N^o d'ordre : 3446

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Houda ANOUN**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Approche logique des grammaires pour les langues naturelles

Soutenue le : 24 Octobre

Après avis des rapporteurs :

Yves Bertot	Directeur de recherche
Philippe de Groote	Directeur de recherche

Devant la commission d'examen composée de :

Alain Lecomte . . .	Professeur	Directeur
Pierre Castéran . . .	Maître de conférence	Co-directeur
Bruno Courcelle . .	Professeur	Président
Yves Bertot	Directeur de recherche . .	Rapporteur
Philippe de Groote	Directeur de recherche . .	Rapporteur
Myriam Quatrini .	Maître de conférence	Examinatrice
Robert Cori	Professeur	Invité
Sylvain Salvati . . .	Chargé de recherche Inria	Invité

– 2007 –

A mes parents pour leur soutien sans faille
A mon cher pays le Maroc, que j'aime énormément

Remerciements

Ma fascination pour le monde de la recherche remonte à mes années de collège. Je n'oublierai jamais notre excellent professeur de mathématiques, Mr Mohamed Ouard, qui a suscité en moi le plaisir du raisonnement grâce aux différents '*casses-têtes*' qu'il nous proposait après les cours. Je me souviens encore de l'engouement que j'ai éprouvé, pendant une année entière (1993-1994), en tentant de résoudre le problème de trisection d'un angle, entrainant tous les membres de ma famille dans mes réflexions obsessionnelles quotidiennes. Ce n'est que des années plus tard que j'ai su qu'une telle construction était impossible à réaliser en utilisant la règle et le compas[1] ! Ce premier contact avec un problème difficile m'a donné une idée préliminaire sur le domaine de la recherche et cette idée s'est confirmée pendant mes 4 années de thèse. De manière imagée, un chercheur peut être vu comme un aventurier audacieux voyageant dans un immense désert dans l'espoir de découvrir une quelconque oasis. Un tel voyage est certes fort excitant, mais il est semé de mirages et de tempêtes. Tout au long de cette exploration, le voyageur sera confronté à des états d'âmes variables entremêlant des phases de doute et des moments d'euphorie. Pour atteindre l'objectif visé, une seule stratégie gagnante s'impose : persévérer avec sérénité et suivre les conseils des explorateurs les plus expérimentés. Mon cheminement durant cette thèse n'aurait jamais pu aboutir sans l'intervention de plusieurs personnes, auxquelles j'adresse ma gratitude la plus sincère.

Tout d'abord, j'exprime mes profonds remerciements à mes deux directeurs Alain Lecomte et Pierre Castéran qui m'ont constamment orientée et soutenue pendant ces années. C'est grâce à la manière pédagogique d'Alain Lecomte et à sa compétence pluridisciplinaire que je me suis intéressée à la modélisation de certains aspects linguistiques avancés, élargissant ainsi ma formation de départ. Ses conseils avisés et ses idées pertinentes m'ont été d'une aide considérable dans l'avancement de ma thèse. J'ai eu la chance de connaitre Pierre Castéran lors de mon stage de DEA et c'est à cette occasion qu'il m'a fait découvrir le domaine passionnant de la démonstration assistée par ordinateur. Je garderai toujours pour exemple sa curiosité, sa rigueur scientifique et son perfectionnisme. Je le remercie pour l'écoute et la disponibilité dont il a fait preuve : cela a été un grand plaisir de travailler avec lui.
Mes remerciements vont également à mes deux rapporteurs Yves Bertot et Philippe de Groote pour leurs commentaires judicieux qui m'ont permis d'améliorer la qualité de ce manuscrit. J'exprime ma gratitude à Bruno Courcelle, je suis très heureuse et honorée qu'il ait présidé mon jury. Je remercie aussi les examinateurs et invités, Myriam Quatrini, Robert Cori et Sylvain Salvati qui ont accepté de juger mon travail. Je tiens aussi à remercier les membres de l'équipe Signe pour leurs discussions fructueuses, leur bienveillance et grande générosité.

[1] La preuve formelle de cette impossibilité a été élaborée par le mathématicien Pierre-Laurent Wantzel en 1837.

Je n'oublie pas toutes les personnes qui m'ont prodigué des conseils éclairés et que j'ai eu l'opportunité de rencontrer lors de différents colloques et conférences. Je remercie de tout cœur Glyn Morrill pour ses encouragements et les longues discussions enrichissantes qu'on a eues à Tbilisi sur les différentes familles de la logique et en particulier la logique de la vie. J'adresse un grand merci à Michael Moortgat, Willemijn Vermaat et Reinard Muskens qui m'ont permis de découvrir de nouvelles voies de recherche et qui étaient toujours disponibles pour répondre à mes questions. Mes plus vifs remerciements vont aussi à Dina Elkassas, Youssef Tahir et Fredrick Hoyt qui m'ont transmis leur passion pour le traitement de la langue arabe. Je remercie également toutes les personnes qui prennaient le temps de répondre à mes questions par e-mails, notamment Laurence Rideau qui m'a beaucoup aidée dans la compréhension des différentes fonctionnalités de l'outil PCoq ainsi que tous les membres du Coq-Club.

Je suis très redevable à mes amis et collègues de bureau qui ont contribué à la mise en place d'une ambiance de travail conviviale et stimulante. Un grand merci à Abdelaziz Elhibaoui qui a relu une grande partie de ma thèse et qui se portait toujours volontaire pour tester mes nouvelles techniques en psychothérapie. Je remercie aussi Mohamed Diouf, Hejer Rejeb et Mohamed Bouklit pour leur bonne humeur, leurs discussions philosophiques éclairantes et leur soutien. Je n'oublie pas le personnel administratif du LaBRI que j'ai du solliciter à maintes reprises : merci à Philippe Biais, Cathy Robineau et Labrousse Maite pour leur aide.

Je tiens à remercier toutes mes amies qui n'ont pas cessé de m'épauler et de me motiver pendant ma thèse. Mes remerciements vont en premier lieu à ma meilleure amie Laila Elbahtouri qui est douée d'une sagesse et d'un pouvoir hypnotique extraordinaires. Elle a le don de trouver des phrases simples qui s'avèrent suffisantes pour me remonter instantanément le moral. Merci aussi à Fiona, Alain et Carole pour leur accueil chaleureux et tout le temps qu'ils ont consacré à la relecture et la correction de mes articles en anglais. Je remercie également Laila Elaimani, Saida Goudrar, Nabila Aboulfadl et Sanae Benneni qui m'ont inlassablement encouragée.

Je tiens aussi à remercier chaleureusement toute ma famille pour son soutien sans faille. En particulier, un grand merci à ma sœur Soumaya qui a eu le courage de relire une partie du chapitre 2 de ma thèse et qui a même réussi à détecter quelques fautes d'orthographe ! Merci également à ma tante Naima, ma grand mère, mes oncles et mes beaux-parents qui n'ont jamais cessé de croire en moi, même s'ils n'arrivent toujours pas à appréhender mon sujet de recherche.
Je suis très reconnaissante à l'égard de mes chers parents qui ont déployé tous leurs efforts pour attiser mon enthousiame et éteindre les flammes du doute qui freinaient, par moment, mon avancement. Si j'en suis la aujourd'hui c'est indubitablement grâce à eux !
"Last but not least", j'aimerai exprimer ma profonde gratitude à mon époux et âme sœur Réda qui a su être patient et présent tout au long de ces années.

Table des matières

Table des figures

Introduction

1.1 Contexte Général

1.1.1 Linguistique et informatique

La linguistique est la discipline qui a pour objet d'étude les langues naturelles. Son objectif est de décrire leur fonctionnement comme système stable, à un moment donné de l'histoire, sans pour autant, se préoccuper de leurs évolutions futures.
Pionnier de la linguistique moderne, De Saussure fut le premier à défendre la thèse *structuraliste* selon laquelle la langue possède une structure interne pouvant être caractérisée de manière rigoureuse [39]. Ce dernier a proposé d'appréhender la langue comme un système composé de signes interdépendants, où chaque signe est constitué de deux faces indissociables : le *signifiant* correspondant à sa forme phonétique et le *signifié* qui désigne son contenu sémantique. Ces signes entretiennent des relations entre eux, en outre, ils sont regroupés sous la forme d'une structure linguistique qui met en évidence leurs différentes dépendances syntaxiques.

L'école structuraliste de Saussure a été d'une grande influence sur les travaux de Chomsky, notamment sa théorie de la Grammaire Générative [29] qui rejoint les préoccupations énoncées par la Grammaire de Port Royal au 17^{eme} siècle. Cette théorie distingue entre les *principes* universels qui représentent les règles syntaxiques communes à toutes les langues naturelles et les *paramètres* muables qui encodent les variations locales de chaque langue. Selon Chomsky, la grammaire universelle serait innée et gravée dans la circuiterie neuronale du cerveau humain. Ceci explique la facilité avec laquelle les enfants apprennent leur langue maternelle ; en effet, ils ont uniquement besoin d'acquérir l'ensemble des caractéristiques intrinsèques à leur langue (e.g., ordre des mots dans un énoncé : **SVO** (*sujet-verbe-objet*) pour le français, **SOV** (*sujet-objet-verbe*) pour le japonnais, etc) et ce à partir des phrases correctes qu'ils entendent. La grammaire générative permet aussi de rendre compte de la capacité créative de la langue humaine : c'est grâce aux règles inconscientes qu'un individu est capable de comprendre et produire des phrases qu'il n'a jamais rencontrées auparavant. Ces règles peuvent être vues comme des *algorithmes* qui spécifient la façon dont on peut combiner un nombre fini de mots pour créer une infinité de structures bien-formées, ainsi que la manière d'interpréter les énoncés nouvellement obtenus.

Chomsky a élaboré plusieurs modèles transformationnels conformes à l'esprit de la Grammaire Universelle avant de déboucher sur un système compact, connu sous le nom de *Programme Minimaliste* [31]. Dans ce modèle, l'ensemble des universaux est réduit à deux primitives la *fusion* (i.e., *merge*) et le déplacement (i.e., *move*). La première opération sert à construire une structure complexe à partir de deux éléments plus simples (e.g., en fusionnant le déterminant '*la*' avec le

nom commun '*linguistique*', on obtient le syntagme nominal '*la linguistique*'). En revanche, la seconde fonction effectue une restructuration de l'énoncé (e.g., dans la phrase interrogative '*which languages does Houda speak _ ?*', le complément d'objet direct '*which languages*' est placé en tête, alors que sa position initiale est marquée par une trace phonétiquement vide _).

Suite au grand essor qu'a connu le domaine de l'informatique dans les années soixante, une nouvelle branche de la linguistique a vu le jour : il s'agit de la *linguistique computationnelle*. Cette discipline s'intéresse à la modélisation rigoureuse du fonctionnement de la langue en utilisant des outils empruntés à l'informatique fondamentale tels que la logique, l'algorithmique ou encore la théorie des langages et la théorie de la complexité. Les objectifs de la linguistique computationnelle sont très différents de ceux du traitement automatique des langues (**TAL** en abrégé). En effet, le **TAL** s'inscrit dans le cadre de l'intelligence artificielle et vise, en particulier, la réalisation d'applications bien ciblées (e.g., correcteurs orthographiques ou grammaticaux, traduction ou résumé automatiques ...) en se basant sur des techniques ad-hoc ou des méthodes statistiques. Toutefois, cette approche empirique se heurte à plusieurs problèmes dès qu'il s'agit de traiter des énoncés plus ou moins complexes qui nécessitent une analyse syntaxique approfondie. La linguistique computationnelle, quant à elle, a pour mission de proposer des modèles formels qui sont soumis aux exigences linguistiques et qui possèdent, en outre, des propriétés pertinentes d'un point de vue calculatoire (e.g., algorithmes d'analyse efficaces, interface syntaxe/sémantique). En héritant du rôle explicatif de la linguistique, ces modèles doivent contribuer à l'approfondissement de nos connaissances de la langue : il ne suffit pas qu'ils exhibent un algorithme d'analyse qui détecte les phrases syntaxiquement correctes, il faut aussi qu'ils soient capables de justifier la raison sous-jacente à l'agrammaticalité de certains énoncés (e.g., expliquer l'impossibilité de coréférence entre le nom propre *Houda* et le pronom personnel *l'* dans une phrase telle que $^{*}Houda_i\ l'_i\ aime$[1]).
Plusieurs formalismes grammaticaux ont été conçus dans le cadre de cette approche. Citons entre autres, les grammaires minimalistes de Stabler [101] qui formalisent le Programme Minimaliste de Chomsky, les grammaires d'arbres adjoints [62] ainsi que les grammaires catégorielles de type logique [67, 77, 37, 87] qui sont au cœur de cette thèse.

1.1.2 Logiques au service de la linguistique computationnelle

Les systèmes déductifs ont été introduits au début du $20^{ème}$ siècle pour formaliser entièrement différents domaines et éviter les paradoxes émanant des définitions intuitives et informelles (e.g., ensemble des ensembles et paradoxe de Russell). Parmi les formalismes utilisés dans l'étude des langues naturelles, nous nous intéressons aux grammaires catégorielles de type logique puisqu'elles garantissent une interface syntaxe/sémantique aisée grâce à la correspondance de Curry-Howard (correspondance entre les preuves constructives et les λ-termes simplement typés) [20]. Ces formalismes sont en phase avec la philosophie de la Grammaire Universelle de Chomsky et proposent d'encoder l'ensemble des principes par des règles d'inférence logiques qui décrivent les mécanismes calculatoires de la syntaxe. L'ensemble des paramètres est, quant à lui, représenté par un *lexique* qui associe à chaque signe un nombre fini de types adéquats qui rendent compte de ses différents comportements syntaxiques.
Il est indubitable que le nombre des mots d'une phrase joue un rôle crucial dans le jugement de sa grammaticalité (e.g., '*Il lit*' est correct mais '**Il il lit*' ne l'est pas), on ne peut donc faire appel à la logique classique ou la logique intuitionniste pour gérer l'agencement des ressources

[1] Par convention, le symbole * est utilisé par les linguistes pour annoter les phrases incorrectes.

linguistiques. En effet, ces deux logiques autorisent l'application des règles d'*affaiblissement* et de *contraction* qui peuvent à tout moment dupliquer ou réduire les hypothèses mises en jeu. Nous avons plutôt besoin d'une logique sensible à la quantité de ressources disponibles, qui considère chaque hypothèse du contexte comme une entité consommable : ces contraintes sont respectées par la logique linéaire définie par Jean-Yves Girard [48].

Dans la littérature des grammaires catégorielles, on distingue entre deux familles de systèmes dédiés à l'analyse des langues naturelles à savoir les systèmes bidirectionnels qui sont fondés sur des variantes non-commutatives de la logique linéaire et les systèmes non-directionnels qui utilisent des variantes commutatives de cette logique. La première famille a été initiée par les travaux de Joachim Lambek notamment le système logique qui porte son nom [67]. Dans cette logique non-commutative, l'ordre des mots est pris en charge par la syntaxe. La flèche linéaire (i.e., l'implication linéaire) est ainsi scindée en deux instances (/, \) utilisées en fonction de l'ordre respectif entre le foncteur et son argument (e.g., un adjectif prénominal porte le type n/n puisqu'il précéde son argument, tandis qu'un adjectif postnominal se voit assigné le type $n\backslash n$ étant donné qu'il suit son argument : '*nouveau document*' vs '*document récent*'). Le système de Lambek étant complet (i.e., il comprend les règles d'élimination et d'introduction des connecteurs utilisés), il présente alors un atout principal par rapport à ses prédécesseurs (e.g., Grammaires *AB* [17]), à savoir la possibilité d'appliquer la technique de *raisonnement hypothétique*. Grâce à cette technique, on peut introduire des ressources hypothétiques durant la dérivation, puis les abstraire ultérieurement si leurs positions sont périphériques. Ceci permet de rendre compte du phénomène d'extraction illustré, par exemple, dans la phrase relative '*la logique que Lambek a définie _*', où le pronom '*que*' fusionne avec une clause incomplète '*Lambek a définie*' à laquelle manque un objet direct.

Certes, la logique de Lambek bénéficie d'une élégance mathématique remarquable, toutefois, elle souffre de plusieurs limitations d'un point de vue linguistique. Plusieurs enrichissements ont été proposés pour combler les lacunes de ce système et l'adapter à la grande complexité des phénomènes linguistiques. Ceci a donné naissance aux grammaires catégorielles multimodales [77] qui font recours au raisonnement structurel pour permettre une reconfiguration contrôlée de ressources. La rigidité du système de Lambek est ainsi contrée en tolérant l'accès à des variantes restreintes de règles structurelles (e.g., associativité, commutativité) qui sont appliquées localement en présence de modes de composition spécifiques. Grâce à ces extensions, les grammaires multimodales sont capables de décrire certains phénomènes linguistiques relativement ardus comme c'est le cas pour les dépendances croisées du néerlandais [80], toutefois, leur pertinence est légèrement fragilisée par leur grande complexité et leur difficulté d'utilisation.

Curry fut le premier à proposer de rendre compte de la syntaxe des langues d'une manière plus souple, en utilisant un système non-directionnel [35]. D'après Curry, il est nécessaire de différencier entre deux niveaux de la grammaire, qualifiés de *tectogrammatique* (syntaxe abstraite) et *phénogrammatique* (forme phonétique). Le premier niveau gère les dépendances syntaxiques entre les signes en faisant appel à une logique commutative, alors que le fardeau de l'ordre des mots est entièrement délégué au second niveau. Cette orientation a été suivie par Oehrle [88, 89] et plus récemment par de Groote [37] et Muskens [87], ce qui a conduit à la définition d'un certain nombre de formalismes dont *mon :LP*, les *Grammaires Catégorielles Abstraites* et les *Lambda-Grammaires*. Dans ces derniers modèles, le niveau concret est généralisé à plusieurs dimensions englobant, par exemple, le calcul de la forme phonétique ainsi que celui de la représentation sémantique. Ces systèmes non-directionnels ont le mérite d'être plus intuitifs et plus compacts que les systèmes bidirectionnels, puisqu'ils ne manipulent qu'un seul connecteur logique à savoir la flèche linéaire ($\multimap$). En outre, ils sont plus modulaires étant donné qu'ils permettent de dégager

les spécifications universelles des variations locales à chaque langue. A titre d'exemple, dans ces modèles, les adjectifs de toutes les langues naturelles se voient attribués la même spécification abstraite coïncidant avec le type $n \multimap n$ (i.e., un adjectif est une fonction qui cherche à se combiner avec un nom commun pour former un nouveau nom). L'ordre respectif entre l'adjectif et son argument est, quant à lui, encodé par la dimension phonétique (e.g., l'adjectif prénominal anglais '*new*' porte la forme phonétique suivante $\lambda x.\ new \bullet x$, tandis que l'adjectif postnominal français '*récent*' admet comme forme phonétique le terme $\lambda x.\ x \bullet récent$[2]).
Le contraste entre l'approche bidirectionnelle et l'approche non-directionnelle est illustré par les deux dérivations ci-dessous repésentant, chacune, une analyse du syntagme nominal '*un document récent*'.

$$\dfrac{\dfrac{}{un : np/n \vdash np/n}\ Ax \quad \dfrac{\dfrac{}{document : n \vdash n}\ Ax \quad \dfrac{}{récent : n\backslash n \vdash n\backslash n}\ Ax}{document : n,\ récent : n\backslash n \vdash n}\ \backslash E}{un : np/n,\ document : n,\ récent : n\backslash n \vdash np}\ /E$$

$$\dfrac{\dfrac{}{\vdash \lambda x.\ un \bullet x : n \multimap np}\ Ax \quad \dfrac{\dfrac{}{\vdash \lambda x.\ x \bullet récent : n \multimap n}\ Ax \quad \dfrac{}{\vdash document : n}\ Ax}{\vdash document \bullet récent : n}\ \multimap E}{\vdash un \bullet document \bullet récent : np}\ \multimap E$$

1.2 Contributions de la thèse

Les contributions majeures de cette thèse s'articulent autour des trois axes de base de la linguistique computationnelle, à savoir la logique, la linguistique et l'informatique. Nous présentons ci-après une description succincte de ces travaux de recherche.

1.2.1 Grammaires Linéaires Etiquetées

Ce travail de recherche a été mené en collaboration avec Alain Lecomte pendant les deux dernières années de ma thèse (entre 2005 et 2007). Son objectif est de créer un pont entre les grammaires de type logique et le Programme Minimaliste [31] en proposant un nouveau formalisme $\mathcal{GLE}$ (i.e., Grammaires Linéaires Etiquetées) [14, 3] qui simule les deux opérations transformationnelles de fusion et de déplacement dans un cadre déductif. Une telle interopérabilité est fructueuse dans la mesure où elle permet de mieux appréhender les différents mécanismes mis en œuvre dans les dérivations des grammaires minimalistes. $\mathcal{GLE}$ est basé sur un fragment compact de la logique linéaire intuitionniste implicative et exponentielle [48, 105], comprenant un ensemble restreint de règles de déduction linguistiquement pertinentes. Ce système non-directionnel admet deux interfaces (syntaxe-phonétique, syntaxe-sémantique) obtenues grâce à la correspondance de *Curry-Howard*. Le calcul de la forme phonétique ainsi que celui de la représentation sémantique résultent de la combinaison des λ-termes dans les deux dimensions du niveau concret. Ce processus se déroule en parallèle avec la dérivation syntaxique, puisque chaque étape de déduction

[2] L'opérateur $\bullet$ est utilisé pour concaténer les constantes phonétiques.

encapsule un pas calculatoire dans le λ-calcul simplement typé.
L'originalité de $\mathcal{GLE}$ provient de son traitement raffiné du raisonnement hypthétique, qui garantit la normalité de toutes les dérivations (i.e., les dérivations ne contiennent aucun détour redondant). Notre modèle vise la conservation des avantages de cette technique (e.g., prise en charge des dépendances non-bornées) tout en contraignant son utilisation dans le but d'augmenter l'efficacité du système. Ainsi, au lieu de considérer une règle d'axiome logique librement accessible, notre système est doté de séquences finies d'hypothèses contrôlées consommables qui sont attachées à certaines entrées lexicales susceptibles de subir un déplacement (e.g., le pronom interrogatif '*which*'). Ces hypothèses sont utilisées pour occuper les positions intermédiaires de leurs entrées respectives. Elles doivent être introduites au cours de la dérivation, dans un ordre précis, puis déchargées au même temps par leur entrée propre qui sera ainsi capable de regagner son site final. L'abstraction simultanée de plusieurs hypothèses contrôlées est possible grâce à l'utilisation de l'exponentielle (« ! ») qui permet d'apporter plus de flexibilité à la gestion de la multiplicité de ressources, en autorisant leur contraction de manière contrainte (i.e., seule la contraction des hypothèses liées à la même entrée lexicale est acceptable).
La distinction entre les deux variantes *visible* et *furtive* de l'opération de déplacement est prise en charge par la forme phonétique des entrées lexicales liées. Dans le cas d'un mouvement visible, les sites transitoires occupés par les variables contrôlées non-prononcées seront remplacées par des traces phonétiquement vides.
Notre contribution montre ainsi que le déplacement est une notion métaphorique qui peut être remplacée par un processus logique plus rigoureux. En effet, cette opération transformationnelle peut être formellement encodée par une règle hybride faisant intervenir une succession d'étapes de déduction dont le raisonnement hypothétique contrôlé, la contraction (dans le cas de déplacements simultanés) et le modus-ponens (simulation logique de la fusion).

1.2.2 Applications linguistiques dans $\mathcal{GLE}$

La pertinence linguistique du formalisme $\mathcal{GLE}$ est mise en évidence en montrant sa capacité de rendre compte de phénomènes non-linéaires avancés qui sont difficilement analysables dans les systèmes bidirectionnels. Nous nous sommes intéressés, en premier, à l'étude des *anaphores* (e.g., réflexifs tels que '*himself*') et des *pronoms personnels* (e.g. '*he*', '*him*') dont la sémantique est étroitement liée à celle d'un autre élément de l'énoncé.
L'approche prédominante traitant le phénomène de liage est celle formulée par Chomsky dans les années 80 [30]. La théorie de liage ainsi proposée comprend un ensemble de principes qui explicitent certaines propriétés structurelles devant être respectées par les antécédents des anaphores et des pronoms personnels. Alors que les anaphores admettent toujours un antécédent faisant partie de leur clause minimale (Principe A), les antécédents potentiels des pronoms personnels se trouvent à l'extérieur de leur domaine local (Principe B). Cette complémentarité entre le comportement des anaphores et celui des pronoms personnels est illustrée par les énoncés ci-dessous :

(1) a. John$_i$ likes himself$_i$/*him$_i$.

b. John$_i$ thinks Jim$_j$ likes himself$_j$/him$_i$/*himself$_i$/*him$_j$.

c. John$_i$ thinks he$_i$/he$_j$ is smart.

Les travaux récents de Kayne [64] ont tenté d'interpréter ces principes en termes de mouvement et ce dans le cadre du Programme Minimaliste. A titre d'exemple, pour établir la coréférence entre le pronom '*he*' et le nom propre '*John*' dans la phrase (1c), Kayne propose que '*John*' et '*he*' soient initialement regroupés au sein d'un même "constituant double" [John, he] qui sera ultérieurement séparé suite au déplacement frontal de son premier composant.

Notre formalisation logique de la théorie de liage est fortement inspirée des idées de Kayne. Les doublets considérés sont encodés au moyen d'entrées lexicales liées, de telle façon que la tête de l'entrée (i.e., l'antécédent) n'est insérée qu'après la contraction et le déchargement simultané de ses hypothèses contrôlées (dont le pronom qui représente une variable prononcée). Suite à ce déchargement, le pronom personnel mis en jeu se trouve lié à son antécédent. Grâce au raisonnement hypothétique contrôlé, notre approche permet de modéliser les contraintes de localité inhérentes à la théorie de liage, et ce de manière simple et élégante.
D'un autre côté, nous proposons de généraliser l'utilisation des constituants doubles afin de prendre en charge le phénomène d'ellipse qui consiste à omettre un ou plusieurs mots d'un énoncé (e.g., verbe, groupe verbal) dans un but de concision (cf. énoncés 2).

(2) a. John likes golf and Mary does too.
b. John read every book that Mary did.

L'utilisation d'une entrée liée pour encoder des doublets tels que [likes golf, does too] et [read, did] permettra d'assurer le partage de la propriété sémantique associée à l'élément élidé (e.g., λx. **Like**(x, **Golf**), λx. λy. **Read**(y, x)), entre la clause source qui est complète et la clause cible qui ne l'est pas.
Nous nous sommes penchés, en dernier, sur l'étude du phénomène d'ilôts qui caractérise certains domaines syntaxiques interdisant l'extraction de leurs sous-constituants (cf. énoncés (3)).

(3) a. *The book that [Mary sleeps and Eric reads _] is interesting.
b. *The book that Mary hates the girl [who read _] is interesting.

Afin de rendre compte de ce phénomène, nous suggérons d'étendre le système de base par une simple règle de sous-typage qui permet de vérifier la bonne formation des ilôts et contraindre par conséquent l'opération d'extraction.

1.2.3 Atelier logique *ICHARATE*

Depuis les années soixante, on fait de plus en plus appel aux assistants de preuves pour formaliser différentes théories et vérifier les propriétés de modèles complexes, notamment dans les domaines du transport, de la cryptologie, des télécommunications etc. Cette activité est fructueuse puisqu'elle garantit la vérification mécanisée de preuves permettant ainsi aux utilisateurs de s'affranchir du moindre risque d'erreur. La modélisation de la syntaxe et de la sémantique des langues naturelles est fondée sur des formalismes de haut niveau caractérisés par leur richesse et leur grand degré de paramétrisation. La manipulation de tels systèmes s'avère souvent ardue, en particulier pour les novices. L'objectif de cette contribution est de mettre à la disposition des chercheurs et étudiants un outil convivial qui leur permet d'étudier et d'appréhender certaines grammaires logiques complexes dédiées au traitement de la langue. Cet outil : l'atelier *ICHARATE*

[10, 8, 7, 6, 13] est composé de bibliothèques pour l'assistant à la démonstration **Coq** [72, 21] qui est basé sur la logique d'ordre supérieur et l'isomorphisme de Curry-Howard.
La bibliothèque principale d'*ICHARATE* est consacrée à la formalisation complète de différentes présentations de la logique multimodale. Ce travail de recherche a été réalisé en collaboration avec Pierre Castéran pendant les deux premières années de ma thèse (entre 2003 et 2005). Le produit final comprend un catalogue de méta-théorèmes, de propriétés génériques et de règles dérivées qui s'appliquent pour des classes entières de grammaires et qui modélisent le comportement de phénomènes linguistiques divers. Ces règles réutilisables offrent aux utilisateurs une grande économie d'effort en leur épargnant la peine de recommencer sans cesse le même raisonnement. Leurs preuves, effectuées une fois pour toute, mettent en œuvre des techniques variées telles que la généralisation, la récurrence sur les dérivations ou encore la réflexion. L'interaction avec l'atelier est facilitée grâce à une interface utilisateur comprenant un ensemble d'outils spécialisés (banque de tactiques, technique de proof-by-pointing ...).
ICHARATE a été récemment étendu pour englober une formalisation du nouveau système non-directionnel *GLE*. L'interaction avec **Coq** nous a été utile pour tester et corriger les versions successives de ce formalisme et pour assimiler pleinement son fonctionnement.

1.2.4 Plan de la thèse

Ce manuscrit est structuré en deux parties : la première est dédiée aux grammaires bidirectionnelles et précisément les grammaires catégorielles multimodales, tandis que la seconde s'intéresse aux grammaires non-directionnelles, et en particulier, aux grammaires logiques étiquetées.
Le chapitre 2 introduit une description synthétique des composantes basiques et caractéristiques fondamentales des grammaires catégorielles bidirectionnelles. Cette présentation porte spécialement sur les grammaires catégorielles multimodales qui constituent un enrichissement du calcul de Lambek et qui proposent d'atténuer ses défaillances en faisant appel au raisonnement structurel contrôlé.
Le chapitre 3 détaille ensuite la partie de l'atelier logique *ICHARATE* qui concerne la formalisation de ces grammaires en **Coq**. Ce chapitre décrit les éléments principaux inhérents à la formalisation de la syntaxe et la sémantique de la logique multimodale. Il présente, en outre, quelques exemples de règles génériques tout en soulignant les techniques déployées pour les prouver. Ce chapitre s'achève par la description des différents outils de l'interface utilisateur, réalisés pour faciliter l'interaction avec l'atelier.
Le chapitre 4 est consacré à la définition formelle des grammaires logiques étiquetées qui sont basées sur un fragment réduit de la logique linéaire intuitionniste implicative et exponentielle. L'accent est notamment mis sur l'aptitude de ce système non-directionnel à simuler les primitives du Programme Minimaliste dans un cadre logique.
Le dernier chapitre illustre la pertinence linguistique des grammaires logiques étiquetées en montrant leur capacité à décrire des phénomènes avancés de manière élégante. L'emphase est notamment portée sur certains phénomènes linguistiques non-linéaires tels que le traitement d'anaphore et la résolution d'ellipse, ainsi que sur la restriction de l'extraction et le phénomène d'ilôts.

Première partie

Grammaires Catégorielles Multimodales

Grammaires Catégorielles Multimodales

Les grammaires catégorielles de type logique se distinguent des grammaires à structure de phrase (i.e., *PSG* : Phrase Structure Grammars) de la hiérarchie de Chomsky par leurs règles de production universelles qui sont indépendantes des terminaux (mots de la langue choisie). Ces règles représentent une panoplie de principes partagés par toutes les langues naturelles. Elles sont encodées sous la forme d'un système déductif qui permet de gérer la combinaison des différentes *catégories* (i.e., *types*) encapsulant le comportement syntaxique des mots. Les paramètres intrinsèques à chaque langue sont réduits à une fonction d'assignation (un *lexique*) qui associe à chaque mot de la langue choisie un ensemble fini de types syntaxiques et de représentations sémantiques. La notion d'analyse syntaxique est ainsi interprétée en terme de démonstration dans la logique considérée. En effet, la preuve qu'une expression m_1,..., m_n vérifie les propriétés grammaticales de la catégorie A coïncide avec la dérivabilité d'un séquent $(m_1 : A_1),...,(m_n : A_n) \vdash A$ dans le système déductif sous-jacent[1]. L'atout majeur des grammaires catégorielles logiques réside dans la connexion aisée entre la syntaxe et la sémantique émanant de la correspondance de Curry-Howard qui établit un lien entre les *preuves* constructives et les λ-*termes* typés [52, 58]. Ainsi, la structure prédicative représentant la sémantique d'une expression bien formée n'est autre que le λ-terme encodant sa dérivation syntaxique, obtenu à partir de la facette calculatoire de la correspondance de Curry-Howard.

Le présent chapitre comprend une étude bibliographique synthétique portant sur les grammaires catégorielles *bidirectionnelles* qui utilisent la dimension syntaxique pour prendre en charge l'ordre des mots dans une phrase. Nous présenterons, dans un premier temps, l'évolution historique de ces grammaires : en commençant par les grammaires catégorielles classiques de Bar-Hillel [17], en passant par le calcul de Lambek [67, 68] et en débouchant, au final, sur la logique multimodale enrichie de Moortgat [77]. Les composantes de ces grammaires seront formellement décrites et illustrées au fur et à mesure par des exemples concrets. Nous nous focaliserons en particulier sur les caractéristiques fondamentales des grammaires catégorielles multimodales qui sont au cœur de cette partie. L'accent sera notamment mis sur l'intérêt linguistique de ces grammaires qui, à la différence de leurs prédécesseurs, s'avèrent capables d'analyser rigoureusement un nombre important de phénomènes avancés de la langue naturelle.

[1] Pour chaque i allant de 1 à n, A_i est une catégorie syntaxique du mot m_i.

2.1 Un peu d'histoire

2.1.1 Premières Grammaires Catégorielles

Le calcul de fractions introduit par Ajdukiewic en 1935 [1] a constitué une source d'inspiration remarquable qui a conduit à la définition des premières grammaires catégorielles. En effet, ce calcul a été ultérieurement enrichi par Bar-Hillel [17] qui a proposé une extension bidirectionnelle du formalisme initial, donnant ainsi naissance aux grammaires *AB*.
Les grammaires *AB* sont totalement lexicalisées : leurs règles de production sont universelles et ne dépendent pas des mots de la langue choisie. Seul le lexique varie d'une grammaire à une autre ; ce dernier n'est autre qu'une fonction d'assignation qui associe à chaque mot un ensemble fini de types décrivant ses différents comportements syntaxiques. Les types syntaxiques considérés par les grammaires *AB* sont classés en deux catégories, les types atomiques et les types composés. Les premiers sont des catégories basiques qui désignent des expressions complètes, tels *np* (i.e., *noun phrase*) qui permet de représenter les syntagmes nominaux (e.g., '*Houda*', '*le sage*'), *n* (i.e., *noun*) qui représente les noms communs, (e.g., '*humilité*') et *s* (i.e., *sentence*) qui n'est autre que la catégorie syntaxique associée aux phrases syntaxiquement correctes.
D'autre part, les types composés sont formés à partir des catégories atomiques en utilisant deux opérateurs bidirectionnels (/, \) qui prennent en compte l'ordre de combinaison des mots. Intuitivement, le foncteur A/B (resp. $B\backslash A$) représente une entité qui attend une expression de type B à sa droite (resp. à sa gauche) pour former une expression de type A. Ainsi, on associe aux déterminants (e.g., '*le*') le type syntaxique np/n puisqu'ils ont besoin de fusionner avec un nom commun (e.g., '*sage*') à leurs droites pour construire un groupe nominal (e.g., '*le sage*'). En outre, on attribue aux verbes transitifs (e.g., '*aime*') la catégorie syntaxique $(np\backslash s)/np$ car ces derniers se combinent en premier avec un syntagme nominal à leurs droites (l'objet) pour former un groupe verbal (e.g., '*aime l'humilité*'), puis fusionnent avec un autre groupe nominal à leurs gauches (le sujet) pour former une phrase (e.g., '*le sage aime l'humilité*'). Il est intéressant de signaler qu'afin de typer les verbes transitifs, nous avons opté pour le choix de la catégorie syntaxique $(np\backslash s)/np$ à la place de $np\backslash(s/np)$ et ce pour rester en phase avec les travaux linguistiques considérant que le verbe forme avec ses compléments un constituant prédicatif (par contre, la combinaison du verbe avec le sujet, (e.g., '*le sage aime*') n'est pas considéré par les linguistes comme un constituant) [92].
L'agencement des catégories syntaxiques est géré par un ensemble de règles de réduction constituant le noyau du calcul *AB*. Ces règles sont présentées dans la figure 1 :

$$\frac{A/B \quad B}{A}\ /_e \qquad \frac{B \quad B\backslash A}{A}\ \backslash_e$$

FIG. 1: Règles de réduction du calcul *AB*

Les règles ci-dessus ne font que décrire formellement le comportement des deux connecteurs bidirectionnels / et \. En effet, la règle $/_e$ (resp. règle $\backslash_e$) stipule que la concaténation du foncteur A/B (resp. $B\backslash A$) à gauche (resp. droite) de son argument B engendre le type A.

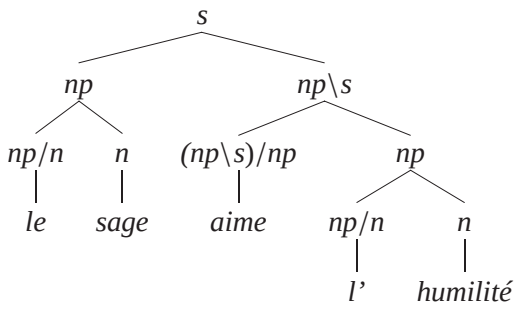
Exemple 2.1.1 On considère la grammaire *AB* définie par le lexique suivant :

Mots	*Types syntaxiques*
le, l'	np/n
sage, humilité	n
aime	$(np\backslash s)/np$

En utilisant les règles du calcul *AB*, on est capable d'assigner à la phrase '*Le sage aime l'humilité*' le type s comme le montre l'analyse suivante :

$$\dfrac{\dfrac{\dfrac{le}{np/n}\,Lex \quad \dfrac{sage}{n}\,Lex}{np}\,/_e \quad \dfrac{\dfrac{aime}{(np\backslash s)/np}\,Lex \quad \dfrac{\dfrac{l'}{np/n}\,Lex \quad \dfrac{humilité}{n}\,Lex}{np}\,/_e}{np\backslash s}\,/_e}{s}\,\backslash_e$$

Cette analyse peut être également représentée de manière descendante sous la forme d'un arbre ressemblant en quelques sortes aux arbres de dérivations syntaxiques produits par les grammaires hors contextes :



Certes, les grammaires *AB* bénéficient d'une grande simplicité et facilité d'utilisation. Cependant, elles souffrent de certaines limites linguistiques dues à leur gestion figée des constituants d'une phrase. En effet, dans ce formalisme, les dérivations sont fortement influencées par le lexique considéré puisque les règles du calcul *AB* ont pour unique but de combiner les parties pour former un composant plus riche et ne permettent, en aucun cas, de rendre compte de l'extraction de constituants. A titre d'exemple, ce dernier phénomène a lieu dans les subordonnées relatives comme '*que le sage aime* _' où le pronom relatif '*que*' est combiné avec une phrase à laquelle manque un objet (l'élément extrait) dont la position est marquée par le symbole _. Ainsi, en assignant au pronom relatif '*que*' le type syntaxique $(n\backslash n)/(s/np)$[2], le calcul *AB* est incapable de dériver la subordonnée précédente puisqu'on ne peut attribuer à l'expression '*le sage aime*' le type s/np en utilisant uniquement la catégorie syntaxique $(np\backslash s)/np$ pour les verbes transitifs. D'une façon générale, il n'est pas difficile de vérifier que le type engendré suite à l'analyse d'une expression dans le calcul *AB* est nécessairement un sous-type d'une des catégories syntaxiques associées aux mots qui la composent.

Nous verrons dans ce qui suit comment la logique de Lambek comble les lacunes du calcul *AB* en assurant plus de souplesse dans le maniement des ressources garantissant par conséquent la prise en charge de certaines formes du phénomène d'extraction notamment l'extraction périphérique (i.e., où l'élément extrait se trouve à l'extrême droite ou l'extrême gauche de l'expression à analyser).

[2]Le pronom relatif '*que*' a besoin de se combiner à sa droite avec une phrase à laquelle manque un objet pour former un modifieur de nom jouant le rôle syntaxique d'un adjectif.

2.1.2 Logique de Lambek

La première formulation logique des grammaires catégorielles est due aux travaux du mathématicien Lambek [67], notamment le système déductif qui porte son nom. L'idée cruciale introduite par Lambek consiste à interpréter les *types syntaxiques* en tant que *formules* et considérer ainsi les constructeurs de types comme des connecteurs logiques. Outre les opérateurs bidirectionnels / et \ qui représentent les deux variantes non-commutatives de l'implication linéaire, le calcul de Lambek (noté **L** en abrégé) introduit également le connecteur dual • (i.e., produit) : une expression porte le type $A \bullet B$ si elle résulte de la concaténation de deux entités de types respectifs A et B.
Le calcul **L** est dédié à la modélisation de la syntaxe des langues naturelles, il est par conséquent basé sur une logique sensible aux ressources qui prend en compte aussi bien le nombre que l'ordre des ressources agencées. Le processus d'analyse est, quant à lui, réduit à la recherche de preuves dans le système déductif considéré. Les règles d'inférence de cette logique viennent pour *compléter* les règles de réduction du calcul *AB* afin de décrire formellement le comportement exact des différents connecteurs syntaxiques. A titre d'exemple, l'interprétation naturelle du sens respectif des deux opérateurs / et \ peut se résumer comme suit :

Une expression est de type A/B (*resp.* $B\backslash A$) **si et seulement si** *elle produit une entité de type A quand elle est suivie* (*resp. précédée*) *par une expression de type B.*

Les règles du calcul *AB* s'avèrent être insuffisantes puisqu'elles ne reflètent que la moitié de cette équivalence (i.e., le sens '**seulement si**'). L'approche de Lambek a pour objectif d'introduire des règles duales afin d'encoder la moitié qui reste (i.e., le sens '**si**').
L'objectif de cette section est de présenter brièvement la syntaxe du calcul de Lambek. Les lecteurs intéressés par l'étude approfondie de ce formalisme, notamment ses différentes méta-propriétés (e.g., interprétation dans les modèles de Kripke, cohérence et complétude) peuvent consulter [66, 97, 77].

La présentation la plus intuitive pour les règles d'inférence de la logique **L** est indubitablement celle de la déduction naturelle. Dans cette présentation, les règles manipulent des séquents de la forme $\Gamma \vdash A$ où :

- Le contexte Γ est une séquence ordonnée de ressources linguistiques typées par des catégories syntaxiques.
- Le symbole $\vdash$ représente l'opérateur de dérivabilité en déduction naturelle.
- La conclusion A est un type syntaxique.

Notation 2.1.1 *Dans toute la partie consacrée au calcul de Lambek, les lettres latines (A, B, ...) représenteront des types syntaxiques tandis que les lettres grecques (*Γ, Σ, *...) seront utilisées pour symboliser des listes de formules étiquetées.*

Nous présentons, dans ce qui suit, l'ensemble de règles d'inférence de la logique **L** décrites dans le style de la déduction naturelle (où Γ et Δ représentent des listes non-vides de formules tandis que Σ et Σ' peuvent être éventuellement vides) :

Définition 2.1.1 *Règles d'inférence de la logique* ***L*** *(déduction naturelle)*

$$\frac{}{a : A \vdash A}\ Ax$$

$$\frac{\Gamma \vdash A/B \quad \Delta \vdash B}{\Gamma, \Delta \vdash A}\ /E \qquad \frac{\Gamma,\ x : B \vdash A}{\Gamma \vdash A/B}\ /I$$

$$\frac{\Gamma \vdash B \quad \Delta \vdash B\backslash A}{\Gamma,\ \Delta \vdash A}\ \backslash E \qquad \frac{x : B,\ \Gamma \vdash A}{\Gamma \vdash B\backslash A}\ \backslash I$$

$$\frac{\Delta \vdash A \bullet B \quad \Sigma,\ x : A,\ y : B,\ \Sigma' \vdash C}{\Sigma,\ \Delta,\ \Sigma' \vdash C}\ \bullet E \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma,\ \Delta \vdash A \bullet B}\ \bullet I$$

Le noyau logique comprend les règles d'élimination et d'introduction des connecteurs binaires considérés. A titre d'exemple, les règles d'élimination des implications / et \ coïncident exactement avec les règles de réduction du calcul *AB*. D'un point de vue logique, ces dernières ne sont autres que des variantes linéaires et non-commutatives de la fameuse règle de modus-ponens (i.e., à partir de $A \rightarrow B$ et A déduire B). Elles décrivent la manière dont une expression incomplète (de type fonctionnel) se combine avec son argument pour former une entité composée.
Il est important de remarquer que les règles d'élimination ne modélisent qu'une facette du comportement des connecteurs syntaxiques, en effet, elles caractérisent la façon dont les expressions de types composés (e.g., A/B, $B\backslash A$, $A \bullet B$) peuvent être *utilisées* dans une dérivation. L'originalité du calcul de Lambek réside dans la définition de règles complémentaires dites d'introduction, qui spécifient comment *dériver* des expressions de types composés. Par exemple, les règles d'introduction des implications bidirectionnelles / et \ sont d'une grande utilité car elles permettent d'introduire la technique de *raisonnement hypothétique*. Ainsi, afin de prouver qu'une expression est de type A/B (resp. $B\backslash A$), il suffit de montrer qu'en concaténant cette expression avec un élément quelconque de type B à sa droite (resp. à sa gauche) on obtient une entité de type A.
Le calcul de Lambek peut être utilisé comme une grammaire en définissant un lexique approprié, conçu pour associer à chaque mot un ensemble fini de types syntaxiques. Dans une telle grammaire, un énoncé $m_1,...,m_n$ sera considéré comme une expression syntaxiquement correcte portant le type F si et seulement si pour chaque mot m_i il existe un type A_i qui lui est attribué par le lexique de sorte que le séquent $(m_i : A_i),...,(m_n : A_n) \vdash F$ soit démontrable dans le calcul de Lambek.

Exemple 2.1.2 On considère la grammaire rigide (i.e., dont le lexique attribue à chaque mot un unique type syntaxique) définie par le lexique ci-après :

Mots	Types syntaxiques
que	(n\n)/(s/np)
tout	(s/(np\s))/n
sage, vertu	n
applique	(np\s)/np

Il est intéressant de noter que le lexique ci-dessus attribue au quantificateur '*tout*' le type d'ordre supérieur $(s/(np\backslash s))/n$ au lieu de lui assigner le type de premier ordre np/n. Un tel typage est particulièrement motivé par la dimension sémantique puisqu'il permet d'interpréter les quantificateurs généralisés (e.g., '*tout sage*') comme des ensembles de propriétés sur les individus et non comme de simples individus (cf. section 2.1.3).
A partir de ce lexique, on peut montrer que la phrase relative '*que tout sage applique*' est bien formée et porte le type syntaxique $n\backslash n$, jouant ainsi le même rôle grammatical que les adjectifs postnominaux (e.g., '*vertu que tout sage applique*').

