple) : c}\,[mg]$$

La preuve reconnaît : *la présidente est élue par le peuple.*

6.3.3 Remnant Movement

On remarquera que les phénomènes de déplacement dépendant d'autres déplacements du type *Remnant Movement* sont également reconnus par les GMCs :

Analyse de la phrase :

(27) Love Mary Peter will

Ici, le déplacement du sujet est réalisé avant le déplacement du verbe dans la GM. Dans la GMC, les hypothèses ne seront dans la même formule qu'après que les hypothèses du verbe soient consommées.

Bien que cette propriété ne semble pas évidente puisque contrairement aux GMs, les constituants des GMCs ne sont pas effectivement présents dans la dérivation avant que leur traitement soit entièrement réalisé, les positions occupées par le déplacement sont marquées par l'introduction d'hypothèses dans la preuve. Les substitutions ne sont que la reformulation de la preuve lorsque toutes les informations nécessaires ont été dérivées.

Pour cela, nous allons montrer ce cas sur un exemple, à partir du lexique suivant :

Lexique 101.

nom propre	$\vdash (\epsilon, Mary, \epsilon) :$	$k \otimes d$
	$\vdash (\epsilon, Peter, \epsilon) :$	$k \otimes d$
verbe	$\vdash (\epsilon, love, \epsilon) :$	$V \,/\, d$
	$\vdash (\epsilon, \epsilon, \epsilon) :$	$k \setminus d \setminus v \,/_< V$
	$\vdash (\epsilon, will, \epsilon) :$	$perf \,/\, v$
	$\vdash (\epsilon, \epsilon, \epsilon) :$	$k \setminus c \,/\, perf$
	$\vdash (\epsilon, \epsilon, \epsilon) :$	$top \setminus c \,/\, c$
	$\vdash (\epsilon, \epsilon, \epsilon) :$	$(top \otimes v) /_< v$

La dérivation se construit en deux phases principales l'une régissant le verbe, l'autre construisant la structure de la phrase.

Dans la première partie de la preuve, le verbe est saturé par une première hypothèse de type d. Puis, il reçoit son comportement syntaxique dans une phrase active. Cette entrée permet l'introduction du cas du DP objet :

$$\dfrac{v : k \vdash (\epsilon, v, \epsilon) : k \qquad \dfrac{\dfrac{\vdash (\epsilon, love, \epsilon) : V \,/\, d \quad u : d \vdash (\epsilon, u, \epsilon) : d}{\vdash (\epsilon, \epsilon, \epsilon) : (k \setminus (d \setminus v)) \,/_< V \qquad u : d \vdash (\epsilon, love, u) : V}\,[mg]}{v : k, u : d \vdash (v, love, u) : d \setminus v}\,[mg]}{}$$

Les deux hypothèses du DP objet étant présentes, le DP objet se substitue à elles. Puis on introduit l'hypothèse du DP sujet. Ce verbe devant être déplacé dans la phrase, son comportement est à nouveau modifié. Pour conserver la forme phonologique du verbe sur la tête de la dérivation, cette opération est réalisée par un déplacement de tête.

$$\dfrac{\dfrac{\vdash (\epsilon, Mary, \epsilon) : k \otimes d \quad v : k, u : d \vdash d \setminus (v, love, u) : v}{\vdash (Mary, love, \epsilon) : d \setminus v}\,[mv]}{\dfrac{\vdash (\epsilon, \epsilon, \epsilon) : (top \otimes v) \,/_< v \qquad \dfrac{w : d \vdash (\epsilon, w, \epsilon) : d \quad \vdash (Mary, love, \epsilon) : d \setminus v}{w : d \vdash (w\ Mary, love, \epsilon) : v}\,[mg]}{w : d \vdash (\epsilon, love, w\ Mary) : top \otimes v}\,[mg]}$$

On obtient alors la première phase de la dérivation.

Parallèlement, on utilise une hypothèse représentant le verbe dans la fusion avec son auxiliaire. La preuve se poursuit par l'introduction de l'hypothèse de type k. C'est à ce point de la dérivation que dans la GM le déplacement est effectif. Ici, les hypothèses ne sont pas présentes dans la dérivation, mais la position est marquée. Lors de la substitution c'est à cette place que la forme phonologique associée au verbe sera placée.

$$\dfrac{\dfrac{\vdash (\epsilon, \epsilon, \epsilon) : (k \setminus c) \,/\, perf \qquad \dfrac{\vdash (\epsilon, will, \epsilon) : perf \,/\, v \quad x : v \vdash (\epsilon, x, \epsilon) : v}{x : v \vdash (\epsilon, will, x) : perf}\,[mg]}{x : v \vdash (\epsilon, \epsilon, will\ x) : k \setminus c}\,[mg]}{}$$

$$\dfrac{y : k \vdash (\epsilon, y, \epsilon) : k \quad x : v \vdash (\epsilon, \epsilon, will\ x) : k \setminus c}{y : k, x : v \vdash (y, \epsilon, will\ x) : c}\,[mv]$$

Enfin, cette phase de la preuve se termine par l'utilisation de l'entrée *comp*.

Celle-ci introduit une hypothèse *top*.

$$\dfrac{z : top \vdash (\epsilon, z, \epsilon) : top \quad \dfrac{\vdash (\epsilon, \epsilon, \epsilon) : (top \setminus c) \,/\, c \quad y : k, x : v \vdash (y, \epsilon, will\ x) : c}{y : k, x : v \vdash (\epsilon, \epsilon, y\ will\ x) : top \setminus c}\,[mg]}{z : top, y : k, x : v \vdash (z, \epsilon, y\ will\ x) : c}\,[mg]$$

Dans cette partie de la preuve, les hypothèses composant la première phase sont présentes, on réalise alors la substitution à ces hypothèses. Cette substitution permet alors de réaliser celle qui était en attente (pour le *DP* sujet) dans la position de y :

$$\dfrac{w : d \vdash (\epsilon, love, w\ Mary) : top \otimes v \quad z : top, y : k, x : v \vdash (z, \epsilon, y\ will\ x) : c}{w : d, x : v \vdash (love\ w\ Mary, \epsilon, y\, will) : c}\,[mv]$$

$$\dfrac{\vdash (\epsilon, Peter, \epsilon) : k \otimes d \quad w : d, x : v \vdash (love\ w\ Mary, \epsilon, y\, will) : c}{\vdash (love\ Mary, \epsilon, Peter\ will) : c}\,[mv]$$

la preuve reconnaît alors *love Mary Peter will*, ce qui montre que les *remnant movements* peuvent être reconnus par les GMCs.

Chapitre 7

Interface syntaxe et sémantique

Sommaire

7.1 Prérequis pour la sémantique computationnelle 175
7.2 Interface syntaxe/sémantique 181
7.3 Exemples 187
7.4 Conclusion 197

La sémantique computationnelle traite d'objets linguistiques, que ceux-ci soient des phrases, des morceaux de texte ou des dialogues. Elle hérite de concepts et de techniques développés par la sémantique formelle - discipline qui utilise des méthodes logiques pour la description du sens des énoncés d'une langue naturelle. Les analyseurs syntaxiques sont utilisés pour calculer effectivement et efficacement les représentations syntaxiques, les formes logiques et les raisonnements que l'on peut extraire automatiquement à partir d'énoncés.

Une sémantique computationnelle pour les langues naturelles pose les questions suivantes :

1. Quel est le sens d'un énoncé ?
2. Quel est le processus mis en œuvre dans la construction sémantique ?
3. Comment automatiser ce processus ?

Une première définition du sens d'un énoncé a été introduite par Frege. Il propose de considérer pour un énoncé les conditions pour lesquelles il est valide. De plus, Frege avance l'hypothèse que le sens d'un énoncé est construit à partir du sens de ses composantes. C'est une conséquence directe de l'observation que toute langue naturelle permet l'élaboration d'un nombre infini d'énoncés à partir d'un nombre fini de mots. Cette hypothèse est appelée *principe de compositionnalité* formulé par Partee dans [PtMW90] :

> *"The meaning of a compound expression is a function of the meaning of its parts"*
> *Le sens d'une expression composée est une fonction du sens de ses parties.*

Frege donne ainsi une première définition de la sémantique formelle à partir de la logique. L'évaluation d'une expression logique dérivée de ses composantes fournit une valeur de vérité dénotant un énoncé.

Le principe de Partee est complété par Montague ainsi :

"The meaning of a compound expression is a function of the meaning of its parts and the syntaxtic rules by which they are combined."
Le sens d'une expression composée est une fonction du sens de ses parties et des règles syntaxiques par lesquelles elles sont combinées.

Cette contrainte de correspondance règle à règle n'est cependant pas nécessaire. Certaines règles peuvent n'avoir qu'une portée syntaxique et inversement, une partie du calcul sémantique peut ne pas trouver sa correspondance directe dans la règle qui l'a déclenchée. Cette vision du calcul basée principalement sur l'apport de chaque entité est très utilisée dans la communauté, tant du point de vue linguistique, [Cho95], que du point de vue computationnel pour tous les formalismes lexicalisés : Combinatory Categorial Grammars - CCG [Ste00] - Tree Adjoining Grammars - TAG [JS97] - ou encore les GMs.

Dans le programme minimaliste, Chomsky suppose que la dérivation dite syntaxique peut à un certain point être dérivée d'une part vers une forme phonologique - PF - que nous avons introduite par les étiquettes des GMCs, et d'autre part vers une forme logique - LF. En utilisant le calcul syntaxique, nous pouvons dériver une structure logique de l'énoncé grâce à la correspondance règle à règle, puis calculer la forme sémantique.

Le but est donc d'utiliser les règles syntaxiques entre constituants pour les combiner. Nous utilisons pour cela la logique des prédicats. Ces formules sont alors interprétées dans un modèle. Le processus mis en œuvre pour la formation des formules est classiquement basé sur les travaux de Montague. Il utilise un typage des constituants et le λ-calcul typé. Cependant, le λ-calcul ne permet pas de rendre compte de tous les phénomènes identifiés. Nous proposons dans ce chapitre un calcul sémantique qui en utilise une extension : le $\lambda\mu$-calcul. Nous commencerons par introduire l'ensemble des concepts avant de définir l'interface syntaxe-sémantique des GMCs.

Avant cela, revenons sur d'anciennes propositions que nous avons formulées, puis abandonnées pour celle-ci. Tout d'abord nous avons tenté de proposer une interface syntaxe/sémantique utilisant le λ-calcul directement à partir des arbres obtenus par les GMs. Comme nous le verrons, l'interprétation sémantique de la fusion n'est pas problématique puisqu'elle correspond naturellement à l'application fonctionnelle. C'est celle du déplacement qui pose plus de questions, bien qu'elle se rapproche grandement de la notion d'abstraction. Deux propositions peuvent être formulées : soit les applications fonctionnelles sont réalisées lors de l'entrée dans la dérivation d'un constituant, soit lorsque celui-ci ne possède plus de trait.

Dans le premier cas, l'opération de déplacement n'est simplement plus utilisée sémantiquement. Bien que nous n'en rendons pas compte actuellement dans notre système, ce cas pose aussi le problème de l'opération de *late adjunction*. Cette dernière permet a posteriori d'ajouter des éléments à la dérivation. Dans ce cas, ils ne peuvent plus intervenir sur la sémantique du constituant dans lequel il s'insère. De plus, il serait surprenant que le déplacement n'ait aucun effet sur la sémantique. Cette proposition ne nous semble pas théoriquement justifiable.

Le second cas suppose alors que les dérivations soient fortement normées. Ainsi, dans le cas des questions où l'objet possède un trait de plus que le sujet, la variable sujet sera la première à être appliquée et prendra la position de la variable objet. Pour pallier à

ce manque, nous avons alors proposé d'utiliser une modélisation du λ-calcul sous-spécifié, [Amb05c], proposée par M. Egg *et al.*, [EKNar]. La sous-spécification permet simplement de résoudre les ambiguïtés de portés de quantificateurs, mais pose deux problèmes. Le premier est intrinsèque à son rapport avec la théorie générative. En effet, pour cette dernière, la modification de portée de quantification devrait vraisemblablement être motivée par une transformation et non être déléguée à un autre type de représentation. Mais plus généralement, si la sous-spécification permet de rendre compte d'ambiguïté de portée de quantification, elle permet également de calculer toutes les portés possibles, ce qui linguistiquement n'est pas acceptable. Par exemple pour la phrase :

(28) À ce qu'il parait, Marie conduit habituellement une cadillac

possède trois quantificateurs : *À ce qu'il parait*, *habituellement* et *une*. Mais seule les interprétations dans lesquelles *A ce qu'il parait* précède *habituellement* sont acceptable. Les systèmes de sous-spécification tel que CCT ne sont actuellement pas capables de prendre en compte ces propriétés. Nous n'avons donc pas poursuivi dans cette voie, mais fait appel à un système dans lequel les représentations intermédiaires sont proches de la sous-spécification et pour lequel il serait possible d'intégrer des réductions à certains moments permettant de limiter la sur-génération de la sous-spécification.

7.1 Prérequis pour la sémantique computationnelle

7.1.1 Logique des prédicats et théorie des modèles

Définition 102. *Soit C un alphabet de* ***constantes*** *individuelles. Soit V un alphabet de* ***variables*** *individuelles.*

La logique des prédicats utilise des ***formules*** *manipulant des* ***termes****.*

Les termes sont définis inductivement :

- *$a \in C$, a est un terme,*
- *$x \in V$, x est un terme,*
- *si P^n est un prédicat d'arité n et pour $i \in [n]$, $\tau_i \in C$ ou $\tau_i \in V$, $P^n(\tau_1, \cdots, \tau_n)$ est un terme.*

Une formule de la logique des prédicats est définie inductivement par :

- *$\neg\alpha$ est une formule bien formée si et seulement si α est une formule bien formée,*
- *$\alpha \wedge \beta$, $\alpha \vee \beta$ $\alpha \rightarrow \beta$ sont des formules bien formées si et seulement si α et β sont des formules bien formées,*
- *si α est une formule bien formée et x est une variable, alors $\exists x\ \alpha$ et $\forall x\ \alpha$ sont des formules bien formées. x est alors dite* ***variable liée****.*

Les variables non liées sont dites ***variables libres****.*

La notion de *vériconditionnalité* associée à la logique des prédicats est définie par Tarski [Tar39] dans le cadre de la *théorie des modèles* pour laquelle l'interprétation d'une formule est réalisée dans un modèle donné.

Un modèle M est un couple (D, I) où D est un domaine d'interprétation des entités et I une fonction d'interprétation qui assigne à chaque constante un élément du domaine. À chaque énoncé on associe une formule α sans variable libre. α est dite *satisfaisable* dans un modèle M s'il existe une fonction g d'assignation de valeurs dans M aux variables de α rendant la formule vraie. Cette fonction associe une dénotation à la formule α.

Une formule est alors la description d'un énoncé et un modèle est une représentation du monde dans laquelle l'assignation cherche les contextes validant la formule. Une formule est valide s'il n'y a pas de contexte dans le modèle l'invalidant.

Comme nous venons de le mentionner, g permet d'associer des valeurs aux variables. Dans la définition du modèle, I permet d'associer des valeurs aux constantes. On note :

$$||\tau|| = \begin{cases} I(\tau) & \text{si } \tau \in C \\ g(\tau) & \text{si } \tau \in V \end{cases}$$

Une formule est satisfaisable, dans un modèle M et pour une fonction d'assignation g, notée $M, g \models \alpha$, on a :

$M, g \models P^n(\tau_1, \cdots, \tau_n)$	*si et seulement si*	$(\|\|\tau_1\|\|, \cdots, \|\|\tau_n\|\|) \in I(P^n)$
$M, g \models \neg\alpha$	*si et seulement si*	$M, g \not\models \alpha$
$M, g \models \alpha \wedge \beta$	*si et seulement si*	$M, g \models \alpha$ et $M, g \models \beta$
$M, g \models \alpha \vee \beta$	*si et seulement si*	$M, g \models \alpha$ ou $M, g \models \beta$
$M, g \models \alpha \rightarrow \beta$	*si et seulement si*	$M, g \not\models \alpha$ ou $M, g \models \beta$
$M, g \models \forall x.\alpha$	*si et seulement si*	*pour tout g' de M tel que* $g(x) = g'(x)$ $M, g' \models \alpha$
$M, g \models \exists x.\alpha$	*si et seulement si*	*il existe g' de M tel que* $g(x) = g'(x)$ $M, g' \models \alpha$

Une formule est vraie dans un modèle si et seulement si, pour toute assignation g de valeurs aux variables libres dans M : $M, g \models \alpha$. On note $M \models \alpha$ si α est vraie dans M.

7.1.2 Types de Montague

Le problème est de fournir des formules à interpréter dans un modèle. Richard Montague est le premier à avoir proposé un calcul sémantique compositionnel pour les langues naturelles. Il cherchait à obtenir une analyse d'une langue naturelle qui aurait le même degré de rigueur que les théories sémantiques pour les langages formels, au point d'être considérée comme une théorie mathématique. La théorie sémantique de Montague se base sur les relations entre prédicats et arguments dans une phrase par l'utilisation de variables liées par des quantificateurs.

La sémantique pour un fragment de langue naturelle porte trois composantes :
- un ensemble de référents,
- un lexique,
- un ensemble de règles de composition.

Les *dénotations* peuvent être de deux natures, soit des références à des entités du monde réel, soit des représentations abstraites de ces entités dans un système de représentations.

Définition 103. *Une **dénotation** est composée de :*
- *D l'ensemble de toutes les entités - l'ensemble de tous les individus du système,*
- *les valeurs de vérité : des booléens,*
- *F l'ensemble des fonctions définies inductivement par : F^0 l'ensemble des fonctions de $D \rightarrow \{0, 1\}$ et on construit les ensemble de fonctions $F^{n+1} : D \rightarrow F^n$.*

À partir de ces définitions, Montague propose un système de types qui dirige le calcul sémantique.

Définition 104. *On définit l'ensemble des types sémantiques Sem comme le plus petit ensemble tel que :*
- *e et t sont des types, $e \in D$ et $t \in \{0, 1\}$*
- *Si τ_1 et τ_2 sont des types $\tau_1 \to \tau_2$ est un type.*

Si une expression α est de type $\tau_1 \to \tau_2$ et β de type τ_1, le résultat de l'application fonctionnelle de α sur β, $\alpha(\beta)$ est de type τ_2.

Quelques exemples de types standard pour des catégories de base dans la théorie montagovienne sont présentés dans la figure 41.

nom	$e \to t$
article	$(e \to t) \to ((e \to t) \to t)$
verbe transitif	$e \to (e \to t)$
groupe nominal	$(e \to t) \to t$
phrase	t

Fig. 41 – Exemples de types.

On remarquera que le type d'un groupe nominal est $(e \to t) \to t$, alors qu'il dénote un individu du modèle (normalement de type e). On appelle alors *type raising* d'un individu la propriété de le dénoter par l'ensemble de ses propriétés (de type $(e \to t) \to t$)).

Le système de types s'interprète alors dans la dénotation. En règle générale, pour la sémantique formelle du point de vue de la linguistique, l'enjeu est de trouver un système de dénotation permettant la bonne interprétation. Nous laisserons le problème de l'interprétation ouvert. Cette courte présentation a pour seul but de donner la perspective dans laquelle ces travaux se placent. Ici nous souhaitons trouver les formules logiques représentant les énoncés. Pour cela, il nous faut un système calculatoire en relation avec les types. Montague utilise alors le λ-calcul simplement typé. Ce calcul, introduit par Church [Chu40] propose au départ de modéliser les fonctions mathématiques.

7.1.3 λ-calcul

Le λ-calcul est basé sur la logique combinatoire et a été proposé par Alonso Church, [Chu40], pour représenter les fonctions mathématiques. Le λ-calcul est composé d'un ensemble dénombrable de symboles, de variables, des parenthèses, du point et du symbole λ.

Définition 105. *Les termes du λ-calcul sont définis inductivement par :*
- *Si x est une variable, x est un terme.*
- *Si x est une variable et t un terme, alors $\lambda x.t$ est un λ-terme, appelé λ-abstraction de la variable x dans t.*
- *Si t_1 et t_2 sont des termes, alors $t_1(t_2)$ est un λ-terme, appelé application.*

Par abus de notation nous appellerons terme un λ-terme

Les termes représentent des abstractions des fonctions. Dans $\lambda x.M$, M est le corps de la fonction contenant une variable x. Dans (MN), M est une fonction à laquelle on applique N comme argument. Nous allons voir que le résultat de l'application se calcule par réduction. Il n'y a pas de distinction entre terme et variable, il est donc possible de

passer en argument d'un terme un autre terme. En mathématiques, cela correspond à passer une fonction en argument d'une autre fonction.

Comme en logique du premier ordre, l'abstracteur λ est un lieur de variables. Ceci nous permet de définir les variables *libres* et *liées*.

Définition 106. *La portée d'un abstracteur λx dans un terme $\lambda x.t$ est le terme t.*

Une occurrence d'une variable x dans le terme t est dite libre *si elle n'est pas dans la portée d'un abstracteur apparaissant dans t.*

Si $\lambda x.\phi$ est un sous-terme de ψ et que x est dans ϕ, cette occurrence de x est dite liée par l'abstracteur λx.

On utilise la curryfication des termes : $\lambda(x,y).f(x,y)$ est noté $\lambda x \lambda y.fxy$.

Définition 107. *Soit $\equiv$ la relation d'équivalence entre deux termes représentant la congruence pour la λ-abstraction et l'application :*

- *$t \equiv N \Rightarrow \lambda x.t \equiv \lambda x.N$, pour toute variable x.*
- *$t \equiv N \Rightarrow$ pour tout terme $P : t(P) \equiv N(P)$ et $P(t) \equiv P(N)$.*

Soit $t[x]$ un terme contenant la variable x. On appelle substitution de x par y le terme t dans lequel les occurrences libres de x sont remplacées par y. On note $t[y/x]$ cette substitution. La substitution peut entraîner des problèmes de capture de variable. Pour cela on introduit l'α-conversion.

Définition 108. *L'*α-conversion *est le renommage des variables. Elle permet d'identifier les fonctions qui ne diffèrent que par le nom de leurs variables :*

$$\lambda x.t \equiv \lambda z.t[x/z]$$

La substitution est donc uniquement le symbole représentant l'α-conversion.

On peut définir inductivement la substitution :

1. *substitution d'une variable libre : $x[z/x] \rightarrow z$*
2. *substitution d'une variable liée : $(\lambda x.t)[z/x] \rightarrow \lambda z.t[z/x]$*
3. *substitution d'une variable libre non présente : $y[z/x] \rightarrow y$ si $y \neq x$*
4. *substitution d'une variable liée non présente : $(\lambda yt)[z/x] \rightarrow \lambda y.t[z/x]$*
5. *substitution lors d'une application : $\lambda x.(t(N))[z/x] \rightarrow \lambda z.(t[z/x](N))$*

Dans un terme toutes les occurrences d'une variable sont soit *libres*, soit *liées*. On montre que si ça n'est pas le cas, il existe un terme équivalent par α-conversion qui a cette propriété.

Définition 109. *(*β-réduction*) L'application fonctionnelle correspond à la substitution d'une variable par un terme :*

$$\lambda x.t(N) \equiv t[N/x]$$

On dit qu'un terme est sous ***forme normale*** *s'il n'est plus β-réductible.*

Nous avons jusqu'à présent introduit le λ-calcul non-typé. Montague utilise le λ-calcul simplement typé. Pour cela nous introduisons un typage des variables et des termes. Ainsi, pour chaque terme son type dépend de celui de ses variables abstraites. L'application fonctionnelle ne peut être réalisée qu'entre des termes de type similaire. Cette modification permet de lever les ambiguïtés du calcul et de prouver la propriété de normalisation forte.

Propriété 110. *Pour tout terme t du λ-calcul typé, toutes les réductions aboutissent à une unique forme normale.*

La propriété de normalisation forte assure la terminaison de tous les calculs. Cependant, elle restreint fortement le pouvoir expressif du système.

Soit un système de types défini comme dans la partie précédente sur un ensemble de types et de l'opérateur $\rightarrow$. Pour chaque variable du λ-calcul, on associe un type. L'application est la composition des types. Ainsi $\lambda x.t$ où x est de type f et t de type ι, est de type $f \rightarrow \iota$.

Comme nous l'avons évoqué, nous utilisons une extension de ce calcul, le $\lambda\mu$-calcul.

7.1.4 $\lambda\mu$-calcul

Le $\lambda\mu$-calcul se base sur la théorie des continuations. Il a été introduit par Parigot [Par92] comme une extension de l'isomorphisme de Curry-Howard à la logique classique. Il permet de rendre compte de la théorie des *continuations*. Ainsi, il devient possible dans un tel calcul de procéder par *appel par nom* ou par *appel par valeurs*.

Une **continuation** est la suite de calculs qui suit une fonction. Cette dernière est alors vue en contexte. Par exemple, pour le calcul de $\phi(n)$ puis $f(\phi(n))$, à partir de $\phi(n)$, la connaissance du contexte est nulle. Pour modéliser la continuation de la fonction ϕ, notée $\overline{\phi}$, on passe en argument de la fonction la suite du calcul dans lequel elle se situe : $\overline{\phi}(n, f) = f(\phi(n))$.

Le **$\lambda\mu$-calcul** est une extension du λ-calcul. On introduit un second alphabet de variables (α, β, γ, ...) que l'on appelle des μ-variables. L'utilisation de ces variables est rendue possible par l'ajout de deux constructeurs : la μ-abstraction ($\mu\alpha.t$) et le nommage (αt). Le principe est de *nommer* les sous-termes par des étiquettes d'instruction. Le nommage permet alors leur utilisation a posteriori (une abstraction peut être vue comme une activation des termes nommés).

Définition 111. *Soient V un ensemble de variables notées $x, y, z, \ldots$ et V_{cont} un ensemble de variables de continuation, notées $\alpha, \beta, \gamma, \ldots$, deux ensembles disjoints. Les $\lambda\mu$-termes sont définis inductivement par la syntaxe suivante :*

$$T ::= x | \lambda x.\mathrm{T} | \mu\alpha\mathrm{T} | (T)\mathrm{T}$$
$$\mathrm{T} ::= T | (\alpha\mathrm{T})$$

Cette version du $\lambda\mu$-calcul est une version plus souple que celle de Parigot, proposée par de Groote où chaque μ-variable est toujours appliquée à exactement un argument.

On note le type absurde $\bot$ qui est le type des entités observables. Les opérateurs des μ-variables respectent les règles de typage suivantes

$$\frac{\alpha : \neg A \quad t : A}{\alpha t : \bot} \qquad \frac{\begin{array}{c}[\alpha : \neg A] \\ \vdots \\ t : \bot\end{array}}{\mu\alpha.t : A}$$

λ et μ sont des constructeurs. Les λ- et μ-constructions sont appelées *abstractions* et $(u)v$ est l'application. Les termes sont considérés modulo α-conversion sur les λ- et μ-variables. On utilise les notations : x (*resp.* α) $\in u$ pour "x (*resp.* α) est une variable libre dans u" . De plus on note $(u)v_1 \dots v_n$ pour $(\dots((u)v_1)v_2\dots)v_n$.

Pour le $\lambda\mu$-calcul simplement typé, les termes sont notés $\langle\Gamma;\Delta\rangle \vdash u : N$, où Γ (*resp.* Δ) contient les déclarations des λ-variables (*resp.* μ-variables). On note ϵ l'ensemble de déclaration de variables vide et $\Gamma[x:N]$ l'ensemble de déclaration Γ contenant la déclaration $x : N$. On note $\to$ le constructeur de type généralisant celui introduit pour les types de Montague. Chaque variable apparaît au plus une fois dans le contexte.

Les règles de dérivation sont :

$$\frac{}{\langle x:N;\epsilon\rangle \vdash x : N}\,[var] \qquad \frac{\langle\Gamma[x:N];\Delta\rangle \vdash u : M}{\langle\Gamma;\Delta\rangle \vdash \lambda x.u : N \to M}\,[\lambda]$$

$$\frac{\langle\Gamma;\Delta,\beta:M\rangle \vdash u : N}{\langle\Gamma;\Delta,\alpha:N\rangle \vdash \mu\beta u : M}\,[\mu] \qquad \frac{\langle\Gamma;\Delta\rangle \vdash u : N \to M \quad \langle\Gamma';\Delta'\rangle \vdash v : N}{\langle\Gamma,\Gamma';\Delta,\Delta'\rangle \vdash (u)v : M}\,[app]$$

La règle $[\lambda]$ (*resp.* $[\mu]$) contient un affaiblissement explicite si x (*resp.* β) n'apparait pas dans le contexte. La règle $[app]$ contient une contraction implicite si les variables apparaissent dans Γ et Γ' ou dans Δ et Δ'. Il y a aussi une contraction implicite dans la règle $[\mu]$ si α apparaît dans Δ.

On introduit également les trois règles de réduction :

1. β-réduction : $(\lambda x.u)v \to_\beta u[v/x]$
2. μ-réduction : $(\mu\alpha.u)v \to_\mu \mu\alpha.u[\alpha(w)v/\alpha w]$
3. μ'-réduction : $(\mu\alpha.u)v \to_{\mu'} \mu\alpha.u[\alpha(v)w/\alpha w]$

La logique classique est naturellement non confluente. La règle de μ'-réduction est symétrique à celle de μ-réduction. L'ajout de cette règle entraîne que le calcul ne satisfait pas la propriété de Church-Rosser [Par00].

Les μ-abstracteurs étant conservés par ces réductions, on introduit une règle de simplification :

$$(simpl)\mu\alpha.u \longrightarrow u[\alpha\ t := t]$$

qui n'est appliquée que pour des termes de type $\bot$, pour t un terme étiqueté par α.

Exemple 112. *Exemples de $\lambda\mu$-réductions pour chacune des règles :*

$$\begin{array}{llll}(\beta) & ((\lambda y\mu\alpha.\alpha y)x) & \to_\beta & \mu\alpha(\alpha x)\\ (\mu) & ((\mu\alpha(\alpha Q))P) & \to_\mu & \mu\alpha.\alpha((Q)P)\\ (\mu') & ((\mu\alpha(\alpha Q))P) & \to_{\mu'} & \mu\alpha.\alpha((P)Q)\\ (simpl) & ((\mu\alpha(\alpha Q)) & \to_{simpl} & Q\end{array}$$

Remarque 113. *L'utilisation des continuations a un impact direct sur les types des fonctions puisque si ϕ est de type $a \to b$, f de type $b \to c$, alors $\overline{\phi}$ est de type $a \to ((b \to c) \to c)$.*

De ce fait, pour les types à la Montague des DPs quantifiés : un livre *: $(e \to t) \to t$ est le continuisé du DP, initialement de type e.*

De Groote [dG01] donne la preuve du typage des DPs quantifiés, en utilisant la double négation :

$$\dfrac{\dfrac{\dfrac{\langle\epsilon;\epsilon\rangle \vdash livre : e \to t \quad \langle x : e;\epsilon\rangle \vdash x : e}{\langle x : e;\epsilon\rangle \vdash livre(x) : t} \qquad \dfrac{\langle x : e;\epsilon\rangle \vdash x : e}{\langle x : e;\alpha : e\rangle \vdash (\alpha x) : t}}{\langle x : e;\alpha : e\rangle \vdash livre(x) \wedge (\alpha x) : t}}{\langle\epsilon;\alpha : e\rangle \vdash \exists x livre(x) \wedge (\alpha x) : t}}{\langle\epsilon;\epsilon\rangle \vdash \mu\alpha.\exists x livre(x) \wedge (\alpha x) : e}$$

La différence avec la version montagovienne classique $\lambda Q.\exists x livre(x) \wedge (Qx)$, dont le type est $(e \to t) \to t$, réside dans le statut associé à la propriété associée à la formule. Dans cette version, le calcul doit spécifier Q immédiatement et prend donc en compte son type $(e \to t)$. Dans la version du $\lambda\mu$-calcul, cette propriété est "mise en attente" donc son type n'est pas directement pris en considération. Ainsi, on unifie les types associés aux DPs. Cette propriété nous conduit à modifier le morphisme de type pour lequel on n'utilise pas le Type-Raising pour les DPs quantifiés mais uniquement du type de base e.

Dans la suite de cette présentation, nous ne noterons pas les ensembles de définitions afin de simplifier les notations. Cependant, il est aisé de calculer ces ensembles au fur et à mesure des dérivations.

À présent que les concepts techniques définissant les termes sémantiques sont définis, nous nous intéressons à l'interface syntaxe-sémantique.

7.2 Interface syntaxe/sémantique

La première proposition d'interface des GMCs a été proposée dans [ALR03]. Comme Chomsky le propose dans le programme minimaliste, la dérivation syntaxique permet de porter le calcul d'une structure logique jusqu'au point de Spell-Out puis laisse le calcul sémantique se terminer après ce point. Le but est de synchroniser un calcul fournissant une (ou plusieurs) structures pouvant être dérivées en formule(s) logique(s) à partir des analyses syntaxiques réalisées par les GMCs.

Les liens entre les analyses syntaxiques et sémantiques peuvent être envisagés de deux manières différentes :

1. soit on associe à une analyse syntaxique une unique représentation sémantique,
2. soit à partir d'une analyse syntaxique on peut produire plusieurs représentations sémantiques relevant d'ambiguïtés.

Plusieurs travaux récents ont choisi la première solution, et c'était également le sens donné aux premiers travaux de Stabler sur l'interface syntaxe/sémantique pour les GMs. Nous préférons la seconde solution qui est plus souple en termes de représentations. Les exemples proposés au chapitre 1 montrent clairement que des ambiguïtés syntaxiques existent. Cependant, nous supposons que le problème de la portée des quantificateurs est une donnée sémantique, pouvant se manifester à d'autres niveaux d'analyse (syntaxe, morphologie, *etc.*). Ainsi, supposer que d'une analyse syntaxique nous pouvons dériver plusieurs représentations sémantiques laisse à chaque niveau son degré de granularité et cela n'exclut pas que des ambiguïtés syntaxiques puissent être détectées.

La synchronisation entre calcul syntaxique et calcul sémantique nécessite qu'à chaque entrée lexicale soit associée une structure sémantique et que chaque règle du calcul syntaxique possède une règle de composition de ces structures.

Les GMCs étant basées sur un système de type logique, l'isomorphisme de Curry-Howard permet de maintenir la cohérence et la synchronisation des deux calculs. On souhaite donc donner une correspondance sémantique à chaque partie du calcul syntaxique.

Le calcul que nous mettons en œuvre est basé sur les *Uniform Theta Assignement Hypothesis* (UTAH) pour lesquelles les relations thématiques sont intégrées dans la structure profonde. L'utilisation des rôles thématiques a été présentée dans la section 1.3.2. Notre calcul rendra alors compte de ces phénomènes. Comme nous allons le voir, l'intégration de ces hypothèses a une conséquence directe : une variable argument du verbe doit, en plus de porter sa sémantique, trouver son rôle thématique. Pour modéliser ces relations, nous introduisons différents types de variables ainsi que la réification des formules. Le système de types mis en œuvre est une version enrichie de celui de Montague.

Nous commencerons par proposer un morphisme de types servant de base à la rédaction des $\lambda\mu$-termes des lexiques, puis nous présenterons la formation des lexiques. À partir de ces lexiques, nous associerons à chaque règle du calcul syntaxique une règle du calcul sémantique permettant de mettre en œuvre cette interface sur des exemples.

7.2.1 Morphisme de types

Comme nous l'avons précédemment évoqué, le calcul sémantique est basé sur un système de types qui est une extension de Montague, [Mon73]. Dans ce calcul, des variables d'événement servent à réifier les formules (la réification des formules est le fait de modéliser la formule par une variable). Nous reviendrons sur la réification dans la section suivante.

Définition 114. *Soit $S_e = \{e, t, \iota\}$ l'ensemble des types sémantiques atomiques où :*
- *e est un individu,*
- *t est un booléen (une valeur de vérité),*
- *ι est un événement.*

Soit $S_f = \{e, \iota\}$ et soit $f_i \in S_f$ pour $i \in \mathbb{N}$: on définit un prédicat à n places comme étant un élément du type :

$$f_1 \to (f_2 \to (f_3 \to (\cdots \to t)\ldots))$$

On définit une fonction à n arguments comme étant un élément du type :

$$f_1 \to (f_2 \to (f_3 \to (\cdots \to f_n)\ldots))$$

On note S_t l'ensemble des types construits à partir de S_e, des types des prédicats et des fonctions.

On définit les constantes du système :
- $\wedge, \vee$ *:* $(t \to t) \to t$
- $\forall, \exists$ *:* $(e \to t) \to t$

Pour définir entièrement notre morphisme de types, nous devons revenir sur une condition de bonne formation des entrées lexicales. Dans les GMCs, les entrées dont les formules utilisent le $\otimes$ doivent combiner une formule de P_1 (une catégorie de base) avec des formules de P_2 (traits de l'item lexical). Nous définissons le type de ces formules comme étant relatif à la catégorie de base.

Définition 115. *Soit l'homorphisme de type $H : S_t \to S_t$ défini inductivement par :*
- $H(a\backslash b) = H(b/a) = H(a) \to H(b)$
- $H(a \otimes b) = H(b)$ *si* $a \in$ P$_2$, $H(a)$ *sinon.*

D'après la définition de la bonne formation des entrées des GMCs proposée à la section 6.1.1, la séquence de $\otimes$ est composée d'un unique élément de P$_1$*, les autres étant dans* P$_2$*. La définition inductive basée sur la séquence de $\otimes$ nous donnera un unique type résultat.*

Les catégories usuelles de P$_1$ *sont du type :*

autour du nom		*autour du verbe*	
$H(d) =$	e	$H(V) =$	$e \to (\iota \to t)$
$H(n) =$	$e \to t$	$H(v) =$	$\iota \to t$
		$H(t) =$	$\iota \to t$

Dans notre système, la catégorie V correspond à un élément devant être combiné avec un individu (qui sera son sujet) et un événement (c'est la conséquence de réification introduite dans notre système). De plus, les différentes formules reprenant le verbe modifié, après que ses arguments ont été introduits $(c, t, \cdots)$ ont toutes le même type $\iota \to t$.

Exemple 116. *On obtient sur un lexique les types sémantiques suivants :*

un	$(k \otimes d)/n$	$(e \to t) \to e$	*lire*	V/d	$e \to (e \to (\iota \to t))$
les	$(k \otimes d)/n$	$(e \to t) \to e$		$k\backslash d\backslash v/V$	$(e \to \iota \to t) \to (e \to (e \to (\iota \to t)))$
enfants	n	$(e \to t)$	*infl*	$k \setminus t/v$	$(\iota \to t) \to (e \to (\iota \to t))$
livre	n	$(e \to t)$	*comp*	c/t	$(\iota \to t) \to t$

7.2.2 Décomposition de la sémantique

Les premières propositions d'interface se heurtaient à des problèmes calculatoires majeurs. Le λ-calcul est un calcul permettant d'attribuer aisément une formule à un énoncé fortement normé. Mais pour les analyses de phénomènes s'éloignant de la norme, comme les questions, le liage des variables ne se fait plus selon l'ordre supposé. La particularité des GMCs est de mettre en attente un constituant dans le calcul pour l'intégrer en fin de dérivation. L'apport sémantique d'un constituant doit pouvoir se faire a posteriori de l'introduction de sa variable.

On remarque que dans le calcul syntaxique, lors de la substitution, on lie deux variables par une même formule. Pour cela, dans le calcul sémantique, on suppose que la sémantique des éléments de type $\otimes$ est celle de l'élément de P$_1$. Ceci est porté dans le morphisme de types par le fait que $A \otimes B$ ait le type de l'élément de P$_1$.

Nous appelons SEM le calcul sémantique. On associe une structure sémantique aux catégories de base (*DP*, *verbe*, ...) et une contrepartie sémantique aux opérations syntaxiques. L'opération *merge* trouve aussi son interprétation dans l'application fonctionnelle. L'interprétation des traits des constituants (traits de déplacement) et de l'opération de déplacement est plus complexe. Pour y parvenir, nous utilisons des concepts issus de la *Discourse Representation Theory* (DRT), [KR93] [RMV97] (qui est basée sur les Discourse Representation Structures - DRS) :

1. introduction d'un ensemble de référents, de type ι (permettant de réifier les formules)
2. déconstruction des quantificateurs.

La réification des formules est le fait de les matérialiser par une variable référente de discours. Ces dernières forment un ensemble distinct des variables du calcul mais ayant le même statut que les variables libres. Elles sont liées lors de la transformation de la DRS en formule de la logique des prédicats.

On appelle $\lambda\mu$-DRS les termes de SEM.

Définition 117. *Soit V_L l'ensemble des variables, notées $x_1, \cdots x_n$, et V l'ensemble des variables du calcul, notées $x, y, z \cdots$. La syntaxe des $\lambda\mu$-DRS est :*

$$\begin{array}{lll} \gamma & ::= & (P\ x)|(\alpha\ x)|x_1 = x_2|\neg K|K_1 \wedge K_2|K_1 \vee K_2|K_1 \Rightarrow K_2 \\ K & ::= & [x_1 ... x_n|\gamma_1, ..., \gamma_m] \end{array}$$

Le symbole | dans la définition de K n'est pas un "ou" traditionnel, mais un simple séparateur. Cette notation ambiguë à ce stade des définitions permet une lecture simplifiée dans la suite de la présentation.

En utilisant les λ et μ-abstractions sur ces termes, on retrouve des termes du $\lambda\mu$-calcul où la quantification est réalisée par des DRS. Par exemple, un terme de SEM *est :*

$$u :: \lambda Q.\mu\alpha.[x|(Q\ x) \wedge (\alpha\ x)]$$

Ce terme est celui d'un article pour une quantification existentielle. Par composition avec un nom (par exemple livre), on obtiendra la $\lambda\mu$-DRS $\mu\alpha.[x|(livre\ x) \wedge (\alpha\ x)]$ de type e(comme nous l'avons exposé précédemment). De la même manière, la quantification universelle est représentée par :

$$v :: \lambda Q.\mu\alpha.[\ |[x|(Q\ x)] \Rightarrow [\ |(\alpha\ x)]]$$

On dira que u et v sont dépendants de la variable libre du calcul x. On note $u(x)$ les termes de SEM.

Cette manière de rendre compte de la quantification a deux conséquences :

1. la portée des quantificateurs n'est pas dépendante des principes de *c-command* (un quantificateur peut lier le reste d'une expression même s'il reste à sa place).
2. les expressions quantifiées n'utilisent pas de type élevé et elles peuvent prendre leur portée de quantification à partir du terme où elles sont positionnées.

La première remarque provient de l'hypothèse que tous les traits d'un constituant doivent porter une part de sa sémantique. Ainsi, un prédicat sur une variable peut se retrouver en dehors de la portée de son quantificateur. La seconde remarque est due aux propriétés du $\lambda\mu$-calcul.

Dans le calcul sémantique, nous modélisons les *Uniform Theta Assignment Hypothesis* [Bak97]. Pour en rendre compte, des prédicats représentant les rôles thématiques sont introduits. On trouvera alors pour le traitement d'un verbe transitif des assignations de variables aux prédicats $\lambda x.patient(x)$ et $\lambda x.agent(x)$.

Les analyses de Kratzer, [Kra94], ou Marantz, [Mar84], avancent l'idée que les arguments ont un statut particulier. Par exemple, l'argument agentif est un argument externe au verbe. Le rôle sémantique de l'agent n'est donc pas apporté par le verbe lui-même mais par un autre mécanisme. Pour Kratzer, c'est la *Voie* qui réalise cela (*Voie* est la tête de

l'élément $VoieP$ qui domine VP). Lors de la fusion entre $VoieP$ et VP, il s'agit alors de combiner deux termes sémantiques distincts (par un mécanisme *ad hoc*).

Ici, cette étape de $Voie$ est remplacée par un type foncteur qui utilise un v complété et un marqueur de cas. Le rôle d'agent est alors simplement ajouté au verbe par une application (dans l'exemple d'analyse de phrase au passif, nous verrons que c'est cette entrée qui est cruciale). Les arguments du prédicat représentant le rôle thématique et du prédicat principal sont unifiés quand le DP vient occuper sa position de spécifieur : c'est précisément ce qui se passe quand l'hypothèse d (qui a introduit un argument d'individu dans le prédicat) et l'hypothèse k (qui a introduit un argument dans la représentation thématique) sont déchargées simultanément par un DP.

Cette description n'est pas limitée au rôle d'agent, mais prévaut pour tous les rôles thématiques. Donc chaque DP doit porter un cas syntaxique et un rôle thématique. Si le verbe n'apporte pas cette information, les prépositions ont ce rôle.

Comme nous l'avons vu, la formule d'un verbe transitif est V/d, de type sémantique $e \to e \to \iota \to t$. Son terme sémantique est

$$\lambda x \lambda y \lambda e.verbe(e, x, y)$$

(pour lequel le typage est correct). La première étape de la dérivation est de saturer l'une de ses positions. Puis le verbe est fusionné avec un modifieur, de type $(\iota \to t) \to e \to e \to \iota \to t$:

$$\lambda P \lambda x_2 \lambda y \lambda e.P(y, e) \wedge patient(e, x_2)$$

C'est cette entrée qui apportera le premier rôle thématique (patient) et joue le rôle de *voie* dans SEM.

Théoriquement, la variable qui a pris position dans le verbe et celle ayant pris position dans le rôle doivent être liées. Le rôle doit alors être assigné à la variable du DP ($DP(x)$) qui se déplace pour occuper la position de spécifieur. Dans le calcul syntaxique, ceci est réalisé par le trait formel k. Il s'agit alors de substituer simultanément aux variables introduites dans le terme sémantique le terme correspondant au DP et la variable dont il dépend. Seule la variable qui a pris position dans le verbe (dans la structure argumentale du verbe) recevra le terme du DP. L'autre variable sera substituée par la variable du terme du DP (la variable quantifiée). Cette étape a donc pour but d'*identifier les variables*, c'est la contrepartie sémantique de *move*.

Ce mécanisme permet de différencier le rôle thématique occupé par une variable dans un énoncé de son apport dans la structure argumentale. Après cette étape de substitution, les deux variables introduites dans la première partie du calcul deviennent identiques.

Les UTAH supposent que le rôle *patient* est assigné en position de spécifieur d'un verbe, *thème* en position de spécifieur d'un VP (après la construction du groupe verbal) et *agent* en spécifieur d'un IP (après que le verbe est reçu son inflexion).

Lexique 118. *Voici quelques exemples des principales catégories syntaxiques et leur $\lambda\mu$-DRS.*

enfants	d	$e \to t$	$\lambda z.enfant(z)$
les	$(k \otimes d)/n$	$(e \to t) \to e$	$\lambda P.\mu Q.[d \mid P(d) \wedge Q(d)]$
lire	V/d	$e \to e \to \iota \to t$	$\lambda x \lambda y \lambda e.lire(e, x, y)$
comp	c/t	$(\iota \to t) \to t$	$\lambda P.P(e)$

7.2.3 Règles de l'interface

Soit SEM le calcul sémantique. Les éléments de SEM sont des $\lambda\mu$-DRS composées à partir de V_L et V. On note $t[u]$ une $\lambda\mu$-DRS t contenant la variable $u \in V$, libre dans t.

Dans SEM, les règles sont synchronisées avec celles des GMCs :

– les fusions sont des applications fonctionnelles,

– les déplacements sont des substitutions d'hypothèses.