Les étapes principales de la dérivation syntaxique sont présentées ci-après. Pour plus de lisibilité, on manipule des contextes composés de ressources linguistiques dont les types sont implicites, ce qui ne pose aucune ambiguïté car notre grammaire est rigide. En particulier, les axiomes logiques $m : A \vdash A$ seront représentés de manière abrégée sous la forme de jugements de typage $m \vdash A$. En outre, nous suggérons de numéroter les différentes hypothèses introduites au cours de la dérivation tout en mentionnant le numéro approprié à la phase de déchargement de chaque hypothèse.

$$\dfrac{\dfrac{\overline{que \vdash (n\backslash n)/(s/np)}\ Ax \qquad \dfrac{\dfrac{\dfrac{\overline{tout \vdash s/(np\backslash s)/n}\ Ax \quad \overline{sage \vdash n}\ Ax}{tout,\ sage \vdash s/(np\backslash s)}\ /E \quad \dfrac{\overline{applique \vdash (np\backslash s)/np}\ Ax \quad \dfrac{\overline{}}{[x : np \vdash np]^1}\ Ax}{applique,\ (x : np) \vdash np\backslash s}\ /E}{tout,\ sage,\ applique,\ (x : np) \vdash s}\ /I,1}{tout,\ sage,\ applique \vdash s/np}\ /E}{que,\ tout,\ sage,\ applique \vdash n\backslash n}\ /E$$

La dérivation syntaxique peut être vue comme un arbre dont la racine est le séquent qu'on souhaite analyser et les feuilles sont des axiomes logiques provenant d'éléments lexicaux ou de ressources hypothétiques. Afin de montrer que la clause incomplète '*tout sage applique*' est un argument adéquat pour le pronom '*que*', il faut prouver que son type est *s/np* (phrase à laquelle il manque un objet à droite). A cette fin, la technique du raisonnement hypothétique est appliquée. Une ressource fictive *x* de type *np* est donc introduite dans le contexte et placée à son extrême droite. Il suffit ainsi de démontrer que '*tout sage applique x*' est une phrase correcte, chose qui se fait assez aisément par simple application de la règle $/E$.

A partir du noyau réduit de la logique **L**, il est possible de prouver une grande variété de règles dérivées ayant diverses applications linguistiques. En particulier, certaines règles de réduction, employées comme primitives dans les grammaires catégorielles combinatoires (**CCG**)[3] de Steedman [103], peuvent être aisément engendrées dans la logique de Lambek. Ceci est le cas des règles suivantes (les notations placées entre parenthèses représentent les identifiants effectifs de ces règles dans **CCG**.)

Définition 2.1.2 *Quelques règles dérivées de* ***L***
- Composition (>B). u : A/B, v : $B/C \vdash A/C$
- Composition (<B). u : $C\backslash B$, v : $B\backslash A \vdash C\backslash A$
- Montée de type (>T). a : $A \vdash B/(A\backslash B)$
- Montée de type (<T). a : $A \vdash (B/A)\backslash B$

A titre d'exemple, la figure 2 illustre comment on peut dériver dans **L**, les deux règles respectives de composition (>B) et de montée de type (>T).

$$\dfrac{\dfrac{\overline{u : A/B \vdash A/B}\ Ax \quad \dfrac{\overline{v : B/C \vdash B/C}\ Ax \quad \overline{x : C \vdash C}\ Ax}{v : B/C\ ,\ x : C \vdash B}\ /E}{u : A/B\ ,\ v : B/C\ ,\ x : C \vdash A}\ /I}{u : A/B\ ,\ v : B/C \vdash A/C}\ /I \qquad \dfrac{\dfrac{\overline{a : A \vdash A}\ Ax \quad \overline{x : A\backslash B \vdash A\backslash B}\ Ax}{a : A\ ,\ x : A\backslash B \vdash B}\ \backslash E}{a : A \vdash B/(A\backslash B)}\ /I$$

FIG. 2: Dérivation des règles (>B) et (>T) dans **L**

[3] Les **CCG** étendent le calcul *AB* en introduisant de nouvelles règles de réduction dont celles présentées dans la définition 2.1.2.

Les règles d'inférence de la logique **L** peuvent être également présentées dans le style du calcul de séquents (i.e., calcul de Gentzen) (où Δ et Γ sont des listes non vides de formules, tandis que Σ et Σ' peuvent être vides) :

Définition 2.1.3 *Règles d'inférence de la logique **L** (calcul des séquents)*

$$\frac{}{x : A \Rightarrow A}\ Ax \qquad \frac{\Delta \Rightarrow A \quad \Sigma, x : A, \Sigma' \Rightarrow C}{\Sigma, \Delta, \Sigma' \Rightarrow C}\ Cut$$

$$\frac{\Delta \Rightarrow B \quad \Sigma, x : A, \Sigma' \Rightarrow C}{\Sigma, f : A/B, \Delta, \Sigma' \Rightarrow C}\ /L \qquad \frac{\Gamma,\ x : B \vdash A}{\Gamma \vdash A/B}\ /R$$

$$\frac{\Delta \Rightarrow B \quad \Sigma, x : A, \Sigma' \Rightarrow C}{\Sigma, \Delta, f : B\backslash A, \Sigma' \Rightarrow C}\ \backslash L \qquad \frac{x : B,\ \Gamma \Rightarrow A}{\Gamma \Rightarrow B\backslash A}\ \backslash R$$

$$\frac{\Sigma, x : A, y : B, \Sigma' \Rightarrow C}{\Sigma, f : A \bullet B, \Sigma' \Rightarrow C}\ \bullet L \qquad \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{\Gamma,\ \Delta \Rightarrow A \bullet B}\ \bullet R$$

Le calcul de Gentzen permet de distinguer entre trois types de règles :

- Règles d'identité : elles modélisent les deux propriétés basiques de la relation de dérivabilité $\Rightarrow$, à savoir la réflexivité qui est exprimée par la règle axiome *Ax* et la transitivité introduite par la règle de coupure *Cut*.
- Règles gauches : elles permettent d'introduire les connecteurs logiques dans le contexte (i.e., à gauche de $\Rightarrow$).
- Règles droites : elles permettent d'introduire les connecteurs logiques dans la formule conclusion (i.e., à droite de $\Rightarrow$). Elles sont totalement identiques aux règles d'introduction de la déduction naturelle.

Même s'il existe des différences notoires entre les deux présentations de la logique de Lambek (i.e., la déduction naturelle et le calcul de Gentzen), ces systèmes déductifs sont équivalents dans la mesure où ils engendrent exactement les mêmes théorèmes [50, 97]. Précisons toutefois que l'utilisation de chacun de ces deux styles est privilégiée dans des contextes particuliers. D'une part, le calcul de séquents est doté de propriétés intéressantes d'un point de vue computationnel (e.g., élimination des coupures, propriété de sous-formule) : il s'avère être le plus adéquat pour la recherche automatique de preuves [67, 97, 77]. D'autre part, la déduction naturelle a le mérite d'être intuitive et simple à manier. En outre, elle présente une interface aisée avec la sémantique de Montague comme nous le verrons dans la section suivante.

2.1.3 Lien avec la sémantique de Montague

Un des atouts majeurs des grammaires de Lambek provient de l'interface aisée entre la syntaxe et la sémantique due à la correspondance de Curry-Howard [20]. En effet, les dérivations syntaxiques considérées ne sont autres que des déductions dans la logique de Lambek qui peuvent naturellement être plongées dans la logique intuitionniste et être ainsi encodées par des λ-termes simplement typés représentant la sémantique des expressions analysées. Un tel plongement est effectué en traduisant les types syntaxiques en des types sémantiques non-directionnels appropriés. Il n'est pas difficile de voir que les deux connecteurs bidirectionnels / et \ seront traduits par l'implication intuitionniste (i.e., flèche intuitionniste) alors que le produit $\bullet$ sera simulé par la

conjonction (i.e., produit cartésien). En suivant Montague [73, 47], l'ensemble des types sémantiques sera basé sur deux catégories atomiques, à savoir *e* (i.e., ensemble des individus) et *t* (i.e., valeurs de vérité).
Formellement, la traduction des types syntaxiques en des types sémantiques adéquats se fait moyennant un morphisme de types noté $()^*$ qui est défini de manière compositionnelle comme l'illustre le tableau suivant :

Types Atomiques	Types composés
$(np)^*=e$	$(A/B)^*=B^*\rightarrow A^*$
$(n)^*=e\rightarrow t$	$(B\backslash A)^*=B^*\rightarrow A^*$
$(s)^*=t$	$(A\bullet B)^*=A^*\times B^*$

Une grammaire de Lambek est ainsi définie par un lexique à deux dimensions englobant aussi bien la description du comportement syntaxique des mots que leurs représentations sémantiques (sous forme de λ-termes simplement typés). Ce dernier associe à chaque mot un ensemble fini de paires (F_i, t_i) composées d'un type syntaxique F_i et d'un λ-terme sémantique t_i dont le type n'est autre que $F_i{}^*$ (i.e., le type sémantique compatible avec le type syntaxique F_i).
L'interface syntaxe/sémantique est établie en transformant les démonstrations de la logique **L** en des démonstrations de la logique intuitionniste à l'aide du morphisme de types $()^*$ qui remplace chaque connecteur de **L** par son analogue intuitionniste. Grâce à l'isomorphisme de Curry-Howard, la dérivation obtenue en logique intuitionniste est représentée par un λ-terme ; en effet, à chaque règle de déduction du système correspond une étape de calcul du λ-calcul simplement typé. A titre d'exemple, l'élimination de $\rightarrow$ (traduisant les règles $/E$ et $\backslash E$) coïncide avec l'application fonctionnelle tandis que l'introduction de $\rightarrow$ (qui résulte de $/I$ et $\backslash I$) correspond à l'abstraction. La figure 3 montre deux exemples de règles d'inférence du système **L** ainsi que l'interface syntaxe/sémantique associée (pour chaque variable syntaxique typée $x : A$, x^* est une variable sémantique de type A^*).

Syntaxe	Sémantique
$\dfrac{\Gamma \vdash A/B \;\; \Delta \vdash B}{\Gamma, \Delta \vdash A}\; /E$	$\dfrac{\Gamma^* \vdash f : B^* \rightarrow A^* \;\; \Delta^* \vdash b : B^*}{\Gamma^*, \Delta^* \vdash (f\ b) : A^*} \rightarrow E$
$\dfrac{\Gamma, x : B \vdash A}{\Gamma \vdash A/B}\; /I$	$\dfrac{\Gamma^*, x^* : B^* \vdash f : A^*}{\Gamma^* \vdash \lambda x^*.f : B^* \rightarrow A^*} \rightarrow I$

FIG. 3: Interface Syntaxe/Sémantique dans L

Le sens final de l'énoncé analysé est calculé à partir du λ-terme obtenu en substituant chaque variable sémantique associée à un mot donné par sa représentation sémantique attribuée par le lexique. Les grammaires de Lambek respectent ainsi le principe de compositionnalité de Frege [46] stipulant que le sens d'une construction grammaticale ne dépend que de son arbre de dérivation syntaxique et des sémantiques partielles de ses sous-expressions.

Exemple 2.1.3 Nous reprenons, dans cette partie, l'exemple 2.1.2 en tenant compte de la dimension sémantique. Le lexique précédemment défini est enrichi comme suit :

Mots	Types syntaxiques	Types sémantiques	Sémantique
vertu	n	e→t	λx. **vertu**(x)
que	$(n\backslash n)/(s/np)$	(e→t)→(e→t)→(e→t)	λP. λQ. λx. P(x)$\wedge$Q(x)
tout	$(s/(np\backslash s))/n$	(e→t)→(e→t)→t	λP. λQ. $\forall$x. P(x)$\Rightarrow$Q(x)
sage	n	e→t	λx. **sage**(x)
applique	$(np\backslash s)/np$	e→e→t	λx. λy. **applique**(y,x)

Parmi les représentations sémantiques figurant dans le lexique ci-dessus, quelques unes font intervenir des constantes sémantiques imprimées en gras. Ainsi, la constante **sage** (resp. **vertu**) est un prédicat à un seul argument représentant un sous-ensemble d'individus ayant la propriété d'être sage (resp. d'être une vertu). En revanche, **applique** est un prédicat à deux arguments interprété dans un modèle donné par l'ensemble de paires d'individus telles que la première composante appliquerait la seconde. Certains mots, dont les quantificateurs (e.g., '*tout*') et les pronoms relatifs (e.g., '*que*') ont une forme logique formelle définie en utilisant les connecteurs logiques usuels[4](e.g., $\Rightarrow$ (implication intuitionniste), $\wedge$ (conjonction), $\forall$ (quantification universelle)). Ces derniers mots ont une sémantique universelle dont l'interprétation est invariable d'un modèle à un autre [47].
La dérivation syntaxique de l'exemple 2.1.2 peut être aisément traduite en logique intuitionniste engendrant ainsi la démonstration suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{tout^* \vdash tout^* : ...}\,Ax \quad \dfrac{}{sage^* \vdash e \to t}\,Ax}{tout^*,\ sage^* \vdash tout^*(sage^*) : (e \to t) \to t}\to E \quad \dfrac{\dfrac{}{applique^* \vdash applique^* : ...}\,Ax \quad \dfrac{}{[x^* : e \vdash x^* : e]^1}\,Ax}{applique^*,\ (x^* : e) \vdash applique^*(x^*) : e \to t}\to E}{tout^*,\ sage^*,\ applique^*,\ (x^* : e) \vdash tout^*(sage^*)(applique^*(x^*)) : t}\to E}{tout^*,\ sage^*,\ applique^* \vdash \lambda x^*.\ tout^*(sage^*)(applique^*(x^*)) : e \to t}\to I,1}{que^*,\ tout^*,\ sage^*,\ applique^* \vdash que^*(\lambda x^*.\ tout^*(sage^*)(applique^*(x^*))) : (e \to t) \to e \to t}\to E$$

La structure prédicative de la subordonnée relative '*que tout sage ignore*' coïncide alors avec le terme suivant : $que^*(\lambda x^*.\ tout^*(sage^*)(applique^*(x^*)))$.
La sémantique finale de l'énoncé analysé est obtenue en remplaçant chaque variable $\dot{m}^*$ par la sémantique lexicale du mot m. Le λ-terme résultant est représenté sous sa forme normale comme suit :

$$(\lambda P.\ \lambda Q.\ \lambda x.\ P(x) \wedge Q(x))(\lambda x^*.\ ((\lambda P.\ \lambda Q.\ \forall x.\ P(x) \Rightarrow Q(x))(\lambda x.\ \mathbf{sage}(x))(\lambda y.\ \mathbf{applique}(y, x^*))))$$
$$\downarrow_\beta$$
$$\lambda Q.\ \lambda x.\ (\forall y.\ \mathbf{sage}(y) \Rightarrow \mathbf{applique}(y, x)) \wedge Q(x)$$

A partir de cette sémantique, on peut facilement formaliser le sens du nom commun modifié '*vertu que tout sage applique*' représentant intuitivement une propriété qui caractérise les entités appliquées par la totalité des sages tout en étant des vertus. Le sens formel est encodé par le terme ci-après :

$$\lambda x.\ (\forall y.\ \mathbf{sage}(y) \Rightarrow \mathbf{applique}(y, x)) \wedge \mathbf{vertu}(x)\ :\ e \to t$$

2.1.4 Restrictions des systèmes catégoriels de base

Malgré son élégance et sa simplicité séduisantes, le système **L** ne s'avère pas très adapté à l'analyse des langues naturelles, et ce pour diverses raisons. D'un côté, ce système a une capacité

[4]Notons que $\Rightarrow$ et $\wedge$ sont de type $t \to t \to t$ tandis que $\forall$ est de type $(e \to t) \to t$. Pour plus de clarté, nous optons pour la notation mathématique de $\forall$ ($\lambda x.\ P(x)$), à savoir $\forall x.\ P(x)$.

générative très limitée. En effet, Pentus a prouvé dans [91] que les grammaires de Lambek sont faiblement équivalentes aux grammaires hors-contextes, alors que les linguistes s'accordent à dire que les langues naturelles font partie des langages légèrement contextuels [63]. Par conséquent, le calcul de Lambek se trouve dans l'incapacité de traiter certains phénomènes linguistiques avancés tels que les dépendances croisées du néerlandais ou du suisse allemand qui génèrent des langages non reconnaissables par une simple grammaire algébrique [100].
D'autre part, l'absence totale de toute forme de commutativité de ressources accentue la rigidité du système **L**. A titre d'exemple, de telles grammaires ne peuvent rendre compte du phénomène linguistique de l'extraction médiane illustrée par la phrase '*L'humilité est la vertu que tout sage applique _ constamment*', étant donné que seules les hypothèses périphériques (la plus à gauche et la plus à droite du contexte) peuvent être abstraites moyennant les règles d'introduction des connecteurs / et \.
D'un autre côté, les grammaires de Lambek souffrent également d'un problème de sur-génération résultant de l'associativité non contrôlée de l'opérateur ',' qui regroupe les ressources dans le contexte. En effet, on peut facilement vérifier que certaines phrases qui violent la contrainte de la structure de coordination [98] (i.e., contrainte selon laquelle les deux membres d'une coordination doivent avoir le même type) sont facilement dérivables dans **L**. Ceci est le cas de la phrase agrammaticale *(*La vertu que Marie dort et Jean applique _ est l'humilité*), comme l'illustre le fragment de la dérivation de la figure 4. Lambek a tenté de limiter ces lacunes en concevant trois

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Marie \vdash np}\,Ax \quad \dfrac{}{dort \vdash np\backslash s}\,Ax}{Marie,\ dort \vdash s}\,\backslash E \quad \dfrac{\dfrac{}{et \vdash (s\backslash s)/s}\,Ax \quad \dfrac{\dfrac{}{Jean \vdash np}\,Ax \quad \dfrac{\dfrac{}{applique \vdash (np\backslash s)/np}\,Ax \quad \dfrac{}{[x : np \vdash np]^1}\,Ax}{applique,\ (x : np) \vdash np\backslash s}\,/E}{Jean,\ applique,\ (x : np) \vdash s}\,\backslash E}{et,\ Jean,\ applique,\ (x : np) \vdash s\backslash s}\,/E}{Marie,\ dort,\ et,\ Jean,\ applique,\ (x : np) \vdash s}\,\backslash E}{Marie,\ dort,\ et,\ Jean,\ applique \vdash s/np}\,/I, 1}{que,\ Marie,\ dort,\ et,\ Jean,\ applique \vdash n\backslash n}\,/E$$

FIG. 4: Exemple de sur-génération du système **L**

nouvelles variantes de son système logique (**NL**, **NLP** et **LP**), obtenues en attribuant à l'opérateur ',' différentes propriétés structurelles (e.g., Ass : associativité, Com : commutativité). Le schéma suivant récapitule la relation entre les quatre variantes du calcul de Lambek :

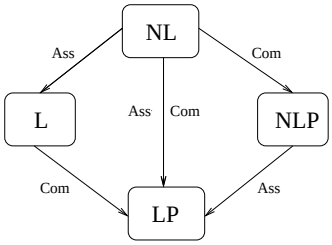


FIG. 5: Relation entre les différents systèmes de Lambek

Toutefois, l'accès global aux différentes propriétés structurelles n'est point pertinent d'un point de vue linguistique. A titre d'exemple, on a certes besoin de la commutativité des ressources dans certains contextes particuliers (e.g., notamment pour inverser l'ordre entre un adjectif et un nom commun en italien), néanmoins, le libre accès à cette propriété (comme c'est le cas dans le système **LP**) est incontestablement néfaste puisque l'ordre des mots est souvent strict dans un grand nombre de constructions grammaticales.
Plusieurs solutions ont été proposées pour enrichir le calcul de Lambek et contrôler la gestion des ressources linguistiques, entre autres par Hepple [54], Morrill [83] et Moortgat [77, 79, 78]. C'est la dernière proposition qui sera notre objet d'étude dans cette partie de la thèse.

2.2 Vers une Logique Multimodale enrichie

Cette section est entièrement dédiée à la présentation des grammaires catégorielles multimodales [77]. Dans un premier temps, nous introduirons formellement la syntaxe des différentes composantes de ces grammaires en veillant à illustrer les nouveaux concepts par des exemples concrets. Précisons néanmoins que nous nous focaliserons uniquement sur la description des éléments qui seront utiles à la compréhension du chapitre suivant. Les lecteurs intéressés par une étude approfondie des différentes propriétés de cette logique sont invités à consulter [77, 66, 81].

2.2.1 Définitions préliminaires

Afin de limiter les faiblesses du calcul de Lambek, Moortgat a défini un système logique enrichi par l'introduction de deux opérateurs unaires duaux ($\Box$, $\Diamond$) [77]. En outre, tous les connecteurs logiques considérés sont indexés par un indice pris dans un ensemble de *modes de composition*, ce qui nous permet d'obtenir une famille d'opérateurs ($/_i$, $\backslash_i$, $\bullet_i$, $\Box_j$, $\Diamond_j$) qui cohabitent. C'est pour cette raison que la logique sous-jacente est dite *multimodale*. Formellement, l'ensemble des types syntaxiques est défini comme suit :

Définition 2.2.1 *Soient $\mathcal{A}$, $\mathcal{I}$ et $\mathcal{J}$ trois ensembles finis. $\mathcal{A}$ est l'ensemble des types atomiques, $\mathcal{I}$ l'ensemble des modes de composition binaires et $\mathcal{J}$ celui des modes de composition unaires. L'ensemble des types syntaxiques $\mathcal{F}$ est défini comme étant le plus petit ensemble comprenant $\mathcal{A}$ et clos par rapport à la famille de connecteurs ($/_i$, $\backslash_i$, $\bullet_i$, $\Diamond_j$, $\Box_j$) (où $i \in \mathcal{I}$ et $j \in \mathcal{J}$) :*

$$\mathcal{F} := \mathcal{A} \mid \mathcal{F}/_i\mathcal{F} \mid \mathcal{F} \backslash_i \mathcal{F} \mid \mathcal{F} \bullet_i \mathcal{F} \mid \Diamond_j \mathcal{F} \mid \Box_j \mathcal{F}$$

Exemple 2.2.1 *Si $\{np, n\} \subseteq \mathcal{A}$, $a \in \mathcal{I}$ et $sg \in \mathcal{J}$, alors $(\Box_{sg} np /_a \Box_{sg} n) \in \mathcal{F}$.*

Les connecteurs unaires servent, entre autres, à coder les traits morphosyntaxiques (genre, nombre ...) [56]. Une formule indexée par $\Diamond_j$ ou $\Box_j$ représentera ainsi une formule distinguée portant le trait *j*. On peut, à titre d'exemple, attribuer au nom commun '*vertu*' le type enrichi $\Box_{sg}\, n$ pour préciser qu'il s'agit d'un nom commun singulier. De la même façon, on associe à l'article défini '*la*' la catégorie $\Box_{sg} np /_a \Box_{sg} n$ pour le forcer à se combiner avec des noms communs singuliers. Un tel codage de traits nous permet d'éviter d'engendrer des énoncés mal formés tels que *(*la vertus*) où on est incapable d'unifier le trait pluriel du nom commun effectif '*vertus*' avec le trait singulier de l'argument attendu par l'article '*la*'.

Les contextes considérés dans les systèmes multimodaux sont plus structurés qu'une simple liste plate. En effet, à chaque famille de connecteurs logiques on associe un connecteur structurel qui permet de regrouper les ressources dans une configuration précise. Les contextes prennent alors la forme d'arbres binaires où chaque noeud interne coïncide soit avec l'opérateur structurel binaire $(\ , \ )^i$ soit avec l'opérateur structurel unaire $< >_j$. Les feuilles de ces arbres ne sont autres que des ressources linguistiques typées par des types syntaxiques. Ceci nous conduit à la définition suivante :

Définition 2.2.2 *L'ensemble $\mathcal{S}$ des contextes structurés est défini inductivement comme ceci :*

$$\mathcal{S} := (r : \mathcal{F}) \mid (\mathcal{S}, \mathcal{S})^i \mid < \mathcal{S} >_j$$

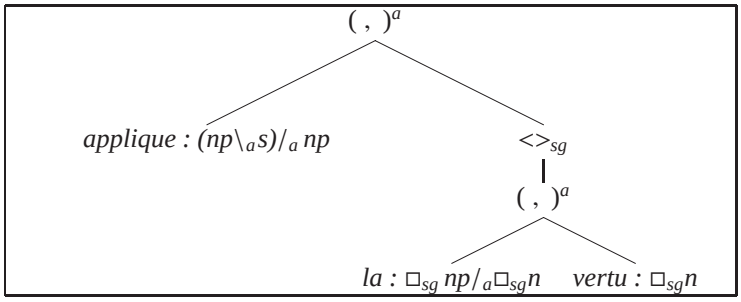
Notation 2.2.1 *Soient r_1, r_2,...,r_n des variables représentant des ressources linguistiques et F_1, F_2, ...,F_n des types syntaxiques. On note par Struct($(r_1 : F_1)$,..., $(r_n : F_n)$) l'ensemble de contextes structurés ayant comme feuilles $(r_1 : F_1)$,..., $(r_n : F_n)$ dans cet ordre.*

Les contextes arborescents sont plus informatifs que les simples listes de ressources linguistiques car ils peuvent encapsuler la structure de la phrase et mettre en évidence leurs différents constituants comme l'illustre l'exemple qui suit :

Exemple 2.2.2 La phrase verbale '*applique la vertu*' peut être structurée moyennant le contexte arborescent encodé par le terme suivant :

$$(applique \ : \ (np\backslash_a s)/_a \ np \ , \ < (la \ : \ \Box_{sg} np/_a \Box_{sg} n \ , \ vertu \ : \ \Box_{sg} n)^a >_{sg})^a$$

Ce terme peut être également représenté de manière hiérarchique sous forme d'un arbre binaire illustré ci-après :



2.2.2 Noyau logique

Les règles d'inférence de la logique multimodale sont obtenues en enrichissant les règles de la logique **L** afin de décrire notamment le comportement des deux opérateurs unaires considérés. Les séquents manipulés par ces règles obéissent à la syntaxe suivante :

Définition 2.2.3 *Un séquent est formé d'une paire (Γ, A) où $\Gamma \in \mathcal{S}$ et $A \in \mathcal{F}$. Un tel séquent est noté $\Gamma \vdash A$ en déduction naturelle et $\Gamma \Rightarrow A$ en calcul des séquents.*

Il est intéressant de souligner que les règles de la logique multimodale peuvent être présentées dans plusieurs styles équivalents, dont la présentation axiomatique, les réseaux de démonstration, la déduction naturelle et le calcul des séquents [77, 66, 81]. Toutefois, dans cette thèse, nous nous

focaliserons uniquement sur les deux derniers formats qui s'apparentent le plus à la théorie des types.
Nous présentons, dans ce qui suit, l'ensemble des règles d'inférence de la logique multimodale aussi bien dans le style de la déduction naturelle (définition 2.2.4) que dans le calcul de Gentzen (définition 2.2.5). Précisons que certaines de ces règles, telle $\diamond_i E$ ou $\diamond_i L$, font intervenir des constructions de la forme $\Gamma[\Delta]$. Dans ce cas, $\Gamma[\ ]$ représente un contexte linéaire (un contexte avec trou) [59] qui définit l'emplacement où sera appliquée la règle en question.

Définition 2.2.4 *Règles d'inférence de la logique multimodale (déduction naturelle)*

$$\frac{}{x : A \vdash A}\ Ax$$

$$\frac{\Gamma \vdash A/_i B \quad \Delta \vdash B}{(\Gamma, \Delta)^i \vdash A}\ /_i E \qquad \frac{(\Gamma, x : B)^i \vdash A}{\Gamma \vdash A/_i B}\ /_i I$$

$$\frac{\Gamma \vdash B \quad \Delta \vdash B\backslash_i A}{(\Gamma, \Delta)^i \vdash A}\ \backslash_i E \qquad \frac{(x : B, \Gamma)^i \vdash A}{\Gamma \vdash B\backslash_i A}\ \backslash_i I$$

$$\frac{\Delta \vdash A \bullet_i B \quad \Gamma[(a : A, b : B)^i] \vdash C}{\Gamma[\Delta] \vdash C}\ \bullet_i E \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma, \Delta)^i \vdash A \bullet_i B}\ \bullet_i I$$

$$\frac{\Delta \vdash \diamond_j A \quad \Gamma[< a : A >_j] \vdash C}{\Gamma[\Delta] \vdash C}\ \diamond_j E \qquad \frac{\Gamma \vdash A}{< \Gamma >_j \vdash \diamond_j A}\ \diamond_j I$$

$$\frac{\Gamma \vdash \Box_j A}{< \Gamma >_j \vdash A}\ \Box_j E \qquad \frac{< \Gamma >_j \vdash A}{\Gamma \vdash \Box_j A}\ \Box_j I$$

Définition 2.2.5 *Règles de la logique multimodale (calcul des séquents)*

$$\frac{}{x : A \Rightarrow A}\ Ax \qquad \frac{\Delta \Rightarrow A \quad \Gamma[x : A] \Rightarrow C}{\Gamma[\Delta] \Rightarrow C}\ Cut$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[x : A] \Rightarrow C}{\Gamma[(u : A/_i B, \Delta)^i] \Rightarrow C}\ /_i L \qquad \frac{(\Gamma, x : B)^i \Rightarrow A}{\Gamma \Rightarrow A/_i B}\ /_i R$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[x : A] \Rightarrow C}{\Gamma[(\Delta, u : B\backslash_i A)^i] \Rightarrow C}\ \backslash_i L \qquad \frac{(x : B, \Gamma)^i \Rightarrow A}{\Gamma \Rightarrow B\backslash_i A}\ \backslash_i R$$

$$\frac{\Gamma[(x : A, y : B)^i] \Rightarrow C}{\Gamma[u : A \bullet_i B] \Rightarrow C}\ \bullet_i L \qquad \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{(\Gamma, \Delta)^i \Rightarrow A \bullet_i B}\ \bullet_i R$$

$$\frac{\Gamma[< x : A >_j] \Rightarrow C}{\Gamma[u : \diamond_j A] \Rightarrow C}\ \diamond_j L \qquad \frac{\Gamma \Rightarrow A}{< \Gamma >_j \Rightarrow \diamond_j A}\ \diamond_j R$$

$$\frac{\Gamma[x : A] \Rightarrow C}{\Gamma[< u : \Box_j A >_j] \Rightarrow C}\ \Box_j L \qquad \frac{< \Gamma >_j \Rightarrow A}{\Gamma \Rightarrow \Box_j A}\ \Box_j R$$

A partir de ces règles d'inférence, il est relativement aisé de dériver la relation de dualité, reliant les deux opérateurs de contrôle, qui s'exprime comme suit : $\Diamond_j\Box_j A \vdash A \vdash \Box_j\Diamond_j A$. La preuve de cette règle dérivée est établie en deux parties ci-dessous :

$$\dfrac{\dfrac{}{a : \Diamond_j\Box_j A \vdash \Diamond_j\Box_j A}\ Ax \quad \dfrac{\dfrac{}{x : \Box_j A \vdash \Box_j A}\ Ax}{< x : \Box_j A >_j \vdash A}\ \Box_j E}{a : \Diamond_j\Box_j A \vdash A}\ \Diamond_j E \qquad \dfrac{\dfrac{}{a : A \vdash A}\ Ax}{\dfrac{< a : A >_j \vdash \Diamond_j A}{a : A \vdash \Box_j\Diamond_j A}\ \Box_j I}\ \Diamond_j I$$

2.2.3 Règles structurelles

Le noyau logique précédemment présenté ne suffit pas pour rendre compte des différents phénomènes de la langue naturelle. Afin d'augmenter le pouvoir expressif des grammaires multimodales, on rajoute un ensemble de règles structurelles qui permettent la gestion des ressources dans le contexte, introduisant ainsi une certaine flexibilité locale (associativité, commutativité etc.). Les grammaires multimodales sont par conséquent paramétrées par un lexique et un paquetage pertinent de règles structurelles ; toutefois, elles partagent le même système déductif sous-jacent.
Les règles structurelles introduites doivent être appliquées de manière contrôlée en présence de modes spécifiques pour éviter le problème de sur-génération. A titre d'exemple, on peut remplacer l'associativité globale dans **L** par une associativité restreinte attribuée uniquement au connecteur structurel $(,)^a$. En outre, les opérateurs structurels unaires $<>_j$ contribuent aussi à restreindre davantage le champ d'application des règles structurelles. Ainsi, on peut par exemple supposer que la commutativité de deux ressources n'a lieu que si la première est enveloppée par $<>_c$. L'utilisation des opérateurs unaires pour restreindre l'applicabilité des règles structurelles s'inspire fortement de l'utilisation des exponentielles en logique linéaire où la duplication et la contraction des ressources se font seulement en présence de formules décorées par les opérateurs (**!** , **?**) [48].
Formellement, chaque règle structurelle est définie par l'application d'une règle de réécriture R à un sous-contexte donné Δ_1, réorganisant ainsi ses ressources sous la forme d'une nouvelle configuration Δ_2. Le schéma général de l'application d'une règle structurelle[5] est donc le suivant :

$$\dfrac{\Delta_1 \xrightarrow{R} \Delta_2 \quad \Gamma[\Delta_2] \vdash C}{\Gamma[\Delta_1] \vdash C}\ S_R$$

La figure 6 illustre deux exemples concrets de règles de réécriture suivies de leurs traductions sous forme de règles structurelles.

$L_1(a)$	$C\Diamond(i,\ c)$
$((\Delta_1 , \Delta_2)^a , \Delta_3)^a \longrightarrow$ $(\Delta_1 , (\Delta_2 , \Delta_3)^a)^a$	$(< \Delta_1 >_c , \Delta_2)^i \longrightarrow$ $(\Delta_2 , < \Delta_1 >_c)^i$
$\dfrac{\Gamma[(\Delta_1 , (\Delta_2 , \Delta_3)^a)^a] \vdash C}{\Gamma[((\Delta_1 , \Delta_2)^a , \Delta_3)^a] \vdash C}$	$\dfrac{\Gamma[(\Delta_2 , < \Delta_1 >_c)^i] \vdash C}{\Gamma[(< \Delta_1 >_c , \Delta_2)^i] \vdash C}$

FIG. 6: Quelques règles structurelles

[5] Ce schéma est exprimé dans le style de la déduction naturelle ; il peut être directement adapté au calcul des séquents en remplaçant l'opérateur de dérivabilité $\vdash$ par $\Rightarrow$.

Les règles structurelles que nous considérons dans cette partie sont toutes sous la forme *faible de Sahlqvist* et vérifient les critères explicités ci-après :

Définition 2.2.6 *Une règle structurelle sous la forme faible de Sahlqvist [66] est une règle de la forme :*

$$\frac{\Gamma[\Psi'(\Delta_1,\ \Delta_2, ..., \Delta_n)] \vdash C}{\Gamma[\Psi(\Sigma_1,\ \Sigma_2, ..., \Sigma_m)] \vdash C}\ S_R$$

où :

1. *Pour tout i de {1..n} et pour tout j de {1..m}, Δ_i et Σ_j sont des variables représentant des contextes arborescents. En outre, $\Psi(\Sigma_1,\ \Sigma_2, ..., \Sigma_m)$ (resp. $\Psi'(\Delta_1,\ \Delta_2, ..., \Delta_n)$) est une configuration de Σ_1, Σ_2,..., et Σ_m (resp. Δ_1, Δ_2,..., et Δ_n) formée à partir des deux connecteurs structurels $(\ ,\ )^i$ et $<>_j$.*
2. *Pour tout i de {1..n}, il existe j de {1..m} tel que Δ_i=Σ_j (i.e., toutes les variables Δ_1, Δ_2,...,Δ_n apparaissent dans Σ_1, Σ_2,..., Σ_m).*
3. *Les variables Σ_1, Σ_2,... et Σ_m sont toutes distinctes.*
4. *La structure $\Psi'(\Delta_1,\ \Delta_2, ..., \Delta_n)$ comprend au moins un opérateur structurel.*

Toutes les règles structurelles sous la forme faible de Sahlqvist sont linéaires à gauche (puisqu'elles vérifient la contrainte (3)), toutefois, elles ne sont pas forcément linéaires. En effet, la linéarité exige une contrainte supplémentaire à savoir que Δ_1, Δ_2,...,Δ_n soit une permutation de Σ_1, Σ_2,...,Σ_m.

Exemple 2.2.3 *Les règles $L_1(a)$ et $C\Diamond(i,\ c)$ de la figure 6 sont sous la forme faible de Sahlqvist (elles sont même linéaires). La règle de contraction restreinte $MC(i, j)$, utilisée par Moortgat pour rendre compte des trous parasites [77] est, quant à elle, un exemple de règle non-linéaire qui est pourtant sous la forme faible de Sahlqvist :*

$$\frac{\Gamma[((\Delta_1, \Delta_3)^j, (\Delta_2, \Delta_3)^j)^i] \vdash C}{\Gamma[((\Delta_1, \Delta_2)^i, \Delta_3)^j] \vdash C}\ MC(i,\ j)$$

En revanche, les règles ci-dessous ne sont pas sous la forme faible de Sahlqvist puisqu'elles violent dans l'ordre les contraintes (1), (2), (3) et (4) de la définition 2.2.6 :

$$\frac{\Gamma[A] \vdash C}{\Gamma[(A/_i B) \bullet_i B] \vdash C}\ E_1(i) \qquad \frac{\Gamma[(\Delta, \Delta')^i] \vdash C}{\Gamma[\Delta] \vdash C}\ E_2(i)$$

$$\frac{\Gamma[(\Delta, \Delta)^j] \vdash C}{\Gamma[(\Delta, \Delta)^i] \vdash C}\ E_3(i,\ j) \qquad \frac{\Gamma[\Delta] \vdash C}{\Gamma[(\Delta, \Delta')^i] \vdash C}\ E_4(i)$$

2.2.4 Interface syntaxe/sémantique

L'enrichissement apporté à la logique de Lambek permet de conserver une interface syntaxe/sémantique aisée. Dans ce qui suit, nous présenterons de manière formelle le lien entre la syntaxe et le calcul des représentations sémantiques en logique multimodale.

Définition 2.2.7 *L'ensemble des types sémantiques $\mathcal{T}$ comprend les deux types de base* ***e*** *(individus) et* ***t*** *(valeurs de vérité), ainsi que tous les types construits à partir de ces derniers moyennant les deux connecteurs binaire $\rightarrow$ (flèche fonctionnelle) et $\times$ (produit cartésien) ainsi que le connecteur unaire $\dagger$:*

$$\mathcal{T} := e \mid t \mid \mathcal{T} \rightarrow \mathcal{T} \mid \mathcal{T} \times \mathcal{T} \mid \dagger\mathcal{T}$$

Le constructeur $\dagger$ est considéré comme l'analogue sémantique des deux opérateurs de contrôle ($\Diamond_j$, $\Box_j$). Il peut être utilisé pour encoder la sémantique intentionnelle de Montague [83] : dans ce cas le type $\dagger(ty)$[6] n'est autre que celui des fonctions de l'ensemble de mondes possibles vers le type *ty* [47].

Exemple 2.2.4 *Le type sémantique $\dagger t \rightarrow e \rightarrow t$ est un élément de $\mathcal{T}$. Dans le cadre de la sémantique intentionnelle de Montague, il peut être utilisé pour typer la représentation sémantique des verbes de croyance (e.g.,* 'croire'*,* 'penser'*...).*

Le λ-calcul utilisé pour représenter la sémantique des langues naturelles est un λ-calcul simplement typé supportant un ensemble d'opérateurs intentionnels à savoir les constructeurs ($^\wedge$, $^\cap$) et les destructeurs ($^\vee$, $^\cup$). La définition formelle de ce λ-calcul est spécifiée ci-après :

Définition 2.2.8 *Soient Σ un ensemble fini de constantes sémantiques, f_Σ une fonction de typage dans Σ et $\mathcal{V}$ un ensemble infini dénombrable de variables sémantiques typées. L'ensemble $\Lambda(\Sigma)$ de λ-termes bien typés est inductivement défini comme suivant :*

1. *$\forall\, u \in \Sigma$, $u \in \Lambda(\Sigma)$ et u est de type $f_\Sigma(u)$.*
2. *$\forall\, (x : A) \in \mathcal{V}$, $x \in \Lambda(\Sigma)$ et x est de type A.*
3. *$\forall\, u\ v \in \Lambda(\Sigma)$, si u est de type $A \rightarrow B$ et v est de type A alors $(u\ v) \in \Lambda(\Sigma)$ et il est de type B.*
4. *$\forall\, u \in \Lambda(\Sigma)$, si u est de type B et x une variable de type A alors $(\lambda x.\ u) \in \Lambda(\Sigma)$ et ce terme est de type $A \rightarrow B$.*
5. *$\forall\, u\ v \in \Lambda(\Sigma)$, si u est de type A et v est de type B alors $\langle u,v \rangle \in \Lambda(\Sigma)$ et il est de type $A \times B$.*
6. *$\forall\, u \in \Lambda(\Sigma)$, si u est de type $A \times B$ alors $\Pi_1(u) \in \Lambda(\Sigma)$ (resp. $\Pi_2(u) \in \Lambda(\Sigma)$) et il est de type A (resp. de type B).*
7. *$\forall\, u \in \Lambda(\Sigma)$, si u est de type A alors $^\wedge u$, $^\cap u \in \Lambda(\Sigma)$ et ils sont tous les deux de type $\dagger A$.*
8. *$\forall\, u \in \Lambda(\Sigma)$, si u est de type $\dagger A$ alors $^\vee u$, $^\cup u \in \Lambda(\Sigma)$ et ils sont de type A.*

Notation 2.2.2 *Dans un but de lisibilité, l'application d'un terme f à un argument a (resp. à une suite de n arguments a_1,...,a_n) sera généralement notée $f(a)$ (resp. $f(a_1, ..., a_n)$) à la place de $(f\ a)$ (resp. $(f\ a_1 \ldots a_n)$).*

Exemple 2.2.5 *La sémantique du verbe de croyance* 'sait' *peut être encodée par le λ-terme $\lambda x.\ \lambda y.$* **Savoir**$(y,\ x)$ *portant le type $\dagger t \rightarrow e \rightarrow t$. Ceci permettra, à titre d'exemple, d'assigner à la phrase intentionnelle* 'Marie sait que Jean admire Godel' *une formule logique telle que* **Savoir**(**Marie**, $^\wedge$**Admirer**(**Jean**, **Godel**)) *[47, 83].*

Définition 2.2.9 *Le λ-calcul défini en 2.2.8 est soumis aux règles de réduction classiques (e.g., β-réduction, Π-réduction, ...) [26, 47, 43], auxquelles on rajoute les deux règles suivantes qui expriment la dualité entre les constructeurs et destructeurs intentionnels :*

[6] Dans la sémantique de Montague, le type $\dagger(ty)$ est noté $s \rightarrow ty$.

$$(^{\vee}{}^{\wedge}u) \rightsquigarrow u \qquad (^{\cup}{}^{\cap}u) \rightsquigarrow u$$

Notation 2.2.3 *Soient u et v deux λ-termes et x une variable (dont le type est celui de v). u*[*x* := *v*] *dénote le terme obtenu en substituant les occurences libres de la variable x dans u par le terme v.*

L'homomorphisme ()* établissant le lien entre les types syntaxiques et les types sémantiques est étendu de la manière suivante :

Définition 2.2.10 *Soit* $()_{at}{}^*$ *une fonction assignant à chaque type syntaxique atomique un type sémantique approprié. L'homomorphisme* ()* *de type* $\mathcal{F} \rightarrow \mathcal{T}$ *est récursivement défini comme suit :*

1. $\forall\, a \in \mathcal{A}$, $(a)^* = (a)_{at}{}^*$.
2. $\forall\, A\ B$, $(A/_i B)^* = (B\backslash_i A)^* = (B)^* \rightarrow (A)^*$.
3. $\forall\, A\ B$, $(A \bullet_i B)^* = (A)^* \times (B)^*$.
4. $\forall\, A$, $(\Diamond_j A)^* = (\Box_j A)^* = \dagger(A^*)$.

A la lumière de ce qui précéde, la définition rigoureuse d'un lexique pour les grammaires multimodales peut être formulée comme suit :

Définition 2.2.11 *Un lexique à deux dimensions est défini comme étant un n-uplets* $<\mathcal{A}, \mathcal{I}, \mathcal{J}, \mathcal{W}, \Sigma, f_\Sigma, ()_{at}{}^*,$ lex$>$ *où :*

- $\mathcal{A}$ *est un ensemble fini de catégories atomiques.*
- $\mathcal{I}$ *est un ensemble fini de modes binaires.*
- $\mathcal{J}$ *est un ensemble fini de modes unaires.*
- $\mathcal{W}$ *est un ensemble fini de mots de la langue choisie.*
- Σ *est un ensemble fini de constantes sémantiques.*
- f_Σ *est une fonction de typage dans* Σ.
- $()_{at}{}^*$ *est une fonction associant à chaque atome de* $\mathcal{A}$ *un type sémantique adéquat.*
- lex *est une fonction d'assignation qui associe à chaque mot* m_i *de* $\mathcal{W}$ *un ensemble fini de paires* (F_i, t_i) *où* $F_i \in \mathcal{F}$, $t_i \in \Lambda(\Sigma)$ *et* t_i *est de type* $F_i{}^*$.

Remarque 2.2.1 *Pour traiter des exemples linguistiques concrets (cf. section 2.3), nous nous contenterons généralement de spécifier la dernière composante du lexique à savoir la fonction* lex.

Le lien entre la syntaxe et la sémantique est assuré grâce à la correspondance de Curry-Howard qui traduit chaque étape de déduction par un pas calculatoire dans le λ-calcul simplement typé [77]. Cet algorithme compositionnel de traduction est explicité par les règles de la figure 7. Notons que ces règles font intervenir des séquents *annotés* de la forme $\Gamma \vdash A \triangleright u$ où u est un λ-terme de type A^* encapsulant la sémantique dérivationnelle du séquent considéré. Précisons encore une fois que pour chaque variable syntaxique typée $(x : C)$ utilisée dans la dérivation (i.e., soit une ressource linguistique ou hypothétique), on associe une variable sémantique x^* de type C^* qui contribuera à la construction de la sémantique dérivationnelle.

La sémantique dérivationnelle d'un séquent $\Gamma \vdash C$ prend la forme d'un terme u contenant des variables sémantiques libres $m_1{}^*, m_2{}^*, ..., m_n{}^*$ associées aux différentes ressources $m_1, ..., m_n$ du contexte Γ. Exprimé autrement, le terme u peut être vu comme un programme qui prend n paramètres $m_1{}^*, m_2{}^*, ..., m_n{}^*$ et construit un λ-terme de type C^*. La sémantique finale de l'expression

$$\frac{}{x : A \vdash A \triangleright x^*}\ Ax \qquad \frac{\Delta \vdash A \bullet_i B \triangleright u \quad \Gamma[(x : A, y : B)^i] \vdash C \triangleright v}{\Gamma[\Delta] \vdash C \triangleright v[x^* := \Pi_1(u), y^* := \Pi_2(u)]}\ \bullet_i E \qquad \frac{\Gamma \vdash A/_i B \triangleright f \quad \Delta \vdash B \triangleright b}{(\Gamma, \Delta)^i \vdash A \triangleright (f\ b)}\ /_i E$$

$$\frac{\Gamma \vdash B \triangleright b \quad \Delta \vdash B \backslash_i A \triangleright f}{(\Gamma, \Delta)^i \vdash A \triangleright (f\ b)}\ \backslash_i E \qquad \frac{\Gamma \vdash A \triangleright u \quad \Delta \vdash B \triangleright v}{(\Gamma, \Delta)^i \vdash A \bullet_i B \triangleright < u, v >}\ \bullet_i I \qquad \frac{(\Gamma, x : B)^i \vdash A \triangleright f}{\Gamma \vdash A/_i B \triangleright \lambda x.\ f}\ /_i I$$

$$\frac{(x : B, \Gamma)^i \vdash A \triangleright f}{\Gamma \vdash B \backslash_i A \triangleright \lambda x.\ f}\ \backslash_i I \qquad \frac{\Delta \vdash \Diamond_j A \triangleright u \quad \Gamma[< x : A >_j] \vdash C \triangleright v}{\Gamma[\Delta] \vdash C \triangleright v[x^* := {}^\vee u]}\ \Diamond_i E \qquad \frac{\Gamma \vdash \Box_j A \triangleright u}{< \Gamma >_j \vdash A \triangleright {}^\cup u}\ \Box_j E$$

$$\frac{< \Gamma >_j \vdash A \triangleright u}{\Gamma \vdash \Box_j A \triangleright {}^\cap u}\ \Box_i I \qquad \frac{\Gamma \vdash A \triangleright u}{< \Gamma >_j \vdash \Diamond_j A \triangleright {}^\wedge u}\ \Diamond_j I \qquad \frac{\Delta_1 \xrightarrow{R} \Delta_2 \quad \Gamma[\Delta_2] \vdash C \triangleright u}{\Gamma[\Delta_1] \vdash C \triangleright u}\ S_R$$

FIG. 7: Interface syntaxe/sémantique en logique multimodale

analysée est ainsi obtenue en substituant chaque paramètre m_i^* par la sémantique lexicale du mot m_i, cette substitution est décrite formellement comme suit :

Définition 2.2.12 *Soit* $\Gamma \vdash C$ *un séquent où* $\Gamma \in Struct((m_1 : A_1), ..., (m_n : A_n))$ *et soit u sa sémantique dérivationnelle. Les différentes représentations sémantiques de l'expression analysée (i.e.,* $m_1,...,m_n$*) sont obtenues par l'intermédiaire de la substitution suivante :*

$$u[m_1^* := v_1, m_2^* := v_2, ..., m_n^* := v_n]$$

où pour tout i allant de 1 à n, la paire (A_i, v_i) *fait partie de* $lex(m_i)$*.*

2.2.5 Propriétés calculatoires

Avant de présenter certaines propriétés calculatoires qui caractérisent les grammaires catégorielles multimodales et leurs langages engendrés, nous proposons d'abord de définir formellement ces deux notions :

Définition 2.2.13 *Une grammaire catégorielle multimodale (**MMCG** en abrégé) est définie comme étant un triplet* $<\mathcal{LEX}, \mathcal{R}, at>$ *où :*

- $\mathcal{LEX}$ *est un lexique à deux dimensions, obéissant à la syntaxe de la définition 2.2.11.*
- $\mathcal{R}$ *est un paquetage de règles structurelles sous la forme faible de Sahlqvist (cf. définition 2.2.6).*
- *at est un atome distingué.*

Définition 2.2.14 *Soit* $\mathcal{G}=<\mathcal{LEX}, \mathcal{R}, at>$ *une grammaire catégorielle multimodale. Le langage engendré par* $\mathcal{G}$ *est l'ensemble de suites de mots* $m_1...m_n$ *vérifiant ce qui suit :*

$$\forall\, i \in \{1..n\}\ \exists (F_i,\ t_i) \in lex(m_i)\ \exists \Gamma \in Struct((m_1 : F_1), ..., (m_n : F_n)) \mid \Gamma \overset{\mathcal{R}}{\vdash} at$$

où $\overset{\mathcal{R}}{\vdash}$ *représente l'opérateur de dérivabilité en logique multimodale enrichie par le paquetage de règles structurelles* $\mathcal{R}$*.*

Capacité générative

La capacité générative faible des grammaires catégorielles multimodales (i.e., ensemble des chaines engendrées) est intimement liée à la nature des règles structurelles utilisées. Le langage engendré de telles grammaires peut passer d'un langage hors contexte à un langage récursivement énumérable suivant la logique sous-structurelle considérée. En effet, Carpenter a prouvé qu'en absence de toute restriction sur la forme des règles structurelles adoptées, les grammaires catégorielles multimodales générent exactement tous les langages reconnus par une machine de Turing (i.e., les langages récursivement énumérables) [25]. Cependant, le pouvoir expressif de ces grammaires peut être limité en se focalisant sur des classes particulières de règles structurelles. A titre d'exemple, Moot a démontré que les systèmes basés sur les règles structurelles linéaires sans-extension[7] (i.e., qui n'augmentent pas le nombre des opérateurs structurels unaires) engendrent uniquement des langages contextuels [81]. Un tel pouvoir expressif s'avère largement suffisant pour englober le langage humain.
Le schéma de la figure 8 récapitule ces différents résultats.

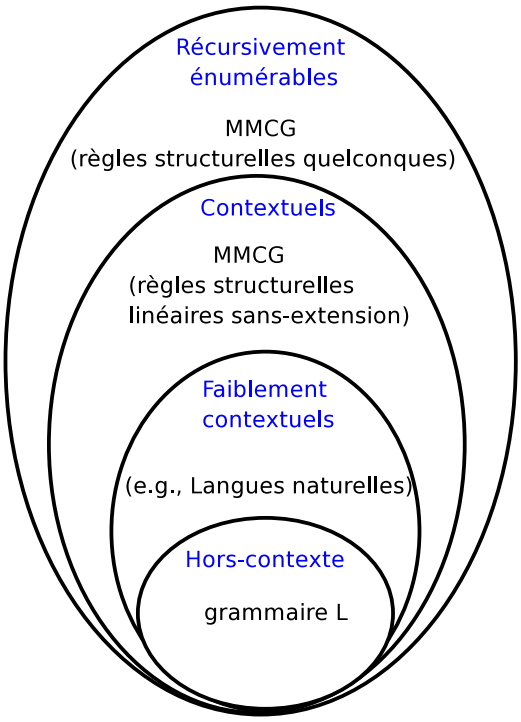


FIG. 8: Capacité générative faible de quelques classes de MMCG

Décidabilité & Complexité

Carpenter a prouvé que le problème de recherche de démonstration en logique multimodale est indécidable si on tolére la présence de règles structurelles non-linéaires qui dupliquent ou effacent certaines ressources [25]. Moot a étendu ce résultat en montrant que ladite logique reste

[7] Non-expanding rules.

indécidable même si on se restreint aux règles structurelles linéaires [81]. Toutefois, la décidabilité a lieu en se limitant à la classe des règles structurelles linéaires sans extension. Dans ce genre de grammaires, Moot a établi que le problème de décision (de la dérivabilité d'un séquent) est équivalent au problème d'analyse dans les grammaires contextuelles et fait ainsi partie de la classe de problèmes *PSPACE-complets* [81].

2.3 La logique Multimodale au service de la Linguistique

Afin de mieux appréhender le fonctionnement des grammaires multimodales, nous présentons, dans cette section, des exemples concrets d'analyse d'énoncés. Les exemples choisis illustrent deux conséquences positives résultant de l'enrichissement apporté au calcul de Lambek. D'une part, le traitement de l'extraction médiane souligne la flexibilité introduite grâce au raisonnement structurel. D'autre part, l'analyse du phénomène d'annexation en Arabe Standard met en évidence la puissance du contrôle établi par les opérateurs unaires permettant ainsi de limiter le problème de sur-génération qui handicapait le calcul de Lambek.

2.3.1 Extraction médiane

L'énoncé qu'on désire analyser dans cette partie est la relative suivante : '*que tout sage applique _ constamment*'. Linguistiquement, cette subordonnée présente deux phénomènes intéressants à savoir les dépendances non bornées et l'extraction médiane [77, 26]. Le premier phénomène émane de l'existence de dépendances sémantiques entre deux éléments éloignés d'une phrase. Dans notre cas il s'agit de l'objet extrait du verbe '*applique*' occupant le site _ et le pronom relatif '*que*'. Le second phénomène est concrétisé par l'extraction de l'objet du verbe '*applique*' dont la position n'est pas périphérique étant donné qu'il est encadré par ledit verbe à sa gauche et l'adverbe '*constamment*' à sa droite. Alors que le premier phénomène est aisément traité dans la logique de Lambek -grâce au raisonnement hypothétique- [26], le second n'est pas pris en charge par cette logique puisque seules les hypothèses périphériques peuvent être déchargées. Nous montrerons dans ce qui suit comment décrire ce dernier phénomène en utilisant la flexibilité structurelle locale des grammaires multimodales. Nous nous focaliserons uniquement sur l'analyse syntaxique de l'énoncé considéré sachant que le calcul de sa représentation sémantique se fait de manière systématique grâce à la correspondance de Curry-Howard (cf. section 2.2.4).
Notre analyse se fera dans un système multimodal qui manipule deux modes de composition a et c et qui comporte les deux postulats d'interaction $L\Diamond(a,c)$ et $P\Diamond(a,c)$ présentés dans la figure 9.

$$\frac{\Gamma[(\Delta_1,(\Delta_2,<\Delta_3>_j)^i)^i]\vdash C}{\Gamma[((\Delta_1,\Delta_2)^i,<\Delta_3>_j)^i]\vdash C}\ L\Diamond(i,\ j) \qquad \frac{\Gamma[((\Delta_1,<\Delta_3>_j)^i,\Delta_2)^i]\vdash C}{\Gamma[((\Delta_1,\Delta_2)^i,<\Delta_3>_j)^i]\vdash C}\ P\Diamond(i,\ j)$$

FIG. 9: Règles structurelles intervenant dans l'analyse de l'extraction médiane

Rappelons qu'une règle structurelle telle que $P\Diamond(i,\ j)$ peut être vue comme une règle de réécriture permettant de transformer n'importe quelle structure de la forme $((\Delta_1,\Delta_2)^i,<\Delta_3>_j)^i$ en $((\Delta_1,<\Delta_3>_j)^i,\Delta_2)^i$. Cette dernière règle augmente la flexibilité du système puisqu'elle autorise un

accès restreint à la commutativité de ressources.
Le lexique considéré est une forme enrichie du lexique de la grammaire de Lambek vu dans l'exemple 2.1.2.

que	tout	sage	applique	constamment
$(n\backslash_a n)/_a(s/_a\diamond_c\square_c np)$	$(s/_a(np\backslash_a s))/_a n$	n	$(np\backslash_a s)/_a np$	$(np\backslash_a s)\backslash_a(np\backslash_a s)$

La modification majeure apportée au lexique concerne le type syntaxique du pronom '*que*' qui considère l'hypothèse fictive occupant la position objet de la clause relative comme un syntagme nominal spécial (de type $\diamond_c\square_c np$). Ladite hypothèse a accès à une forme restreinte de commutativité grâce à la règle $P\diamond(a,\ c)$; elle peut donc se déplacer assez aisément de son site périphérique de départ à son site médian final. La figure 10 montre les étapes principales de l'analyse syntaxique de la subordonnée en question.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{que \vdash ...}\ Ax \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{applique \vdash (np\backslash_a s)/_a np}\ Ax \quad \dfrac{\dfrac{}{[x : \square_c np \vdash \square_c np]^1}\ Ax}{< x : \square_c np >_c \vdash np}\ \square_c E}{(applique, < x : \square_c np >_c)^a \vdash np\backslash_a s}\ /_a E}{((applique, < x : \square_c np >_c)^a, constamment)^a \vdash np\backslash_a s}\ /_a E}{(tout, sage)^a \vdash s/_a(np\backslash_a s)}\ /_a E}{((tout, sage)^a, ((applique, < x : \square_c np >_c)^a, constamment)^a)^a \vdash s}\ /_a E}{((tout, sage)^a, ((applique, constamment)^a, < x : \square_c np >_c)^a)^a \vdash s}\ P\diamond(a,\ c)}{(((tout, sage)^a, (applique, constamment)^a)^a, < x : \square_c np >_c)^a \vdash s}\ L\diamond(a,\ c)}{(((tout, sage)^a, (applique, constamment)^a)^a, y : \diamond_c\square_c np)^a \vdash s}\ \diamond_c E}{((tout, sage)^a, (applique, constamment)^a)^a \vdash s/_a\diamond_c\square_c np}\ /_a I, 1}{(que, ((tout, sage)^a, (applique, constamment)^a)^a)^a \vdash n\backslash_a n}\ /_a E$$

Fig. 10: Dérivation syntaxique d'une subordonnée relative

L'étape principale de l'analyse est celle qui permet de placer la ressource hypothétique y de type $\diamond_c\square_c np$ introduite par la règle $/_a I$ en position d'objet direct du verbe '*applique*'. Ceci se fait en appliquant les règles structurelles présentées dans la figure 6 qui assurent la restructuration du contexte. L'application de ces règles s'avère possible étant donné que le type de l'objet extrait est décoré par l'opérateur unaire $\diamond_c$ qui peut passer à sa forme structurelle $<>_c$ grâce à la règle de base $\diamond_c E$.

2.3.2 Phénomène d'annexation en Arabe Standard

Quelques mots sur l'Arabe Standard

Nous présenterons tout d'abord certaines caractéristiques capitales de l'Arabe Standard (**AS**) qui sont susceptibles de faciliter la compréhension de l'analyse qui suit. Plus de détails concernant la grammaire de la langue Arabe peuvent être trouvés dans les ouvrages [23, 99].
AS est une langue sémitique, elle est par conséquent fortement flexionnelle (i.e., dont les mots changent de forme suivant leur rapport grammatical avec les autres composantes la phrase). A titre d'exemple, les cas sont généralement marqués par des suffixes (e.g., *-u* : cas nominatif, *-i* : cas génitif). En outre, l'indicateur de définitude en **AS** est incorporé à l'intérieur des noms et ce à la différence des langues indo-européennes (e.g., anglais, français) qui utilisent des déterminants

(e.g., *le*, *un*, *the*, *a*) pour construire des groupes nominaux définis ou indéfinis. Ainsi, le préfixe *al-*[8] sert à former des noms définis tandis que le suffixe *-n* marque les noms indéfinis. Par exemple, *kitāb-u-n* (i.e., *un livre*-[nom]) est un nom nominatif indéfini alors que *al-kitāb-i* (i.e., le livre-[gen]) représente un nom génitif défini.
En **AS**, il est possible de construire des phrases qui ne contiennent aucun verbe étant donné l'absence des verbes copulatifs (e.g., *être*, *devenir*, ...) dans cette langue. Ces phrases dites *nominales* sont composées de deux constituants à savoir un *primat* qui prend la forme d'un syntagme nominal nominatif en général défini et un *prédicat* qui peut être soit un groupe nominal nominatif indéfini, un adjectif indéfini ou une phrase propositionnelle.
Finalement, précisons que l'écriture en **AS** est basée sur un alphabet spécifique, et sa direction est de droite à gauche, toutefois, dans un but de lisibilité, nous utiliserons plutôt la transcription proposée par le paquetage *arabtex*, disponible à l'adresse suivante : www.informatik.uni-stuttgart.de/ifi/bs/research/arab_e.html.

Analyse du phénomène d'annexation dans MMCG

En Arabe Standard, on peut former des groupes nominaux composés moyennant le phénomène d'annexation [23]. Ces noms composés ont la forme suivante '$cn = n_1\ n_2\ ...\ n_k$' ($k \geq 2$), où chaque n_j ($1 \leq j < k$) est un nom en *état d'annexation* (i.e., ne portant ni l'indicateur défini ni l'indicateur indéfini), tandis que n_k est un syntagme nominal simple (défini ou indéfini). Le groupe nominal composé *cn* hérite le trait de définitude (défini ou indéfini) du nom n_k, tandis que son cas est celui de n_1 (tous les autres noms n_j ($j \geq 2$) portent le cas génitif). Nous présentons ci-après quelques exemples de groupes nominaux composés formés par annexation :

(4) *ibn-u*/**al-ibn-u* *'l-mudarris-i*
fils-[nom]/*le fils-[nom] l'instituteur-[gen]
'le fils de l'instituteur'

(5) *kitāb-u*/**kitāb-u-n* *mudarris-i-n*
livre-[nom]/*le livre-[nom] un instituteur-[gen]
'le livre d'un instituteur'

(6) *kitāb-u* *ibn-i* *'l-mudarris-i*
livre-[nom] fils-[gen] l'instituteur-[gen]
'le livre du fils de l'instituteur'

Dans la présente analyse, nous manipulons une grammaire multimodale constituée d'un seul mode binaire rigide noté 0 (i.e., $\mathcal{I}=\{0\}$) qui est à la fois non-associatif et non-commutatif, et d'un ensemble de modes unaires encodant différents traits morphosyntaxiques tels que les cas (e.g., *nom* : nominatif, *gen* : génitif) et les traits de définitude (e.g., *def* : défini, *ind* : indéfini). Afin de rendre compte du phénomène d'annexation, nous assignons d'abord un type syntaxique adéquat à chacune des trois classes de noms simples de **AS**, à savoir les *al-noms* (noms simples définis avec le préfixe *al-*), les *ea-noms* (noms en état d'annexation) et les *nn-nouns* (noms simples indéfinis par le suffixe *-n*).

[8] Le préfixe *al-* est abrégé en *'l-* quand le nom en question est précédé par un mot se terminant par une voyelle.

al-noms	nn-noms	ea-noms
$\Box_{al}\Box_{cas}$np	$\Box_{nn}\ \Box_{cas}$np	$\Box_{ea}(\ \Box_{cas}$np $/_0\ \Box_{gen}$ np)

Remarquons que les *al-noms* et les *nn-noms* sont complets et auto-dépendants ; ils peuvent apparaitre dans des contextes variés et occuper différentes fonctions grammaticales (i.e., sujets, primats). Néanmoins, les *ea-noms* sont incomplets ; ils sont uniquement utilisés dans le but de construire des syntagmes nominaux composés. Leur type syntaxique composé reflète leur besoin de se combiner, à leur droite, avec un groupe nominal dans le but de construire une expression complète.
La grammaire que nous considérons comprend le paquetage de règles structurelles suivant : $\mathcal{R}$= K(*def*) ∪ K(*ind*) ∪ Inc(*def, al*) ∪ Inc(*ind, nn*), où les règles $K(j)$ et $Inc(i,\ j)$ sont définies dans la figure 11.

$$\frac{\Gamma[(<\Delta_1>_{ea}, <\Delta_2>_j)^0] \vdash C}{\Gamma[<(\Delta_1,\Delta_2)^0>_j] \vdash C}\ K(j) \qquad \frac{\Gamma[<\Delta>_j] \vdash C}{\Gamma[<\Delta>_i] \vdash C}\ Inc(i,\ j)$$

Fig. 11: Règles structurelles dédiées à l'analyse du phénomène d'annexation

La règle structurelle *K(def)* (resp. *K(ind)*) exprime une forme de distributivité forte de l'opérateur structurel $<>$ [56]. Intuitivement, ce postulat stipule qu'un constituant composé est défini (resp. indéfini) si sa tête est en état d'annexation (e.g., un *ea-nom*) et son complément est défini (resp. indéfini). D'autre part, la règle *Inc(def, al)* (resp. *Inc(ind, nn)*) n'est autre qu'un principe d'inclusion qui spécifie que tous les *al-noms* (resp. *nn-noms*) sont forcément définis (resp. indéfinis).
Grâce au paquetage $\mathcal{R}$, seuls les noms composés bien formés '$n_1 ... n_k$' peuvent être dérivés, ils portent soit le type $\Box_{def}\Box_{c_1} np$ si n_k est un *al-nom* ou le type $\Box_{ind}\Box_{c_1} np$ si n_k est un *nn-nom* (c_1 étant le cas de n_1). En effet, le paquetage $\mathcal{R}$ permet l'application d'une stratégie relativement puissante dite *serrure/clef* (i.e., lock/key), cette dernière est largement utilisée dans la littérature notamment pour rendre compte de phénomènes complexes tels que les dépendances croisées en néerlandais [80]. Dans le présent cas, la technique serrure/clef procède comme suivant. Dans un premier temps, les règles récursives $K(j)$ sont appliquées dans le but d'ouvrir la serrure $\Box_{ea}$ enveloppant chacun des noms respectifs n_i ($1 \leq i < k$) (la clef $<>_{ea}$ est capable d'ouvrir la serrure $\Box_{ea}$ car $<\Box_{ea}A>_{ea}A \vdash A$). Notons que cette étape ne réussit que si tous ces noms sont en état d'annexation, puisque le séquent $<\Box_x A>_{ea} A \vdash A$ n'est dérivable que dans le cas où x=*ea*. Les règles $Inc(i,\ j)$ sont ensuite utilisées pour vérifier le trait de définitude du nom n_k. Finalement, la dérivation s'achève par une succession d'éliminations du connecteur $/_0$. Il s'avère par conséquent facile d'analyser une phrase nominale enrichie (dont le primat est un nom composé) telle que (7) :

(7) *ibn-u* *'l-mudarris-i* *kabīr-u-n*
fils-[nom] l'instituteur grand
'le fils de l'instituteur est grand'

Les étapes cruciales de la dérivation du primat de cette phrase sont récapitulées ci-dessous :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) \vdash \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np)}\,Ax}{< ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) >_{ea} \vdash \Box_{nom}np/_0\Box_{gen}np}\,\Box_{ea}E \quad \dfrac{\dfrac{}{'lmudarrisi : \Box_{al}\Box_{gen}np \vdash \Box_{al}\Box_{gen}np}\,Ax}{<'lmudarrisi : \Box_{al}\Box_{gen}np >_{al} \vdash \Box_{gen}np}\,\Box_{al}E}{(< ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) >_{ea} , <'lmudarrisi : \Box_{al}\Box_{gen}np >_{al})^0 \vdash \Box_{nom}np}\,/_0E}\,Inc(def,al)}{(< ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) >_{ea} , <'lmudarrisi : \Box_{al}\Box_{gen}np >_{def})^0 \vdash \Box_{nom}np}\,K(def)}{< (ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) , 'lmudarrisi : \Box_{al}\Box_{gen}np)^0 >_{def} \vdash \Box_{nom}np}\,\Box_{def}I}{(ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) , 'lmudarrisi : \Box_{al}\Box_{gen}np)^0 \vdash \Box_{def}\Box_{nom}np}$$

Les différents mécanismes de contrôle employés par la stratégie serrure/clef garantissent l'exclusion des phrases nominales agrammaticales dont le primat est un nom composé non-défini comme c'est le cas pour (8) :

(8) **kitāb-u* *mudarris-i-n* *mufīd-u-n*
le livre-[nom] un instituteur intéressant

En associant au prédicat *mufīd-u-n* le type $\Box_{def}\Box_{nom}np\backslash_0 s$, la dérivation de la phrase mal-formée (8) est systématiquement bloquée suite à l'échec de vérification du trait défini du primat composé. L'analyse partielle qui suit détermine la raison sous-jacente à ce blocage salutaire, à savoir l'absence de la règle structurelle *Inc(def, nn)* du système :

$$\dfrac{\dfrac{\dfrac{\dfrac{(*)}{(< ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) >_{ea} , < mudarrisin : \Box_{nn}\Box_{gen}np >_{nn})^0 \vdash \Box_{nom}np}\,(*)Inc(def,nn)}{(< ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) >_{ea} , < mudarrisin : \Box_{nn}\Box_{gen}np >_{def})^0 \vdash \Box_{nom}np}\,K(def)}{< (ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) , mudarrisin : \Box_{nn}\Box_{gen}np)^0 >_{def} \vdash \Box_{nom}np}\,\Box_{def}I}{(ibnu : \Box_{ea}(\Box_{nom}np/_0\Box_{gen}np) , mudarrisin : \Box_{nn}\Box_{gen}np)^0 \vdash \Box_{def}\Box_{nom}np}$$

Pour calculer les représentations sémantiques des énoncés analysés, nous utilisons des λ-termes d'ordre supérieur spécifiant la sémantique lexicale de chacune des classes de noms de **AS**. Dans le but de conserver la correspondance entre la syntaxe et la sémantique, nous procédons à une montée du type syntaxique *np* (i.e., on utilise le type d'ordre supérieur $(s/_0np)\backslash_0 s$ à la place du type primitif *np*). Ainsi à titre d'exemple, les noms singuliers de **AS** se voient attribuer les sémantiques lexicales respectives ci-après :

ea-nom : m_{ea} (**Rel**)	$\lambda P_{(e\to t)\to t}\ \lambda Q_{e\to t}.\ \exists$ x. $\mathbf{m}_{pred}$(x) $\wedge$ Q(x) $\wedge$ P(λy. **Rel**(x,y) $\wedge\ \forall$ z. $\mathbf{m}_{pred}$(z) $\wedge$ **Rel**(z,y) $\Rightarrow$ z=x)
al-nom : m_{al}	$\lambda Q_{e\to t}.\ \exists$ x. $\mathbf{m}_{pred}$(x) $\wedge$ Q(x) $\wedge$ ($\forall$ z. $\mathbf{m}_{pred}$(z) $\Rightarrow$ z=x)
nn-nom : m_n	$\lambda Q_{e\to t}.\ \exists$ x. $\mathbf{m}_{pred}$(x) $\wedge$ Q(x)

Précisons que la semantique de chaque nom m_x est basée sur un prédicat $\mathbf{m}_{pred}$ représentant l'ensemble des individus qui partagent une propriété spécifique (e.g., **Insituteur**, **Fils**...). En outre, le sens des *ea-noms* est étroitement lié à une relation **Rel** qui lie ces individus à leur complément (e.g., **Rel** peut être une relation de possession, de parenté ...). Finalement, c'est intéressant de remarquer que la sémantique des *ea-noms* ainsi que celle des *al-noms* établit une certaine condition d'unicité. Par exemple, le sens d'un *ea-nom* m_{ea} stipule que l'intersection entre l'ensemble des individus vérifiant la propriété $\mathbf{m}_{pred}$ et la collection des entités liées au complément de m_{ea} par la relation **Rel** n'est autre qu'un singleton.

En se basant sur la sémantique lexicale précédente, on peut aisément prouver que le sens de la phrase (7) est représenté par la formule de premier ordre suivante :

$\exists x.\ \mathbf{Fils}(x) \wedge \mathbf{Grand}(x) \wedge \exists y.\ \mathbf{Instituteur}(y) \wedge \mathbf{Parenté}(x,y) \wedge$
$(\forall\ z.\ \mathbf{Fils}(z) \wedge \mathbf{Parenté}(z,y) \Rightarrow z{=}x) \wedge (\forall z.\ \mathbf{Instituteur}(z) \Rightarrow z{=}y)$

Notre étude peut être également appliquée pour rendre compte de la formation des noms composés en hébreu. En effet, le phénomène d'annexation est géré par le même ensemble de principes syntaxiques aussi bien en hébreu qu'en **AS**, comme l'illustre l'exemple suivant issu de [108] :

(9) *yaldei* *mnahhel* *taxnot* *ha-rakkebt*
enfants-[ea] directeur-[ea] stations-[ea] le train
'les enfants du directeur des stations de train'

2.4 Conclusion

Dans ce chapitre, nous avons présenté les composantes et les caractéristiques fondamentales des grammaires catégorielles multimodales qui sont basées sur une extension du calcul pur de Lambek. Grâce au raisonnement structurel, ces grammaires enrichies permettent le réagencement de ressources de manière contrôlée limitant ainsi les problèmes de sur-génération et sous-génération dont souffrait la logique de base.
Toutefois, la pertinence linguistique des grammaires multimodales vient au détriment de leur simplicité : en effet, la manipulation de ce formalisme s'avère ardue notamment pour les novices. L'outil interactif $\mathcal{ICHARATE}$, décrit en détail dans le chapitre suivant, a pour objectif de faciliter la compréhension de ces grammaires ainsi que l'étude de leurs propriétés génériques.

ICHARATE : une bibliothèque Coq dédiée aux Grammaires Multimodales

Grâce à leur richesse, leur grand degré de paramétrabilité et leur capacité générative raisonnable, les grammaires catégorielles multimodales s'avèrent pertinentes d'un point de vue linguistique. Néanmoins, l'adéquation de ce formalisme vient au détriment de sa simplicité et sa complexité d'analyse. En effet, l'utilisation et la manipulation de ce modèle linguistique sont relativement ardues notamment pour les novices.
D'autre part, le problème de recherche de démonstration dans la logique multimodale de Moortgat étant indécidable, on ne peut envisager de réaliser des analyseurs totalement automatiques qui prennent en charge toutes les classes de grammaires. Toutefois, il existe un nombre de prouveurs automatiques basés sur des classes particulières de logique multimodale où la recherche de preuves devient décidable. Parmi ces outils citons l'analyseur syntaxique **Grail** réalisé par Moot en prolog afin de prendre en charge la sous-classe de grammaires multimodales basées sur les règles structurelles linéaires sans-extension [81]. Cet analyseur est certes automatique, mais ne permet guère de raisonner sur le modèle sous-jacent en prouvant des propriétés universelles vérifiées par des classes entières de grammaires. En effet, on ne peut effectuer une dérivation dans **Grail** qu'une fois que tous les paramètres de la grammaire sont déterminés.
Il serait donc intéressant d'avoir un outil qui prenne en charge toutes les classes de grammaires multimodales sans aucune restriction, qui offre la possibilité d'en étudier les propriétés génériques et qui, de plus, permette aux utilisateurs d'effectuer leurs dérivations interactivement pour mieux appréhender le fonctionnement de la logique multimodale. L'atelier logique *ICHARATE*, basé sur le système d'aide à la démonstration **Coq** [72, 21], se veut une réponse à ces besoins.
L'atelier *ICHARATE* est un outil de recherche qui étend et enrichit notre contribution antérieure dédiée à la formalisation en **Coq** du calcul de Lambek [5, 11]. Notre objectif est de créer une bibliothèque **Coq** pour la logique multimodale qui permettra de faciliter la compréhension de cette dernière aux différents utilisateurs (chercheurs/étudiants, logiciens, ...).
Le noyau d'*ICHARATE* comprend la formalisation de deux présentations de la logique multimodale à savoir le calcul des séquents et la déduction naturelle. Ces systèmes possèdent différentes règles d'inférence mais engendrent le même langage [77]. Notre atelier contient aussi une bibliothèque de règles dérivées et théorèmes universels susceptibles d'être importés et réutilisés à tout moment. Ces bibliothèques permettent d'étudier les méta-propriétés de certaines familles de grammaires, elles peuvent être étendues par les utilisateurs souhaitant raisonner sur de nouvelles classes de règles structurelles. Finalement, *ICHARATE* est doté d'une interface utilisateur qui met en œuvre plusieurs techniques avancées (tactiques spécialisées, proof-by-pointing) afin de faciliter l'interaction avec les utilisateurs.

Dans ce qui suit, nous décrirons les composantes de base de cet atelier. Pour plus de lisibilité, les énoncés des théorèmes seront présentés en utilisant les notations mathématiques classiques. Toutefois, le lecteur intéressé par la formalisation en **Coq** trouvera les sources de l'atelier dans [12].

3.1 Présentation succincte de Coq

L'outil **Coq** est un assistant de preuves basé sur le Calcul des Constructions Inductives (CCI) [33, 90] qui est une variante de la logique intuitionniste d'ordre supérieur. Ce système permet, au-delà du codage des énoncés à démontrer, d'encoder intégralement les preuves (par des λ-termes) selon l'isomorphisme de Curry-Howard.
Coq est constitué de deux composantes principales : un langage de preuves et un vérificateur de types. Le premier constituant permet aux utilisateurs de construire leurs preuves interactivement grâce à l'utilisation de tactiques et procédures de décisions qui simplifient au fur et à mesure le but courant en appliquant une ou plusieurs étapes d'inférence. Quand la construction de la preuve est achevée, le vérificateur de type s'assure que cette dernière est correcte, ceci revient à vérifier que son type est convertible avec celui du but initial. Le lecteur intéressé trouvera un bref panorama du système **Coq** en annexes (cf. Annexe B).
L'assistant de preuves **Coq** a été choisi comme méta-langage pour formaliser différentes théories logiques et ce grâce à sa richesse de types (types inductifs, types dépendants). A titre d'exemple, citons la formalisation de la logique épistémique par Lescanne [71] et la logique linéaire temporelle par Coupet-Grimal [34]. L'outil $ICHARATE$ vient prolonger ces travaux en offrant une formalisation complète de la logique multimodale en **Coq**.

3.2 Représentation des Grammaires Multimodales en Coq

3.2.1 Plongement profond de la syntaxe de la logique multimodale

Le noyau de notre atelier est basé sur un plongement profond (*deep embedding*) de la syntaxe de la logique multimodale en **Coq**. Ce plongement est obtenu en représentant les différentes structures de données de la logique multimodale (types, contextes, règles structurelles, dérivations) par des types inductifs *paramétrés* (par les atomes, modes, règles structurelles ...). Une telle représentation est bénéfique dans la mesure où elle permet de considérer les différentes structures de données comme des objets de première classe. Il s'avère donc facile d'explorer la structure de ces objets grâce au filtrage (*pattern matching*). En outre, on est capable de raisonner récursivement sur ces derniers par l'intermédiaire des théorèmes de récurrence engendrés automatiquement par le système[1].
Nous décrirons dans ce qui suit la formalisation en **Coq** des composantes de base inhérentes à la syntaxe de la logique multimodale. Nous éviterons au maximum de rentrer dans les détails techniques de l'implémentation, les lecteurs intéressés par l'encodage en **Coq** des différentes composantes du noyau (spécifications, réalisations, ...) sont invités à consulter le répertoire `Kernel` de notre contribution [12].

[1] A la différence du plongement superficiel, i.e., *shallow embedding*, qui ne permet ni d'exploiter la structure syntaxique des objets considérés ni de prouver des propriétés logiques dessus.

Types syntaxiques et contextes

Les structures de données de base de la logique multimodale (types syntaxiques, ressources et contextes structurés) sont formalisés moyennant des types inductifs polymorphes qui sont paramétrés par des entités telles que les modes de composition I et J, l'ensemble des atomes A ou l'ensemble des mots W. A titre d'exemple, la définition du type `Form` associé aux formules syntaxiques se fait comme suit :

```
Section Main.
Variables I J (* modes de composition *)
          A (* types atomiques *)
          : Set.

Inductive Form : Set :=
| At : A -> Form
| Slash : I -> Form  -> Form  -> Form
| Backslash : I -> Form -> Form  -> Form
| Dot : I -> Form  -> Form  -> Form
| Box : J -> Form  -> Form
| Diamond : J -> Form  -> Form

End Main.
```

La définition inductive du type `Form` est basée sur cinq constructeurs. Ainsi, le constructeur `At` spécifie que si a est une formule atomique (i.e., un terme de type `A`), alors le terme " `At` a " est une formule syntaxique (i.e., un terme de type `Form`.) De la même façon, si i est un mode binaire, A et B des formules, alors le terme " `Slash` i A B " représente la formule qu'on note d'habitude $A/_i B$. Remarquons que cette définition est effectuée dans un environnement où les variables `I`, `J` et `A` sont déjà déclarées. En effet, le système **Coq** autorise la paramétrisation de manière élégante grâce à son mécanisme de sections [72, 21]. A la clôture de la section concernée, le type `Form` dépendra aussi bien des modes que des types atomiques. Par exemple, si `binaires`, `unaires` et `atomes` sont trois types de données, le type de formules syntaxiques correspondant ne sera autre que l'application " `Form binaires unaires atomes` ". La définition inductive des ressources linguistiques ainsi que celle des contextes structurés sont similaires à la définition des formules. Ainsi, les ressources linguistiques sont formées grâce à deux constructeurs *word* et *hyp* dont le rôle est de distinguer entre les ressources concrètes (i.e., les mots de W) et les ressources hypothétiques (i.e., utilisées lors d'un raisonnement hypothétique). La définition inductive des contextes est, quant à elle, basée sur trois constructeurs : le premier permettant de former une feuille à partir d'une ressource linguistique et d'un type syntaxique (i.e., $u : A$), le deuxième est associé à la construction binaire $(\ldots,\ldots)^j$ tandis que le dernier formalise l'opérateur structurel $< >_j$.

Comme nous l'avons vu au chapitre précédent, certaines règles de déduction telles que $\blacklozenge_i E$, $\Diamond_j E$ et **Cut** (cf. définitions 2.2.4 et 2.2.5) font appel à une décomposition de contextes sous la forme $\Gamma[\Delta]$. Dans notre implémentation, nous considérons une telle décomposition comme un remplissage d'un *contexte linéaire* $\Gamma[\,]$ (i.e., un contexte muni d'un trou, représenté ici par $[\,]$) par un contexte spécifique Δ. Les contextes linéaires sont représentés par des structures de données

temporaires efficaces à savoir les *zippers* [59]. Ces structures représentent des arbres dotés d'un pointeur sur l'un de leurs sous-arbres ; elles sont compatibles avec la programmation fonctionnelle et sont utilisées dans plusieurs bibliothèques entre autres dans le domaine de la linguistique computationnelle [60, 95]. Dans ce cadre, les zippers sont formalisés par un type inductif `zcontext` qui est muni d'une fonction de remplissage `zfill` définie récursivement comme suit :

```
(* zippers : contextes lineaires *)
Inductive zcontext : Set :=
   | zroot : zcontext
   | zleft  : I ->  zcontext -> context -> zcontext
   | zright : I -> context->  zcontext -> zcontext
   | zdown : J ->  zcontext -> zcontext.

(* remplissage d'un zipper par un contexte *)
Fixpoint zfill (z:zcontext)(Gamma : context) {struct z}
   : context :=
   match z with zroot => Gamma
              | zleft i z1 Gamma2 =>
                   zfill z1 (Comma i Gamma Gamma2)
              | zright i Gamma1 z2 =>
                   zfill z2 (Comma i Gamma1 Gamma)
              | zdown j z1 => zfill z1 (TDiamond j Gamma)
          end.
```

La figure 12 montre comment représenter un contexte linéaire sous la forme d'un zipper obtenu en mettant l'arbre de départ à l'envers à partir du trou [ ].

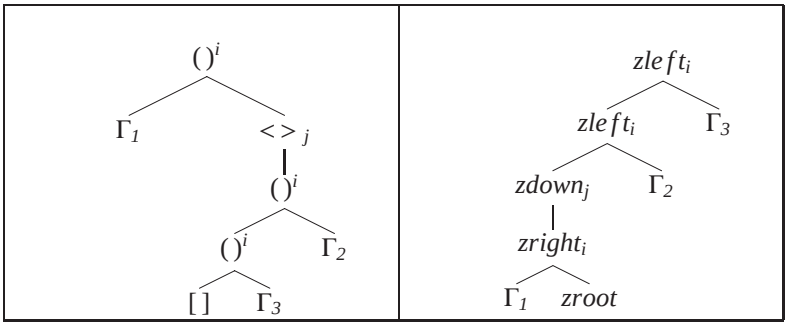


FIG. 12: Exemple de contexte linéaire et de son zipper correspondant

Règles structurelles

Les règles structurelles sont définies de manière abstraite comme des paires de contextes symboliques dont les feuilles sont des variables (encodées par des entiers en **Coq**). La première composante de chaque paire coïncide avec la structure globale du contexte à réécrire tandis que la

seconde n'est autre que la structure résultante de la réécriture. A titre d'exemple, la règle de réécriture associée à L◇(i, j) (cf. Fig 9, page 30) est représentée en **Coq** comme suit :

```
Definition L◇(I J:Set) (i:I)(j:J) : structrule I J:=
( ((1 ,i 2) ,i < 3 >j) , (1 ,i (2 ,i < 3 >j)) ).
```

La représentation des règles de réécriture par une structure de donnée est salutaire puisqu'elle permet de faciliter la définition et la vérification de différentes propriétés portant sur ces dernières (linéarité, linéarité à gauche, ...). A titre d'exemple, le test de linéarité à gauche s'effectue aisément en vérifiant que toutes les feuilles du premier contexte symbolique sont mutuellement distinctes et que l'ensemble des feuilles du second contexte est inclus dans l'ensemble des feuilles du premier.

L'interprétation de ces règles structurelles symboliques est définie grâce à la fonction partielle `apply-rule` qui assure leur application sur des contextes concrets (dont les feuilles sont des types étiquetés par des ressources linguistiques). Etant donné une règle de réécriture abstraite r=(rp_1, rp_2) et un contexte concret Γ_1, le comportement de la fonction `apply-rule` est synthétisé ci-après :

$$(\texttt{apply-rule}\ r\ \Gamma_1) = \begin{cases} Some(\Gamma_2) & \text{s'il existe une substitution } \sigma \text{ telle que } \Gamma_1 = \sigma(rp_1) \\ & \text{alors } \Gamma_2 = \sigma(rp_2) \\ None & \text{sinon (i.e., si } \Gamma_1 \text{ et } rp_1 \text{ n'ont pas la même structure)} \end{cases}$$

Afin de faciliter l'application des règles structurelles au cours des dérivations descendantes ou ascendantes, nous introduisons un certain nombre de lemmes portant sur la fonction d'interprétation `apply-rule`. Ces lemmes sont fort utiles car ils sont prouvés une seule fois et peuvent être utilisés autant de fois qu'on le souhaite, chose qui permet d'éviter le déroulement des calculs complexes effectués par la fonction `apply-rule`. L'extrait ci-dessous illustre deux catégories de lemmes introduits : les lemmes de réécritures (e.g., `L◇-rw`) qui sont appliqués dans les dérivations descendantes et les lemmes de réécriture inverse[2] (e.g., `L◇-inv`) utilisés au cours des dérivations ascendantes.

```
Lemma L◇-rw : ∀ I J  ... (Δ1 Δ2 Δ3 : context I J A W) i j,
          apply-rule L◇(i, j) ((Δ1 ,i Δ2) ,i <Δ3>j) =
          Some (Δ1 ,i (Δ2 ,i <Δ3>j)).
...

Lemma L◇-inv : ∀ I J  ... (Δ1 Δ2 Δ3 : context I J A W) i j,
           apply-rule L◇(i, j) Δ =
           Some (Δ1 ,i (Δ2 ,i <Δ3>j))->
           Δ = ((Δ1 ,i Δ2) ,i <Δ3>j).
...
```

Dérivations

Le noyau de l'atelier *ICHARATE* comprend la formalisation des deux styles de présentation de la logique multimodale : le calcul des séquents et la déduction naturelle. A titre

[2]Ce type de lemmes ne concerne pas les règles structurelles d'affaiblissement qui suppriment certaines ressources après réécriture.

d'exemple, le type `natded` associé aux déductions naturelles est un type inductif *dépendant* : le type (`natded` Γ `A`) est habité si et seulement si le séquent $\Gamma \vdash A$ est dérivable. Chaque constructeur de `natded` est associé à une règle de dérivation, par conséquent, le type inductif mis en œuvre comporte 12 constructeurs. Nous présentons ci-dessous un extrait de la définition de `natded` :

```
Section natDed.
...
Variables I J At W : Set. (* modes, atomes et mots *)
Variable R : extension I J. (* ensemble de regles structurelles *)

Inductive natded : context I J At W → Form I J At → Set:=
...
  | Ax     : ∀ x A,
             natded (x:A) A
  | SlashI : ∀ i x Γ A B,
             natded (Γ , (x:B))^i A →
             natded Γ  A/_i B
  | SlashE : ∀ i Γ Δ A B,
             natded Γ A/_i B →
             natded Δ B →
             natded (Γ , Δ )^i A
  | Struct : ∀ Γ_z Δ_1 Δ_2 C r,
             r ∈ R →
             (apply-rule r Δ_1) = Some Δ_2 →
             natded (zfill Γ_z Δ_2) C →
             natded (zfill Γ_z Δ_1) C.
```

Notons que l'utilisation d'un type dépendant est nécessaire pour exprimer les différentes contraintes exigées par les règles d'inférence. Par exemple, le constructeur `SlashI` prend comme paramètres un mode binaire `i`, une ressource hypothétique `x`, un contexte Γ, deux formules `A` et `B` et un terme d_1 de type (`natded (Γ, x :B)`^{i} `A`) et retourne un terme de type (`natded` Γ `A`/$_i$`B`). Si le terme d_1 avait un autre type (e.g., `natded` Γ `A`), l'application de ce constructeur serait impossible.

L'arbre de dérivation d'un séquent $\Gamma \overset{R}{\vdash} A$ est ainsi représenté par un terme construit en appliquant une séquence finie de constructeurs du type `natded`. La correspondance entre la présentation arborescente des étapes de déduction de la logique multimodale et leur encodage sous forme de λ-termes est illustrée par la figure 13 (les notations $/_I$, $/_E$ et $\bullet_E$ sont des abréviations des constructeurs `SlashI`, `SlashE` et `DotE`).

3.2.2 Approche mixte pour la sémantique

Pour représenter la sémantique des expressions syntaxiquement correctes, nous utilisons le λ-calcul simplement typé. A la différence des langues naturelles, le λ-calcul est un langage sans ambiguïté ; il est donc considéré comme le langage pivot permettant d'accéder à l'interprétation des phrases dans les modèles [47]. Chaque modèle est vu comme une représentation abstraite d'un

Nom	Règle	Terme associé
Ax	$\dfrac{}{x : A \overset{R}{\vdash} A}\ Ax$	$\mathbf{Ax}(x, A)$
$/_i$I	$\dfrac{\dfrac{d}{(\Gamma\,,\,(x : B))^i \overset{R}{\vdash} A}}{\Gamma \overset{R}{\vdash} A/_i B}\ /_i I$	$/_I(i, x, \Gamma, A, B, d)$
$/_i$E	$\dfrac{\dfrac{d_1}{\Gamma \overset{R}{\vdash} A/_i B} \quad \dfrac{d_2}{\Delta \overset{R}{\vdash} B}}{(\Gamma, \Delta)^i \overset{R}{\vdash} A}\ /_i E$	$/_E(i, \Gamma, \Delta, A, B, d_1, d_2)$
$\bullet_i$ E	$\dfrac{\dfrac{d_1}{\Delta \overset{R}{\vdash} A \bullet_i B} \quad \dfrac{d_2}{\Gamma[(x : A, y : B)^i] \overset{R}{\vdash} C}}{\Gamma[\Delta] \overset{R}{\vdash} C}\ \bullet_i E$	$\bullet_E(i, x, y, \Gamma[\,], \Delta, A, B, C, d_1, d_2)$
...	...	...

Fig. 13: Encodage des étapes de déduction sous forme de termes

monde possible encapsulant ainsi un ensemble de données et de situations, le but de cette interprétation est de déterminer la valeur de vérité de l'énoncé dans de telles circonstances.
L'atelier $\mathcal{ICHARATE}$ prend en charge le calcul de la sémantique de Montague. Notre module comprenant la formalisation du λ-calcul simplement typé en **Coq** s'inspire fortement de la contribution de Barras [19]. Contrairement à cette dernière, notre plongement profond du λ-calcul en **Coq** est défini par un type inductif polymorphe qui est paramétré par l'ensemble des constantes considérées. En outre, nous distinguons deux classes de variables : les variables liées encodées par des indices de De Bruijn et les variables localement libres qui représentent les ressources hypothétiques pouvant être introduites au cours de la dérivation. Nous présentons ci-dessous un extrait de la définition inductive `lambda-terms` issue de notre atelier[3] :

```
Section lambdaX.
(* X represente l'ensemble des constantes du lambda-calcul *)
Variable X:Set.

(* definition des lambda-termes *)
Inductive lambda_terms:Set:=
| num  : nat -> lambda_terms (* indices de De Bruijn *)
| ress : X ->lambda_terms (* constantes *)
| hyps : nat->semType->lambda_terms (* ressources hypothetiques *)
| appl : lambda_terms -> lambda_terms -> lambda_terms (* application *)
| abs  : semType ->lambda_terms  -> lambda_terms (* abstraction, le
premier parametre etant le type de la variable liee *)
...
```

[3] La définition complète se trouve dans le fichier `Kernel/lambda_bruijn.v` de [12].

Le module de λ-calcul considéré comporte un ensemble de fonctions et de tactiques permettant d'effectuer différents traitements sur les λ-termes profondément plongés (e.g., abstraction d'une ressource hypothétique, substitution d'une ressource hypothétique par un λ-terme, β-réduction etc). Ces utilitaires seront employés pour faciliter le calcul de la sémantique d'un énoncé à partir de sa dérivation syntaxique.