Cette procédure permet d'abstraire la variable en structure profonde, puis de lui substituer les termes vus comme des variables grâce au $\lambda\mu$-calcul.

Définition 119. *On étiquette le calcul syntaxique par des $\lambda\mu$-DRS. Les règles de l'interface sont alors :*

$$\frac{x : \Gamma \vdash A/B \quad y : \Delta \vdash B}{(x)y : \Gamma, \Delta \vdash A}\,[mg]$$

$$\frac{y : \Delta \vdash B \quad x : \Gamma \vdash B \setminus A}{(x)y : \Delta, \Gamma \vdash A}\,[mg]$$

Move est l'élimination d'un tenseur et la substitution des variables associées dans le calcul sémantique :

$$\frac{x(c) : \Gamma \vdash A \otimes B \quad y[u, v] : \Delta[u : A, v : B] \vdash C}{y[c/u, x(c)/v] : \Delta[\Gamma] \vdash C}\,[mv]$$

Le déplacement est donc la substitution à deux variables libres d'une $\lambda\mu$-DRS et de sa variable libre. Ceci permet d'associer à chaque trait d'un constituant une contrepartie sémantique.

Certaines opérations des GMCs sont réalisées à partir d'hypothèses : si la formule associée à cette hypothèse ne comporte pas de $\otimes$, leur contrepartie sémantique est alors une variable de V, libre dans la $\lambda\mu$-DRS ; sinon, c'est une variable à laquelle on associe une variable libre de V_L (représentant la variable liée dans le terme que l'on substituera).

Cependant, cette procédure de substitution dans le terme fait qu'une variable est en dehors de la portée de son quantificateur. Nous utiliserons donc une phase d'*internalisation* des prédicats à la fin du calcul porté par la syntaxe. Ainsi, deux formules seront dites équivalentes lorsque les prédicats sur une variable seront dans la portée de son quantificateur. La non-linéarité de cette partie du calcul sémantique est assumée pour des prédicats assignant une condition à une variable de type e et une variable de type ι.

SEM est composé de deux phases. La première est synchronisée au calcul syntaxique des GMCs, puis la seconde internalise les prédicats, pour que les variables soient liées par les quantificateurs, et résout les μ-réductions ainsi que la simplification des μ-abstracteurs. Pour cela nous associons à chaque entrée lexicale une $\lambda\mu$-DRS qui respecte le typage induit par le morphisme type.

Après que la dérivation syntaxique soit terminée, SEM se poursuit. On définit alors deux opérations supplémentaires.

Définition 120. *L'internalisation des $\lambda\mu$-DRSs est le fait de rassembler dans la portée d'un quantificateur tous les prédicats dépendants de la variable quantifiée.*

Les prédicats sur des variables interprétées comme libres dans une formule contenant une DRS sur cette variable sont déplacés pour lier cette variable. Ainsi : $[x|P(x)] \wedge Q(x) \rightarrow [x|P(x) \wedge Q(x)]$

De plus, on réécrit les applications dans ces $\lambda\mu$-DRSs ainsi : $f(x,y) \rightarrow ((f\ x),y)$

Définition 121. *La réduction des $\lambda\mu$-DRS est la clôture d'une $\lambda\mu$-DRS par la μ-réduction, la μ'-réduction et la simplification.*

La partie de SEM synchronisée sur les GMCs produit des $\lambda\mu$-DRS non réduites. La partie spécifique à SEM est composée de deux phases :

1. internalisation des prédicats
2. réduction des $\lambda\mu$-DRS.

Cette dernière phase n'est pas confluente. Cependant, l'utilisation des μ-abstractions pour représenter la quantification permet de calculer les différentes portées de quantification.

On résume le calcul sémantique par la figure 42. Les règles du calcul sont reprises dans l'annexe D.4.

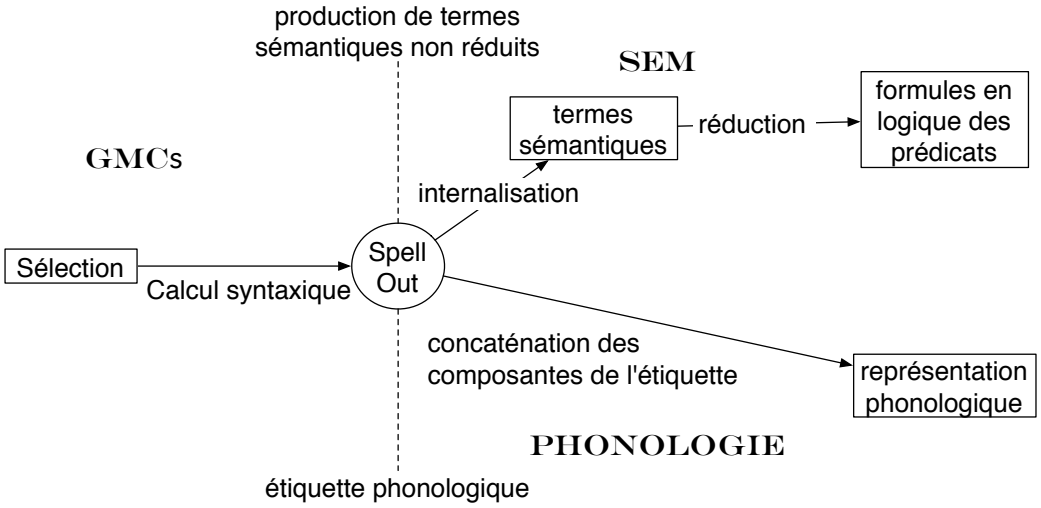


FIG. 42 – calculs dans les GMCs.

7.3 Exemples

Dans cette section, nous présentons trois exemples d'analyses sémantiques pour :

1. une phrase Sujet-Verbe-Objet
2. une question portant sur l'objet du verbe. Dans ces exemples, le liage des variables par le λ-calcul est inversé. Nous verrons comment réaliser le liage correct dans ces cas-là.

3. une phrase au passif. Dans ce type de phrase, les rôles thématiques ne sont plus ceux des cas.

Nous reprenons les analyses syntaxiques présentées dans le chapitre précédent (section 6.3) On présente les différentes étapes du calcul syntaxique étiquetées par les $\lambda\mu$-DRSs. Pour chaque preuve, le type des formules est associé en dessous. Pour des raisons de simplification nous utilserons *mg* pour *merge* et *mv* pour *move*

7.3.1 phrase Sujet-Verbe-Objet

Analyse de la phrase :

(29) les enfants lisent un livre.

On utilise le lexique suivant :

Lexique 122.

les	$\lambda P.\mu Q.[\|[d\|P(d) \Rightarrow Q(d)]]$:	$(e \to t) \to e$	$\vdash k \otimes d/n$
un	$\lambda P.\mu Q.[c\|P(c) \wedge Q(c)]$:	$(e \to t) \to e$:	$\vdash (k \otimes d)/n$
enfants	$\lambda z.enfant(z)$:	$e \to t$	$\vdash n$
livre	$\lambda z.livre(z)$:	$e \to t$	$\vdash n$
lire	$\lambda x\lambda y\lambda e.lire(e,x,y)$:	$e \to e \to \iota \to t$	$\vdash V/d$
modi	$\lambda P\lambda x_2\lambda y\lambda e.P(y,e) \wedge patient(e,u)$:	$(e \to \iota \to t) \to e \to e \to \iota \to t$	$\vdash (k\backslash(d\backslash v))/_< V$
inflexion	$\lambda P\lambda y_2\lambda e.P(e) \wedge present(e) \wedge agent(e,y_2)$:	$(\iota \to t) \to e \to \iota \to t$	$\vdash (k \setminus t)/_< v$
comp	$\lambda P.P(e)$:	$(\iota \to t) \to t$	$\vdash c/t$

La dérivation commence par saturer le premier argument du verbe qui introduit une variable libre :

$$\dfrac{\begin{array}{cc}\lambda x\lambda y\lambda e.lire(e,x,y) :\vdash V/d & u : d \vdash d \\ e \to e \to \iota \to t & e\end{array}}{\begin{array}{c}\lambda y\lambda e.lire(e,u,y) : d \vdash V \\ e \to \iota \to t\end{array}}\,[mg]$$

Puis le comportement syntaxique du verbe est modifié. Cette entrée ajoute dans la formule le rôle thématique *patient* :

$$\dfrac{\begin{array}{cc}\lambda P\lambda x_2\lambda y\lambda e.P(y,e) \wedge patient(e,x_2) :\vdash (k\backslash(d\backslash v))/_< V & \lambda y\lambda e.lire(e,u,y) : d \vdash V \\ (e \to \iota \to t) \to e \to e \to \iota \to t & e \to \iota \to t\end{array}}{\begin{array}{c}\lambda x_2\lambda y\lambda e.lire(e,u,y) \wedge patient(e,x_2) : d \vdash k\backslash(d\backslash v) \\ e \to e \to \iota \to t\end{array}}\,[mg]$$

Puis la dérivation introduit une nouvelle variable libre représentant le cas. Elle est introduite dans le prédicat thématique :

$$\dfrac{\begin{array}{cc}v : k \vdash k & \lambda x_2\lambda y\lambda e.lire(e,u,y) \wedge patient(e,x_2) : d \vdash k\backslash(d\backslash v) \\ e & e \to e \to \iota \to t\end{array}}{\begin{array}{c}\lambda y\lambda e.lire(e,u,y) \wedge patient(e,v) : k, d \vdash d\backslash v \\ e \to \iota \to t\end{array}}\,[mg]$$

Parallèlement on construit le DP objet *un livre* :

$$\frac{\begin{array}{cc}\lambda P.\mu Q.[c|P(c)\wedge Q(c)] :\vdash (k\otimes d)/n & \lambda z.livre(z) :\vdash n\\ (e\to t)\to e & e\to t\end{array}}{\begin{array}{c}\mu Q.[c|livre(c)\wedge Q(c)] :\vdash k\otimes d\\ e\end{array}}\,[mg]$$

que l'on introduit dans la dérivation principale par un *move* qui substitue aux variables libres la $\lambda\mu$-DRS et la variable quantifiée (appartenant à V_L ou libre dans le calcul) :

$$\frac{\begin{array}{cc}\mu Q.[c|livre(c)\wedge Q(c)] :\vdash k\otimes d & \lambda y\lambda e.lire(e,u,y)\wedge patient(e,v) : k, d\vdash d\backslash v\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda y\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],y)\wedge patient(e,c) : d\vdash k\backslash(d\backslash v) : k, d\vdash d\backslash v\\ e\to\iota\to t\end{array}}\,[mv]$$

Puis, on introduit une nouvelle variable libre correspondant au DP sujet :

$$\frac{\begin{array}{cc}w : d\vdash d & \lambda y\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],y)\wedge patient(e,c) :\vdash d\backslash v\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],w)\wedge patient(e,c) :\vdash v\\ \iota\to t\end{array}}\,[mg]$$

Le verbe reçoit son inflexion. L'étape IP est atteinte, le second rôle thématique *agent* est introduit dans la dérivation :

$$\frac{\begin{array}{cc}\lambda P\lambda y_2\lambda e.P(e)\wedge present(e)\wedge agent(e,y_2) : & \lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],w)\wedge\\ \vdash (k\setminus t)/_< v & patient(e,c) :\vdash v\\ (\iota\to t)\to e\to\iota\to t & \iota\to t\end{array}}{\begin{array}{c}\lambda y_2\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],w)\wedge patient(e,c)\wedge present(e)\\ \wedge agent(e,y_2) :\vdash k\setminus t\\ e\to\iota\to t\end{array}}\,[mg]$$

La preuve introduit une seconde variable de rôle thématique qui permettra de réaliser le déplacement du sujet :

$$\frac{\begin{array}{cc}y : k\vdash k & \lambda y_2\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],w)\wedge patient(e,c)\wedge present(e)\wedge\\ e & agent(e,y_2) :\vdash k\setminus t\\ & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda e.lire(e,\mu Q.[c|livre(c)\wedge Q(c)],w)\wedge patient(e,c)\wedge present(e)\wedge\\ agent(e,y) :\vdash t\\ \iota\to t\end{array}}\,[mg]$$

Parallèlement, on construit le DP sujet *les enfants* :

$$\frac{\begin{array}{cc}\lambda P\mu Q.[|[d|P(d)\Rightarrow Q(d)]] :\vdash (k\otimes d)/n & \lambda z.enfant(z) :\vdash n\\ (e\to t)\to e & e\to t\end{array}}{\begin{array}{c}\mu Q.[|[d|enfant(d)\Rightarrow Q(d)]] :\vdash k\otimes d\\ e\end{array}}\,[mg]$$

que l'on substitue dans la dérivation principale :

$$\frac{\begin{array}{c}\mu Q.[|[d|enfant(d) \Rightarrow Q(d)]] : \\ \vdash k \otimes d \\ e\end{array} \qquad \begin{array}{c}\lambda e.lire(e, \mu Q.[c|livre(c) \wedge Q(c)], w) \wedge \\ patient(e,c) \wedge present(e) \wedge agent(e,y) :\vdash t \\ \iota \rightarrow t\end{array}}{\begin{array}{c}\lambda e.lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge \\ patient(e,c) \wedge present(e) \wedge agent(e,d) :\vdash t \\ \iota \rightarrow t\end{array}}\ [mv]$$

À la fin de la preuve, l'entrée *comp* termine la dérivation en introduisant la variable de réification de la formule :

$$\frac{\begin{array}{c}\lambda P.P(e) :\vdash c/t \\ (\iota \rightarrow t) \rightarrow t\end{array} \qquad \begin{array}{c}\lambda e.lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge \\ patient(e,c) \wedge present(e) \wedge agent(e,d) :\vdash t \\ \iota \rightarrow t :\vdash t\end{array}}{\begin{array}{c}lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge \\ patient(e,c) \wedge present(e) \wedge agent(e,d) :\vdash c \\ t\end{array}}\ [mg]$$

Le calcul sémantique fournit la formule :

$$lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e,c) \wedge present(e) \wedge agent(e,d), \text{ de type } t$$

La partie de SEM synchronisée au calcul syntaxique des GMCs est terminée. On réalise alors l'internalisation des prédicats :

$$((lire(e), \mu Q.[c|livre(c) \wedge (Qc) \wedge patient(e,c)]), \mu P.[|[d|enfant(d) \Rightarrow [|(Pd) \wedge agent(e,d)]]) \wedge present(e)$$

Il faut alors réaliser les μ-réductions. Deux scénari de réduction sont possibles : soit on réalise toutes les réductions sur l'une des μ-variables, soit on applique les réductions successivement sur chacune des μ-variables.

Premier scénario de μ-réduction :

1. μ'-réduction : (Qc) est remplacé par $Qlire(e)c$:
   $\mu Q.[c|livre(c) \wedge (Q(lire(e)c)) \wedge patient(e,c)]), \mu P.[|[d|enfant(d) \Rightarrow [|(Pd) \wedge agent(e,d)]]) \wedge present(e)$
2. μ-réduction : $(Q(lire(e)c))$ est remplacé par $\mu P.(P((lire(e)c)([d|enfant(d) \Rightarrow Q(d) \wedge agent(e,d)]))))$:
   $\mu Q.[c|livre(c) \wedge (Q((lire(e)c)(\mu P.[|[d|enfant(d) \Rightarrow [|(Pd) \wedge agent(e,d)]]) \wedge patient(e,c)])) \wedge present(e)$
3. simplification :
   $[c|livre(c) \wedge ((lire(e)c)(\mu P.[|[d|enfant(d) \Rightarrow [|(Pd) \wedge agent(e,d)]]) \wedge patient(e,c)]) \wedge present(e)$
4. μ-réduction : (Pd) est remplacé par $(P((lire(e)c)d))$:
   $[c|livre(c) \wedge (\mu P.[|[d|enfant(d) \Rightarrow [|(P((lire(e)c)d)) \wedge agent(e,d)]]) \wedge patient(e,c)]) \wedge present(e)$
5. simplification :
   $[c|livre(c) \wedge ([|[d|enfant(d) \Rightarrow [|((lire(e)c)d) \wedge agent(e,d)]]) \wedge patient(e,c)]) \wedge present(e)$

Cette lecture est celle pour laquelle à un moment e il y a un livre que tous les enfants lisent.

Second scénario de μ-réduction :

1. μ'-réduction : (Qc) est remplacé par $Qlire(e)c$:
   $\mu Q.[c|livre(c) \wedge (Q(lire(e)c)) \wedge patient(e,c)]), \mu P.[|[d|enfant(d)] \Rightarrow [|(Pd) \wedge agent(e,d)]) \wedge present(e)$
2. μ'-réduction : (Pd) est remplacé par $(P(\mu Q.[c|\cdots]))$:
   $\mu P.[|[d|enfant(d)] \Rightarrow [|(P(\mu Q.[c|livre(c) \wedge (Q(lire(e)c)) \wedge patient(e,c)], d)) \wedge agent(e,d)]]) \wedge present(e)$
3. simplification (Q c) :
   $[|[d|enfant(d)] \Rightarrow [|(\mu Q.[c|livre(c) \wedge (Q(lire(e)c)) \wedge patient(e,c)], d) \wedge agent(e,d)]]) \wedge present(e)$
4. μ-réduction : $(Q(lire(e)c))$ est remplacé par $(P((lire(e)c))d$:
   $[|[d|enfant(d)] \Rightarrow [|(\mu Q.[c|livre(c) \wedge (Q((lire(e)c)d) \wedge patient(e,c)] \wedge agent(e,d)]]) \wedge present(e)$
5. simplification :
   $[|[d|enfant(d)] \Rightarrow [|[c|livre(c) \wedge ((lire(e)c)d) \wedge patient(e,c)] \wedge agent(e,d)]] \wedge present(e)$

Cette lecture correspond à la situation où pour chaque enfant, il y a un livre qui est lu par lui (ou elle) au moment e.

7.3.2 Questions

Analyse de la phrase :

(30) Quel livre lisent les enfants ?

On utilise le lexique suivant :

Lexique 123.

les	$\lambda P.\mu Q.[|[d|P(d) \wedge Q(d)]] :$	$(e \to t) \to e$	$\vdash k \otimes d/n$
Quel	$\lambda P.\mu Q.[c|P(c) \wedge Q(c)] :$	$(e \to t) \to e$	$\vdash (wh \otimes (k \otimes d))/n$
enfants	$\lambda z.enfant(z) :$	$e \to t$	$\vdash n$
livre	$\lambda z.livre(z) :$	$e \to t$	$\vdash n$
lire	$\lambda x \lambda y \lambda e.lire(e,x,y) :$	$e \to e \to \iota \to t$	$\vdash V/d$
modifeur	$\lambda P \lambda x_2 \lambda y \lambda e.P(y,e) \wedge patient(e,u) :$	$(e \to \iota \to t) \to e \to e \to \iota \to t$	$\vdash (k \backslash (d \backslash v))/_< V$
inflexion	$\lambda P \lambda y_2 \lambda e.P(e)$		
	$\wedge present(e) \wedge agent(e,y_2) :$	$(\iota \to t) \to e \to \iota \to t$	$\vdash (k \setminus t)/_< v$
comp	$\lambda P \lambda u.P(e) \wedge question(e,u)$	$(\iota \to t) \to e \to t$	$\vdash (wh \setminus c)/_< t$

Cette analyse est similaire à la précédente à l'exception du traitement du *DP* objet qui possède un trait supplémentaire qui correspond au trait de question. On utilise un prédicat *question* pour le représenter. Le *DP* est substitué à trois variables libres. Pour cela, on utilise une hypothèse composée représentant la $\lambda\mu$-DRS et la variable qu'elle quantifie.

La preuve commence par l'introduction d'une variable libre :

$$\dfrac{\begin{array}{cc} \lambda x \lambda y \lambda e.lire(e,x,y) :\vdash V/d & u : d \vdash d \\ e \to e \to \iota \to t & e \end{array}}{\begin{array}{c} \lambda y \lambda e.lire(e,u,y) : d \vdash V \\ e \to \iota \to t \end{array}} [mg]$$

Puis, comme dans la dérivation précédente, le modifieur du verbe introduit le rôle *patient* :

$$\frac{\begin{array}{cc}\lambda P\lambda x_2\lambda y\lambda e.P(y,e)\wedge patient(e,x_2) :\vdash (k\backslash(d\backslash v))/_< V & \lambda y\lambda e.lire(e,u,y) : d \vdash V\\ (e\to\iota\to t)\to e\to e\to\iota\to t & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda x_2\lambda y\lambda e.lire(e,u,y)\wedge patient(e,x_2) : d\vdash k\backslash(d\backslash v)\\ e\to e\to\iota\to t\end{array}}\,[mg]$$

La preuve utilise une seconde variable libre :

$$\frac{\begin{array}{cc}v : k\vdash k & \lambda x_2\lambda y\lambda e.lire(e,u,y)\wedge patient(e,x_2) : d\vdash k\backslash(d\backslash v)\\ e & e\to e\to\iota\to t\end{array}}{\begin{array}{c}\lambda y\lambda e.lire(e,u,y)\wedge patient(e,v) : d\vdash k\backslash(d\backslash v) : k, d\vdash d\backslash v\\ e\to\iota\to t\end{array}}\,[mg]$$

Le *DP* ne peut pas être introduit à ce moment de la preuve car il nécessite un trait supplémentaire dans la dérivation. On lie les deux hypothèses / variables libres par une variable libre r dépendant d'une variable libre du calcul $t(\in V_L)$:

$$\frac{\begin{array}{cc}r(t) : k\otimes d\vdash k\otimes d & \lambda y\lambda e.lire(e,u,y)\wedge patient(e,v) : d\vdash k\backslash(d\backslash v) : k, d\vdash d\backslash v\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda y\lambda e.lire(e,r(t),y)\wedge patient(e,t) : d\vdash k\backslash(d\backslash v) : k, d\vdash d\backslash v\\ e\to\iota\to t\end{array}}\,[mv]$$

Enfin, on introduit la variable libre correspondant au DP sujet :

$$\frac{\begin{array}{cc}w : d\vdash d & \lambda y\lambda e.lire(e,r(t),y)\wedge patient(e,t) : d\vdash k\backslash(d\backslash v) : k, d\vdash d\backslash v\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda e.lire(e,r(t),w)\wedge patient(e,t) : d\vdash k\backslash(d\backslash v) : k, d\vdash v\\ \iota\to t\end{array}}\,[mg]$$

La preuve atteint l'étape IP. À nouveau, on introduit le prédicat *agent* :

$$\frac{\begin{array}{cc}\lambda P\lambda y_2\lambda e.P(e)\wedge & \\ agent(e,y_2) :\vdash (k\setminus t)/_< v & \lambda e.lire(e,r(t),w)\wedge patient(e,t) : k, d\vdash v\\ (\iota\to t)\to e\to\iota\to t & \iota\to t\end{array}}{\begin{array}{c}\lambda y_2\lambda e.lire(e,r(t),w)\wedge patient(e,t)\wedge present(e)\wedge agent(e,y_2) : d, k\otimes d\vdash k\setminus t\\ e\to\iota\to t\end{array}}\,[mg]$$

La preuve nécessite alors l'introduction d'une nouvelle variable libre, correspondant au cas :

$$\frac{\begin{array}{cc} & \lambda y_2\lambda e.lire(e,r(t),w)\wedge patient(e,t)\wedge present(e)\wedge agent(e,y_2) :\\ y : k\vdash k & d, k\otimes d\vdash k\setminus t\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda e.lire(e,r(t),w)\wedge patient(e,t)\wedge present(e)\wedge agent(e,y) : k, d, k\otimes d\vdash t\\ \iota\to t\end{array}}\,[mg]$$

Parallèlement, on construit le *DP* sujet *les enfants* :

$$\dfrac{\begin{array}{cc}\lambda P\mu Q.[|[d|P(d) \Rightarrow Q(d)]] :\vdash (k \otimes d)/n & \lambda z.enfant(z) :\vdash n \\ (e \to t) \to e & e \to t\end{array}}{\begin{array}{c}\mu Q.[|[d|enfant(d) \Rightarrow Q(d)]] :\vdash k \otimes d \\ e\end{array}}\,[mg]$$

Il se déplace dans la dérivation principale et prend ses positions dans la $\lambda\mu$-DRS :

$$\dfrac{\begin{array}{cc}\mu Q.[|[d|enfant(d) \Rightarrow Q(d)]] :\vdash k \otimes d & \lambda e.lire(e, r(t), w) \wedge patient(e, t) \wedge \\ e & present(e) \wedge agent(e, y) : k, d, k \otimes d \vdash t \\ & \iota \to t\end{array}}{\begin{array}{c}\lambda e.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, t) \wedge \\ present(e) \wedge agent(e, d) : k \otimes d \vdash t \\ \iota \to t\end{array}}\,[mv]$$

À la fin de la preuve, l'entrée *comp* introduit un prédicat *question* permettant de spécifier sur quelle variable la question est posée.

$$\dfrac{\begin{array}{cc}\lambda P\lambda u.P(e) \wedge question(e, u) : & \lambda e.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge \\ \vdash (wh \setminus c)/_{<}t & patient(e, t) \wedge present(e) \wedge agent(e, d) : k \otimes d \vdash t \\ (\iota \to t) \to e \to t & \iota \to t\end{array}}{\begin{array}{c}\lambda z.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, t) \wedge \\ present(e) \wedge agent(e, d) \wedge question(e, z) : k \otimes d \vdash wh \setminus c \\ e \to t\end{array}}\,[mg]$$

On introduit une variable libre qui correspond à la variable sur laquelle la question est posée :

$$\dfrac{\begin{array}{cc}z : wh \vdash wh & \lambda z.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, t) \wedge \\ e & present(e) \wedge agent(e, d) \wedge question(e, z) : k \otimes d \vdash wh \setminus c \\ & e \to t\end{array}}{\begin{array}{c}\lambda z.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, t) \wedge \\ present(e) \wedge agent(e, d) \wedge question(e, z) : wh, k \otimes d \vdash c \\ t\end{array}}\,[mg]$$

Parallèlement, on construit le *DP* objet *Quel livre*

$$\dfrac{\begin{array}{cc}\lambda P.\mu Q.[c|P(c) \wedge Q(c)] :\vdash (wh \otimes (k \otimes d))/n & \lambda z.livre(z) :\vdash n \\ (e \to t) \to e & e \to t\end{array}}{\begin{array}{c}\mu Q.[c|livre(c) \wedge Q(c)] :\vdash wh \otimes (k \otimes d) \\ e\end{array}}\,[mg]$$

Il est substitué dans la dérivation principale en utilisant la variable $r(t)$ qui permet de propager t au travers de la séquence de $\otimes$.

$$\dfrac{\begin{array}{cc}\mu Q.[c|livre(c) \wedge Q(c)] : & \lambda z.lire(e, r(t), \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, t) \wedge \\ \vdash wh \otimes (k \otimes d) & present(e) \wedge agent(e, d) \wedge question(e, z) : wh, k \otimes d \vdash c \\ e & t\end{array}}{\begin{array}{c}lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, c) \wedge \\ present(e) \wedge agent(e, d) \wedge question(e, c) :\vdash c \\ t\end{array}}\,[mv]$$

Le calcul sémantique fournit la formule :

$$lire(e, \mu Q.[c|livre(c) \wedge Q(c)], \mu Q.[|[d|enfant(d) \Rightarrow Q(d)]]) \wedge patient(e, c) \wedge present(e) \wedge agent(e, d) \wedge question(e, c), \text{ de type } t$$

La phase d'internalisation des prédicats fournit la $\lambda\mu$-DRS suivante :

$$((lire(e), \mu Q.[c|livre(c) \wedge Q(c) \wedge patient(e, c) \wedge question(e, c)]), \mu Q.[|[d|enfant(d) \Rightarrow Q(d) \wedge agent(e, d)]]) \wedge present(e), \text{ de type } t$$

Après les μ-réductions et simplifications, SEM fournit les deux formules :

1. $[|[d|enfant(d) \Rightarrow [c|livre(c) \wedge ((lire(e)c)d) \wedge patient(e, c) \wedge question(e, c)] \wedge agent(e, d)]] \wedge present(e)$
2. $[c|livre(c) \wedge [|[d|enfant(d) \Rightarrow ((lire(e)c)d) \wedge agent(e, d)]] \wedge patient(e, c) \wedge question(e, c)] \wedge present(e)$

La première correspond à la lecture selon laquelle pour chaque enfant existe-t-il un livre qu'il lit et la seconde correspond à la lecture pour laquelle est-ce-qu'il existe un livre que les enfants lisent.

7.3.3 Forme passive

Nous reprenons la dérivation précédemment présentée pour une phrase au passif.

(31) La présidente est élue par le peuple

La preuve reconnaissant cet énoncé est réalisée à partir du lexique suivant :

Lexique 124.

la	$\lambda P\mu Q.[\|[d\|P(d) \Rightarrow Q(d)]] :$	$(e \to t) \to e$	$\vdash (k \otimes d)/n$
le	$\lambda P\mu Q.[\|[c\|P(c) \Rightarrow Q(c)]] :$	$(e \to t) \to e$	$\vdash (k \otimes d)/n$
peuple	$\lambda z.peuple(z) :$	$e \to t$	$\vdash n$
présidente	$\lambda z.présidente(z) :$	$e \to t$	$\vdash n$
DPprep	$\lambda x\lambda y\lambda P.x :$	$e \to e \to (\iota \to t) \to e$	$\vdash (k\backslash(par\backslash(d_{agent} \otimes ablatif)))/d$
préposition	$\lambda e.e :$	$\iota \to t$	$\vdash par$
élire	$\lambda x\lambda y\lambda e.élire(e, x, y) :$	$e \to e \to \iota \to t$	$\vdash V/d$
modifieur	$\lambda P\lambda x_2\lambda y\lambda e.P(y, e) \wedge$		
	$agent(e, x_2) :$	$(e \to \iota \to t) \to$	
		$e \to e \to \iota \to t$	$(d_{agent}\backslash(ablatif\backslash v_{pass}))/_< V$
aux passif	$\lambda P\lambda y_2\lambda e.P(x) \wedge$		
	$present(e) \wedge patient(e, y_2) :$	$(\iota \to t) \to e \to \iota \to t$	$\vdash (k\backslash t)/v_{pass}$
comp	$\lambda P.P(e) :$	$(\iota \to t) \to t$	$\vdash c/t$

La preuve débute par l'introduction d'une variable libre.

$$\dfrac{\begin{array}{cc}\lambda x\lambda y\lambda e.élire(e, x, y) :\vdash V/d & u : d \vdash d \\ e \to e \to \iota \to t & e\end{array}}{\begin{array}{c}\lambda y\lambda e.élire(e, u, y) : d \vdash V \\ e \to \iota \to t\end{array}}\,[mg]$$

Le modifieur du verbe est, en fait, la *voie* utilisée dans l'énoncé. Cette fois, il introduit le prédicat *agent* pour le *DP* agentif :

$$\frac{\begin{array}{cc}\lambda P\lambda x_2\lambda y\lambda e.P(y,e)\wedge agent(e,x_2): & \\ \vdash (d_{agent}\backslash(ablatif\backslash v_{pass}))/_< V & \lambda y\lambda e.élire(e,u,y): d\vdash V \\ (e\to\iota\to t)\to e\to e\to\iota\to t & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda x_2\lambda y\lambda e.élire(e,u,y)\wedge agent(e,x_2): d\vdash d_{agent}\backslash(ablatif\backslash v_{pass})\\ e\to e\to\iota\to t\end{array}}\,[mg]$$

Puis, la dérivation introduit deux variables libres représentant le *DP* agentif et son cas :

$$\frac{\begin{array}{cc}v: d_{agent}\vdash d_{agent} & \lambda x_2\lambda y\lambda e.élire(e,u,y)\wedge agent(e,x_2): \\ e & d\vdash d_{agent}\backslash(ablatif\backslash v_{pass}) \\ & e\to e\to\iota\to t\end{array}}{\begin{array}{c}\lambda y\lambda e.élire(e,u,y)\wedge agent(e,v): d\vdash ablatif\backslash v_{pass}\\ e\to\iota\to t\end{array}}\,[mg]$$

$$\frac{\begin{array}{cc}w: ablatif\vdash ablatif & \lambda y\lambda e.élire(e,u,y)\wedge agent(e,v): d\vdash ablatif\backslash v_{pass}\\ e & e\to\iota\to t\end{array}}{\begin{array}{c}\lambda e.élire(e,u,w)\wedge agent(e,v): ablatif, d\vdash v_{pass}\\ \iota\to t\end{array}}\,[mg]$$

Parallèlement, le complément d'agent *par le peuple* est dérivé. On commence par construire la *DP* :

$$\frac{\begin{array}{cc}\lambda P.\mu Q.[|[c|P(c)\Rightarrow Q(c)]]:\vdash (k\otimes d)/n & \lambda z.peuple(z):\vdash n\\ (e\to t)\to e & e\to t\end{array}}{\begin{array}{c}\mu Q.[|[c|peuple(c)\Rightarrow Q(c)]]:\vdash k\otimes d\\ e\end{array}}\,[mg]$$

On utilise dans la preuve une entrée particulière qui construit le complément d'agent. Sémantiquement, ce traitement est neutre. Il permet de faire passer la sémantique du *DP* à tout le complément. Elle nécessite l'introduction de deux variables :

$$\frac{\begin{array}{cc}\lambda x\lambda y\lambda P.x:\vdash (k\backslash(par\backslash(d_{agent}\otimes ablatif)))/d & x: d\vdash d\\ e\to e\to\iota\to e & e\end{array}}{\begin{array}{c}\lambda y\lambda P.u: d\vdash k\backslash(par\backslash(d_{agent}\otimes ablatif))\\ e\to\iota\to e\end{array}}\,[mg]$$

$$\frac{\begin{array}{cc}y: k\vdash k & \lambda y\lambda P.u: d\vdash k\backslash(par\backslash(d_{agent}\otimes ablatif))\\ e & e\to\iota\to e\end{array}}{\begin{array}{c}\lambda P.u: k, d\vdash par\backslash(d_{agent}\otimes ablatif)\\ \iota\to e\end{array}}\,[mg]$$

Puis à ces deux variables, on substitue la sémantique du *DP* :

$$\frac{\begin{array}{cc}\mu Q.[|[c|peuple(c)\Rightarrow Q(c)]]:\vdash k\otimes d & \lambda P.u: k, d\vdash par\backslash(d_{agent}\otimes ablatif)\\ e & \iota\to e\end{array}}{\begin{array}{c}\lambda P.\mu Q.[|[c|peuple(c)\Rightarrow Q(c)]]:\vdash par\backslash(d_{agent}\otimes ablatif)\\ \iota\to e\end{array}}\,[mv]$$

Le complément d'agent est finalement construit par l'ajout de la préposition *par* :

$$\frac{\begin{array}{cc}\lambda e.e :\vdash par & \lambda P.\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]] :\vdash par\backslash(d_{agent} \otimes ablatif) \\ \iota & \iota \rightarrow e\end{array}}{\begin{array}{c}\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]] :\vdash d_{agent} \otimes ablatif \\ e\end{array}}\,[mg]$$

La dérivation se poursuit en introduisant la sémantique du DP agent avec la première partie dérivée :

$$\frac{\begin{array}{cc}\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]] : & \lambda e.élire(e,u,w) \wedge agent(e,v) : \\ \vdash d_{agent} \otimes ablatif & ablatif, d \vdash v_{pass} \\ e & \iota \rightarrow t\end{array}}{\begin{array}{c}\lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge agent(e,c) :\vdash v_{pass} \\ \iota \rightarrow t\end{array}}\,[mv]$$

La preuve atteint l'étape IP pour une phrase à la voie passive. Dans ce cas, le prédicat introduit est *patient*.

$$\frac{\begin{array}{cc}\lambda P\lambda y_2\lambda e.P(x) \wedge present(e) \wedge & \lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge \\ patient(e,y_2) :\vdash (k\backslash t)/v_{pass} & agent(e,c) :\vdash v_{pass} \\ (\iota \rightarrow t) \rightarrow e \rightarrow \iota \rightarrow t & \iota \rightarrow t\end{array}}{\begin{array}{c}\lambda y_2\lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge \\ agent(e,c) \wedge present(e) \wedge patient(e,y_2) : d \vdash k\backslash t \\ e \rightarrow \iota \rightarrow t\end{array}}\,[mg]$$

L'étape IP induit l'introduction d'une variable libre dans le prédicat *patient*.

$$\frac{\begin{array}{cc}\multicolumn{2}{c}{\lambda y_2\lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge} \\ y : k \vdash k & agent(e,c) \wedge present(e) \wedge patient(e,y_2) : d \vdash k\backslash t \\ e & e \rightarrow \iota \rightarrow t\end{array}}{\begin{array}{c}\lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge \\ agent(e,c) \wedge present(e) \wedge patient(e,y) : k, d \vdash t \\ \iota \rightarrow t\end{array}}\,[mg]$$

Parallèlement, on construit la sémantique du DP sujet.

$$\frac{\begin{array}{cc}\lambda P\mu R.[|[d|P(d) \Rightarrow R(d)]] :\vdash (k \otimes d)/n & \lambda z.présidente(z) :\vdash n \\ (e \rightarrow t) \rightarrow e & e \rightarrow t\end{array}}{\begin{array}{c}\mu R.[|[d|présidente(d) \Rightarrow R(d)]] :\vdash k \otimes d \\ e\end{array}}\,[mg]$$

Celle-ci est substituée dans la preuve principale. Dans ce cas, la variable entrée en première position dans la structure argumentale du verbe est identifiée avec celle jouant le rôle de *patient* :

$$\frac{\begin{array}{cc}\mu R.[|[d|présidente(d) \Rightarrow & \lambda e.élire(e,u,\mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge \\ R(d)]] :\vdash k \otimes d & agent(e,c) \wedge present(e) \wedge patient(e,y) : k, d \vdash t \\ e & \iota \rightarrow t\end{array}}{\begin{array}{c}\lambda e.élire(e,\mu R.[|[d|présidente(d) \Rightarrow R(d)]], \mu Q.[|[c|peuple(c) \Rightarrow Q(c)]]) \wedge \\ agent(e,c) \wedge present(e) \wedge patient(e,d) :\vdash t \\ \iota \rightarrow t\end{array}}\,[mv]$$

La preuve se termine par l'entrée *comp* qui introduit la variable d'événement.

$$\dfrac{\begin{array}{cc}\lambda P.P(e) :\vdash c/t & \lambda e.\acute{e}lire(e,\mu R.[|[d|pr\acute{e}sidente(d) \Rightarrow R(d)]],\mu Q.[|[c|peuple(c)\wedge \\ (\iota \to t) \to t & Q(c)]]) \wedge agent(e,c) \wedge present(e) \wedge patient(e,d) :\vdash t \\ & \iota \to t\end{array}}{\begin{array}{c}\acute{e}lire(e,\mu R.[|[d|pr\acute{e}sidente(d) \Rightarrow R(d)]],\mu Q.[|[c|peuple(c) \Rightarrow \\ Q(c)]]) \wedge agent(e,c) \wedge present(e) \wedge patient(e,d) :\vdash c \\ t\end{array}}\ [mg]$$

Les $\lambda\mu$-DRSs associées aux entrées gérant la voie de l'énoncé permettent d'associer les rôles thématiques des différentes variables.

La phase d'internalisation des prédicats transforme la formule pour donner :

$$\begin{array}{c}\acute{e}lire(e,\mu R.[|[d|pr\acute{e}sidente(d) \Rightarrow R(d) \wedge patient(e,d)]], \\ \mu Q.[|[c|peuple(c) \Rightarrow Q(c) \wedge agent(e,c)]]) \wedge present(e)\end{array}$$

Les μ-réductions fournissent les deux formules :

1. $[|[d|pr\acute{e}sidente(d) \Rightarrow [|[c|peuple(c) \Rightarrow ((elire(c)c)d) \wedge agent(e,c)]] \wedge patient(e,d)]] \wedge present(e)$
2. $[|[c|peuple(c) \Rightarrow [|[d|pr\acute{e}sidente(d) \Rightarrow ((elire(c)c)d) \wedge patient(e,d)]] \wedge agent(e,c)]] \wedge present(e)$

La première formule correspond à la lecture selon laquelle il y a une présidente que tous les peuples élisent (donc on a unicité de la présidente quel que soit le peuple), et la seconde correspond à la lecture selon laquelle il y a un peuple pour lequel il existe une présidente qu'il élit (donc on a autant de présidente que de peuple).

7.4 Conclusion

Pour la mise en place d'une interface syntaxe-sémantique l'important réside dans la prise en compte simultanée de deux problèmes qui ont l'air antinomiques : d'un côté, la réalisation du liage des variables en structure profonde et de l'autre, la nécessité d'attendre qu'un constituant ait pris sa position de surface pour obtenir sa sémantique complète. Actuellement, peu d'interfaces prennent ce problème en considération, alors qu'il semble être le point d'achoppement pour la mise en place de véritables interfaces syntaxe/sémantique.

Les GMCs sont un système de type logique et cette particularité les rend plus performantes que les GMs. Il est donc possible, grâce à l'isomorphisme de Curry-Howard d'extraire une forme sémantique du calcul syntaxique. Dans ce dernier nous passons en paramètres des termes sémantiques basés sur le $\lambda\mu$-calcul. L'autre particularité des GMCs, très différente, est qu'un constituant n'entre dans la dérivation qu'au moment où toutes ses places - relativement aux différents déplacements qu'il est susceptible de subir - sont présentes dans la dérivation principale et ce, tout en proposant un système élégant de liage des variables en structure profonde.

Comme Kratzer par exemple, on utilise la réification des formules dans le calcul sémantique. En utilisant cette variable et la phase d'internalisation des prédicats dans les $\lambda\mu$-DRSs, SEM est en mesure d'associer un rôle thématique aux différentes variables liées dans la formule. Cependant, l'introduction de la réification génère de nouvelles ambiguïtés qui ne sont pas traitées dans cette proposition. En effet, on ne distingue pas les événements

pour chaque cas induit par les formules : quand *les enfants lisent un livre*, est-ce que tous les enfants doivent lire le même livre au même moment ou doit-il exister un moment où le livre est lu par un enfant ? L'utilisation de la quantification sur les variables d'événements introduites en fin de dérivation est une possibilité pour rendre compte de ces problèmes. Les scénarii de μ-réduction fourniront les ambiguïtés de lectures possibles en fonction du positionnement du quantificateur d'événement.

SEM est un calcul basé sur une perspective chomskyenne pour rendre compte de la sémantique : on utilise une première phase de calcul portée par les dérivations syntaxiques, puis après le *Spell-Out*, SEM devient autonome pour les phases d'internalisation et de réduction. Une autre particularité de SEM par rapport à l'isomorphisme de Curry-Howard, est qu'il n'utilise que des applications. Ceci est une conséquence du choix fait dans le calcul syntaxique qui est uniquement composé d'éliminations de règles logiques.

Nous allons dans le chapitre suivant proposer un fragment d'une GM pour le français et nous verrons comment l'interface syntaxe-sémantique permet de rendre compte de phénomènes sémantiques.

Chapitre 8

Fragments d'une grammaire du français

Sommaire

8.1 Définitions linguistiques 200
8.2 Analyse syntaxique des clitiques 204
8.3 contrepartie sémantique 218
8.4 Autour du groupe verbal - cas simple 222
8.5 Verbes enchâssés 225
8.6 Autour du groupe nominal 232
8.7 Conclusion 234

À présent que nous avons étudié les grammaires minimalistes et présenté un formalisme équivalent qui s'étend en une interface syntaxe sémantique, nous souhaitons développer des fragments d'une grammaire du français plus élaborés que celui que nous avons introduit au fur et à mesure de ce manuscrit. Nous avons montré qu'il était possible de traiter les phrases simples qui respectent l'ordre Sujet-Verbe-Objet ainsi que les questions avec inversion du verbe et du sujet et montée du constituant question. Afin de garder une cohérence dans ce fragment de grammaire, nous avons travaillé autour de la modélisation des clitiques. Les études de ces constituants syntaxiques ont été au coeur des travaux de P. Miller, [Mil92], [MM03] qui nous ont permis de mieux appréhender leurs utilisations dans le cadre de la grammaire française, mais par leur utilisation dans les langues romanes. Afin de conserver une cohérence entre ces fragments, nous nous sommes référés aux travaux de C. Muller, notamment [Mul02]. Enfin, le lien entre syntaxe et sémantique pour les clitiques découle de [MR05].

Les clitiques sont très utilisés dans les langues romanes pour leur capacité de reprise anaphorique et en particulier en français où ils montent au-dessus du groupe verbal. Cependant, leur utilisation relève de bien des raffinements et font de ces éléments des charnières dans la construction d'énoncés élaborés. Nous avons donc décidé de poser comme base leur modélisation, ce qui nous a permis d'aborder les problèmes posés par d'autres phénomènes syntaxiques. Nous proposons une modélisation simple et homogène dont le résultat est un fragment d'une grammaire du français.

Une première version du traitement des clitiques du français a été présentée par Ed

Stabler [Sta01] comme illustration de l'utilisation de la fusion de tête - *Head-Movement*. Nous proposons ici une extension de cette formalisation à un plus grand nombre de clitiques en français. Les GMCs étant un formalisme lexicalisé, nous présenterons systématiquement les nouvelles entrées lexicales nécessaires, et expliciterons des analyses générales de leur utilisation.

Dans un premier temps, nous présenterons comment définir et utiliser cette classe d'éléments particuliers, puis comment les intégrer dans les GMCs. Cette étude nous conduit à augmenter la couverture de la grammaire. Nous verrons comment intégrer les clitiques avec la négation puis la construction de groupes verbaux complexes (verbes à supports, verbes à contrôles, etc.) et enfin leur intégration avec les groupes nominaux. Nous exposerons les implications sémantiques des clitiques et comment l'interface syntaxe-sémantique présentée dans le chapitre précédent peut en rendre compte. Nous examinerons plusieurs circuits représentant les lexiques exposés dans ce chapitre.

8.1 Définitions linguistiques

Pour inclure les clitiques dans la grammaire, nous commençons par définir ce que sont ces objets syntaxiques, et quel est leur rôle sémantique dans l'élaboration de discours ou de textes.

8.1.1 Définition et propriétés des clitiques

Il existe une ambivalence forte dans le terme même de *clitique*. On peut l'utiliser, soit en prosodie pour définir la classe des mots atones, soit en syntaxe. On appelle *atone* les mots ne pouvant pas porter de marque tonique, en général, en français, le dernier mot de la phrase.

Nous devons aussi donner la différence entre pronoms disjoint et conjoint (ou fort/faible).

Définition 125. *Parmi les pronoms, on fait la distinction entre pronom fort (disjoint) et pronom faible (conjoint). Si l'on peut séparer le pronom du verbe il est disjoint. Il jouit d'une certaine mobilité car il possède un comportement syntaxique analogue à celui du groupe nominal séparé du verbe (par une pause, une préposition etc.). Les pronoms forts sont :*

$$\{moi,\ toi,\ lui,\ elle,\ cela/ça,\ nous,\ vous,\ eux,\ elles\}$$

(32) (a) Jean parle encore d'elle.
(b) Jean partira avec toi.
(c) Toi seul pourra nous sauver.
(d) Jean se promène avec lui.

Si un pronom n'est pas disjoint, il est conjoint et est appelé *clitique*. L'observation de l'utilisation des pronoms forts est soumise à une hypothèse difficilement représentable, mais plus importante pour la génération que pour l'analyse : les pronoms forts ne s'utilisent que si l'on ne peut pas produire un énoncé équivalent en utilisant une phrase avec un clitique.

Les clitiques sont atones. La seule façon de marquer une inflexion est d'utiliser une reprise pronominale :

(33) *Jean TE le rend.
Jean le rend à toi.

Cette hypothèse est en particulier nécessaire pour la focalisation des clitiques, ou dans la résolution de conflits qui apparaissent sur les restrictions d'ordre entre les clitiques. Nous étudierons ces derniers cas dans la partie suivante.

Kayne (1984) suppose que les pronoms objets sont des clitiques syntaxiques, et les pronoms sujets sont des clitiques phonologiques.

Nous utiliserons la définition suivante des clitiques du point de vue de la syntaxe.

Définition 126. *On appelle clitiques les* pronoms personnels conjoints - *par opposition aux pronoms disjoints* - prosodiquement atones. *Syntaxiquement, ils sont* antéposés *au verbe dont ils dépendent, formant avec ce dernier une séquence syntaxique et prosodique soudée.*

À partir de cette définition, nous pouvons donner des propriétés importantes régissant le comportement syntaxique de ces éléments. En français, les pronoms clitiques :

1. se placent devant le verbe,
2. forment une séquence homogène,
3. prennent leur position indépendamment de leur fonction syntaxique,
4. possèdent leurs propres règles.

Les règles déterminant leur position dans la phrase leur assurent de former un groupe uni autour du verbe. Ainsi seul un autre clitique peut s'intercaler entre un clitique et le verbe dont il dépend.

(34) Jean la (*habituellement) donne à Marie.
Jean la donne habituellement à Marie.

En règle générale, les clitiques se placent devant le verbe. Cependant, il existe certains cas où les clitiques ne peuvent être antéposés, et doivent se postposer, en particulier pour l'impératif. Dans ce cas, c'est le clitique qui porte la marque de tonicité.

(35) Donne-le lui !
Donne m'en.

L'accent sur le pronom n'est ici qu'un effet de la position syntaxique.

Les pronoms clitiques faibles occupent une position dérivée. Il se placent donc sur un verbe ayant reçu son inflexion.

Les clitiques ne sont pas coordonnables :

(36) *Jean te et le conduira.
*Jean conduira toi et elle.
Jean vous conduira.

8.1.2 Typologie des pronoms clitiques faibles

Description casuelle

Pronoms clitiques sujet (cas nominatif) : $\{je, tu, il, elle, on, ce, nous, vous, ils, elles\}$

(37) Elle vend son corps pour quatre sous.

Pronoms clitiques objets directs (cas accusatif) : $\{le, la, les, me, te, se, nous, vous\}$

(38) Jean la répare.

Pronoms clitiques objet indirect (cas datif) : $\{lui, leur, me, te, se, nous, vous\}$

(39) Jean lui donne un vélo.

Pronom clitique génitif : $\{en\}$

(40) Marie en donne.

Pronom clitique oblique : $\{y\}$

(41) Marie y partira pour les vacances.

Les séquences de pronoms clitiques

L'ordre des clitiques en français est fortement contraint. Nous utilisons la description des clitiques faite par Perlmutter [Per71] qui définit un filtre permettant de déterminer l'ordre correct pour les séquences de clitiques dans les langues romanes, où chaque position peut accueillir un élément.

$$[\{je/tu/\cdots\}|ne|\{me/te/se/\cdots\}|\{le/la/les/\cdots\}|\{lui/leur\}|y|en].$$
[nominatif | négatif | réflexif | accusatif | datif | locatif | génitif],

Voici une série d'exemples représentant diverses séquences acceptantes de clitiques :

(42) (a) [Il ne me le] donne pas.
(b) [Vous ne les leur] rendez pas.
(c) [Je vous en] ramènerai un.
(d) [Nous y en] mettrons une.

Cependant, toutes les séquences ne sont pas réalisables. Il existe certaines restrictions de cooccurrences.

1. Deux clitiques de première ou seconde personne ou réfléchis ne peuvent pas apparaître ensemble.
2. Un clitique de première ou seconde personne ou réfléchi précède toujours un clitique de troisième personne.
3. Un clitique de première ou seconde personne ou réfléchi ne peut pas apparaître avec un datif.
4. Deux clitiques de troisième personne apparaissent dans l'ordre ACC-DAT.

En général, pour résoudre les conflits conséquents de la première restriction, on peut utiliser un pronom fort.

On peut donc raffiner le filtre par :

$$\{1^{ère}|2^{ème}|Refl|Dat\}$$

Nous conserverons l'idée que deux clitiques réflexif et datif ne peuvent pas apparaître dans la même phrase.

Ces descriptions ne prennent pas en compte les pronoms inhérents et éthiques :

(43) Jean en a bavé.
Je vais te lui casser la figure.

Nous ne les inclurons pas dans nos descriptions, cependant, il est possible d'analyser des énoncés les contenant, pour des cas simples.

8.1.3 Approches syntaxiques

Il existe au moins deux façons d'aborder la question de l'analyse syntaxique des pronoms clitiques :

1. l'approche en génération basique : "Les pronoms clitiques sont générés en structure profonde directement sur le verbe". Cependant cette vision du traitement des clitiques se heurte au problème des dépendances à longue distance. Les clitiques montent au-dessus, non pas du verbe, mais du groupe verbal, donc de l'auxiliaire :

   (44) (a) Ce livre lui a été donné.
   (b) * Ce livre a lui été donné.

   Et ils peuvent aussi être des reprises d'arguments nominaux - montée longue des clitiques :

   (45) Marie en a lu le premier chapitre.

2. l'approche en mouvement : "Les pronoms clitiques sont insérés en structure profonde dans leur position d'arguments, puis sont déplacés sur le verbe en structure de surface" Kaynes [Kay75] :

   (46) Structure profonde

X	*V*	*Y*	*NP+[pro]*	*Z*
1	2	3	4	5
Jean	voit	souvent	[pro]	là-bas

   Structure de surface

1	*4+2*	*3*	*5*
Jean	le_voit	souvent	là-bas

Nous utilisons la description de l'analyse syntaxique des clitiques proposée par Sportiche [Spo92], [Spo98] pour lequel les clitiques sont des marqueurs phonologiques coréférents d'arguments non marqués phonologiquement.

(47) Structure phonologique :

Jean	*la*	*répare*
Jean	la_i	répare	ϵ_i

Structure syntaxique :

Jean	*la*	*répare*
Jean	ϵ_i	répare	DP_{acc}

Cette approche permet de lier en structure profonde tous les arguments du verbe et de retrouver en structure de surface les arguments dans l'ordre standard. Le liage des clitiques avec leur position argumentale est l'enjeu de cette description.

8.1.4 Interprétation sémantique

Les clitiques sont des reprises anaphoriques, ce que nous venons d'évoquer. Ils ont un rôle et une fonction cruciaux dans le système énonciatif et dans la cohérence du discours. Les clitiques sont scindés en deux catégories sémantico-référentielles.

Définition 127. *Les* pronoms déictiques *: "me, te, je, nous, vous, on" n'ont pas d'antécédent. Le référent est identifié à partir de la situation de discours dans laquelle ils sont employés.*

Les pronoms de troisième personne : "il, elle, le, la, les, lui, leur" ne sont pas des protagonistes de l'énonciation mais des anaphores dont le référent est identifié à partir d'éléments du contexte.

Une expression est anaphorique si son interprétation dépend d'une autre expression qui figure dans le texte. Les pronoms clitiques servent donc à conserver la structure et la cohérence textuelle. Nous verrons dans la formalisation de l'analyse syntaxique et l'utilisation de l'interface syntaxe-sémantique comment ces références sont marquées, laissant la question de leur résolution ouverte à la sémantique.

8.2 Analyse syntaxique des clitiques

Dans la plupart des formalismes de TAL, les clitiques ont été traités par l'énumération des séquences possibles - LFG, TAG. Nous n'utilisons pas cette technique et nous supposons que ce sont des arguments du verbe qui *doivent* être cliticisés. Pour cela, le verbe prend ses arguments et une phase particulière de l'analyse le cliticise. Ainsi, la cliticisation est non-déterministe et homogène pour tous les cas.

Avant d'aller plus avant dans la modélisation des clitiques, nous introduisons une nouvelle notation qui permet de simplifier la lecture des preuves. On note $d_{[u]}$ une hypothèse d dépendante de la variable de chaîne u ($(\epsilon, u, \epsilon) : d$).

8.2.1 Version de Ed Stabler

Dans [Sta99], Stabler propose une version simple d'un lexique permettant de traiter une partie des clitiques en français. Ces travaux sont inspirés de l'analyse des clitiques proposée par D. Sportiche [Spo92] selon lequel :

- les clitiques ne sont pas des éléments déplacés de la position XP* mais sont coréférents de cette position (la position XP* correspond à la projection maximale du verbe dans la structure en constituant),
- les clitiques apparaissant dans l'analyse portent tous les traits de la position dont ils sont coréférents.

La cliticisation se décompose ainsi en une position vide prenant la position de l'argument du verbe et un second item coréfère à cet argument.

Sportiche suppose que les clitiques ne forment pas des objets syntaxiquement homogènes, mais se construisent à partir d'une unité hôte - c'est-à-dire l'objet avec lequel ils sont coréférents, ce qui nécessite un partage des traits entre ces deux éléments.

Stabler propose de dissocier deux parties dans la cliticisation : la première est un élément phonologiquement vide qui prend position dans la structure argumentale du verbe et la seconde est le contrepoint de cet élément. Elle est donc phonologiquement marquée - et atone en prosodie, coréférent de cet élément et prenant les traits caractéristiques de ce dernier. La partie phonologique du clitique apparaît dans la structure de surface.

Pour rendre compte de ce traitement, nous ajoutons dans le lexique une entrée phonologiquement vide reprenant les traits standards des DPs et un trait particulier supplémentaire

qui devra être annulé par le trait équivalent de l'élément portant la marque phonologique du clitique. Ces deux éléments dans la dérivation sont *reliés* par un déplacement, qui unifie les deux traits et permet de marquer le liage de ces deux positions. Grâce à cette nécessité d'unifier les traits, si l'un des deux n'apparaît pas dans la dérivation, elle échouera.

Nous résumons ce traitement par la suite des étapes d'une dérivation comme suit (les annotations entre crochets reprennent les hypothèses ; si elles sont marquées d'une barre au-dessus elles sont manquantes pour un élément $[\overline{d}]$, et les annotations sans crochet pour les ϵ reprennent le mot dont ϵ est la trace) :

(48) donne $\epsilon_{[\overline{F}]}$
Jean$_{[\overline{k}]}$ la$_{[F]}$ donne $\epsilon_{[\overline{F}]}$
Jean$_{[\overline{k}]}$ ϵ_{la} la donne
Jean ϵ_{la} la donne

la est la partie phonologique du clitique et ϵ_{la} la position argumentale coréférente du clitique. Ce traitement n'exclut pas de pouvoir cliticiser plusieurs arguments du verbe.

> Le mécanisme utilisé pour reconnaître les clitiques est le suivant : un verbe prend un argument phonologiquement vide mais portant un trait spécifique marquant cet élément. Lors de la résolution des clitiques on s'assure de la présence simultanée d'un élément ayant la marque phonologique du clitique et de l'élément introduit. Ceci est réalisé grâce à un déplacement de l'élément à phonologie vide.

Pour respecter l'ordre des clitiques, on différencie les *états* du groupe verbal. Par exemple, on appelle *Acc*3 un groupe verbal ayant reçu un clitique accusatif de troisième personne et V un verbe lexicalisé. Les entrées lexicales portant la partie phonologique des clitiques font passer le groupe verbal d'un état dans un autre en fonction du type de clitique. Le lexique restreint l'ordre des transitions possibles. Cette façon d'encoder les clitiques multiplie le nombre d'entrées lexicales ayant la même forme phonologique mais assure la bonne formation de la séquence de clitiques.

Ainsi, on pourra construire à partir de *lui* en sélectionnant un V une tête $Dat3$ - [lui V] - qui pourra être choisie par *le : : Acc* pour former [le lui V].

Pour obtenir ces dérivations, il est nécessaire d'utiliser des déplacements de tête. En effet, beaucoup de fusions ont lieu autour du verbe pour modifier sa composition en traits, mais à chaque fois, ce dernier est remonté jusqu'à la nouvelle entrée pour conserver le contrôle de la structure. Le verbe n'apparaît plus en bas de la dérivation mais remonte en fonction des entrées utilisées.

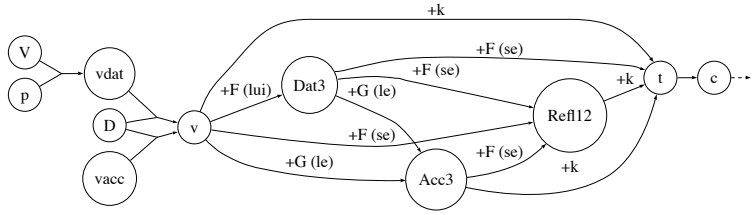


Fig. 43 – Première version du traitement des clitiques.

Voici le lexique proposé pour reconnaître les clitiques accusatifs, datifs et réflexifs avec les GMs :

ϵ : : =T C			
ϵ : :=Refl12 +k T	ϵ : :=Acc3 +k T	ϵ : :=Dat3 +k T	ϵ : : =v +k T
se : :=Acc3 +F Refl12	se : :=Dat3 +F Refl12	se : : =v +F Refl12	
le : :=Dat3 +G Acc3	le : :=v +G Acc3		
lui : :=v +F Dat3			
ϵ : :vacc<= =D v	ϵ : : vdat<= =D +k vacc	ϵ : :V<= =p vdat	
montrera : :V	repare : :V		
ϵ : : P<= p	a : :=D +k P	ϵ : : p -F	
Jean : :D -k	Marie : :D -k		
le : : =N D -k	ϵ : : D -k -F	ϵ : :D -k -G	
roi : :N	livre : :N		
ϵ =>V =D +k =D v			

Dans cette présentation, le lexique proposé a pour vocation de modéliser uniquement le traitement de la cliticisation. Les items verbaux portent directement l'ensemble de leurs comportements syntaxiques. Nous adapterons cette proposition à ce que nous avons proposé dans les chapitres précédents.

On représente sous forme de circuit les expressions dérivables à partir du lexique - la figure 43 présente la partie du lexique pour l'analyse de la forme verbale avec clitique(s), chaque transition correspondant à un type d'entrée lexicale.

Comme nous l'avons montré, nous pouvons transformer ce lexique pour proposer une GMC équivalente. À partir de ce dernier, nous allons donner le détail d'une dérivation pour l'analyse syntaxique des clitiques en français.

Dérivation 128.

(49) Jean la donne.

Sélection du lexique :

Jean	$k \otimes d$	ϵ	$C \,/\, T$	ϵ	$(G \otimes (k \otimes d))$
donne	V	ϵ	$d \setminus k \setminus d \setminus v/_{>}V$		
ϵ	$k \setminus T/Acc3$	*la*	$G \setminus Acc3/v$		

1. *sélection de l'entrée lexicale :* $(\epsilon, donne, \epsilon) : V$
   sélection de l'entrée lexicale : $\vdash (\epsilon, \epsilon, \epsilon) : d \setminus k \setminus d \setminus v/_{>}V$. *Cette entrée permet d'apporter au verbe son comportement syntaxique.*
   Head-Movement entre les deux éléments précédents où la partie phonologique du verbe remonte vers la nouvelle tête de la dérivation.
   $$\frac{\vdash (\epsilon, \epsilon, \epsilon) : d \setminus k \setminus d \setminus v/_{>}V \quad \vdash (\epsilon, donne, \epsilon) : V}{\vdash (\epsilon, donne, \epsilon) : d \setminus k \setminus d \setminus v}\,[mg]$$
2. *fusion avec une hypothèse :* $(\epsilon, u, \epsilon) : d \vdash (\epsilon, u, \epsilon) : d$
   (c'est la position argumentale du verbe à phonologie vide).
   $$\frac{u : d \vdash (\epsilon, u, \epsilon) : d \quad \vdash (\epsilon, donne, \epsilon) : d \setminus k \setminus d \setminus v}{u : d \vdash (u, donne, \epsilon) : k \setminus d \setminus v}\,[mg]$$
3. *fusion avec une hypothèse :* $v : k \vdash (\epsilon, v, \epsilon) : k$. *La présence des deux hypothèses déclenche un déplacement. Dans ce cas, le déplacement est cyclique. La preuve lie les deux hypothèses :*

$$\frac{v : k \vdash (\epsilon, v, \epsilon) : k \quad u : d \vdash (u, donne, \epsilon) : k \setminus d \setminus v}{v : k, u : d \vdash (vu, donne, \epsilon) : d \setminus v}\,[mg]$$

$$\frac{w : k \otimes d \vdash (\epsilon, w, \epsilon) : k \otimes d \quad v : k, u : d \vdash (vu, donne, \epsilon) : d \setminus v}{w : k \otimes d \vdash (w, donne, \epsilon) : d \setminus v}\,[mv]$$

4. *à nouveau la structure verbale nécessite une hypothèse de type d qui correspond à la position de la catégorie de base du groupe déplacé :*

$$\frac{x : d \vdash (\epsilon, x, \epsilon) : d \quad w : k \otimes d \vdash (w, donne, \epsilon) : d \setminus v}{x : d, w : k \otimes d \vdash (x \bullet w, donne, \epsilon) : v}\,[mg]$$

5. *la dérivation passe alors dans la phase de cliticisation : sélection de l'entrée* $(\epsilon, la, \epsilon) : G \setminus Acc_3 \,/\, v$ *et fusion :*

$$\frac{\vdash (\epsilon, la, \epsilon) : G \setminus Acc_3 \,/\, v \quad x : d, w : k \otimes d \vdash (x \bullet w, donne, \epsilon) : v}{x : d, w : k \otimes d \vdash (\epsilon, la, x \bullet w\ donne) : G \setminus Acc_3}\,[mg]$$

6. *déplacement :*
   introduction d'une hypothèse représentant le trait de vérification de la cliticisation G *:*
   $$\frac{y : G \vdash (\epsilon, y, \epsilon) : G \qquad x : d, w : k \otimes d \vdash (\epsilon, la, x \bullet w\ donne) : G \setminus Acc_3}{y : G, x : d, w : k \otimes d \vdash (y, la, x \bullet w\ donne) : Acc_3}\,[mg]$$
   et substitution :
   $$\frac{\vdash (\epsilon, \epsilon, \epsilon) : (G \otimes (d \otimes k)) \qquad y : G, x : d, w : k \otimes d \vdash (y, la, x \bullet w\ donne) : Acc_3}{x : d \vdash (\epsilon, la, x\ donne) : Acc_3}\,[mv]$$

7. *sélection de l'entrée lexicale permettant de sortir de la cliticisation :* $(\epsilon, \epsilon, \epsilon) : k \setminus T \,/\, Acc_3$ *et fusion :*

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : k \setminus T \,/\, Acc_3 \quad x : d \vdash (\epsilon, la, x\ donne) : Acc_3}{x : d \vdash (\epsilon, \epsilon, la\ x\ donne) : k \setminus T}\,[mg]$$

8. *déplacement :*
   introduction d'une hypothèse : $z : k \vdash (\epsilon, z, \epsilon) : k$

$$\frac{z : k \vdash (\epsilon, z, \epsilon) : k \quad x : d \vdash (\epsilon, \epsilon, la\ x\ donne) : k \setminus T}{z : k, x : d \vdash (z, \epsilon, la\ x\ donne) : T}\,[mg]$$

   et substitution :

$$\frac{\vdash (\epsilon, Jean, \epsilon) : k \otimes d \quad z : k, x : d \vdash (z, \epsilon, la\ x\ donne) : T}{\vdash (Jean, \epsilon, la\ donne) : T}\,[mv]$$

9. *sélection de l'entrée lexicale :* $\vdash (\epsilon, \epsilon, \epsilon) : C \setminus T$ *permettant de conclure la preuve et d'accepter la chaîne : "Jean la donne".*

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : C \setminus T \quad \vdash (Jean, \epsilon, la\ donne) : T}{\vdash (Jean, \epsilon, la\ donne) : T}\,[mg]$$

Dans sa présentation, Stabler propose un lexique permettant de reconnaître les clitiques accusatifs, datifs et réflexifs. Il assure que l'analyse les reconnaît dans le bon ordre en utilisant un *type* verbal qui évolue en fonction de l'état de la dérivation : l'analyse est conduite par la catégorie de la tête et le clitique suivant qui peut être introduit assignera une nouvelle catégorie verbale en fonction de l'ordre du filtre de Perlmutter.

On remarquera que pour empêcher la présence d'un réflexif et d'un datif dans un énoncé, le lexique utilise la même hypothèse F. En effet, la présence simultanée de plusieurs mêmes hypothèses violerait la SMC. Ces analyses ne seront donc pas acceptées.

À partir de cette première étude du traitement des clitiques en français, nous proposons une extension à l'ensemble des clitiques en français s'intégrant dans les lexiques proposés précédemment. Cette partie de l'analyse des clitiques forme une séquence dans l'analyse générale que nous appellerons **cluster de cliticisation**.

8.2.2 Extension : clitiques génitif, oblique et nominatif

Dans cette première approche, il ressort de la typologie des clitiques qu'il est nécessaire d'étendre ce modèle à l'ensemble des clitiques. Il manque notamment les clitiques nominatifs, génitif (*en*) et oblique (*y*) ainsi que la négation. Nous reviendrons sur le traitement de la négation dans la section suivante. D'autre part, les clitiques nominatifs ont une propriété particulières : ils sont coordonnables. Bien que nous ne présenterons pas de traitement de la coordination qui est un phénomène complexe en syntaxe, nous marquerons une rupture dans le cluster de cliticisation. Le traitement des clitiques est homogène jusqu'à la position de la négation. Le statut de clitique du marqueur de négation ainsi que des pronoms personnels est sujet à discussion. Cependant, son traitement sera pour nous identique.

Afin de modéliser clairement le cluster de cliticisation, nous différencions un état particulier *clitique* permettant de passer dans cette phase, et un second état où elle est terminée. Ceci permet de factoriser une partie des entrées lexicales nécessaires en dissociant l'entrée dans le cluster de son fonctionnement interne. Ce cluster se termine avant le traitement de la négation. Nous les appellerons ici *clitique* et *finclitique*. Le fait d'être dans un état signifie que ce cas est présent/traîté dans l'analyse. Nous devons maintenant ajouter la possibilité d'avoir ces clitiques tout en respectant l'ordre du filtre de Perlmutter.

Génitif et oblique

Il est nécessaire d'introduire de nouvelles entrées lexicales pour les clitiques se positionnant à la gauche des clitiques traités dans la version de Stabler.

Le traitement des clitiques suit l'ordre inverse du filtre de Perlmutter, le premier clitique que nous devons ajouter pour conserver le bon ordre est le clitique *génitif*. La première entrée ajoutée est l'argument phonologiquement vide portant le trait relatif au clitique génitif :

$$(\epsilon, \epsilon, \epsilon) : (EN \otimes (k \otimes d)$$

Pour la seconde phase de la cliticisation, nous ajoutons un nouvel état *génitif* accessible depuis l'origine du cluster, soit la catégorie *clitique*. Le passage dans l'état génitif se fait par une entrée lexicale dont la partie phonologique est *en* et qui nécessite une hypothèse *en* qui sera utilisée par l'entrée précédente, soit :

$$(\epsilon, en, \epsilon) : EN \setminus genitif /_< clitique$$

On passe ensuite à toutes les autres formes de clitiques et à l'état *finclitique*. On peut donc ajouter un oblique et passer à tous les autres clitiques. La transition vers l'état *oblique* se fait par une entrée lexicale dont la partie phonologique est y. Toutes les autres transitions sont réalisées par une entrée portant une forme phonologique concernant l'état d'arrivée, sauf celle permettant de sortir de la cliticisation qui, elle, est vide.

génitif vers réflexif	:	(ϵ, me, ϵ) :	$F \setminus refl /_< genitif$
génitif vers accusatif	:	(ϵ, le, ϵ) :	$G \setminus acc /_< genitif$
		$(\epsilon, les, \epsilon)$:	$G \setminus acc /_< genitif$
génitif vers datif	:	$(\epsilon, lui, \epsilon)$:	$F \setminus dat /_< genitif$
		$(\epsilon, leur, \epsilon)$:	$F \setminus dat /_< genitif$
génitif vers oblique	:	(ϵ, y, ϵ) :	$Y \setminus oblique /_< genitif$
sortie simple du génitif	:	$(\epsilon, \epsilon, \epsilon)$:	$finclitique /_< genitif$

Le cas oblique se traîte de la même manière, à ceci près que depuis *oblique* on ne peut pas revenir vers l'état *génitif*.

L'ajout de l'item lexical prend la position argumentale du verbe, ce qui impose la présence d'une hypothèse :

$$(\epsilon, \epsilon, \epsilon) : (Y \otimes (k \otimes d))$$

Entrée dans l'oblique :

$$(\epsilon, y, \epsilon) : Y \setminus oblique /_< clitic$$

$$(\epsilon, y, \epsilon) : Y \setminus oblique /_< genitif$$

Chacune de ces entrées introduira une hypothèse Y, qui utilisera la position argumentale. Les transitions permettant de passer à cet état sont phonologiquement marquées par y, et celles pour en sortir sont, soit vides, soit portant la marque phonologique du clitique suivant.

oblique vers réflexif	:	(ϵ, se, ϵ)	:	$F \setminus refl /_< oblique$
		(ϵ, me, ϵ)	:	$F \setminus refl /_< oblique$
oblique vers accusatif	:	(ϵ, se, ϵ)	:	$G \setminus acc /_< oblique$
		$(\epsilon, les, \epsilon)$	:	$G \setminus acc /_< oblique$
oblique vers datif	:	$(\epsilon, lui, \epsilon)$	:	$F \setminus dat /_< oblique$
		$(\epsilon, leur, \epsilon)$	:	$F \setminus dat /_< oblique$
sortie simple de l'oblique	:	$(\epsilon, \epsilon, \epsilon)$	:	$finclitique /_< oblique$

Nous avons donc ajouté le traitement de deux nouveaux cas de clitiques. La possibilité de les introduire dans l'analyse dépendra uniquement du type verbal et des éléments présents dans la structure argumentale du verbe.

Le cluster est présenté par le circuit de la figure 44.

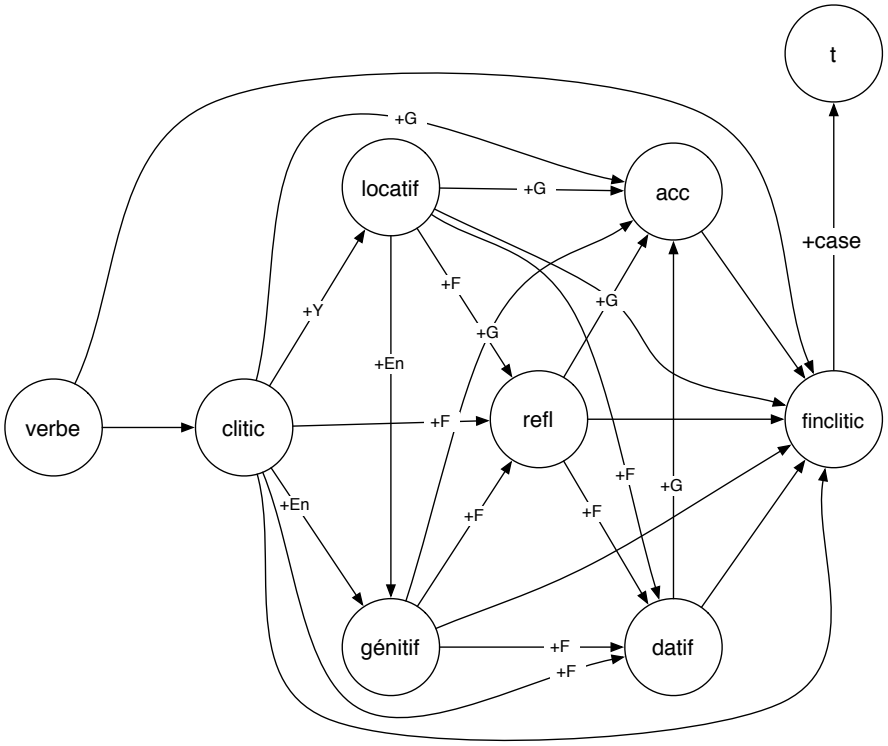


FIG. 44 – Cluster de cliticisation.

Nominatif

Le traitement des clitiques nominatifs se distingue des autres traitements par la possibilité de coordonner des groupes verbaux avec eux :

(50) Il part en vacances et y travaillera.

Ces clitiques se distinguent aussi par l'attribution de cas pour le nominatif qui se produit lors de la dernière étape de la dérivation, avant de procéder à l'introduction du type *complementizer*. Les entrées nécessaires à ce traitement sont modifiées par rapport à celles que nous avons exposées précédemment. De plus, il se situe à l'extérieur du *cluster*.

Ces particularités permettent de poser la question du statut de clitique des nominatifs et de la même façon pour le clitique marqueur de négation. Cependant, nous les considérons tous les deux comme des objets se traitant de manière identique, ce qui est aussi le cas dans un grand nombre de théories linguistiques. Par exemple, chez Miller [Mil92], les pronoms nominatifs sont en français des clitiques car ils ne portent pas de marque tonique et fonctionnent comme des reprises anaphoriques de l'argument sujet vide du verbe.

Nous proposons donc un traitement de la reconnaissance des clitiques nominatifs à l'extérieur du cluster. De fait, toutes les entrées relatives à leur forme phonologique se baseront sur l'état du verbe *finclitique* et rendront un nouvel état *Nom* (pour nominatif). Nous ajoutons donc un argument du verbe phonologiquement vide, qui sera incluse avant

le traitement des clitiques mais qui n'aura pas encore reçu son cas, il y a donc inversion de ses traits : $(\epsilon, \epsilon, \epsilon) : (k \otimes (subj \otimes d))$.

Le mécanisme utilisé garantit que seul le sujet n'a pas encore été traité à ce moment de l'analyse, il n'y a donc pas d'ambiguïté sur le cas qu'il recevra par la suite.

Le synopsis de l'analyse sera le suivant :

– la donne $\epsilon_{[\overline{Nom}]}$ ϵ
– Je$_{[Subj]}$ la donne $\epsilon_{[\overline{Nom}]}$
– ϵ_{Je} Je la donne

Nous ajoutons enfin dans le lexique un trait *Nom* et les entrées lexicales pour les pronoms :

$$(\epsilon, je, \epsilon) : Suj \setminus Nom \,/\, finclitique \qquad (\epsilon, nous, \epsilon) : Suj \setminus Nom \,/\, finclitique$$

$$(\epsilon, tu, \epsilon) : Suj \setminus Nom \,/\, finclitique \qquad (\epsilon, vous, \epsilon) : Suj \setminus Nom \,/\, finclitique$$

$$(\epsilon, il, \epsilon) : Suj \setminus Nom \,/\, finclitique \qquad (\epsilon, ils, \epsilon) : Suj \setminus Nom \,/\, finclitique$$

$$(\epsilon, elle, \epsilon) : Suj \setminus Nom \,/\, finclitique \qquad (\epsilon, elles, \epsilon) : Suj \setminus Nom \,/\, finclitique$$

$$(\epsilon, on, \epsilon) : Suj \setminus Nom \,/\, finclitique$$

Une dernière entrée permet de passer vers la fin de la dérivation à partir de l'état *Nom* :

$$(\epsilon, \epsilon, \epsilon) : k \setminus t \,/\, Nom$$

On a ainsi ajouté le traitement de toutes les formes de clitiques en français. Nous résumons ces traitements par le circuit de la figure 45

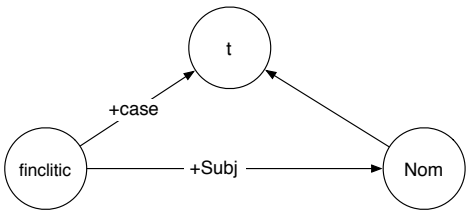


Fig. 45 – Traitement des clitiques nominatifs.

8.2.3 Négation

Nous allons maintenant proposer un traitement pour le marqueur de négation *ne* en nous inspirant du traitement standard de la cliticisation. Remarquons que la négation présente des cas particuliers (placement à l'impératif, sur un verbe à l'infinitif) sur lesquels nous reviendrons dans la section suivante de ce chapitre.

Négation standard

Commençons par ajouter le traitement de la négation simple : *ne ... pas*.

La première partie de la négation est considérée comme un clitique. Or, la séquence de clitiques doit être homogène en cas de pronom personnel sujet. D'autre part, ce marqueur se place juste avant le verbe, et c'est justement ce que nous venons de traiter.

Les analyses des GMs sont basées sur le verbe et nous avons vu que nous traitions les différents constituants dans l'ordre inverse d'apparition. Ainsi, la première partie de la négation rencontrée est donc celle du forclusif (concluant la négation), *pas* ou *jamais* dans les exemples suivants :

(51) Jean ne la donne pas.
Jean ne la porte jamais.

Seul le *ne* est considéré comme un clitique. Cependant, nous proposons un traitement pour lequel le forclusif intervient dans la dérivation en même temps que l'inflexion. Pour cela on suppose que l'inflexion est complexe. Cet ajout sur l'inflexion implique que lors de son introduction dans la dérivation, il ne sera plus lexical et donc se placera à droite de l'analyse qu'il modifie. Cependant, pour éviter les mauvaises formations phonologiques, l'inflexion se concatènera avec le forclusif - via un *Head-Movement* - et prendra un argument à phonologie vide qui est la première partie de la cliticisation du *ne*. Ce clitique n'est pas utilisé comme reprise pronominale, mais comme marqueur intentionnel fonctionnant comme un argument particulier d'un foncteur sur le verbe.

Nous ajoutons les entrées suivantes pour la négation :

$$(\epsilon, inflexion, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe /_{<} neg_2$$

$$(\epsilon, pas, \epsilon) : neg_2$$

$$(\epsilon, \epsilon_{ne}, \epsilon) : Ne \otimes neg_1$$

Elles permettent de construire l'inflexion complexe :

$$\frac{\vdash (\epsilon, inflex, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe /_{<} neg_2 \quad \vdash (\epsilon, pas, \epsilon) : neg_2}{\vdash (\epsilon, inflex\ pas, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe}\ [mg]$$

$$\frac{u : neg_1 \vdash (\epsilon, u, \epsilon) : neg_1 \quad \vdash (\epsilon, inflex\ pas, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe}{u : neg_1 \vdash (u, inflex\ pas, \epsilon) : little\_v_{>} \setminus verbe}\ [mg]$$

Le forclusif vient donc se placer après le verbe. Le traitement du marqueur se déroulera après le cluster de cliticisation et avant le traitement des pronoms sujets. À ce moment de l'analyse, il nous faut introduire la seconde partie de la cliticisation du *ne* qui complétera la preuve pour y introduire une hypothèse Ne relative à ce marqueur. Une fois cette négation introduite, la dérivation continue en cliticisant ou non le sujet. On retrouve le même état qu'en *finclitique*. On ajoute l'entrée :

$$(\epsilon, ne, \epsilon,) : Ne \setminus finclitique_{>} / finclitique$$

Le nombre d'introductions d'hypothèses Ne n'est pas contraint par la dérivation mais par le nombre d'hypothèses neg_1 se combinant avec lui. Or ces hypothèses ne peuvent être

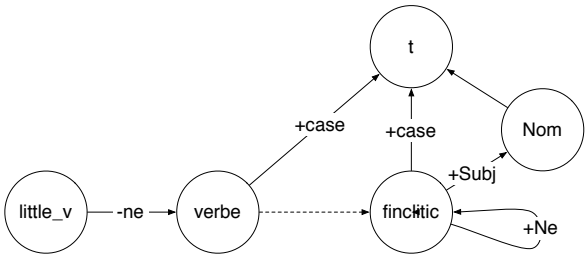


Fig. 46 – Traitement standard de la négation.

introduites qu'une fois par verbe sur l'inflexion. Il y aura donc au plus un passage dans la branche de cliticisation de la négation par verbe.

On représente sous forme de circuit l'ajout de la négation dans la figure 46.

Nous étendons la possibilité d'avoir des négations avec des adverbes de négation en ajoutant également dans le lexique les entrées leur correspondant, par exemple :

$$(\epsilon, jamais, \epsilon) : neg_2$$

$$(\epsilon, guerre, \epsilon) : neg_2$$

Négation avec sujet forclusif

Dans certains cas, la négation ne porte pas sur le verbe mais sur le sujet.

(52) Personne ne se rend au rendez-vous.
Rien ne sera fait.

Dans ce cas, nous ajoutons un trait de négation Ne à ces éléments, à la manière des pronoms personnels :

$$\vdash (\epsilon, personne, \epsilon) : (k \otimes (Ne \otimes d))$$

$$\vdash (\epsilon, Rien, \epsilon) : (k \otimes (Ne \otimes d))$$

La preuve doit contenir une hypothèse Ne introduite dans la cliticisation et utilisée par le sujet négatif (qui prendra malgré tout sa position lors de la résolution du cas car c'est la dernière hypothèse introduite dans le déplacement).

Le fait de dissocier les marqueurs de négation dans leur propre formule permet que leurs présences ne soient pas relatives l'une de l'autre. Une hypothèse négative introduite puis utilisée rend l'analyse acceptante. On a donc une unique entrée pour le marqueur de négation. Voici un exemple où l'on voit le caractère factorisé du formalisme et où l'ajout d'entrées pour le cas du sujet forclusif permet d'utiliser les dérivations précédentes.

Enfin, une fois de plus, du fait de la SMC, nous ne pourrons pas avoir simultanément une négation sur le verbe et une sur le sujet car au moment de la cliticisation du *ne* il y aurait deux hypothèses Ne en même temps. Ceci nous permet de refuser des phrases du type :

(53) * Personne ne ne le répare pas.

8.2.4 Exemple d'analyse

Nous présentons en détails une analyse obtenue pour un énoncé utilisant plusieurs clitiques (dont un pronom personnel) et la négation :

(54) Je ne la lui donne pas

Nous rappelons que la terminologie de *minimaliste* doit être considérée pour l'ensemble du système proposé et non uniquement pour la taille des résultats, ce que nous avons exposé au chapitre 1.

Afin de simplifier la dérivation nous supposerons que le verbe possède déjà son comportement syntaxique. Dans la présentation de la dérivation sémantique de la section suivante nous reprendrons une dérivation standard.

Dérivation 129. *On rappelle que l'on note $d_{[u]}$ une hypothèse d dépendante de la variable de chaîne u ($(\epsilon, u, \epsilon) : d$).*

Soit le lexique :

je	$(\epsilon, je, \epsilon) :$	$Subj \backslash Nom \,/\, finclitique$
la	$(\epsilon, la, \epsilon) :$	$G \backslash acc \,/_< dat$
lui	$(\epsilon, lui, \epsilon) :$	$F \backslash dat \,/_< clitique$
donne	$(\epsilon, donne, \epsilon) :$	$(d \backslash (k \backslash v)) \,/\, p$
ϵ_{je}	$(\epsilon, \epsilon, \epsilon) :$	$(k \otimes (subj \otimes d))$
ϵ_{lui}	$(\epsilon, \epsilon, \epsilon) :$	$F \otimes p$
ϵ_{la}	$(\epsilon, \epsilon, \epsilon) :$	$G \otimes (k \otimes d)$
ϵ_{ne}	$(\epsilon, \epsilon, \epsilon) :$	$Ne \otimes neg_1$
pas	$(\epsilon, pas, \epsilon) :$	neg_2
ne	$(\epsilon, ne, \epsilon) :$	$Ne \setminus finclitique_> \,/\, finclitique$
sortie de la dérivation	$(\epsilon, \epsilon, \epsilon) :$	$case \setminus t \,/\, Nom$
inflexion	$(\epsilon, \epsilon, \epsilon) :$	$neg_1 \setminus little\_v_> \setminus verbe_> \,/\, neg_2$
complément de la forme verbale	$(\epsilon, \epsilon, \epsilon) :$	$d \setminus little\_v_> \,/\, v$
comp	$(\epsilon, \epsilon, \epsilon) :$	$c \,/\, t$
sortie de la cliticisation	$(\epsilon, \epsilon, \epsilon) :$	$finclitique_> \,/\, acc$
entrée dans la cliticisation	$(\epsilon, \epsilon, \epsilon) :$	$clitique_> \,/\, verbe$

1. *entrée lexicale :* $(\epsilon, donne, \epsilon) : (d \backslash (k \backslash v)) / p$
   hypothèse : $u : p \vdash (\epsilon, u, \epsilon) : p$
   fusion : introduction de la position de l'argument à phonologie vide du datif :
   $$\frac{(\epsilon, donne, \epsilon) : d \backslash k \backslash v \,/\, p \quad u : p \vdash (\epsilon, u, \epsilon) : p}{u : p \vdash (\epsilon, donne, u) : d \backslash k \backslash v} [mg]$$
2. *hypothèse :* $v : d \vdash (\epsilon, v, \epsilon) : d$
   fusion : introduction de la position de l'argument à phonologie vide de l'accusatif :
   $$\frac{v : d \vdash (\epsilon, v, \epsilon) : d \quad u : p \vdash (\epsilon, donne, u) : d \backslash k \backslash v}{v : d, u : p \vdash (v, donne, u) : k \backslash v} [mg]$$
3. *déplacement : résolution du cas pour l'objet :*
   introduction d'une hypothèse $w : k \vdash (\epsilon, w, \epsilon) : k$

$$\frac{w : k \vdash (\epsilon, w, \epsilon) : k \quad v : d, u : p \vdash (v, donne, u) : k\backslash v}{w : k, v : d, u : p \vdash (w\ v, donne, u) : k\backslash v}\,[mg]$$

puis substitution dans la preuve :

$$\frac{x : k \otimes d \vdash (\epsilon, x, \epsilon) : k \otimes d \quad w : k, v : d, u : p \vdash (w\ v, donne, u) : k\backslash v}{x : k \otimes d, u : p \vdash (x, donne, u) : v}\,[mv]$$

4. *entrée lecicale :* $(\epsilon, \epsilon, \epsilon) : d \setminus little\_v_{>} / v$
   fusion : introduction de la demande d'un DP (sujet) pour le verbe :

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : d \setminus little\_v_{>} / v \quad x : k \otimes d, u : p \vdash (x, donne, u) : v}{x : k \otimes d, u : p \vdash (\epsilon, donne, x\ u) : d \setminus little\_v}\,[mg]$$

5. *entrée lexicale : introduction de la position de l'argument à phonologie vide du nominatif :* $y : d \vdash (\epsilon, y, \epsilon) : d$
   fusion : introduction de l'argument à phonologie vide du nominatif :

$$\frac{y : d \vdash (\epsilon, y, \epsilon) : d \quad x : k \otimes d, u : p \vdash (\epsilon, donne, x\ u) : d \setminus little\_v}{y : d, x : k \otimes d, u : p \vdash (y, donne, x\ u) : little\_v}\,[mg]$$

6. *entrée lexicale :* $(\epsilon, \epsilon, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe_{>} / neg_2$
   entrée lexicale : $(\epsilon, pas, \epsilon) : neg_2$
   fusion : construction de la première partie de l'inflexion négative :

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe_{>} / neg_2 \quad \vdash (\epsilon, pas, \epsilon) : neg_2}{\vdash (\epsilon, \epsilon, pas) : neg_1 \setminus little\_v_{>} \setminus verbe}\,[mg]$$

7. *entrée lexicale :* $z : neg_1 \vdash (\epsilon, z, \epsilon) : neg_1$
   fusion : fin de la construction de l'inflexion négative :

$$\frac{z : neg_1 \vdash (\epsilon, z, \epsilon) : neg_1 \quad \vdash (\epsilon, inflex\ pas, \epsilon) : neg_1 \setminus little\_v_{>} \setminus verbe}{z : neg_1 \vdash (z, inflex\ pas, \epsilon) : little\_v_{>} \setminus verbe}\,[mg]$$

8. *fusion entre cette inflexion complexe et la dérivation sur le verbe. Nous utilisons une version simplifiée de la dérivation précédente pour des questions de taille des arbres.*

$$\frac{y : d, x : k \otimes d, u : p \vdash (y, donne, x\ u) : little\_v \quad \begin{array}{l} z : neg_1 \vdash (z, inflex\ pas, \epsilon) : \\ little\_v_{>} \setminus verbe \end{array}}{y : d, x : k \otimes d, u : p, z : neg_1 \vdash (z, donne\ inflex\ pas, y\ x) : verbe}\,[mg]$$

9. *entrée lexicale :* $(\epsilon, \epsilon, \epsilon) : clitique_{>} / verbe$
   fusion : entrée dans la cliticisation :

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : clitique_{>} / verbe \quad \begin{array}{l} y : d, x : k \otimes d, u : p, z : neg_1 \vdash \\ (z, donne\ inflex\ pas, y\ x) : verbe \end{array}}{y : d, x : k \otimes d, u : p, z : neg_1 \vdash (\epsilon, donne\ inflex\ pas, z\ y\ x) : clitique_{>} / verbe}\,[mg]$$

10. *entrée lexicale :* $(\epsilon, lui, \epsilon) : F\backslash dat /_{<} clitique$
    fusion : cliticisation du datif :

$$\frac{\begin{array}{ll}\vdash (\epsilon, lui, \epsilon) : & y : d, x : k \otimes d, u : p, z : neg_1 \vdash \\ F\backslash dat \,/_< \, clitique & (\epsilon, donne\ inflex\ pas, z\ y\ x) : clitique_> \,/\, verbe\end{array}}{y : d, x : k \otimes d, u : p, z : neg_1 \vdash (\epsilon, lui\ donne\ inflex\ pas, z\ y\ x) : F\backslash dat}\ [mg]$$

11. déplacement : introduction d'une hypothèse F puis substitution du ϵ_{lui} :

$$\frac{\begin{array}{cc}a : F \vdash (\epsilon, a, \epsilon) : F & y : d, x : k \otimes d, u : p, z : neg_1 \vdash \\ & (\epsilon, lui\ donne\ inflex\ pas, z\ y\ x) : F\backslash dat\end{array}}{a : F, y : d, x : k \otimes d, u : p, z : neg_1 \vdash (a, lui\ donne\ inflex\ pas, z\ y\ x) : dat}\ [mg]$$

$$\frac{\begin{array}{cc}\vdash (\epsilon, \epsilon_{lui}, \epsilon) : F \otimes p & a : F, y : d, x : k \otimes d, u : p, z : neg_1 \vdash \\ & (a, lui\ donne\ inflex\ pas, z\ y\ x) : dat\end{array}}{y : d, x : k \otimes d, z : neg_1 \vdash (\epsilon_{lui}, lui\ donne\ inflex\ pas, z\ y) : dat}\ [mv]$$

12. la cliticisation de l'accusatif se fait de manière analogue :
entrée lexicale : $(\epsilon, la, \epsilon) : G\backslash acc \,/_< dat$
fusion : cliticisation de l'accusatif :

$$\frac{\begin{array}{cc}\vdash (\epsilon, la, \epsilon) : G\backslash acc \,/_< dat & y : d, x : k \otimes d, z : neg_1 \vdash \\ & (\epsilon_{lui}, lui; donne\ inflex\ pas, z\ y) : dat\end{array}}{y : d, x : k \otimes d, z : neg_1 \vdash (\epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : G\backslash acc}\ [mg]$$

introduction d'une hypothèse $b : G \vdash (\epsilon, b, \epsilon) : G$ *:*

$$\frac{\begin{array}{cc}b : G \vdash (\epsilon, b, \epsilon) : G & y : d, x : k \otimes d, z : neg_1 \vdash \\ & (\epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : G\backslash acc\end{array}}{b : G, y : d, x : k \otimes d, z : neg_1 \vdash (b\ \epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : acc}\ [mg]$$

Substitution par $(\epsilon, \epsilon_{la}, \epsilon) : G \otimes (k \otimes d)$ *:*

$$\frac{\begin{array}{cc}\vdash (\epsilon, \epsilon_{la}, \epsilon) : G \otimes (k \otimes d) & b : G, y : d, x : k \otimes d, z : neg_1 \vdash \\ & (b\ \epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : acc\end{array}}{y : d, z : neg_1 \vdash (\epsilon_{la}\ \epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : acc}\ [mv]$$

13. sortie du cluster de clitiques :
entrée lexicale : $(\epsilon, \epsilon, \epsilon) : finclitique_> \,/\, acc$ *:*

$$\frac{\begin{array}{cc}\vdash (\epsilon, \epsilon, \epsilon) : finclitique_> \,/\, acc & y : d, z : neg_1 \vdash \\ & (\epsilon_{la}\ \epsilon_{lui}, la\ lui\ donne\ inflex\ pas, z\ y) : acc\end{array}}{y : d, z : neg_1 \vdash (\epsilon, la\ lui\ donne\ inflex\ pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : finclitique}\ [mg]$$

14. entrée lexicale : $(\epsilon, ne, \epsilon) : Ne \setminus finclitique_> \,/\, finclitique$
fusion : introduction du marqueur de négation :

$$\frac{\begin{array}{ll}\vdash (\epsilon, ne, \epsilon) : & y : d, z : neg_1 \vdash (\epsilon, la\ lui\ donne\ inflex \\ Ne \setminus finclitique_> \,/\, finclitique & pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : finclitique\end{array}}{y : d, z : neg_1 \vdash (\epsilon, ne\ la\ lui\ donne\ inflex\ pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : Ne \setminus finclitique}\ [mg]$$

introduction d'une hypothèse Ne *:*

$$\frac{c : Ne \vdash (\epsilon, c, \epsilon) : Ne \qquad \begin{array}{l} y : d, z : neg_1 \vdash (\epsilon, nela\ lui\ donne\ inflex \\ pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : Ne \setminus finclitique \end{array}}{c : Ne, y : d, z : neg_1 \vdash (c, ne\ la\ lui\ donne\ inflex\ pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : finclitique}\ [mg]$$

Substitution par $(\epsilon, \epsilon, \epsilon) : Ne \otimes neg_1$ *:*

$$\frac{\begin{array}{ll} \vdash (\epsilon, \epsilon, \epsilon) : & c : Ne, y : d, z : neg_1 \vdash \\ Ne \otimes neg_1 & (c, nela\ lui\ donne\ inflex\ pas, \epsilon_{la}\ \epsilon_{lui}\ z\ y) : finclitique \end{array}}{y : d \vdash (\epsilon_{ne}, ne\ la\ lui\ donne\ inflex\ pas, \epsilon_{la}\ \epsilon_{lui}\ y) : finclitique}\ [mv]$$

15. *entrée lexicale :* $(\epsilon, je, \epsilon) : Subj\backslash Nom\ /\ finclitique$ *fusion : introduction du clitique nominatif :*

$$\frac{\begin{array}{ll} \vdash (\epsilon, je, \epsilon) : & y : d \vdash (\epsilon_{ne}, nela\ lui\ donne\ inflex \\ Subj\backslash Nom\ /\ finclitique & pas, \epsilon_{la}\ \epsilon_{lui}\ y) : finclitique \end{array}}{y : d \vdash (\epsilon\ je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}\ y) : Subj\backslash Nom}\ [mg]$$

introduction d'une hypothèse $Subj$ *:*

$$\frac{\begin{array}{ll} d : Subj \vdash (\epsilon, d, \epsilon) : Subj & y : d \vdash (\epsilon\ je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex \\ & pas\ \epsilon_{la}\ \epsilon_{lui}\ y) : Subj\backslash Nom \end{array}}{d : Subj, y : d \vdash (d, je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}\ y) : Nom}\ [mg]$$

Substution par $Subj \otimes d$ *:*

$$\frac{\begin{array}{ll} e : Subj \otimes d \vdash & d : Subj, y : d \vdash (d, je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex \\ (\epsilon, e, \epsilon) : Subj \otimes d & pas\ \epsilon_{la}\ \epsilon_{lui}\ y) : Nom \end{array}}{e : Subj \otimes d \vdash (e, je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) : Nom}\ [mv]$$

16. *entrée lexicale :* $(\epsilon, \epsilon, \epsilon) : case \setminus t\ /\ Nom$
fusion : vers une structure de phrase :

$$\frac{\begin{array}{ll} \vdash (\epsilon, \epsilon, \epsilon) : & e : Subj \otimes d \vdash (e,\ je, \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex \\ case \setminus t\ /\ Nom & pas\ \epsilon_{la}\ \epsilon_{lui}) : Nom \end{array}}{e : Subj \otimes d \vdash : (\epsilon, \epsilon, e\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) case \setminus t}\ [mg]$$

introduction d'une hypothèse case :

$$\frac{\begin{array}{ll} f : case \vdash & e : Subj \otimes d \vdash : (\epsilon, \epsilon, e\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex \\ (\epsilon, f, \epsilon) : case & pas\ \epsilon_{la}\ \epsilon_{lui}) case \setminus t \end{array}}{case_{[f]}, e : Subj \otimes d \vdash : (f, \epsilon, e\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) : t}\ [mg]$$

Substitution par $(\epsilon, \epsilon, \epsilon) : (k \otimes (subj \otimes d))$ *:*

$$\frac{\begin{array}{ll} \vdash (\epsilon, \epsilon, \epsilon) : & case_{[f]}, e : Subj \otimes d \vdash : (\epsilon, \epsilon, e\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex \\ (k \otimes (subj \otimes d)) & pas\ \epsilon_{la}\ \epsilon_{lui}) : t \end{array}}{\vdash : (\epsilon_{je}, \epsilon, je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) : t}\ [mg]$$

17. *puis passage dans la partie de complementizer : entrée lexicale :* $(\epsilon, \epsilon, \epsilon) : c\ /\ t$ *:*

$$\frac{\vdash (\epsilon, \epsilon, \epsilon) : c\ /\ t \quad \vdash : (\epsilon_{je}, \epsilon, je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) : t}{\vdash : (\epsilon, \epsilon, \epsilon_{je}\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui}) : c}\ [mg]$$

Fin de la dérivation qui reconnaît alors la chaîne :

$concat((\epsilon, \epsilon, \epsilon_{je}\ je\ \epsilon_{ne}\ ne\ la\ lui\ donne\ inflex\ pas\ \epsilon_{la}\ \epsilon_{lui})) =$ Je ne la lui donne pas.