Le calcul de la sémantique dérivationnelle (i.e., la structure prédicative) d'une expression se fait de manière compositionnelle grâce à une fonction récursive prenant comme paramètre une dérivation syntaxique de l'expression en question. Cette fonction explore la structure de la dérivation et associe à chaque étape de déduction une construction du λ-calcul typé comme le montre l'extrait simplifié suivant :

```
Fixpoint semanticTerm (der:Γ ⊢^R A):=
match der with
| Ax((hyp x), A) => hyps x* A* (* x est une ressource hypothetique *)
| Ax((word w), A) => ress w*    (* w est une ressource linguistique de W *)
| /I(i, x, Γ, A, B, der') =>λ(x*:B*). (semanticTerm der')
| /E(i, Γ, Δ, A, B, der1 der2) => appl (semanticTerm der1)(semanticTerm der2)
| ...
end.
```

La sémantique finale d'un énoncé est calculée à partir de sa structure prédicative en substituant chaque variable libre représentant un mot m_i par sa sémantique lexicale.
L'utilisation d'un plongement profond du λ-calcul pour représenter la sémantique dérivationnelle des énoncés est bénéfique car elle nous permet de raisonner sur la structure des termes obtenus. Ainsi, on est capable de prouver le lemme :

$$\forall\ \Gamma\ C\ R\ (d : \Gamma \overset{R}{\vdash} C),\ \mathit{linear}(R) \Rightarrow \mathit{linear\_term}(\mathit{semanticTerm}(d))$$

Ce lemme énonce que dans un système multimodal supportant des règles structurelles linéaires, tous les termes représentant la sémantique dérivationnelle d'une expression sont forcément linéaires (chaque symbole λ lie exactement une seule variable).
La manipulation des λ-termes profondément plongés s'avère toutefois difficile. En effet, ces termes sont relativement illisibles à cause de la présence des indices de De Bruijn utilisés à la place des variables nommées. En outre, ce plongement profond ne nous permet pas d'utiliser les outils prédéfinis du méta-langage (tactiques, schémas de récurrence ...). Il n'est donc pas efficace d'envisager l'utilisation de cette approche pour raisonner sur le contenu sémantique des phrases.
Afin de remédier aux faiblesses du plongement profond du λ-calcul, nous adoptons une approche mixte pour rendre compte de la sémantique de **Montague** dans notre atelier. Ainsi, le plongement profond est utilisé pour raisonner sur la structure des termes tandis que nous avons recours au λ-calcul de **Coq** pour raisonner sur la sémantique des langues naturelles. La communication entre ces deux niveaux de présentation se fait par l'intermédiaire de deux fonctions d'interprétation `trans-lamb` et `trans-type` qui traduisent respectivement les termes et les types sémantiques profondément enchâssés en des λ-termes et des types de **Coq** et ce de manière récursive. Précisons que la fonction `trans-lamb` est construite de manière interactive, elle associe à chaque λ-terme `u`, profondément plongé et portant le type sémantique `ty`, un terme **Coq** de type `(trans-type ty)` et ce dans le cas où `u` ne comprend aucune ressource hypothétique (toutes les ressources hypothétiques sont déchargées à la fin de la dérivation). Cette traduction dépend d'un domaine `E`

(interprétant l'ensemble des individus), d'une liste de variables nommées $\mathcal{V}$ servant à interpréter les indices de De Bruijn, et de deux fonctions $\mathcal{L}$ et $\mathcal{S}$ interprétant respectivement les constantes logiques (e.g., `IMPL` : implication, `FORALL` : quantification universelle) et les ressources linguistiques (e.g., *Dalai Lama*, *modeste*). Le tableau ci-après récapitule certains cas cruciaux de cette interprétation. Pour plus de lisibilité, nous utilisons la même notation ⟦ ⟧ pour représenter les deux fonctions `trans-lamb` et `trans-type`. Les deux symboles `-->` et `|*|` dénotent respectivement le plongement profond de la flèche fonctionnelle et du produit cartésien (i.e., deux constructeurs du type inductif `semType`).

Interprétation des types	Interprétation des termes
⟦e⟧=E	$\llbracket \texttt{num i} \rrbracket^{\mathcal{V}}=\mathcal{V}\texttt{[i]}$
⟦t⟧=Prop	$\llbracket \texttt{ress w} \rrbracket^{\mathcal{V}}$=si w est une constante logique alors $\mathcal{L}$(w) sinon $\mathcal{S}$(w)
⟦a-->b⟧=⟦a⟧->⟦b⟧	$\llbracket \texttt{appl u v} \rrbracket^{\mathcal{V}}=(\llbracket \texttt{u} \rrbracket^{\mathcal{V}}\ \llbracket \texttt{v} \rrbracket^{\mathcal{V}})$
⟦a |*| b⟧=⟦a⟧*⟦b⟧	$\llbracket \texttt{abs ty u} \rrbracket^{\mathcal{V}}=\texttt{fun (x :}\llbracket \texttt{ty} \rrbracket\texttt{)=>}\llbracket \texttt{u} \rrbracket^{(x::\mathcal{V})}$

A titre d'exemple, le tableau ci-dessous illustre le contraste entre le plongement profond de la sémantique lexicale du quantificateur '*tout*' et son plongement superficiel obtenu automatiquement par la fonction d'interprétation ⟦ ⟧.

Plongement profond	`abs (e-->t) (abs (e-->t) (appl FORALL (abs e (appl (appl IMPL (appl (num 2)(num 0))) (appl (num 1)(num 0)))))) : (e-->t)--> (e-->t)-->t.`
Plongement superficiel	`fun (P : E->Prop) => fun (Q : E->Prop) =>forall (x : E), P(x)->Q(x) : (E->Prop)->(E->Prop)->Prop.`

Grâce à cette traduction, on est capable de prouver formellement divers syllogismes tel que le suivant :

$$\frac{\llbracket\text{Tout sage est modeste}\rrbracket \quad \llbracket\text{Dalai Lama est un sage}\rrbracket}{\llbracket\text{Dalai Lama est modeste}\rrbracket}$$

L'utilisation du λ-calcul inhérent à **Coq** permet de transformer ce syllogisme en un but automatiquement démontrable grâce à la tactique *firstorder* :

$$\frac{\forall x.\ \textbf{sage}(x) \Rightarrow \textbf{modeste}(x) \quad \textbf{sage}(\textbf{Dalai}-\textbf{Lama})}{\textbf{modeste}(\textbf{Dalai}-\textbf{Lama})}$$

Notons que l'utilisation du calcul des constructions inductives pour exprimer la sémantique lexicale des mots présente d'autres avantages. En effet, ceci permet d'enrichir la sémantique de **Montague** par l'introduction de définitions inductives ou de spécifications fortes. A titre d'exemple, le calcul des prédicats de premier ordre ne permet pas d'exprimer aisément la sémantique des quantificateurs indéfinis qui représentent un groupe quelconque d'individus de cardinalité bien déterminée (e.g. *un, trois, dix*). La possibilité de définir une grande variété de types abstraits de données en **Coq** nous permet de raffiner ces expressions. Dans ce qui suit, nous proposons une définition en **Coq** de la sémantique du quantificateur *'trois'*, qu'on compare à celle de

Montague :

Spécification avancée	$\lambda P.\ \lambda Q.\ \exists p : (\Sigma\ (F : FSet).\|F\| = 3),$ $(\forall x,\ \ x \in \Pi_1(p)\ \Rightarrow\ P(x)\ \wedge\ Q(x))$
Montague	$\lambda P.\ \lambda Q.\ \exists\ x_1\ x_2\ x_3,\ (x_1 \neq x_2) \wedge$ $(x_2 \neq x_3) \wedge (x_1 \neq x_3) \wedge P(x_1) \wedge P(x_2)$ $\wedge P(x_3) \wedge Q(x_1) \wedge Q(x_2) \wedge Q(x_3)$

Dans cette définition, nous utilisons un type abstrait de données *FSet* (de la bibliothèque standard de **Coq**) représentant les ensembles finis ainsi que la spécification forte Σ[4] pour exprimer l'existence constructive d'un élément vérifiant une certaine propriété. L'entité *p* qui est de type (Σ *(F : FSet). |F|=3*) peut être vue comme une paire dont le premier élément est un ensemble fini et le second est la preuve que cet ensemble a bien la cardinalité 3. Ainsi, la sémantique de la phrase *'Trois filles lisent Amok'* peut s'exprimer dans le calcul des constructions inductives de **Coq** comme ceci :

$$\exists p : (\Sigma(F : FSet).\,|F| = 3),\ (\forall x,\ \ x \in \Pi_1(p) \Rightarrow \textbf{Fille}(x) \wedge \textbf{Lire}(x,\ \textbf{Amok}))$$

Informellement, cette proposition signifie qu'il existe un ensemble fini de cardinalité trois dont les éléments sont tous des filles qui lisent *Amok*.
D'autre part, nous pouvons recourir aux définitions inductives afin de rassembler les différents sens d'un mot dans une seule composante sémantique. Ceci est le cas par exemple pour le mot équivoque *'bibliothèque'*.

```
Inductive Bibliotheque (ind:E):Prop:=
|sens1 : endroit ind ->
         contenir_livres ind ->
         Bibliotheque ind
|sens2 : collection_prgms ind->
         Bibliotheque ind.
```

La définition ci-dessus précise qu'une *'bibliothèque'* peut être soit un endroit contenant des livres (énoncé du constructeur sens1) soit une collection de programmes (énoncé du constructeur sens2).

3.3 Méta-linguistique en Coq

Comme dans tout formalisme logique, l'ensemble réduit de règles de base définissant la notion de dérivation doit s'enrichir d'un catalogue de règles dérivées et méta-théorèmes, qui permettent d'étudier les propriétés du formalisme considéré. Dans le cas des grammaires multimodales, ce point est particulièrement important, étant donné le grand degré de paramétrisation des systèmes déductifs sous-jacents. Avoir accès à un tel ensemble de règles et faciliter la dérivation de nouvelles règles sont deux fonctionnalités fructueuses pour le chercheur. En effet, ceci lui facilite l'étude de la façon dont l'interaction entre différents modes permet de rendre compte de phénomènes linguistiques, tant génériques que spécifiques à telle ou telle classe de grammaires.

[4]Cet opérateur est représenté par le type *sig* en **Coq**

3.3.1 Règles dérivées

Plusieurs règles dérivées génériques sont prouvées dans l'atelier. La forme générale de ces règles est la suivante :

$$\frac{cond(R) \quad \Gamma_1 \overset{R}{\vdash} A_1 \; ... \; \Gamma_n \overset{R}{\vdash} A_n}{\Gamma \overset{R}{\vdash} A}$$

La contrainte *cond(R)* permet de définir des règles conditionnées dont l'application est limitée aux grammaires supportant un ensemble de règles structurelles *R* qui vérifient cette condition. Quand aucune contrainte n'est spécifiée (i.e., la règle est valide dans toutes les grammaires), l'opérateur de déduction $\overset{R}{\vdash}$ sera noté $\vdash$.
Dans ce qui suit, nous présenterons quelques exemples de ces règles dérivées. Les démonstrations de ces dernières mettent en œuvre des techniques variées telles la récurrence sur les dérivations, la généralisation, la réflexion [2] etc. Afin d'alléger l'écriture de ces règles, les quantifications universelles seront généralement omises. En outre, nous ferons souvent abstraction des ressources linguistiques qui décorent les différents types syntaxiques ; nous manipulerons par conséquent des séquents dont les contextes sont non-étiquetés.

Règles dérivées simples

Les règles dérivées sont dites simples si elles admettent une preuve directe qui est élaborée par l'application d'un nombre fini de règles d'inférence. De telles règles constituent des schémas de dérivations dans la mesure où elles factorisent une succession d'étapes de déduction, elles sont prouvées une seule fois et peuvent être utilisées autant de fois que l'on souhaite.
Dans ce qui suit, nous présentons un exemple de règles dérivées simples, il s'agit de la règle E_m utilisée pour l'analyse du phénomène de l'extraction médiane. Cette dernière est un cas typique de règles permettant l'étude et la compréhension des principes d'interaction et leur influence sur la syntaxe des langues.

$$\frac{P\Diamond(i,\ j) \in R \qquad L\Diamond(i,\ j) \in R}{(Z_1, (X/_i Y, X\backslash_i (Z_1\backslash_i Z_2))^i)^i \overset{R}{\vdash} Z_2/_i\Diamond_j\Box_j Y}\ E_m$$

La règle E_m est vérifiée dans toute grammaire multimodale supportant à la fois les deux règles structurelles $P\Diamond(i,\ j)$ et $L\Diamond(i,\ j)$ (cf. Fig. 9). Les étapes principales de sa preuve sont présentées de manière synthétique dans la figure 15.

On peut utiliser cette règle pour effectuer une dérivation simplifiée de la subordonnée relative '*que Dalai-Lama applique constamment*' comme le montre la figure 14 :

$$\frac{\dfrac{}{que \vdash (n\backslash_a n)/_a(s/_a\Diamond_c\Box_c np)}\ Ax \qquad \dfrac{}{(Dalai\ Lama, (applique, constamment)^a)^a \vdash (s/_a\Diamond_c\Box_c np)}\ E_m}{(que, (Dalai\ Lama, (applique, constamment)^a)^a)^a \vdash n\backslash_a n}\ /_aE$$

FIG. 14: Application de la règle dérivée E_m

$$\dfrac{\dfrac{\dfrac{}{\Diamond_j\Box_j Y \overset{R}{\vdash} \Diamond_j\Box_j Y}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Box_j Y \overset{R}{\vdash} \Box_j Y}\,Ax}{< \Box_j Y >_j \overset{R}{\vdash} Y}\,\Box_j E}{\vdots}}{(Z_1, ((X/_i Y, < \Box_j Y >_j)^i, X\backslash_i(Z_1\backslash_i Z_2))^i \overset{R}{\vdash} Z_1\backslash_i Z_2}\,\backslash_i E}{(Z_1, ((X/_i Y, < \Box_j Y >_j)^i, X\backslash_i(Z_1\backslash_i Z_2))^i)^i \overset{R}{\vdash} Z_2}\,P\Diamond(i,\ j)}{((Z_1, (X/_i Y, X\backslash_i(Z_1\backslash_i Z_2))^i, < \Box_j Y >_j)^i)^i \overset{R}{\vdash} Z_2}\,L\Diamond(i,\ j)}{((Z_1, (X/_i Y, X\backslash_i(Z_1\backslash_i Z_2))^i)^i, \Diamond_j\Box_j Y)^i \overset{R}{\vdash} Z_2}\,\Diamond_j E}{(Z_1, (X/_i Y, X\backslash_i(Z_1\backslash_i Z_2))^i)^i \overset{R}{\vdash} Z_2/_i\Diamond_j\Box_j Y}\,/_i I$$

FIG. 15: Preuve de la règle dérivée E_m

L'application de la règle dérivée E_m se fait en unifiant le but courant avec le schéma général proposé. Cette unification permet au système **Coq** d'instancier automatiquement les différentes variables quantifiées universellement comme ceci :

$$i := a;\quad j := c;\quad Z_1 := np;\quad Y := np;\quad X := np\backslash_a s;\quad Z_2 := s$$

Règles avec calculs auxiliaires

Nous présentons dans cette section un exemple de règles dont l'utilisation provoque une réorganisation complexe du contexte, permettant ainsi de remplacer l'application d'un nombre important de règles élémentaires par un simple calcul. Il s'agit de la règle dédiée au traitement du phénomène de dépendances non bornées. Ce phénomène linguistique a lieu quand deux éléments arbitrairement éloignés d'une phrase sont sémantiquement reliés [26]. Les exemples (10) ci-après illustrent bien ce phénomène (où _ représente l'emplacement de l'objet extrait) :

(10) a. La vertu que tout sage applique _ est l'humilité.
b. La vertu que Marie pense que tout sage applique _ est l'humilité.
c. La vertu que Jean croit que Marie pense que tout sage applique _ est l'humilité.

Après l'introduction de l'élément extrait, ce dernier se place en surface à l'extrême droite du fragment, il faut donc restructurer le contexte pour remettre l'élément extrait à sa place d'origine. C'est ce que fait la règle *Dep* ci-dessous, qui place l'élément extrait comme frère droit du dernier sous-contexte dominé par le connecteur structurel $(\ ,\ )^j$ et se trouvant dans la branche la plus à droite du contexte initial.

$$\dfrac{L\Diamond(i,\ j) \in R \quad \rho(\Gamma,\ < A >_j,\ i) \overset{R}{\vdash} C}{(\Gamma, \Diamond_j A)^i \overset{R}{\vdash} C}\ Dep$$

Cette règle utilise la fonction auxiliaire ρ définie récursivement par :

$\rho(\Gamma,\ \Delta,\ i)$ = `si` Γ `est de la forme` $(\Gamma_1,\ \Gamma_2)^i$
`alors` $(\Gamma_1\ ,\ \rho(\Gamma_2,\ \Delta,\ i))^i$
`sinon` $(\Gamma\ ,\ \Delta)^i$

La preuve de *Dep* se fait en démontrant par récurrence sur Γ que le contexte $\rho(\Gamma,\ < A >_j,\ i)$ est obtenu en réécrivant un nombre fini de fois le contexte initial $(\Gamma, < A >_j)^i$ et ce par la règle de réécriture $L\Diamond(i,\ j)$.
On peut appliquer cette règle pour effectuer une dérivation de '*que Marie pense que tout sage applique* _'. L'élément extrait sera placé automatiquement à son endroit initial (objet du verbe '*applique*'). Cette restructuration est présentée de manière plus illustrée dans la figure 16.

$$\dfrac{\dfrac{\dfrac{\dots}{(Marie,(pense,(que,((tout,sage)^a,(applique,<\_:\Box_c np>_c)^a)^a)^a)^a)^a \vdash s}}{((Marie,(pense,(que,((tout,sage)^a,applique)^a)^a)^a)^a,(\_:\Diamond_c\Box_c np))^a \vdash s}\ Dep}{(Marie,(pense,(que,((tout,sage)^a,applique)^a)^a)^a)^a \vdash s/_a\Diamond_c\Box_c np}\ /_aI$$

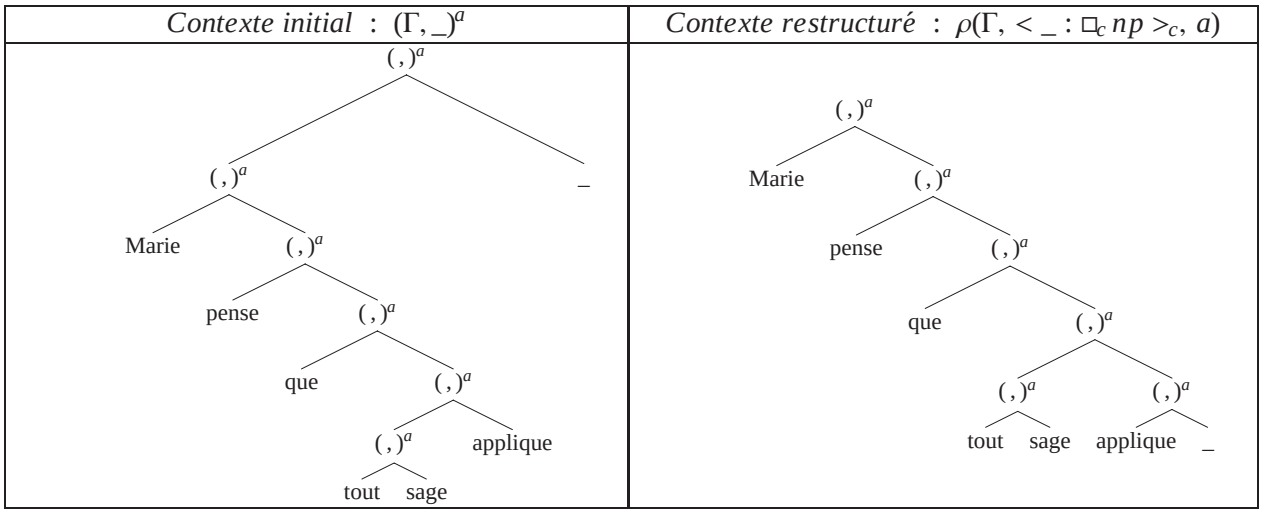


FIG. 16: Exemple de restructuration de contextes par la fonction ρ

Remarque 3.3.1 *Notons que l'application de cette règle est restreinte. En effet, elle place toujours l'élément extrait dans la branche la plus à droite du contexte à analyser. Par conséquent, cette règle n'est pas compatible avec les fragments où l'extraction est médiane. A titre d'exemple, l'application de cette règle sur l'exemple de la partie précédente placera l'objet extrait comme frère droit de l'adverbe* 'constamment'*, cette position inadéquate bloquera le déroulement de l'analyse.*

3.3.2 Règles dérivées & Récurrence sur les dérivations

Les techniques employées précédemment (e.g., application directe des règles d'inférence) admettent un champ d'application relativement restreint. En effet, la résolution de plusieurs problèmes ne se limite pas à l'application de ces techniques mais nécessite le recours à un outil plus

puissant à savoir la récurrence sur les dérivations. Parmi ces problèmes citons la règle dérivée d'*isotonicité* présentée ci-après.

La règle d'isotonicité

Dans ce qui suit, nous considérons les grammaires multimodales basées sur un paquetage R de règles structurelles sous la forme faible de Sahlqvist (cf. définition 2.2.6) .
La règle d'isotonicité pour le connecteur $/_i$ s'exprime comme suivant :

$$\frac{A \vdash B}{A/_i C \vdash B/_i C}\ /_i Iso$$

Toute tentative de recherche d'une démonstration directe de cette règle est vouée à l'échec. En effet, considérons une dérivation arbitraire d_1 du séquent $A \vdash B$ et essayons de construire une dérivation de $A/_i C \vdash B/_i C$. La seule règle qui pourrait mener à un tel séquent est la règle $/_i I$. Afin de l'utiliser, on doit disposer d'une dérivation du séquent $(A/_i C\ ,\ C)^i \vdash B$. Encore une fois, seule la règle $/_i E$ peut engendrer ce dernier séquent, mais l'application de cette règle exige une dérivation de $A/_i C \vdash B/_i C$: nous tombons alors dans un cercle vicieux !

Il est intéressant de noter que la règle d'isotonicité est aisément dérivable dans le calcul de séquents. En outre, on peut facilement construire des dérivations directes pour des instances particulières de cette règle en déduction naturelle (obtenues en replaçant A, B et C par des formules spécifiques) comme le montre la figure 17. Remarquons toutefois que la dérivation ci-dessous ne contient aucune sous-dérivation du séquent $np \vdash s/_i(np\backslash_i s)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np/_i n \vdash np/_i n}\ Ax \quad \dfrac{}{n \vdash n}\ Ax}{(np/_i n, n)^i \vdash np}\ /_i E \qquad \dfrac{}{np\backslash_i s \vdash np\backslash_i s}\ Ax}{((np/_i n, n)^i, np\backslash_i s)^i \vdash s}\ \backslash_i E}{(np/_i n, n)^i \vdash s/_i (np\backslash_i s)}\ /_i I}{np/_i n \vdash (s/_i (np\backslash_i s))/_i n}\ /_i I$$

FIG. 17: Dérivation d'une instance de la règle d'isotonicité

La règle d'isotonicité peut être aisément prouvée si on réussit à démontrer que la règle de coupure *cut* est une règle dérivée de la déduction naturelle :

$$\frac{\Delta \vdash A \quad \Gamma[A] \vdash B}{\Gamma[\Delta] \vdash B}\ cut$$

La figure 18 montre comment on peut construire une preuve de la règle d'isotonicité[5] en appliquant la règle *cut*.

Notre objectif dès lors est de dériver la règle de coupure en déduction naturelle. Avant de décrire les étapes principales de notre preuve, il est utile de présenter brièvement le principe de récurrence sur les dérivations engendré automatiquement par le système **Coq**.

[5]La règle d'isotonicité est prouvée dans l'atelier. Le lemme correspondant `Slash_isotonicity` se trouve dans le fichier `Meta/derivedRulesNatDed.v`

$$\dfrac{\dfrac{\dfrac{}{A/_iC \vdash A/_iC}\ Ax \quad \dfrac{}{C \vdash C}\ Ax}{(A/_iC,\ C)^i \vdash A}\ /_iE \qquad A \vdash B}{\dfrac{(A/_iC,\ C)^i \vdash B}{A/_iC \vdash B/_iC}\ /_iI}\ cut$$

FIG. 18: Preuve de $/_i$ Iso en utilisant cut

Récurrence sur les dérivations

Soit P un prédicat -ou plus généralement une fonction quelconque[6]- défini sur les dérivations. Etant donné que les dérivations sont des habitants d'un type dépendant (i.e., le type *natded*), P prend en réalité trois arguments : un contexte Γ, une formule syntaxique A et une dérivation d du séquent $\Gamma \vdash A$. Toutefois, puisque Γ et A peuvent être facilement inférés à partir de d, nous nous contenterons souvent d'écrire $P(d)$ à la place de $P(\Gamma, A, d)$.

Le schéma de récurrence sur les dérivations (i.e., le théorème *natded_ind*) est récapitulé ci-après. Pour plus de clarté, nous présentons ce principe sous la forme d'une règle logique paramétrée par un *prédicat P* sachant que ce schéma peut être facilement adapté pour prendre en charge des fonctions quelconques.

Afin de prouver l'énoncé suivant :

$$\forall\, \Gamma\, \forall\, A\, \forall\, d : (\Gamma \vdash A),\ P(\Gamma, A, d)$$

il est suffisant de prouver ce qui suit :

Règle d'Axiome : $\forall\, A_1,\ P(Ax(A_1))$.

Règle $/_iI$ **:**

$$\forall\, A_1,\, B_1,\, \Gamma_1,\, i,\ d_1 : ((\Gamma_1, B_1)^i \vdash A_1), P(d_1) \Longrightarrow P(/_I(i,\, \Gamma_1,\, A_1,\, B_1,\, d_1))$$

Règle $/_iE$ **:**

$$\begin{array}{c}\forall\, \Gamma_1,\, \Gamma_2,\, A_1,\, A_2,\, i,\, d_1 : (\Gamma_1 \vdash A_1/_iA_2),\, d_2 : (\Gamma_2 \vdash A_2),\\ P(d_1) \wedge P(d_2) \Longrightarrow P(/_E(i, \Gamma_1, A_1, \Gamma_2, A_2, d_1, d_2))\end{array}$$

...

Règles structurelles :

$$\begin{array}{c}\forall \Gamma_1,\, \Delta,\, A_1,\, r \in R,\, d_1 : (\Gamma_1[r(\Delta)] \vdash A_1)\\ P(d_1) \Longrightarrow P(S_R(r, \Gamma_1,\, \Delta, A, d_1))\end{array}$$

Au total, douze conditions doivent être vérifiées lors d'un raisonnement par récurrence, et la dernière dépend d'un paquetage quelconque de règles structurelles. En outre, trois règles $\bullet E$, $\diamond_j E$ et S_R sont appliquées à des sous-contextes arbitrairement imbriqués dans le contexte initial, ainsi leur preuve associée exige souvent une deuxième récurrence sur la profondeur de l'endroit où elles sont appliquées. Le recours à un assistant de preuves tel que **Coq** est indubitablement bénéfique, puisqu'il permet d'assurer la construction interactive de preuves complètes traitant avec rigueur tous les cas de récurrence engendrés.

[6]Les prédicats sont des cas particuliers de fonctions qui construisent des propositions.

Dérivation de la règle de coupure en déduction naturelle :

Un premier essai Afin de dériver la règle *cut*, une première tentative consiste à raisonner par récurrence sur la dérivation du séquent $\Gamma'[A] \vdash B$. Dans ce cas, la propriété de récurrence $P(\Gamma, B, d)$ est définie comme suit : $\forall\, \Gamma',\ \Gamma'[A] = \Gamma \Longrightarrow \Gamma'[\Delta] \vdash B$.

Malheureusement, cette piste n'est pas fructueuse. En effet, supposons que la dernière étape de la dérivation coïncide avec l'application d'une règle structurelle non-linéaire telle que la règle *MC(i, j)* :

$$\dfrac{\dfrac{d_1}{\Gamma_1[((\Delta_1\ ,\ \Delta_3)^j, (\Delta_2\ ,\ \Delta_3)^j)^i] \vdash B}}{\Gamma_1[((\Delta_1\ ,\ \Delta_2)^i\ ,\ \Delta_3)^j] \vdash B}\ MC(i,\ j)$$

Comme Γ=$\Gamma'[A]$, il se peut que la formule A fasse partie du sous-contexte Δ_3 qui est dupliqué après réécriture. Dans ce cas, l'hypothèse de récurrence ne peut être appliquée puisque cette dernière ne prend pas en charge les remplacements multiples (i.e., substitution de A par Δ à plusieurs endroits).

Règle de coupure généralisée Il arrive des fois qu'on soit incapable de démontrer une propriété particulière par une simple récurrence. Pour remédier à cela, il est fréquent de tenter de prouver une généralisation de cette propriété. Dans notre cas, au lieu de considérer des contextes linéaires comprenant un seul trou, e.g., $\Gamma'[\ ]$, nous généralisons la règle de coupure aux contextes dotés d'un nombre quelconque de trous qu'on note par $\Gamma'\{\ \}$. Ainsi la nouvelle variante de la règle de coupure qui tient en compte les remplacements multiples prend la forme suivante :

$$\dfrac{\Delta \vdash A \quad \Gamma'\{A\} \vdash B}{\Gamma'\{\Delta\} \vdash B}\ cut'$$

Afin de prouver cette nouvelle règle, nous procédons par récurrence sur la dérivation d de $\Gamma'\{A\} \vdash B$, où le prédicat considéré $P(\Gamma, B, d)$ n'est autre que $\forall\, \Gamma',\ \Gamma'\{A\} = \Gamma \Longrightarrow \Gamma'\{\Delta\} \vdash B$.

Croquis de la preuve La démontration de la règle *cut'* consiste à vérifier les douze conditions engendrées par la schéma de récurrence. Cette preuve est entièrement formalisée en **Coq**, elle peut être consultée dans le fichier `Meta/derivedRulesNatDed.v` (lemme `cut_rule`) de notre atelier [12]. Plusieurs cas sont prouvés de manière similaire, aussi nous contenterons nous, dans ce qui suit, de présenter les cas les plus emblématiques.

Règle d'axiome : Si la dérivation d est réduite à l'application de la règle d'axiome, alors Γ' est un contexte vide, $B = A$, $\Gamma'\{\Delta\} = \Delta$ et par conséquent le séquent $\Gamma'\{\Delta\} \vdash B$ est dérivable (puisqu'on a bien $\Delta \vdash A$).

Règle $/_I$: Si la dernière règle de déduction de d est $/_I$, alors B a la forme $B_1/_iB_2$ et d admet une sous-derivation d_1 de $(\Gamma'\{A\}\ ,\ B_2)^i \vdash B_1$. Grâce à l'hypothèse de récurrence, on peut construire une dérivation d_1' : $(\Gamma'\{\Delta\}\ ,\ B_2)^i \vdash B_1$, et, en appliquant la règle $/_I$, on obtient facilement une dérivation du séquent envisagé $\Gamma'\{\Delta\} \vdash B$.

Règle $/_E$: Si la dérivation d se termine par une application de la règle $/_E$, alors $\Gamma'\{A\}$ peut être décomposé en $(\Gamma_1\{A\}\,,\,\Gamma_2\{A\})^i$ et d est formée à partir des deux sous-dérivations d_1 de $\Gamma_1\{A\} \vdash B/_iB_2$ et d_2 de $\Gamma_2\{A\} \vdash B_2$. Par hypothèse de récurrence, on peut construire deux dérivations d_1' de $\Gamma_1\{\Delta\} \vdash B/_iB_2$ et d_2' de $\Gamma_2\{\Delta\} \vdash B_2$. En rassemblant ces deux sous-dérivations moyennant la règle $/_E$ on obtient alors une dérivation de $\Gamma'\{\Delta\} \vdash B$

Règle $\bullet_E$: Si la dérivation d se termine par une application de la règle $\bullet_E$, alors Γ doit être à la fois de la forme $\Gamma'\{A\}$ et $\Gamma_1[\Delta_1]$. Il existe alors deux contextes à trous multiples Γ_1' et Δ_1' tels que $\Gamma{=}\Gamma_1'\{A\}[\Delta_1'\{A\}]$. La dérivation d prend alors l'allure suivante :

$$\dfrac{\dfrac{d_1}{\Delta_1'\{A\} \vdash A_1 \bullet_i B_1} \quad \dfrac{d_2}{\Gamma_1\{A\}[(A_1, B_1)^i] \vdash C}}{\Gamma_1'\{A\}[\Delta_1'\{A\}] \vdash C}\ \bullet_E$$

D'après les hypothèses de récurrence, on peut construire une dérivation d_1' de $\Delta_1'\{\Delta\} \vdash A_1 \bullet_i B_1$ à partir de d_1 et une dérivation d_2' de $\Gamma_1'\{\Delta\}[(A_1, B_1)^i] \vdash C$ à partir de d_2. La combinaison des deux sous-dérivations d_1' et d_2' par la règle $\bullet_E$ engendre une preuve du séquent $\Gamma_1'\{\Delta\}[\Delta_1'\{\Delta\}] \vdash C$. Ceci permet d'achever la preuve puisque $\Gamma_1'\{\Delta\}[\Delta_1'\{\Delta\}]{=}\Gamma'\{\Delta\}$.

Règles structurelles : On suppose dans ce cas que la dernière étape de déduction de d coïncide avec l'application d'une règle structurelle sous la forme faible de Sahlqvist (cf. définition 2.2.6). Un lemme intermédiaire est utilisé pour prouver que toutes ces règles vérifient la propriété de substitution énoncée comme suit :

$$\forall\, \Sigma_1\, \Sigma_2\, \Sigma_1'\, A,\ \Sigma_1 \xrightarrow{r} \Sigma_2 \wedge \Sigma_1 = \Sigma_1'\{A\} \Longrightarrow \exists \Sigma_2' \,|\, \Sigma_2 = \Sigma_2'\{A\} \wedge \forall \Delta,\ \Sigma_1'\{\Delta\} \xrightarrow{r} \Sigma_2'\{\Delta\}$$

En appliquant ce lemme, la dernière étape de déduction de d prend la forme suivante (où $\Sigma_1{=}\Sigma_1'\{A\}$ et $\Sigma_2 = \Sigma_2'\{A\}$) :

$$\dfrac{\dfrac{d_1}{\Sigma_1 \xrightarrow{r} \Sigma_2 \quad \Gamma_1'\{A\}[\Sigma_2'\{A\}] \vdash C}}{\Gamma_1'\{A\}[\Sigma_1'\{A\}] \vdash C}\ S_R(r)$$

Grâce à l'hypothèse de récurrence, on peut construire une dérivation d_1' de $\Gamma_1'\{\Delta\}[\Sigma_2'\{\Delta\}] \vdash C$. Or, d'après la propriété de substitution, la réécriture est conservée si on substitue un nombre arbitraire de formules A par le contexte Δ dans les deux contextes Σ_1 et Σ_2 (i.e., $\Sigma_1'\{\Delta\} \xrightarrow{r} \Sigma_2'\{\Delta\}$). C'est ainsi qu'on peut déduire une dérivation du séquent $\Gamma_1'\{\Delta\}[\Sigma_1'\{\Delta\}] \vdash C$ par simple application de la règle structurelle r.

Remarque Notons que la dérivation de la règle de coupure est basée sur un algorithme récursif dont le rôle est de transformer une dérivation utilisant un lemme intermédiaire en une dérivation directe (sans lemme auxiliaire). A titre d'exemple, la figure 19 présente la dérivation d'une instance de la règle $/_iIso$ qui fait recours à la règle de coupure pour introduire le lemme intermédiaire $(np/_i\, n\,,\, n)^i \vdash np$. L'application de la procédure encapsulée par la règle de coupure engendre la dérivation directe présentée dans la figure 17.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{np/_i n \vdash np/_i n}\,Ax \quad \dfrac{}{n \vdash n}\,Ax}{(np/_i n, n)^i \vdash np}\,/_i E \qquad \dfrac{\dfrac{\dfrac{}{np \vdash np}\,Ax \quad \dfrac{}{np\backslash_i s \vdash np\backslash_i s}\,Ax}{(np, np\backslash_i s)^i \vdash s}\,\backslash_i E}{np \vdash s/_i (np\backslash_i s)}\,/_i I}{(np/_i n, n)^i \vdash s/_i (np\backslash_i s)}\,cut}{np/_i n \vdash (s/_i (np\backslash_i s))/_i n}\,/_i I$$

FIG. 19: Dérivation d'une instance de la règle $/_i$ Iso (utilisant *cut*)

3.3.3 Critères de non dérivabilité

Coq offre un bon environnement qui combine harmonieusement le raisonnement et le calcul. Le raisonnement est un outil plus puissant que le calcul, toutefois, il est difficilement automatisable. Le système **Coq**, comme la majorité des assistants de preuves, vérifie le principe de Poincaré [18] (i.e., les calculs n'ont pas besoin de preuves, ils se font de manière systématique). Il est donc salutaire de pouvoir basculer, quand ceci est possible, du raisonnement au calcul pour pouvoir tirer profit de l'automatisation complète de ce dernier. Cette méthode est au cœur de la technique de *réflexion* [2] qui est fréquemment utilisée dans notre atelier. Le but de la réflexion est de pouvoir effectuer des raisonnements complexes en ayant recours au calcul. La preuve que ce raisonnement est réductible au calcul se fait une seule fois, mais elle peut être utilisée autant qu'on le souhaite pour automatiser le raisonnement.

Nous nous servons de cette technique pour prouver formellement qu'un séquent $\Gamma \overset{R}{\vdash} C$ n'est pas dérivable dans une logique multimodale donnée. Pour atteindre cet objectif, il suffit de trouver un *invariant* à savoir une condition nécessaire définie de manière calculatoire et vérifiée par tous les séquents dérivables. Par conséquent, tous les séquents ne vérifiant pas cette condition seront forcément non dérivables.

Polarité & Atomes

On illustre, dans ce qui suit, l'utilisation de la technique de réflexion par le critère de non dérivabilité basé sur le calcul de polarité d'atomes [97].
On définit la polarité d'un atome p dans un type syntaxique (resp. un contexte) comme étant la différence entre le nombre d'occurences positives et le nombre d'occurences négatives de cet atome dans le type (resp. le contexte) en question :

$$\begin{aligned}
pol_p(q) &= si\ (p = q)\ alors\ 1\ sinon\ 0 \\
pol_p(A/_i B) &= pol_p(A) - pol_p(B) \\
pol_p(B\backslash_i A) &= pol_p(A) - pol_p(B) \\
pol_p(A \bullet_i B) &= pol_p(A) + pol_p(B) \\
pol_p(\Diamond_j A) &= pol_p(A) \\
pol_p(\Box_j A) &= pol_p(A) \\
pol_p((\Delta_1\ ,\ \Delta_2)^i) &= pol_p(\Delta_1) + pol_p(\Delta_2) \\
pol_p(<\Delta>_j) &= pol_p(\Delta)
\end{aligned}$$

On peut aisément prouver par récurrence sur les dérivations que si les règles du paquetage R sont toutes linéaires et si le séquent $\Gamma \overset{R}{\vdash} A$ est dérivable alors $pol_p(\Gamma)=pol_p(A)$. Ceci nous conduit à l'énoncé du théorème de polarité Pol_A qui exhibe une condition suffisante automatiquement vérifiable qui garantit la non dérivabilité d'un séquent.

$$\frac{linear(R) \quad pol_p(\Gamma) \neq pol_p(C)}{\Gamma \overset{R}{\nvdash} C}\ Pol_A$$

On peut, par exemple, appliquer ce théorème pour prouver l'agrammaticalité de la phrase (11) :

(11) *Money is bad master.

Pour cela, on considère une grammaire dotée d'un paquetage quelconque de règles structurelles linéaires et du lexique ci-dessous :

Money	master	is	a	bad
np	n	$(np\backslash_a s)/_a np$	$np/_a n$	$n/_a n$

Soit Γ le contexte structuré représentant l'énoncé (11). Il est facile de constater que la polarité d'un atome p dans le contexte Γ ne dépend aucunement de la structure arborescente de ce dernier mais uniquement du contenu de ses feuilles. Ainsi on a ce qui suit :

- $pol_{np}(\Gamma)=pol_{np}(np) + pol_{np}((np\backslash_a s)/_a np) + pol_{np}(n/_a n) + pol_{np}(n)=1+(-2)+0+0=-1$.
- $pol_{np}(s)=0$.

Puisque $pol_{np}(\Gamma)\neq pol_{np}(s)$, on déduit que $\Gamma \overset{R}{\nvdash} s$, ce qui confirme la mal-formation de l'énoncé (11).

Polarité & Opérateurs unaires

On présentera dans ce qui suit un second critère de non dérivabilité basé sur un simple calcul d'occurrences des opérateurs unaires et vérifié pour une classe particulière de règles structurelles. Commençons par définir une fonction récursive ψ_j comme ceci :

$$\begin{aligned}
\psi_j(at) &= 0 \quad \text{(Pour tout atome } at\text{)} \\
\psi_j(A/_i B) &= \psi_j(A) - \psi_j(B) \\
\psi_j(B\backslash_i A) &= \psi_j(A) - \psi_j(B) \\
\psi_j(A \bullet_i B) &= \psi_j(A) + \psi_j(B) \\
\psi_j(\Box_i A) &= si\ (i = j)\ alors\ (\psi_j(A) - 1)\ sinon\ \psi_j(A) \\
\psi_j(\Diamond_i A) &= si\ (i = j)\ alors\ (\psi_j(A) + 1)\ sinon\ \psi_j(A) \\
\psi_j((\Gamma, \Gamma')^i) &= \psi_j(\Gamma) + \psi_j(\Gamma') \\
\psi_j(< \Gamma >_i) &= si\ (i = j)\ alors\ (\psi_j(\Gamma) + 1)\ sinon\ \psi_j(\Gamma)
\end{aligned}$$

Intuitivement, cette fonction calcule la différence entre la polarité des connecteurs $\Diamond_j$ et $< >_j$ et celle de l'opérateur dual $\Box_j$.

Soit $linear\Diamond_j$ la classe de règles structurelles linéaires qui conservent la multiplicité de l'opérateur structurel $< >_j$. Les règles de la figure 6 font partie de cette classe alors que ce n'est pas le cas pour la règle $K(i,\ j)$ ci-dessous qui incrémente de 1 le nombre d'occurrences de l'opérateur $< >_j$.

$$\frac{\Gamma[(< \Delta_1 >_j, < \Delta_2 >_j)^i] \vdash C}{\Gamma[< (\Delta_1, \Delta_2)^i >_j] \vdash C}\ K(i\ ,\ j)$$

Par simple récurrence sur la dérivation de $\Gamma \overset{R}{\vdash} C$, on démontre assez aisément le théorème suivant :

$$\forall\ \Gamma\ C\ R\ j,\ (linear\Diamond_j\ R)\ \Rightarrow\ \Gamma \overset{R}{\vdash} C\ \Rightarrow\ \psi_j(\Gamma) = \psi_j(C)$$

Ce théorème implique sa forme contraposée qui représente le noyau de notre critère de non-dérivabilité :

$$\forall\ \Gamma\ C\ R\ j,\ (linear\Diamond_j\ R)\ \Rightarrow\ \psi_j(\Gamma) \neq \psi_j(C)\ \Rightarrow\ \Gamma \overset{R}{\nvdash} C$$

Par conséquent, la preuve qu'une expression m_1, ..., m_n n'est pas de type syntaxique A peut se réduire, dans certains cas, à une simple vérification de trois conditions définies de manière calculatoire comme le montre le corollaire qui suit :

$$\frac{linear\Diamond_j(R) \quad rigid(lex) \quad \sum_{i=1}^{n} \psi_j(lex(m_i)) \geq \psi_j(A)}{m_1 ... m_n \notin \mathcal{L}_G(A)}\ ref$$

La démonstration se fait en remarquant que si le contexte structuré Γ a comme feuilles respectives les types t_1, t_2, ..., et t_n alors $\psi_j(\Gamma) \geq \sum_{i=1}^{n} \psi_j(t_i)$. Ce corollaire est utilisé dans la définition d'une tactique de réfutation permettant de prouver l'agrammaticalité de certaines clauses et ce en ayant recours à un calcul simple qui se fait automatiquement par le système.
Appliquons maintenant ce critère de non dérivabilité sur un exemple concret. Afin de prouver l'agrammaticalité de la subordonnée relative **(que tout sage appliquent constamment)*, on enrichit le lexique de la section 2.3.1 en introduisant explicitement l'encodage du trait morphosyntaxique *nombre* qui permet de distinguer entre les syntagmes singuliers et pluriels :

tout	$\Diamond_{sg}(s/_a(np\backslash_a s))/_a\Diamond_{sg} n$
sage	$\Diamond_{sg}\ n$
appliquent	$(\Diamond_{pl}\ np\backslash_a s)/_a np$

La figure 20 montre une preuve formelle de l'agrammaticalité de la clause relative ci-dessus dans la grammaire multimodale **NL** qui ne supporte aucune règle structurelle.

$$\frac{\overline{linear\Diamond_{sg}(\emptyset)} \quad \overline{rigid(lex)} \quad \dfrac{\overline{1 \geq 0}}{\overline{\sum_{i=1}^{5} \psi_{sg}(lex(m_i)) \geq \psi_{sg}(n\backslash_a n)}}}{que, tout, sage, appliquent, constamment \notin \mathcal{L}_{NL}(n\backslash_a n)}\ ref$$

FIG. 20: Preuve formelle de l'agrammaticalité d'une subordonnée relative

3.3.4 Méta-théorèmes

Interopérabilité entre les systèmes multimodaux :

L'atelier comprend la preuve complète de l'équivalence entre les deux présentations multimodales : le calcul des séquents et la déduction naturelle. La démonstration effectuée prend la forme

de fonctions certifiées qui permettent le passage d'un formalisme source à un formalisme cible supportant les règles structurelles du premier. Cette interopérabilité est salutaire dans la mesure où elle nous permet d'économiser un grand effort. En effet, la preuve étant constructive, on est donc capable d'exporter, assez aisément, toutes les règles dérivées et tous les théorèmes prouvés dans un formalisme pour les réutiliser dans le second formalisme sans avoir à les redémontrer.
La figure 22 illustre les étapes de traduction de certaines règles gauches (resp. règles d'élimination) du calcul de séquents (resp. de la déduction naturelle) en déduction naturelle (resp. calcul des séquents). Rappelons que les règles droites du calcul de séquents coïncident, quant à elles, avec les règles d'introduction de la déduction naturelle.