Cette dérivation réalisée avec la GM équivalente est présentée dans l'annexe E. Les GMs permettent de mieux appréhender les différentes étapes du calcul car les éléments déplacés sont toujours présents.

8.3 contrepartie sémantique

Le traitement sémantique de la cliticisation est relativement clair. De la même manière que nous avons supposé que la phonologie était partagée par chacune des parties de la cliticisation, nous supposons que le même phénomène se produit pour la sémantique.

Il y a donc une partie sémantiquement vide et une autre qui porte la sémantique. C'est l'élément qui est argument du verbe (phonologiquement vide) qui introduira la sémantique du clitique. Celle-ci sera une variable devant être liée dans le contexte, comme cela est classiquement le cas dans les interfaces syntaxe/sémantique. Nous proposons d'utiliser les analyses et algorithmes permettant, pour la syntaxe générative, de lier les variables dans le contexte comme proposé dans [Bon06]. La partie vide de la cliticisation est une fonction d'identité particulière. En effet, comme nous avons supposé que chaque trait possède une contrepartie sémantique, une variable correspondant au trait de déplacement du clitique (vérification de la présence simultanée d'un argument à phonologie vide et d'un clitique) est nécessaire. Cependant, son apport sémantique n'est pas défini. Pour cela, on utilise une variable non reprise dans la définition du corps du terme.

entrée lexicale	forme syntaxique	forme sémantique
la	$G \setminus acc /_{<} clitique$	$\lambda P \lambda u.P$
ϵ_{la}	$G \otimes (k \otimes d)$	$x^{(1)}$

(1) Variable libre, liée dans le contexte

Nous introduisons la sémantique de la négation. De manière analogue, elle se divise entre le forclusif et le clitique en suivant les mêmes principes que pour la cliticisation. La sémantique est donc apportée uniquement par le forclusif. Ce traitement permet de construire les formules pour les négations avec sujet forclusif, comme *personne*. Dans le cas d'utilisation de sujet forclusif, la formule n'est pas une négation mais c'est la quantification qui porte cette idée de négation. Il n'existerait donc pas x dans la formule associée à *personne* et la formule globale de l'énoncé serait positive.

Nous reprenons un exemple contenant deux clitiques dont un sujet ainsi que la négation, pour présenter le calcul des formules sémantiques avec clitique et négation :

(55) Je ne la répare pas.

Pour cela, nous allons associer à chaque entrée lexicale une contrepartie sémantique :

entrée lexicale	forme syntaxique	forme sémantique
je	$Subj \setminus Nom \,/\, finclitique$	$\lambda P \lambda u.P$
la	$G \setminus acc \,/_{<} clitique$	$\lambda P \lambda u.P$
répare	$v \,/\, d$	$\lambda x \lambda y \lambda e.réparer(e, x, y)$
ϵ_{je}	$k \otimes (Subj \otimes d)$	moi
ϵ_{la}	$G \otimes (k \otimes d)$	$x^{(1)}$
ϵ_{ne}	$Ne \otimes neg_1$	d
pas	Neg_2	$\lambda P.\neg P$
ne	$Ne \setminus finclitique_{>} \,/\, finclitique$	$\lambda P \lambda u.P$
inflexion	$Neg_1 \setminus little\_v_{>} \setminus verbe_{>} \,/\, Neg_2$	$\lambda P \lambda u \lambda Q \lambda e.P(pres(e))$
forme verbale étendue	$k \setminus d \setminus little\_v_{>} \,/\, v$	$\lambda P \lambda x_2 \lambda y \lambda e.P(y, e) \wedge patient(e, x_2)$
comp	$c \,/\, t$	$\lambda P.P(e)$
sortie de la cliticisation	$finclitique_{>} \,/\, acc$	$\lambda P.P$
entrée cliticisation	$clitique_{>} \,/\, verbe$	Id
verbe cliticisé	$k \setminus t \,/\, Nom$	$\lambda P.P$

(1) Variable liée dans le contexte.

L'analyse syntaxique avec la GMC est analogue à la précédente. Celle que nous utilisons comme support pour les étapes du calcul sémantique étant moins longue, le lecteur pourra se reporter à la section précédente pour davantage de détails syntaxiques.

Dans la première partie, il s'agit de fournir une première variable au verbe puis son comportement syntaxique implique l'introduction du prédicat *patient*.

$$\frac{\lambda x \lambda y \lambda e.réparer(e, x, y) :\vdash v \,/\, d \quad u : d \vdash d}{\lambda y \lambda e.réparer(e, u, y) : d \vdash v}\,[mg]$$

$$\frac{\lambda y \lambda e.réparer(e, u, y) : d \vdash v \qquad \begin{array}{l}\lambda P \lambda x_2 \lambda y \lambda e.P(y, e) \wedge patient(e, x_2) : \\ \vdash k \setminus d \setminus little\_v_{>} \,/\, v\end{array}}{\lambda x_2 \lambda y \lambda e.réparer(e, u, y) \wedge patient(e, x_2) : d \vdash d \setminus little\_v}\,[mg]$$

Puis, la dérivation ajoute une nouvelle hypothèse et, pour que la preuve soit sous forme normale, elle lie les deux hypothèses ensemble.

$$\frac{v : k \vdash k \quad \lambda x_2 \lambda y \lambda e.réparer(e, u, y) \wedge patient(e, x_2) : u : d \vdash d \setminus little\_v_{>} \,/\, v}{\lambda y \lambda e.réparer(e, u, y) \wedge patient(e, v) : k, d \vdash d \setminus little\_v}\,[mg]$$

$$\frac{w(t) : k \otimes d \vdash k \otimes d \quad \lambda y \lambda e.réparer(e, u, y) \wedge patient(e, v) : k, d \vdash d \setminus little\_v_{>} \,/\, v}{\lambda y \lambda e.réparer(e, w(t), y) \wedge patient(e, t) : k \otimes d \vdash d \setminus little\_v}\,[mv]$$

Puis on introduit une nouvelle hypothèse (DP sujet) :

$$\frac{x : d \vdash d \quad \lambda y \lambda e.réparer(e, w(t), y) \wedge patient(e, t) : k \otimes d \vdash d \setminus little\_v}{\lambda e.réparer(e, w(t), x) \wedge patient(e, t) : d, k \otimes d \vdash little\_v}\,[mg]$$

Parallèlement on construit l'inflexion complexe pour la négation, en construisant un prédicat présent négatif pour la variable de réification de la formule :

$$\frac{\lambda P\lambda u\lambda Q\lambda e.P(pres(e)) \wedge Q :\vdash Neg_1 \setminus little\_v_{>} \setminus verbe_{>} / Neg_2 \quad \lambda P.\neg P : Neg_2}{\lambda u\lambda Q\lambda e.\neg pres(e) \wedge Q :\vdash Neg_1 \setminus little\_v_{>} \setminus verbe}\,[mg]$$

$$\frac{y : Neg_1 \vdash Neg_1 \quad \lambda u\lambda Q\lambda e.\neg pres(e) \wedge Q :\vdash Neg_1 \setminus little\_v_{>} \setminus verbe}{\lambda Q\lambda e.\neg pres(e) \wedge Q : Neg_1 \vdash little\_v_{>} \setminus verbe}\,[mg]$$

Ce terme est appliqué à la dérivation principale précédente :

$$\frac{\begin{array}{ll}\lambda e.réparer(e, w(t), x) \wedge patient(e, t) : & \lambda Q\lambda e.\neg pres(e) \wedge Q : \\ d, k \otimes d \vdash little\_v & Neg_1 \vdash little\_v_{>} \setminus verbe\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : d, k \otimes d, Neg_1 \vdash verbe}\,[mg]$$

À présent, la dérivation entre dans la phase de cliticisation :

$$\frac{\lambda P.P :\vdash clitique \,/\, verbe \quad \begin{array}{l}\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : \\ d, k \otimes d, Neg_1 \vdash verbe\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : d, k \otimes d, Neg_1 \vdash clitique}\,[mg]$$

Selon l'ordre de traitement des clitiques, on cliticise l'objet. Le terme foncteur ne modifie pas la formule et permet l'application avec une variable non-utilisée dans le corps du terme :

$$\frac{\lambda P\lambda u.P : G \setminus acc /_{<} clitique \quad \begin{array}{l}\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : \\ d, k \otimes d, Neg_1 \vdash clitique\end{array}}{\lambda u\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : d, k \otimes d, Neg_1 \vdash G \setminus acc}\,[mg]$$

Ceci implique l'introduction d'une variable pour l'hypothèse G qui annule l'abstracteur. La substitution suivante sur l'argument du verbe à phonologie vide porte la sémantique du clitique :

$$\frac{B :\vdash G \otimes (k \otimes d) \quad \begin{array}{l}\lambda u\lambda e.\neg pres(e) \wedge réparer(e, w(t), x) \wedge patient(e, t) : \\ d, k \otimes d, Neg_1 \vdash G \setminus acc\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e, B, x) \wedge patient(e, B) : d, Neg_1 \vdash acc}\,[mg]$$

La dérivation sort du cluster de clitique

$$\frac{\lambda P.P :\vdash finclitique \,/\, acc \quad \begin{array}{l}\lambda e.\neg pres(e) \wedge réparer(e, B, x) \wedge patient(e, B) : \\ d, Neg_1 \vdash acc\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e, B, x) \wedge patient(e, B) : d, Neg_1 \vdash finclitique}\,[mg]$$

L'énoncé possède une négation, on la cliticise avant de poursuivre. Dans la contrepartie sémantique, il ne se passe rien puisque c'est le forclusif qui a introduit le marqueur de négation. On utilise ici aussi un terme avec une variable qui n'apparaît pas dans le corps du terme.

$$\frac{\begin{array}{ll}\lambda P\lambda u.P : & \lambda e.\neg pres(e) \wedge réparer(e, B, x) \wedge patient(e, B) : \\ \vdash Ne \setminus finclitique_{>} / finclitique & d, Neg_1 \vdash finclitique\end{array}}{\lambda u\lambda e.\neg pres(e) \wedge réparer(e, B, x) \wedge patient(e, B) : d, Neg_1 \vdash Ne \setminus finclitique}\,[mg]$$

$$\frac{c : Ne \vdash Ne \qquad \begin{array}{l}\lambda u\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : \\ d, Neg_1 \vdash Ne \setminus finclitique\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : Ne, d, Neg_1 \vdash finclitique}\,[mg]$$

$$\frac{d :\vdash Ne \otimes Neg_1 \qquad \begin{array}{l}\lambda u\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : \\ Ne, d, Neg_1 \vdash Ne \setminus finclitique\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : d \vdash finclitique}\,[mv]$$

Puis on cliticise le sujet en introduisant un prédicat avec une variable qui n'apparaît pas dans le corps du terme :

$$\frac{\begin{array}{ll}\lambda P\lambda u.P : & \lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : \\ \vdash Subj \setminus Nom \,/\, finclitique & d \vdash finclitique\end{array}}{\lambda u\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : d \vdash Subj \setminus Nom}\,[mg]$$

$$\frac{e : Subj \vdash Subj \quad \lambda u\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : d \vdash Subj \setminus Nom}{\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : Subj, d \vdash Nom}\,[mg]$$

$$\frac{f(t) : Subj \otimes d \vdash Subj \otimes d \qquad \begin{array}{l}\lambda e.\neg pres(e) \wedge réparer(e,B,x) \wedge patient(e,B) : \\ Subj, d \vdash Nom\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge patient(e,B) : Subj \otimes d \vdash Nom}\,[mv]$$

Enfin, on sort de la cliticisation, ce qui permet d'introduire la variable relative à l'argument agentif du verbe, représenté par *moi* :

$$\frac{\begin{array}{ll}\lambda P\lambda y_2\lambda e.P(e) \wedge agent(e,y_2) : & \lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge patient(e,B) : \\ \vdash k \setminus t \,/\, Nom & Subj, d \vdash Nom\end{array}}{\lambda y_2\lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge patient(e,B) \wedge agent(e,y_2) : Subj \otimes d \vdash k \setminus t}\,[mv]$$

$$\frac{g : k \vdash k \qquad \begin{array}{l}\lambda y_2\lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge \\ patient(e,B) \wedge agent(e,y_2) : Subj \otimes d \vdash k \setminus t\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge patient(e,B) \wedge agent(e,g) : k, Subj \otimes d \vdash t}\,[mv]$$

$$\frac{moi :\vdash k \otimes (Subj \otimes d) \qquad \begin{array}{l}\lambda e.\neg pres(e) \wedge réparer(e,B,f(t)) \wedge \\ patient(e,B) \wedge agent(e,g) : Subj \otimes d \vdash t\end{array}}{\lambda e.\neg pres(e) \wedge réparer(e,B,moi) \wedge patient(e,B) \wedge agent(e,moi) :\vdash t}\,[mv]$$

Finalement, comme dans toutes les dérivations, nous utilisons l'entrée de *complementizer* :

$$\frac{\lambda P.P(e) \vdash c \,/\, t \;\; \lambda e.\neg pres(e) \wedge réparer(e,B,moi) \wedge patient(e,B) \wedge agent(e,moi) :\vdash t}{\neg pres(e) \wedge réparer(e,B,moi) \wedge patient(e,B) \wedge agent(e,moi) :\vdash c}\,[mv]$$

On obtient donc la formule

$$\neg pres(e) \wedge réparer(e,B,moi) \wedge patient(e,B) \wedge agent(e,moi)$$

où B doit être lié dans le contexte.

Dans l'algorithme de Bonato, [Bon06], certaines variables apparaissent avec une marque particulière permettant ou non de les lier soit dans la *local clause*, soit à l'extérieur de la *local clause*, cela étant dépendant de leur construction sémantique. La dissociation du traitement des clitiques permet alors d'associer une partie de la sémantique de chacune des composantes. De plus, la distinction apportée sur le domaine de liage est très dépendante de la forme phonologique prise par le clitique. Ainsi, il est possible d'associer un prédicat définissant la frontière du liage de la variable à l'entrée portant la forme phonologique du clitique, ce qui permet de spécifier le type de clitique qui intervient dans l'énoncé. Enfin, la *Discourse Representative Theory* (DRT) est un outil pertinent pour la modélisation des clauses dans les énoncés. Les informations collectées au cours de la dérivation permettent d'apporter des données supplémentaires aux algorithmes de résolution d'anaphore.

8.4 Autour du groupe verbal - cas simple

Ce type de traitement pour les clitiques en français peut simplement être intégré dans des analyses à couverture plus large. Nous allons voir dans cette partie comment des phénomènes syntaxiques autour du groupe verbal peuvent être encodés et comment la cliticisation fonctionne avec ceux-ci.

8.4.1 Montée sur les auxiliaires

La montée des clitiques dans ce type d'analyses se fait par dessus tout le groupe verbal, donc en particulier par dessus l'auxiliaire. Le problème est d'introduire cet auxiliaire dans l'analyse.

Pour cela, on suppose que l'inflexion transforme un *little_v* en un nouveau type nécessitant un auxiliaire si le temps est composé. L'auxiliaire se combine donc avec le verbe après la phase d'inflexion et se concatène avec le verbe par une fusion de tête. Le résultat est alors un élément portant la catégorie *verbe*. Les clitiques prennent leur place une fois que cette partie de l'analyse a été effectuée. Par exemple, pour la phrase :

(56) Je l'ai vu.

Nous construisons le groupe verbal ai vu et la montée des clitiques se fait sur tout ce groupe.

(57) ai vu $[\overline{Nom}]$ $[\overline{F}]$

l'$_{[F]}$ ai vu $[\overline{Nom}]$ $[\overline{F}]$ $\rightarrow$ l'ai vu $[\overline{Nom}]$

Je$_{[Nom]}$ l'ai vu $[\overline{Nom}]$ $\rightarrow$ Je l'ai vu

De plus, d'autres mots peuvent être insérés entre le verbe et son auxiliaire. Par exemple, pour des phrases avec adverbe, on analyse l'énoncé de manière analogue en construisant le groupe auxiliaire abverbe verbe. Les clitiques montent donc au-dessus de tout le groupe (donc de l'abverbe). Par exemple : *Je l'ai souvent vu* sera traité de la manière suivante :

(58) ai souvent vu $\epsilon_{[\overline{Nom}]}$ $\epsilon_{[\overline{F}]}$

l'$_{[F]}$ ai souvent vu $\epsilon_{[\overline{Nom}]}$ $\epsilon_{[\overline{F}]}$ $\rightarrow$ l'ai souvent vu ϵ_{-Nom}

Je$_{[Nom]}$ l'ai souvent vu $\epsilon_{[\overline{Nom}]}$ $\epsilon_{[\overline{F}]}$ $\rightarrow$ Je l'ai souvent vu

La montée des clitiques sur les auxiliaires n'a pas de conséquence sur le calcul sémantique. En effet, rien n'est ajouté, c'est simplement une utilisation plus globale des possibilités de calcul.

8.4.2 Impératif

Le mode impératif est un cas particulier pour les clitiques. Dans ce cas, si la séquence de clitiques ne contient pas de pronom réflexif, elle est placée à droite du verbe dans le même ordre, sinon on utilise un pronom fort. Nous modélisons cette situation par le fait que le verbe est déplacé en début de phrase à la fin de l'analyse. Ceci permet de conserver un unique traitement pour les clitiques et de déléguer le problème du placement du verbe à un autre mécanisme.

Le traitement que nous proposons ici ne gère pas l'utilisation des réflexifs à l'impératif (et même les accepte, ce qui n'est pas une bonne solution). Cependant, nous souhaitons mettre en avant une proposition de traitement de l'impératif qui ouvre sur une remarque plus générale en ce qui concerne les opérations utilisées sur le verbe.

Placement inverse des clitiques

À l'impératif, les clitiques se positionnent à droite du verbe.

(59) Donne le lui !

Nous voulons déplacer le verbe - et uniquement lui - à la fin de la dérivation. Nous utilisons une inflexion particulière, qui ne peut pas être complexe, comme dans le cas de la négation, mais qui doit introduire une rupture dans la chaîne des *Head-Movements*. Ainsi, le verbe avec son inflexion ne remonte plus le long de la dérivation et reste dans sa position avec un trait particulier permettant de le déplacer lorsque l'on utilise l'entrée *comp*.

De plus, le verbe ne doit pas contenir de sujet. Il faut donc que l'inflexion se situe avant qu'il reçoive tout son comportement syntaxique, c'est-à-dire lorsque c'est un v et non un *little_v*. On ajoute donc les entrées suivantes pour l'inflexion impérative :

$$(\epsilon, inflex, \epsilon) : (imp \otimes imp_1)_{>} / v$$

$$(\epsilon, \epsilon, \epsilon) : verbe / imp_1$$

Ainsi un verbe reçoit son inflexion et est utilisé par la seconde entrée qui réalise une fusion simple. Exemple de verbe recevant une inflexion impérative :

$$\frac{\vdash (\epsilon, inflex, \epsilon) : (imp \otimes imp_1)_{>} / v \quad \vdash (\epsilon, verbe, \epsilon) : v}{\vdash (\epsilon, verbeinflex, \epsilon) : (imp \otimes imp_1)} [mg]$$

La seconde entrée introduit une hypothèse imp_1 et l'entrée *comp* l'hypothèse *imp*. Le trait *imp* sera annulé par un trait d'une entrée *comp* pour l'impératif autorisant un déplacement du verbe et qui ne contiendra pas de donation de cas pour le nominatif.

Si le verbe contient une partie de cliticisation on ajoute :

$$(\epsilon, \epsilon, \epsilon) : IMP \setminus c / finclitique$$

Et pour les phrases à l'impératif sans clitique :

$$(\epsilon, \epsilon, \epsilon) : IMP \setminus c \,/\, verbe$$

Nous avons ainsi une modélisation pour les séquences de clitiques à l'impératif sans négation. Cependant, c'est la première fois que nous avons besoin d'utiliser un déplacement sur un verbe. Soit cette modélisation n'est pas pertinente, soit ne pas utiliser le déplacement pour le verbe est une contrainte trop forte. Il nous semble que le problème se pose de manière plus profonde et provient de l'ambiguïté de *Head-Movement* qui est en réalité un déplacement particulier. Cet autre niveau de déplacement devrait alors être modélisé par un autre connecteur, par exemple en utilisant le connecteur non-commutatif $\otimes$. Actuellement, nous ne pouvons pas proposer un traitement homogène mais ces remarques ouvrent une perspective d'amélioration du formalisme que les GMs n'offrent pas.

Impératif négatif

L'impératif avec négation ne modifie pas l'ordre entre le verbe et la séquence de clitiques.

(60) Ne le lui donne pas !

Cependant, pour conserver une homogénéité du traitement de l'impératif, nous souhaitons garder le même traitement en fin d'analyse. Il nous faut donc utiliser un déplacement sur une partie du verbe qui est vide. L'inflexion négative étant complexe, c'est à l'intérieur de celle-ci que nous veillerons à ne pas briser la suite de *Head-Movements*. La partie phonologique sera donc déplacée tout au long de la cliticisation. La dernière opération changera la position du verbe qui aura été, auparavant, complètement vidée de sa phonologie.

On ajoute de ce fait les entrées suivantes pour la construction de l'inflexion impérative, négative :

$$ne \setminus v_{>} \setminus (imp_1 \otimes imp)_{>} \,/\, neg_2$$

Ceci permet de construire une inflexion complexe reprenant les deux propositions précédentes avec la négation et l'impératif.

Dans la contrepartie sémantique, le traitement se fait de la même manière que pour la négation en général, où c'est le forclusif qui l'introduit sur le verbe.

Ainsi, on peut résumer le traitement syntaxique de l'impératif par le circuit de la figure 47

Un problème subsiste pour les cas de cliticisation à l'impératif. En effet, pour l'impératif sans négation, les pronoms réflexifs ne peuvent pas être cliticisés, mais doivent être des pronoms forts. Le traitement proposé ici accepte les phrases avec pronoms forts, mais également celles avec réflexifs faibles. Une solution rapide et efficace serait de dupliquer le cluster de clitiques pour le seul cas d'impératif non-négatif en enlevant la partie pour les réflexifs mais cela nécessite la duplication d'un grand nombre d'entrées lexicales. Or, la complexité des algorithmes d'analyse étant fonction du nombre d'entrées, cette solution ne semble pas pertinente. Il reste à noter que le traitement de l'impératif ouvre plusieurs questions structurelles sur le formalisme.

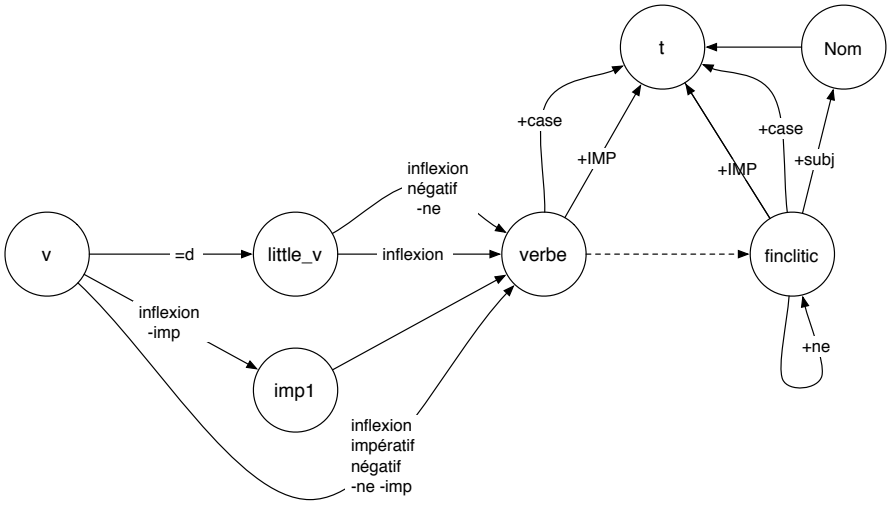


Fig. 47 – Inflexion impérative.

Verbes pronominaux

Nous avons introduit dans le cluster de clitiques la possibilité d'utiliser des clitiques pronominaux. Nous devons pouvoir les ajouter au verbe. On leur attribue pour cela un type nécessitant un argument réflexif vide. Dans ce cas, il faut introduire une entrée à phonologie vide particulière nécessitant une cliticisation réflexive :

$$F \otimes refl$$

Ainsi, un verbe pronominal doit fusionner avec une telle entrée pour devenir un verbe sans inflexion, représenté par un circuit dans la figure 48 :

$$(\epsilon, laver, \epsilon) : v \,/\, refl$$

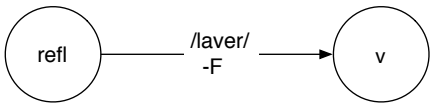


Fig. 48 – Verbes pronominaux.

8.5 Verbes enchâssés

Un autre phénomène syntaxique se produit fréquemment, il s'agit de l'enchâssement de groupes verbaux. Dans ce cas, le verbe du groupe enchâssé est à l'infinitif, et il peut ou non recevoir des clitiques en fonction du type de verbe qui le régit. Dans un

premier temps, nous montrerons comment construire des verbes à l'infinitif, puis nous présenterons les différentes possibilités et la manière dont nous les traitons syntaxiquement et sémantiquement.

8.5.1 Infinitif

Nous ajoutons une inflexion infinitive. De manière analogue à celle utilisée pour l'impératif, elle se place sur les verbes n'ayant pas encore reçu la possibilité de posséder un *DP* sujet (c'est-à-dire la voie). Elle prend donc un élément de type v pour rendre un *verbe*, en ajoutant la partie phonologique de l'infinitif à droite du verbe :

$$(\epsilon, -er, \epsilon) : verbe /_< v$$

Un verbe ayant reçu une inflexion infinitive n'est pas accepté comme une phrase, ce qui syntaxiquement ne pose pas de problème et se traduit dans notre système par le fait que pour passer au trait acceptant à partir de *verbe*, il nous faut réaliser une donation de cas pour le sujet (chose impossible car le verbe n'a justement pas reçu de sujet.) Les verbes à l'infinitif reçoivent comme les autres verbes la séquence de clitiques, sauf pour le placement du forclusif. C'est l'un des rares cas particuliers où la séquence de clitiques n'est pas homogène.

(61) Il souhaite ne pas en parler.

Nous proposons encore la construction d'une inflexion complexe contenant la négation. Celle-ci n'est pas agglomérée sur l'inflexion et chacune de ses composantes porte un trait particulier. Puis après la cliticisation, de *finclitique* on traite le forclusif vers un nouveau type *Negatif*. Celui-ci est directement suivi du traitement du *ne* qui retourne vers *finclitique*. On ajoute les entrées :

$$(\epsilon, pas, \epsilon) : neg \otimes neg_2$$

$$(\epsilon, jamais, \epsilon) : neg \otimes neg_2$$

$$(\epsilon, -er, \epsilon) : neg_1 \setminus v_{>} \setminus verbe / neg_2$$

Elles permettent de construire l'inflexion complexe suivante :

$$\frac{\vdash (\epsilon, -er, \epsilon) : neg_1 \setminus v_{>} \setminus verbe / neg_2 \quad u : neg_2 \vdash (\epsilon, u, \epsilon) : neg_2}{u : neg_2 \vdash (\epsilon, -er, u) : neg_1 \setminus v_{>} \setminus verbe}$$

$$\frac{v : neg_1 \vdash (\epsilon, v, \epsilon) : neg_1 \quad u : neg_2 \vdash (\epsilon, -er, u) : neg_1 \setminus v_{>} \setminus verbe}{v : neg_1, u : neg_2 \vdash (v, -er, u) : v_{>} \setminus verbe}$$

où l'hypothèse neg_2 sera celle de *pas* et l'hypothèse neg_1 celle de *ne*. Une fois de plus, cette inflexion se placera à droite dans l'analyse globale mais aucune partie phonologique ne restera dans cette structure.

De l'autre côté, dans la partie de cliticisation, nous ajoutons les entrées suivantes :

$$(\epsilon, \epsilon, \epsilon) : Neg \setminus negatif_{>} / finclitique$$

$$(\epsilon, ne, \epsilon) : ne \setminus finclitique / Negatif$$

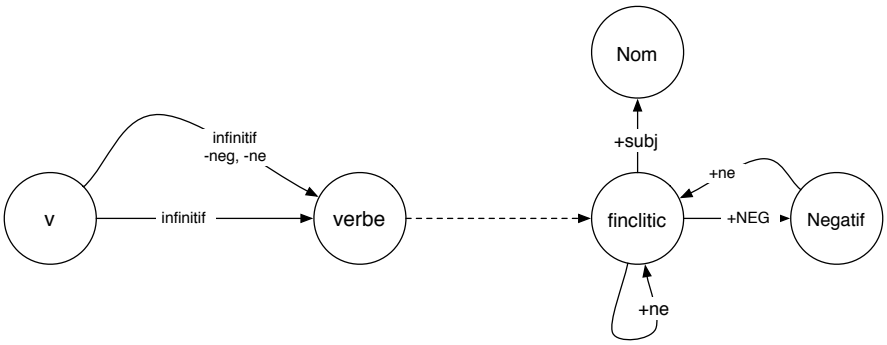


Fig. 49 – Infinitif.

On représente la construction d'un verbe à l'infinitif par le circuit de la figure 49.

Un groupe verbal avec un verbe à l'infinitif est donc soit de type *finclitique* s'il possède des clitiques et/ou une négation, soit de type verbe s'il n'a ni clitique, ni négation. La particularité de ces groupes est donc qu'ils n'ont pas reçu de *DP* sujet et ne peuvent en eux-mêmes être des dérivations acceptantes. Dans la contrepartie sémantique c'est la suite de la dérivation qui se chargera de leur attribuer une variable sujet.

Il existe un cas particulier de constituant avec un verbe à l'infinitif, lorsqu'on l'utilise comme *DP*. Dans ce cas, le verbe doit avoir reçu tous ses arguments (directement dans la première phase, soit par cliticisation) et est utilisé comme des *DP*s.

(62) Aimer permet de déplacer des montagnes.

Cependant, il ne peut pas être utilisé librement pour remplacer n'importe quel *DP*. Nous ne présentons donc pas de solution pour ces analyses, bien que l'on pourrait les modifier en *DP* à partir de *finclitique* et *verbe*. Pour assurer que ce sont bien des groupes avec verbe à l'infinitif, il faut ajouter un trait dans l'inflexion qui sera à vérifier dans cette transition, ce qui une nouvelle fois ne permet pas de proposer un traitement homogène.

Nous allons à présent voir comment utiliser ces groupes verbaux pour construire des phrases avec verbes enchâssés.

8.5.2 Verbe à contrôle

En français moderne, on distingue une classe particulière de verbes : les semi-auxiliaires [Gré75]. Les comportements syntaxique et sémantique des clitiques avec ces verbes sont plus complexes.

Définition 130. *Les semi-auxiliaires sont des verbes qui prennent comme argument un autre verbe à l'infinitif. Les deux verbes partagent alors un même argument.*

Par exemple :

(63) Il semble le lui donner

où *Il* est le sujet des deux verbes *sembler* et *donner*.

Cependant, à l'intérieur de cet ensemble, il faut distinguer des sous-ensembles car les semi-auxiliaires n'ont pas le même comportement.

Définition 131. *On distingue trois sous-classes de semi-auxiliaires :*

– *les* semi-auxiliaires causatifs *: verbes qui prennent un verbe et s'assemblent avec lui. Ceci donne les constructions syntaxiques suivantes :*

$$[clitique(s) + [semi - auxiliaire + verbe]]$$

faire faire quelque chose

(64) Je la fais réparer.

– *les* semi-auxiliaires lâches *: verbes qui prennent un verbe dont la forme verbale est complètement remplie et se placent devant eux. Ceci donne les constructions syntaxiques suivantes :*

$$[semi - auxiliaire + [clitique(s) + verbe]]$$

penser

(65) Je pense la réparer.

vouloir

(66) Je veux la réparer.

– *les* semi-auxiliaires mixtes *(plus rares) : ils permettent le positionnement des clitiques devant le verbe ou devant eux-mêmes. On obtient donc les constructions suivantes :*

$$[clitique(s) + semi - auxiliaire + [clitique(s) + verbe]]$$

laisser

(67) Je la laisse le lui donner.

faire : faire faire quelque chose à quelqu'un.

(68) – Je le fait la réparer.

Pour prendre en compte ces semi-auxiliaires, on leur donne un type syntaxique particulier prenant en argument un verbe à l'infinitif.

Définition 132. *On définit les* semi-auxiliaires causatifs *comme prenant un verbe sans sujet ayant reçu une inflexion infinitive mais n'ayant pas encore été cliticisé (s'il doit l'être). Ces éléments sont de type verbe.*

Le semi-auxiliaire utilise cet élément et devient un verbe sans sujet et n'ayant pas encore reçu son inflexion, donc un élément v pouvant lui aussi prendre de nouveaux arguments.

Par exemple :

$$(\epsilon, fait, \epsilon) : v \,/\, verbe$$

Grâce à cette définition, la cliticisation n'a pas encore eu lieu, et c'est après que le semi-auxiliaire a pris son inflexion qu'elle sera réalisée. Les clitiques se placeront donc devant lui. On représente cette procédure par le circuit de la figure 50 où la procédure ajoutée est représentée par une transition en gras et pointillés longs.

Définition 133. *On définit les* semi-auxiliaires lâches *comme prenant un verbe à l'infinitif (sans sujet) et terminant leur cliticisation. Le semi-auxiliaire intervient et permet de passer dans un chemin qui lui est propre pour recevoir son inflexion et son sujet.*

Par exemple :

$$(\epsilon, penser, \epsilon) : d \setminus big\_v \,/\, modifieur2$$

$$(\epsilon, pouvoir, \epsilon) : d \setminus big\_v \,/\, modifieur2$$

$$(\epsilon, vouloir, \epsilon) : d \setminus big\_v \,/\, modifieur2$$

Il faut donc lui donner son inflexion. Une fois qu'il a reçu son inflexion, ce groupe verbal peut soit être une phrase, soit être enchâssé dans une autre construction avec un semi-auxiliaire, mais la cliticisation a déjà été réalisée. Il faut donc pouvoir passer dans l'état t ou finclitique.

Il est possible de factoriser les entrées ajoutées par l'inflexion en introduisant un type intermédiaire et une entrée à phonologie vide permettant de passer à finclitique ou t. C'est la solution que nous choisissons pour cette grammaire.

$$(\epsilon, inflexion, \epsilon) : Big\_v_{>} \,/\, big\_v$$

$$(\epsilon, \epsilon, \epsilon) : t_{>} \,/\, Big\_v$$

$$(\epsilon, \epsilon, \epsilon) : finclitique_{>} \,/\, Big\_v$$

Une fois encore, nous modélisons tout ceci par un circuit, figure 50, où la procédure ajoutée est représentée par les transitions en gras.

Enfin, il reste à définir les semi-auxiliaires mixtes.

Définition 134. *Les* semi-auxiliaires *mixtes passent d'un verbe ayant recu ses clitiques (ou non) à l'état de verbe sans sujet, sans tous ses arguments, sans inflexion et pouvant être cliticisé (c'est-à-dire un élément de type v).*

Par exemple pour le verbe laisser, la première occurrence correspond à une construction avec infinitive avec clitique et la seconde sans :

$$(\epsilon, laisse, \epsilon) : d \setminus case \setminus v \,/\, finclitique$$

$$(\epsilon, laisse, \epsilon) : d \setminus case \setminus v \,/\, verbe$$

Ce traitement est modélisé par les transitions en pointillés gras sur la figure 50

Remarque 135. *Ces constructions peuvent également être itératives. On peut en effet construire des phrases avec plusieurs semi-auxiliaires, par exemple :*

(69) 1. *Je pense la faire réparer.*

2. *Je veux le laisser la réparer.*

Ces constructions sont réalisables directement à partir du lexique proposé par récursivité du système.

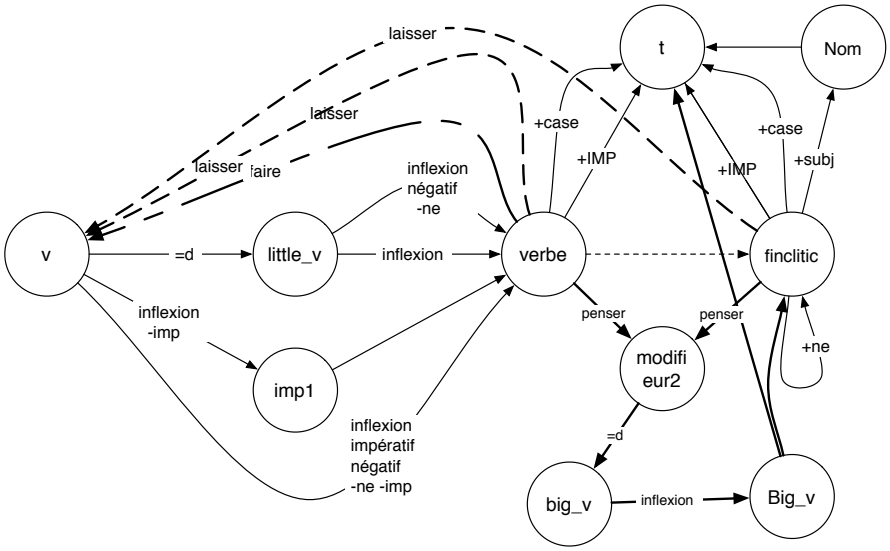


FIG. 50 – Semi-auxiliaires.

8.5.3 Subordonnées relatives

En s'inspirant du traitement des clitiques, nous proposons une modélisation des subordonnées relatives. La partie incise est composée d'une phrase à laquelle il manque un argument, ou une phrase ayant un argument à phonologie vide. C'est le relatif qui permet de décharger cet argument et de combiner la phrase avec un autre *DP* pour le modifier. On construit un élément de type t ayant un clitique non cliticisé. Le type de cliticisation manquant permet de différencier les utilisations d'un *que* et d'un *qui*.

La dérivation suit alors le scénario suivant :

(70)
- aime$_{verbe}$ $[\overline{F}]$
- Pierre aime$_t$ $[\overline{F}]$
- que$_{d\backslash d/F}$ Pierre aime $[\overline{F}]$
- que$_{d\backslash d}$ Pierre aime
- Marie$[\overline{case}]$ que$_d$ Pierre aime
- Marie que Pierre aime mange$_t$ une pomme.

Le relatif prend la place de l'entrée *comp* et est du type :

$$[que] :: [= t, +G, = d, d].$$

Si l'argument à partager est le sujet de la relative cela pose un problème car le constituant n'a pas encore reçu son cas. Pour cela, il prendra une dérivation de type *finclitique* ou *verbe* et la transformera en *DP* en attribuant à ce constituant son cas et sa cliticisation temporaire.

$$(\epsilon, qui, \epsilon) : Subj \setminus case \setminus d \setminus d \,/\, finclitique$$
$$(\epsilon, qui, \epsilon) : Subj \setminus case \setminus d \setminus d \,/\, verbe$$

(71) – aime$_{verbe}$ Marie
– $[\overline{Subj}]$ aime$_{verbe}$ Marie
– qui$_{Subj\backslash case\backslash d\backslash d}$ $[\overline{Subj}]$ aime Marie
– qui$_{d\backslash d}$ aime Marie
– Pierre$_{[\overline{case}]}$ qui$_d$ aime Marie
– Pierre qui aime Marie mange$_t$ une pomme.

Grâce à ces trois entrées on peut faire partager les *DP*s quelles que soient leurs positions. Dans la contrepartie sémantique c'est le relatif qui portera la seconde partie de la variable permettant de la construire, on présente ce traitement par le circuit de la figure 51.

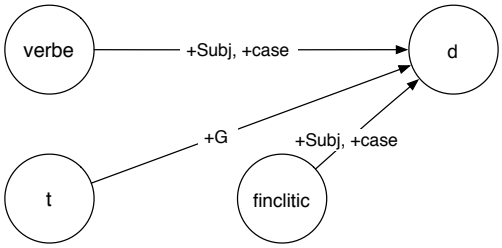


Fig. 51 – Subordonnées relatives.

Cependant, le relatif ne peut laisser partager qu'un seul argument du verbe. Son utilisation doit se substituer uniquement à un argument vide non cliticisé pour éviter des dérivations comme :

(72) – aime$_{verbe}$ $[\overline{F}]$
– $[\overline{Subj}]$ aime$_{verbe}$ $[\overline{F}]$
– qui$_{Subj\backslash case\backslash d\backslash d}$ $[\overline{Subj}]$ aime $[\overline{F}]$
– qui$_{d\backslash d}$ aime $[\overline{F}]$
– Pierre$_{[\overline{case}]}$ qui$_d$ aime $[\overline{F}]$
– Pierre qui aime la mange$_t$ une pomme.

Pour cela, nous devons vérifier qu'il ne reste aucun trait de cliticisation avant de modifier le DP. C'est le même procédé que celui utilisé dans la cliticisation standard qui se produit à la dernière étape, lorsque l'on a atteint le type acceptant. Nous devons l'utiliser à ce moment-là pour vérifier que les séquences ϵ *qui verbe objet* et ϵ *que sujet verbe* ne contiennent plus d'autre trait.

8.5.4 Sémantique des verbes enchâssés

La sémantique des verbes enchâssés est directe. On utilise pour cela des formules qui permettent de partager les variables de la structure argumentale. Comme nous l'avons vu, le verbe à l'infinitif n'a pas de sujet, ou plus exactement ne reçoit pas de variable représentant le référent du sujet. C'est alors la structure du verbe semi-auxiliaire qui permet de faire passer l'un de ses arguments comme variable sujet au verbe à l'infinitif.

Le type de variable partagée entre les deux verbes de la phrase dépend du type de verbe. Les verbes à montée partagent le même sujet :

(73) Il semble la lui donner

Ce sont donc des prédicats qui prennent un sujet et un verbe à l'infinitif et appliquent une variable à ce verbe. On suppose que la variable de réification, représentant l'événement relatif au verbe à l'infinitif est close par le verbe à montée. Il y a une variable e', spécifiée par l'entrée du verbe à montée. Les $\lambda\mu$-DRSs, contreparties semantiques des entrées lexicales, sont donc :

sembler	$\lambda S\lambda v_1\lambda e.(sembler\ (e, v_1, e') \wedge S(v_1, e)) \wedge agent(e', v_1)$
réparer	$\lambda x\lambda y\lambda e.réparer(e, x, y)$

Pour l'analyse de la phrase :

(74) Je semble la réparer.

on utilisera la référence à celui qui parle pour le *Je*, c'est-à-dire *moi*. Le clitique sujet monte par dessus le verbe et est appliqué aux deux verbes grâce à la structure de *sembler*. C'est dans ces situations que l'on utilise la non-linéarité du calcul. Il faut distinguer deux phases dans nos analyses. La première est la résultante du système logique qui nous permet de faire une correspondance entre analyse syntaxique et λ-termes. Elle est due à l'isomorphisme de Curry-Howard et ce calcul est linéaire. À ce calcul nous passons comme variable des $\lambda\mu$-DRS qui, elles, peuvent ne pas être linéaires. C'est cette dernière propriété que nous utilisons ici pour la duplication de variable.

L'analyse syntaxique construit une structure dans laquelle les applications (que nous notons @ afin de simplifier les représentations) sont réalisées dans l'ordre suivant :

$$(je@(inflexion@(sembler@(la(infinitif@réparer)))))$$

Cela nous permet de calculer la sous-formule : *la reparer* :

$$\lambda y\lambda e.réparer(e, x, Y) \wedge patient(e, Y)$$

où Y est une variable liée dans le contexte.

Ce terme est ensuite appliqué au verbe à montée qui fournit le terme :

$$\lambda v_1.sembler(e, v_1) \wedge réparer(e, v_1, Y) \wedge agent(e', v_1) \wedge patient(e', Y)$$

Dans la suite de la dérivation, le verbe principal *sembler* prend son sujet et l'instanciation de cette variable permettra de fournir une même variable à tous les prédicats relatifs à cette dernière.

Le clitique sujet est ainsi monté syntaxiquement par dessus le verbe principal, et sémantiquement la variable du clitique sujet est partagée par les deux verbes.

8.6 Autour du groupe nominal

Nous avons vu quelles étaient les implications d'un ajout du traitement des clitiques dans une GM pour le français autour du groupe verbal. Cependant, ces mécanismes d'analyse peuvent s'étendre à d'autres phénomènes autour du groupe nominal. Nous en énoncerons deux dans cette présentation : la *dislocation* et l'*extraction* d'un *DP*.

8.6.1 Dislocation

En se basant sur l'analyse de la dislocation de [DRDS04], nous distinguons un premier cas : la dislocation à gauche. Nous reviendrons par la suite sur l'autre type de dislocation qui contient la dislocation médiane et à droite. Une dislocation à gauche est la présence en début de phrase d'un *DP* qui est repris par un clitique dans la principale.

(75) Ce type$_i$, Marie le$_i$ voit trop.

Nous sommes en présence d'un phénomène très proche du cas des questions puisqu'un groupe nominal est déplacé en première partie de phrase.

La mise en place de la dislocation dans notre système se base sur l'idée qu'il est possible que les clitiques soient des reprises phonologiques d'arguments phonologiquement non-vides du verbe.

Dans ce cas, nous devons donc pouvoir modifier un *DP* pour qu'il puisse être repris par un clitique et disloqué en fin d'analyse, ce qui se fait par une entrée qui ne modifie pas le type du *DP* mais lui ajoute deux traits, chacun répondant à une des nécessités précitées. Ainsi il devient un *DP* qui doit être repris par un clitique puis disloqué.

$$(\epsilon, \epsilon, \epsilon) : (disloc \otimes (F \otimes d)) \,/\, d$$

Comme cet ajout doit se produire avant que le *DP* soit introduit dans le verbe - au moment où le verbe entrera dans la cliticisation - l'hypothèse F permettra l'introduction de la reprise. La seconde hypothèse *disloc* sera, quant à elle, utilisée par une formule que l'on ajoute à l'entrée *comp* qui déclenche un déplacement fort de tout le *DP*.

$$(\epsilon, \epsilon, \epsilon) : DISLOC \setminus c \,/\, t$$

Le traitement des autres dislocations (médiane et à gauche) se fera par un procédé similaire. Une remarque de [DRDS04] suggère que la dislocation à droite n'est qu'un cas particulier de dislocation médiane. Ainsi, l'argument reste dans sa position dans la phrase, en étant mis *à l'extérieur* par des virgules, avant d'être repris par un clitique.

(76) Marie le$_i$ voit trop, ce type$_i$.

Dans ce cas, le groupe nominal argument du verbe doit être modifié pour devenir un *DP* qui doit être repris et disloqué (ce qui est la même entrée que précédemment) mais la dernière étape *comp* déclenche un déplacement faible et non pas fort, laissant ainsi cet argument dans sa position dans l'analyse.

$$(\epsilon, \epsilon, \epsilon) : disloc \setminus c \,/\, t$$

Si l'argument est le dernier, la dislocation se fera à droite, sinon elle sera médiane.

8.6.2 Extraction d'un groupe nominal

La cliticisation telle que nous l'avons présentée, se charge de reprendre des arguments vides du verbe. Cependant, il est possible de construire des DPs complexes. Et ces *DP*s peuvent eux aussi ne pas posséder leur argument. Dans ce cas, il est possible de le cliticiser.

(77) Pierre en voit la fin.
Pierre voit la fin du film.

La construction des *DP*s complexes se fait par adjonction sur le *DP* principal. Nous introduisons la possibilité de construire *la fin* $\epsilon_{[\overline{en}]}$ où $\epsilon_{[\overline{en}]}$ est l'argument vide de *la fin* et lors du passage dans le cluster de clitique, cet assigné sera utilisé pour être cliticisé en une extraction par un clitique génitif : *Pierre en voit la fin.*

Nous résumons ces deux traitements autour du groupe nominal par le circuit de la figure 52.

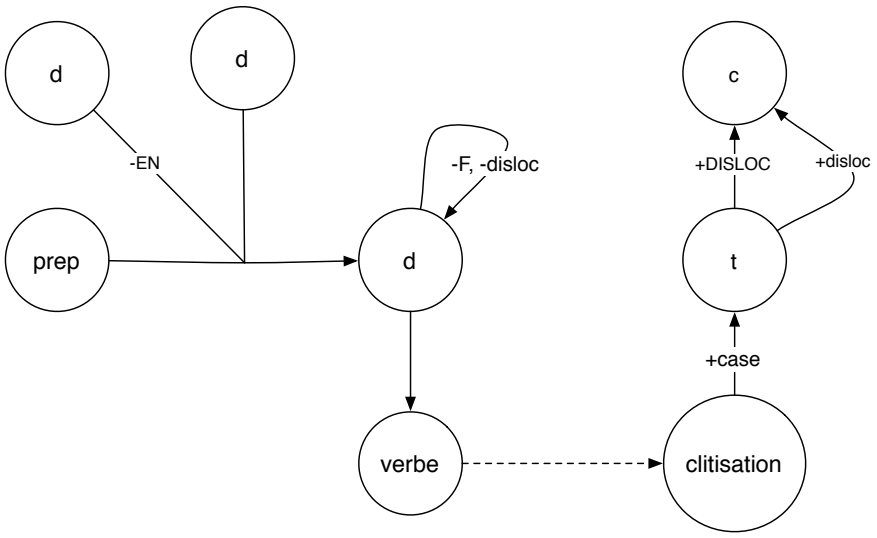


FIG. 52 – Autours des DPs.

8.7 Conclusion

Dans ce chapitre, nous avons présenté des fragments d'une grammaire du français pour les GMCs à travers l'analyse des clitiques. Ces marqueurs atones ont la particularité d'être régis par de nombreuses règles en fonction de leur utilisation.

Une première approche de l'analyse des clitiques a été proposée par Stabler. À partir de celle-ci, nous avons étendu la formalisation à l'ensemble des clitiques du français en introduisant un cluster de cliticisation ainsi que le traitement de la négation et celui des pronoms sujets.

Le traitement proposé en deux parties, l'une phonologiquement vide et l'autre portant la partie phonologique, nous a conduit à étendre la reconnaissance d'autres phénomènes : montée au-dessus des auxiliaires, traitement de l'impératif - à l'inverse du positionnement de la séquence de clitiques sans négation et position standard avec négation. De plus, les clitiques ont la particularité de pouvoir être partagés pour les verbes supports et à contrôle, ce que nous avons modélisé grâce à cette hypothèse dissociant le traitement des clitiques sur deux marqueurs. Nous proposons également un traitement des subordonnées relatives.

Enfin, le même traitement est étendu aux *DP*s, pour lesquels les clitiques ne sont plus des reprises de parties phonologiquement vides. Ceci nous permet de reconnaître les

dislocations et les extractions d'arguments nominaux.

Nous avons également ajouté la contrepartie sémantique pour les clitiques. De la même façon que nous faisions une dissociation entre les deux éléments de la cliticisation pour la forme phonologique, nous le faisons pour la sémantique. Ainsi l'argument phonologiquement vide - ou le premier pour le cas d'arguments non vides - porte la sémantique et le second est sémantiquement vide. La dissociation de la sémantique sur tous les traits des éléments permet d'expliquer la construction de la sémantique globale du clitique dans le calcul de liage des variables comme proposé par Bonato, [Bon06].

Cet ensemble de modélisations forme un fragment d'une grammaire du français. Il manque principalement la modélisation de la coordination (phénomène très complexe faisant l'objet de nombreuses études).

Enfin, cette présentation des clitiques ne met pas en avant les propriétés de la DRT qui peuvent être réutilisées, comme pour l'interprétation des *Donkey sentences*, et elle ne propose pas d'analyse pour les phénomènes étranges qui peuvent apparaître entre les quantificateurs et les clitiques, dans les phrases comme par exemple :

(78) J'en ai vu un.

Je les ai tous vus.

Je la laisse tous les lui donner.

Ces séquences, où quantification et clitiques coopèrent, restent à étudier. Cette présentation soulève aussi une question importante sur la modélisation de l'opération *Head-Movement* qui suggère la nécessité de davantage de finesse dans les analyses à réaliser.

Conclusion

L'interface syntaxe/sémantique pour l'analyse des langues naturelles et ce à partir des GMs était la thématique originale de ces travaux. Les premières versions d'interfaces se sont heurtées à de nombreux problèmes, tant les phénomènes à prendre en considération sont nombreux. Cet état de fait nous a conduit dans un premier temps à axer nos recherches sur la question de l'analyse syntaxique, avant de poursuivre vers la sémantique.

Le thème d'étude étant accessible à tous, puisque chacun possède au moins une langue maternelle, nous avons essayé d'illustrer systématiquement les formalismes utilisés par des exemples aux propriétés différentes.

9.1 Contributions

Dans la perspective de la théorie générative, la proposition des grammaires minimalistes de Stabler a été une première réalisation effective des théories de Chomsky, dont nous avons exposé les arguments principaux dans le chapitre 1. Le programme minimaliste est issu d'une longue tradition linguistique, et fait consensus dans la communauté linguistique.

Nous nous sommes inscrits dans le prolongement de ces travaux. Pour cela, nous avons, au chapitre 2, présenté ces grammaires sous un jour nouveau : les arbres minimalistes y sont décrits par des termes algébriques. Ces définitions fournissent une caractérisation logique du formalisme, afin de prouver des propriétés fréquemment admises, en particulier l'unicité de l'élément tête d'un arbre minimaliste ou l'antisymétrie de la relation de projection. Grâce à ce système calculatoire basé sur un lexique ainsi qu'aux opérations *merge* et *move*, nous pouvons rendre compte de nombreux phénomènes linguistiques. Nous avons présenté en particulier l'analyse d'une question avec inversion du verbe et du sujet, dans la section 2.5.2.

Nous avons étudié les mécanismes internes de ce formalisme. Si l'opération *merge* est intuitivement compréhensible, celle de *move* a des conséquences théoriques plus complexes à définir. Nous avons alors proposé des notions d'équivalence entre GMs (section 3.4.1) permettant de soumettre une normalisation des lexiques. Nous avons utilisé cette dernière pour avancer une modélisation abstraite des lexiques des GMs par des circuits se révélant être un outil performant pour vérifier la cohérence des lexiques lors de leur phase de rédaction (3.5).

De plus, cette modélisation peut être vue comme un filtre pour l'analyse. En effet, les GMs ont la caractéristique d'inclure des entrées à phonologie vide donc n'apparaissant pas de manière évidente dans les données fournies à l'analyse. Le problème de la détermination

du sous-ensemble permettant de réaliser l'analyse d'un énoncé est combinatoirement complexe. Seuls les sous-ensembles formant un arbre couvrant des circuits peuvent fournir une analyse.

Parallèlement, nous avons essayé d'isoler les propriétés relatives à l'opération de déplacement, et ce par l'analyse de dérivations de langages théoriques, qui sont présentées dans le chapitre 4. Nous avons alors étudié les langages de compteurs et leur généralisation, ainsi que les langages de copies. Une généralisation des propriétés issues de ces analyses est proposée dans l'analyse des mots sur une lettre avec en puissance la fonction de Fibonacci. Ces dérivations ont été construites pour comparer les mécanismes mis en œuvre dans les GMs et dans les automates à piles de piles. Ces travaux tendaient à fournir une transcription vers des formalismes ayant des analyseurs plus performants, puis l'identification de conditions sur la rédaction des lexiques (comme peut être vue l'équivalence pour la fusion). Nous n'avons pas poursuivi plus avant cette comparaison, suite à la publication des travaux de Michaelis.

Ces travaux clôturent la partie propre aux grammaires minimalistes.

La suite de nos travaux est en relation directe avec le sujet de départ : l'interface syntaxe/sémantique. Traditionnellement, on emploie des formalismes issus de systèmes logiques pour utiliser l'isomorphisme de Curry-Howard et fournir une représentation, sous forme de formule, des relations entre les éléments d'un énoncé. Nous nous sommes ainsi attachés à proposer un nouveau formalisme basé sur un cadre logique.

Les travaux dans ce sens utilisaient le calcul de Lambek avec produit. Nous avons identifié des analyses pour lesquelles ce formalisme n'était pas adapté à cause de la non-commutativité du connecteur produit. Nous avons donc étudié un cadre logique plus large, possédant des connecteurs commutatifs et non-commutatifs : la logique mixte.

Pour cela, nous nous sommes penchés sur la question de la normalisation des preuves dans ce calcul, afin de vérifier la propriété de la sous-formule qui nous assure la cohérence des preuves. Le chapitre 5 présente les normalisations et les preuves de la propriété de la sous-formule pour la logique mixte ainsi que pour le calcul de Lambek avec produit, qui en est un cas particulier.

Ces formes normales sont obtenues à partir de la notion de *k-redex-étendu* qui sont des redex classiques (succession d'une règle d'introduction et d'une règle d'élimination pour une même formule) contenant d'autres règles. Nous avons démontré plusieurs propriétés sur le contenu de ces redex-étendus.

D'une part, nous avons donné une normalisation unique pour le calcul de Lambek avec produit, pour laquelle les éliminations de produits sont considérées comme étant *le plus haut possible* ; et d'autre part, nous avons proposé une forme normale pour les preuves de la logique mixte. La non-unicité provient de successions d'éliminations de produits commutatifs dans lesquelles il n'y a pas d'ordre préférentiel.

Dans le chapitre 6, nous utilisons un fragment de la logique mixte pour définir des ensembles de règles qui modélisent les opérations *merge* et *move*. La logique mixte envisage les preuves sous l'angle de la consommation des ressources. Une preuve va alors associer des

formules entre elles, tout en introduisant des hypothèses, et, à certains moments définis, les ressources sont consommées en étant remplacées par d'autres preuves. Ce fragment n'utilise que des règles d'élimination et nous appelons grammaire minimaliste catégorielle ce formalisme.

Nous utilisons la non-commutativité de l'élimination d'implication pour conserver un ordre gauche/droite entre les éléments, ce qui nous permet de modéliser *merge*. D'autre part, dans ce formalisme, les déplacements ne sont pas effectifs. En cela, la dérivation accumule des hypothèses où chacune représente soit une catégorie de base relative à l'élément dans la dérivation, soit un trait de cet élément. Lorsque toutes les hypothèses relatives à un élément devant se déplacer sont présentes, la dérivation les consomme pour prendre sa place, ce qui modélise l'opération *move*.

Enfin, nous montrons l'inclusion des langages engendrés par les GMs dans les langages engendrés par les GMCs. Puis nous terminons par des exemples de dérivation.

Une fois ce nouveau formalisme mis en place, nous avançons une interface syntaxe/sémantique utilisant le cadre logique portant les GMCs (au cours du chapitre 7). Pour cela, nous avons identifié un point où il faut dissocier les calculs. Nous devons pouvoir lier une variable d'une formule sans réaliser l'application fonctionnelle sous-jacente. Cette distinction permet d'associer une interprétation sémantique à l'opération de déplacement : un constituant est d'abord introduit dans la dérivation (liage de la variable en structure argumentale) puis il est déplacé (il a donc pris toutes ses positions et il peut alors réaliser pleinement sa sémantique). Cette dissociation est rendue possible par l'opération de substitution des GMCs (élimination du produit).

Dans cette interface, nous avons à la fois pris en considération les problèmes de portée des quantificateurs qui sont modélisés par le $\lambda\mu$-calcul et montré une modélisation des principes des UTAHs. Le $\lambda\mu$-calcul est une extension du λ-calcul à la logique classique, qui permet de modéliser les continuations. Ainsi, les prédicats associés au *DP* ne prennent part à la formule que lorsque celle-ci est close. Dans le calcul sémantique, on associe à chaque trait syntaxique une contrepartie sémantique. Ainsi, la donation de cas devient la prise de rôle thématique. À cet effet, on introduit des prédicats à deux arguments pour chacun des rôles. Le problème réside alors dans le lien entre une variable de ce prédicat et celle du *DP* qui lui est associé, ce que réalise l'opération de déplacement. Cependant, l'ordre d'introduction des prédicats dans la formule positionne ces prédicats de rôle en dehors de la portée de leur quantificateur. On utilise ainsi une modélisation de nos formules inspirée de la DRT, à laquelle on ajoute une phase d'internalisation des prédicats.

Lorsque le calcul syntaxique est terminé, nous obtenons une $\lambda\mu$-DRS pour laquelle nous devons internaliser les prédicats dans la portée de leur quantification, puis résoudre les $\lambda\mu$-réductions. C'est la partie propre au calcul sémantique qui se déroule après le *spell-out*.

Le dernier chapitre présente un fragment de grammaire pour le français. L'axe d'étude a été la modélisation des clitiques qui ont un comportement syntaxique spécifique et apparaissent dans de nombreux phénomènes syntaxiques. Nous proposons une modélisation de l'ensemble des clitiques en français pour une majorité de leurs utilisations. On dissocie ce traitement autour du groupe verbal puis du groupe nominal. Ce chapitre permet de présenter l'interface syntaxe-sémantique en œuvre.

Les thématiques abordées lors de ces travaux sont nombreuses et diverses. Elles s'articulent autour des questions de modélisations et de langages formels, ou de questions plus linguistiques. Nous avons tenté de présenter les enjeux dans les deux voies pour chacune des questions abordées, bien que les outils soient radicalement différents.

9.2 Perspectives

Nous commençons par reprendre les perspectives directement issues de ce manuscrit, puis ouvrons notre travail vers deux voies différentes.

L'*interface syntaxe-sémantique* reste encore à développer, et ce avec deux enjeux majeurs. Nous devons d'une part approfondir la couverture sémantique et, d'autre part, mieux prendre en considération le module de sémantique lui-même. Actuellement, nous fournissons une formule rendant compte de certaines propriétés qui nous semblent primordiales, mais nous n'avons pas d'autres expertises sur les propriétés dont la représentation devrait mieux rendre compte. Ces problématiques appartiennent au domaine de la sémantique. La prise en compte des propriétés de la DRT dans la formalisation de l'interface syntaxe/sémantique est pour nous une avancée dans ce sens.

D'autre part, nous souhaitons approfondir l'analyse des mécanismes mis en œuvre dans l'interface syntaxe/sémantique par la comparaison avec d'autres approches, par exemple celle de Kobele qui, pour les GMs, utilise le *Cooper Storage*. Ce mécanisme est, a priori, équivalent à l'utilisation du $\lambda\mu$-calcul. Cette partie est en cours d'étude.

Pour les questions relatives à l'interface syntaxe/sémantique, l'étape cruciale est le passage à l'implémentation effective. Actuellement, nous disposons d'un module interfacé à l'analyseur existant pour les GMs, mais qui utilise uniquement le λ-calcul (présenté dans l'annexe B). Nous serons donc amenés à proposer une bibliothèque pour le $\lambda\mu$-calcul dans un premier temps, puis à utiliser un analyseur pour les GMCs.

Une perspective d'étude théorique issue de la normalisation de la logique mixte reste à développer. Nous avons vu que la normalisation dans ce calcul n'est pas unique, mais la décomposition des règles d'entropie permettrait d'ajouter des contraintes sur l'ordre des éliminations de produits commutatifs. Cette propriété assurerait la confluence du calcul. On rappelle que le fragment utilisé par les GMs possède, lui, cette propriété. De plus, le lien entre réseau de démonstration et logique mixte reste encore à explorer.

Les derniers travaux autour du programme minimaliste de Chomsky ont introduit la notion de *phases* dans le calcul syntaxique, [Cho99] [che06]. Dans cette proposition, une dérivation se décompose en plusieurs étapes linguistiquement cohérentes. À la fin d'une phase, on distingue son résultat des informations portées à sa périphérie. Cette notion de phase n'est pas prise en compte dans les GMs, bien que plusieurs points d'insertion se présentent.

Les phases sont généralement la modification d'état du verbe. L'utilisation d'un *Head-Movement* ou de la fusion pour les gérer permet de distinguer ces moments particuliers. La périphérie est alors composée des éléments en relation de spécifieur. Utiliser un vocabulaire spécifique ne nous a pas semblé suffisant pour être intégré dans les GMs. Cependant, les travaux sur les GMCs et sur l'interface syntaxe-sémantique nous posent deux problèmes différents pour lesquels la notion de phase pourrait être une réponse partielle.

L'ambiguïté de terminologie sur l'opération de *Head-Movement* nous semble plus profonde que la proposition faite dans les GMs et sa contrepartie dans les GMCs. Dans ce cas, il y aurait un second type de *déplacement* dans les analyses, qui concernerait non pas les arguments d'une tête mais les têtes elles-mêmes. Pour les GMs, il est assez complexe d'envisager l'introduction d'une nouvelle règle qui remettrait en cause tout le système calculatoire. Par contre, étant basées sur la logique mixte, les GMCs peuvent être enrichies de nouvelles règles utilisant des relations différentes. En cela, l'opération de *Head-Movement* pour les GMCs se rapprocherait davantage de l'utilisation du produit commutatif. Cependant, la gestion des ressources, du point de vue des déplacements classiques, et de celui des *Head-Movements* doit être approfondie, mais son intégration serait alors une manifestation de la notion de phase.

Baser les phases sur l'utilisation d'une élimination de produit entraîne une complexification de la gestion des formes phonologiques. Cette perspective de modélisation n'a pas été présentée dans ce manuscrit, car l'analyse des conséquences est un travail en cours.

Pour l'interface syntaxe/sémantique et le calcul de la portée des quantificateurs, nous n'avons présenté que des cas où le nombre de ces quantificateurs n'était pas très élevé. Ont été définis des *island*s, desquels les quantificateurs ne peuvent pas sortir pour prendre leur portée. Nous pensons que ces *island*s correspondent principalement aux phases d'une analyse. Ainsi, la modélisation des phases permettrait, dans l'interface syntaxe/sémantique, de déclencher la résolution d'une partie des μ-réductions avant la fin du calcul, tout en laissant un quantificateur principal non résolu, car dans la périphérie de la phase.

L'introduction de ces μ-réductions au cours du calcul permettrait simplement de diminuer le nombre de portées de quantifications possibles et impossibles. Actuellement, le système proposé se contente de fournir toutes celles qui sont possibles dans un énoncé. La prise en compte des phases semble être un point important pour la modélisation syntaxique et pour ses conséquences sur l'interface elle-même.

La *couverture de la grammaire* du français proposée reste limitée. Actuellement, il n'est pas réaliste de l'utiliser pour des analyses sur de grands corpus. Les problèmes liés au passage à l'échelle sont de deux ordres. Le premier est la validation linguistique des analyses obtenues et le faible nombre de personnes travaillant sur cette question en est la raison principale. Le fragment de grammaire proposé ici tendrait à être une première étape d'une dynamique plus large.

Le second problème provient des analyseurs utilisés. Les solutions actuelles restent beaucoup moins performantes que d'autres analyseurs existants. L'implémentation de filtres sur l'extraction de sélection des lexiques est une première étape. Or, comme nous l'avons évoqué, de nombreux cas ne doivent pas être testés car ils ne peuvent jamais aboutir à une analyse acceptante. L'enjeu est donc de proposer des filtres d'analyse ainsi que des conditions sur la rédaction des lexiques, comme nous avons essayé de le faire dans les premiers chapitres. Cette perspective reste toujours ouverte.

Sur les questions de modélisation, nous travaillons sur la modélisation de la *langue des signes française* avec Émile Voisin. Cette langue, qui a la particularité d'être pratiquée, et non écrite, nous interroge car elle n'utilise pas les référentiels supposés pour les autres

langues naturelles. Pour autant, son statut de langue naturelle peut difficilement être remis en cause, par exemple on observe l'apparition de système de communication gestuelle dans des situations très diverses pour palier à un problème d'oral. On essaie ainsi d'associer du sens non pas à un son mais à un geste. Dans cette langue, la linéarité entre les éléments (gauche-droite en français) n'est plus de mise.

Une structure normative semble cependant être dégagée, structure grâce à laquelle, nous pouvons proposer des lexiques de GMs fournissant des analyses. Pour transformer le résultat en séquence signée, nous avons introduit la notion de signème qui reprend l'ensemble des informations nécessaires pour réaliser le signe. Par exemple, certains verbes ont besoin de connaître leur objet pour être signés : c'est le cas de *manger* qui se fera en fonction de la forme de ce qui est mangé (pour une pomme avec la proforme de l'objet pomme). Mais la manière dont l'élément est mangé peut prévaloir sur l'objet (pour *picorer* par exemple). Dans ce cas pour réaliser le signe de *manger*, une machinerie connaissant les proformes (forme prise par la main pour représenter l'objet) des arguments du verbe, transformera la séquence en signe.

Nous utilisons dans ce cas la notion de trace qui est implicite au déplacement. Ainsi l'ensemble des positions prises par un élément peut être utilisé pour construire la séquence signée.

Bibliographie

[Abe93] Anne Abeillé. *Les nouvelles syntaxes : Grammaires d'unification et analyse du Français*. Armand Colin, Paris, 1993.

[AB00] Anne Abeillé et Philippe Blache. Grammaires et analyseurs syntaxiques. *Traité IC2*, vol. Ingénierie des langues, 2000.

[ADMdS04] Anne Abeillé, Jenny Doetjes, Arie Molendijk et Henrïette de Swart. Adverbs and quantification. dans Henrïette de Swart Francis Corblin, Èditeur, *Handbook of French Semantic*, pages 185–209. Standford, CSLI, 2004.

[Abr93] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, vol. 111 : 3–57, 1993.

[Abr91] V. Michele Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *The Journal of Symbolic Logic*, vol. 56, n° 4 : 1403–1451, December 1991.

[AR99] V. Michele Abrusci et Paul Ruet. Non-commutative logic I : The multiplicative fragment. *Annals of pure and applied logic*, vol. 101, n° 1 : 29–64, 1999.

[Ajd35a] K. Ajduckiewicz. Die syntaktische konnexität. *Studia Philosophica*, pages 1–27, 1935.

[Ajd35b] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, vol. 1 : 1–27, 1935. (pages 207–231).

[Alb02] Daniel M. Albro. An earley-style parser for multiple context-free parser. unpublished manuscript, 2002.

[Amb03] Maxime Amblard. Représentations sémantiques pour les grammaires minimalistes. Master's thesis, Universié de Bordeaux 1, 2003.

[Amb05a] Maxime Amblard. Counting dependencies and minimalist grammars. *student session of LACL05*, 2005.

[Amb05b] Maxime Amblard. Représentation sémantique pour les grammaires minimalistes. Rapport technique n° 5360, INRIA, 2005.

[Amb05c] Maxime Amblard. Synchronisation syntaxe sémantique, des grammaires minimalistes catégorielles aux constraint languages for lambda structures. *RECITAL*, 2005.

[Amb06] Maxime Amblard. Treating clitics with minimalist grammars. *Formal grammar - Malaga*, 2006.

[Amb07] Maxime Amblard. *Calcul de représentations sémantiques et suntaxe générative : les grammaires minimalistes catégorielles*. PhD thesis, université de Bordeaux 1, 2007.

[AAL06] Maxime Amblard, Houda Anoun et Alain Lecomte. Ellipse et coordination dans les grammaires de type logique. *Journée Sémantique et Modélisation*, 2006.

[ALR03] Maxime Amblard, Alain Lecomte et Christian Retoré. Syntax and semantics interacting in a minimalist theory. *Prospect and advance in the Syntax/Semantic interface*, 2003.

[ALR04] Maxime Amblard, Alain Lecomte et Christian Retoré. Synchronization syntax semantic for a minimalism theory. *Journée Sémantique et Modélisation*, 2004.

[AR07] Maxime Amblard et Christian Retore. Natural deduction and normalisation for partially commutative linear logic and lambek calculus with product. *Computation and Logic in the Real World, CiE 2007*, vol. Quaderni del Dipartimento di Scienze Matematiche e Informatiche "Roberto Magari", june 2007.

[Ano06] Houda Anoun. Logical grammar with emptiness. *Formal grammar*, 2006.

[Bak97] M. Baker. Thematic Roles and Syntactic Structure. dans L. Haegeman, Èditeur, *Elements of Grammar, Handbook of Generative Syntax*, pages 73–137. Kluwer, Dordrecht, 1997.

[Bak88] M. C. Baker. *Incorporation : a theory of grammatical function changing*. University of Chicago Press, 1988.

[Bak01] Mark Baker. *The Atoms of Language*. Basic Books, New York, 2001.

[BH53] Yehoshua Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, vol. 29 : 47–58, 1953.

[BHGS63] Yehoshua Bar-Hillel, Chaim Gaifman et Eli Shamir. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, vol. F, n° 9 : 1–16, 1963.

[Bar04] Chris Barker. Continuation in natural language. *proceedings of the Fourth ACM SIGPLAN Continuation Workshop (CW'04)*, 2004.

[BdGR97] Denis Bechet, Philippe de Groote et Christian Retoré. A complete axiomatisation of the inclusion of series-parallel partial orders. dans H. Comon, Èditeur, *Rewriting Techniques and Applications, RTA'97*, volume 1232 of *LNCS*, pages 230–240. Springer Verlag, 1997.

[BM] R. Bernardi et M. Moortgat. Continuation semantics for symmetric categorial grammar. to appear.

[BE96] Robert Berwick et Samuel Epstein. On the convergence of 'minimalist' syntax and categorial grammars, 1996.

[BB01] Philippe Blache et Jean-Marie Balfourier. Property grammars : a flexible constraint-based approach to parsing. dans *IWPT*. Tsinghua University Press, 2001.

[BHar] Cedric Boeckx et Norbert Hornstein. *Universals in light of the varying aims of linguistic theory*. Oxford University Press, to appear.

[BG01] Olivier Bonami et Danièle Godard. Inversion du sujet, constituance des mots. *cahier Jean-Claude Milner*, 2001.

[BGKM04] Olivier Bonami, Danièle Godard et Brigitte Kampers-Manhe. Adverbs classification. dans Henrïette de Swart Francis Corblin, Èditeur, *Handbook of French Semantic*, pages 143 – 184. Standford, CSLI, 2004.

[Bon05] Roberto Bonato. Towards a computational treatment of binding theory. *Logical Aspect of Computational Linguistic*, pages 35–50, 2005.

[Bon06] Roberto Bonato. *An Integrated Computational Approach to Binding Theory*. PhD thesis, University of Verona, 2006.

[BM76] JA Bondy et USR Murphy. *Graph Theory with Application*. MacMillan press, 1976.

[BDP99] D. Bouchard, C. Dubuisson et A-M. Parisot. Les facteurs articulatoires qui détermine l'ordre en lsq. *Actes de l'ACL*, vol. 1, 1999.

[Bre82] Joan Bresnan. *The mental representation of grammatical relations*. MIT Press, Cambridge, MA, 1982.

[cdl07] cahier de l'Herne. Chomsky, 2007.

[Car96] Bob Carpenter. *Lectures on Type-Logical Semantics*. MIT Press, Cambridge, Massachussetts and London, England, 1996.

[CMP01] Emmanuel Chailloux, Pascal Manoury et Bruno Pagano. *Développement d'applications avec Ocaml*. O'Reilly, 2001.

[che06] Cristiano chesi. *Phases and Cartography in Linguistic Computation : toward a Cognitively Motivated Computational Model of Linguistic Competence*. PhD thesis, MIT, 2006.

[Cho55] Noam Chomsky. *The Logical Structure of Linguistic Theory*. Plenum, New York, 1955. (1975).

[Cho57] Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

[Cho63] Noam Chomsky. Formal properties of grammars. dans *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass., 1965.

[Cho73] Noam Chomsky. Conditions on transformations. dans S.R. Anderson & P. Kiparsky, Èditeur, *A Festschrift for Morris Halle*, pages 232–286. Holt Rinehart and Winston, 1973.

[Cho81] Noam Chomsky. *Lectures on Government and Binding*. Foris Pub., Dordrecht, 1981.

[Cho95] Noam Chomsky. *The Minimalist Program*. MIT Press, Cambridge, 1995.

[Cho98] Noam Chomsky. *Minimalist Enquiries : the framework*. MIT, draft,, 1998.

[Cho99] Noam Chomsky. *Derivation by phase*. ms, MIT, 1999.

[Chu40] A. Church. A simple theory of types. *Journal of Symbolic Logic*, vol. 5 : 56–68, 1940.

[Chu06] Sarah Churng. Synchronizing modalities : A model for synchronization of gesture and speech as evidenced by american sign language. dans Cascadilla Proceedings Project, Èditeur, *Proceedings of the 25th West Coast Conference on Formal Linguistics*, 2006.

[CS70] John Cocke et JT. Schwartz. Programming languages and their compilers. Rapport technique, Courant Institute of Mathematical Sciences, 1970.

[Coh06] Cyril Cohen. From mg to rtg. Master's thesis, université de Bordeaux 1 et ENS Cachan., 2006.

[col] collectif. *Dictionnaire international des termes Littéraires*.

[CKPR72] Alain Colmerauer, Henry Kanoui, Robert Pasero et Philippe Roussel. Un système de communication en français. Rapport technique, Groupe Intelligence Artificelle, Faculté des Sciences de Luminy, Université Aix-Marseille II, France, October 1972.

[Com95] B. Comrie. *Language universals and linguistic typology : syntax and morphology*. Blackwell, Oxford, 1995.

[Cor02] Francis Corblin. *Représentation du discours et sémantique formelle. Introduction et applications au francais.* Linguistique Nouvelle. Presses Universitaires de France (PUF), 2002.

[CVDT04] Francis Corblin, Henr ïette de Swart Viviane Déprez et Lucia Tovena. Negative concord. dans F. Corblin et H. de Swart, Èditeurs, *Handbook of French Semantic*, pages 417–452. Standford, CSLI, 2004.

[Cor04] Thomas Cornell. Lambek calculus for transformational grammars. *Journal of Research on Language and Computation*, vol. 2(1) : 105–126, 2004.

[Cre04] Denis Creissels. *Formes verbales non finies et complémentation*, 2004.

[Cux00] C. Cuxac. *La Langue des Signes Française, les voies de l'iconicité*. Ophrys, Le Mans, 2000.

[Dag96] Anne Dagnac. Interprétation du sujet des infinitives prépositionnels : le cas de pour causal. dans *Rencontre Internationale des Jeunes Linguistes*, 1996.

[Dag02] Anne Dagnac. L'interprétation de PRO dans les infinitives circonstancielles. *Le sujet*, 2002.

[dG94] Philippe de Groote. On the relation between lambda-mu-calculus and the syntactic theory of sequential control. *LPAR'94*, vol. 822 : 31–43, 1994.

[dG96] Philippe de Groote. Partially commutative linear logic : sequent calculus and phase semantics. dans Vito Michele Abrusci et Claudia Casadio, Èditeurs, *Third Roma Workshop : Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*, pages 199–208. Bologna :CLUEB, 1996.

[dG98] Philippe de Groote. An environement machine for the $\lambda\mu$-calculus. *Math. Str. in Computer Science*, vol. 8 : 637–669, 1998.

[dG01] Philippe de Groote. Type raising, continuations, and classical logic. *Thirteenth Amsterdam Colloquium*, pages 97–101, 2001.

[dGR96] Philippe de Groote et Christian Retoré. Semantic readings of proof nets. dans Geert-Jan Kruijff, Glyn Morrill et Dick Oehrle, Èditeurs, *Formal Grammar*, pages 57–70, Prague, 1996. FoLLI.

[DRDS04] Elisabeth Delais-Roussarie, Jenny Doetjes et Petra Sleeman. Dislocation. dans F. Corblin et H. de Swart, Èditeurs, *Handbook of French Semantic*, pages 501–528. Standford, CSLI, 2004.

[Dow01] Gilles Dowek. *Higher-Order Unufication and Matching.* Elsevier Science Publishers, Handbook of automated reasoning, 2001.

[EKNar] Markus Egg, Alexander Koller et Joachim Niehren. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, pages 457–485, To appear.

[FS06] Fratani et Senizergues. Iterated pushdown automata and sequences of rational numbers. *ANNALSPAL : Annals of Pure and Applied Logic*, vol. 141, 2006.

[FG02] W. Frey et Hans-Martin Gaërtner. On the treatment of scrambling and adjunction in minimalist grammars. *Formal Grammar*, pages 41–52, 2002.

[Gam91a] L. T. F. Gamut. *Logic, Language and Meaning*, volume 2. The University of Chicago Press, 1991.

[Gam91b] L. T. F. Gamut. *Logic, Language and Meaning – Volume 2 : Intensional logic and logical grammar.* The University of Chicago Press, 1991.

[GS97] Ferenc Gécseg et Magnus Steinby. Tree languages. dans *Handbook of Formal Languages* [RS97], chapter 1.

[Gen34a] Gehrard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, vol. 39 : 176–210, 1934. Traduction Française de R. Feys et J. Ladrière : Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[Gen34b] Gehrard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, vol. 39 : 405–431, 1934. Traduction française de J. Ladrière et R. Feys : Recherches sur la déduction logique, Presses Universitaires de France, Paris, 1955.

[Gen36] Gerhard Gentzen. Die Widerspruchsfreiheit der reine Zahlentheorie. *Mathematische Annalen*, vol. 112 : 493–565, 1936.

[GK06] Kim Gerdes et Sylvain Kahane. A polynomial parsing algorithm for the topological model : Synchronizing constituent and dependency grammars, illustrated by german word order phenomena. dans *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1097–1104, Sydney, Australia, July 2006. Association for Computational Linguistics.

[GS97] Ferenc Géseg et Magnus Steinby. Tree languages. *Handbook of Formal Language*, vol. 3, 1997.

[Gir87a] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, vol. 50, n° 1 : 1–102, 1987.

[Gir87b] Jean-Yves Girard. *Proof Theory and Logical Complexity*. Bibliopolis, Naples, 1987.

[Gir97] Jean-Yves Girard. Du pourquoi au comment : la théorie de la démonstration de 1950 à nos jours. unpublished manuscript, 1997.

[GLT88] Jean-Yves Girard, Yves Lafont et Paul Taylor. *Proofs and Types*. N° 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[God04] Danièle Godard. French negative dependency. dans F. Corblin et H. de Swart, Èditeurs, *Handbook of French Semantic*, pages 351–389. Standford, CSLI, 2004.

[Gre63] J. H. Greenberg. Some universal of grammar with particular reference to the order of meaningful elements. *Universal language : report of a conference Held at Dobbs Ferry, New-York april 13-15*, vol. 1, 1963.

[Gré75] Grévisse. *Le français correct*. grevisse, 1975.

[Gre01] Günther Grewendorf. Multiple wh-fronting. *Linguistic Inquiry*, 2001.

[Gug02] Alessio Guglielmi. A system of interaction and structure, September 25 2002.

[Gui04] Matthieu Guillaumin. *Conversions between Midly Context Sensitive Grammars*, 2004.

[Hal03] John Hale. *Grammar, Uncerainy and Sentence Processing*. PhD thesis, John Hopkins univrsity, 2003.

[Har01] Henk Harkema. A characterization of minimalist languages. *Logical Aspect of Computational Linguistics*, vol. 2099 Springer-Verlag, 2001.

[HK98] Irene Heim et Angelika Kratzer. *Semantics in Generative Grammar*. Blackwell, 1998.

[How80] William A. Howard. The formulae-as-types notion of construction. dans J. Hindley et J. Seldin, Èditeurs, *To H.B. Curry : Essays on Combinatory Logic, λ-calculus and Formalism*, pages 479–490. Academic Press, 1980.

[Hue75] Gérard Huet. A unification algorithm for typed λ-calculus. *Journal of Theoretical Computer Science*, pages 27–57, 1975.

[Hue02a] Gérard Huet. Constructive computation theory. note de cours de DEA, université de Bordeaux 1, 2002.

[Hue02b] Gérard Huet. The zen computational linguistics toolkit. *ESSLLI 2002 Lecture*, 2002.

[Hue97] Gérard P. Huet. The zipper. *J. Funct. Program*, vol. 7, n° 5 : 549–554, 1997.

[Jac95] Ray Jackendoff. *The Architecture of the Language Faculty*. M.I.T., 1995.

[Jac02] Ray Jackendoff. *Foundations of Language*. Oxford University Press, 2002.

[Jan82] Theo M.V. Janssen. Compositional semantic and relative clause formation in montague grammar. *MCTRACT*, 82.

[JJ79] J.E. Hopcroft et J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[JLT75] Joshi, Levy et Takahashi. Tree adjunct grammars. *JCSS : Journal of Computer and System Sciences*, vol. 10, 1975.

[Jos85] Aravind K. Joshi. Tree adjoining grammars : How context-sensitivity is required to provide reasonable structural description ? *Natural Language Parsing, Psychological, Computational, and Theoritical Perspectives*, pages 206–250, 1985.

[JS97] Aravind K. Joshi et Yves Shabes. Tree-adjoining grammars. *Handbook of Formal Language*, vol. 3, 1997.

[Kah06a] Sylvain Kahane. Polarized unification grammars. dans *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Sydney, Australia, July 2006. Association for Computational Linguistics.

[Kah06b] Sylvain Kahane. Polarized unification grammars. dans *ACL*. The Association for Computer Linguistics, 2006.

[KR93] H. Kamp et U. Reyle. *From Discourse to Logic.* Kluwer Academic Publishers, Dordrecht, 1993.

[KMMD+04] Brigitte Kampers-Manhe, Jean-Marie Marandin, Franck Drijkoningen, Jenny Doetjes et Aafke Hulk. Subject np inversion. dans F. Corblin et H. de Swart, Èditeurs, *Handbook of French Semantic*, pages 553–579. Standford, CSLI, 2004.

[KB82a] Ronald Kaplan et Joan Bresnan. Lexical functional grammar : a formal system for grammatical representation. dans Joan Bresnan, Èditeur, *The mental representation of grammatical relations.* M.I.T. Press, Cambridge, MA, 1982.

[KB82b] Ronald M. Kaplan et Joan Bresnan. Lexical-Functional Grammar : A formal system for grammatical representation. dans Joan Bresnan, Èditeur, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA, 1982. Reprinted in Mary Dalrymple, Ronald M. Kaplan, John Maxwell, and Annie Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 29–130. Stanford : CSLI Publications. 1995.

[Kas65] T. Kasami. An efficient recognition and syntax-analysis algorithm for context free languages. Rapport technique, Air Force Cambridge Research Lab, 1965.

[Kay75] Richard S Kayne. *French syntax. The transformational Cycle.* MIT press, 1975.

[Kay98] Richard S Kayne. Overt vs covert movment. *Syntax 1,2*, pages 128–191, 1998.

[Kob06] Gregory Kobele. *Generating Copies : An Investigation into Structural Identity in Language and Grammar.* PhD thesis, University of California, Los Angeles, 2006.

[KBW04] Alexander Koller, Aljoscha Burchardt et Stephan Walter. Computational semantics. *ESSLLI*, 2004.

[KS00] H. Koopman et A. Szabolcsi. *A verbal Complex*. MIT Press, Cambridge, 2000.

[Kra94] A. Kratzer. External arguments. dans E. Benedicto et J. Runner, Èditeurs, *Functional Projections*. University of Massachussets, Occasional Papers, Amherst, 1994.

[Kri90] Jean-Louis Krivine. *Lambda Calcul — Types et Modèles*. Etudes et Recherches en Informatique. Masson, Paris, 1990.

[Lam58] Joachim Lambek. The mathematics of sentence structures. *American mathematical monthly*, pages 154–170, 1958.

[Lam61] Joachim Lambek. How to program an (infinite) abacus. *Canadian Mathematical Bulletin*, vol. 4 : 295–302, 1961.

[Lau03] Olivier Laurent. polarized proof-nets and $\lambda\mu$-calculus. *Theoritical Computer Science*, vol. TCS, n° 290 : 161–188, janvier 2003.

[Laz94] G. Lazard. *L'actance*. Presses Universitaires de France, 1994.