Simulation de règles structurelles

L'atelier $\mathcal{ICHARATE}$ contient aussi un certain nombre de lemmes de simulation dont l'objectif est de créer des ponts entre des systèmes multimodaux basés sur des paquetages différents de règles structurelles. La forme générale de ces lemmes est la suivante :

$$\frac{\Gamma \overset{R}{\vdash} C}{\Gamma \overset{R'}{\vdash} C}\ Sim$$

L'énoncé *Sim* stipule qu'il est possible de simuler toute réécriture résultante du paquetage *R* par une séquence de réécritures émanant de l'application des règles structurelles du paquetage *R'*. Par exemple, la figure 21 montre comment on peut simuler la règle de réécriture du paquetage $R{=}MA_1(i,\ j)$ moyennant le paquetage $R'{=}MA_2(i,\ j) \cup P(i) \cup P(j)$ où les règles structurelles $MA_1(i,\ j)$, $MA_2(i,\ j)$ et $P(i)$ sont définies comme suit :

$$\frac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3)^j)^i] \vdash C}{\Gamma[((\Delta_1,\Delta_2)^i,\Delta_3)^j] \vdash C}\ MA_1(i,\ j) \quad \frac{\Gamma[((\Delta_1,\Delta_2)^i,\Delta_3)^j] \vdash C}{\Gamma[(\Delta_1,(\Delta_2,\Delta_3)^j)^i] \vdash C}\ MA_2(i,\ j) \quad \frac{\Gamma[(\Delta_1,\Delta_2)^i] \vdash C}{\Gamma[(\Delta_2,\Delta_1)^i] \vdash C}\ P(i)$$

$$\begin{array}{|l|}\hline \dfrac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3)^j)^i] \overset{R'}{\vdash} C}{\Gamma[((\Delta_2,\Delta_3)^j,\Delta_1)^i] \overset{R'}{\vdash} C}\ P(i) \\ \dfrac{\Gamma[((\Delta_2,\Delta_3)^j,\Delta_1)^i] \overset{R'}{\vdash} C}{\Gamma[((\Delta_3,\Delta_2)^j,\Delta_1)^i] \overset{R'}{\vdash} C}\ P(j) \\ \dfrac{\Gamma[((\Delta_3,\Delta_2)^j,\Delta_1)^i] \overset{R'}{\vdash} C}{\Gamma[(\Delta_3,(\Delta_2,\Delta_1)^i)^j] \overset{R'}{\vdash} C}\ MA_2(i,\ j) \\ \dfrac{\Gamma[(\Delta_3,(\Delta_2,\Delta_1)^i)^j] \overset{R'}{\vdash} C}{\Gamma[((\Delta_2,\Delta_1)^i,\Delta_3)^j] \overset{R'}{\vdash} C}\ P(j) \\ \dfrac{\Gamma[((\Delta_2,\Delta_1)^i,\Delta_3)^j] \overset{R'}{\vdash} C}{\Gamma[((\Delta_1,\Delta_2)^i,\Delta_3)^j] \overset{R'}{\vdash} C}\ P(i) \\ \hline \end{array}$$

FIG. 21: Simulation de la règle $MA_1(i,\ j)$ moyennant le paquetage $MA_2(i,\ j) \cup P(i) \cup P(j)$

Règles du calcul de séquents	*Traduction en déduction naturelle*
$\dfrac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(A/_i B, \Delta)^i] \Rightarrow C}\ /_i L$	$\dfrac{\dfrac{\overline{A/_i B \vdash A/_i B}\ Ax \quad \Delta \vdash B}{(A/_i B, \Delta)^i \vdash A}\ /_i E \quad \Gamma[A] \vdash C}{\Gamma[(A/_i B, \Delta)^i] \vdash C}\ cut$
$\dfrac{\Gamma[(A, B)^i] \Rightarrow C}{\Gamma[A \bullet_i B] \Rightarrow C}\ \bullet_i L$	$\dfrac{\overline{A \bullet_i B \vdash A \bullet_i B}\ Ax \quad \Gamma[(A, B)^i] \Rightarrow C}{\Gamma[A \bullet_i B] \vdash C}\ \bullet_i E$
$\dfrac{\Gamma[< A >_j] \Rightarrow C}{\Gamma[\Diamond_j A] \Rightarrow C}\ \Diamond_j L$	$\dfrac{\overline{\Diamond_j A \vdash \Diamond_j A}\ Ax \quad \Gamma[< A >_j] \vdash C}{\Gamma[\Diamond_j A] \vdash C}\ \Diamond_j E$
$\dfrac{\Gamma[A] \Rightarrow C}{\Gamma[< \Box_j A >_j] \Rightarrow C}\ \Box_j L$	$\dfrac{\dfrac{\overline{\Box_j A \vdash \Box_j A}\ Ax}{< \Box_j A >_j \vdash A}\ \Box_j E \quad \Gamma[A] \vdash C}{\Gamma[< \Box_j A >_j] \vdash C}\ cut$

Règles de la déduction naturelle	*Traduction en calcul de séquents*
$\dfrac{\Gamma \vdash A/_i B \quad \Delta \vdash B}{(\Gamma, \Delta)^i \vdash A}\ /_i E$	$\dfrac{\Delta \Rightarrow B \quad \dfrac{\overline{A \Rightarrow A}\ Ax}{(A/_i B, \Delta)^i \Rightarrow A}\ /_i L}{\Gamma \Rightarrow A/_i B \quad (\Gamma, \Delta)^i \Rightarrow A}\ Cut$
$\dfrac{\Delta \vdash A \bullet_i B \quad \Gamma[(A, B)^i] \vdash C}{\Gamma[\Delta] \vdash C}\ \bullet_i E$	$\dfrac{\Delta \Rightarrow A \bullet_i B \quad \dfrac{\Gamma[(A, B)^i] \vdash C}{\Gamma[A \bullet_i B] \Rightarrow C}\ \bullet_i L}{\Gamma[\Delta] \Rightarrow C}\ Cut$
$\dfrac{\Delta \vdash \Diamond_j A \quad \Gamma[< A >_j] \vdash C}{\Gamma[\Delta] \vdash C}\ \Diamond_j E$	$\dfrac{\Delta \vdash \Diamond_j A \quad \dfrac{\Gamma[< A >_j] \Rightarrow C}{\Gamma[\Diamond_j A] \Rightarrow C}\ \Diamond_j L}{\Gamma[\Delta] \Rightarrow C}\ Cut$
$\dfrac{\Gamma \vdash \Box_j A}{< \Gamma >_j \vdash A}\ \Box_j E$	$\dfrac{\dfrac{\Gamma \Rightarrow \Box_j A}{< \Gamma >_j \Rightarrow \Diamond_j \Box_j A}\ \Diamond_j R \quad \dfrac{\dfrac{\overline{A \Rightarrow A}\ Ax}{< \Box_j A >_j \Rightarrow A}\ \Box_j L}{\Diamond_j \Box_j A \Rightarrow A}\ \Diamond_j L}{< \Gamma >_j \Rightarrow A}\ Cut$

FIG. 22: Interopérabilité entre calcul des séquents et déduction naturelle

3.4 Interface Utilisateur

La bibliothèque présentée dans la section précédente ne constitue pas un catalogue fermé de règles dérivées. En effet, elle peut très bien être enrichie par les utilisateurs souhaitant étudier les propriétés de nouvelles classes de grammaires. Une telle extension est facilitée grâce à l'utilisation d'un ensemble d'outils décrits dans ce qui suit.

3.4.1 Tactiques Spécialisées

$\mathcal{ICHARATE}$ est muni d'une banque de tactiques spécialisées définies dans le langage **Ltac** qui est fourni avec le système **Coq** [40, 41]. Ce dernier langage est assez riche pour garantir la définition de tactiques efficaces récursives ou non en se basant éventuellement sur un filtrage de données dans le but ou le contexte courant sans avoir à passer par la programmation en **OCaml**.
Le but de cette banque de tactiques est de mettre à la disposition des utilisateurs des outils de semi-automatisation efficaces qui faciliteront leurs preuves dans un formalisme multimodal donné. Ces tactiques peuvent intervenir aussi bien dans les preuves de nouvelles règles dérivées génériques que dans les dérivations syntaxiques particulières effectuées dans une grammaire précise. Elles permettent de simplifier le but courant en appliquant un ensemble de règles (de base ou dérivées), ce qui engendre une liste de sous-buts potentiellement plus simples à résoudre.
Plusieurs tactiques implémentées sont inspirées de celles de **Coq** notamment les suivantes :

- Tactiques d'introduction : elles permettent d'introduire des ressources hypothétiques dans le contexte lors de l'application de règles basées sur le raisonnement hypothétique (comme $/_i I$, $\backslash_i I$ ...). A la différence de la logique intuitionniste, les ressources introduites sont placées à un endroit bien précis du contexte (extrême droite pour $/_i I$ et extrême gauche pour $\backslash_i I$).
- Tactiques d'élimination : elles permettent d'éliminer les différents connecteurs logiques en appliquant les règles de base associées. L'application de certaines règles telles $/_i E$, $\backslash_i E$ et S_R (cf. Fig. 2.2.4), dont les variables quantifiées universellement n'apparaissent pas toutes dans les types en-tête, engendre l'apparition de variables existentielles qui seront instanciées au cours de la démonstration.

D'autres tactiques sont définies pour traiter les règles faisant intervenir un contexte linéaire de la forme $\Gamma[\ ]$, comme c'est le cas pour la tactique `zpath`[7]. Cette tactique est paramétrée par le chemin qui mène au sous-arbre où sera effectué le remplacement. Ceci permet de réécrire un contexte Γ sous la forme de $\Gamma_z[\Delta_1]$ (formalisé en **Coq** par $(z\,fill\,\Gamma_z\,\Delta_1)$) et ce en fonction du chemin passé en paramètre. L'utilisation de cette dernière tactique est illustrée dans l'extrait suivant (où Γ_z=(`zright a (tout , sage)`^{a} `zroot`)). Le chemin spécifié `r : :nil` permet de se focaliser sur le sous-arbre droit, en l'occurence pour appliquer la règle de réécriture $P\Diamond(a,\ c)$.

```
...
H : P◇(a, c)∈R
=================================
((tout , sage)ª , ((applique , constamment)ª , <(x :  □c np)>c)ª)ª ⊢R s

Coq < zpath (r::nil) (* ou zright tout court *).
```

[7] cf. fichier `Tacs/tacticsDed.v` de [12].

```
H : P◇(a, c)∈R
=================================
(zfill Γz ((applique , constamment)a , <(x :  □cnp)>c)a)a ⊢R s

Coq < struct H (*application de la regle P◇(a, c) *).

H : P◇(a, c)∈R
=================================
((tout , sage)a , ((applique ,  <(x :□cnp)>c)a , constamment)a)a ⊢R s
```

Outre les tactiques qui appliquent uniquement les règles de base, nous avons défini un ensemble de tactiques sophistiquées qui utilisent des règles dérivées (e.g., critères de non-dérivabilité) pour simplifier le but courant. Ces dernières tactiques prennent en charge l'automatisation de certains sous-buts considérés comme évidents par l'utilisateur, particulièrement la vérification des contraintes engendrées par l'application de règles dérivées conditionnées, par exemple, la vérification d'une propriété portant sur l'ensemble des règles structurelles (linéarité à gauche, linéarité, etc.) ou bien la preuve qu'une règle structurelle donnée est supportée par le système considéré. L'automatisation de certaines parties de nos preuves se fait grâce à l'utilisation d'une base de données comprenant un ensemble de lemmes utilisés par la tactique *auto* de **Coq**.

3.4.2 L'interface Pcoq et $\mathcal{ICHARATE}$

Afin de faciliter l'interaction des utilisateurs avec $\mathcal{ICHARATE}$, nous utilisons l'interface graphique **Pcoq** [4]. Cette interface permet de définir aisément différentes règles d'affichage qui améliorent la lisibilité de la syntaxe concrète grâce à l'utilisation des notations mathématiques usuelles ($\forall$, $\exists$, $\vdash$ ...).
En outre, l'interface **Pcoq** assure une interactivité conviviale avec les utilisateurs lors de la construction de preuves et ce grâce à la technique de *proof-by-pointing* [22]. Cette technique permet de simplifier le but à résoudre par l'intermédiaire de simples clics de souris. En analysant la structure du but courant et la position des clics de la souris, le système engendre automatiquement des commandes complexes qu'il renvoie à **Coq**, ceci permet de progresser dans la preuve sans avoir à connaître la syntaxe exacte des tactiques. L'algorithme de proof-by-pointing proposé dans [22] concerne les règles de la logique intuitionniste. Toutefois, ce dernier est facilement extensible à d'autres systèmes logiques tels que la logique temporelle [45].
La figure 23 montre quelques règles du proof-by-pointing en logique multimodale.

$$\frac{}{x : A \vdash \boxed{A}}\ Ax \qquad \frac{\Gamma \vdash \boxed{A} \quad \Delta \vdash B}{(\Gamma, \Delta)^i \vdash \boxed{A} \bullet_i B}\ \bullet_i I \qquad \frac{(\Gamma, x : B)^i \vdash \boxed{A}}{\Gamma \vdash \boxed{A} /_i B}\ /_i I$$

$$\frac{(x : B, \Gamma)^i \vdash \boxed{A}}{\Gamma \vdash B \backslash_i \boxed{A}}\ \backslash_i I \qquad \frac{< \Gamma >_j \vdash \boxed{A}}{\Gamma \vdash \Box_j \boxed{A}}\ \Box_i I \qquad \frac{\Gamma \vdash \boxed{A}}{< \Gamma >_j \vdash \Diamond_j \boxed{A}}\ \Diamond_j I$$

FIG. 23: Quelques règles du proof-by-pointing en logique multimodale

Les carrés encadrant certaines expressions dans les séquents conclusions délimitent les endroits pertinents pour les clics de souris. Ainsi, l'application de la règle $/_i I$ n'est déclenchée que si l'utilisateur clique sur une sous-formule se trouvant à gauche du connecteur $/_i$.
La technique de proof-by-pointing s'applique de manière récursive, ceci permet à certaines règles d'appeler l'algorithme sous-jacent sur l'une de leurs prémisses (par exemple $/_i$I, $\bullet_i$I ...). Ceci se fait en répercutant la position sélectionnée par le clic de souris sur la prémisse en question. La récursivité de la technique de proof-by-pointing est indubitablement un atout. En effet, elle permet d'enchaîner une séquence de simplifications automatiques à moindre effort.

Exemple

Nous présentons dans ce qui suit un exemple simple d'utilisation de la technique de proof-by-pointing. Il s'agit de prouver la nouvelle règle générique *box_diam* qui est valide pour toutes les classes de grammaires multimodales.

$$Lemma\ \ box\_diam\ : \forall A\ B\ i\ j\ a\ R,\ \ a : A \overset{R}{\vdash} (\Box_j \Diamond_j A \bullet_i B)/_i B$$

La démonstration de ce lemme peut être élaborée par deux clics de souris. Ainsi, en cliquant sur le type A le plus imbriqué dans la conclusion du séquent, on simplifie énormément le but courant. Suite à ce clic, un seul sous-but subsiste ; il prend la forme d'un axiome $b : B \overset{R}{\vdash} B$. Ce dernier peut être résolu en cliquant sur sa conclusion B. La figure 24 montre les différentes étapes de déductions déclenchées automatiquement grâce à ces clics.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a : A \vdash A}\ Ax}{< a : A >_j \overset{R}{\vdash} \Diamond_j A}\ \Box_j I \qquad \dfrac{}{b : B \overset{R}{\vdash} \boxed{\text{B}}}\ Ax}{a : A \overset{R}{\vdash} \Box_j \Diamond_j A \qquad\qquad b : B \overset{R}{\vdash} \boxed{\text{B}}}\ \bullet_i I}{(a : A\ ,\ b : B)^i \overset{R}{\vdash} \Box_j \Diamond_j A \bullet_i B}\ /_i I}{a : A \overset{R}{\vdash} (\Box_j \Diamond_j \boxed{\text{A}} \bullet_i B)/_i B}$$

Fig. 24: Exemple d'utilisation du proof-by-pointing

Le premier clic a pour rôle de simplifier la conclusion du séquent en faisant remonter en surface le type sélectionné A. Etant donné que la position de ce clic est située respectivement à gauche des deux opérateurs $/_i$ et $\bullet_i$ et à droite des deux connecteurs $\Diamond_j$ et $\Box_j$, l'algorithme de proof-by-pointing permet d'appliquer consécutivement les règles d'introduction de ces différents opérateurs. Le deuxième clic, quant à lui, ne fait que résoudre la seconde prémisse engendrée par la règle $\bullet_i I$.

3.4.3 Interaction et automatisation

Durant son stage de Master Recherche, Y. Devisschere a mis au point quelques techniques d'automatisation qui dispensent les non spécialistes de **Coq** de certaines tâches fastidieuses concernant la manipulation des règles structurelles [42]. Ainsi, une nouvelle commande **Add**

Rule a été définie en **OCaml** pour étendre les commandes prédéfinies de **Coq**. Cette commande prend en paramètre l'identifiant d'une règle structurelle symbolique nouvellement définie, vérifie la bonne formation de cette règle et engendre furtivement un paquet de lemmes prouvés (e.g. lemme de réécriture, lemme de bonne formation ...) sur la règle en question. Cette automatisation permet d'éviter aux utilisateurs d'écrire eux mêmes les différents énoncés en **Coq** puis les prouver en utilisant les tactiques de l'atelier. La deuxième contribution de Y. Devisschere concerne la réalisation d'un traducteur automatique basé sur les outils lex/yacc. Ce traducteur offre aux utilisateurs la possibilité de saisir les règles structurelles sous un format plus lisible qui est proche de la présentation usuelle. Le paquet de règles saisi sera systématiquement compilé en un fichier **Coq** comprenant les différentes définitions et propriétés formelles des règles symboliques associées. Ces mêmes techniques peuvent être employées pour faciliter la saisie des autres composantes d'une grammaire multimodale tel le lexique.

3.5 Conclusion

Nous avons présenté dans ce chapitre l'atelier logique $\mathcal{ICHARATE}$ dédié au traitement formel des langues naturelles. Notre atelier est destiné aux chercheurs qui désirent appréhender des formalismes logiques complexes telles les grammaires catégorielles multimodales et ce en interagissant avec l'ordinateur. Il permet à ces utilisateurs d'effectuer interactivement des dérivations syntaxiques pour des grammaires données ou de prouver de nouvelles règles et propriétés universelles portant sur des classes de grammaires afin de rendre compte de phénomènes linguistiques divers. L'interaction avec l'atelier est facilitée grâce à l'utilisation de différentes techniques avancées (tactiques, proof-by-pointing, ...).

Deuxième partie

Grammaires Linéaires Etiquetées

Grammaires Linéaires Etiquetées

4.1 Bi-directionnalité vs Non-directionnalité

4.1.1 Faiblesse des systèmes bi-directionnels

La majorité des grammaires de type logique [67, 77, 83] sont basées sur des systèmes déductifs bi-directionnels. Ces systèmes sous-structurels représentent différentes variantes de la logique linéaire non-commutative où l'implication linéaire est remplacée par deux foncteurs (/, \) garantissant ainsi la gestion[1] de l'ordre des mots par la syntaxe. Malgré leur grande élégance, ces modèles souffrent de plusieurs faiblesses d'un point de vue linguistique. En effet, le problème majeur des systèmes bi-directionnels émane de la notion de périphérie qui est intimement liée au comportement des foncteurs / et \. Ces deux connecteurs ne peuvent typer que les expressions incomplètes auxquelles il manque un argument en position périphérique (à l'extrême gauche ou droite). Par conséquent, on peut facilement rendre compte de l'extraction d'un élément périphérique comme c'est le cas pour 12a mais il s'avère beaucoup plus compliqué d'analyser des expressions ayant subi une extraction médiane telles que 12b :

(12) a. le savant que (Marie respecte _).

b. le savant que (Marie respecte _ énormément).

L'extraction médiane ne peut pas être analysée dans un système pur de Lambek. Il est nécessaire d'enrichir la logique de base pour compenser le comportement rigide des deux implications. A titre d'exemple, Moortgat propose d'ajouter des règles structurelles contrôlées telles que $L\diamond(i,\ j)$ et $P\diamond(i,\ j)$ (cf. section 2.3.1). afin d'introduire localement l'associativité et la commutativité des ressources [77, 81, 106]. L'hypothèse à extraire doit donc se déplacer de sa position médiane initiale à une position périphérique avant d'être déchargée. Outre le fait d'être relativement contre-intuitive, une telle solution complique la théorie sous-jacente et augmente considérablement la complexité des dérivations. Par exemple, la dérivation du groupe nominal 13 composé de *n* subordonnées imbriquées nécessite *3n-2* applications de la règles structurelle $L\diamond(i,\ j)$ et une application de la règle $P\diamond(i,\ j)$.

(13) le savant que (A_1 croit que A_2 pense que ... que A_n respecte _ énormément).

[1] Cette gestion peut être partielle dans les grammaires multimodales en présence de règles structurelles de commutativité locale.

Le problème de périphérie dont souffrent les systèmes bi-directionnels a également des conséquences néfastes sur le calcul des différentes représentations sémantiques, notamment dans le cas des ambiguïtés de portée (*scope ambiguities*). Pour des raisons similaires, un quantificateur généralisé dont la position est médiane ne peut pas avoir une portée sémantique large dans les systèmes de base. Par exemple, la lecture (∃∃∀) de la phrase 14, stipulant l'existence d'un unique livre prêté par un professeur à chacun des étudiants, ne peut pas se faire de manière aisée dans ce genre de systèmes.

(14) Un professeur a prêté un livre à tous les étudiants.

4.1.2 Vers des systèmes non-directionnels

Curry fut le premier logicien à souligner les lacunes des systèmes bi-directionnels. Il proposa dans [35] d'utiliser une grammaire à deux niveaux qualifiés de *tectogrammatique* et *phénogrammatique*. Curry considère ainsi que l'ordre des mots est pris en charge par le second niveau grâce à la manipulation de foncteurs qui ne sont autres que des expressions phonétiques contenant des places vides ordonnées telles que *f*='*both* $\__1$ *and* $\__2$'. Lorsqu'un foncteur se combine avec ses arguments, ces derniers occupent dans l'ordre les différents sites numérotés. A titre d'exemple, l'application du foncteur *f* aux deux arguments respectifs '*red*' et '*blue*' produit l'expression close '*both red and blue*'. Cette solution permet de contourner le problème de périphérie des systèmes bi-directionnels puisque les positions des arguments d'un foncteur peuvent être quelconques.
Le niveau tectogrammatique, quant à lui, assure la gestion des dépendances syntaxiques des expressions phonétiques et garantit la correction de leur agencement et ce grâce à l'utilisation d'une hiérarchie de catégories abstraites. Dans son papier [35], Curry assigne à titre d'exemple la catégorie (**F** *N N*) aux adjectifs '*red*' et '*blue*' et la catégorie de second ordre ($\mathbf{F}_2$ (**F** *N N*) (**F** *N N*) (**F** *N N*))[2] au foncteur *f*, chose qui permet de légitimer la combinaison de ce dernier avec les deux adjectifs précédents.
L'article [35] de Curry a inspiré plusieurs auteurs dont R. Oehrle, P. de Groote et R. Muskens qui ont proposé d'en revenir à des systèmes commutatifs afin de pallier la rigidité des systèmes bi-directionnels. Plusieurs formalismes non-directionnels ont ainsi vu le jour -tels le système *mon :LP* de R. Oehrle [88, 89], les *Grammairs Catégorielles Abstraites* de P. de Groote [37] et les *Lambda-Grammaires* de R. Muskens [87]- et ce en formalisant les idées élégantes de Curry dans le cadre du λ-calcul linéaire simplement typé. En effet, il n'est pas difficile de remarquer la grande similitude entre les foncteurs définis par Curry et les λ-termes linéaires ainsi qu'entre les catégories et les types linéaires comme l'illustre le tableau suivant :

Curry (1961)	*Logique linéaire*
both $\__1$ *and* $\__2$	$\lambda x.\ \lambda y.$ *both* $\bullet\ x \bullet$ *and* $\bullet\ y$
(**F** *N N*)	$n \multimap n$
($\mathbf{F}_2$ (**F** *N N*)(**F** *N N*)(**F** *N N*))	$(n \multimap n) \multimap (n \multimap n) \multimap n \multimap n$

Les systèmes non-directionnels partagent tous la même architecture présentée dans la figure qui suit :

[2] Le constructeur **F** désigne les foncteurs à un argument, tandis que $\mathbf{F}_2$ représente les foncteurs à deux arguments.

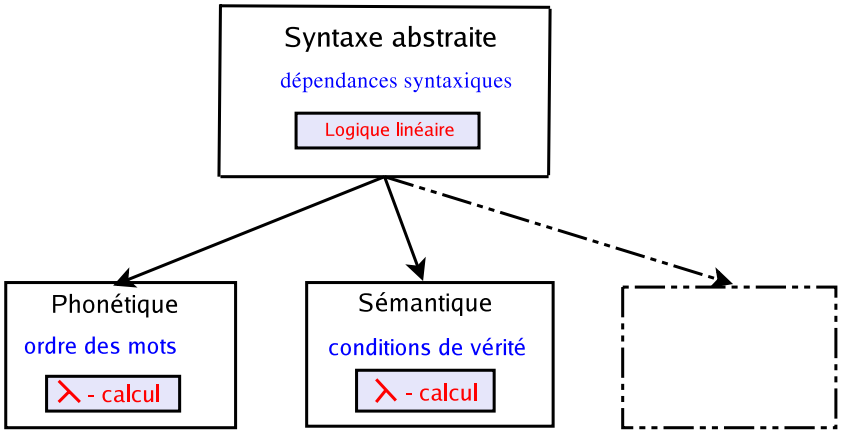


Fig. 25: Architecture à deux niveaux des systèmes non-directionnels

Ces modèles distinguent entre deux niveaux : un niveau syntaxique abstrait et un niveau concret. La couche abstraite prend en charge les dépendances grammaticales des mots ; ces dernières sont encodées moyennant des types syntaxiques abstraits dont les règles d'agencement sont rigoureusement décrites par une logique linéaire commutative. Le niveau concret, quant à lui, est multidimensionnel, il comprend un ensemble de composantes ayant divers rôles linguistiques. On se restreint dans la présente étude à la composante phonétique qui gère l'ordre des mots et la composante sémantique qui prend en charge le calcul des conditions de vérité. Les entitées de base manipulées par ces différentes composantes concrètes sont les λ-termes simplement typés. Leur combinaison est assurée par la facette calculatoire de la correspondance de Curry-Howard qui interprète chacune des règles d'inférence appliquées dans le niveau abstrait par une étape de calcul bien précise.
Les systèmes non-directionnels offrent plusieurs atouts. Au niveau de la forme, ces modèles sont plus compacts dans la mesure où ils proposent un traitement uniforme pour gérer toutes les composantes concrètes. Cette uniformité fait malheureusement défaut aux systèmes bi-directionnels présentant une asymétrie flagrante entre le calcul de la syntaxe qui est non-commutatif et le calcul de la sémantique qui est insensible à l'ordre des ressources. En outre, l'irrégularité de ces systèmes est accentuée par la distinction entre les outils employés pour calculer l'ordre des mots d'une part et la représentation sémantique d'autre part. Tandis que l'ordre des mots résulte de la partie logique de la correspondance de Curry-Howard, le calcul de la représentation sémantique est en revanche pris en charge par sa partie calculatoire.
La figure 26 illustre le contraste entre l'homogénéité des systèmes non-directionnels et l'hétérogénéité des systèmes bi-directionnels.

A part leur uniformité, les systèmes non-directionnels sont dotés d'une grande modularité. En effet, la couche abstraite est entièrement dédiée à l'encodage des *principes* communs à toutes les langues, tandis que les *paramètres* variables sont gérés par les composantes concrètes. A titre d'exemple, un verbe transitif direct est défini comme étant un verbe qui cherche à se combiner avec un groupe nominal accusatif (objet direct) et un deuxième nominatif (sujet) pour former une phrase syntaxiquement correcte. Cette spécification générique peut être formalisée par le type syntaxique abstrait $d_{acc} \multimap d_{nom} \multimap c$. Les positions respectives du sujet et de l'objet par rapport aux verbes transitifs directs étant variables, elles sont donc encodées par la composante phonétique.

Systèmes bi-directionnels	Systèmes non-directionnels
$\left[\begin{array}{lcl} mot & : & \text{type syntaxique bi-directionnel} \\ & & \text{(logique non-commutative)} \\ \lambda - terme & : & \text{type sémantique non-directionnel} \end{array}\right]$	$\left[\begin{array}{c} \Phi - terme \\ \lambda - terme \end{array}\right]$: type abstrait (calcul) (logique linéaire)

FIG. 26: Symétrie des systèmes commutatifs vs Asymétrie des modèles non-commutatifs

Ainsi, dans une langue SVO tel le français, l'ordre des mots est spécifié par un Φ-terme de la forme λx. λy. y•V•x (e.g., λx. λy. y•mange•x) plaçant le sujet en préfixe et l'objet direct en suffixe du verbe. En revanche, dans une langue VSO tel l'arabe, les verbes transitifs directs sont réalisés par des Φ-termes structurés comme suit : λx. λy. V•y•x (e.g., λx. λy. akala•y•x).
Cette modularité est fructueuse car, outre son élégance, elle nous permet d'exprimer aisément des propriétés linguistiques universelles. Par exemple, dans de tels systèmes, on peut facilement énoncer et prouver que la combinaison d'un verbe transitif direct avec un groupe nominal accusatif et un second nominatif produit bien une phrase correcte. Cette propriété abstraite est traduite par le séquent suivant :

$$\mathrm{n}_1 : d_{acc} \ , \ \mathrm{v} : d_{acc} \multimap d_{nom} \multimap c \ , \ \mathrm{n}_2 : d_{nom} \vdash c$$

Il est intéressant de noter que dans les systèmes bi-directionnels, la formalisation d'une telle propriété ne peut se faire en faisant abstraction de l'ordre des mots. Ainsi, au lieu d'avoir un seul énoncé générique, il s'avère nécessaire d'énumérer les six cas possibles associés aux différentes classes de langues (SVO, SOV, VSO, VOS, OSV, OVS) comme est illustré ci-dessous[3] :

SVO	$\mathrm{n}_1 : d_{nom} \ , \ v : (d_{nom} \backslash c)/d_{acc} \ , \ \mathrm{n}_2 : d_{acc} \vdash c$
SOV	$\mathrm{n}_1 : d_{nom} \ , \ \mathrm{n}_2 : d_{acc} \ , \ \mathrm{v} : d_{acc} \backslash (d_{nom} \backslash c) \vdash c$
VSO	$\mathrm{v} : (c/d_{acc})/d_{nom} \ , \ \mathrm{n}_1 : d_{nom} \ , \ \mathrm{n}_2 : d_{acc} \vdash c$
VOS	$\mathrm{v} : (c/d_{nom})/d_{acc} \ , \ \mathrm{n}_2 : d_{acc} \ , \ \mathrm{n}_1 : d_{nom} \vdash c$
OSV	$\mathrm{n}_2 : d_{acc} \ , \ \mathrm{n}_1 : d_{nom} \ , \ \mathrm{v} : d_{nom} \backslash (d_{acc} \backslash c) \vdash c$
OVS	$\mathrm{n}_2 : d_{acc} \ , \ \mathrm{v} : (d_{acc} \backslash c)/d_{nom} \ , \ \mathrm{n}_1 : d_{nom} \vdash c$

Les avantages qu'offrent les systèmes commutatifs, notamment leur compacité et modularité, adhèrent complétement à l'esprit de la théorie générative de Chomsky [29] et en particulier aux idées de son Programme Minimaliste [31]. Ces systèmes semblent être de bons candidats pour simuler les Grammaires Minimalistes dans un environnement logique dans le but d'expliquer avec plus de rigueur leurs différentes transformations sous-jacentes. C'est dans cette orientation que nous nous situons dans cette partie.

4.2 Définitions formelles

Nous présentons dans cette section les définitions mathématiques de différentes notions fondamentales intervenant dans cette partie. Ces définitions seront suivies de commentaires informels et d'exemples illustratifs.

[3]Pour simplifier, ces énoncés sont présentés comme des théorèmes de la logique **L** de Lambek.

4.2.1 Termes, Types et Signes

Définition 4.2.1 *Soit $\mathcal{A}$ un ensemble fini de types atomiques. L'ensemble de types syntaxiques abstraits $\mathcal{T}(\mathcal{A})$ construits à partir de $\mathcal{A}$ est défini inductivement comme ceci :*

$$\mathcal{T}(\mathcal{A}) := \mathcal{A} \mid \mathcal{T}(\mathcal{A}) \multimap \mathcal{T}(\mathcal{A}) \mid !\mathcal{T}(\mathcal{A})$$

Dans cette partie, nous considérons un ensemble d'atomes $\mathcal{A}$ plus riche que celui qu'utilisent traditionnellement les grammaires de type logique. Notre ensemble inclut également certaines catégories de base fréquemment employées dans les grammaires minimalistes (e.g., *c* : phrase correcte, *cp* : complémenteur, *d* : groupe nominal, d_{nom} : groupe nominal nominatif, d_{acc} : groupe nominal accusatif etc).
Les types syntaxiques abstraits permettent de décrire les dépendances syntaxiques des mots moyennant les deux connecteurs de la logique linéaire : l'implication linéaire $\multimap$ et l'exponentielle **!**. A titre d'exemple, on peut assigner à l'adverbe *énormément* le type abstrait $(d_{nom} \multimap c) \multimap d_{nom} \multimap c$ pour exprimer formellement le nombre et le type d'arguments qu'il recherche. L'analyse de ce type nous permet de déduire que notre adverbe cherche à se combiner avec deux arguments, le premier étant un syntagme verbal (VP) et le second un groupe nominal nominatif. Le résultat d'une telle combinaison produira une phrase syntaxiquement correcte.

Définition 4.2.2 *Soit **s** le type concret des chaines de caractères. L'ensemble des Φ-types $\mathcal{T}_\Phi$ est défini comme suit :*

$$\mathcal{T}_\Phi := s \mid \mathcal{T}_\Phi \multimap \mathcal{T}_\Phi$$

Définition 4.2.3 *Soit **e** le type des individus et **t** celui des valeurs de vérités. L'ensemble des λ-types $\mathcal{T}_\lambda$ est engendré par la grammaire suivante :*

$$\mathcal{T}_\lambda := e \mid t \mid \mathcal{T}_\lambda \to \mathcal{T}_\lambda$$

Le niveau concret du système non-directionnel que nous allons présenter comprend deux dimensions. La dimension phonétique est typée par l'ensemble concret $\mathcal{T}_\Phi$ tandis que la dimension sémantique est représentée par l'ensemble de λ-types $\mathcal{T}_\lambda$. Notons que les Φ-types sont forcément linéaires, tandis que pour des raisons linguistiques[4] les λ-types peuvent ne pas l'être, ce qui justifie l'emploi de l'implication intuitionniste dans la définition de $\mathcal{T}_\lambda$.

Notation 4.2.1 *Nous utilisons les lettres latines majuscules (e.g., A, B, ...) pour noter les types syntaxiques abstraits. Les types concrets, quant à eux, seront représentés par des lettres latines décorées par un indice (Φ pour les Φ-type et λ pour les λ-types) (e.g., A_Φ, B_λ, etc).*

Définition 4.2.4 *L'homomorphisme τ_Φ de type $\mathcal{T}(A) \to \mathcal{T}_\Phi$ est récursivement défini comme suit :*

1. $\forall\, a \in \mathcal{A}$, $\tau_\Phi(a) = \mathbf{s}$
2. $\forall\, A\ B$, $\tau_\Phi(A \multimap B) = \tau_\Phi(A) \multimap \tau_\Phi(B)$
3. $\forall\, A$, $\tau_\Phi(!A) = \tau_\Phi(A)$

[4] La sémantique lexicale d'un nombre important de constituants tels les quantificateurs est représentée par des termes non-linéaires.

Définition 4.2.5 *Soit $\tau_{\lambda at}$ une fonction assignant à chaque type syntaxique atomique un λ-type approprié. L'homomorphisme τ_λ de type $\mathcal{T}(A) \rightarrow \mathcal{T}_\lambda$ est récursivement défini comme suit :*

1. $\forall\ a \in \mathcal{A}$, $\tau_\Phi(a) = \tau_{\lambda at}(a)$
2. $\forall\ A\ B$, $\tau_\lambda(A \multimap B) = \tau_\lambda(A) \rightarrow \tau_\lambda(B)$
3. $\forall\ A$, $\tau_\lambda(!A) = \tau_\lambda(A)$

Les deux homomorphismes τ_ϕ et τ_λ transforment les types syntaxiques abstraits respectivement en des Φ-types et des λ-types adéquats tout en préservant la structure globale de leurs paramètres. Ces homomorphismes peuvent être vus comme des ponts reliant la dimension abstraite aux différentes dimensions concrètes du système. La fonction $\tau_{\lambda at}$ est similaire à celle utilisée par la sémantique de **Montague**. On suppose par exemple que $\tau_{\lambda at}(c)=t$, $\tau_{\lambda at}(n) = e \rightarrow t$ et $\tau_{\lambda at}(d_{cas}) = e$.

Définition 4.2.6 *Soient Σ_Φ un ensemble fini de constantes phonétiques, $(\Sigma_\Phi{}^*, \bullet, \epsilon)$ son monoide libre associé et $\mathcal{V}_\Phi$ un ensemble infini dénombrable de variables phonétiques typées. L'ensemble $\Lambda_\Phi(\Sigma_\Phi)$ de Φ-termes linéaires bien typés est inductivement défini comme suivant :*

1. $\forall\ u_\Phi \in \Sigma_\Phi{}^*$, $u_\Phi \in \Lambda_\Phi(\Sigma_\Phi)$ *et* u_Φ *est de type s.*
2. $\forall\ (x_\Phi : A_\Phi) \in \mathcal{V}_\Phi$, $x_\Phi \in \Lambda_\Phi(\Sigma_\Phi)$ *et* x_Φ *est de type* A_Φ.
3. $\forall\ u_\Phi\ v_\Phi \in \Lambda_\Phi(\Sigma_\Phi)$, *si* u_Φ *est de type* $A_\Phi \multimap B_\Phi$, v_Φ *est de type* A_Φ *et si en outre* u_Φ *et* v_Φ *ne partagent aucune variable libre alors* $(u_\Phi\ v_\Phi) \in \Lambda_\Phi(\Sigma_\Phi)$ *et il est de type* B_Φ.
4. $\forall\ u_\Phi \in \Lambda_\Phi(\Sigma_\Phi)$, *si* u_Φ *est de type* B_Φ *et* x_Φ *une variable de type* A_Φ *apparaissant exactement une seule fois dans le corps du terme* u_Φ *alors* $(\lambda x_\Phi.\ u_\Phi) \in \Lambda_\Phi(\Sigma_\Phi)$ *et ce terme est de type* $A_\Phi \multimap B_\Phi$.

Définition 4.2.7 *Soient Σ_λ un ensemble fini de constantes sémantiques, f_{Σ_λ} une fonction de typage dans Σ_λ et $\mathcal{V}_\lambda$ un ensemble infini dénombrable de variables sémantiques typées. L'ensemble $\Lambda_\lambda(\Sigma_\lambda)$ de λ-termes bien typés est inductivement défini comme suivant :*

1. $\forall\ u_\lambda \in \Sigma_\lambda$, $u_\lambda \in \Lambda_\lambda(\Sigma_\lambda)$ *et* u_λ *est de type* $f_{\Sigma_\lambda}(u_\lambda)$.
2. $\forall\ (x_\lambda : A_\lambda) \in \mathcal{V}_\lambda$, $x_\lambda \in \Lambda_\lambda(\Sigma_\lambda)$ *et* x_λ *est de type* A_λ.
3. $\forall\ u_\lambda\ v_\lambda \in \Lambda_\lambda(\Sigma_\lambda)$, *si* u_λ *est de type* $A_\lambda \rightarrow B_\lambda$ *et* v_λ *est de type* A_λ *alors* $(u_\lambda\ v_\lambda) \in \Lambda_\lambda(\Sigma_\lambda)$ *et il est de type* B_λ.
4. $\forall\ u_\lambda \in \Lambda_\lambda(\Sigma_\lambda)$, *si* u_λ *est de type* B_λ *et* x_λ *une variable de type* A_λ *alors* $(\lambda x_\lambda.\ u_\lambda) \in \Lambda_\lambda(\Sigma_\lambda)$ *et ce terme est de type* $A_\lambda \rightarrow B_\lambda$.

Notation 4.2.2 *Soit t un χ-terme ($\chi \in \{\Phi, \lambda\}$), si t est de type concret A alors on note t : A.*

Notation 4.2.3 *Soient u et v deux χ-termes ($\chi \in \{\Phi, \lambda\}$). La notation (u v) de la théorie des types, représentant l'application du terme u au terme v, sera parfois remplacée par la notation mathématique classique u(v), et ce dans un but de clarté.*

Notation 4.2.4 *La relation de β-réduction définie aussi bien sur les Φ-termes que les λ-termes est notée $\rightarrow^*_\beta$.*

Définition 4.2.8 *Un signe à deux dimensions est un type abstrait étiqueté par deux termes concrets noté sous la forme suivante $(l_\Phi, l_\lambda) : A$ où :*

- $A \in \mathcal{T}(\mathcal{A})$ *(e.g., A est un type abstrait).*
- $l_\Phi \in \Lambda_\Phi(\Sigma_\Phi)$ *et* $l_\Phi : \tau_\Phi(A)$.
- $l_\lambda \in \Lambda_\lambda(\Sigma_\lambda)$ *et* $l_\lambda : \tau_\lambda(A)$.

Notation 4.2.5 *Nous adoptons deux notations équivalentes pour représenter les signes à deux dimensions. Outre la notation horizontale vue précédemment, nous utiliserons parfois une notation verticale illustrée ci-contre :*

$$\left(\begin{array}{c} l_\Phi \\ l_\lambda \end{array} \right) : A$$

A titre d'exemple, le signe ci-dessous est associé à l'adverbe *énormément*. Pour plus de lisibilité, ce signe est présenté sous forme de colonne de deux dimensions :

$$\left(\begin{array}{l} \lambda P_\Phi .\ \lambda x_\Phi .\ x_\Phi \bullet P_\Phi(\epsilon) \bullet \textit{énormément} \\ \lambda P_\lambda .\ \lambda x_\lambda .\ (\textbf{Enormement}(P_\lambda))(x_\lambda) \end{array} \right) : (d_{nom} \multimap c) \multimap d_{nom} \multimap c$$

Il est aisé de vérifier la bonne formation de ce signe. En effet, le type concret de la forme phonétique et celui de la représentation sémantique sont compatibles avec le type abstrait du signe car :

1. $\lambda\ P_\Phi .\ \lambda\ x_\Phi .\ \mathrm{x}_\Phi \bullet P_\Phi(\epsilon)$•énormément : $(s \multimap s) \multimap s \multimap s = \tau_\Phi((d_{nom} \multimap c) \multimap d_{nom} \multimap c)$
2. $\lambda\ P_\lambda . \lambda\ \mathrm{x}_\lambda .\ (\textbf{Enormement}(\mathrm{P}_\lambda))(\mathrm{x}_\lambda) : (e \to t) \to e \to t = \tau_\lambda((d_{nom} \multimap c) \multimap d_{nom} \multimap c)$

Définition 4.2.9 *Un signe à deux dimensions est dit variable si et seulement si ses deux composantes phonétique et sémantique sont des variables.*

Notation 4.2.6 *Soit t un χ-terme ($\chi \in \{\Phi, \lambda\}$), si t est de type abstrait A, alors on notera par abus de notation t : A.*

L'interprétation de la notation ($t : A$) dépend fortement de la nature du type A. Si A est abstrait alors cette notation représente une sorte de signe à une seule dimension et le terme t est donc de type concret $\tau_\chi(A)$. Sinon, la notation est interprétée comme une relation de typage usuel. On suppose dans ce qui suit que cet abus de notation n'est adopté que dans le cas ou il n'entraine aucune ambiguïté.

4.3 Logique Linéaire Intuitionniste Implicative et Exponentielle

4.3.1 Système déductif

Le comportement des connecteurs logiques $\multimap$ et ! est décrit par les règles de déduction de la logique linéaire intuitionniste implicative et exponentielle (**IIELL** en abrégé) [48, 105]. Ces règles explicitent rigoureusement la façon dont se combinent les types syntaxiques abstraits. Grâce à la correspondance de **Curry-Howard**, chaque règle encapsule une partie déductive qui gère les types abstraits et une partie calculatoire qui prend en charge la combinaison des termes concrets (Φ-termes et λ-termes).
Les règles logiques opèrent sur des séquents définis comme suit :

Définition 4.3.1 *Un séquent est un jugement de type de la forme* $\Gamma \vdash (l_\Phi, l_\lambda) : A$ *où :*

– *Le contexte* Γ *est un multi-ensemble de signes variables pouvant être vide.*
– $\vdash$ *est l'opérateur de déduction.*
– *(*l_Φ*,*l_λ*) : A est un signe quelconque à deux dimensions.*

Notation 4.3.1 *Les variables représentant des contextes seront notées par des lettres grecques (e.g.,* Γ*,* Δ*, ...).*

Notation 4.3.2 *Les contextes composés des signes portant tous un type décoré par l'exponentielle seront notés par des lettres grecques préfixées par ! (e.g.,* $!\Gamma, !\Delta$*, ...).*

Les règles de la logique **IIELL** sont présentées dans la figure 27 et ce dans le style de la déduction naturelle [105]. Pour plus de lisibilité, ces règles manipulent des signes à une dimension (types abstraits décorés par une seule étiquette). Ceci étant, l'algorithme gérant la combinaison des étiquettes reste universel, il est donc commun pour les Φ-termes et les λ-termes. Les règles de la

$$\frac{}{x : A \vdash x : A}\ Ax$$

$$\frac{\Gamma\ ,\ x : A \vdash u : B}{\Gamma \vdash \lambda x.\ u : A \multimap B}\ \multimap I \qquad \frac{\Gamma \vdash u : A \multimap B \quad \Delta \vdash v : A}{\Gamma\ ,\ \Delta \vdash (u\ v) : B}\ \multimap E$$

$$\frac{!\Gamma \vdash u : A}{!\Gamma \vdash u :!A}\ !I \qquad \frac{\Gamma \vdash u :!A \quad \Delta\ ,\ v : A \vdash w : B}{\Gamma\ ,\ \Delta \vdash w[v := u] : B}\ !E$$

$$\frac{\Gamma \vdash u : B \quad \Delta \vdash v :!A}{\Gamma\ ,\ \Delta \vdash u : B}\ !E^W \qquad \frac{\Delta \vdash z :!A \quad \Gamma\ ,\ x :!A\ ,\ y :!A \vdash u : B}{\Gamma\ ,\ \Delta \vdash u[x := z, y := z] : B}\ !E^C$$

FIG. 27: Logique linéaire intuitionniste implicative et exponentielle

figure 27 sont classées en trois catégories : règles d'identité, règles logiques implicatives et règles logiques exponentielles. La première classe comprend une unique règle à savoir celle de l'axiome logique. Le second groupe est composé des règles d'élimination et d'introduction de l'implication linéaire $\multimap$. La dernière catégorie, quant à elle, contient quatre règles décrivant la gestion particulière des ressources dont le type est préfixé par l'exponentielle. Il est aisé de remarquer que la logique **IIELL** est sensible aux ressources. En effet, seules les ressources dont le type est décoré par l'exponentielle peuvent être dupliquées (par la règle d'affaiblissement $!E^W$) ou contractées (par la règle de contraction $!E^C$). Toutes les autres ressources sont utilisées une seule fois et disparaissent systématiquement du contexte après leur déchargement.
La logique **IIELL** peut servir en tant que système déductif non-directionnel dédié à l'analyse syntaxique et sémantique des langues naturelles. Les grammaires de type logique fondées sur **IIELL** seront nommées **IIELLG**. Précisons que ces grammaires représentent un enrichissement des Grammaires Catégorielles Abstraites de P. de Groote [37] et des Lambda-Grammaires de R. Muskens [87] qui sont basées uniquement sur le fragment implicatif de la logique linéaire intuitionniste. L'introduction de l'exponentielle est fructueuse dans la mesure où elle élargit le spectre des phénomènes linguistiques traités, incluant ainsi certains phénomènes non-linéaires relativement complexes tels l'ellipse et le liage d'anaphores.
Les grammaires **IIELLG** sont lexicalisées, leurs entrées lexicales coïncident avec des assertions

initiales qui explicitent le comportement phonétique et la représentation sémantique des différentes composantes linguistiques considérées. Remarquons que contrairement aux systèmes bidirectionnels, les séquents de **IIELL** sont composés de contextes locaux qui sauvegardent uniquement les variables introduites lors d'un raisonnement hypothétique au cours d'une dérivation. Il n'est plus utile de considérer les contextes comme des environnements globaux comprenant en outre les déclarations des ressources linguistiques intervenant dans la phrase à analyser puisque l'ordre des mots est délégué à la composante phonétique. C'est pour cette raison que les entrées lexicales des grammaires **IIELLG** prennent la forme d'axiomes propres (nommées aussi axiomes extra-logiques). Les arbres de dérivation de **IIELLG** auront ainsi deux sortes de feuilles, à savoir, les axiomes propres représentant les entrées lexicales associées à différentes ressources linguistiques et les axiomes logiques permettant d'introduire des hypothèses locales qui doivent être déchargées pour garantir la convergence de la dérivation.
Plus formellement, nous avons les définitions suivantes :

Définition 4.3.2 *Une entrée lexicale d'une grammaire **IIELLG** est un axiome propre qui prend la forme suivante :*

$$\vdash (a_\phi\ ,\ a_\lambda) : E$$

où $(a_\phi\ ,\ a_\lambda) : E$ *est un signe à deux dimensions.*

Définition 4.3.3 *Le lexique d'une grammaire **IIELLG** n'est autre qu'un ensemble fini d'entrées lexicales.*

La figure 28 montre un exemple de lexique simple comprenant cinq axiomes propres.