[Leb91] David Lebeaux. Relative clause, licensing, and the nature of the derivation. *Syntax and Semantics*, 1991.

[Lec02] Alain Lecomte. Rebuilding mp on a logical ground. *Research on Language and Computation*, 2002.

[Lec04a] Alain Lecomte. Derivations as proof : a logical approach to minimalism. *Catgorial Grammar*, 2004.

[Lec04b] Alain Lecomte. Rebuilding the minimalist program on a logical ground. *Journal of Research on Language and Computation*, vol. 2(1) : 27–55, 2004.

[Lec05] Alain Lecomte. Categorial grammar for minimalism. *Language and Grammar : Studies in Mathematical Linguistics and Natural Language*, vol. CSLI Lecture Notes, n° 168 : 163–188, 2005.

[LNA05] Alain Lecomte et Aresky Nait-Abdallah. On expressing vague quantification and scalar implicatures in the logic of partial information. *Logical Aspect of Computational Linguistic*, 2005.

[LR99] Alain Lecomte et Christian Retoré. Towards a logic for minimalist. *Formal Grammar*, 1999.

[LR01] Alain Lecomte et Christian Retoré. Extending Lambek grammars : a logical account of minimalist grammars. dans *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL 2001*, pages 354–361, Toulouse, July 2001. ACL.

[LR02] Alain Lecomte et Christian Retoré. Bi-grammars, a logical system for syntax, semantic and their correspondence. *Formal grammar*, 2002.

[LC] Jean-Jacques Levy et Robert Cori. *Algorithmes et Programmation*. Ecole Polytechnique.

[Lju05] Peter Ljunglöf. A polynomial time extension of parallel multiplis context-free grammar. *LACL*, 2005.

[Mar84] A. Marantz. *On The Nature of Grammatical Relations.* Phd thesis, MIT, 1984.

[MP87] Igor Melćuk et Alain Polguere. A formal lexicon in meaning-text theory (or how to do lexica with words). *Computational Linguistics*, vol. 13, n° 3-4 : 261–275, 1987.

[Mer06] Bruno Mery. Grammaires légèrement contextuelles pour l'analyse syntaxique du langage naturel. Master's thesis, université de Bordeaux 1, 2006.

[Mic98] Jens Michaelis. Derivational minimalism is midly context-sensitive. *Logical Aspect of Computational Linguistics*, vol. Springer-Verlag, n° 2014 : 179–198, 1998.

[Mic05] Jens Michaelis. An additional observation on strict derivational minimalism. *Formal Grammar*, 2005.

[MG03] Jens Michaelis et Hans-Martin Gaërtner. A note on countercyclicity and minimalist grammars. *Formal Grammar*, pages 103–114, 2003.

[MG05] Jens Michaelis et Hans-Martin Gaërtner. A note on the complexity of constraint interaction : Locality conditions and minimalist grammars. *Logical Aspect of Computational Linguistics*, vol. Springer-Verlag, n° 3492 : 114–130, 2005.

[MK05] Jens Michaelis et Gregory Kobele. Two type 0-variant of minimalist grammars. *Formal Grammar*, 2005.

[MMM00] Jens Michaelis, Uwe Mönnich et Franck Morawietz. Algebraic description of derivational minimalism. *Algebraic Methods in Language Processing*, pages 125–141, 2000.

[MM03] Philip Miller et Paola Monachesi. Les pronoms clitiques dans les langues romanes. *Langues Romanes, problèmes de la phrase simple*, 2003.

[Mil92] Philip H. Miller. *Clitics and Constituents in Phrase Structure Grammar.* PhD thesis, university of Utrecht, 1992.

[MMK01] Uwe Mönnich, Franck Morawietz et Stephen Kepser. A regular query for context-sensitive relations. *IRCS Workshop Linguistic Database*, pages 187–195, 2001.

[Mon73] Richard Montague. The proper treatment of quantification in ordinary english. *Approaches to natural language : proceedings of the 1970 Stanford workshop on Grammar and Semantics*, 1973.

[Moo88] M. Moortgat. *Categorial Investigations, Logical and Linguistic Aspects of the Lambek Calculus.* Foris, Dordrecht, 1988.

[Moo96a] M. Moortgat. In situ binding : a modal analysis. dans P. Dekker et M. Stokhof, Èditeurs, *Proceedings Tenth Amsterdam Colloquium.* ILLC, Amsterdam, 1996.

[Moo96b] M. Moortgat. Multimodal linguistic inference. *JoLLI*, vol. 5 : 349–385, 1996.

[Moo97] M. Moortgat. Categorial type logics. dans J. van Benthem et A. ter Meulen, Èditeurs, *Handbook of Logic and Language*, chapter 2, pages 93–178. Elsevier, 1997.

[Moo99] M. Moortgat. Constants of grammatical reasoning. dans Kruijff Bouma, Hinrichs, Èditeur, *Constraints and Resources in Natural Language Syntax and Semantics*, Studies in Logic, Language and Information. CSLI Publications, Stanford, 1999.

[Moo96] Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, vol. 5, n° 3/4 : 349–385, 1996.

[Moo02a] Michael Moortgat. Categorial grammar and formal semantic. *Encyclopedia of Cognitive Science*, 2002.

[Moo02b] Richard Moot. *Proof Nets for Linguistic Analysis.* PhD thesis, Utrecht university, 2002.

[MR05] Richard Moot et Christian Retoré. Les indices pronominaux du français dans les grammaires catégorielles. *Submitted to Linguisticae Investigationes*, 2005.

[Mul02] C. Muller. *Les bases de la syntaxe : syntaxe contrastive, français-langues voisines.* Presses Universitaires de Bordeaux, 2002.

[NSSW06] Marc Namur, Damien Schlachter, Pierre Schweitzer et Benoit Wagler. Implémentation d'un module sémantique en ocaml pour le parser de j. hale. Master's thesis, Universié de Bordeaux 1, 2006.

[Neg02a] Sara Negri. A normalizing system of natural deduction for intuitionestic linear logic. *Archive for Mathematical Logic*, 2002.

[Neg02b] Sara Negri. A normalizing system of natural deduction for intuitionistic linear logic. *Archive for Mathematical Logic*, 2002.

[Neg02c] Sara Negri. Varieties of linear calculi. *Journal of Philosophical Logic*, vol. 31 : 569–590, 2002.

[Nir03] Yannick Le Nir. *Structures des analyses syntaxiques catégorielles. Application à l'inférénce grammaticale.* PhD thesis, université de Renne 1, decembre 2003.

[Par00] Parigot. On the computational interpretation of negation. dans *CSL : 14th Workshop on Computer Science Logic.* LNCS, Springer-Verlag, 2000.

[Par92] Michel Parigot. $\lambda\mu$-calculus : An algorithmic interpretation of classical natural deduction. *proceedings of the Fourth International Conference on Types Lambda Calculi and Applications*, pages 190–201, 1992.

[Par03] A-M Parisot. *Accord et cliticisation : l'accord des verbes a forme rigide en LSQ.* PhD thesis, UQAM, 2003.

[PBD+04] A-M. Parisot, D. Bourbonnais, C. Dubuisson, Lelièvre et J. Rinfret. L'ordre des signes et l'économie articulatoire : bouger moins pour signer lsq. *ACFAS, Montréal*, vol. 1, 2004.

[Par07] Yannick Parmentier. *SemTAG : une plate-forme pour le calcul sémantique à partir de Grammaires d'Arbres Adjoints.* PhD thesis, université de Nancy 1, 2007.

[PtMW90] Barbara H. Partee, Alice ter Meulen et Robert Wall. *Mathematical methods in linguistics.* Kluwer, 1990.

[PS87] Fernando C.N. Pereira et Stuart M. Shieber. *Prolog and Natural-Language Analysis*, volume 10 of *CSLI Lecture Notes*. Chicago University Press, Stanford, 1987.

[Per71] D. Perlmutter. Deep and surface structure constraint in syntax. *Rinehart & Winston*, 1971.

[Per00] Guy Perrier. Interaction grammars. dans *COLING*, pages 600–606. Morgan Kaufmann, 2000.

[Pol89] Jean-Yves Pollock. Verb movement, ug, and the structure of ip. *Linguistic Inquiry*, vol. 20, 1989.

[Pol97] Jean-Yves Pollock. *Langage et Cognition*. Presses universitaires de France, 1997.

[RMV97] J. van Benthem R. Muskens et A. Visser. Dynamics. dans J. van Benthem et A. ter Meulen, Èditeurs, *Handbook of Logic and Language*, chapter 10, pages 587–648. Elsevier, 1997.

[Rad97] Andrew Radford. *Syntactic theory and the structure of English*. Cambridge textbooks in linguistics, 1997.

[Ran04] Aarne Ranta. Computational semantics in type theory. *Mathématiques et Sciences humaines*, vol. 165 : 31–57, 2004.

[Rei07] Tanya Reinhart. Les structures mentales et la langue : la théorie du langage de Chomsky. *Chomsky*, pages 78–93, 2007.

[Ret97] Christian Retoré. Pomset logic : a non-commutative extension of classical linear logic. dans Philippe de Groote et James Roger Hindley, Èditeurs, *Typed Lambda Calculus and Applications, TLCA'97*, volume 1210 of *LNCS*, pages 300–318, 1997.

[Ret00] Christian Retoré. Systèmes déductifs et traitement des langues : un panorama des grammaires catégorielles, April 2000.

[Ret03] Christian Retoré. Semantic aspects of minimalist grammars. *Algebraic Methods in Language Processing – AMiLP*, vol. 21, 2003.

[Ret04] Christian Retoré. A description of the non-sequential execution of petri nets in partially commutative linear logic. dans Jan van Eijck, Vincent van Oostrom et Albert Visser, Èditeurs, *Logic Colloquium 99*, Lecture Notes in Logic, pages 152–181. ASL and A. K. Peters, 2004.

[Ret05] Christian Retoré. The logic of categorial grammars – lecture notes. Research Report n° 5703, INRIA, 2005. 108 pp.

[RS04] Christian Retoré et Edward Stabler. *Reseach on Language and Computation*, volume 2(1). Christian Retoré and Edward Stabler, 2004.

[RS03] Christian Retroré et Edward Stabler. Generative grammars in resource logics. *Research on Language and Computation*, 2003.

[RS97] G. Rozenberg et A. Salomaa. *Handbook of Formal Languages*. G. Rozenberg and A. Salomaa, Berlin, 1997.

[SP87] Ivan A. Sag et Carl J. Pollard. Head-driven phrase structure grammar : An informal synopsis. CSLI Report n° 87-79, Stanford University, Stanford University, 1987.

[Sau05] Alexis Saurin. Separation with streams in the lambdaμ-calculus. dans *LICS*, pages 356–365. IEEE Computer Society, 2005.

[Sch85] Stuart M. Schieber. Evidence against the context-freeness of natural language. *Linguistic and Philosophy*, vol. 8, n° 3 : 333–343, august 1985.

[SC63] Marcel-Paul Schützenberger et Noam Chomsky. The algebraic theory of context free languages. *Computer Programming and Formal Languages*, pages 118–161, Amsterdam 1963.

[SKP84] Stuart M. Shieber, Lauri Karttunen et F.C.N. Pereira. Notes from the unification underground : a compilation of papers on unification based grammar formalism. Technical note n° 327, CSLI, Stanford, Ca., July 1984.

[SN97] Klaas Sikkel et Anton Nijholt. Parsing of context-free languages. dans *Handbook of Formal Languages* [RS97], chapter 2.

[Ski74] Burrhus Skinner. *About Behaviorism*. Random House Inc, 1974.

[Spo92] Dominique Sportiche. Clitic constructions. *Phrase Structure and the Lexicon*, 1992.

[Spo98] Dominique Sportiche. French predicate clitics and clause structure. *Small Clauses*, 1998.

[Sta96] Edward Stabler. Computing quantifier scope. *Ways of Scope Taking*, 1996.

[Sta97] Edward Stabler. Derivational minimalism. *Logical Aspect of Computational Linguistic*, vol. 1328, 1997.

[Sta99] Edward Stabler. Remnant movement and structural complexity. *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299–326, 1999.

[Sta01] Edward Stabler. Recognizing head movement. *Logical Aspects of Computational Linguistics*, vol. Springer-Verlag, n° 2099, 2001.

[Ste91] M. Steedman. Structure and intonation. *Language*, vol. 68 : 260–296, 1991.

[Ste81] Mark Steedman. Parsing spoken language using combinatory grammars. dans Masaru Tomita, Èditeur, *Current Issues in Parsing Technology*, pages 113–126. Kluwer Academic Publishers, Dordrecht, 1981.

[Ste00] Mark Steedman. *The syntactic process*. MIT Press, Cambridge, 2000.

[Ste93] Mark J. Steedman. Categorial grammar. *Lingua*, vol. 90 : 221–258, 1993.

[Tar39] Alfred Tarski. On undecidable statements in enlarged systems of logic and the concept of truth. *Journal of Symbolic Logic*, vol. 4 : 105–112, 1939.

[Tel05] Isabelle Tellier. *Modéliser l'acquiqition de la syntaxe du langage naturel via l'hypothèse de la primauté du sens*. Habilitation à diriger des recherches, 2005.

[Ver99] Willemijn Vermaat. Controlling movement : Minimalism in a deductive perspective. Master's thesis, Universiteit Utrecht, 1999.

[Ver05] Willemijn Vermaat. *The logic of variation. A cross-linguistic account of wh-question formation*. PhD thesis, Utrecht university, 2005.

[Voi05] E. Voisin. Flexion et ordre des signes en lsf. *Actes du 9é atelier des doctorants en linguistique*, vol. 1 : 257–261, 2005.

[VKre] E. Voisin et L. Kervajan. Typologie des verbes et formes verbales non marqués en lsf : incidences sur l'organisation syntaxique. *Silexicales*, vol. 1, a paraitre.

[You67] Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, vol. 10, n° 2 : 189–208, 1967.

Annexe A

Définition des GM sous forme de chaînes

Dans cette version des GMs, on utilise les mêmes définitions de la grammaire, à l'exception de l'ensemble de règles. Nous rappelons ici cette définition.

Une *grammaire minimaliste* est définie par un quintuplet $\langle V, Traits, Lex, \Phi, c\rangle$ où :

$V = \{P \cup I\}$ est l'ensemble fini des traits non-syntaxiques avec,

P est l'ensemble des formes phonologiques
I est l'ensemble des formes logiques

$Traits = \{B \cup S \cup L_a \cup L_e\}$ est l'ensemble fini des traits syntaxiques,
Lex est l'ensemble des expressions construites à partir de P et de $Traits^*$,
$\Phi = \{merge, move\}$ est l'ensemble des fonctions génératrices,
$c \in Traits$ est le trait acceptant.

Une entrée lexicale est composée de façon analogue et on la note $s : E$ où s est la chaîne associée à cette entrée et E la séquence de traits.

Dans la suite, on utilisera la notation $s : E$ pour signifier que c'est un élément dérivé, donc non-lexical.

Lorsque le caractère lexical ou composé n'est pas défini, nous utiliserons le symbole '$\cdot$'.

L'opération de fusion est alors définie par plusieurs expressions dépendant du caractère lexical des éléments qui entrent en jeu.

Sélection d'une expression simple par une entrée lexicale : si l'élément qui porte le sélecteur est lexical et que le trait de base avec lequel il est combiné est le dernier de la séquence, on concatène les chaînes en plaçant la tête gauche :

$$\frac{s :: = f\gamma \quad t \cdot f, \alpha_1, \cdots, \alpha_k}{st : \gamma, \alpha_1, \cdots, \alpha_k}$$

Sélection d'une expression simple par une entrée dérivée : si l'élément qui porte le sélecteur est dérivé et que le trait de base avec lequel il est combiné est le dernier de la séquence, on concatène les chaînes en plaçant la tête à droite :

$$\frac{s :=f\gamma, \alpha_1, \cdots, \alpha_k \quad t \cdot f, \beta_1, \cdots, \beta_k}{st : \gamma, \alpha_1, \cdots, \alpha_k, \beta_1, \cdots, \beta_k}$$

Sélection d'une expression composite : ce cas correspond à l'utilisation d'un trait de base qui n'est pas le dernier trait de la séquence. Dans ce cas, cette expression devra être ultérieurement déplacée. On se contente de stocker toutes les données dans la même chaîne :

$$\frac{s \cdot =f\gamma, \alpha_1, \cdots, \alpha_k \quad t \cdot f\delta, \beta_1, \cdots, \beta_k}{s : \gamma, \alpha_1, \cdots, \alpha_k, t : \delta, \beta_1, \cdots, \beta_k}$$

Le mouvement se différencie en fonction du nombre d'assignés que l'élément porte encore.

Mouvement final d'un élément : ainsi, si c'est le dernier, le déplacement est effectivement réalisé :

$$\frac{s : +f\gamma, \alpha_1, \cdots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \cdots, \alpha_k}{ts : \gamma, \alpha_1, \cdots, \alpha_{i-1}, \alpha_{i+1}, \cdots, \alpha_k}$$

Mouvement non-final d'un élément : sinon, il reste dans la liste :

$$\frac{s : +f\gamma, \alpha_1, \cdots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \cdots, \alpha_k}{s : \gamma, \alpha_1, \cdots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \cdots, \alpha_k}$$

On remarquera qu'avec ces définitions, on ne peut utiliser que les déplacements forts.

De plus, le résultat de ce type de dérivation n'est pas un arbre minimaliste tel que nous l'avons défini au chapitre 2. En effet, ce que l'on obtient représente la suite des opérations utilisées pour obtenir la dérivation. L'arbre possède donc des noeuds binaires et unaires. Cependant, on montre qu'avec un simple transducteur sur l'arbre on reconstruit les arbres dérivés, et donc l'arbre de dérivation.

Annexe B

Implémentation pour les GMs

Sommaire

B.1 Analyseurs syntaxiques 259
B.2 Autres implémentations 263
B.3 Conclusion 267

La réalisation d'analyseurs syntaxiques performants est évidemment un enjeu capital pour toute formalisation de linguistique informatique. La mise en œuvre de nos nouvelles grammaires - GMC - étant trés récente, leur implémentation n'a pas encore été effectuée. C'est certainement la prochaine étape cruciale de ce travail. Cependant, basées sur les GMs beaucoup plus anciennes, ces dernières ont quant à elles plusieurs versions d'analyseur syntaxique.

Nous allons voir comment ceux-ci peuvent être utilisés pour obtenir des analyses syntaxiques, puis les différentes modifications que nous avons proposées autour de ces questions.

B.1 Analyseurs syntaxiques

Il existe deux analyseurs syntaxiques pour les GMs, qui sont ceux de :
- John Hale : implémenté en Ocaml et basé sur l'algorithme CYK.
- Guillaumin : proposant une transformation des GMs vers les MCFGs.

B.1.1 Parser de John Hale

Une première version d'un parser pour les GMs et les grammaires multimodales a été développée par Ed Stabler écrit en Prolog et basé sur CYK - Cocke, Younger et Kasami [CS70], [Kas65] et [You67]. À partir de cette version, John Hale [Hal03] a proposé une implémentation en Ocaml pour les GMs.

Les GMs étant lexicalisées, l'enjeu pour la rédaction de grammaires est de concevoir les lexiques. Ces derniers sont rédigés en Prolog pour pouvoir être partagés entre les deux versions du parser. Ils se caractérisent par une suite d'entrées lexicales telles que nous les avons présentées. Chaque entrée est composée d'une partie phonologique et d'une séquence

de traits :

[*partie phonologique*] :: [*séquence de traits*].

À partir de ces lexiques, on construit une représentation en Ocaml en associant à chaque item un tuple dont la première composante est la partie phonologique et le reste des traits, créés avec des constructeurs les différenciant.

Pour utiliser ce parser, il suffit de charger la bibliothèque de fonctions du package - qui se trouve dans le fichier main.ml. Ceci fait, nous pouvons utiliser une fonction *parser* qui prend en argument une chaîne de caractères représentant l'énoncé à analyser et le chemin vers le lexique par rapport auquel il faut que l'analyse soit faite.

```
parse "lexiquefrancais/corpuseurotra/francais.pl" "Jean aimer Marie";;
accepted as category c in 0.13 seconds with 0.12 seconds for
initialization, chart size: 78
- : Chart.DerivationSet.t = <abstr>
```

Le résultat de la dérivation est la succession d'étapes nécessaires à l'analyse que l'on peut retrouver :

```
# let result = parse "lexiquefrancais/corpuseurotra/francais.pl"
"Jean aimer Marie";;
accepted as category c in 0.13 seconds with 0.10 seconds for
initialization, chart size: 78
val result : Chart.DerivationSet.t = <abstr>
# Chart.DerivationSet.elements result;;
- : Chart.DerivationSet.elt list =
[Chart.Binary ("c", "::=t c", "t", Chart.Lexical "::=t c",
  Chart.Unary ("t", "+case t,-case",
   Chart.Binary ("+case t,-case", "::=>verbe +case t", "verbe,-case",
    Chart.Lexical "::=>verbe +case t",
    Chart.Binary ("verbe,-case", "::=>little_v verbe", "little_v,-case",
     Chart.Lexical "::=>little_v verbe",
     Chart.Binary ("little_v,-case", "=d little_v", "::d -case",
      Chart.Binary ("=d little_v", "::=>v =d little_v", "v",
       Chart.Lexical "::=>v =d little_v",
       Chart.Unary ("v", "+case v,-case",
        Chart.Binary ("+case v,-case", "::=d +case v", "::d -case",
         Chart.Lexical "::=d +case v", Chart.Lexical "::d -case"))),
      Chart.Lexical "::d -case")))))]
```

Nous avons développé une suite de fonctions pour pouvoir afficher sous forme d'arbre la dérivation d'une analyse acceptante d'un énoncé par rapport à un lexique donné.

```
# #use "amblard.ml";;
val ajout2 : Chart.DerivationSet.t -> string = <fun>
val fichier : string -> Chart.DerivationSet.t -> unit = <fun>
#  let result = parse "tests/larsonian.pl" "John love -s a woman";;
accepted as category c in 0.30 seconds with 0.09 seconds for
initialization, chart size: 106
val result : Chart.DerivationSet.t = <abstr>
```

```
# fichier "johnloves.dot" result;;
- : unit = ()
```

Items

À partir des entrées lexicales, on construit des éléments de type *item* pour représenter chaque étape de la dérivation. On ajoute un booléen pour différencier les éléments construits des lexicaux. D'autre part, une autre composante d'*item* est *exposed* qui correspond à la séquence de traits restant accessible à la suite de la dérivation pour le parser.

Un autre ensemble est masqué par les définitions ci-dessus, c'est la partie non instanciée des items. En effet, on essaie de minimiser cet ensemble pour obtenir une dérivation. On définit donc deux types pour ces derniers : *flist* et *clist* qui sont des listes variables utilisées pour représenter la dérivation comme des règles de déduction des grammaires minimalistes elles-mêmes en déduction naturelle.

Chart

Chart est la structure de données où les items dérivés sont mémorisés. Cette étape est divisée en deux grandes parties : *lookup* et *derivation*.

Look up : L'application des règles de construction est complètement déterminée par le premier trait de la tête - la tête de la *exposed list*. De manière intuitive, si le parser rencontre un sélecteur, il voudra utiliser une fusion. Pour cela, il cherchera un élément ayant le type de base correspondant. C'est la même chose pour le mouvement.

Donc, Chart est indexée par le premier élément de la liste de traits et est donc composée :

1. d'un tableau de listes d'entrées.
2. d'une table de hashage qui recherche le tableau d'indices à partir du premier élément de la liste de traits.

Derivation : c'est l'ensemble qui mémorise la dérivation pour la restituer à l'utilisateur par la suite. La manière la plus naturelle de le fabriquer est de lui assigner l'union de l'ancienne dérivation et de la nouvelle étape trouvée par le parser. Cette approche nous donne alors un parser polynômial. Malgré cela, si le nombre de dérivations est exponentiel, comme pour une approche naïve, nous aurons un nombre exponentiel d'ensembles de dérivations.

Pour limiter ce nombre, on utilise *Backpointers* : au lieu de stocker toute la dérivation comme un arbre ou une structure complexe, on mémorise le genre de dérivation obtenue pour une entrée donnée et un pointeur vers l'entrée précédente. Nos entrées doivent également connaître leur propre adresse et ainsi on a un ensemble de *Backpointer* pour retrouver la dérivation à partir d'une analyse acceptante, et ce de manière unique.

Queue

L'ordre des opérations effectuées est l'un des champs des items qui retrace le résultat de la dérivation. Pour l'instant, cette opération n'est pas intégrée dans Chart. On la modélise

alors par une *queue* qui dirige la stratégie d'exploration - on peut aussi la modéliser par une pile ou une queue de priorité. En effet, c'est elle qui mémorise l'ensemble de toutes les dérivations lancées en même temps et cherche à les faire aboutir dans la même direction.

Pour que l'extraction d'éléments dans la queue soit plus rapide, l'ordre est indexé par le premier trait comme pour les items. La stucture générique utilisée est *IndexedQueue* dans laquelle les éléments arrivent avec une fonction de classification qui leur assigne un entier. À l'intérieur de cette queue, on utilise des pointeurs pour marquer le début et la fin.

Unification

L'unification laisse à l'auteur des règles de déduction un grand degré de liberté. En effet, il n'est pas nécessaire de donner tous les détails de composition des items mais seulement les conditions à satisfaire. En particulier, le parser utilise l'algorithme *d'unification par transformation*, qui essaie tous les types de fusion ou de déplacement sur les items en tête de liste et n'accepte que les opérations réalisables répondant aux principes minimalistes. Ainsi, l'ensemble des items à unifier diminue jusqu'à rencontrer le symbole acceptant pour le fragment ou la phrase proposé(e).

À chaque pas, un item *déclenchant* une opération est sorti de la queue. Pour celui-ci, on applique toutes les opérations possibles en sortant avec Chart la liste de tous les items dont le premier trait peut se combiner avec l'item déclenchant. On récupère l'ensemble des résultats positifs et on recommence avec la nouvelle queue.

Cette description du parser est très succincte car le fonctionnement interne est assez élaboré et, pour le moment, en cours de modification car J. Hale essaie d'intégrer la *Queue* à Chart ce qui donnera une version plus efficace. Toutefois, cette version du parser ne prend pas en compte la sémantique, il ne fonctionne et n'est écrit que pour l'analyse syntaxique.

B.1.2 Guillaumin et al.

L'idée est d'utiliser une conversion des GMs vers les MCFGs - proposée par Guillaumin dans [Gui04] et étudiée par Méry, [Mer06] - puis d'utiliser un parser pour les MCFGs -par exemple celui présenté dans [Alb02].

On cherche la clôture d'un lexique d'une GM par les règles de dérivation et on le transforme en MCFG. On utilise les propriétés des MCFGs permettant, pour chaque règle du système, d'associer une composition d'argument. Ainsi, dans une dérivation d'une GM, une fusion vidant un élément de ses traits implique la concaténation des chaînes associées dans la MCFG. Deux arguments de la règle seront alors concaténés, comme nous l'avons présenté dans le chapitre 4.

La procédure de conversion est assez complexe mais le résultat est une modélisation quasiment directe de l'ensemble des dérivations réalisables avec la grammaire. Cette transformation donne une MCFG de taille importante et est coûteuse en temps. Cependant le temps de recherche de la dérivation est beaucoup plus efficace puisqu'il y a beaucoup moins d'ambiguïtés dans la dérivation.

B.2 Autres implémentations

À partir de ces analyseurs, nous avons proposé deux extensions : l'une étant l'implémentation des circuits introduits au chapitre 4, puis une première version simplifiée d'un calcul sémantique à partir du parser de John Hale.

B.2.1 Représentations sous forme de circuit

Nous avons rédigé un module basé sur la version des lexiques pour les GMs de John Hale qui transforme un lexique en un circuit. Il est donc écrit en Ocaml.

Cette implémentation pour la réalisation de circuit se base sur celle utilisée pour la réalisation de représentation graphique - sous forme d'arbre - décrite dans [Amb03]. Il s'agit de créer un fichier *.dot* - graphviz - qui contient un noeud pour chaque type de trait de base et les transitions extraites des entrées lexicales.

Pour cela, on écrit la fonction *lex_to_graph* qui crée ce fichier et applique la fonction *lextograph2*. C'est la fonction qui écrit le contenu du fichier *.dot*, c'est-à-dire la suite de transitions présentes dans le circuit. Pour cela cette fonction construit le lexique relatif au nom qu'on lui a passé en argument, une table globale - via la fonction *lextotab* - et une table normalisée - via *lextograph* et *normalisation*. Nous allons présenter le résultat de ces trois fonctions et reviendrons sur la fonction principale par la suite.

Fonction lextograph

```
let rec lextograph lex =
  let rec traitementitem item res =
    match item with
a::lis -> (traitementitem lis (rempl a res))
      |[] -> res
  in
  match lex with
      a::lis -> (traitementitem a [[];[];[];[]])::(lextograph lis)
    |[] -> []
;;
```

La fonction *lextograph* prend en argument un lexique. Elle examine tous les items du lexique et, pour chacun d'eux, elle applique un traitement qui consiste à remplir une liste vide avec les différents traits de l'entrée lexicale par la fonction *rempl*.

```
let rempl f [a; b; c; d] =
match f with
  Feature.Select     f -> [Feature.show_name f::a;b;c;d]
 |Feature.Category   f -> [a;Feature.show_name f::b;c;d]
 |Feature.Attract    f -> [a;b;Feature.show_name f::c;d]
 |Feature.Licensee   f -> [a;b;c;Feature.show_name f::d]
 |Feature.RIncorp    f -> [Feature.show_name f::a;b;c;d]
 |Feature.LIncorp    f -> [Feature.show_name f::a;b;c;d]
 |Feature.RAffhop    f -> [Feature.show_name f::a;b;c;d]
 |Feature.LAffhop    f -> [Feature.show_name f::a;b;c;d]
```

```
 |_ -> [a;b;c;d]
;;
```

Elle remplit un tableau à quatre cases en mettant le string du trait matché :

- dans la première case si c'est un selecteur,
- dans la seconde case si c'est un trait de base,
- dans la troisième case si c'est un assignateur,
- dans la dernière case si c'est un assigné.

La fonction *lextograph* construit donc une table contenant toutes les séquences de traits rencontrées dans un lexique. À partir de ce résultat, nous voulons extraire une table ne contenant qu'un seul représentant pour chaque séquence de traits. Pour cela nous normalisons le résultat par la fonction *normalisation*.

```
(*fonction de normalisation de *)
let rec normalisation lis res =
  let dedans2 a b = a=b in
  match lis with
      a::liss -> if (List.exists (dedans2 a) liss)
       then (normalisation liss res)
      else (normalisation liss [a]@res)
    |[] -> res
```

Cette dernière examine linéairement les items présents dans la table et cherche pour chacun s'il est encore présent dans la table. S'il l'est, elle ne le considère pas, sinon elle l'ajoute dans la table résultat.

Ainsi, après avoir appliqué *normalisation* et *lextograph* on obtient un tableau qui contient toutes les séquences de traits du lexique sans doublon, répartis selon leur type de traits .

```
# normalisation (lextograph "lexiquefrancais/larsonianclitics7bis.pl") [];;
- : string list list list =
[[["t"]; ["c"]; []; []]; [["t"]; ["c_rel"]; ["wh_rel"]; []];
 [["d"; "v"]; ["little_v"]; []; []]; [["little_v"]; ["verbe"]; []; []];
 [["v"]; ["verbe"]; []; []]; [["big_v"]; ["Big_v"]; []; []];
 [["finclitic"]; ["modifieur2"]; []; []]; [["Big_v"]; ["t"]; ["case"]; []];
  [["d"; "d"; "finclitic"]; ["little_v"]; ["case"]; []];
 [["d"; "d"; "verbe"]; ["little_v"]; ["case"]; []];
  [...]; ...]; ...]
```

fonction lextotab

```
(*fonction qui à partir d'un lexique créer la table cherchée*)
let lextotab f =
       let tableauintermediaire tab =
              let rec itemtocouple ite =
                   match ite with
                      [] -> (Feature.Category (Feature.Const "empty"),[])
                     |a::lis -> match a with
```

```
                        Feature.Category f -> (a, ite)
                      |_ -> itemtocouple lis
            in
            let rec tableinter tab res =
                    match tab with
                     a::lis -> tableinter lis ((itemtocouple a)::res)
                    |[] -> res
            in
                    (tableinter ( tab) [])
      in
      triage (tableauintermediaire (lexique (grammar f)))
;;
```

Cette fonction crée une table que nous allons détailler maintenant, puis elle la trie, nous verrons selon quelle procédure ensuite. Elle emploie la fonction intermédiaire *tableauintermediaire* qui, à partir d'un lexique, construit un nouveau tableau via la sous-fonction *tableinter*, et un tableau vide. *Tableinter* est une fonction récursive qui traite tous les éléments d'un lexique en transformant chaque item par la fonction *itemtocouple*. Cette dernière cherche la catégorie de base de l'item et lui associe la liste composée de la catégorie et des assignés.

Elle utilise la fonction : *tableauintermediaire* (prend un lexique et crée un nouveau tableau)→ *tableinter* (à partir d'un lexique applique sur chaque item la fonction *itemtocouple*) → *itemtocouple* (crée des couples contenant chacun une catégorie de base et une liste d'assignés).

Ce résultat doit être trié suivant la fonction *triage*. Cette fonction fait appel à plusieurs sous-fonctions pour vérifier les différents cas possibles et renvoie un tableau contenant des couples dont le premier élément est une catégorie de base et le second les différentes séquences d'assignés que l'on peut trouver après ce type de base.

Par exemple pour les clitiques, il y a plusieurs entrées de type p pouvant être suivies de différentes séquences d'assignés.

```
# lextotab "lexiquefrancais/larsonianclitics7bis.pl";;
- : (Feature.t * string) list =
[(Feature.Category (Feature.Const "p"), "/-Y/-EN/-F")]
```

Revenons à la fonction principale *lextograph*.

Elle utilise une table contenant les différents types du lexique sans doublon - appelée *tab* dans la fonction - et une autre contenant les séquences d'assignés possibles pour chaque catégorie de base - appelé *tabglobal* dans la fonction.

On passe ces deux arguments avec un compteur (initialisé à 0) à une fonction récursive qui pour chaque élément de *tab* (toute suite de traits) donne le code formant une transition de l'élément sélectionné vers le trait de base en utilisant :

- le compteur pour nommer cette nouvelle transition,
- la *tabglobal* pour étiqueter les transitions avec les différentes séquences d'assignés possibles sur la demi-transition entrante et les séquences d'assignateurs sur la demi-transition sortante.

```
let lextograph2 f =
```

```
  let rec remm res2 i tabglobal =
    match res2 with
a::suit -> (trait a tabglobal i)^(remm suit (i+1) tabglobal)
      |[] -> ""
  in
  let lex = lexique (grammar f) in
  let tabglobal = lextotab f in
  let tab = normalisation (lextograph lex) [] in
    lex;
    tabglobal;
    tab;
    "digraph G {\n node [shape=circle width=0.5]"^(remm tab 0 tabglobal)^
     "\n }"
;;
```

On a ainsi créé un fichier qu'il suffit de compiler avec *graphviz* pour obtenir le circuit représentant la version abstraite du lexique de départ.

B.2.2 Module de sémantique simple pour le parser de J. Hale

Nous avons encadré un projet de Master 1 dont le sujet était implémentation d'une interface syntaxe sémantique pour le parser de John Hale, [NSSW06].

La version du calcul sémantique qui a été utilisée est relativement simple. Il s'agit d'associer des λ-termes classiques - sans opérateur $\oplus$ - à chaque entrée du lexique et de réaliser les applications fonctionnelles au moment de la fusion de deux constituants en utilisant le module de λ-calcul CCT, implémenté par G. Huet, [Hue02a].

L'enjeu principal est donc de faire le lien entre le résultat d'une analyse syntaxique obtenue grâce aux parsers et de construire la bonne dérivation sémantique grâce à la bibliothèque CCT. Pour faire cette jonction, il faut a posteriori pouvoir retrouver les entrées lexicales utilisées dans l'analyse pour chercher leur contrepartie sémantique et la passer avec la dérivation au module sémantique.

Le schéma de fonctionnement est celui représenté dans la figure 53

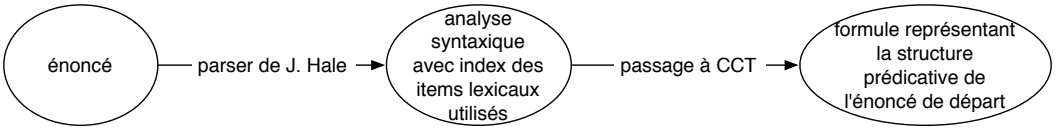


FIG. 53 – Processus vers un calcul sémantique

Pour cela, ils ont suivi l'idée que je leur ai proposée qui est d'insérer un index dans le lexique construit et de faire passer cet index tout au long de la dérivation syntaxique jusqu'au résultat final. Ainsi, une fois l'analyse terminée, les "feuilles" - ou positions des items lexicaux dans la dérivation - contiennent non plus des séquences de traits mais un index permettant de retrouver l'item utilisé.

Une fonctionnalité directement ajoutée par ce groupe d'étudiants est d'extraire des représentations sous forme d'arbre du parser avec les formes phonologiques sur les feuilles,

rendant leur lecture plus abordable à des néophytes. Une fois cette opération réalisée, il ne reste qu'à lancer automatiquement CCT et lui passer les bons arguments. Ceci ne se fait pas directement car le parser est rédigé en Ocaml et CCT en Camlp4, mais un simple appel à une fonction de renversement permet de passer de l'un à l'autre.

L'utilisation de ce module est élémentaire car les fonctions syntaxiques sont reprises dans les fonctions de sémantique. Ainsi il suffit de demander le calcul sémantique pour obtenir directement le résultat. Ce dernier peut être visualisé directement en ligne de commande via un string ou peut être donné sous forme mathématique pour un document LaTeX.

```
# value r = semantic "lexique.pl" "un chat dormir";
accepted as category c ...
[...]

# List.map (fun x -> formulae_of_nodelist console_tokens
      (nodeinfo_of_node x) .expr) r;
-: llist string = ["Ev | (chat v) && (dormir v)"]
```

B.3 Conclusion

L'étape clé dans l'évolution de ces travaux est de proposer un analyseur pour les GMCs (présentées comme un sous-fragment de la logique linéaire, il est possible d'envisager d'utiliser un parser pour celle-ci).

En parallèle de cette implémentation, il reste à proposer une implémentation d'un calcul sémantique - $\lambda\mu$-calcul - incluant l'opérateur algébrique $\oplus$. Une fois ceci réalisé, il sera directement possible de relier ces deux modules pour obtenir les formules escomptées.

Enfin, une dernière partie du travail d'implémentation restant est de proposer un environnement convivial - ou moins universitaire - pour présenter les différents résultats et possibilités offertes par ces systèmes. Actuellement, seuls les utilisateurs confirmés peuvent comprendre autant ce qui doit être analysé que le résultat des analyses.

Annexe C

Extension des dérivations par file et compteur en double fonction : le cas de $a^n b^{2^n}$

On peut étendre la version du compteur a^{2^n} à un compteur dont il est en exposant direct d'un terminal et en exposant d'exposant sur un deuxième, comme pour les mots du type $a^n b^{2n}$.

La principale modification est l'ajout d'un marqueur spécifique pour la deuxième forme phonologique et d'insérer ce dernier à chaque fin de pas d'itération.

On utilise le lexique suivant :

Lexique 136.

	phase d'initialisation		
type : 1	*w -m*	*type* : 2	*=w x -l*
	phase d'itération		
type : 3	*=x +M y -m -o*	*type* : 4	*=y +L z -l*
type : 5	*=z y -l*	*type* : 6	*=z x -l*
	phase de conclusion		
type : 7	*=x +M d*	*type* : 8	*=d +L d /b/*
type : 9	*=d c*	*type* : 10	*=c +O c /a/*

On peut représenter le lexique par le circuit présenté par la figure 54.

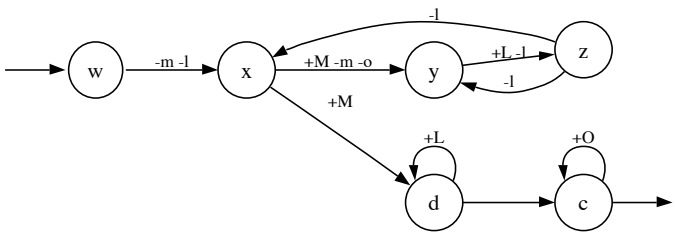


FIG. 54 – Représentation du lexique pour a^{2^n}.

Dérivation 137.

1. *entrée lexicale de type 1 :*

   $w\ {-m}$

   entrée lexicale de type 2 :

   $=w\ x\ {-l}$

   fusion : $\underbrace{\mathbf{x}\ \mathbf{-l}, -m}$

   positionnement du marqueur de file et du premier marqueur pour le terminal dont le compteur est en exposant d'exposant. C'est-à-dire $n = 0$, $a^0b^{2^0} = b$.

2. *passage dans une itération, entrée lexicale de type 3 :*

   $=x\ {+M}\ y\ {-m}\ {-o}$

   fusion : $\underbrace{\mathbf{+M\ y\ -m\ -o}, \underbrace{\mathbf{-l}, -m}}$

   On copie/déplace le marqueur de file et on introduit un marqueur pour le deuxième terminal $-o$

   déplacement : $\underbrace{\mathbf{y\ -m\ -o}, -l}$

3. *entrée lexicale de type 4 :* $=y\ {+L}\ z\ {-l}$

   fusion : effacement et introduction d'un marqueur de terminal :

   $\underbrace{\mathbf{+L\ z\ -l}, \underbrace{\mathbf{-m\ -o}, -l}}$

   déplacement : $\underbrace{\mathbf{z\ -l}\ , -m\ -o}$

4. *Fin de l'itération. Entrée lexicale de type 6 : duplication :* $=z\ x\ {-l}$

   fusion : $\underbrace{\mathbf{x\ -l}\ , \underbrace{\mathbf{-l}\ , -m\ -o}}$

   Si on passe en phase de conclusion, on obtient $a^1b^{2^1} = ab^2$.

5. *entrée lexicale de type 3 :* $=x\ {+M}\ y\ {-m}\ {-o}$

   fusion : $\underbrace{\mathbf{+M\ y\ -m\ -o}, \underbrace{\mathbf{-l}\ , \underbrace{\mathbf{-l}\ , -m\ -o}}}$

   déplacement : $\underbrace{-o, \mathbf{y\ -m\ -o}, \underbrace{\mathbf{-l}\ , -l}}$

6. *entrée lexicale de type 4 :* $=y\ {+L}\ z\ {-l}$

   fusion : $\underbrace{\mathbf{+L\ z\ -l}, \underbrace{-o, \mathbf{-m\ -o}, \underbrace{\mathbf{-l}\ , -l}}}$

   déplacement : $\underbrace{\mathbf{z\ -l}, \underbrace{-o, \mathbf{-m\ -o}, -l}}$

Remarque 138. *Dans la suite de la dérivation, il y a une explosion du nombre de crochets en-dessous. Nous supposerons induites les relations de tête par l'ordre gauche-droite.*

7. *Suite de l'itération, entrée lexicale de type 5 :* $=z\ y\ -l$

   fusion : $\underline{\mathbf{y}\ -\mathbf{l},\ -\mathbf{l},\ -o,\ -\mathbf{m}\ -\mathbf{o},\ -l}$

8. *entrée lexicale de type 4 :* $=y\ +L\ z\ -l$

   fusion : $\underline{+\mathbf{L}\ \mathbf{z}\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -o,\ -\mathbf{m}\ -\mathbf{o},\ -l}$

   déplacement : $\underline{\mathbf{z}\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -o,\ -\mathbf{m}\ -\mathbf{o}}$

9. *passage vers la fin de l'itération, entrée lexicale de type 6 : duplication* $=z\ x\ -l$

   fusion $\underline{\mathbf{x}\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -o,\ -\mathbf{m}\ -\mathbf{o}}$

   On réitère et on passe vers la fin de la dérivation, ce qui se fait en 23 étapes successives avant de retrouver la même situation avec davantage de marqueurs.

30. *entrée lexicale de type 7 : effacement du marqueur de file* $=x\ +M\ d$

    fusion : $\underline{+\mathbf{M}\ \mathbf{d},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ \underline{-o,\ -\mathbf{o}},\ -\mathbf{m}\ -\mathbf{o}}$

    déplacement : $\underline{\underline{-o,\ -\mathbf{o}},\ -\mathbf{m}\ -\mathbf{o},\ \mathbf{d},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -\mathbf{l},\ -l}$

    Dans la phase finale, il ne reste plus qu'à substituer chaque $-l$ par un /b/ et chaque $-o$ par un /a/. Ces substituions sont réalisées dans cet ordre. Il y a trois étapes pour chaque marqueur : l'entrée lexicale 8 ou 10, une fusion et un déplacement. Nous présentons le traitement du dernier marqueur, donc 30 étapes suivantes : résultat :
    $\underline{c\ /a/,/a/,/b/,/b/,/b/,/b/,/b/,/b/,/b/,/b/,\ -o}$

36. *entrée lexicale de type 10 :* $=c\ +O\ c\ /a/$

    fusion : $\underline{+O\ c\ /a/,/a/,/a/,/b/,/b/,/b/,/b/,/b/,/b/,/b/,/b/,\ -o}$

    déplacement : $\underline{c\ /a/,/a/,/a/,/b/,/b/,/b/,/b/,/b/,/b/,/b/,/b/}$

Annexe D

Systèmes de règles

Sommaire

D.1 Grammaires Minimalistes 273
D.2 Logique mixte 274
D.3 Grammaires Minimalistes Catégorielles 275
D.4 Interface syntaxe-sémantique 276

D.1 Grammaires Minimalistes

Structure des listes de traits :

$$S(S \cup L_a)^* B(L_e)^*/FP/(FL)$$
ou bien $B(L_e)^*/FP/(FL)$

$$L ::= {=b}\ S_1 \mid B$$
$$S_1 ::= {=b}\ S_1 \mid {+d}\ S_1 \mid B$$
$$B ::= b\ S_2 \mid b$$
$$S_2 ::= -d\ S_2 \mid -d$$

merge

$$merge(t, t') = \begin{cases} < (H_t[l :: E], H_{t'}[l' :: E']) & \text{si } t \in Lex, \\ < (H_{t'}[l' :: E'], (H_t[l :: E]) & \text{sinon.} \end{cases}$$

move : $T_{MG} \to T_{MG}$
pour tout arbre $t = C[l : {+g}\ E, l' : -g\ E']$.
Il existe $C_1, C_2 \in S_t$ tels que : C_2 est la projection maximale de la feuille l' et C_1 est un 2-contexte représentant t privé de C_2.