1. $\vdash \left(\begin{array}{c} \textit{savant} \\ \textbf{Savant} \end{array} \right) : n.$

2. $\vdash \left(\begin{array}{l} \lambda P_\Phi.\ \lambda m.\ \ m \bullet que \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda.\ \lambda Q_\lambda\ \lambda x.\ \ P_\lambda(x) \wedge Q_\lambda(x) \end{array} \right) : (d_{acc} \multimap c) \multimap n \multimap n.$

3. $\vdash \left(\begin{array}{c} \textit{Marie} \\ \textbf{Marie} \end{array} \right) : d_{nom}$

4. $\vdash \left(\begin{array}{l} \lambda x.\ \lambda y.\ \ y \bullet respecte \bullet x \\ \lambda x.\ \lambda y.\ \ \textbf{Respecte}(y, x) \end{array} \right) : d_{acc} \multimap d_{nom} \multimap c$

5. $\vdash \left(\begin{array}{l} \lambda P_\Phi.\ \lambda x.\ \ x \bullet P_\Phi(\epsilon) \bullet \textit{énormément} \\ \lambda P_\lambda.\ \lambda x.\ \ (\textbf{Enormement}(P_\lambda))(x)) \end{array} \right) : (d_{nom} \multimap c) \multimap d_{nom} \multimap c.$

Fig. 28: Entrées lexicales sous forme d'axiomes propres

Finalement, comme dans toute grammaire de type logique, l'analyse d'une phrase ou d'une expression dans **IIELLG** n'est autre que la preuve d'un théorème (séquent) dans la logique sous-jacente **IIELL**. Nous avons ainsi ce qui suit :

Définition 4.3.4 *Soient $\mathcal{G}$ une **IIELLG**, **at** un type atomique abstrait (**at**$\in\mathcal{A}$) et m_1, ... , m_n des constantes phonétiques ($m_i \in \Sigma_\Phi$). On dit que la séquence $m_1 m_2 ... m_n$ est de type abstrait **at** si et*

seulement si :

$$\exists\, a_\phi, a_\lambda \mid \vdash (a_\phi,\ a_\lambda) : at \ \wedge\ a_\phi \overset{\beta}{\Rightarrow} m_1 \bullet m_2 \bullet ... \bullet m_n$$

4.3.2 Exemple d'analyse

Nous présentons, dans ce qui suit, un exemple de dérivation dans la grammaire **IIELLG** définie à partir du lexique de la figure 28. On se limite dans cet exemple à la description du rôle linguistique du connecteur $\multimap$, l'intérêt de l'exponentielle sera, quant à lui, souligné dans les sections suivantes.
L'objectif est de montrer, d'abord, que le nom modifié '*savant que Marie respecte énormément*' est grammatical (de type abstrait *n*) et déduire ensuite sa représentation sémantique.
Ci-dessous, on récapitule les étapes principales de la dérivation en se limitant, dans un premier temps, à l'interface syntaxe/phonétique. Pour plus de clarté, la preuve est scindée en deux partie.

$$\dfrac{\dfrac{(3)}{\vdash Marie : d_{nom}}\ Lex \qquad \dfrac{\dfrac{(5)}{\vdash ...}\ Lex \qquad \dfrac{\dfrac{(4)}{\vdash \lambda x.\ \lambda y.\ y \bullet respecte \bullet x : d_{acc} \multimap d_{nom} \multimap c}\ Lex \qquad \dfrac{}{x_\Phi : d_{acc} \vdash x_\Phi : d_{acc}}\ Ax}{x_\Phi : d_{acc} \vdash \lambda y.\ y \bullet respecte \bullet x_\Phi : d_{nom} \multimap c}\ \multimap E}{x_\Phi : d_{acc} \vdash \lambda y.\ y \bullet respecte \bullet x_\Phi \bullet énormément : d_{nom} \multimap c}\ \multimap E}{x_\Phi : d_{acc} \vdash Marie \bullet respecte \bullet x_\Phi \bullet énormément : c}\ \multimap E$$

Dans un premier temps, une ressource hypothétique x_Φ est introduite pour occuper la position objet de la proposition relative. La progression de la dérivation partielle ci-dessus est assurée grâce à la règle $\multimap$E qui fusionne les différents constituants de base pour former des composantes complexes.

$$\dfrac{\dfrac{(1)}{\vdash savant : n}\ Lex \qquad \dfrac{\dfrac{(2)}{\vdash ...}\ Lex \qquad \dfrac{\dfrac{}{x_\Phi : d_{acc} \vdash Marie \bullet respecte \bullet x_\Phi \bullet énormément : c}\ \multimap E}{\vdash \lambda x_\Phi.\ Marie \bullet respecte \bullet x_\Phi \bullet énormément : d_{acc} \multimap c}\ \multimap I}{\vdash \lambda m.\ m \bullet que \bullet Marie \bullet respecte \bullet \epsilon \bullet énormément : n \multimap n}\ \multimap E}{\vdash savant \bullet que \bullet Marie \bullet respecte \bullet \epsilon \bullet énormément : n}\ \multimap E$$

L'hypothèse introduite est ensuite déchargée par l'intermédiaire de la règle $\multimap$I construisant ainsi une subordonnée close de type $d_{acc} \multimap c$. Cette subordonnée s'avère être l'argument attendu par le pronom relatif '*que*'. Comme le spécifie le Φ-terme associé au pronom '*que*', la combinaison de ce dernier avec sa subordonnée donne naissance à une trace ϵ qui occupe le site de l'objet extrait (la position de la variable x_Φ) et place le pronom en tête de l'expression ainsi obtenue. La dérivation s'achève en combinant la subordonnée relative avec son antécédent '*savant*'. L'application de la définition 4.3.4 suffit pour déduire la bonne formation du nom commun modifié '*savant que Marie respecte énormément*'.
Il est facile de remarquer que le problème de périphérie ne se pose point dans ce modèle. En effet, le déchargement d'une hypothèse ne dépend pas de sa position dans la composante phonétique.

Le calcul de la représentation sémantique se fait grâce à l'interface syntaxe/sémantique qui est analogue à l'interface syntaxe/phonétique. Les étapes de la dérivation restent invariables, seules les étiquettes changent comme le montrent les deux parties de la dérivation ci-dessous :

$$\dfrac{\dfrac{(3)}{\vdash \textbf{Marie} : d_{nom}}\ Lex \quad \dfrac{\dfrac{(5)}{\vdash \ldots}\ Lex \quad \dfrac{\dfrac{(4)}{\vdash \lambda x.\ \lambda y.\ \textbf{Respecte}(y,x) : d_{acc} \multimap d_{nom} \multimap c}\ Lex \quad \dfrac{}{x_\lambda : d_{acc} \vdash x_\lambda : d_{acc}}\ Ax}{x_\lambda : d_{acc} \vdash \lambda y.\ \textbf{Respecte}(y,x_\lambda) : d_{nom} \multimap c}\ \multimap E}{x_\lambda : d_{acc} \vdash \lambda x.\ (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x_\lambda)))(x) : d_{nom} \multimap c}\ \multimap E}{x_\lambda : d_{acc} \vdash (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x_\lambda)))(\textbf{Marie}) : c}\ \multimap E$$

$$\dfrac{\dfrac{(1)}{\vdash \textbf{Savant} : n}\ Lex \quad \dfrac{\dfrac{(2)}{\vdash \ldots}\ Lex \quad \dfrac{\dfrac{\dfrac{}{x_\lambda : d_{acc} \vdash (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x_\lambda)))(\textbf{Marie}) : c}\ \multimap E}{\vdash \lambda x_\lambda.\ (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x_\lambda)))(\textbf{Marie}) : d_{acc} \multimap c}\ \multimap I}{\vdash \lambda Q_\lambda\ \lambda x.\ (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x)))(\textbf{Marie}) \wedge Q_\lambda(x) : n \multimap n}\ \multimap E}{\vdash \lambda x.\ (\textbf{Enormement}(\lambda y.\ \textbf{Respecte}(y,x)))(\textbf{Marie}) \wedge \textbf{Savant}(x) : n}\ \multimap E$$

4.4 Grammaires Linéaires Étiquetées

Nous présentons dans cette section un nouveau modèle linguistique nommé *Grammaires Linéaires Etiquetées* ($\mathcal{GLE}$) dont l'objectif est de constituer un point de rencontre entre les Grammaires de Type Logique (**TLG**) et les Grammaires Minimalistes (**MG**) [31, 101, 102] afin de tirer profit de leurs avantages complémentaires. En restant fidèle à l'esprit des **TLG**, toutes les *transformations* syntaxiques dans $\mathcal{GLE}$ sont effectuées moyennant des règles d'inférence logiques, chose qui nous garantit le calcul automatique de la représentation sémantique. En outre, en suivant la philosophie des **MG**, les grammaires considérées sont fortement lexicalisées, elles sont basées sur un ensemble minimal de primitives universelles permettant ainsi de réduire la complexité de recherche d'analyses.

L'originalité du modèle $\mathcal{GLE}$ réside dans sa façon de traiter le raisonnement hypothétique. Cette dernière technique est certes nécessaire pour rendre compte des phénomènes non-locaux tels les dépendances non-bornées, toutefois, cette opération est fort coûteuse en calcul car elle augmente considérablement l'espace de recherche des analyses. Pour résoudre ce dilemme, nous avons opté pour un compromis en limitant l'accès à la technique de raisonnement hypothétique et soumettant son application à certaines contraintes. Ainsi, le formalisme $\mathcal{GLE}$ est fondé sur un sous-système de la logique **IIELL** comprenant un ensemble restreint de règles d'inférence linguistiquement pertinentes. En particulier, au lieu de considérer une règle d'axiome logique librement applicable, notre système est doté d'un ensemble fini d'axiomes contrôlés qui figurent explicitement dans le lexique. Ces axiomes sont attachés à certaines entrées susceptibles de subir un déplacement ; ils sont ainsi utilisés pour occuper leurs sites intermédiaires.

Dans cette section, nous introduirons les différentes composantes du modèle $\mathcal{GLE}$. Nous nous focaliserons à priori sur la syntaxe de ces composantes, en revanche, nous consacrerons la section 4.5 à leur interprétation détaillée dans le cadre du Programme Minimaliste.

4.4.1 Entrées lexicales

Les entrées lexicales considérées dans la section précédente (cf. définition 4.3.2) prenaient la forme d'axiomes propres simples. Le formalisme $\mathcal{GLE}$ offre la possibilité de définir des entrées enrichies pouvant être liées à des séquences finies d'axiomes nommés *hypothèses contrôlées*.

Définition 4.4.1 *Une entrée lexicale d'une grammaire $\mathcal{GLE}$ est un axiome propre obéissant à la syntaxe suivante :*

$$\vdash (a_\phi\ ,\ a_\lambda) : E \quad \multimap\!\!\!\!\cdot\ \ l_{hyps}$$

où :

- $\vdash (a_\phi\ ,\ a_\lambda) : E$ *est un axiome propre nommée tête de l'entrée (ou encore axiome propre principal de l'entrée).*
- l_{hyps} *est une séquence ordonnée d'axiomes contrôlés de longueur* $|l_{hyps}|$.

Définition 4.4.2 *Une entrée lexicale est dite libre si elle n'est attachée à aucune hypothèse contrôlée, i.e.,* $|l_{hyps}|=0$.

Définition 4.4.3 *Une entrée lexicale est dite liée si sa liste d'axiomes contrôlés est non vide, i.e.,* $|l_{hyps}|>0$.

Intuitivement, les entrées liées représentent des constituants susceptibles de se déplacer au cours d'une dérivation. Chaque axiome de la liste l_{hyps} occupe ainsi une position intermédiaire, l'entrée propre associée sera qualifiée d'*instable* jusqu'à ce qu'elle décharge simultanément toutes ses hypothèses contrôlées. A ce moment, elle retrouve sa stabilité et regagne sa position finale.
Dans un premier temps, on considère que les entrées lexicales sont spécifiées comme suit :

Définition 4.4.4 *Une entrée lexicale est bien formée ssi une des conditions ci-dessous est vérifiée :*

1. *La liste des hypothèses contrôlées est vide, i.e.,* $|l_{hyps}|=0$
2. *La liste d'axiomes contrôlés est non vide, i.e.,* $|l_{hyps}|= k > 0$. *Elle prend alors la forme suivante :* $([X_1 : H \vdash X_1' : H], ..., [X_k : H \vdash X_k' : H])$ *où* $\forall\ i\in\{1..k\}$, $X_i=(x_{\Phi i}\ ,\ x_{\lambda i})$ *et* $X_i'=(u_{\Phi i}\ ,\ x_{\lambda i})$ *avec* $x_{\Phi i} \in \mathcal{V}_\Phi$ *(Φ-variable),* $x_{\lambda i} \in \mathcal{V}_\lambda$ *(λ-variable) et* $u_{\Phi i}\in \Lambda_\Phi(\Sigma_\Phi)$. *En outre, selon la valeur de k, le type* E *de l'entrée lexicale doit être conforme à l'une des spécifications ci-dessous :*
   a. $k=1$ *et* $E=A_1\multimap ...A_n\multimap (H\multimap F)\multimap G$.
   b. $k>1$ *et* $E=A_1\multimap ...A_n\multimap (\ !H\multimap F)\multimap G$
   c. $k>1$ *et* $E=A_1\multimap ...A_n\multimap\ !H$.

On remarque que les hypothèses contrôlées rattachées à une entrée bien formée partagent toutes le même type H. Chaque hypothèse contrôlée de rang i est représentée par un axiome pouvant être logique si $x_{\Phi i}$ coïncide avec $u_{\Phi i}$ ou extra-logique dans le cas contraire. Les axiomes extra-logiques nous seront d'une grande utilité notamment lors de l'analyse de phénomènes non-linéaires faisant intervenir des variables phonétiquement prononcées tels les pronoms. Notons aussi que le type H intervient forcément comme type imbriqué dans la structure de E. En outre, il est préfixé par l'exponentielle lorsque l'entrée propre est liée à une liste l_{hyps} comprenant au moins deux axiomes. Le type final de E peut être simple lorsqu'il coïncide avec le type $!H$ ou composé

lorsqu'il est représenté par une variante de la montée de type (i.e., type raising) de H qui prend la forme suivante : $(H \multimap F) \multimap G$[5] (resp. $(!H \multimap F) \multimap G$ quand k>1).

Définition 4.4.5 *Une entrée liée bien formée est dite non-linéaire simple si elle est conforme à la spécification [2.c] de la définition 4.4.4.*

Définition 4.4.6 *Une entrée liée bien formée est dite linéaire composée si elle respecte la condition [2.a] de la définition 4.4.4 et non-linéaire composée si elle est conforme à la spécification [2.b] de la définition 4.4.4.*

Définition 4.4.7 *Une entrée liée bien formée est dite non-linéaire si elle est non-linéaire simple ou non-linéaire composée.*

Remarque 4.4.1 *Il n'est pas difficile de remarquer que les entrées lexicales linéaires simples vérifiant la spécification suivante (k=1 et $E=A_1 \multimap ... A_n \multimap H$) sont exclues du système puisqu'elles sont mal-formées. Ces dernières peuvent être simulées efficacement par des entrées libres comme nous montrerons ultérieurement.*

Exemple 4.4.1 Illustrons maintenant les définitions précédentes sur des entrées concrètes. A titre d'exemple, on peut modifier l'entrée lexicale du pronom relatif '*que*' présentée dans la figure 28 en la reliant à un axiome contrôlé de type d_{acc} comme suit :

$$\vdash \left(\begin{array}{l} \lambda P_\Phi .\ \lambda m.\ \ m \bullet que \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda .\ \lambda Q_\lambda .\ \lambda x.\ \ P_\lambda(x) \wedge Q_\lambda(x) \end{array} \right) : (d_{acc} \multimap c) \multimap n \multimap n \quad \exists \ ([(x_\Phi, x_\lambda) : d_{acc} \vdash (x_\Phi, x_\lambda) : d_{acc}]).$$

Cette nouvelle entrée est une entrée liée composée car elle est conforme à la spécification [2.a] de la définition 4.4.4. L'axiome contrôlé occupera dans ce cas la position objet de la proposition relative. Le pronom relatif ne pourra atteindre sa position finale et préfixer sa subordonnée qu'en déchargeant cette hypothèse.
Le second exemple ci-dessous présente une entrée bien formée liée à deux hypothèses contrôlées :

$$\vdash \left(\begin{array}{l} \lambda P_\Phi .\ Marie \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda .\ P_\lambda(\textbf{Marie}) \end{array} \right) : (d_{nom} \multimap c) \multimap c \quad \exists \ ([(x_{\Phi 1}, x_{\lambda 1}) : d_{nom} \vdash (x_{\Phi 1}, x_{\lambda 1}) : d_{nom}] ; \\ [(x_{\Phi 2}, x_{\lambda 2}) : d_{nom} \vdash (elle, x_{\lambda 2}) : d_{nom}]).$$

Cette fois ci, notre entrée est conforme à la spécification [2.b] de la définition 4.4.4. Une telle entrée est liée à un premier axiome logique et un second extra-logique permettant d'introduire le pronom personnel '*elle*' en tant que variable prononcée (de forme phonétique non-vide). On verra ultérieurement comment on peut se servir d'une telle entrée composée pour lier l'anaphore '*elle*' à son antécédent '*Marie*' dans une phrase telle que '*Marie croit qu'elle respecte le savant*'.

Finallement, on définit le lexique d'une $\mathcal{GLE}$ comme suit :

Définition 4.4.8 *Un lexique d'une Grammaire Linéaire Etiquetée est un ensemble fini d'entrées lexicales bien formées.*

[5] La montée de type de H représente usuellement le type d'ordre supérieur $(H \multimap F) \multimap F$. Dans notre cas, nous utilisons une forme plus générale $(H \multimap F) \multimap G$ qui offre aux deux types G et F la possibilité d'être distincts.

4.4.2 Noyau de $\mathcal{GLE}$

Dans la littérature, plusieurs auteurs ont opté pour l'utilisation de systèmes logiques partiels afin de décrire certains phénomènes de la langue naturelle ou formaliser une théorie linguistique donnée. Citons à titre d'exemple les grammaires **CCG** de Steedman [103] et leurs extensions multimodales introduites par Baldridge [16], qui manipulent des règles de calcul dérivées de la logique de Lambek ou de sa version multimodale. D'autre part, les Grammaires Catégorielles Minimalistes définies par Lecomte et Retoré [70, 69] sont basées sur un système déductif minimal comprenant uniquement les règles d'élimination des connecteurs logiques utilisés. Dans les différents cas de figure, seules les règles les plus utiles à la théorie linguistique traitée sont conservées. En passant d'un modèle initial à un sous-fragment bien déterminé, on sacrifie indubitablement la complétude de la logique sous-jacente. Toutefois, ces systèmes bénéficient d'une plus grande simplicité et peuvent être facilement appréhendés par les linguistes. En outre, ils sont computationnellement plus intéressants étant donné que leur complexité d'analyse est raisonnable : par exemple, l'analyse dans les grammaires **CCG** et leur extension multimodale se fait en temps polynomial [16]. Le système $\mathcal{GLE}$ décrit ci-dessous adopte cette même philosophie.

Soit $\mathcal{LEX}$ un lexique d'une Grammaire Linéaire Etiquetée.
Le noyau de $\mathcal{GLE}$ comprend un ensemble de règles logiques présentées dans la figure 29. Pour plus de lisibilité, ces règles font intervenir des signes à une seule dimension. Précisons, en outre, que certaines de ces règles admettent des pré-conditions (†, ‡, ⋆, ♭), ces dernières seront explicitées ultérieurement.

Axiomes :

$$\frac{(\vdash u : A \multimap l_{hyps}) \in \mathcal{LEX}}{\vdash u : A}\ Lex \qquad \frac{(\vdash u : A \multimap l_{hyps}) \in \mathcal{LEX} \quad (x : C \vdash x' : C) \in l_{hyps}}{x : C \vdash x' : C}\ Ax$$

Règles logiques :

$$\frac{\Gamma \vdash u : A \multimap B \quad \Delta \vdash v : A}{\Gamma, \Delta \vdash (u\ v) : B} \multimap E$$

$$\frac{\Gamma \vdash u :!A \quad \Delta, x :!A \vdash v : B}{\Gamma, \Delta \vdash v[x := u] : B} \multimap IE_s\ (\dagger) \qquad \frac{\Gamma \vdash w : (A \multimap B) \multimap C \quad \Delta, x : A \vdash v : B}{\Gamma, \Delta \vdash (w\ (\lambda x.\ v)) : C} \multimap IE_c\ (\ddagger)$$

Règles structurelles :

$$\frac{\Gamma, x : A \vdash w : B}{\Gamma, x :!A \vdash w : B}\ !L\ (\star) \qquad \frac{\Gamma, x :!A, y :!A \vdash w : B}{\Gamma, z :!A \vdash w[x := z, y := z] : B}\ !L^c\ (\flat)$$

FIG. 29: Règles d'inférence des Grammaires Linéaires Etiquetées

Afin de réduire la complexité du système, l'application de la technique de raisonnement hypothétique est limitée dans $\mathcal{GLE}$. Ainsi, la règle d'axiome logique librement accessible dans **IIELL** est remplacée par un ensemble fini d'axiomes consommables explicitement mentionnés par les entrées lexicales liées. D'autre part, la règle d'introduction $\multimap$I est utilisée de manière masquée par les deux nouvelles règles hybrides $\multimap IE_s$ et $\multimap IE_c$.

Le formalisme $\mathcal{GLE}$ est fondé sur un sous-système de la logique **IIELL**. En effet, chacune des nouvelles règles logiques ou structurelles figurant dans le noyau de $\mathcal{GLE}$ représente une règle dérivée de **IIELL**[6] comme le montre la figure 30. A titre d'exemple, les deux règles logiques $\multimap IE_s$ et $\multimap IE_c$ encapsulent deux étapes de déduction : une étape d'introduction et une autre d'élimination du connecteur $\multimap$, d'où leur identifiant commun IE.

Règle $\multimap$ IE$_s$	**Règle $\multimap$ IE$_c$**
$\dfrac{\Gamma \vdash' u : !A \quad \dfrac{\Delta\ ,\ x : !A \vdash' v : B}{\Delta \vdash' \lambda x.\ v : !A \multimap B}\ \multimap I}{\Gamma\ ,\ \Delta \vdash' v[x := u] : B}\ \multimap E$	$\dfrac{\Gamma \vdash' w : (A \multimap B) \multimap C \quad \dfrac{\Delta\ ,\ x : A \vdash' v : B}{\Delta \vdash' \lambda x.\ v : A \multimap B}\ \multimap I}{\Gamma\ ,\ \Delta \vdash' (w\ (\lambda x.\ v)) : C}\ \multimap E$

Règle !L	**Règle !L^c**
$\dfrac{\Gamma\ ,\ x : A \vdash' w : B \quad \dfrac{}{x : !A \vdash' x : !A}\ Ax}{\Gamma\ ,\ x : !A \vdash' w : B}\ !E$	$\dfrac{\Gamma\ ,\ x : !A\ ,\ y : !A \vdash' w : B \quad \dfrac{}{z : !A \vdash' z : !A}\ Ax}{\Gamma\ ,\ z : !A \vdash' w[x := z, y := z] : B}\ !E^c$

Fig. 30: Preuve que les nouvelles règles de $\mathcal{GLE}$ sont des règles dérivées de **IIELL**

Dans ce qui suit, nous présentons une description détaillée de chacune des règles d'inférence de la figure 29 en soulignant les contraintes éventuelles liées à leur application.

Axiomes

Tous les axiomes d'une Grammaire Linéaire Etiquetée sont explicitement donnés par son lexique. Ainsi, la première règle de déduction *Lex* nous permet d'extraire la tête des entrées lexicales. Cette dernière prend la forme d'un axiome propre. Une entrée lexicale sera qualifiée d'*instable* dès qu'elle subit l'application de la règle *Lex* visant l'extraction de son axiome propre principal. Ladite entrée ne retrouvera sa stabilité qu'en déchargeant au même temps toutes ses hypothèses contrôlées.
D'autre part, la seconde règle *Ax* sert à récupérer les différents axiomes contrôlés attachés à une entrée lexicale afin de les insérer dans la dérivation.
Pour plus de lisibilité, nous omettrons d'expliciter les premisses de ces règles lors d'une dérivation. On se contentera de mentionner l'identifiant de l'entrée lexicale représentant l'origine des axiomes propres et des hypothèses contrôlées.

Règles structurelles

Les règles structurelles supportées par $\mathcal{GLE}$ introduisent une gestion particulière des hypothèses contrôlées dans le cas où leur entrée propre est liée à au moins deux axiomes (i.e., entrée non-linéaire). Le but est de pouvoir regrouper toutes les hypothèses contrôlées de type A liées à une entrée lexicale donnée dans une ressource de type $!A$ afin de pouvoir les décharger simultanément. Une telle opération est nécessaire pour faire partager la même sémantique entre les différentes

[6] Pour éviter toute ambiguïté, l'opérateur de déduction dans **IIELL** est représenté par $\vdash$'

positions intermédiaires occupées par l'entrée en question.
Cette procédure se fait en deux étapes. Dans un premier temps, la règle *!L* est appliquée afin de décorer par l'exponentielle, le type *A* d'une hypothèse contrôlée. L'application de cette règle dépend de la satisfaction de la pré-condition ($\star$) qui stipule que la liste l_{hyps} d'où provient l'hypothèse contrôlée *x* comprend au moins deux éléments (i.e., $|l_{hyps}| > 1$). Le rôle de la pré-condition ($\star$) est de prévenir le recours inutile à la règle *!L* notamment dans le cas où l'entrée lexicale est attachée à un seul axiome contrôlé (i.e., $|l_{hyps}|=1$).
La seconde étape consiste à appliquer la règle $!L^c$ afin de rassembler toutes les hypothèses contrôlées dans une unique hypothèse. Notons que cette règle de contraction n'est pas linéaire car elle réduit le nombre de ressources. Toutefois, cette contraction se fait de manière locale et contrainte car elle est limitée aux ressources dont le type est décoré par l'exponentielle. Par conséquent, l'application de cette règle doit forcément être précédée par celle de *!L*. Finalement, la pré-condition (♭) permet de contraindre davantage l'utilisation de $!L^c$ en limitant la contraction aux seules hypothèses rattachées à la même entrée lexicale.

Règles logiques

Les trois règles logiques de $\mathcal{GLE}$ introduisent diverses techniques servant à combiner deux branches de dérivations partielles en un seul arbre global.
La première règle $\multimap$E n'est autre que la variante linéaire du modus-ponens, son rôle linguistique a été largement décrit dans la section 4.3.
L'utilisation de la règle $\multimap\text{IE}_c$ est, quant à elle, motivée par les entrées lexicales *composées* qui sont encore instables, d'où l'indice *c* qui orne l'identifiant de la règle. Cette règle leur permet de décharger leurs hypothèses contrôlées et regagner leur position d'équilibre en intégrant la dérivation globale. L'application de la règle $\multimap\text{IE}_c$ est liée à la satisfaction de la pré-condition (‡) qui spécifie un ensemble de contraintes synthétisées ci-dessous :

a. La dérivation $\Gamma \vdash w : (A \multimap B) \multimap C$ se sert de la tête (i.e., l'axiome propre) d'une entrée lexicale instable *e*. Ceci revient à dire que l'axiome propre principal de *e* constitue une feuille de la dérivation de la première prémisse et qu'en outre, l'entrée *e* n'a toujours pas déchargé ses hypothèses contrôlées. Le type abstrait $(A \multimap B) \multimap C$ n'est autre que le type final de la tête de l'entrée *e*. A titre d'exemple, si *e* respecte la spécification [2.b] de la définition 4.4.4, *A* sera de la forme *!H*, où *H* coïncide avec le type commun des hypothèses contrôlées rattachées à *e*.

b. La dérivation de la seconde prémisse fait intervenir tous les axiomes contrôlés attachés à l'entrée *e*. En outre, ces axiomes sont introduits dans la dérivation suivant un ordre bien précis, à savoir en parcourant la liste l_{hyps} de droite à gauche. Plus précisément, ceci revient à dire que l'hypothèse la plus à droite (resp. à gauche) de l_{hyps} doit être la plus (resp. moins) profonde dans l'arbre de dérivation. Ainsi, si l'entrée *e* est conforme à la spécification [2.b] de la définition 4.4.4, tous les axiomes $[X_k : H \vdash X_k' : H] \ldots [X_1 : H \vdash X_1' : H]$ doivent être insérés comme feuilles de la dérivation du séquent $\Delta\ ,\ x :!H \vdash v : F$ et ce dans cet ordre respectif.

Remarquons que la règle $\multimap\text{IE}_c$ est une règle générique qui décrit de manière uniforme le processus de déchargement des hypothèses contrôlées liées à une entrée lexicale composée qu'elle soit linéaire (i.e., rattachée à une seule hypothèse) ou non-linéaire (i.e., rattachée à plus d'une hypothèse).

Dans le premier cas, la variable A prendra la valeur H, représentant le type de l'unique hypothèse contrôlée qui s'apprête à être déchargée. Le schéma général ci-dessous résume l'application de la règle $-\circ\text{IE}_c$ en présence d'une entrée composée linéaire e. Un tel schéma servira à décrire certains phénomènes linguistiques linéaires tel le qu-mouvement (cf. section 4.5).

$$\dfrac{\dfrac{\dfrac{(e)}{\vdash a : A_1 \multimap \ldots \multimap A_n \multimap (H \multimap F) \multimap G}\,Lex}{\vdots}}{\Gamma \vdash w : (H \multimap F) \multimap G} \qquad \dfrac{\dfrac{(e)}{x : H \vdash x' : H}\,Ax}{\vdots}\ \Delta\ ,\ x : H \vdash v : F}{\Gamma\ ,\ \Delta \vdash (w\ (\lambda x.\ v)) : G}\ -\circ IE_c$$

En revanche, dans le cas où l'entrée e qui cherche à se stabiliser est non-linéaire, la variable A prendra la valeur $!H$, où H est le type commun des axiomes contrôlés qui seront déchargés. Dans de telles circonstances, notre dérivation se fera en suivant les étapes récapitulées ci-dessous. Ce schéma factorise l'ébauche générale qui nous sera utile pour analyser certains phénomènes non-linéaires comme le liage d'anaphores (cf. section 5.2.2).

$$\dfrac{\dfrac{\dfrac{(e)}{\vdash a : A_1 \multimap \ldots \multimap A_n \multimap (!H \multimap F) \multimap G}\,Lex}{\vdots}}{\Gamma \vdash w : (!H \multimap F) \multimap G} \qquad \dfrac{\dfrac{(e)}{x_1 : H \vdash x'_1 : H}\,Ax \quad \dfrac{(e)}{x_k : H \vdash x'_k : H}\,Ax}{\vdots \qquad\qquad \vdots}\ \Delta\ ,\ x :!H \vdash v : F}{\Gamma\ ,\ \Delta \vdash (w\ (\lambda x.\ v)) : G}\ -\circ IE_c$$

Finalement, la règle $-\circ\text{IE}_s$ se charge d'abstraire les hypothèses contrôlées associées aux entrées lexicales non-linéaires *simples*, ce qui justifie la présence de l'indice s comme suffixe du nom de la règle. La pré-condition (†) qui gère le bon usage de cette règle est analogue à la contrainte (‡). En effet, elle stipule que la dérivation de la première prémisse fait intervenir la tête d'une entrée non-linéaire simple e qui n'a pas encore déchargé ses hypothèses contrôlées. En outre, la dérivation de la seconde prémisse nécessite l'introduction de tous les axiomes contrôlés liés à e et ce en respectant l'ordre pré-établi cité précédemment (i.e., en parcourant l_{hyps} de droite à gauche).
Le fragment ci-dessous schématise les circonstances propices qui déclenchent l'application de la règle $-\circ\text{IE}_s$. Ce schéma de dérivation nous permettra de rendre compte de phénomènes linguistiques non-linéaires nécessitant un mouvement de tête telle l'ellipse (cf. section 5.3.2).

$$\dfrac{\dfrac{\dfrac{(e)}{\vdash a : A_1 \multimap \ldots \multimap A_n \multimap !H}\,Lex}{\vdots}}{\Gamma \vdash u :!H} \qquad \dfrac{\dfrac{(e)}{x_1 : H \vdash x'_1 : H}\,Ax \quad \dfrac{(e)}{x_k : H \vdash x'_k : H}\,Ax}{\vdots \qquad\qquad \vdots}\ \Delta\ ,\ x :!H \vdash v : F}{\Gamma\ ,\ \Delta \vdash v[x := u] : F}\ -\circ IE_s$$

Il est intéressant de noter que la règle $-\circ\text{IE}_s$ est un cas particulier de la règle de coupure présentée ci-dessous :

$$\dfrac{\Gamma \vdash u : A \quad \Delta\ ,\ x : A \vdash v : B}{\Gamma\ ,\ \Delta \vdash v[x := u] : B}\ Cut$$

En effet, on se restreint uniquement au cas ou la formule de coupure A est enveloppée par l'exponentielle. Cette restriction est complétement cohérente avec l'asymétrie présente dans la spécification des entrées lexicales (cf. définition 4.4.4) qui exige des entrées simples d'être non-linéaires. Il est totalement inutile de définir des entrées linéaires simples et faire abstraire leur unique hypothèse contrôlée moyennant la règle *Cut* qui représente la généralisation de $\multimap$IE$_s$, puisqu'une telle opération, schématisée ci-dessous, encapsule un détour redondant qu'il serait souhaitable d'éviter.

$$\dfrac{\dfrac{\dfrac{(e)}{\vdash a : A_1 \multimap ... \multimap A_n \multimap H}\ Lex}{\vdots}}{\Gamma \vdash u : H} \qquad \dfrac{\dfrac{\dfrac{(e)}{x : H \vdash x : H}\ Ax}{\vdots}}{\Delta\ ,\ x : H \vdash v : F} \qquad \dfrac{\Gamma\ ,\ \Delta \vdash v[x := u] : F}{}\ Cut$$

Ce fragment de dérivation peut très bien être réduit en remplaçant la feuille axiome (x :H $\vdash$ x :H) par la branche de dérivation de ($\Gamma \vdash$ u :H), chose qui nous permet de se passer de l'hypothèse contrôlée. Par conséquent, toute entrée lexicale linéaire simple peut être simulée de manière plus efficace par une entrée libre comme le montre le schéma suivant :

Entrée linéaire simple (*mal formée*)	*Entrée libre associée* (*bien formée*)
$\vdash (a_\Phi, a_\lambda) : A_1 \multimap ... \multimap A_n \multimap B \multimap [X : B \vdash X : B]$	$\vdash (a_\Phi, a_\lambda) : A_1 \multimap ... \multimap A_n \multimap B$

Le système $\mathcal{GLE}$ permet de filtrer un sous-ensemble propre des dérivations de **IIELLG**. Il ne conserve que les déductions qui sont linguistiquement pertinentes notamment dans le cadre du Programme Minimaliste. De nombreuses dérivations permises dans **IIELLG** sont systématiquement rejetées dans $\mathcal{GLE}$ car elles violent certains critères d'économie. Ainsi, à titre d'exemple, les déductions *non-normales* qui utilisent inutilement le raisonnement hypothétique. ajoutant ainsi des détours facultatifs à l'analyse sont totalement exclues. Ces détours sont produits lorsque la règle $\multimap$I est directement suivie de l'application de $\multimap$E comme le montre le schéma de la figure 31.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash' x : A}\ Ax}{\vdots}}{\Gamma\ ,\ x : A \vdash' w : B}}{\dfrac{\Gamma \vdash' \lambda x.\ w : A \multimap B}{\Gamma\ ,\ \Delta \vdash' w[x := u] : B}\ \multimap I} \qquad \dfrac{\Delta \vdash' u : A}{}\ \multimap E$$

FIG. 31: Détour redondant pouvant intervenir dans les dérivations de **IIELLG**

Une telle séquence d'étapes de déduction ne peut avoir lieu dans le système $\mathcal{GLE}$ comme le stipule la proposition qui suit.

Proposition 4.4.1 *Toutes les dérivations de $\mathcal{GLE}$ sont en forme normale.*

Démonstration Pour prouver que toute dérivation de $\mathcal{GLE}$ est forcément normale il est suffisant de montrer qu'elle ne peut contenir un détour semblable à celui de la figure 31. Etant donné l'absence de la règle d'axiome logique librement accessible ainsi que celle d'introduction du connecteur $-\circ$, on ne peut construire explicitement la séquence d'étapes d'inférence formant le détour en question. En outre, une simple inspection des pas élémentaires encapsulés par les règles de déduction de $\mathcal{GLE}$ (cf. Fig. 30) suffit pour remarquer que ledit détour n'est masqué par aucune de ces règles. Il est intéressant de souligner, encore une fois, que la mal formation des entrées linéaires simples est nécessaire pour se garantir la normalisation des dérivations dans le formalisme $\mathcal{GLE}$.

□

Exemple 4.4.2 Il est temps de présenter un exemple de dérivation dans $\mathcal{GLE}$. On considère le lexique défini ci-dessous ; pour des raisons de compacité, les composantes des entrées lexicales sont regroupées dans un tableau.

Id	Φ-terme	λ-terme	Type abstrait	Axiomes Contrôlés
e_1	$\lambda m.\ \lambda P_\Phi.$ un•m•$P_\Phi(\epsilon)$	$\lambda P_\lambda.\ \lambda Q_\lambda.\ \exists z.\ P_\lambda(z) \wedge Q_\lambda(z)$	$n -\circ (d_{nom} -\circ c) -\circ c$	$[(z_\Phi, z_\lambda) : d_{nom} \vdash (z_\Phi, z_\lambda) : d_{nom}]$
e_2	$\lambda m.\ \lambda P_\Phi.\ P_\Phi$(un•m)	$\lambda P_\lambda.\ Q_\lambda.\ \exists y.\ P_\lambda(y) \wedge \lambda Q_\lambda(y)$	$n -\circ (d_{acc} -\circ c) -\circ c$	$[(y_\Phi, y_\lambda) : d_{acc} \vdash (y_\Phi, y_\lambda) : d_{acc}]$
e_3	$\lambda m.\ \lambda P_\Phi.\ P_\Phi$(tous•les•m)	$\lambda P_\lambda.\ \lambda Q_\lambda.\ \forall x.\ P_\lambda(x) \Rightarrow Q_\lambda(x)$	$n -\circ (d_{dat} -\circ c) -\circ c$	$[(x_\Phi, x_\lambda) : d_{dat} \vdash (x_\Phi, x_\lambda) : d_{dat}]$
e_4	professeur	$\lambda x_\lambda.$ **Professeur**(x_λ)	n	$\emptyset$
e_5	livre	$\lambda x_\lambda.$ **Livre**(x_λ)	n	$\emptyset$
e_6	étudiants	$\lambda x_\lambda.$ **Etudiant**(x_λ)	n	$\emptyset$
e_7	$\lambda x_\Phi.\ \lambda y_\Phi.\ \lambda z_\Phi.$ z_Φ•a•prêté•y_Φ•à•x_Φ	$\lambda x_\lambda.\ \lambda y_\lambda.\ \lambda z_\lambda.$ **Prêter**$_{Past}(z_\lambda, y_\lambda, x_\lambda)$	$d_{dat} -\circ d_{acc} -\circ$ $d_{nom} -\circ c$	$\emptyset$

FIG. 32: Lexique d'une grammaire $\mathcal{GLE}$

L'objectif de cet exemple est d'illustrer la capacité des grammaires $\mathcal{GLE}$ de traiter, de manière aisée, les ambiguïtés de portée. Nous nous focaliserons sur l'analyse de la phrase '*Un professeur a prêté un livre à tous les étudiants*' dans la grammaire $\mathcal{GLE}$ définie par le lexique de la figure 32. Nous montrerons ainsi que toutes les lectures sémantiques de cette phrase sont possibles notamment la lecture $\exists\exists\forall$ qui ne peut être obtenue dans les systèmes bi-directionnels simples (cf. section 4.1.1). Pour plus de lisibilité, l'interface syntaxe/phonétique et l'interface syntaxe/sémantique sont présentées séparément. En outre, la dérivation est scindée en plusieurs sous-blocs qui sont commentés au fur et à mesure.

Interface Syntaxe/Phonétique (Bloc $\mathcal{B}_{\Phi 1}$)

$$\dfrac{\dfrac{(e_7)}{\vdash \lambda x_\Phi.\ \lambda y_\Phi.\ \lambda z_\Phi.\ z_\Phi \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet x_\Phi : d_{dat} -\circ d_{acc} -\circ d_{nom} -\circ c}\ Lex \quad \dfrac{(e_3)}{x_\Phi : d_{dat} \vdash x_\Phi : d_{dat}}\ Ax}{x_\Phi : d_{dat} \vdash \lambda y_\Phi.\ \lambda z_\Phi.\ z_\Phi \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet x_\Phi : d_{acc} -\circ d_{nom} -\circ c}\ -\circ E$$

Interface Syntaxe/Sémantique (Bloc $\mathcal{B}_{\lambda 1}$)

$$\dfrac{\dfrac{(e_7)}{\vdash \lambda x_\lambda.\ \lambda y_\lambda.\ \lambda z_\lambda.\ \textbf{Prêter}_{Past}(z_\lambda,\ y_\lambda,\ x_\lambda) : d_{dat} -\circ d_{acc} -\circ d_{nom} -\circ c}\ Lex \quad \dfrac{(e_3)}{x_\lambda : d_{dat} \vdash x_\lambda : d_{dat}}\ Ax}{x_\lambda : d_{dat} \vdash \lambda y_\lambda.\ \lambda z_\lambda.\ \textbf{Prêter}_{Past}(z_\lambda,\ y_\lambda,\ x_\lambda) : d_{acc} -\circ d_{nom} -\circ c}\ -\circ E$$

Dans un premier temps, la règle *Lex* est utilisée pour extraire l'axiome propre principal de l'entrée e_7 associée au verbe ditransitif '*a prêté à*'. L'axiome propre obtenu est ensuite combiné

avec l'hypothèse contrôlée liée à l'entrée e_3. Cette hypothèse étant de type abstrait d_{dat} (groupe nominal portant le cas datif), elle occupe ainsi la position d'objet indirect du verbe ditransitif considéré.

Bloc $\mathcal{B}_{\Phi 2}$

$$\dfrac{\dfrac{\overline{\mathcal{B}_{\Phi 1}} \qquad \dfrac{(e_2)}{y_\Phi : d_{acc} \vdash y_\Phi : d_{acc}}\, Ax}{y_\Phi : d_{acc}, x_\lambda : d_{dat} \vdash \lambda z_\Phi.\ z_\Phi \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet x_\Phi : d_{nom} \multimap c}\, \multimap E \quad \dfrac{(e_1)}{z_\Phi : d_{nom} \vdash z_\Phi : d_{nom}}\, Ax}{z_\Phi : d_{nom}, y_\Phi : d_{acc}, x_\lambda : d_{dat} \vdash z_\Phi \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet x_\Phi : c}\, \multimap E$$

Bloc $\mathcal{B}_{\lambda 2}$

$$\dfrac{\dfrac{\overline{\mathcal{B}_{\lambda 1}} \qquad \dfrac{(e_2)}{y_\lambda : d_{acc} \vdash y_\lambda : d_{acc}}\, Ax}{y_\lambda : d_{acc}, x_\lambda : d_{dat} \vdash \lambda z_\lambda.\ \mathbf{Prêter}_{Past}(z_\lambda,\ y_\lambda,\ x_\lambda) : d_{nom} \multimap c}\, \multimap E \quad \dfrac{(e_1)}{z_\lambda : d_{nom} \vdash z_\lambda : d_{nom}}\, Ax}{z_\lambda : d_{nom}, y_\lambda : d_{acc}, x_\lambda : d_{dat} \vdash \mathbf{Prêter}_{Past}(z_\lambda,\ y_\lambda,\ x_\lambda) : c}\, \multimap E$$

La dérivation se poursuit en combinant le bloc $\mathcal{B}_1$ ($\mathcal{B}_{\Phi 1}$, $\mathcal{B}_{\lambda 1}$) avec l'hypothèse contrôlée attachée à l'entrée e_2 qui occupe alors la position d'objet direct. Le constituant engendré fusionne ensuite avec l'axiome contrôlé lié à l'entrée e_1 qui prend la place initiale du sujet. On obtient par conséquent une sorte de phrase paramétrée par les trois variables introduites qui sont prêtes à être déchargées par leurs entrées propres respectives. Les contextes du formalisme $\mathcal{GLE}$ étant des multi-ensembles, l'ordre de déchargement de ces hypothèses peut être quelconque. C'est grâce à cette flexibilité que les ambiguïtés de portée sont analysées de manière aisée. D'une façon générale, l'ordre de déchargement des hypothèses est inversement proportionnel à la largeur de portée de leurs quantificateurs généralisés associés. De ce fait, plus une variable est abstraite très tôt, plus la portée de son entrée liée est étroite et inversement. En appliquant ce principe, on est capable de calculer, sans aucune peine, les quatre sens possibles de notre phrase. Nous illustrons dans ce qui suit les étapes de déduction visant la computation de la lecture $\exists\exists\forall$.