- $C_2[l' : -g\ E'] = proj_{max}(C[l' : -g\ E])$
- $C_1[l : {+g}\ E, x_1] = proj_{max}(C[l : {+g}\ E, x_1])$

$$move(t) = >(C_2[l' : E'], C_1[l : E, x_1])$$

Head-Movement

$$merge(t,t') = \begin{cases} < (H_t[l'l :\ E], H_{t'}[\epsilon :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[\epsilon :\ E'], (H_t[l'l :\ E]) & \text{sinon.} \end{cases}$$

Soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l :=\!>\!h\ E]$ et $t' = H_{t'}[l : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[ll' :\ E], H_{t'}[\epsilon :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[\epsilon :\ E'], (H_t[ll' :\ E]) & \text{sinon.} \end{cases}$$

Affix-Hopping :
Soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : <\!=\!h\ E]$ et $t' = H_{t'}[l : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[\epsilon :\ E], H_{t'}[l'l :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[l'l :\ E'], (H_t[\epsilon :\ E]) & \text{sinon.} \end{cases}$$

et l'Affix-Hopping avec adjonction de la forme phonologique à droite par : soient t et $t' \in T_{MG}(A)$ tels que $t = H_t[l : h\!=\!> E]$ et $t' = H_{t'}[l : h\ E']$ avec $h \in B$:

$$merge(t,t') = \begin{cases} < (H_t[\epsilon :\ E], H_{t'}[ll' :\ E']) & \text{si } t \in Lex, \\ < (H_{t'}[ll' :\ E'], (H_t[\epsilon :\ E]) & \text{sinon.} \end{cases}$$

D.2 Logique mixte

D.2.1 Calcul de Lambek avec produit

Syntaxe des formules de la logique mixte :

$$\text{L} ::= \text{P} \mid \text{L} \odot \text{L} \mid \text{L}/\text{L} \mid \text{L}\backslash\text{L}$$

Règles de la logique mixte :

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash C}{< \Gamma; \Delta > \vdash C}\,[\backslash_e] \qquad \frac{\Delta \vdash A/C \quad \Gamma \vdash A}{< \Gamma; \Delta > \vdash C}\,[/_e]$$

$$\frac{< A; \Gamma > \vdash C}{\Gamma \vdash A\backslash C}\,[\backslash_i] \qquad \frac{< \Gamma; A > \vdash C}{\Gamma \vdash C/A}\,[/_i]$$

$$\frac{\Delta \vdash A \quad \Gamma \vdash B}{< \Delta; \Gamma > \vdash A \odot B}\,[\odot_i] \qquad \frac{\Delta \vdash A \odot B \quad \Gamma, < A; B >, \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[\odot_e]$$

$$\frac{}{A \vdash A}\,[axiom] \qquad \frac{\Gamma \vdash C}{\Gamma' \vdash C}\,[\text{entropis — pour } \Gamma' \sqsubset \Gamma]$$

D.2.2 Logique mixte

Syntaxe des formules de la logique mixte :

$$\text{L} ::= \text{P} \mid \text{L} \odot \text{L} \mid \text{L} \otimes \text{L} \mid \text{L}/\text{L} \mid \text{L}\backslash\text{L} \mid \text{L} \multimap \text{L}$$

Syntaxe des contextes des formules de la logique mixte :

$$\text{CTX} ::= \text{L} \mid \langle \text{CTX}; \text{CTX} \rangle \mid \{\text{CTX}, \text{CTX}\}$$

Règles de la logique mixte :

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash C}{< \Gamma; \Delta > \vdash C}\,[\backslash_e] \qquad \frac{\Delta \vdash A/C \quad \Gamma \vdash A}{< \Gamma; \Delta > \vdash C}\,[/_e] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \multimap C}{(\Gamma, \Delta) \vdash C}\,[\multimap_e]$$

$$\frac{< A; \Gamma > \vdash C}{\Gamma \vdash A\backslash C}\,[\backslash_i] \qquad \frac{< \Gamma; A > \vdash C}{\Gamma \vdash C/A}\,[/_i] \qquad \frac{(A, \Gamma) \vdash C}{\Gamma \vdash A \multimap C}\,[\multimap_i]$$

$$\frac{\Delta \vdash A \odot B \quad \Gamma, < A; B >, \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[\odot_e] \qquad \frac{\Delta \vdash A \otimes B \quad \Gamma, (A, B), \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[\otimes_e]$$

$$\frac{\Delta \vdash A \quad \Gamma \vdash B}{< \Delta; \Gamma > \vdash A \odot B}\,[\odot_i] \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{(\Delta, \Gamma) \vdash A \otimes B}\,[\otimes_i]$$

$$\frac{}{A \vdash A}\,[axiom] \qquad \frac{\Gamma \vdash C}{\Gamma' \vdash C}\,[\text{entropie} — \text{si } \Gamma' \sqsubset \Gamma]$$

D.3 Grammaires Minimalistes Catégorielles

Syntaxe des formules :

$$\begin{aligned} \text{L} &::= (\text{B}) \,/\, \text{P}_1 \mid \text{C} \\ \text{B} &::= \text{P}_1 \setminus (\text{B}) \mid \text{P}_2 \setminus (\text{B}) \mid \text{C} \\ \text{C} &::= \text{P}_2 \otimes (\text{C}) \mid \text{C}_1 \\ \text{C}_1 &::= \text{P}_1 \end{aligned}$$

Règles des GMCs :
merge :

$$\frac{\frac{\vdash (r_{spec}, r_{tete}, r_{comp}) : A \,/\, B \quad \Delta \vdash s : B}{\Delta \vdash (r_{spec}, r_{tete}, r_{comp} \bullet Concat(s)) : A}\,[/_e]}{\Delta \vdash (r_{spec}, r_{tete}, r_{comp} \bullet Concat(s)) : A}\,[\sqsubset]$$

$$\frac{\frac{\Delta \vdash s : B \quad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus A}{(\Delta; \Gamma\rangle \vdash (Concat(s) \bullet r_{spec}, r_{tete}, r_{comp}) : A}\,[\backslash_e]}{\Delta, \Gamma \vdash (Concat(s) \bullet r_{spec}, r_{tete}, r_{comp}) : A}\,[\sqsubset]$$

move :

$$\frac{\Gamma \vdash r_1 : A \otimes B \quad \Delta[u : A, v : B] \vdash r_2 : C}{\Delta[\Gamma] \vdash r_2[Concat(r_1)/u, \epsilon/v] : C}\,[\otimes_e]$$

move cyclique :

$$\frac{x : A \otimes B \vdash (\epsilon, x, \epsilon) : A \otimes B \quad \Delta[u : A, v : B] \vdash r : C}{\Delta[A \otimes B] \vdash r[x/u, \epsilon/v] : C}\,[\otimes_e]$$

fort/faible :

$$\frac{s : \Gamma \vdash A \otimes B \quad r[u, v] : \Delta[u : A, v : B] \vdash C}{r[Concat(s)/u, \epsilon/v] : \Delta[\Gamma] \vdash C}\,[move_{fort}]$$

$$\frac{s : \Gamma \vdash A \otimes B \quad r[u, v] : \Delta[u : A, v : B] \vdash C}{r[\epsilon/u, Concat(s)/v] : \Delta[\Gamma] \vdash C}\,[move_{faible}]$$

Head-Movement :

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A /_< B \qquad s : \Delta \vdash B}{\Gamma, \Delta \vdash (r_{spec}, r_{tete} \bullet s_{tete}, r_{comp} \bullet Concat(s_{-tete})) : A}\,[mg]$$

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A_> / B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, s_{tete} \bullet r_{tete}, r_{comp} \bullet Concat(s_{-tete})) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B_> \setminus A}{\Delta, \Gamma \vdash (Concat(s_{-tete}) \bullet r_{spec}, s_{tete} \bullet r_{tete}, r_{comp}) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus_< A}{\Delta, \Gamma \vdash (Concat(s_{-tete}) \bullet r_{spec}, r_{tete} \bullet s_{tete}, r_{comp}) : A}\,[mg]$$

Affix-Hopping :

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A /_> B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, \epsilon, r_{comp} \bullet Concat((s_{spec}, r_{tete} \bullet s_{tete}, s_{comp}))) : A}\,[mg]$$

$$\frac{\Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : A_< / B \qquad \Delta \vdash s : B}{\Gamma, \Delta \vdash (r_{spec}, \epsilon, r_{comp} \bullet Concat((s_{spec}, s_{tete} \bullet r_{tete}, s_{comp}))) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B_< \setminus A}{\Delta, \Gamma \vdash (Concat((s_{spec}, s_{tete} \bullet r_{tete}, s_{comp})) \bullet r_{spec}, \epsilon, r_{comp}) : A}\,[mg]$$

$$\frac{\Delta \vdash s : B \qquad \Gamma \vdash (r_{spec}, r_{tete}, r_{comp}) : B \setminus_> A}{\Delta, \Gamma \vdash (Concat(s_{spec}, r_{tete} \bullet s_{tete}, s_{comp}) \bullet r_{spec}, \epsilon, r_{comp}) : A}\,[mg]$$

D.4 Interface syntaxe-sémantique

D.4.1 $\lambda\mu$-calcul

Syntaxe du $\lambda\mu$-calcul :

$$T ::= x|\lambda x.\mathrm{T}|\mu\alpha\mathrm{T}|(T)\mathrm{T}$$
$$\mathrm{T} ::= T|(\alpha\mathrm{T})$$

$$\frac{}{x : N \vdash x : N|}\,[var] \qquad \frac{\Gamma, x : N \vdash u : M|\Delta}{\Gamma \vdash \lambda x.u : N \to M|\Delta}\,[abs]$$

$$\frac{\Gamma \vdash u : N|\beta : M, \Delta}{\Gamma \vdash \mu\beta[\alpha]u : M|\alpha : N, \Delta}\,[\mu] \quad \frac{\Gamma \vdash u : N \to M|\Delta \quad \Gamma' \vdash v : M|\Delta'}{\Gamma, \Gamma' \vdash (u)v : M|\Delta, \Delta'}\,[app]$$

Soit l'homorphisme de type $H : S_x \to S_e$ défini inductivement :
- $H(a\backslash b) = H(a/b) = H(a) \to H(b)$
- $H(a \otimes b) = H(b)$ si $a \in \mathrm{P}_2$, H(a) sinon.

D.4.2 $\lambda\mu$-DRS

Syntaxe des **$\lambda\mu$-DRS** :

$$\begin{array}{lll} \gamma & ::= & (P\ x)|(\alpha\ x)|x_1 = x_2|\neg K|K_1 \wedge K_2|K_1 \vee K_2|K_1 \Rightarrow K_2 \\ K & ::= & [x_1 ... x_n|\gamma_1, ..., \gamma_m] \end{array}$$

merge :

$$\frac{x : \Gamma \vdash A/B \quad y : \Delta \vdash B}{(x)y : \Gamma, \Delta \vdash A}\,[mg]$$

$$\frac{y : \Delta \vdash B \quad x : \Gamma \vdash B \setminus A}{(x)y : \Delta, \Gamma \vdash A}\,[mg]$$

Move :

$$\frac{x(c) : \Gamma \vdash A \otimes B \quad y[u, v] : \Delta[u : A, v : B] \vdash C}{y[c/u, x(c)/v] : \Delta[\Gamma] \vdash C}\,[mv]$$

Annexe E

Analyse avec une GM de la phrase “Je ne la lui donne pas”

Nous présentons l'analyse de la phrase :

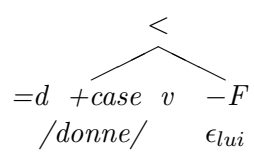
(79) Je ne la lui donne pas

qui utilise plusieurs clitiques ainsi que la négation.

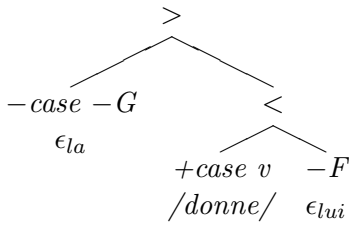
Dérivation 139. *Soit le lexique*

je	$= finclitique\ +Subj\ Nom$
la	$dat <=\ +G\ acc$
lui	$clitique <=\ +F\ dat$
donne	$= p\ = d\ +case\ v$
ϵ_{je}	$d\ -Subj\ -case$
ϵ_{lui}	$p\ -F$
ϵ_{la}	$d\ -case\ -G$
ϵ_{ne}	$neg_1 - Ne$
pas	neg_2
ne	$=> finclitique\ +Ne\ finclitique$
sortie de la cliticisation	$= Nom\ +case\ t$
inflexion	$=> neg_2\ = neg_1\ => little\_v\ verbe$
complément de la forme verbale	$=> v\ = d\ little\_v$
comp	$= t\ c$
sortie de la cliticisation	$=> acc\ finclitique$
entrée dans la cliticisation	$=> verbe\ clitique$

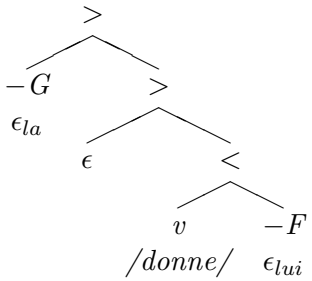
1. *entrée lexicale :* $[donne] :: [= p, = d, +case, v]$
   entrée lexicale : $[] :: [p, -F]$
   fusion : introduction de l'argument à phonologie vide du datif.



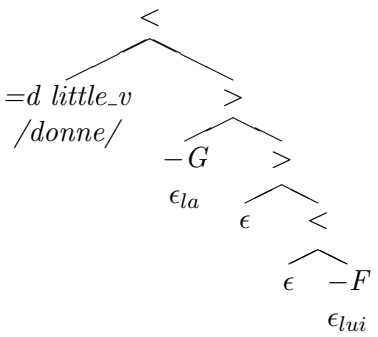
2. *entrée lexicale :* $[] :: [d, -case, -G]$
   fusion : introduction de l'argument à phonologie vide de l'accusatif.



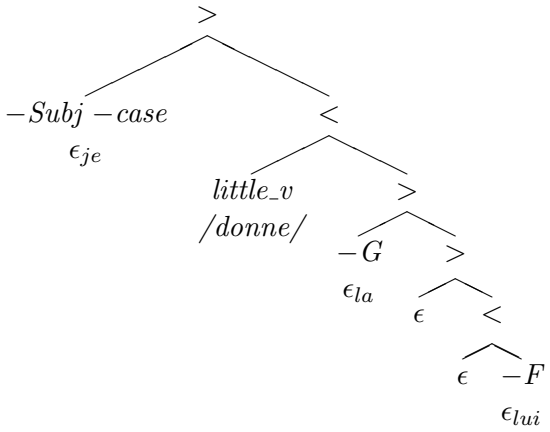
3. *déplacement : résolution du cas pour l'objet.*



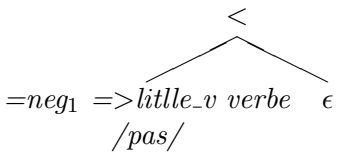
4. *entrée lecicale :* $[] :: [=> v, = d, little\_v]$
   fusion : introduction de la nécessité d'avoir un DP (sujet) pour le verbe.



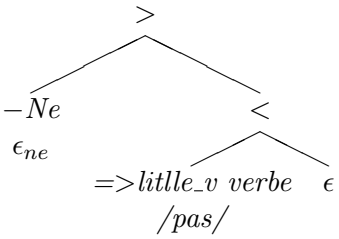
5. *entrée lexicale :* $[] :: [d, -Subj, -case]$.
   fusion : introduction de l'argument à phonologie vide du nominatif.



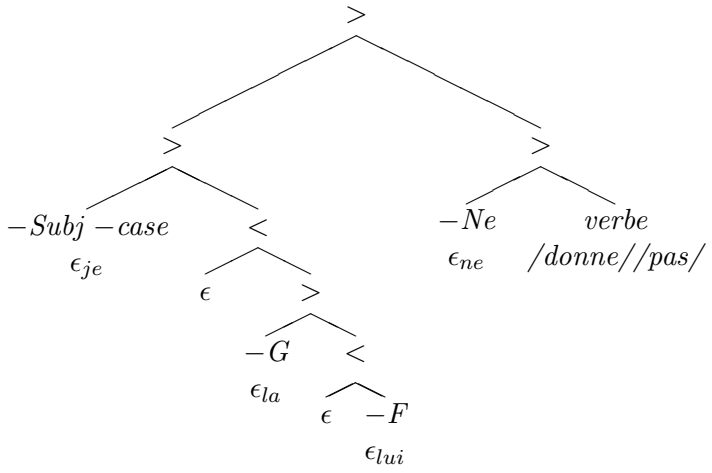
6. *entrée lexicale :* $[inflexion] :: [=> neg_2, = neg_1, => litlle\_v, verbe]$.
   entrée lexicale : $[pas] :: [neg_2]$.
   fusion : construction de la première partie de l'inflexion négative.



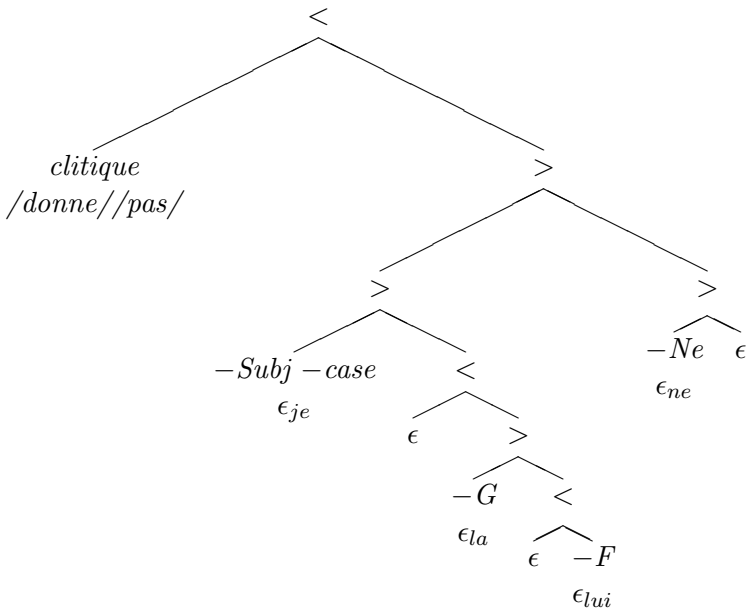
7. *entrée lexicale :* $[] :: [neg_1 - ne]$
   fusion : fin de la construction de l'inflexion négative.



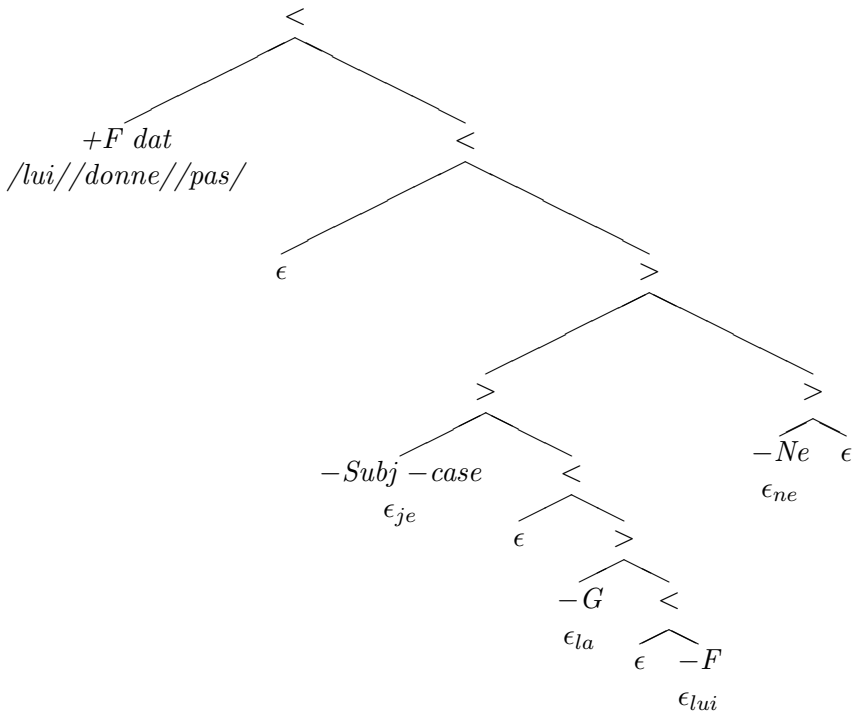
8. *fusion entre cette inflexion complexe et la dérivation sur le verbe. Nous utilisons une version simplifiée de la dérivation précédente pour des questions de taille des arbres.*



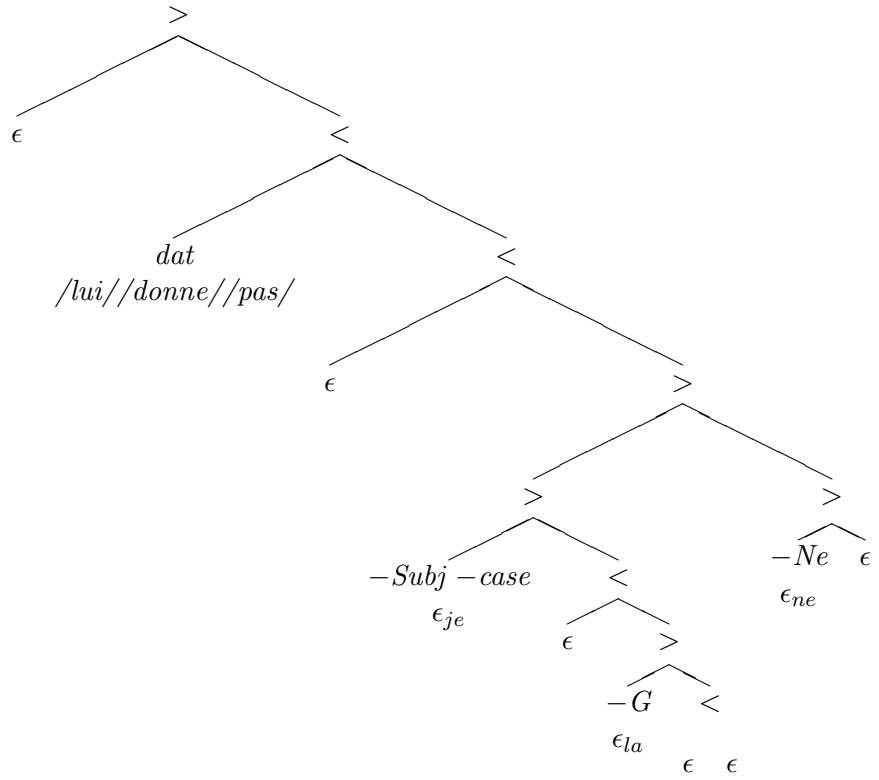
9. *entrée lexicale : [] : :[=>verbe, clitique].*
   fusion : entrée dans la cliticisation.



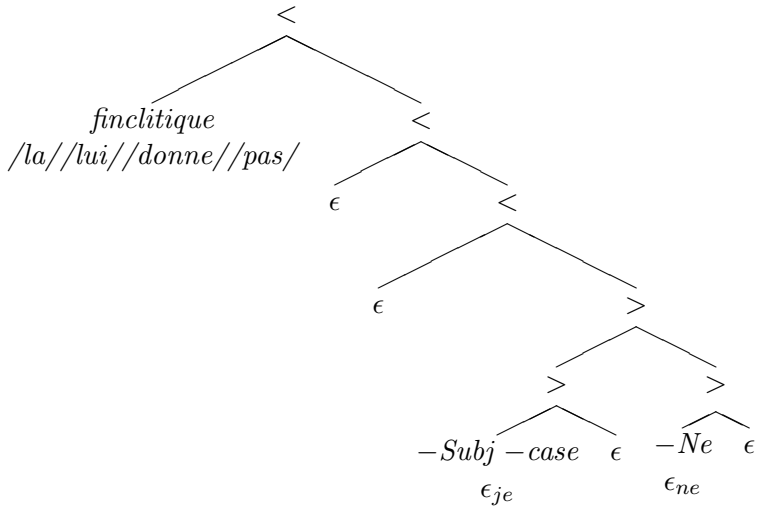
10. *entrée lexicale :* $[lui] :: [clitique <=, +F, dat]$.
    fusion : cliticisation du datif.



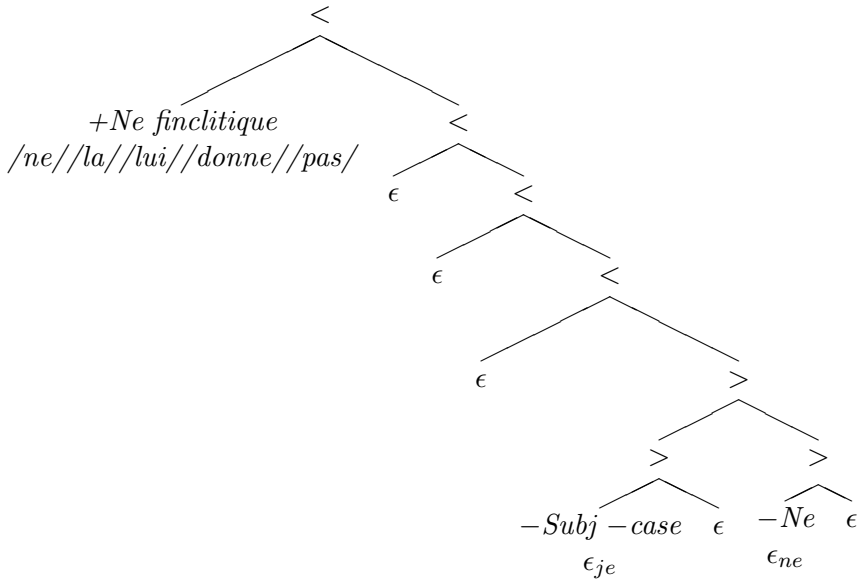
11. déplacement.



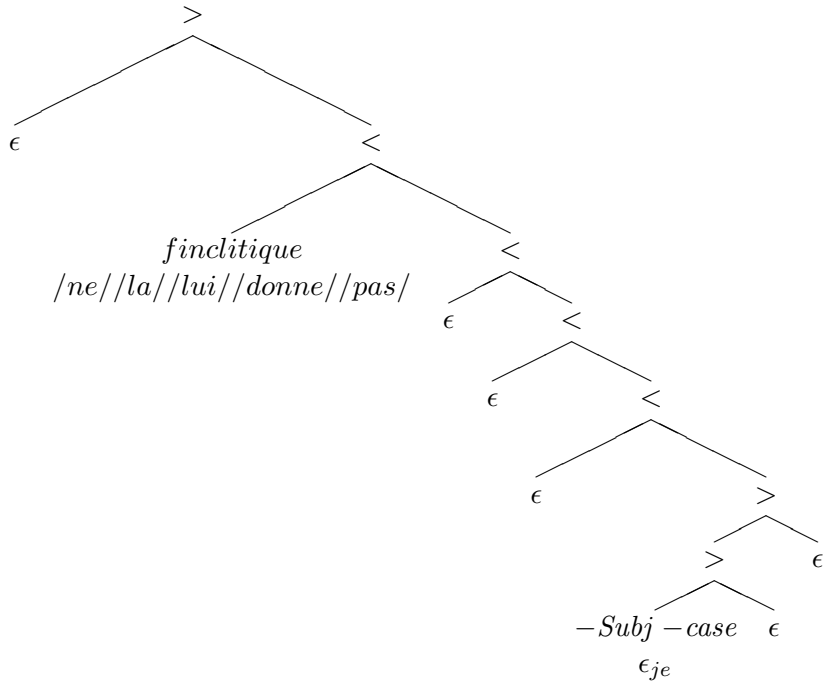
12. *Une fois de plus nous simplifions la représentation graphique dans les parties haute et basse de l'arbre. La clitisation de l'accusatif se fait de manière analogue :*
    entrée lexicale : $[la] :: [dat <=, +G, acc]$.
    fusion : cliticisation de l'accusatif.
    puis déplacement
13. *entrée lexicale :* $[] :: [=> acc, finclitique]$.
    fusion : vers la fin du cluster de cliticisation.



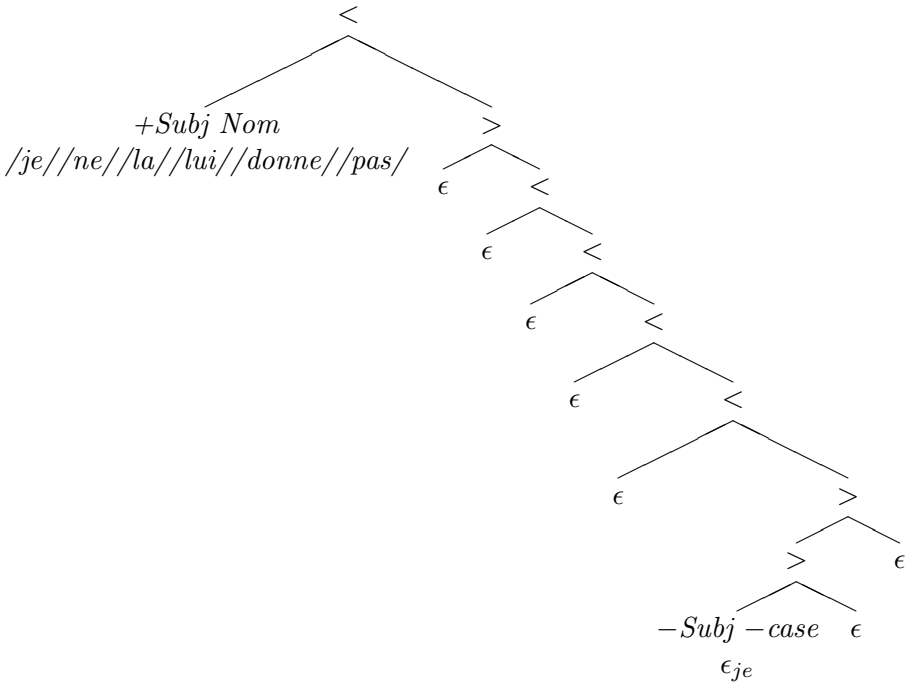
14. entrée lexicale : $[ne] :: [=> finclitique, +Ne, finclitique]$.
fusion : introduction du marqueur de négation.



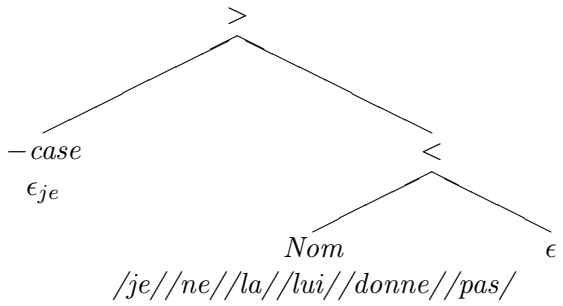
15. déplacement :



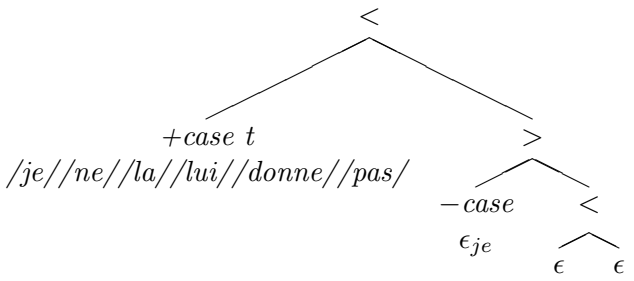
16. *entrée lexicale :* $[je] :: [= finclitique, +Subj, Nom]$.
fusion : introduction du clitique nominatif.



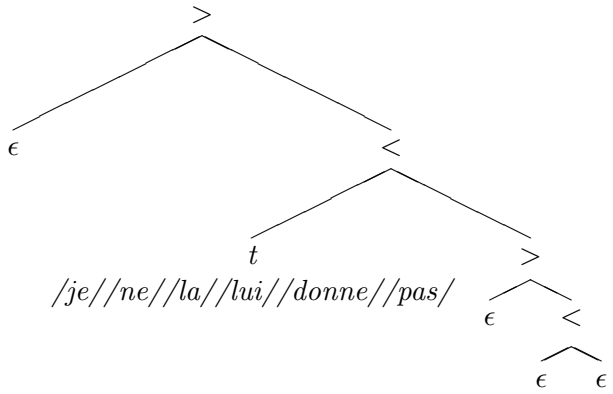
17. *déplacement :*



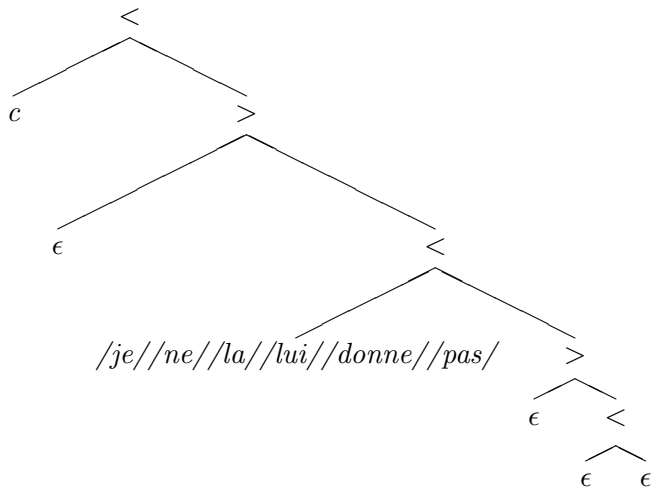
18. *entrée lexicale :* $[] :: [=> Nom, +case, t]$.
fusion : vers un verbe avec tous ses arguments.



19. *déplacement.*



20. *entrée lexicale :* $[] :: [= t, c]$.
21. *fusion : fin de la dérivation.*


$<$
c
$>$
ϵ
$<$
/je//ne//la//lui//donne//pas/
$>$
ϵ
$<$
ϵ
ϵ

Annexe F

Une modélisation de la Langue des Signes Française par les GMs

Sommaire

F.1 Présentation de la LSF 289
F.2 Cadre théorique 293
F.3 Modélisation 299
F.4 Exemple de dérivation 301

en collaboration avec Émilie VOISIN.

F.1 Présentation de la LSF

F.1.1 Historique

Avant de commencer notre étude, il nous faut replacer la LSF dans le contexte historique particulier qui lui est propre et qui conditionne à l'heure actuelle les études qui sont faites. Pour donner quelques repères, commençons avec l'Abbé de l'Épée (1712-1789), en 1760 qui est l'un des premiers entendants à s'intéresser à la LSF. Il reconnaît et se charge de diffuser l'information selon laquelle les signes peuvent exprimer la penser de la même manière qu'une langue orale. L'Abbé de l'épée crée une école à Paris et se charge de l'enseignement du français à l'aide de la LSF et invente, pour cela, une convention gestuelle qui a été abandonnée par la suite (car peu naturelle). Jusqu'en 1880, l'enseignement et la diffusion de la LSF était considéré comme incontournable par les différents acteurs du monde des sourds et des entendants. C'est à cette date qu'a eu lieu le Congrès de Milan réunissant des entendants militants pour l'oralisation. Le résultat de ce Congrès a eu pour conséquence l'interdiction de l'usage des signes dans l'éducation des sourds. En France, il a fallu attendre un siècle pour voir revenir l'usage de la LSF dans l'éducation des sourds (Loi Fabius, 1991). Ce contexte particulier explique assez bien pourquoi l'étude de la LSF est à ce point à ses prémisses. En effet, les études menées à l'heure actuelle en France s'intéressent plus précisément à l'étude sémiogénétique (recherche de l'origine de la création des signes), citons par exemple [Cux00]. L'approche syntaxique étant donc moins favorisée, nous avons choisi de nous intéresser d'une part à l'organisation syntaxique de

la phrase simple en LSF et d'autre part à la morphologie verbale. Nous verrons dans les lignes qui vont suivre l'importance de la flexion verbale cette dernière conditionnant en grande partie l'organisation de l'énoncé.

F.1.2 Particularités

La LSF, et plus généralement les Langues Signées (LS), possèdent des particularités linguistiques qui rendent la formalisation plus périlleuse. Les Langues Orales, LO, sont des langues auditivo-orales, par opposition aux LS qui sont visuo-gestuelles. Cette caractéristique, primordiale, se traduit bien dans une citation de Cuxac : "la LSF dit en même temps qu'elle donne à voir". Une des caractéristiques majeures accompagnant cette citation de Cuxac est bien entendu la notion d'iconicité. Nous ne traiterons pas de cette problématique ici tant elle est complexe, néanmoins cette particularité ne doit pas être oubliée lorsque l'on s'intéresse aux Langues Signées. De nombreuses caractéristiques sont liées à cet aspect visuel, d'une part, et à l'aspect gestuel d'autre part. Dans les parties qui vont suivre, nous allons présenter ces caractéristiques qui différencient radicalement les LS des LO.

Particularités gestuelles

Le signe que nous entendons ici comme étant l'articulation de la main en vue de produire du sens, se compose de différents paramètres, au nombre de cinq :

- la localisation du signe dans l'espace de signation ;
- le mouvement ;
- les expressions du visage ;
- la configuration manuelle (l'ensemble des configurations manuelles ont été listées de manière exhaustive)
- l'orientation, c'est-à-dire la direction vers laquelle le signe va être réalisé (orientation de la main).

Particularités visuelles

Il faut aussi tenir compte des expressions du visage (parfois appelées mimiques faciales) qui jouent un rôle fondamental dans la reconnaissance du signe (joues gonflées ou non ; yeux ouverts, fermés ou plissés ; sourcils levés ou froncés ; formes de la bouche ; etc). Donnons un exemple : Soit la paire : [SERIEUX] et [TRISTE] Ces deux signes ont exactement les mêmes paramètres de formation et seule va différer l'expression du visage [SERIEUX]sourcils froncés [TRISTE]coins de la bouche baissés

La gestion de l'espace est une particularité qui intervient dans la notion de linéarité. Même si ce constat peut être nuancé par des indices liés à l'intonation, au rythme prosodique, les LO sont dites linéaires. La gestion même de l'espace de signation, qui est en trois dimensions, dans lequel, à la manière d'une scène de théâtre, les objets du discours vont pouvoir être réalisés, mis en relation, en interaction les uns avec les autres. La conséquence est que de nombreux éléments du discours vont pouvoir se superposer, grâce à la gestion de l'espace et grâce aux expressions du visage.

F.1.3 Ordre des signes en LSF

Préalable : le principe d'économie d'après Voisin, 2005

Nous proposons ici un principe d'économie qui diffère de celui qui a été proposé par [BDP99]. En effet, ce qui attire notre attention dans le principe d'économie tel que nous le décrivons, ce sont les aspects syntaxique et morphologique. Morphologique d'une part car la reprise de la proforme va modifier la racine du verbe ; syntaxique d'autre part car comme nous le verrons dans les exemples qui vont suivre, la flexion engendrée va modifier profondément l'ordonnancement des signes dans la phrase simple. En effet, lorsque la proforme utilisée pour former l'un des actants du verbe est reprise par ce dernier, nous constatons qu'une relation étroite se crée. Le verbe et le nom ont donc tendance à rester inséparables. C'est en ce sens que nous envisageons la notion d'économie et cette flexion contraint fortement l'organisation syntaxique : par exemple, pour la phrase : le garçon (S) mange (V) une pomme (O) (que nous simplifions par SVO), nous aurons, en LSF, une reprise de la configuration manuelle utilisée pour signer "pomme" (configuration C) dans le verbe "manger". En conséquence de quoi, "pomme" et "manger" vont fonctionner comme un amas. Nous aurons donc l'ordre préférentiel : SOV (garçon - pomme - manger).

Organisation syntaxique

Mais ce n'est pas la seule raison d'influence sur l'organisation syntaxique. Dans le cadre des unités lexicales morphologiquement marquées, les marques (proforme) sont des clitiques renvoyant aux actants des verbes. La relation étroite entre ces clitiques et les actants du verbe rend possible la liberté de l'ordre dans lequel ces derniers sont énoncés. A contrario, l'unité lexicale non marquée est construite avec la configuration de base et des ancrages spatiaux neutres. Dans ce cas, c'est l'ordre contraint des actants autour du verbe qui permettra d'établir la relation entre ces derniers. En effet, l'étude des langues orales telles que le basque et l'allemand nous montre que plus les flexions sont nombreuses, plus l'ordre des mots tend à être variable, voir [Gre63] ; [Com95]. Donc, quand ces flexions sont absentes (non marquage), l'ordre des signes est nécessairement contraint pour qu'il n'y ait pas d'ambiguïté possible. [PBD+04]) reprend cette idée selon laquelle l'ordre des signes peut être relativement libre dans la mesure où "les relations entre les signes peuvent être établies autrement que par la séquentialité et l'aspect fonctionnel de l'ordre y est donc beaucoup moins important".

Nous allons examiner dans la partie qui va suivre des exemples divers, recueillis avec des locuteurs sourds natifs, nous permettant d'examiner les divers ordres que 'on peut trouver dans une phrase simple en LSF.

(80) Exemple fourni par le locuteur 1 :

L'ordre présenté dans cette réalisation peut être qualifié d'ordre prototypique (c'est- à-dire qui est généralement attendu). Plusieurs arguments viennent justifier cette assertion. En Langue des Signes Française, il est fréquent que le prédicat verbal soit repoussé en fin d'énoncé. Plus particulièrement, et pour nous attacher plus précisément à la description de notre exemple, [MANGER] est un verbe semi-rigide (d'après [Par03]), c'est- à-dire que sa forme va varier uniquement en proforme (configuration manuelle) ou en loci (lieux de réalisation situés dans l'espace de signation). [MANGER] peut fléchir uniquement en

proforme : il peut prendre le classificateur utilisé par un de ses arguments présent dans sa valence. Pour l'exemple qui nous intéresse, [MANGER] va s'accorder avec [POMME] qui porte dans sa réalisation le classificateur "C" (qui désigne des objets petits et ronds). En vertu du principe d'économie [Voi05] qui contraint fortement l'ordre des signes, [POMME] va se trouver à proximité du prédicat verbal [MANGER]. Le principe d'économie reflète l'iconicité de l'énoncé grâce à la fluidité entre les concepts lorsque le proforme a la même forme que le nom et a donc pour conséquence une reprise de la configuration manuelle qui se retrouve sous la forme de proforme dans le prédicat verbal :

[GARçON] [POMME]conf. C [MANGER]FLEXION : conf. C

(81) Exemple fourni par le locuteur 2 :

[GARçON] [MANGER]FLEXION : conf. C [POMME]conf. C

Cette réalisation est une variante de celle que nous avons décrite précédemment. L'ordre syntaxique correspondrait à celui du Français Signé, néanmoins, le principe d'économie est respecté (voir partie sur la flexion verbale).

(82) Exemple fourni par le locuteur 3 :

[POMME]conf. C [GARçON] [MANGER]FLEXION : conf. C

Dans cet exemple, par rapport à ce que nous avons décrit comme étant l'ordre syntaxique prototypique (locuteur 1), il s'agit ici de la transcription de la réalisation d'un locuteur ayant eu certaines intentions communicatives particulières. En effet, le fait de placer [POMME] en tête d'énoncé entraîne une focalisation particulière sur le segment [POMME] (c'est de la pomme/cette pomme qu'un garçon mange).

(83) Exemple fourni par le locuteur 4 :

[GARçON] [MANGER] [POMME]conf. C

Cette réalisation ne diffère de celle du locuteur 2 que par l'absence de flexion. Nous traiterons de ce problème de flexion dans la partie qui va suivre.

Flexion verbale

Généralement, les verbes en LSF sont flexionnels. Néanmoins, et d'après les travaux de [Par03], nous pouvons distinguer (dans les grandes lignes) trois catégories de verbes :

- les verbes dits "souples", ce sont des verbes qui peuvent se modifier en fonction des actants (en loci (lieu d'articulation) et en proforme (configuration manuelle), par exemple le verbe "donner" dont les lieuxx de réalisation peuvent changer et dont la configuration va varier en fonction de ce qui est donné ;
- les verbes dits "semi-rigides" qui ne vont varier qu'en loci ou qu'en proforme, nous trouvons dans cette catégorie le verbe "manger" qui ne peut se modifier qu'en proforme (configuration) ;
- les verbes "rigides" qui restent dans leur forme neutre quels que soient les actants (par exemple le verbe "s'inquiéter").

Par flexion, nous entendons [Mul02] "prédicat particulier associé morphologiquement au verbe". Cette flexion verbale associe des informations de nature diverse : informations modale, temporelle ainsi que les marques d'accord avec tel ou tel actant. Précisons qu'en LSF, la flexion verbale est entendue principalement comme les marques d'accord qui existent entre le verbe et ces actants, les marques modale et temporelle étant exprimées par des

morphèmes indépendants généralement placés en tête de phrase. La première chose dont nous ayons besoin avant même de commenter les exemples est de savoir ce qu'est la flexion verbale en Langues des Signes Française. Les états constitutifs de la réalisation d'un signe (et même pour ce qui nous intéresse d'un prédicat verbal) en Langue des Signes Française sont constitués de paramètres. Certains ne varient pas (racine verbale) et d'autres observent une variation syntaxique. Les paramètres pouvant varier deviennent des morphèmes. Seule une réalisation mérite notre attention, la réalisation effectuée par le locuteur 4. Comme nous l'avons commenté précédemment, en plus de l'organisation syntaxique particulière (calquée sur le Français), on peut remarquer l'absence de flexion sur [MANGER]. Cette absence, couplée à l'ordre des signes, nous montre que nous sommes en face d'un exemple de Français Signé, éliminant toute difficulté pouvant gêner l'intercompréhension. Le principe d'économie n'est pas respecté et cela se remarque par le fait que le verbe [MANGER] est un verbe soumis à la flexion puisque classé dans la catégorie des verbes à forme semi-rigide.

Incorporation nominale

Dans cette étude, nous allons examiner plus particuliérement le phénoméne d'incorporation nominale, à partir des travaux menés par [Laz94] et [Bak88]. En effet, suite à un travail récent sur la flexion verbale [VKre] nous avons constat é que la flexion verbale pouvait se rapprocher du phénoméne de coalescence [Laz94] et plus particuliérement de l'incorporation nominale, décrite (entres autres) par [Bak88], tout comme nous pouvons le voir dans de nombreuses études sur les langues amérindiennes. Par exemple en nahuatl où le nom (l'actant verbal) est inséré entre le préfixe actanciel et la racine verbale. Les caractéristiques de l'incorporation nominale résident dans la perte de la fonction actancielle. Le plus fréquemment, la fonction de ce nom est objet. En LSF, nous avons constaté dans les exemples étudiés qu'il s'agit en effet des objets qui sont ordinairement intégrés au verbe. On constate aussi que les instrumentaux peuvent être intégrés au verbe ("couper avec un couteau" par exemple").

F.2 Cadre théorique

F.2.1 Les grammaires minimalistes

Les grammaires minimalistes (GMs) ont été proposées par Stabler, [Sta97]. Elles implémentent le programme minimaliste de Chomsky, [Cho95]. Cette théorie linguistique veut expliquer, dans la perspective de la théorie générative, comment les analyses syntaxiques sont réalisées. Ce cadre dérivationnel se prête aisément à une formalisation. Afin de lever toute ambiguïté, la problématique minimaliste est à entendre du point de vue général de la théorie et non dans le résultat effectif.

Le postulat du programme minimaliste est d'associer une forme logique à un son et ce par un calcul syntaxique qui dirige l'analyse. Le calcul sémantique n'étant pas l'objet d'étude, nous renvoyons le lecteur à d'autres publications l'exposant, notamment [Amb07]. Le calcul *phonologique* est un étiquetage par les formes phonologiques des feuilles de l'arbre obtenu en fin de dérivation, appelé *arbre dérivé*. L'analyse est basée sur la théorie *X-barre* (relations spécifieur/tête et tête/complément) et elle est réalisée à partir de deux

opérations : la fusion et le déplacement.