Bloc $\mathcal{B}_{\Phi 3}$

$$\dfrac{\dfrac{\dfrac{(e_3)}{\vdash \lambda m.\ \lambda P_\Phi.\ P_\Phi(tous \bullet les \bullet m) : n \multimap (d_{dat} \multimap c) \multimap c}\, Lex \quad \dfrac{(e_6)}{étudiants : n}\, Lex}{\vdash \lambda P_\Phi.\ P_\Phi(tous \bullet les \bullet étudiants) : (d_{dat} \multimap c) \multimap c}\, \multimap E \quad \overline{\mathcal{B}_{\Phi 2}}}{z_\Phi : d_{nom}, y_\Phi : d_{acc} \vdash z_\Phi \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet tous \bullet les \bullet étudiants : c}\, \multimap IE_c$$

Bloc $\mathcal{B}_{\lambda 3}$

$$\dfrac{\dfrac{\dfrac{(e_3)}{\vdash \lambda P_\lambda.\ \lambda Q_\lambda.\ \forall x.\ P_\lambda(x) \Rightarrow Q_\lambda(x) : n \multimap (d_{dat} \multimap c) \multimap c}\, Lex \quad \dfrac{(e_6)}{\lambda x_\lambda.\ \mathbf{Etudiant}(x_\lambda) : n}\, Lex}{\vdash \lambda Q_\lambda.\ \forall x.\ \mathbf{Etudiant}(x) \Rightarrow Q_\lambda(x) : (d_{dat} \multimap c) \multimap c}\, \multimap E \quad \overline{\mathcal{B}_{\lambda 2}}}{z_\lambda : d_{nom}, y_\lambda : d_{acc} \vdash \forall x.\ \mathbf{Etudiant}(x) \Rightarrow \mathbf{Prêter}_{Past}(z_\lambda,\ y_\lambda,\ x) : c}\, \multimap IE_c$$

Afin d'obtenir la représentation sémantique $\exists\exists\forall$, il faut nécessairement commencer par abstraire la variable contrôlée z (resp. z_Φ, z_λ) liée au quantificateur généralisé '*tous les étudiants*'. Ceci se fait en appliquant la règle hybride IE_c qui permet à cet objet indirect effectif de se

stabiliser en retrouvant sa position finale. Notons que la contrainte ‡ est systématiquement vérifiée dans notre cas puisque ledit quantificateur généralisé est construit à partir de l'entrée linéaire composée e_3 qui est effectivement attachée à l'hypothèse z.

Bloc $\mathcal{B}_{\Phi 4}$

$$\dfrac{\dfrac{\dfrac{(e_1)}{\vdash \lambda m.\ \lambda P_\Phi.\ un \bullet m \bullet P_\Phi(\epsilon) : n \multimap (d_{nom} \multimap c) \multimap c}\ Lex \quad \dfrac{(e_4)}{\vdash professeur : n}\ Lex}{\vdash \lambda P_\Phi.\ un \bullet professeur \bullet P_\Phi(\epsilon) : (d_{nom} \multimap c) \multimap c}\ \multimap E \quad \overline{\mathcal{B}_{\Phi 3}}}{y_\Phi : d_{acc} \vdash un \bullet professeur \bullet a \bullet prêté \bullet y_\Phi \bullet à \bullet tous \bullet les \bullet étudiants : c}\ \multimap IE_c$$

Bloc $\mathcal{B}_{\lambda 4}$

$$\dfrac{\dfrac{\dfrac{(e_1)}{\vdash \lambda P_\lambda.\ \lambda Q_\lambda.\ \exists z.\ P_\lambda(z) \wedge Q_\lambda(z) : n \multimap (d_{nom} \multimap c) \multimap c}\ Lex \quad \dfrac{(e_4)}{\vdash \lambda x_\lambda.\ \textbf{Professeur}(x_\lambda) : n}\ Lex}{\vdash \lambda Q_\lambda.\ \exists z.\ \textbf{Professeur}(z) \wedge Q_\lambda(z) : (d_{nom} \multimap c) \multimap c}\ \multimap E \quad \overline{\mathcal{B}_{\lambda 3}}}{y_\lambda : d_{acc} \vdash \exists z.\ \textbf{Professeur}(z) \wedge \forall x.\ \textbf{Etudiant}(x) \Rightarrow \textbf{Prêter}_{Past}(z,\ y_\lambda,\ x) : c}\ \multimap IE_c$$

Les étapes de déduction intervenant dans le bloc 4 sont analogues à celles du bloc 3. Cette fois ci, c'est l'hypothèse contrôlée x (x_Φ, x_λ) qui est déchargée par le sujet effectif prenant la forme du quantificateur généralisé '*un professeur*'. En suivant cet ordre de déchargement, la portée sémantique du sujet s'avère être plus large que celle de l'objet indirect.

Bloc $\mathcal{B}_{\Phi 5}$

$$\dfrac{\dfrac{\dfrac{(e_2)}{\vdash \lambda m.\ \lambda P_\Phi.\ P_\Phi(un \bullet m) : n \multimap (d_{acc} \multimap c) \multimap c}\ Lex \quad \dfrac{(e_5)}{\vdash livre : n}\ Lex}{\vdash \lambda P_\Phi.\ P_\Phi(un \bullet livre) : (d_{acc} \multimap c) \multimap c}\ \multimap E \quad \overline{\mathcal{B}_{\Phi 4}}}{\vdash un \bullet professeur \bullet a \bullet prêté \bullet un \bullet livre \bullet à \bullet tous \bullet les \bullet étudiants : c}\ \multimap IE_c$$

Bloc $\mathcal{B}_{\lambda 5}$

$$\dfrac{\dfrac{\dfrac{(e_2)}{\vdash \lambda P_\lambda.\ \lambda Q_\lambda.\ \exists y.\ P_\lambda(y) \wedge Q_\lambda(y) : n \multimap (d_{acc} \multimap c) \multimap c}\ Lex \quad \dfrac{(e_4)}{\vdash \lambda x_\lambda.\ \textbf{Livre}(x_\lambda) : n}\ Lex}{\vdash \lambda Q_\lambda.\ \exists y.\ \textbf{Livre}(y) \wedge Q_\lambda(y) : (d_{acc} \multimap c) \multimap c}\ \multimap E \quad \overline{\mathcal{B}_{\lambda 4}}}{\vdash \exists y.\ \textbf{Livre}(y) \wedge \exists z.\ \textbf{Professeur}(z) \wedge (\forall x.\ \textbf{Etudiant}(x) \Rightarrow \textbf{Prêter}_{Past}(z,\ y,\ x)) : c}\ \multimap IE_c$$

La dérivation s'achève par l'abstraction de la dernière hypothèse contrôlée y (y_Φ, y_λ) associée au complément d'objet direct '*un livre*'. La lecture sémantique souhaitée est ainsi obtenue. Notons que le même sens peut être calculé en permutant l'ordre de déchargement des hypothèses x et y. En effet, l'abstraction de la variable z, en premier, garantit à elle seule l'étroitesse de la portée de l'objet indirect '*tous les étudiants*' par rapport aux portées respectives du sujet '*un professeur*' et de l'objet '*un livre*'.

Représentation arborescente des dérivations

Pour plus de lisibilité, les dérivations peuvent être modélisées de manière plus compacte sous une forme arborescente similaire à la présentation à la Prawitz [93] sauf qu'elle est inversée (i.e., la tête en haut et les feuilles en bas). Dans ce nouveau format, les déductions sont traduites en des arbres d'analyse descendante dont la racine coïncide avec l'énoncé final et les feuilles ne sont autres que les conclusions (parties droites) des axiomes (logiques ou extra-logiques) considérés. Les hypothèses contrôlées sont placées entre crochets et co-indexées avec la tête de leur entrée associée. Chaque étape d'inférence logique de $\mathcal{GLE}$ admet une unique représentation arborescente comme le montre la figure ci-après :

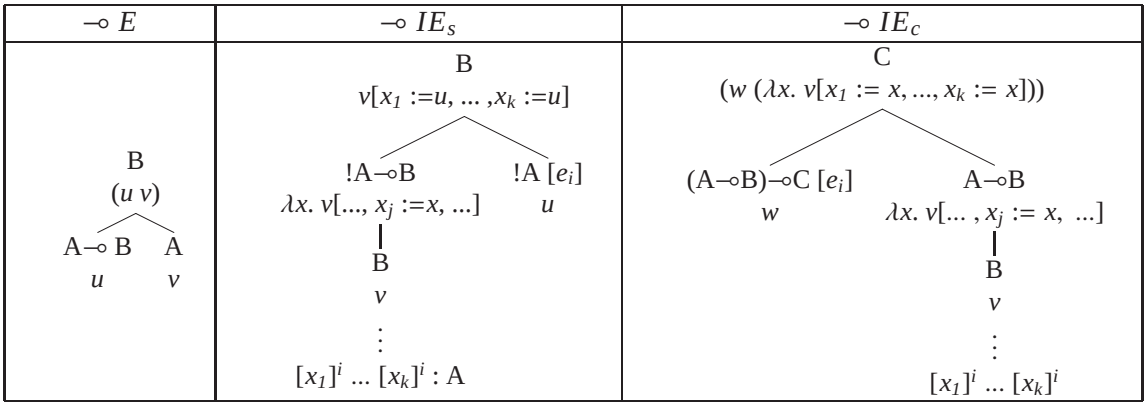


FIG. 33: Représentation arborescente des règles logiques de $\mathcal{GLE}$

Cette présentation permet de souligner l'homogéneité entre les règles logiques $\multimap E$, $\multimap IE_s$ et $\multimap IE_c$. En effet, elles permettent toutes les trois de combiner deux sous branches moyennant l'application fonctionnelle. La branche gauche représente le foncteur et celle de droite est son argument.

La représentation arborescente d'une analyse se construit progressivement à partir des feuilles jusqu'à la racine et ce en fusionnant les arbres partiels formés à l'étape antérieure sans leur apporter la moindre modification. Pour cette raison, l'application des règles structurelles ne peut apparaitre explicitement dans cette modélisation étant donné que ces règles modifient l'état initial des hypothèses en altérant leur type, leur identifiant ou leur nombre. Ainsi, l'utilisation des règles structurelles sera reflétée de manière masquée sur la représentation arborescente. Puisque le rôle des règles structurelles est intimement lié à l'application des deux règles hybrides $\multimap IE_s$ et $\multimap IE_c$, le plus naturel est d'incorporer leurs fonctionnalités dans les traductions arborescentes respectives des règles $\multimap IE_s$ et $\multimap IE_c$ (cf. Fig. 33). A titre d'exemple, l'application de la règle $\multimap IE_s$ fait intervenir k hypothèses contrôlées $x_1 ... x_k$ (où k>1) de type abstrait A. Avant d'être déchargées, ces hypothèses partagent un même identifiant x et leur type commun s'enveloppe par l'exponentielle. Ces deux dernières étapes prises en charge usuellement par les règles structurelles se trouvent encapsulées dans la représentation arborescente associée à la règle $\multimap IE_s$.

La figure 34 illustre l'utilisation de la modélisation arborescente en présentant l'analyse sémantique de la phrase '*Un professeur a prêté un livre à tous les étudiants*' qui produit la lecture $\exists_p \forall \exists_l$[7], où l'objet indirect a une portée plus large que celle de l'objet direct.

[7]Cette lecture signifie qu'il existe un certain professeur qui a prêté à chaque étudiant un livre particulier.

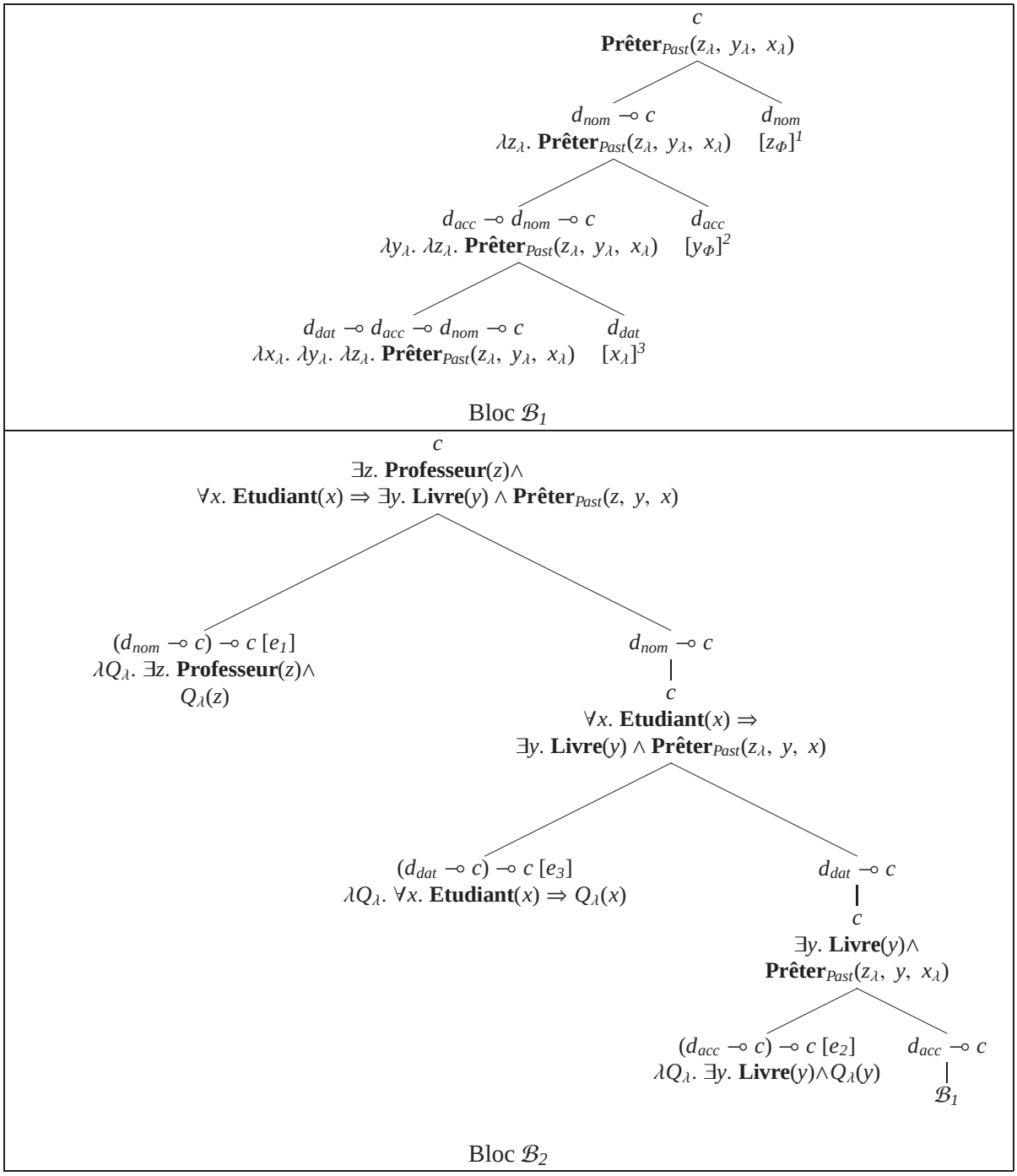


FIG. 34: Analyse sémantique des ambiguïtés de portées sous une forme arborescente

4.4.3 Formalisation de $\mathcal{GLE}$ en Coq

Le système **Coq** a été utilisé pour tester et corriger différentes versions antérieures des grammaires $\mathcal{GLE}$. L'intéraction avec l'assistant de preuves nous a permis de mieux comprendre le fonctionnement de ces grammaires et déboucher par conséquent sur une version stable du formalisme [8]. L'expérience acquise lors de la conception et la réalisation de l'atelier $\mathcal{ICHARATE}$ dédié aux grammaires multimodales (cf. chapitre 3) nous a été d'un grand apport dans la formalisation des grammaires $\mathcal{GLE}$ en **Coq**. Le plongement profond du système déductif sous-jacent à $\mathcal{GLE}$ est effectué par l'intermédiaire d'un type inductif dépendant `deduction` représentant l'opérateur de dérivabilité $\vdash$. Les jugements de typage ainsi formalisés prennent la forme raffinée suivante : $\Gamma \vdash (l_\Phi , l_\lambda) : A; E$ où :

- Le contexte Γ est un multi-ensemble fini composé de signes variables (implémenté en **Coq** par une liste). Chaque variable est marquée par un indice explicitant l'entrée lexicale à laquelle l'hypothèse est attachée (e.g., ${x_\Phi}^{\uparrow^i}$: hypothèse liée à l'entrée e_i).
- $(l_\Phi , l_\lambda) : A$ est un signe à deux dimensions.
- E est un multi-ensemble fini qui comprend les identifiants des entrées lexicales qui sont instables, à savoir les entrées liées utilisées dans la dérivation et dont les hypothèses contrôlées ne sont pas encore déchargées.

Les constructeurs du type inductif `deduction` coïncident avec les énoncés formels des règles logiques de $\mathcal{GLE}$ présentés ci-après :

Axiomes :

$$\frac{e_i = (\vdash u : A \multimap l_{hyps})}{\vdash u : A;\ \ si\ l_{hyps} = ()\ alors\ \emptyset\ sinon\ \{e_i\}}\ Lex \qquad \frac{e_i = (\vdash \_ \multimap l_{hyps}) \quad l_{hyps}[j] = (x : C \vdash x' : C)}{x^{\uparrow^i} : C \ \vdash\ x' : C;\ \emptyset}\ Ax$$

Règles logiques :

$$\frac{\Gamma \vdash u :!A;\ \{e_i\} \cup E_1 \quad \Delta , x^{\uparrow^i} :!A \vdash v : B;\ E_2}{\Gamma , \Delta \vdash v[x := u] : B;\ \ E_1 \cup E_2} \multimap IE_s\ (\dagger) \qquad \frac{\Gamma \vdash u : A \multimap B;\ E_1 \quad \Delta \vdash v : A;\ E_2}{\Gamma , \Delta \vdash (u\ v) : B;\ E_1 \cup E_2} \multimap E$$

$$\frac{\Gamma \vdash w : (A \multimap B) \multimap C; \{e_i\} \cup E_1 \quad \Delta , x^{\uparrow^i} : A \vdash v : B;\ E_2}{\Gamma , \Delta \vdash (w\ (\lambda x.\ v)) : B;\ E_1 \cup E_2} \multimap IE_c\ (\dagger)$$

Règles structurelles :

$$\frac{\Delta , x^{\uparrow^i} : A \vdash w : B;\ E}{\Delta , x^{\uparrow^i} :!A \vdash w : B;\ E}\ !L \qquad \frac{\Delta , x^{\uparrow^i} :!A , y^{\uparrow^i} :!A \vdash w : B;\ E}{\Delta , z^{\uparrow^i} :!A \vdash w[x := z, y := z] : B;\ E}\ !L^c$$

Grâce à cette formalisation, les deux pré-conditions († et ‡) inhérentes à l'application des règles $\multimap IE_s$ et $\multimap IE_c$ sont exprimées de la même manière (i.e., il faut que les hypothèses contrôlées liées à l'entrée e_i soient introduites suivant un ordre précis, en parcourant la liste l_{hyps} de droite à gauche). Afin de formaliser cette pré-condition commune (†), nous supposons que chaque hypothèse $x^{\uparrow^i}$ encapsule une sorte d'*historique* utilisé pour mémoriser certaines données pertinentes

[8] Les fichiers **Coq** de cette formalisation peuvent être téléchargés de l'URL suivante : www.labri.fr/~anoun/Icharate.

(e.g., profondeur d'une hypothèse dans la dérivation). Ce paramètre supplémentaire n'a aucun impact sur le système déductif. Il est introduit pour assurer plus d'efficacité dans la construction des dérivations en facilitant la vérification de la pré-condition †.
L'historique d'une hypothèse $x^{\uparrow^i}$ est encodé par un ensemble de paires d'entiers naturels. Le premier nombre de chaque paire représente le rang de l'hypothèse contrôlée impliquée dans la liste l_{hyps}, tandis que le second nombre n'est autre que la profondeur de cette hypothèse dans la dérivation ascendante (i.e., le nombre d'étapes de déduction entre la phase d'introduction de l'hypothèse et l'état courant de la dérivation).

Chaque étape de déduction met à jour l'historique de toutes les variables incluses dans le contexte. A titre d'exemple, la règle Ax assure l'introduction d'une hypothèse contrôlée spécifique portant le rang j (i.e., la $j^{ième}$ hypothèse de l_{hyps}) et initialise son historique avec la paire $(j,\ 0)$. Nous montrons ci-dessous deux exemples de règles dotées de leur gestion explicite des historiques. La notation $x^{\uparrow^i}\lfloor\sigma\rfloor$ est employée pour associer l'historique σ à l'hypothèse $x^{\uparrow^i}$. L'opération d'incrémentation des historiques est notée par $()^{++}$, où $\{(k_1,\ d_1)\,;\ldots;(k_i,\ d_i)\,;\ldots;(k_n,\ d_n)\}^{++} = \{(k_1,\ d_1{+}1)\,;\ldots;\ (k_i,\ d_i{+}1)\,;\ldots;(k_n,\ d_n{+}1)\}$.

$$\frac{e_i = (\vdash\ \_\ \dashv\ l_{hyps}) \quad l_{hyps}[j] = (x : C \vdash x' : C)}{x^{\uparrow^i}\lfloor\{(j,0)\}\rfloor : C\ \vdash\ x' : C;\ \emptyset}\ Ax \qquad \frac{\Delta, x^{\uparrow^i}\lfloor\sigma_1\rfloor :!B, y^{\uparrow^i}\lfloor\sigma_2\rfloor :!B \vdash w : A;\ E}{\Delta, z^{\uparrow^i}\lfloor{\sigma_1}^{++} \cup {\sigma_2}^{++}\rfloor :!B \vdash w[x := z, y := z] : A;\ E}\ !L^c$$

Par conséquent, la pré-condition † est formellement énoncée comme suit (où la fonction `hist` est utilisée pour renvoyer l'historique associée à une variable) :

$$\dagger \quad \Leftrightarrow \quad \begin{cases} \forall k,\ 1 \leq k \leq |\ l_{hyps}\ | \Rightarrow\ \exists!\ d \mid (k,\ d) \in hist(x^{\uparrow^i}) \\ \forall (k,\ d) \in hist(x^{\uparrow^i})\ \forall (k',\ d') \in hist(x^{\uparrow^i}),\quad k < k' \ \Rightarrow\ d < d' \end{cases}$$

La pré-condition † nécessite la vérification de deux contraintes. La première stipule que toutes les hypothèses contrôlées liées à l'entrée e_i doivent être introduites une seule et unique fois dans la dérivation intermédiaire. La seconde sous-condition explicite l'ordre d'introduction de ces hypothèses qui est inversement proportionnel à leur rang dans la liste l_{hyps} : plus le rang d'une hypothèse est élevé, plus sa profondeur dans la dérivation est grande.

4.5 $\mathcal{GLE}$ et le Programme Minimaliste

Malgré leurs différences apparentes, les Grammaires Minimalistes (**MG** en abrégé, cf. Annexe C pour un petit tour d'horizon) et les Grammaires de Type Logique partagent la même philosophie centrale. En effet, ils sont tous les deux lexicalisés et manipulent des ressources consommables. Néanmoins, alors que le second formalisme est rigoureusement défini, le premier souffre toujours de certaines lacunes cruciales d'un point de vue computationnel. Ces lacunes sont dûes à l'imprécision du calcul de formes logiques émanant de l'absence d'une procédure explicite qui permet de construire la représentation sémantique d'un énoncé à partir de son arbre syntaxique. Il est donc intéressant d'étudier la possibilité de plonger les primitives des (**MG**) dans un système déductif compact afin de tirer profit de la connexion facile avec la sémantique de Montague. Plusieurs travaux ont été réalisés pour établir une certaine interopérabilité entre ces deux formalismes. Citons à titre d'exemple l'approche de Lecomte et Retoré [70, 69] qui est basée sur une variante minimale

du calcul de Lambek ainsi que celle de Vermaat [106] qui utilise comme méta-langage les Grammaires Catégorielles Multimodales. Toutefois, ces deux systèmes présentent certaines failles qui sont incompatibles avec l'objectif visé. En effet, dans le premier formalisme, la correspondance entre la syntaxe et la sémantique est totalement rompue étant donné que la logique sous-jacente comprend uniquement les règles d'élimination des connecteurs utilisés (/, \, $\otimes$). D'autre part, le modèle défini par Vermaat est loin d'être minimal car il est basé sur la logique bi-directionnelle multimodale enrichie par un paquetage spécifique de règles structurelles.
Le système $\mathcal{GLE}$ que nous proposons répond plus aux besoins requis : c'est un système logique compact qui contient un nombre restreint de règles conservant totalement l'homomorphisme entre la syntaxe et la sémantique. Nous illustrerons dans ce qui suit, la façon dont $\mathcal{GLE}$ permet d'interpréter les transformations des (**MG**) dans un environnement logique. Nous soulignerons au fur et à mesure les points de concordance et de divergence entre les mécanismes transformationnels et leurs correspondants logiques.

4.5.1 Simulation logique de la fusion

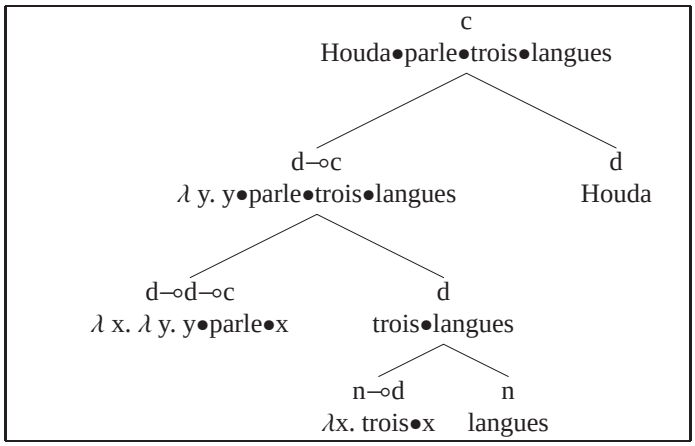
Comme c'est le cas pour les contributions antérieures [70, 69, 106], nous simulons l'opération de fusion moyennant la règle de modus-ponens ($\multimap$E). Une expression portant le trait = a correspond donc à une entité de type fonctionnel $a \multimap d$ qui cherche à se combiner avec son paramètre de type a. Cette opération syntaxique est reflétée par une application de Φ-termes dans la dimension phonétique. Dans notre modèle, aucune contrainte sur l'ordre respectif entre le foncteur et son argument n'est à priori exigée. Ainsi, la définition logique de la fusion s'avère être plus flexible et plus générale. Elle permet, entre autres, de gérer facilement la combinaison d'un verbe di-transitif avec ses deux objets en les plaçant directement à sa droite (en position complément). Une telle flexibilité réduit considérablement la complexité de la dérivation proposée dans le **MP** qui nécessite le recours à l'opération de déplacement pour garantir l'obtention de l'ordre recherché [94].
Il est toutefois possible de traduire exactement l'opération de fusion des **MG** dans notre système logique. Il suffit d'introduire certaines restrictions sur la structure phonétique des entrées lexicales considérées. A titre d'exemple, le terme phonétique d'une entrée libre représentant le mot m devrait obéir à la syntaxe suivante :

$$\lambda x_1.\lambda x_2.\ ...\ \lambda x_n.\ \ x_n \bullet x_{n-1} \bullet ... \bullet x_2 \bullet m \bullet x_1\ \ :\ \ a_1 \multimap a_2 \multimap ... \multimap a_n \multimap a_{n+1}$$

où chacun des types a_i est atomique. Cette spécification impose de placer le premier argument d'un foncteur à sa droite (en position complément) et tous les arguments qui suivent à sa gauche (en position spécifieur). A la lumière de cette restriction, l'entrée (1) ci-dessous est parfaitement acceptée alors que (1') est systématiquement rejetée.

$$(1)\ \vdash \lambda x.\lambda y.\ y \bullet \mathit{parle} \bullet x : d_{acc} \multimap d_{nom} \multimap c$$
$$(1')\ \vdash \lambda x.\lambda y.\ x \bullet \mathit{parle} \bullet y : d_{nom} \multimap d_{acc} \multimap c$$

Exemple 4.5.1 La séquence de fusions sous-jacente à la dérivation de la phrase '*Houda parle trois langues*' (cf. Annexe C, Fig. 63) peut être traduite dans un premier temps comme suit :



Cette première traduction logique (et systématique) de la dérivation présentée dans la figure 63 ne pose aucun problème pour l'interface syntaxe/phonétique. Toutefois, pour se garantir le lien avec la sémantique, les groupes nominaux (de type *d*) sont considérés comme des expressions quantifiées [47, 26] portant le type abstrait élevé $(d \multimap c) \multimap c$. L'utilisation de ce type d'ordre supérieur sera commenté et illustré suite à la description de la définition logique du déplacement.

4.5.2 Simulation logique du déplacement

La simulation de l'opération de mouvement dans un cadre logique s'avère être plus ardue et moins directe que celle de la fusion. Dans la littérature, plusieurs procédures ont été proposées. Ainsi, dans [70, 69], Lecomte et Retoré traduisent le déplacement par la règle rudimentaire d'élimination du produit permettant de décharger simultanément deux hypothèses qui occupent respectivement la position initiale et finale de l'élément déplacé. Vermaat, en revanche, considère que l'opération de mouvement est décomposée en plusieurs opérations basiques à savoir la réécriture structurelle, le raisonnement hypothétique et la fusion [106].
Notre approche partage plusieurs points communs avec celle de Vermaat. En effet, nous traduisons l'opération de déplacement par la règle hybride $\multimap IE_c$ (cf. Fig. 29) qui encapsule une étape de raisonnement hypothétique engendrant l'abstraction des hypothèses contrôlées. Cette première étape est suivie par une phase de fusion avec le constituant déplacé qui finit par intégrer sa position d'équilibre finale. Toutefois, l'utilisation d'un système non-directionnel nous permet d'économiser le recours au raisonnement structurel[9], produisant par conséquent un encodage logique plus efficace de l'opération de déplacement.
L'application de la règle $\multimap IE_c$ qui reflète notre définition logique du mouvement est déclenchée par les entrées lexicales composées. Chaque entrée représente une chaine dont la tête est l'élément déplacé et les hypothèses contrôlées sont des éléments transitoires qui remplissent ses sites intermédiaires. D'après la définition 4.4.4, toute chaine doit obéir à la syntaxe suivante :

$$\vdash (a_\Phi\ ,\ a_\lambda) : A_1 \multimap ... \multimap A_n \multimap (<! > H \multimap F) \multimap G \quad \multimap \exists \quad l_{hyps}$$

où $<!>H \in \{H, !H\}$ et H est le type abstrait des hypothèses contrôlées de l_{hyps}. Cette spécification générique stipule que l'entrée propre doit d'abord fusionner avec *n* expressions de types respectifs

[9] Le raisonnement structurel utilisé par Vermaat vise le déplacement des hypothèses de leurs positions initiales à leurs positions périphériques finales afin d'être déchargées.

$A_1...A_n$ avant de retrouver sa place finale. L'objectif de ces fusions est de construire une projection maximale qui constitue, d'après les contraintes de Koopman et Szabolcsi [65], l'unique composant ayant le droit de se déplacer.
Afin de rendre compte des différentes conditions liées à l'application du mouvement dans le **MP**, il est nécessaire de restreindre davantage la spécification des chaines illustrée ci-dessus. Dans ce qui suit, nous présentons les contraintes primordiales devant être introduites pour traduire certains principes de base inhérents au **MP** :

a. Les types $A_1...A_n$ sont atomiques : cette condition permet de traduire avec fidélité le rôle de l'opération de fusion qui exige des foncteurs de sélectionner uniquement des expressions complètes (dont le type est atomique).

b. Le type H est atomique : cette contrainte impose au type des hypothèses contrôlées d'être de base. En effet, chaque hypothèse représente une position intermédiaire du constituant déplacé qui est forcément une expression complète (c'est une projection maximale d'après [65]).

c. Le type F est atomique : cette condition traduit la priorité de la fusion par rapport au déplacement (i.e., *merge over move principle*) [31]. En effet le déplacement s'avère être une opération plus coûteuse ne devant s'appliquer qu'en cas de nécessité. Suivant ce principe, la structure intermédiaire (de type F) qui s'apprête à participer dans l'opération de mouvement doit porter un type atomique lui évitant de s'impliquer dans une nouvelle fusion.

Notons, en revanche, que le type abstrait G peut être quelconque. Ce type est celui de la structure finale obtenue suite à la combinaison de la projection maximale déplacée avec un constituant de type $<!H> \multimap F$. Un tel constituant résulte de l'abstraction simultanée des hypothèses contrôlées dans la structure intermédiaire (de type F). A la différence des travaux de Vermaat, nous n'imposons pas à F et G d'être confondus. En effet, il existe des exemples où le type de la structure finale est totalement différent de celui de la structure intermédiaire comme c'est le cas pour les pronoms relatifs qui transforment leur subordonnée en un adjectif postnominal (cf. ex. 4.4.1).
La règle logique raffinée $\multimap IE_c$ formalise uniquement les mécanismes syntaxiques abstraits sous-jacents au déplacement. C'est le rôle de la composante phonétique d'expliciter la position finale de l'élément déplacé et de distinguer ainsi entre les deux variantes visible et furtive de cette opération. A titre d'exemple, dans le cas où G est atomique, la représentation phonétique de la tête de l'entrée composée devrait avoir une des formes ci-après. Il est intéressant de souligner que les contraintes syntaxiques énoncées dans les points [a., b., c.] ci-dessus sont nécessaires pour garantir la bonne formation de ces Φ-termes.

Mouvement visible	Mouvement furtif
$a_\Phi = \lambda x_1. \ ... \ \lambda x_n. \ \lambda P. \ \ x_n \bullet ... \bullet x_2 \bullet m \bullet x_1 \bullet P(\epsilon)$: $s \multimap ... \multimap s \multimap (s \multimap s) \multimap s$	$a_\Phi = \lambda x_1. \ ... \ \lambda x_n. \ \lambda P. \ \ P(x_n \bullet ... \bullet x_2 \bullet m \bullet x_1)$: $s \multimap ... \multimap s \multimap (s \multimap s) \multimap s$

Ainsi, dans le cas d'un mouvement visible, la projection maximale est placée en position frontale et tous les sites intermédiaires occupés par des hypothèses phonétiques non-prononcées sont remplacés par une trace ϵ. Par contre, lorsque le mouvement est furtif, la projection maximale construite est utilisée pour remplir les positions transitoires qui n'ont pas de réalisation phonétique.

Exemple 4.5.2 Illustrons maintenant le fonctionnement du plongement logique du déplacement sur l'exemple interrogatif '*Quelle langue Sam peut parler ?*' (cf. Annexe C, Fig. 64 pour une

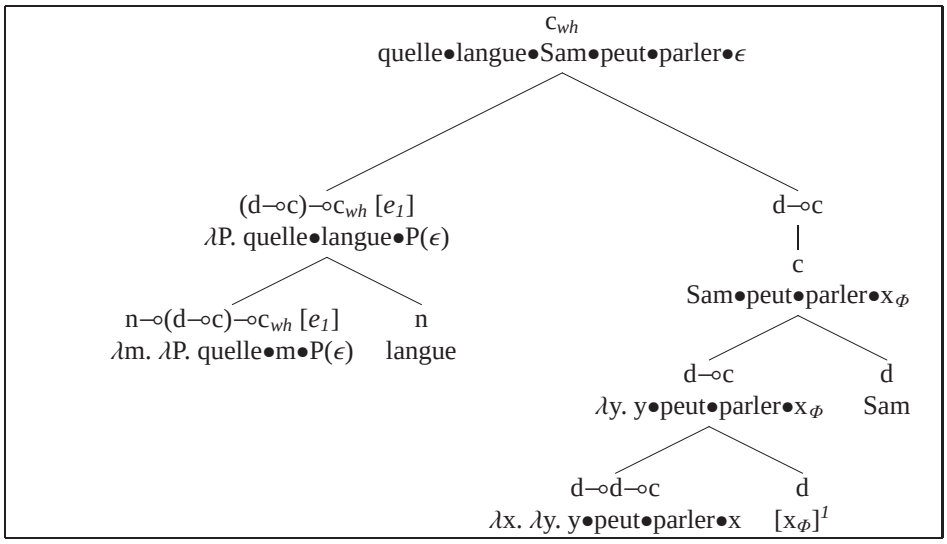
dérivation de cet exemple dans les **MG**). On associe ainsi au qu-déterminant '*quelle*' l'entrée composée ci-dessous afin de rendre compte de sa capacité de se déplacer une fois combiné avec son nom commun.

$$(e_1) \quad \vdash \begin{pmatrix} qu_\Phi \\ qu_\lambda \end{pmatrix} : n \multimap (d \multimap c) \multimap c_{wh} \quad \dashv [(x_\Phi, x_\lambda) : d \vdash (x_\Phi, x_\lambda) : d]$$

Le qu-déterminant '*quelle*' peut avoir les deux représentations phonétiques suivantes :

qu-in-situ	qu-ex-situ
qu_Φ=λm. λP. P(quelle•m)	qu_Φ=λm. λP. quelle•m•P(ϵ)

La première forme phonétique permet de dériver les questions qu-in-situ (e.g., les questions écho) où le qu-constituant subit un déplacement invisible. La deuxième représentation phonétique est, quant à elle, utilisée pour prendre en charge les phrases interrogatives qu-ex-situ dont les qu-expressions se déplacent de manière visible en position frontale. C'est le second Φ-terme que nous employons pour effectuer une analyse de notre exemple. Le schéma qui suit illustre l'interface syntaxe/phonétique de cette dérivation.



Ce schéma montre clairement que l'analyse globale est obtenue en composant deux dérivations partielles. La première, située à gauche, utilise la tête de l'entrée composée e_1 pour construire la projection maximale déplacée '*quelle langue*'. La seconde sous-dérivation (à droite), représente par contre une structure intermédiaire comprenant l'hypothèse contrôlée x_Φ qui occupe la position initiale de la qu-expression. La combinaison de ces deux dérivations partielles se fait grâce à la règle $\multimap IE_c$ qui, en collaboration avec la composante phonétique réussit à simuler l'effet d'un déplacement visible. Il est important de souligner que même si la règle logique $\multimap IE_c$ et la primitive de mouvement du **MP** produisent des résultats similaires, ces deux opérations sont basées sur des mécanismes très différents. A titre d'exemple, le déplacement transformationnel du **MP** est déclenché par la nécessité de vérifier (et annuler) les traits ininterprétables impliqués dans une dérivation afin de garantir sa convergence. En revanche, son homologue logique (i.e., la règle $\multimap IE_c$) est motivé par l'obligation d'abstraire toutes les hypothèses contrôlées introduites au cours d'une déduction et ce par leurs entrées propres associées.

Exemple 4.5.3 Nous proposons une seconde application de la définition logique du déplacement qui met en évidence la distinction entre ses deux variantes furtive et visible. Il s'agit de décrire le comportement syntaxique des syntagmes nominaux nominatifs et accusatifs notamment dans une dérivation qui met en évidence l'introduction de l'inflexion. Précisons que les groupes nominaux sujets se déplacent ouvertement en dehors du VP-shell (constitué du verbe et ses arguments) et atterrissent en position de spécifieur de l'inflexion alors que les syntagmes nominaux objets subissent un déplacement invisible qui garantit à leur forme phonétique de conserver sa place de départ. Ces actions dissemblables sont décrites par les deux entrées lexicales composées suivantes :

(e_{nom})	$\vdash$	$\begin{pmatrix} \lambda P.\ m \bullet P(\epsilon) \\ m_\lambda \end{pmatrix}$	$: (d_{nom} \multimap ip) \multimap ip$	$\multimap [(x_\Phi, x_\lambda) : d_{nom} \vdash (x_\Phi, x_\lambda) : d_{nom}]$
(e_{acc})	$\vdash$	$\begin{pmatrix} \lambda P.\ P(m) \\ m_\lambda \end{pmatrix}$	$: (d_{acc} \multimap vp) \multimap vp$	$\multimap [(x_\Phi, x_\lambda) : d_{acc} \vdash (x_\Phi, x_\lambda) : d_{acc}]$

Pour rendre compte de l'effet de l'inflexion, nous utilisons deux nouveaux types atomiques *vp* et *ip*. La première catégorie représente les VP-shell tandis que la seconde correspond aux clauses obtenues après l'introduction de l'inflexion. L'inflexion constitue donc le pont qui relie ces deux catégories, grâce à son type fonctionnel $vp \multimap ip$. La figure 35 présente la dérivation de $\mathcal{GLE}$ qui traduit l'analyse de la phrase '*Sam peut parler le Farsi*' (cf. Annexe C, Fig. 65).

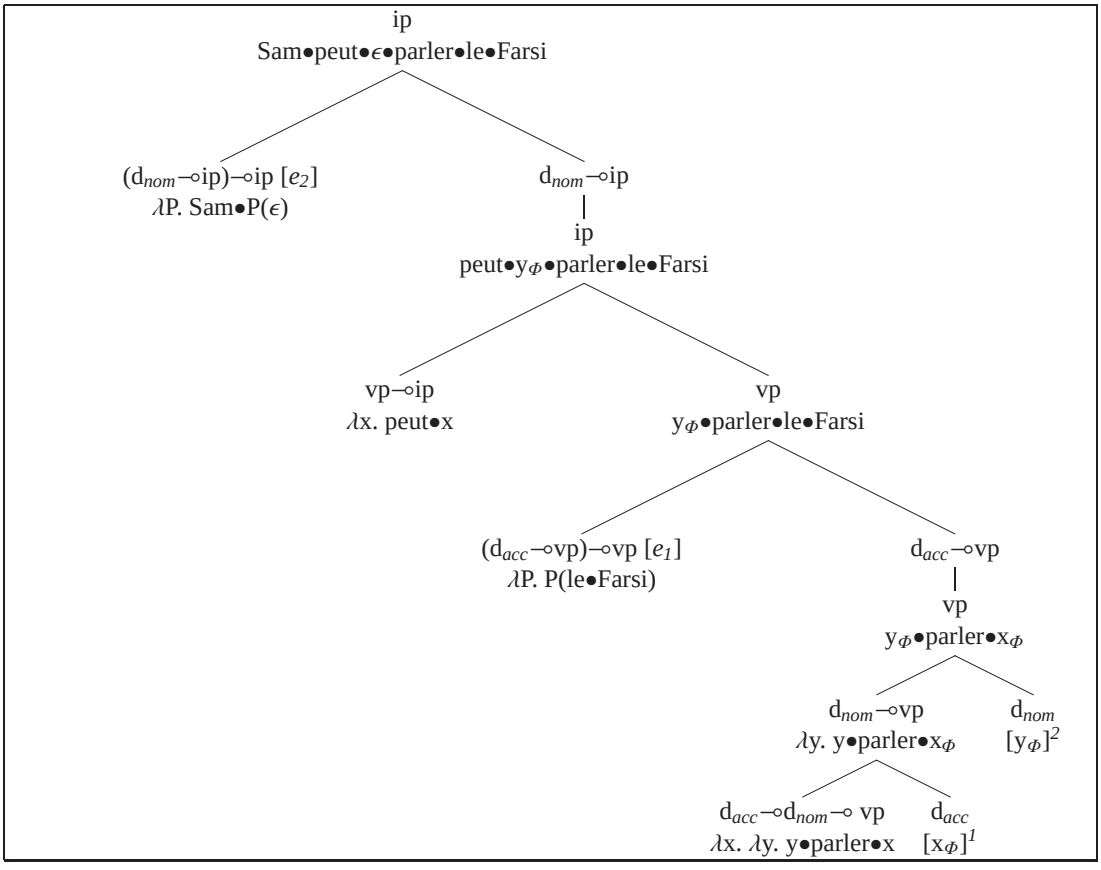


FIG. 35: Déplacements visible et furtif dans $\mathcal{GLE}$

Dans la structure intermédiaire représentant le VP-shell, les deux hypothèses contrôlées x_Φ et y_Φ occupent les sites d'origine respectifs de l'objet et du sujet. L'objet effectif '*le Farsi*' subit un déplacement invisible en déchargeant son hypothèse contrôlée, il garde alors sa position initiale (celle de x_Φ). Le sujet, quant à lui, ne retrouve sa position finale qu'après la combinaison du VP-shell avec le verbe modal '*peut*' qui coïncide avec l'inflexion. Le site de y_Φ est donc rempli par une trace ϵ et le sujet '*Sam*' est placé au début de la phrase.

4.5.3 Interface syntaxe/sémantique

L'utilisation de $\mathcal{GLE}$ comme méta-langage formalisant rigoureusement les mécanismes du **MP** garantit l'obtention de la sémantique de l'énoncé analysé grâce à la correspondance de Curry-Howard. A titre d'exemple, l'interface syntaxe/sémantique qui correspond à l'exemple 4.5.3 est présentée ci-après. Pour plus de simplicité, nous négligeons la sémantique du verbe modal '*pouvoir*'.

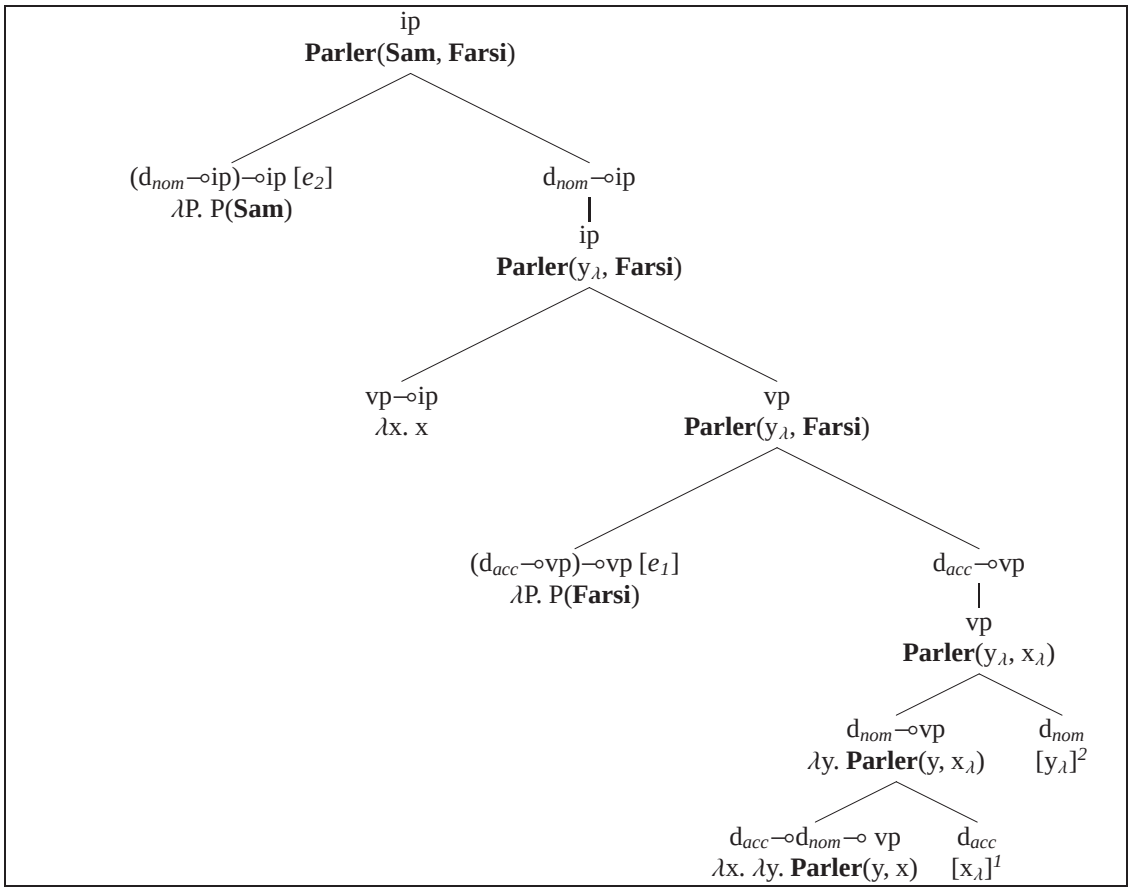


FIG. 36: Interface syntaxe/sémantique

Le sens de la phrase interrogative présentée dans l'exemple 4.5.2 est relativement plus ardu à

obtenir. Dans la littérature, plusieurs théories ont été proposées pour rendre compte de la sémantique formelle des questions dont l'approche de sens structuré (i.e., *structured meaning approach*) [107] que nous utiliserons dans la suite. Cette dernière approche considère que le sens d'une question est intimement lié à celui de sa réponse. Ainsi, les énoncés interrogatifs ne constituent pas des phrases autonomes, en revanche, ils forment une unité syntaxique et sémantique avec leurs réponses. A titre d'exemple, la question '*Quelle langue Sam peut parler ?*' peut être vue comme un foncteur qui cherche à se combiner avec un groupe nominal formant une réponse potentielle pour produire un énoncé complet. Par conséquent, le type abstrait c_{wh} associé à cette question n'est autre qu'un alias représentant le type fonctionnel $d \multimap c$.
Afin de calculer le sens de notre exemple, nous attribuons au qu-déterminant '*quelle*' la sémantique lexicale suivante :

$$qu_{\lambda} = \lambda P.\ \lambda Q.\ \lambda x.\ P(x) \wedge Q(x) : (e \to t) \to (e \to t) \to e \to t$$

L'interface syntaxe/sémantique de $\mathcal{GLE}$ nous permet de déduire, sans effort, le sens de la question précédente représenté par le λ-terme ci-après :

$$\lambda x.\ \mathbf{Langue}(x) \wedge \mathbf{Parle}(\mathbf{Sam},\ x)$$

La fusion de notre phrase interrogative avec sa réponse potentielle '*le Farsi*' (représentée sémantiquement par l'individu **Farsi**) engendre ainsi la proposition qui suit :

$$\mathbf{Langue}(\mathbf{Farsi}) \wedge \mathbf{Parle}(\mathbf{Sam},\ \mathbf{Farsi})$$

4.6 Conclusion

Dans ce chapitre, nous avons présenté le nouveau formalisme $\mathcal{GLE}$ qui est basé sur un fragment réduit de la logique linéaire intuitionniste implicative et exponentielle. L'originalité de ce modèle provient de sa capacité à utiliser le raisonnement hypothétique de manière contrainte. En effet, seuls les axiomes qui sont explicitement mentionnés par certaines entrées lexicales peuvent être introduites au cours de la dérivation. Cette restriction est fructueuse car elle augmente la compacité du système et garantit la normalité de toutes ses dérivations.
Le modèle $\mathcal{GLE}$ possède des atouts qui sont en phase avec l'esprit du Programme Minimaliste. Nous avons montré qu'il est possible d'utiliser $\mathcal{GLE}$ pour simuler les mécanismes transformationnels sous-jacents aux Grammaires Minimalistes et tirer par conséquent profit de la connexion aisée avec la sémantique de Montague. Ainsi, l'opération de fusion est encodée de manière logique par la règle simple de modus-ponens. En revanche, la primitive de mouvement est prise en charge par une règle hybride combinant une étape de raisonnement hypothétique avec une phase de fusion.