Le système calculatoire est entièrement basé sur la notion de traits. À partir de ces derniers, des règles de composition sont définies afin d'établir les structures grammaticales. Le résultat est alors un arbre. Les arbres obtenus sont des *arbres minimalistes* qui possèdent trois relations : les deux relations usuelles pour définir la notion d'arbre (*dominance* et *précédence*) et la relation de projection qui est notée $<$ et $>$ dans les noeuds. On note T_{MG} l'ensemble de ces arbres. Les traits sont associés à des items lexicaux (nous reviendrons sur la rédaction et l'utilisation des lexiques dans la suite). Les différents types de traits dénotent l'utilisation linguistique qui est faite des items lexicaux dans ces grammaires.

Les étapes des dérivations réalisées par les GMs sont déclenchées par le premier trait des entrées lexicales. Ces grammaires sont entièrement lexicalisées et, de fait, définies par la donnée de leur lexique. Les règles de composition des expressions formées sont, quant à elles, toujours les mêmes.

Une **grammaire minimaliste** (GM) est définie par un quintuplet $\langle V, Traits, Lex, \Phi, c\rangle$ où :

- V l'ensemble fini des traits non-syntaxiques qui se décompose en P l'ensemble des formes phonologiques et I l'ensemble des formes logiques.
- $Traits = \{B \cup S \cup L_a \cup L_e\}$ l'ensemble fini des traits syntaxiques,
- Lex est l'ensemble des expressions construites à partir de P et de $Traits^*$ (les items lexicaux),
- $\Phi = \{merge, move\}$ l'ensemble des fonctions génératrices,
- $c \in Traits$ est un trait distinctif, permettant de reconnaître les dérivations acceptantes.

Dans ces grammaires, les formes phonologiques sont utilisées comme entrées du lexique et comme formes associées aux listes de traits. Elles constituent les "terminaux" de la grammaire. Une lecture gauche-droite des formes phonologiques sur les structures dérivées et acceptées fournit la séquence de terminaux reconnue.

Le langage reconnu par G une grammaire minimaliste ($L(G)$) est la clôture du lexique par les fonctions génératrices ϕ. À tout énoncé accepté par une GM correspond un arbre minimaliste obtenu à partir des règles de composition. Nous reviendrons sur les définitions des règles dans la suite. Leur fonctionnement permet, pour une phrase d'une langue naturelle, d'obtenir un arbre d'analyse proche des arbres génératives traditionnels.

Les traits

Une GM est définie par un lexique qui stocke les ressources. On associe à chaque entrée du lexique une liste de traits qui encode son comportement lors d'une dérivation. Une GM contient des traits de deux sortes : **les traits syntaxiques** et **les traits non-syntaxiques**.

On note V l'ensemble des traits non-syntaxiques composé de :
- **traits phonologiques** (ou forme - FP) notés entre barres obliques - / /.
- **traits sémantiques** (ou forme - FL) notés entre parenthèses - ( ).

L'ensemble des traits syntaxiques est composé de deux sous-ensembles : l'ensemble des catégories de base, noté B et l'ensemble des traits de déplacement, noté D. En utilisant

ces sous-ensembles, on définit les différents types de traits utilisés dans les listes des items lexicaux de la grammaire :

- soit $B = \{v,\ dp,\ c, \cdots\}$ l'ensemble des **catégories de base**,
- soit $S = \{=d \mid d \in B\}$ l'ensemble des **sélecteurs**,
- soit $L_a = \{+k \mid k \in D\}$ l'ensemble des **assignateurs**,
- soit $L_e = \{-k \mid k \in D\}$ l'ensemble des **assignés**.

Revenons sur le rôle et la signification de chacun de ces éléments.

Lorsque l'on modélise une langue naturelle, les éléments de B dénotent des concepts linguistiques standards, par exemple v pour un verbe, n pour un nom, p pour une préposition, ... Dans cet ensemble, on distingue un type particulier appelé **trait acceptant**, qui sera le symbole acceptant de la grammaire. Généralement on utilise le trait c qui représente la position "complementizer" de la dérivation, c'est à dire l'état pendant lequel on vérifie la terminaison de la phrase.

Les *sélecteurs* expriment une demande par rapport à une autre expression possédant le trait de base "équivalent". Si α est un trait de base, $=\alpha$ est un sélecteur. Il exprime la demande d'une expression possédant le même trait α.

Les *assignateurs* sont les traits qui assignent une propriété à une expression et qui sont dans une relation spécifieur-tête par rapport à celle qui les porte. À nouveau, lors de la modélisation d'une langue naturelle, les assignateurs sont utilisés pour apporter une propriété à une autre expression (par exemple *le cas* pour les langues naturelles). Cette dernière vient occuper une nouvelle place en relation spécifieur/tête par rapport à elle.

Parallèlement, l'expression recevant le trait assigné doit être appropriée, autrement dit, elle doit demander à recevoir ce trait. On traduit cela par le fait qu'elle possède le trait complémentaire de $+f$ qui est noté $-f$. Pour le traitement d'une langue naturelle, le correspondant du trait $+cas$ sera $-cas$, possédé uniquement par les groupes nominaux pour répondre à l'hypothèse selon laquelle ils *doivent nécessairement recevoir un cas.*

L'union de ces ensembles forme l'ensemble des traits syntaxiques de la grammaire utilisés pour modéliser le comportement syntaxique des items lexicaux $Traits = \{B \cup S \cup L_e \cup L_a\}$.

À partir de ces ensembles, on définit la structure d'une *entrée lexicale*, pour des GMs ne prenant pas en considération la sémantique, comme une liste de la forme :

$$/FP/ : (S(S \cup L_a)^*)^* B(L_e)^*$$
$$\text{ou bien} \quad /FP/ : B(L_e)^*$$

Ces listes peuvent être reconnues par un automate régulier donné en figure 55. Dans cette structure, on distingue deux parties, la première contenant des sélecteurs et des assignateurs, traits déclenchants les règles (comme nous allons le voir), puis un trait de base (la catégorie du constituant construit) et des assignés, traits attendant d'être composés dans la suite de la dérivation (les propriétés devant être interprétées dans la suite de la dérivation).

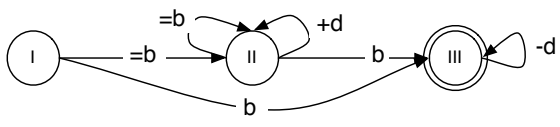


Fig. 55 – Structure de la liste de traits d'un item lexical

Les règles des GMs

Le résultat fourni par les règles est un arbre binaire dont les noeuds contiennent la relation de projection et les feuilles les listes de traits restant dans la séquence de la dite feuille. La structure d'arbre utilisée est donc : $\tau = (N_\tau, \lhd_\tau^*, \prec_*^\sim, <_\tau^\sim)$. Toutes ces relations sont détaillées dans [Amb07].

On appelle **tête** d'un arbre, l'élément minimal pour la relation de projection. Il est aisé de déterminer cet élément en suivant le sens indiqué par la relation de projection dans les noeuds des arbres minimalistes. On appelle **projection maximale** d'une feuille f le plus grand sous-arbre dont f est la tête. Pour la suite des définitions, pour t un arbre minimaliste et t' un sous arbre de t, on note $t - t'$ l'arbre t privé du sous-arbre t'. Enfin, pour un arbre minimaliste t, on note t_{-1} cet arbre dont le premier trait de la tête est effacé.

La fusion (merge)

La *fusion* est une opération qui unit deux arbres pour en former un troisième. Elle est déclenchée par la présence d'un sélecteur en première position de la liste de traits d'un arbre et d'un trait de base correspondant en première position de la liste de traits d'un autre arbre. Les traits utilisés pour cette opération sont alors effacés. La position spécifieur/tête ou complément/tête est définie par la lexicalité de l'arbre portant le sélecteur. $merge$: $T_{MG} \times T_{MG} \rightarrow T_{MG}$

Soient t et $t' \in T_{MG}$ tels que le premier élément de la tête de t est $=h\ E$ et celui de la tête de t' est le trait $h\ E'$ avec $h \in B$:

$$merge(t, t') = \begin{cases} < (t_{-1}, t'_{-1}) & \text{si } t \in Lex, \\ < (t'_{-1}, t_{-1}) & \text{sinon.} \end{cases}$$

La représentation graphique de cette règle est donnée par la figure 56. La fusion est l'opération qui met en relation les différentes expressions construites au fur et à mesure de la dérivation. La tête du nouvel arbre pointe vers l'expression portant le sélecteur.

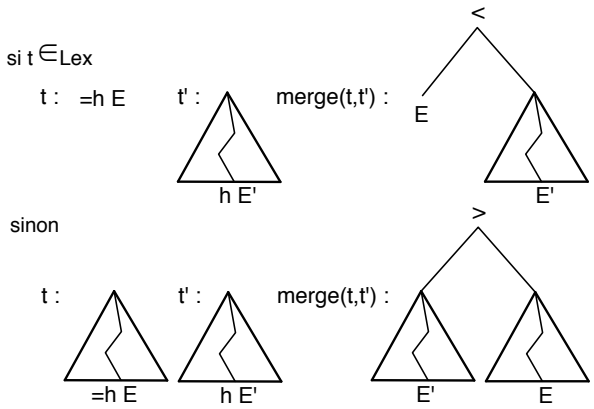


FIG. 56 – Représentation sous forme d'arbre de la fusion

Le déplacement (move)

Cette seconde opération est primordiale dans la formalisation du programme minimaliste. Elle correspond au déplacement effectif d'un constituant en première position de

la dérivation, c'est-à-dire en haut de l'arbre dérivé. Elle réalise une restructuration d'un arbre minimaliste. La présence simultanée d'un élément de L_a en première position de la liste de traits de la tête et d'un élément de L_e équivalent en première position d'une liste de traits d'une des occurrences du même arbre déclenche cette opération.

De manière intuitive la procédure est la suivante : lorsque le premier trait de la tête d'une dérivation est un assignateur $(+)$, on cherche dans le reste de la dérivation une feuille dont le premier trait est l'assigné $(-)$ équivalent. Si on en trouve un, on déclenche un déplacement en faisant passer en haut de l'arbre la projection maximale de la feuille portant l'assigné. La tête de la nouvelle expression reste celle de l'expression avant déplacement.

$move : T_{MG} \rightarrow T_{MG}$

pour tout arbre t dont la tête est $+g\ E$, s'il existe une feuille $-g\ E'$ et soit t' la projection maximale de cette feuille :

$$move(t) = >(t'_{-1}, t_{-1} - t'_{-1})$$

Le sous-arbre est alors en relation spécifieur-tête. Les deux traits ayant permis le déplacement sont alors supprimés. La représentation graphique des règles est présentée dans la figure 57.

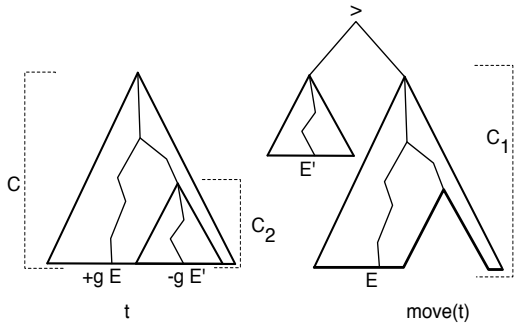


Fig. 57 – Représentation sous forme d'arbre du déplacement

Tous les déplacements ne sont pas envisageables. On ajoute (ou non) des conditions sur les déplacements possibles. Une condition majeure dans la définition même du programme minimaliste est la condition d'économie. Ainsi, en cas d'ambiguïté, le déplacement doit avoir lieu sur l'élément "le plus proche" de la tête. Nous utilisons une condition forte sur le principe d'économie (qui l'englobe) pour laquelle il ne doit pas y avoir d'ambiguïté dans le déclenchement d'un déplacement - Shortest Move Condition (SMC). Dans ce cas, chaque dérivation est déterministe.

Une condition de localité peut aussi être utilisée, la Specifier Island Condition - SPIC. Les "islands" définissent les domaines qui interdisent des extractions. Elles imposent que pour être déplacé, un élément ne soit pas en position de spécifieur à l'intérieur d'un sous-arbre. Cette condition a été introduite par Stabler dans [Sta99] en s'inspirant des travaux de [KS00] et [Kay98] qui proposent que les éléments déplacés soient uniquement en position de complément.

F.2.2 Signème

Après avoir présenté le cadre théorique dans lequel nous allons opérer notre étude, celui des grammaires minimalistes, examinons plus en détail la manière dont la théorie de l'incorporation ainsi que les modifications syntaxiques engendrées peuvent venir s'y inscrire. Nous posons une notion centrale de notre étude, la notion de signème. Les signèmes sont les représenations de surfaces de la morphologie des signes. D'après la théorie proposée précédemment, nous souhaitons accumuler autour d'un élément des informations provenant d'autres parties de l'analyse qui en dépendent (par exemple les proformes des signes associés).

Plusieurs solutions ont été envisagées. La première a été l'utilisation de la notion de *copie*. En général, l'introduction de copie dans un systéme complexifie considérablement la partie calculatoire. Or, la base des théories de modélisation des langues naturelles est l'hypothèse que nous analysons les énoncés en temps polynomial. Toute complexification doit donc être fortement remise en cause. La seconde, celle que nous avons retenue, est que les informations nécessaires à la réalisation d'un constituant sont apportées à un autre moment de l'analyse que la réalisation terminale.

Dans le système que nous utilisons ici, l'opération de déplacement permet de passer d'une structure profonde à une structure de surface. On suppose que les éléments nécessaires à la réalisation en signes d'un constituant sont ceux **dans la projection maximale** de ce constituant (dont il est le régisseur). Si les éléments dont il est dépendant ont été déplacés, **leur trace** (marque laissée par le déplacement) contient cette information.

On appelle **signème** d'un constituant la liste composée des réalisations de surface ainsi que des traces dans la projection maximale de la tête du constituant. Comme les traces possèdent les informations relatives à la réalisation en signe de l'élément qu'elles reprennent, on utilise le signème pour calculer le signe du constituant.

Ainsi, pour construire le signe associé à un verbe, nous devons connaître toutes les informations présentes dans la phrase pour calculer son signe. Revenons sur l'exemple que nous utiliserons dans cet article :

(84) Le garçon mange la pomme.

Il existe un signe générique pour la réalisation de "manger" (signe dans la périphérie de la bouche avec mouvement vertical). Le signe associé à "manger" dans cet énoncé doit utiliser la proforme de la pomme (position de la main en forme de petite sphère). C'est la combinaison de ces deux propriétés qui permet la réalisation du signe associé au verbe.

Dans les analyses des GMs, nous supposons que les *DP*s doivent "recevoir un cas". Cette propriété est la plus simple réalisation de la vérification de traits associés aux constituants par l'opération de déplacement. En faisant varier les types de traits dans les lexiques, ce formalisme nous permet de donner une implémentation des propriétés de "principes et paramètres". Ainsi, la définition du type de déplacement permet d'obtenir des énoncés dont la structure est différente. Traditionnellement, on suppose que les éléments déplacés dans les analyses minimalistes laissent dans leur position d'origine une *trace* de leur passage.

Lors d'une analyse, le verbe commence par recevoir son objet sur sa droite (position de complément selon la théorie *X-barre*) puis son sujet sur sa gauche (position de spécifieur). Cependant, ces deux *DP*s ne sont pas encore dans leur position finale puisque la dérivation

doit encore vérifier leurs traits (et au moins la donation de cas). Ainsi, dans la structure argumentale du verbe, on construit la structure :

(85) Sujet Verbe Objet

puis la dérivation déplace les DPs pour former la séquence (nous présenterons dans la suite un exemple complet d'analyse). Ainsi, ils occupent d'abord une première place située dans la structure argumentale du verbe puis ils sont déplacés :

(86) Sujet Objet $\underline{\mathrm{t}_{Sujet}\ \text{Verbe}\ \mathrm{t}_{Objet}}$

Autour du verbe on conserve donc les traces des différents DPs. La trace reste donc dans la porté du verbe et elle possède les conditions de réalisation relatives à cet élément. La partie de l'analyse mise en avant par le crochet présente les différentes composantes du signème du verbe. Les limites du signème sont aisément calculables à partir de la notion de projection maximale. C'est l'analyse de cette séquence qui permet de produire le signe du verbe avec par exemple la proforme de l'objet si celle-ci est plus prégnante que celle du sujet.

L'introduction de la notion de signème est due à la problématique des relations fortes entre éléments de la phrases. La réalisation des signes est dépendante du nombre d'éléments présents dans la phrase (mais pas tous). Ainsi, seuls les éléments nécessaires dans la structure argumentale auront une influence sur la réalisation en signes du constituant. Cette réalisation permet de ne pas contraindre la suite de l'analyse pour les constituants utilisés, tout en s'intégrant élégamment dans les GMs.

F.3 Modélisation

La question de la modélisation des Langues Signées à partir des GMs a été introduite dans [Chu06]. Ces travaux posaient la question de l'analyse des questions dans la Langue des Signes Américaine (ASL). Nous dirigeons notre approche vers des analyses complètes et automatisées d'énoncés simples.

Les analyses produites par les GMs supposent une structure d'analyse normalisée. Par exemple, pour les phrases du français, cette structure est celle de la *deep-structure* dans la théorie GB. Elle correspond à la structure argumentale du verbe.

C'est à partir de cette construction et des informations supplémentaires présentes dans la phrase apportées par l'extraction lexicale que l'analyse autorise ou non des réalisations différentes.

Une première analyse des phrases signées nous conduit à proposer une typologie de l'ordre des signes dans la réalisation d'une phrase. En règle générale, les énoncés commencent par donner les informations spécifiant la situation : les marqueurs de **temps** et d'**espace**.

Dans le corpus de phrases utilisé, et lors de l'enseignement de cette langue, il se dégage que la spécification temporelle est le premier marqueur qui apparaît. Il est difficile de différencier si la marque temporelle correspond ou non à la flexion verbale. Pour le moment, nous supposons que la réalisation temporelle et la flexion sont deux parties différentes du calcul. Cependant on remarquera que le marquage temporel n'intervient pas en lui même dans la construction du signe, le donner en première position de la séquence de signes permet de limiter la distance des dépendances entre les autres signes.

La réalisation spatiale est une information nécessaire pour pouvoir construire la séquence de signes. Dans de nombreux exemples, la réalisation d'un signe est fortement dépendante de la position dans l'espace des actants de la scène. Or le positionnement des actants nécessite de définir la scène créée.

Une fois que la situation est marquée spatio-temporellement, on présente les actants puis l'action elle même. Leur ordre d'apparition varie en fonction de la réalisation des signèmes. Si un signe dépend d'une proforme particulière, la distance de réalisation du signe correspondant répondra à un critère d'économie. Les actants peuvent être réalisés dans un ordre non défini. Cependant, l'analyse des phrases du corpus montre qu'en cas de non utilisation du critère d'économie, l'ordre sujet-objet-verbe que nous avons présenté précédemment est privilégié.

Nous proposons donc la typologie suivante :

Tems — Espace — Actants — Action

Dans les analyses produites par les GMs, nous distinguons plusieurs états particuliers du verbe, c'est-à-dire que nous supposons que le verbe est lexicalement du type v. Puis il est combiné avec une entrée particulière à phonologie vide qui lui attribue un **comportement syntaxique**. Ce dernier spécifiera combien d'arguments doit prendre cette réalisation du verbe. La distinction entre verbes transitifs ou intransitifs doit être faite par une distinction lexicale.

Puis, lorsque le verbe a reçu tous ses arguments, nous supposons qu'il est modifié par une entrée représentant **l'inflexion**. Elle permet de spécifier certaines propriétés sur le temps du verbe, sur les règles des groupes nominaux utilisés, *etc.* Ce stade est le point permettant la cliticisation en français, comme présenté dans [Amb06].

Et enfin, lorsque toutes les transformations induites par l'inflexion sont opérées, nous utilisons un dernier stade dans l'évolution de la catégorie verbale qui est appelée **complementizer**. Sa fonction est de vérifier le non enchassement dans une relative ou encore la spécification du type question pour la phrase. C'est cette entrée qui introduit dans la dérivation la catégorie acceptante pour l'analyse. Le problème est alors que la dérivation annule tous les traits syntaxiques introduits.

La succession de catégories verbales minimales pour une dérivation est donc : "comportement syntaxique - inflexion - complémentizer". On remarquera que cette définition des analyses est proche de la notion de phase qui est introduite dans la théorie générative. Nous proposons des contreparties à ces états particuliers lors de l'analyse d'énoncé en Langue des Signes Française .

Le comportement syntaxique permet au verbe de recevoir ses différents arguments. Chacun possède des traits qui doivent être utilisés dans la suite de la dérivation. Les DPs nécessitent au moins une attribution de cas. Celle de l'objet est réalisée dans cette phase de la dérivation par un déplacement.

La position d'inflexion est la réalisation contextuelle, c'est-à-dire qu'à ce point de l'analyse, le verbe a pris tous ses arguments (le signème relatif au verbe est alors complet). Après cette étape, le sujet peut recevoir son cas et l'objet est à nouveau déplacé au dessus de l'inflexion. L'ordre de ces déplacements permet d'obtenir les séquences objet-sujet-verbe ou sujet-objet-verbe. Le traitement de ces deux groupes nominaux est alors terminé. Nous

rappelons qu'à ce moment de l'analyse, les marqueurs de temps et d'espace sont toujours dans la première structure calculée.

Pour une analyse de phrase affirmative simple, l'étape de complementizer n'apporte pas d'information propre. Elle a pour fonction de clore le traitement en déplaçant les marqueurs de temps et d'espace. Ces éléments sont donc des spécifieurs du verbe (et au delà, de la phrase).

La figure 58 présente sous forme d'arbre les différents stades d'une dérivation qui suit cette description pour l'analyse des phrases affirmatives simples. On remarquera que ce type d'analyse respecte la typologie des signes dans un énoncé en Langue des Signes Française présentée dans la section précédente. En particulier, la dernière phase de la dérivation ordonne les marqueurs de temps et d'espace, alors que la phase précédente a ordonné les actants de l'énoncé. La première phase a pour rôle de rassembler tous les éléments de la phrase ensemble, ce qui permet d'élaborer le signème du verbe.

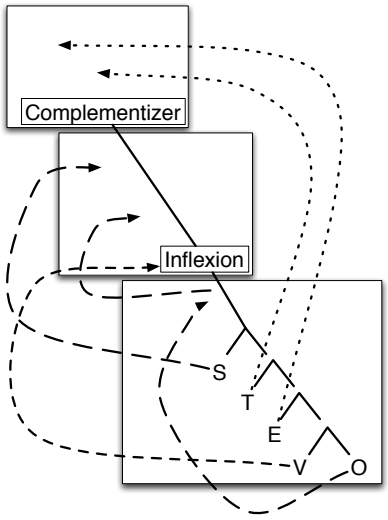


Fig. 58 – Structure d'une analyse en langue des signes française.

F.4 Exemple de dérivation

Nous allons maintenant présenter un exemple de dérivation pour l'énoncé (84) qui est une phrase affirmative simple. Ainsi nous verrons comment les différents phénomènes présentés sont mis en oeuvre dans ce calcul.

Pour l'analyser, nous utilisons le lexique suivant :

le :	=n d -case -p	*manger* :	v
la :	=n d -case -p	ϵ_{verbe} :	=¿v =tmps = lieu =d +acc =d verbe
garçon :	n	ϵ_{infl} :	¡=verbe +p +nom +p t
pomme :	n	ϵ_{comp} :	=t +lieu +tmp c
ϵ_{temps} :	$tmp - tmp$	ϵ_{lieu} :	$lieu - lieu$

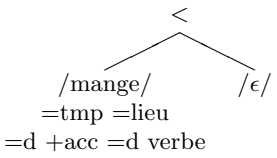
Dans ce lexique, les articles se composent d'un nom pour former le groupe nominal. Les traits associés sont ceux du cas et de la position du signe. On intègre deux marqueurs sans réalisation de surface pour le lieu et l'espace qui ne sont pas marqués dans la phrase.

Le verbe est dissocié en quatre entrées. La première est la lexicalisation du verbe. La seconde construit la structure argumentale du verbe contenant tous ses arguments. La troisièle représente l'inflexion qui permet de positionner les signes objet et sujet. Et la quatrième est celle de la phase de complemtizer plaÁant les signes de lieu et de temps.

Dérivation

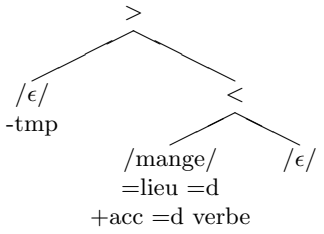
1. énumération = {le, la, garçon, pomme, manger, ϵ_{verbe}, ϵ_{infl}, ϵ_{comp}}
2. entrée lexicale "prendre" : v

   entrée lexicale "ϵ_{verbe}" : =¿v =tmp =lieu =d +acc =d verbe
   fusion : le verbe reçoit son comportement syntaxique. De plus, la forme phonologique du verbe passe sur la tête de la dérivation.

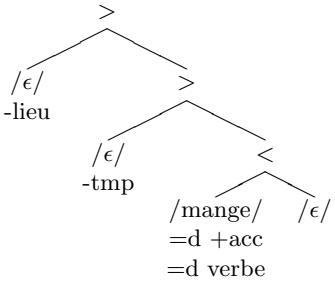


3. le verbe reçoit son argument de temps.

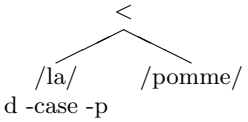
   Ici on utilise un argument "vide", c'est-à-dire qui ne spécifie pas le temps.



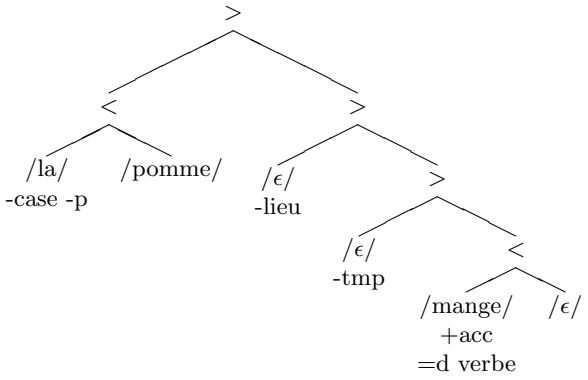
4. De manière analogue, le verbe reçoit son argument de lieu. Ici on utilise aussi un argument "vide", c'est-à-dire qui ne spécifie pas le lieu.



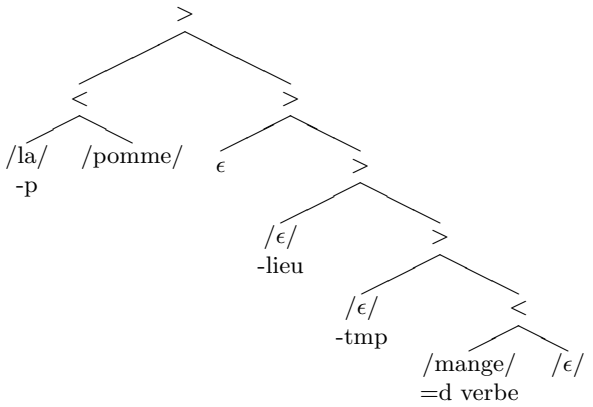
5. entrée lexicale "la" : =n d -case -p
   entrée lexicale "pomme" : n
   fusion entre les deux pour construire un groupe nominal :



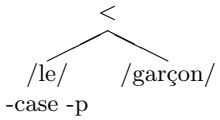
6. les deux dérivations sont alors fusionnées. Le verbe reçoit un premier argument qui sera l'objet de la phrase.



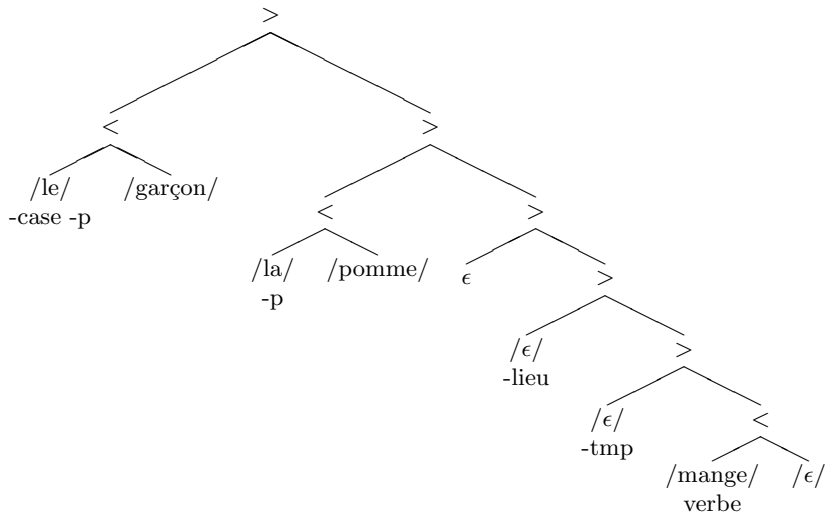
7. le premier trait sur la tête de la structure est : +case. On déclenche donc un déplacement. C'est la vérification du cas pour un groupe nominal, ici "accusatif" fourni par le verbe.



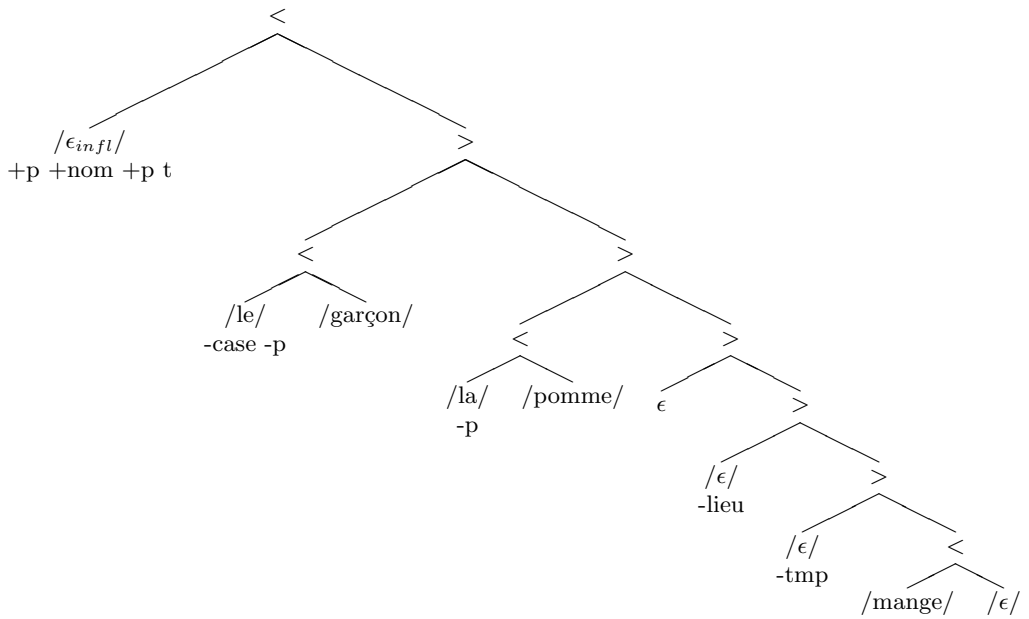
8. entrée lexicale "le" : =n d -case -p
   entrée lexicale "garçon" : n
   fusion pour construire un groupe nominal :



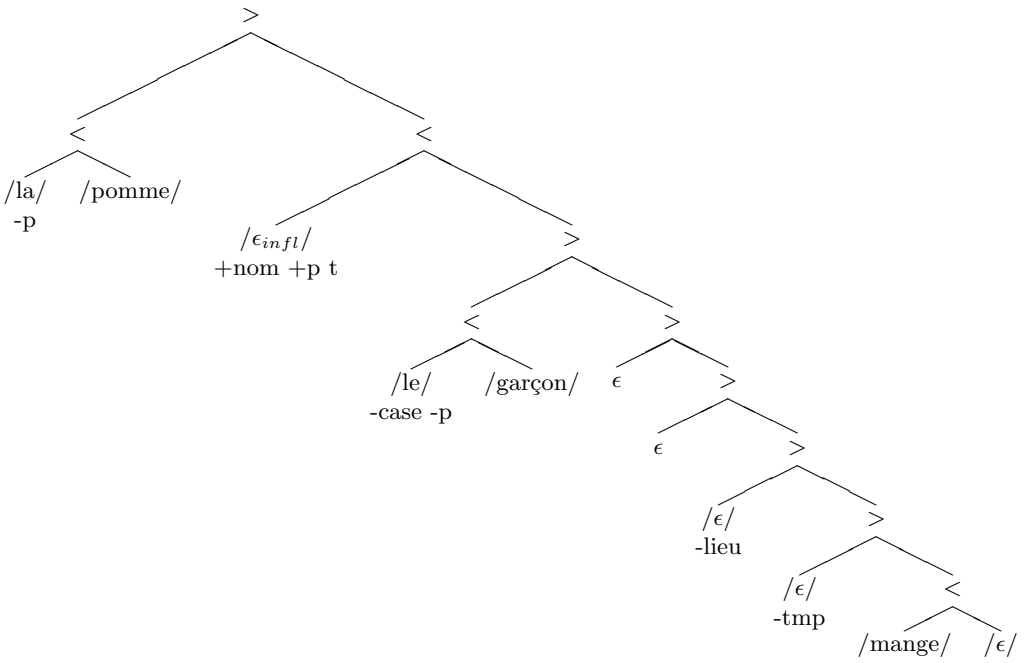
9. fusion entre le groupe nominal construit et la dérivation principale



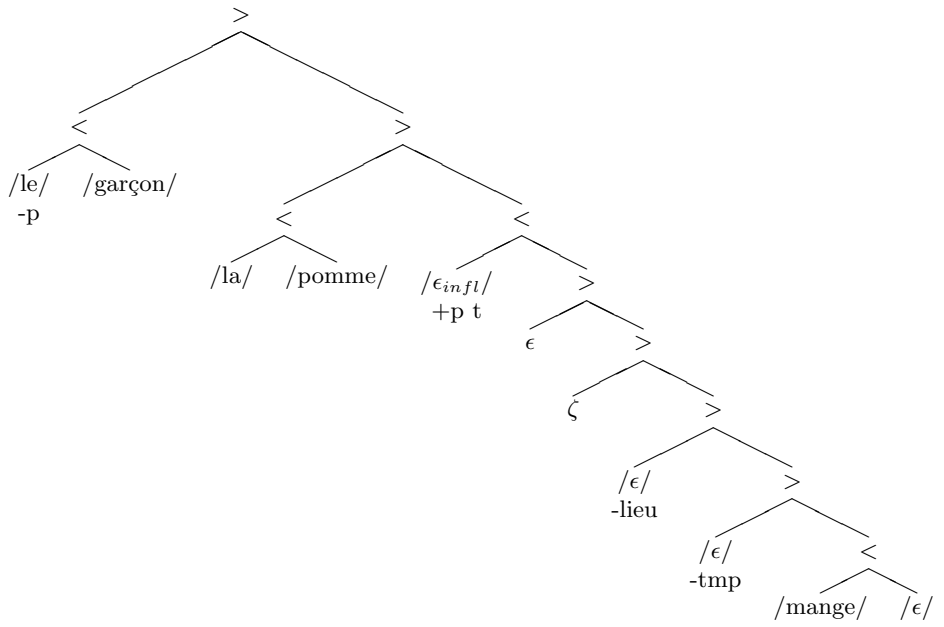
10. à ce stade de la dérivation, la première phase est terminée. La dérivation contient tous les arguments et certains ont commencé à être traités. À présent, nous entrons dans la phase suivante qui est ouverte par l'ajout de l'nflexion.
entrée lexicale "-ϵ" : <=verbe +p +nom +p t
On remarquera à nouveau que c'est l'inflexion qui introduit la donation de cas pour le sujet. Ainsi, nous respecterons l'hypothèse selon laquelle seuls les verbes ayant reçus leur inflexion peuvent être le centre d'une phrase.



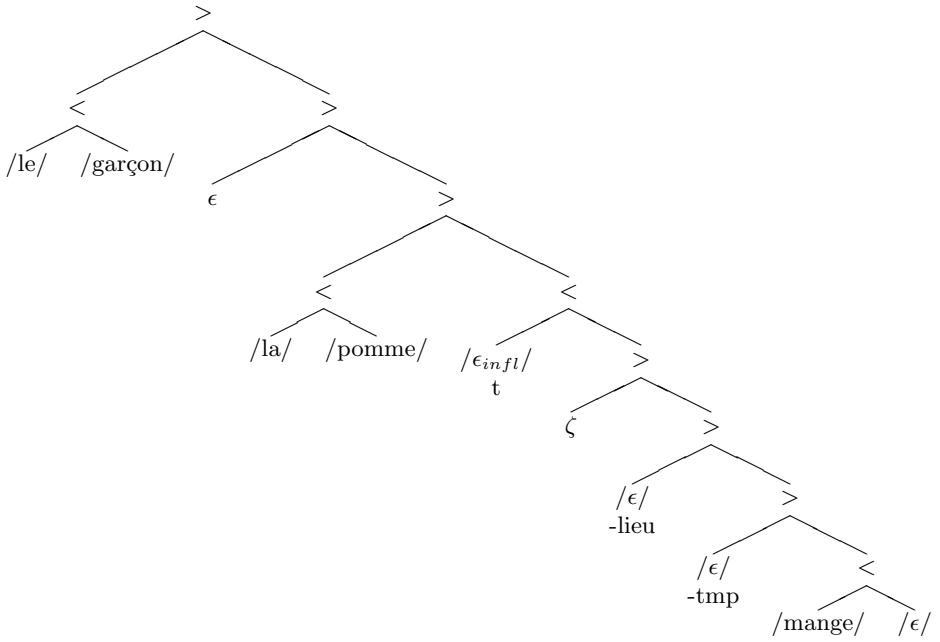
11. Grâce à l'entrée de l'inflexion, on réalise trois déplacements : deux qui ordonnent le sujet et l'objet, et un autre qui attribue le cas nominatif. L'ordre dans lequel ils sont réalisés modifie le résultat. Dans ce cas, on commence par traiter le groupe nominal objet.



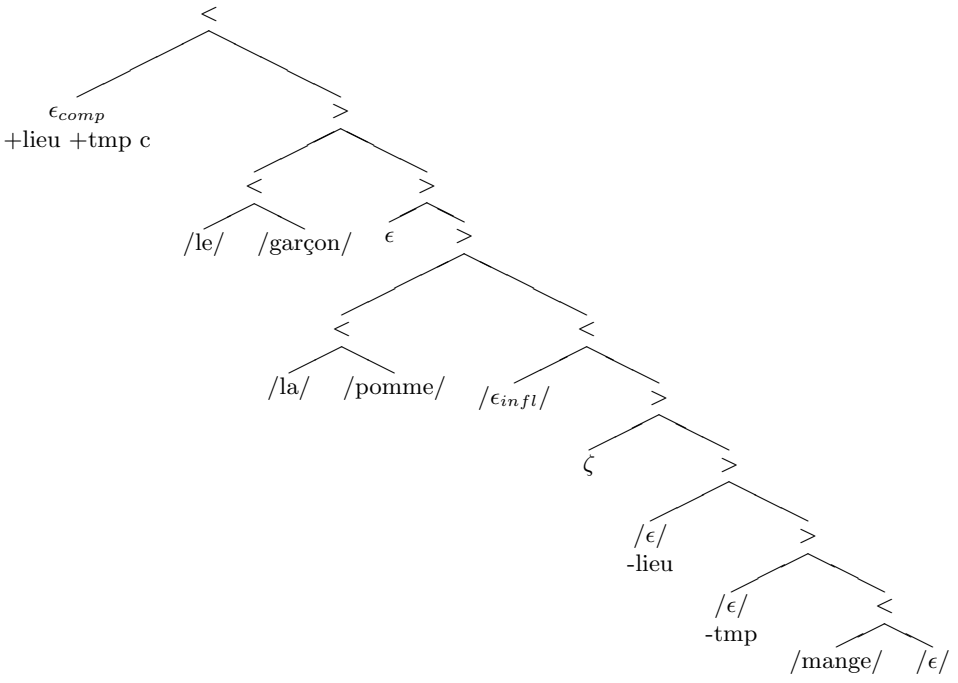
Afin de simplifier les notations, nous contractons les feuilles ϵ en position de spécifieur en une seule. Les opérations suivantes en feront apparaître une nouvelle qui sera intégrée dans la représentation suivante dans ce groupe de ϵ. Pour le différencier, nous le notons ζ puis on attribue le cas nominatif :



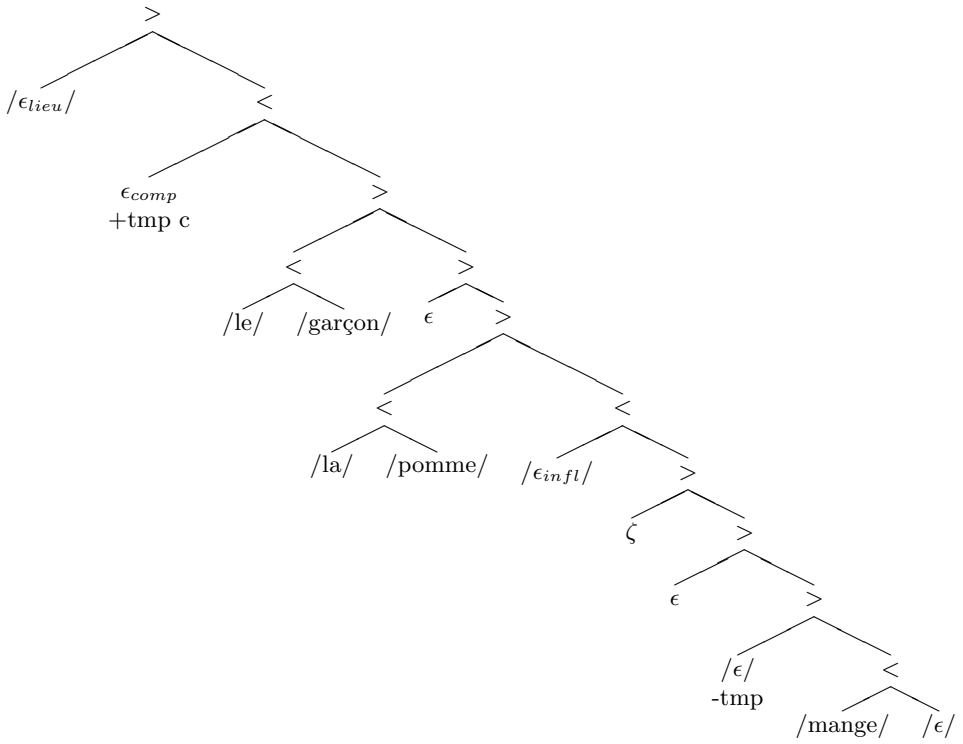
Et enfin, on déplace ce constituant dans sa position finale



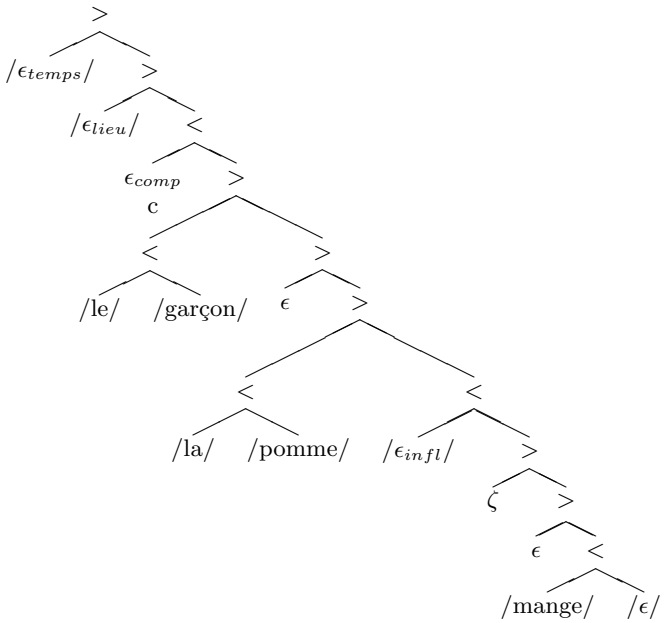
12. Ces trois déplacements terminent la phase ouverte par l'inflexion. La dérivation doit encore gérer le cas des marqueurs d'espace et de temps. C'est ce que permet de réaliser la dernière phase de complementizer. Fusion avec : entrée lexicale "ϵ_{comp}" : =t +lieu +tmp c



13. Cette entrée entraîne deux déplacements, l'un pour le marqueur de lieu puis un pour le marqueur d'espace. L'ordre de déplacement permet d'obtenir une séquence respectant la typologie des énoncés en langue des signes française .



puis



On note alors t_x la trace relative à l'élément x, c'est-à-dire que t_x est une ancienne position prise par le constituant x au cours de la dérivation. À la fin de la dérivation on obtient alors la séquence :

(87) $\underbrace{\epsilon_{temps}\epsilon_{lieu}\epsilon_{comp}}$ $\underbrace{\text{le garçon}}$ $\underbrace{\text{la pomme}}$ ϵ_{inflex} $\underbrace{\text{t}_{sujet}\ \text{t}_{objet}\ \text{t}_{lieu}\ \text{t}_{temps}\ \text{verbe}}$

Dans cette représentation les signèmes sont marqués par les crochet en-dessous. Ils correspondent au contenu de la projection maximale d'une feuille. Par exemple, la feuille qui porte le verbe est la tête (l'élément principal) du groupe contenant les traces de tous ses arguments. Ainsi, sa réalisation en signes est relative à son utilisation contextuelle. On obtient la séquence de signes :

(88) $\underbrace{\text{garçon}}$ $\underbrace{\text{pomme}}$ $\underbrace{\text{manger}_{proforme\ pomme}}$

Dans le cas où l'objet est prépondérant sur le sujet. Nous utilisons d'autres traits que le trait p, par exemple p_+. Ainsi l'entrée d'inflexion pour le cas d'un objet prépondérant sera :

$$\epsilon_{infl} :<=\text{verbe } +\text{nom } +\text{p } +\text{p}_+ \text{ t},$$

ce qui oblige à traiter complètement le sujet avant de finir le traitement de l'objet. Le fait d'utiliser des traits distincts nous permet de ne pas être bloqués par la SMC. On obtient alors la séquence :

(89) $\underbrace{\text{pomme}}$ $\underbrace{\text{garçon}}$ $\underbrace{\text{manger}_{proforme\ pomme}}$

Conclusion

Ce travail pose les premières bases nécessaires pour la modélisation d'énoncés en langue des signes française à partir des GMs. Il essaie de tenir compte de l'ordre de la phrase simple et des modifications engendrées par la présence ou l'absence de flexion verbale. Cette flexion verbale est plus particulièrement envisagée comme étant la manifestation du phénomène d'incorporation nominale et met en évidence la notion de signème qui nous permet de traiter de la structure de surface et de la structure profonde d'un énoncé quelconque tout en marquant la présence ou l'absence de flexion verbale. Cette étude n'en est qu'à ses débuts et de nombreuses questions se posent pour la suite de notre travail. En effet, il reste un doute quand à la nature de cette flexion. Nous avons choisi de la traiter comme une incorporation, néanmoins reste toujours la possibilité que nous soyons en face de pronoms clitiques. Et c'est cette possibilité que nous devons examiner prochainement avec des tests pour vérifier si nous sommes confrontés ou non à des pronoms clitiques. De plus, il serait intéressant et indispensable d'étendre notre étude à des types de phrases plus complexes.