Applications linguistiques dans $\mathcal{GLE}$

Après avoir conçu un système logique basé sur une architecture compacte et modulaire compatible avec la philosophie du Programme Minimaliste, la priorité doit être accordée au test de sa pertinence linguistique. La définition d'un formalisme complet pouvant décrire avec élégance et simplicité tous les phénomènes linguistiques existants reste un projet ambitieux qui est loin d'être réaliste[1]. De manière plus raisonnable, on peut qualifier un modèle linguistique de pertinent à partir du moment où il permet d'analyser un certain nombre de phénomènes linguistiques complexes tout en proposant des explications qui facilitent leur compréhension.
L'objectif de ce chapitre est donc de présenter quelques unes des applications linguistiques avancées du formalisme $\mathcal{GLE}$. Nous nous focaliserons en particulier sur la description de certains phénomènes difficiles à analyser dans les grammaires catégorielles classiques. Une large partie sera ainsi dédiée aux phénomènes non-linéaires tels que le liage d'anaphores et l'ellipse qui nécessitent un partage contraint de ressources. Pour chaque phénomène abordé, notre contribution sera comparée avec les travaux antérieurs notamment ceux qui sont effectués au sein des sytèmes logiques bi-directionnels.

5.1 Discontinuité

5.1.1 Discontinuité dans les systèmes bi-directionnels

Le phénomène de discontinuité survient dans une variété de langues naturelles sous la forme d'une séquence de mots qui occupent des positions éloignées dans une phrase tout en représentant une seule unité sémantique. Ceci est le cas, par exemple, de la négation courante en français (i.e., ne ... pas), des verbes à particules en anglais (e.g., ring ... up) ainsi que des expressions idiomatiques (e.g., recevoir ... à bras ouverts).
Il n'est pas possible de rendre compte de la discontinuité dans le calcul pur de Lambek car les connecteurs de type utilisés ont une interprétation concaténative : ils ne peuvent exprimer que l'agencement côte à côte des ressources. De nombreux travaux ont tenté d'enrichir ce calcul en introduisant un nombre d'opérateurs discontinus tels que les connecteurs d'infixation et d'extraction ($\downarrow$, $\uparrow$) [74, 86, 55, 85], qui généralisent les rôles de suffixation et préfixation des implications / et \. A titre d'exemple, la proposition de Morrill et Solias [86, 85], est basée sur un système logique complet qui manipule des types étiquetés par les termes d'une algèbre prosodique bimodale. Les étiquettes considérées sont formées à partir de deux opérations : une première opération de concaténation (+) qui construit des chaines plates et une deuxième ($<\ ,\ >$) qui engendre des couples. La

[1] Si on tient en compte l'état actuel de nos connaissances, au jour de la rédaction de la présente thèse.

notion de couple permet de marquer un point d'insertion qui coïncide avec le site de l'argument manquant du constituant discontinu. Ainsi, le verbe anglais à particule séparable *rang ... up* est représenté dans cette approche par l'entrée lexicale suivante :

Etiquette	Type syntaxique	Forme Sémantique
<rang , up >	(np\s)↑np	λx. λy. **Phone**(y, x)

La combinaison de ce constituant discontinu avec son objet infixé se fait grâce à la règle d'élimination du connecteur d'extraction ↑ comme le montre le schéma ci-après présenté dans le style de Prawitz[93] :

Règle logique	Application Linguistique
$\dfrac{< a_1 \ , \ a_2 >: A \uparrow B \quad b : B}{a_1 + b + a_2 : A} \uparrow E$	$\dfrac{< rang \ , \ up >: (np\backslash s) \uparrow np \quad John : np}{rang + John + up : d_{nom}\backslash c} \uparrow E$

Les choses deviennent plus compliquées lorsqu'il s'agit de traiter les constituants discontinus comprenant plus d'un site d'extraction infixe comme c'est le cas de '*if ... then ... otherwise*' illustré dans la phrase ci-après :

(15) If Mary succeeds then John will be happy otherwise Jim will be sad.

La solution proposée par Morrill et Merenciano [85] consiste à définir une famille de connecteurs discontinus généralisés (e.g., $\uparrow_n$ et $\downarrow_n$) permettant de manipuler des n-uplets de chaines prosodiques. Ainsi, on peut assigner au triplet $< if \ , \ then \ , \ otherwise >$ correspondant au constituant discontinu précédent le type syntaxique suivant $((s/s) \uparrow_1 s)\uparrow_2 s$. Le numéro *n* décorant l'opérateur $\uparrow_n$ indique le rang du site d'extraction ciblé. La première clause qui fusionnera avec ce constituant occupera alors le second site d'extraction du triplet tandis que la seconde se placera dans le premier site. Pour effectuer la première fusion, on ne peut se servir de la règle $\uparrow E$ précédemment décrite, il est nécessaire d'enrichir la logique considérée par la nouvelle règle d'élimination ci-dessous :

$$\frac{< a_1 \ , \ a_2 \ , \ a_3 >: A \uparrow_2 B \quad b : B}{< a_1 \ , \ b + a_2 >: A} \uparrow_2 E$$

La dérivation de la phrase 15 est ainsi synthétisée dans le schéma ci-après (où C_1, C_2 et C_3 représentent respectivement les clauses de *if*, *then* et *otherwise*) :

$$\dfrac{\dfrac{\dfrac{< if \ , \ then \ , \ otherwise >: ((s/s) \uparrow_1 s) \uparrow_2 s \quad C_2 : s}{< if \ , \ then + C_2 + otherwise >: (s/s) \uparrow_1 s} \uparrow_2 E \quad C_1 : s}{if + C_1 + then + C_2 + otherwise : s/s} \uparrow_1 E \quad C_3 : s}{if + C_1 + then + C_2 + otherwise + C_3 : s} /E$$

FIG. 37: Exemple d'utilisation des connecteurs discontinus

L'approche de Morrill et Solias [86, 85] a été ultérieurement reprise et formalisée dans le cadre d'un système multimodal [83, 53]. Les opérateurs discontinus sont donc exprimés par des variantes non concaténatives des implications (/, \) qui sont décorées par des modes spécifiques tels que le mode d'emballage (i.e., *wrapping*) *w*. Ainsi, le type $A/_w B$ coïncide avec le type $A \uparrow B$ représentant les expressions incomplètes de type *A* qui leur manque un constituant de type *B* quelque part.

Dans un même ordre d'idées, le site d'insertion est localisé grâce à l'utilisation d'un opérateur structurel binaire non associatif regroupant les fragments de l'expression discontinue. L'argument manquant du constituant discontinu réussit à trouver son site médian par l'intermédiaire d'un paquetage de postulats d'interaction qui régissent la communication entre le mode *w* et les autres modes de composition du système (e.g., *n* : mode non-associatif et *a* : mode associatif). La règle structurelle la plus importante est celle qui introduit une sorte de commutativité locale nécessaire au déplacement visé. Cette dernière règle est représentée ci-après :

$$((\Gamma_1 \, , \, \Gamma_2)^n \, , \, \Gamma_3)^n \overset{WN}{\longleftrightarrow} (\Gamma_1 \, , \, (\Gamma_3 \, , \, \Gamma_2)^a)^a$$

A la différence de la première approche qui utilise des étiquettes raffinées [86, 85], les grammaires multimodales ne permettent pas de regrouper tous les fragments d'une expression discontinue dans la même entrée lexicale. En effet, dans cet environnement, chaque sous-expression doit être considérée comme une entrée autonome. A titre d'exemple, dans [53], Hendriks propose de rendre compte de l'usage du comparatif discontinu '*more ... than*' au sein d'un système multimodal afin de permettre l'analyse de phrases telles que '*Pat ate more bagels than Kim bought donuts*'. Les deux ressources '*more*' et '*than*' sont ainsi typées séparément. Alors que le comparatif *more* est considéré comme un quantificateur spécial, l'expression '*than*' est traitée comme une conjonction coordonnant deux clauses pour lesquelles il manque un quantificateur quelque part[2].
Outre le fait d'être contre-intuitif, le traitement séparé des différents constituants d'une expression discontinue pose généralement certains inconvénients. D'un point de vue syntaxique, cette méthode augmente considérablement la complexité du lexique. En effet, afin d'éviter tout problème de sur-génération, il est nécessaire de lier les différents constituants discontinus par un trait commun encodé moyennant un opérateur de contrôle particulier. A titre d'exemple, pour prendre en charge le verbe di-transitif '*donne ... à*', on assigne à la tête '*donne*' le type syntaxique $(d\backslash s)/\Box_a pp/d$ et on attribue à la préposition '*à*' la catégorie $\Box_a pp/d$. L'utilisation de l'opérateur $\Box_a$ force le verbe '*donne*' à avoir comme objet indirect une phrase prépositionnelle préfixée par la préposition '*à*'. Une telle technique augmente fortement le nombre de modes du système et diminue par conséquent sa compacité et lisibilité. D'autre part, cette approche ne s'avère adéquate que si la sémantique de l'expression discontinue impliquée est définie de manière compositionnelle. Elle ne peut pas alors être utilisée pour traiter certaines formes de discontinuité constituant une seule unité sémantique comme c'est le cas pour les verbes à particules ou les expressions idiomatiques (e.g., le sens du verbe '*give ...up*' n'est pas obtenu à partir du sens du verbe '*give*').

5.1.2 Traitement de la discontinuité dans $\mathcal{GLE}$

Tous les traitements de la discontinuité proposés dans les systèmes bi-directionnels augmentent la complexité de la logique sous-jacente à cause des différentes extensions adoptées (e.g., introduction de nouveaux connecteurs de discontinuité ou de nouveaux postulats d'interaction ...). En revanche, le formalisme $\mathcal{GLE}$ permet de rendre compte du phénomène de discontinuité de manière directe et intuitive sans avoir recours à un enrichissement supplémentaire. En effet, $\mathcal{GLE}$ offre la possibilité de rassembler tous les constituants discontinus dans une même entrée lexicale. Le site d'insertion infixe de l'expression discontinue sera, quant à lui, marqué par une variable liée. L'utilisation des Φ-termes permet ainsi d'exprimer aisément l'idée de couples prosodiques

[2] Les deux clauses coordonnées dans l'exemple cité sont '*Kim bought _ donuts*' et '*Pat ate _ bagels*', où _ marque le site du quantificateur manquant.

introduite par Morrill et Solias [86]. En outre, seule l'implication linéaire $\multimap$ suffit pour simuler toute la famille d'opérations d'extraction $\uparrow_n$ comme le montre le tableau suivant :

Approche de Morrill et Solias	$\mathcal{GLE}$
$< ring\ ,\ up > : (np \backslash s) \uparrow np$	$\lambda x.\ \lambda y.\ y \bullet\ rang \bullet x \bullet up : d_{acc} \multimap d_{nom} \multimap c$
$< if\ ,\ then\ ,\ otherwise >: ((s/s)\ \uparrow_1 s) \uparrow_2 s$	$\lambda x.\ \lambda y.\ \lambda z.\ if \bullet y \bullet then \bullet x \bullet otherwise \bullet z :$ $c \multimap c \multimap c \multimap c$

FIG. 38: Exemples de traductions des opérateurs d'extraction dans $\mathcal{GLE}$

Par conséquent, $\mathcal{GLE}$ permet d'avoir des analyses plus lisibles, compactes et homogènes puisque les règles logiques de l'implication linéaire $\multimap$ suffisent pour exprimer le comportement des différents connecteurs binaires continus et discontinus ($/$, $\backslash$, $\uparrow$, $\uparrow_2$ ...).

Exemple 5.1.1 Nous proposons dans ce qui suit d'illustrer l'analyse syntaxique et sémantique de la discontinuité dans $\mathcal{GLE}$ et ce sur l'exemple (16) suivant :

(16) More logicians knew Godel than physicists met Einstein

D'une façon générale, le comparatif '*more ...than*' établit une liaison entre deux phrases pour lesquelles il manque un déterminant quelque part [53]. Dans notre cas, par exemple, les deux déterminants absents sont en position sujet. L'expression '*more*' remplit alors la position du déterminant manquant dans la clause de gauche, alors que le site du quantificateur de la clause de droite reste vide. D'autre part, la sémantique de ce comparatif stipule que la cardinalité du premier ensemble d'individus (e.g., les logiciens qui ont connu Godel) est plus grande que celle du deuxième (e.g., les physiciens qui ont rencontré Einstein).
Dans ce cas de figure, la description du comportement syntaxique, phonétique et sémantique du comparatif discontinu est encapsulée dans l'entrée ci-après :

$$\left(\begin{array}{c} \lambda x.\ \lambda P.\ \lambda y.\ \lambda Q.\ more \bullet y \bullet Q(\epsilon) \bullet than \bullet x \bullet P(\epsilon) \\ \lambda P_1.\ \lambda P_2.\ \lambda Q_1.\ \lambda Q_2.\ \mathbf{More}(\lambda x.\ Q_1(x) \wedge Q_2(x), \\ \lambda x.\ P_1(x) \wedge P_2(x)) \end{array}\right) : n \multimap (d_{nom} \multimap c) \multimap n \multimap (d_{nom} \multimap c) \multimap c$$

Les étapes principales de la dérivation de cet exemple sont présentées dans la figure 39. On se focalise uniquement sur l'interface syntaxe/phonétique sachant que la synchronisation entre la syntaxe et la sémantique se fait de manière similaire.

Il n'est pas difficile de vérifier que l'interface syntaxe/sémantique produit effectivement le sens souhaité à savoir :

$$\mathbf{More}(\lambda x.\ \mathbf{Logician}(x) \wedge \mathbf{Knew}(x,\ \mathbf{Godel}),\ \lambda x.\ \mathbf{Physicist}(x) \wedge \mathbf{Met}(x,\ \mathbf{Einstein}))$$

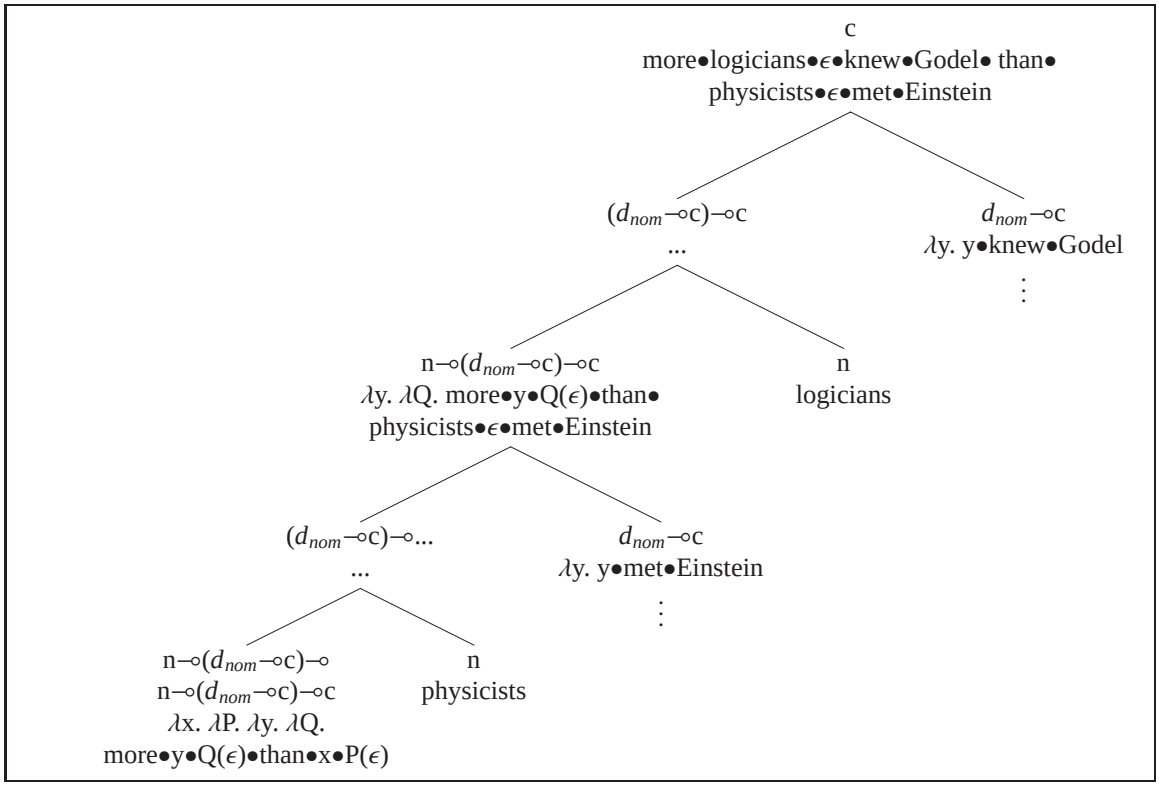


FIG. 39: Exemple d'analyse du comparatif discontinu '*more ...than*'

5.2 Liage des Anaphores et Pronoms

5.2.1 Contributions antérieures

Approche Générative

L'approche prépondérante traitant le phénomène de liage est celle formulée par Chomsky dans les années 80 [30]. Cette dernière prend la forme d'un module qui contient un ensemble de règles régissant les anaphores (e.g., réflexifs tels que *himself*), les pronoms non-réflexifs (e.g., pronoms personnels tels que '*he*') et les expressions pleinement référentielles (e.g., *John, the man*). Chacune de ces règles énonce une contrainte de localité qui délimite certaines configurations structurales dans lesquelles ces éléments admettent, exigent ou excluent un terme de même référence. Les principes de la théorie de liage peuvent être synthétisés comme suit (où un terme est dit *lié* s'il est co-indexé avec un élément qui le c-commande[3] et *libre* s'il n'est pas lié) :

- Principe *A* : Une anaphore doit être liée dans son domaine local.
- Principe *B* : Un pronom (non-réflexif) doit être libre dans son domaine local.
- Principe *C* : Une expression pleinement référentielle doit être libre.

La notion de *domaine local* mentionnée par ces principes est relativement complexe à définir, nous nous contenterons de dire que ceci représente la plus petite clause comprenant l'anaphore ou le pronom en jeu.
Les principes de la théorie de liage sont capables de justifier la grammaticalité ou l'agrammaticalité des énoncés ci-après :

(17) a. John$_i$ likes himself$_i$.
b. *John$_i$ thinks that Mary likes himself$_i$.
c. *John$_i$ likes him$_i$.
d. John$_i$ thinks he$_i$ is smart.
e. *He$_i$ thinks that John$_i$ will succeed.

En effet, d'après le principe *A*, le liage entre le réflexif '*himself*' et le nom propre '*John*' est nécessaire dans l'énoncé (17a), puisque ce réflexif requiert un antécédent dans sa clause minimale et le seul candidat potentiel dans une telle circonstance est le sujet. En revanche, ce liage est interdit dans l'énoncé (17b) où '*John*' se trouve à l'extérieur du domaine local du réflexif, qui coïncide dans ce cas avec la subordonnée complétive imbriquée. D'autre part, en appliquant le principe *B*, '*John*' ne peut être lié au pronom personnel '*him*' dans l'énoncé (17c) car ils sont tous les deux localisés dans la même clause minimale, alors que ceci est possible dans l'exemple (17d). Finalement, la référence disjointe exigée entre '*John*' et '*he*' dans l'exemple (17e) est expliquée par le principe *C* qui interdit aux expressions référentielles d'être dans le domaine d'un terme portant le même indice (i.e., d'être c-commandées par ce terme).
Des travaux récents ont tenté d'interpréter les principes de la théorie du liage en termes de mouvement et ce dans le cadre du Programme Minimaliste. Parmi ces travaux, citons l'approche de Kayne décrite dans [64]. Dans sa proposition, Kayne considère que le pronom et son antécédent intégrent simultanément la dérivation sous la forme d'un constituant double, e.g., [John, he]. Un tel constituant est ultérieurement séparé suite au déplacement de son spécifieur (i.e., l'antécédent)

[3] On dit que A c-commande B ssi A ne domine pas B et si le premier constituant qui domine A domine aussi B.

à une position frontale engendrant ainsi le liage du pronom en question.
A titre d'exemple, en suivant l'idée de Kayne, l'énoncé (17d) peut être analysé par la dérivation intuitive suivante :

$$\textit{thinks}\ [\textit{John},\ \textit{he}]\ \textit{is smart} \longrightarrow \textit{John}_i\ \textit{thinks}\ [t_i,\ \textit{he}]\ \textit{is smart}$$

La formalisation de la relation entre le pronom et son antécédent moyennant l'opération de mouvement est fructueuse car elle permet d'exprimer de manière élégante les différents principes de la théorie du liage. Ainsi, il n'est pas difficile de constater que le principe *C*, par exemple, découle directement de l'impossibilité de déplacement de haut en bas (i.e., *downward move*) comme le montre le schéma suivant :

$$[\textit{John},\ \textit{he}]\ \textit{thinks is smart} \nrightarrow [t_i,\ \textit{he}]\ \textit{thinks John}_i\ \textit{is smart}$$

Approche Pragmatique de la référence disjointe

La théorie de liage propose de rendre compte du phénomène de la référence disjointe (i.e., impossibilité de coréférence) par des règles syntaxiques pures (principes *B* et *C*). Toutefois ces principes négatifs sont très rigides et peuvent être violés dans des contextes particuliers comme c'est le cas pour les énoncés suivants qui sont corrects même s'ils ne respectent pas le principe *C* :

(18) a. [The man whom I saw]$_i$ was John$_i$.
b. i. Tom : Did anyone vote for John ?
ii. Jerry : Only John$_i$ voted for John$_i$.

Pour remédier à ce problème, certains auteurs dont Dowty [44] et Reinhart [96] suggèrent de limiter le rôle de la syntaxe à la détermination de la possibilité ou la nécessité de liage d'un pronom. Ils proposent d'introduire la dimension pragmatique pour expliquer le phénomène de référence disjointe, notamment, en ayant recours au principe conversationnel de Grice qui peut être paraphrasé comme suit :

> *Soient A et B deux expressions qui peuvent exprimer le sens χ. Si A est sémantiquement non ambigüe et B est sémantiquement ambigüe pouvant signifier χ ou ψ, alors l'énoncé A est le plus approprié pour exprimer le sens χ. L'énoncé B, quant à lui, devra être réservé à l'expression du sens ψ.*

De ce point de vue, l'interprétation coréférencielle de la phrase '*John likes him*' est considérée comme étant syntaxiquement valide. En revanche, elle est rejetée pour des raisons pragmatiques car elle peut être exprimée de manière non-ambigüe par l'énoncé '*John likes himself*'. Cette approche a été suivie par Bonato dans son traitement computationnel de la théorie de liage [24].

Phénomène de liage et systèmes bi-directionnels

Le phénomène de liage est un phénomène non-linéaire qui fait intervenir un partage de ressources au niveau de la sémantique. Ce phénomène constitue un vrai défi pour les grammaires bi-directionnelles basées sur des logiques sensibles aux ressources qui interdisent généralement

l'utilisation multiple des prémisses.
Dans la littérature, on trouve plusieurs travaux qui tentent de rendre compte du liage des anaphores et pronoms dans les grammaires catégorielles dont [104, 75, 54, 83, 84].
Afin de traiter la non-linéarité exigée par les réflexifs, Szabolcsi [104], Moortgat [75] et Morrill [83, 84] suggèrent d'attribuer à ces anaphores une sémantique lexicale formelle qui encode la duplication des ressources requise. L'étude de Szabolcsi est réalisée dans le cadre des grammaires catégorielles combinatoires [103], mais elle reste aussi valide dans le calcul de Lambek. Dans cette approche, les réflexifs dépendants du sujet sont considérés comme étant des foncteurs sur des verbes (transitifs ou di-transitifs) qui décrémentent l'arité de leur argument. D'un point de vue sémantique, un λ-terme non-linéaire est assigné au pronom réflexif. Ce λ-terme permet d'identifier l'argument éliminé avec le sujet. A titre d'exemple, le réflexif '*himself*' de l'énoncé (17a) peut être vu comme une fonction portant le type monté $((np\backslash s)/np)\backslash(np\backslash s)$ dont le rôle est de transformer un verbe transitif en un verbe non-transitif. Sa sémantique formelle est décrite par le terme $\lambda P.\ \lambda x.\ P(x,\ x)$. L'entrée ainsi définie peut être également utilisée pour analyser l'énoncé (19a) ci-après puisque la combinaison du verbe di-transitif '*shows*' avec son objet direct '*Mary*' engendre un constituant ayant le même comportement syntaxique qu'un verbe transitif.

(19) a. John$_i$ shows Mary himself$_i$.
b. John$_i$ gave himself$_i$ a nice present.
c. John$_i$ talked about himself$_i$.

Toutefois, il est nécessaire de rajouter de nouvelles entrées lexicales quand ledit réflexif occupe d'autres positions notamment lorsqu'il coïncide avec l'objet direct d'un verbe di-transitif (19b) ou quand il est imbriqué dans un complément prépositionnel (19c).
Afin de traiter les réflexifs dépendants du sujet de manière uniforme, Moortgat propose dans [75] d'utiliser son connecteur de quantification $\Uparrow$ [76]. Dans cette approche, toutes les anaphores liées au sujet portent le même type syntaxique $np \Uparrow (np\backslash s)$ puisqu'elles se comportent syntaxiquement comme des groupes nominaux (de type np) qui ne seront interprétés qu'au niveau du syntagme verbal (de type $np\backslash s$). Le schéma ci-dessous présente la dérivation proposée par Moortgat de l'énoncé (17a). Pour plus de commodité, le comportement global du connecteur $\Uparrow$ est récapitulé dans la partie gauche du schéma[4].

Comportement de $\Uparrow$	Dérivation de l'énoncé (17a)
$\dfrac{\dfrac{\alpha : B \Uparrow A}{x : B} \Uparrow E^n \\ \vdots \\ \beta : A}{\alpha(\lambda x.\ \beta) : A}\ n$	$\dfrac{\dfrac{\dfrac{\dfrac{}{\textit{John}}}{\textbf{John} : np} \quad \dfrac{\dfrac{\dfrac{}{\textit{likes}}}{\lambda x.\ \lambda y.\ \textbf{Like}(y,\ x) : (np\backslash s)/np} \quad \dfrac{\dfrac{\dfrac{}{\textit{himself}}}{\lambda P.\ \lambda x.\ P(x,\ x) : np \Uparrow (np\backslash s)} \Uparrow E^0}{x : np}\ /E}{\lambda y.\ \textbf{Like}(y,\ x) : np\backslash s}}{\lambda x.\ \textbf{Like}(x,\ x) : np\backslash s}\ 0}{\textbf{Like}(\textbf{John},\ \textbf{John}) : s}$

Toutefois, les deux approches de Szabolcsi et Moortgat souffrent de certaines lacunes. En effet, ces études ne prennent en charge que les anaphores dont l'antécédent est le sujet. Cette première faiblesse résulte de l'utilisation d'une logique concaténative qui force les verbes di-transitifs à se

[4] Notons que le correspondant sémantique du type syntaxique $A \Uparrow B$ est $(A^* \to B^*) \to B^*$.

combiner d'abord avec leur complément d'objet direct avant de fusionner avec leur objet indirect. Il est néanmoins clair que l'anaphore doit intégrer la dérivation principale avant son antécédent pour avoir accès à sa sémantique. C'est pour cette raison qu'il s'avère impossible de spécifier une entrée lexicale pour les anaphores dépendant d'un objet direct, excluant ainsi l'analyse d'énoncés corrects tels que 20a. En revanche, il est tout à fait possible de définir une entrée liant, à tort, une anaphore à un objet indirect, chose qui permettrait d'engendrer des phrases agrammaticales telles que 20b.

(20) a. John showed Mary$_i$ herself$_i$ in the mirror.
b. *John showed herself$_i$ Mary$_i$ in the mirror.

Le second problème posé par ces deux approches concerne le non-respect de la contrainte de localité exigée par le principe *A*. En effet, même si ces modèles forcent les réflexifs à être liés, rien ne les empêche de choisir un antécédent éloigné ne faisant pas partie de leur domaine local. Une telle flexibilité permet d'engendrer assez aisément des énoncés incorrects tels que (17b) [54, 26].
Afin de remédier à ces failles, Morrill [83] propose d'enrichir le calcul de base en élargissant l'éventail des connecteurs logiques utilisés. Ainsi, ce dernier se sert d'une famille de connecteurs discontinus généralisés dans le but de spécifier une nouvelle entrée qui décrit le comportement des réflexifs dépendants des objets directs. L'idée principale de Morrill est de forcer les verbes di-transitifs à se combiner d'abord avec leur objet indirect pour former un constituant discontinu de type $(np\backslash s)\uparrow np$[5] (e.g., $< showed, herself >$). Ensuite, l'objet direct est intégré dans sa position médiane par l'intermédiaire de la règle $\uparrow E$ présentée dans la section 5.1.1. La combinaison du verbe avec le réflexif (occupant le site de l'objet indirect) précéde ainsi sa fusion avec l'objet direct, chose qui assure le liage souhaité.
D'autre part, Morrill suggère d'employer le connecteur modal de contrôle $\Box$ afin de délimiter les frontières des clauses locales et interdire ainsi le liage à longue distance. Néanmoins, les solutions adoptées par Morrill viennent au détriment de la simplicité et la lisibilité du système de base.
Les approches dédiées au traitement des anaphores ne peuvent s'appliquer pour rendre compte des pronoms personnels puisque la distance entre ces derniers et leurs antécédents est non-bornée. En outre, on est dans l'impossibilité de deviner exactement la position structurelle de l'antécédent par rapport au pronom et la nature des ressources qui les séparent. Par conséquent, il n'est pas admissible de déléguer l'expression du partage de ressources entre un pronom personnel et son antécédent à la sémantique lexicale du pronom. Plusieurs auteurs dont Hepple [54] et Jäger [61] proposent de refléter l'utilisation multiple de ressources sémantiques sur la syntaxe.
Ces derniers travaux considèrent que le sens du pronom est exprimé par la fonction d'identité $\lambda x.\ x$ qui permet d'identifier la sémantique du pronom à celle de son antécédent. Pour garantir la correspondance entre la syntaxe et la sémantique, il est nécessaire de projeter ce comportement fonctionnel sur le type syntaxique du pronom. A cette fin, Hepple et Jäger ont introduit différentes extensions du calcul de Lambek qui partagent la même philosophie. Ainsi, Jäger [61] par exemple, propose l'introduction d'une troisième implication notée | qui ne consomme pas son argument. Intuitivement, une expression porte le type $A|B$ si et seulement si elle se comporte comme une expression de type *A* quand elle est précédée par une expression de type *B*. Ainsi, Jäger assigne aux pronoms personnels le type $np|np$ puisqu'ils requièrent un antécédent de type np qui sera utilisé sans être consommé. Le comportement formel de ce nouveau connecteur est décrit par les règles logiques suivantes :

[5] Un groupe verbal auquel il manque un syntagme nominal quelquepart

$$\frac{\beta : B \; ... \; f : A|B}{\beta : B \; ... \; (f\ \beta) : A}\ |E \qquad\qquad \frac{\dfrac{\overline{x : B}^{\,i} \quad \overline{y : p}^{\,i} \quad \vdots}{(x, y, f) : B \bullet p \bullet A}}{\lambda x.\ f : A|B}\ |I, i$$

Ainsi, la règle d'élimination $|E$ énonce qu'une expression de type $A|B$ peut se combiner avec une ressource qui la précéde de type B pour former un constituant de type A. Cette application ne consomme pas l'antécédent B mis en jeu. D'autre part, la règle d'introduction $|I$ stipule que pour prouver qu'une expression α est de type $A|B$ il suffit de démontrer qu'elle se comporte syntaxiquement comme une ressource de type A si elle est précédée par un antécédent hypothétique de type B. La deuxième ressource hypothétique $y : p$ est utilisée pour représenter le bloc d'expressions intervenant entre α et son antécédent.
L'approche de Jäger est basée sur une seule contrainte structurelle, à savoir celle de la précédence linéaire entre l'antécédent et le pronom. Cette contrainte est suffisante pour bloquer l'analyse de phrases violant le principe C telles que (21c) mais elle n'est pas assez puissante pour rendre compte du principe B et ainsi légitimer l'agrammaticalité de l'énoncé (21d).

(21) a. John$_i$ said he$_i$ walked.

b. John$_i$ said he$_j$ walked.

c. *He$_i$ said John$_i$ walked.

d. *John$_i$ likes him$_i$.

Nous illustrons dans ce qui suit l'utilisation de l'approche de Jäger pour dériver les deux lectures coréférentielle (21a) et libre (21b) de la phrase '*John said he walked*' [61].

$$\dfrac{\dfrac{\dfrac{\dfrac{\textit{John}}{[\mathbf{John} : np]_i}\ Lex \quad \dfrac{\dfrac{\textit{said}}{\lambda P.\ \lambda x.\ \mathbf{Say}(x,\ P) : (np\backslash s)/s}\ Lex \quad \dfrac{\dfrac{\textit{he}}{[\lambda x.\ x : np|np]_i}\ Lex}{\mathbf{John} : np}\ |E}{\mathbf{Walk}(\mathbf{John}) : s}\ /E}{\lambda x.\ \mathbf{Say}(x,\ \mathbf{Walk}(\mathbf{John})) : np\backslash s}\ \backslash E \quad \dfrac{\textit{walked}}{\lambda x.\ \mathbf{Walk}(x) : np\backslash s}\ Lex}{\mathbf{Say}(\mathbf{John},\ \mathbf{Walk}(\mathbf{John})) : s}\ \backslash E$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{x : [np]_i}^{\,1} \quad \overline{y : p}^{\,1}}{(x, y) : np \bullet p}\ \bullet I \quad \dfrac{\dfrac{\textit{John}}{\mathbf{John} : np}\ Lex \quad \dfrac{\dfrac{\textit{said}}{\lambda P.\ \lambda z.\ \mathbf{Say}(z,\ P) : (np\backslash s)/s}\ Lex \quad \dfrac{\dfrac{\dfrac{\textit{he}}{[\lambda x.x : np|np]_i}\ Lex}{x : np}\ |E \quad \dfrac{\textit{walked}}{\lambda x.\ \mathbf{Walk}(x) : np\backslash s}\ Lex}{\mathbf{Walk}(x) : s}\ \backslash E}{\lambda z.\ \mathbf{Say}(z,\ \mathbf{Walk}(x)) : np\backslash s}\ /E}{\mathbf{Say}(\mathbf{John},\ \mathbf{Walk}(x)) : s}\ \bullet I}{(x, y, \mathbf{Say}(\mathbf{John},\ \mathbf{Walk}(x))) : np \bullet p \bullet s}\ |I, 1}{\lambda x.\ \mathbf{Say}(\mathbf{John},\ \mathbf{Walk}(x)) : s|np}$$

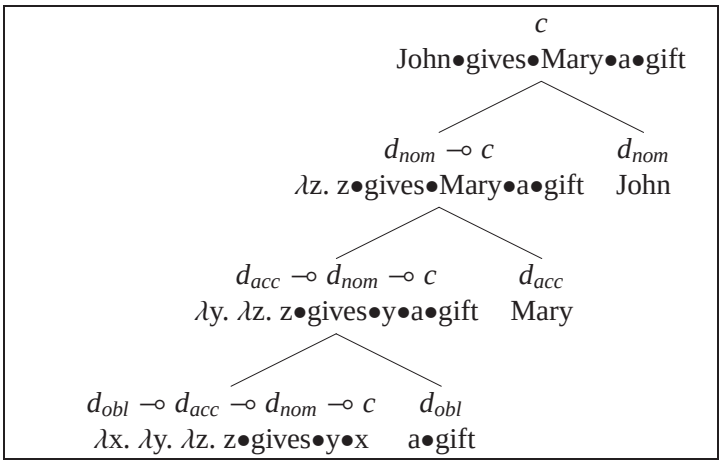
Notons que la dérivation de la lecture libre souffre malheureusement d'un manque d'élégance et de lisibilité, notamment à cause de l'utilisation de la règle complexe $|I$ qui introduit une formule arbitraire p au cours du raisonnement hypothétique effectué.

5.2.2 Traitement du liage dans $\mathcal{GLE}$

Les différents traitements du phénomène de liage réalisés dans les systèmes bi-directionnels ont nécessité l'extension de la logique de base par des connecteurs supplémentaires (opérateurs de discontinuité, lieurs etc) qui sont généralement difficiles à manipuler. A la différence de ces modèles, $\mathcal{GLE}$ offre la possibilité de rendre compte de ce phénomène de manière intuitive, élégante et facile à appréhender. Nous nous proposons dans ce qui suit de décrire les diverses techniques mises en œuvre pour traiter les réflexifs et les pronoms personnels dans le cadre de $\mathcal{GLE}$.

Encodage logique de la relation de c-commande

La relation de c-commande est centrale dans la théorie de liage formulée par Chomsky [30]. En effet, il est stipulé dans cette dernière qu'un pronom (réflexif ou non) ne peut être lié qu'à un antécédent qui le c-commande. Il est donc primordial de simuler cette relation dans un cadre logique si on souhaite rendre compte de ladite théorie dans $\mathcal{GLE}$.
Dans la littérature, certaines approches dont les grammaires de Montague [73] utilisent une relation parallèle à celle de c-commande dite de *F-commande* qui est entièrement exprimée en terme d'ordre de sous-catégorisation (i.e., ordre des arguments) dans un foncteur. Ainsi, un argument d'un foncteur F-commande tous les autres arguments de ce même foncteur qui ont été introduits antérieurement ainsi que leurs sous-constituants. Un tel encodage de la relation de c-commande s'avère être bien adapté à notre modèle comme l'illustre la structure arborescente ci-dessous. En effet, dans cette dérivation arborescente, le groupe nominal '*John*' F-commande les deux objets du verbe di-transitif '*gives*' car ils sont tous les trois *coarguments* (arguments du même foncteur) et ledit sujet suit les deux compléments dans l'ordre de sous-catégorisation. D'autre part, l'objet direct '*Mary*' F-commande l'objet indirect '*a gift*' car il fusionne avec la tête lexicale '*gives*' après ce dernier.



La relation de F-commande traduit bien la notion de c-commande dans le formalisme $\mathcal{GLE}$ puisque l'ordre de sous-catégorisation d'un foncteur est fixé par le lexique et ne peut être modifié au cours d'une dérivation étant donné l'absence de la règle d'introduction du connecteur $\multimap$. Soulignons que cette relation n'est point pertinente pour les grammaires munies d'un calcul syntaxique qui ne conserve pas la structure hiérarchique des constituants (e.g., calcul de Lambek, Grammaires Catégorielles Abstraites, ...). En effet, ces derniers systèmes légitiment la modification de l'ordre

initial des arguments d'un foncteur en utilisant la technique du raisonnement hypothétique[6].

Traitement des réflexifs

Dans notre approche, nous considérons que les réflexifs se comportent comme des clitiques qui s'agglutinent autour du verbe et identifient deux de ses arguments. Ainsi, le partage de ressources nécessaire entre un réflexif et son antécédent est entièrement encodé par la sémantique lexicale de cette anaphore qui prend la forme du λ-terme non-linéaire suivant[7] :

$$\lambda P_\lambda.\ \lambda x_1. \ldots \lambda x_n.\ \ P_\lambda(x_i,\ x_1,\ \ldots\ x_i,\ \ldots,\ x_n) :\ (e \to e \to \ldots \to t) \to e \to \ldots. \to t$$

Cette philosophie est en phase avec les travaux de Szabolcsi [104], Moortgat [75] et Morrill [83, 84] et diffère substantiellement de la démarche suivie par Hepple et Jäger qui consiste à refléter l'utilisation multiple de ressources sur la syntaxe. Il est intéressant de souligner que l'approche adoptée est fondée sur une contrainte structurelle stipulant que le verbe doit d'abord se combiner avec le réflexif en jeu avant de fusionner avec son antécédent (e.g., l'antécédent F-commande le réflexif). Cette contrainte, exprimée en terme d'ordre de sous-catégorisation, peut être vue comme la traduction logique de la relation de *c-commande* qui est au cœur de la théorie de liage de Chomsky [30].

Dans un premier temps, nous nous focaliserons sur l'étude de certains réflexifs de la langue anglaise représentant des arguments immédiats du verbe (e.g., objet direct, objet indirect). Ainsi, pour exprimer la dépendance d'un réflexif avec le sujet, nous introduisons les deux entrées lexicales *libres* suivantes (où *x-self* dénote le réflexif en jeu, e.g., *himself, herself* ...) :

$e_1 \quad \vdash \begin{pmatrix} \lambda P_\Phi.\ \lambda x_\Phi.\ P_\Phi(x\text{-}self, x_\Phi) \\ \lambda P_\lambda.\ \lambda x_\lambda.\ P_\lambda(x_\lambda,\ x_\lambda) \end{pmatrix} : (d_{acc} \multimap d_{nom} \multimap c) \multimap d_{nom} \multimap c$
$e_2 \quad \vdash \begin{pmatrix} \lambda P_\Phi.\ \lambda x_\Phi.\ \lambda y_\Phi.\ P_\Phi(x\text{-}self,\ x_\Phi,\ y_\Phi) \\ \lambda P_\lambda.\ \lambda x_\lambda.\ \lambda y_\lambda.\ P_\lambda(y_\lambda,\ x_\lambda,\ y_\lambda) \end{pmatrix} : (d_{dat} \multimap d_{acc} \multimap d_{nom} \multimap c) \multimap d_{acc} \multimap d_{nom} \multimap c$

FIG. 40: Entrées lexicales pour les réflexifs dépendants du sujet

Nous avons besoin de deux entrées décrivant les différentes positions structurelles pouvant être occupées par le réflexif, à savoir la position d'objet direct (entrée e_1) et celle d'objet indirect (entrée e_2). Ainsi, l'entrée e_1 peut être utilisée pour dériver de manière directe les exemples (22a, 22b, 22c, 22d) tandis que l'entrée e_2 permet d'analyser aisément l'énoncé (22e).

(22) a. John$_i$ likes himself$_i$.
b. John$_i$ considers himself$_i$ deaf.
c. Everyone$_i$ depends on himself$_i$.
d. Everyone$_i$ believes himself$_i$ to be the best.
e. John$_i$ shows Mary himself$_i$.

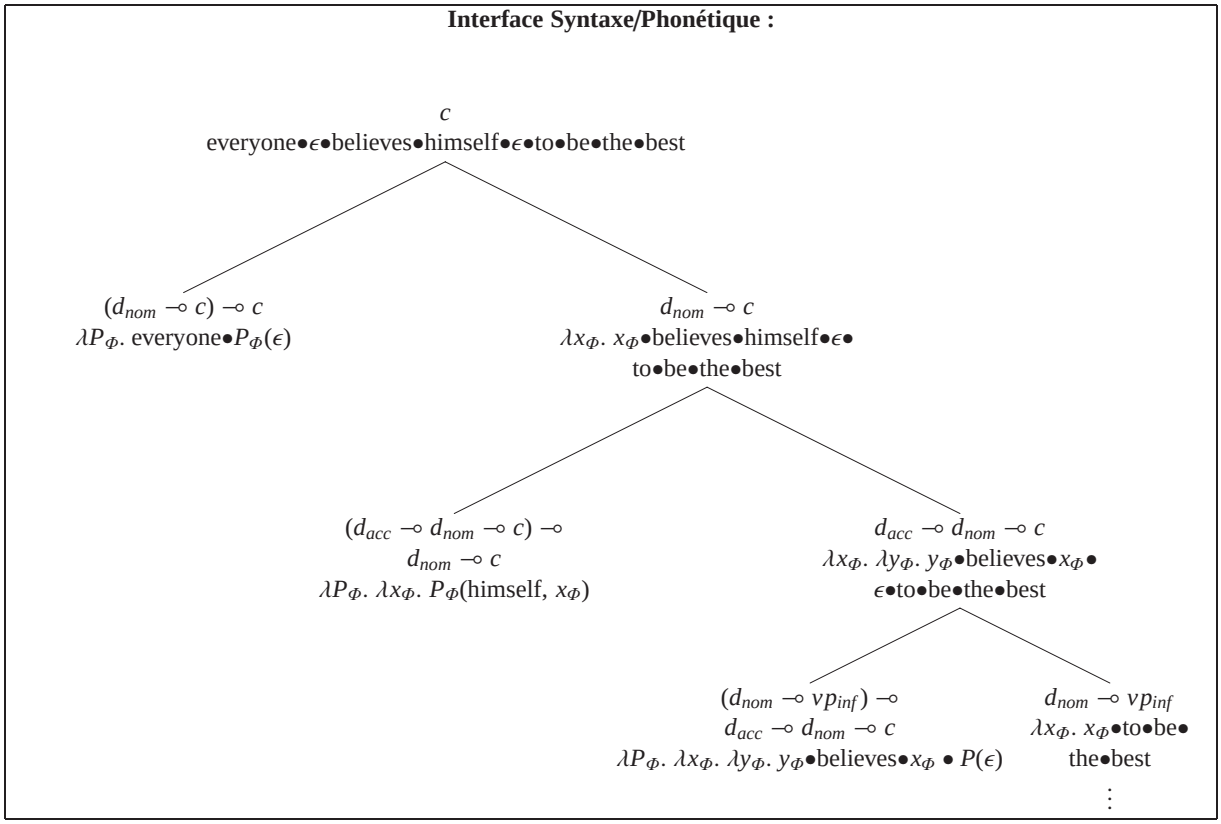
[6] A titre d'exemple, le séquent $d_{acc} \multimap d_{nom} \multimap c \vdash d_{nom} \multimap d_{acc} \multimap c$ est dérivable en logique linéaire

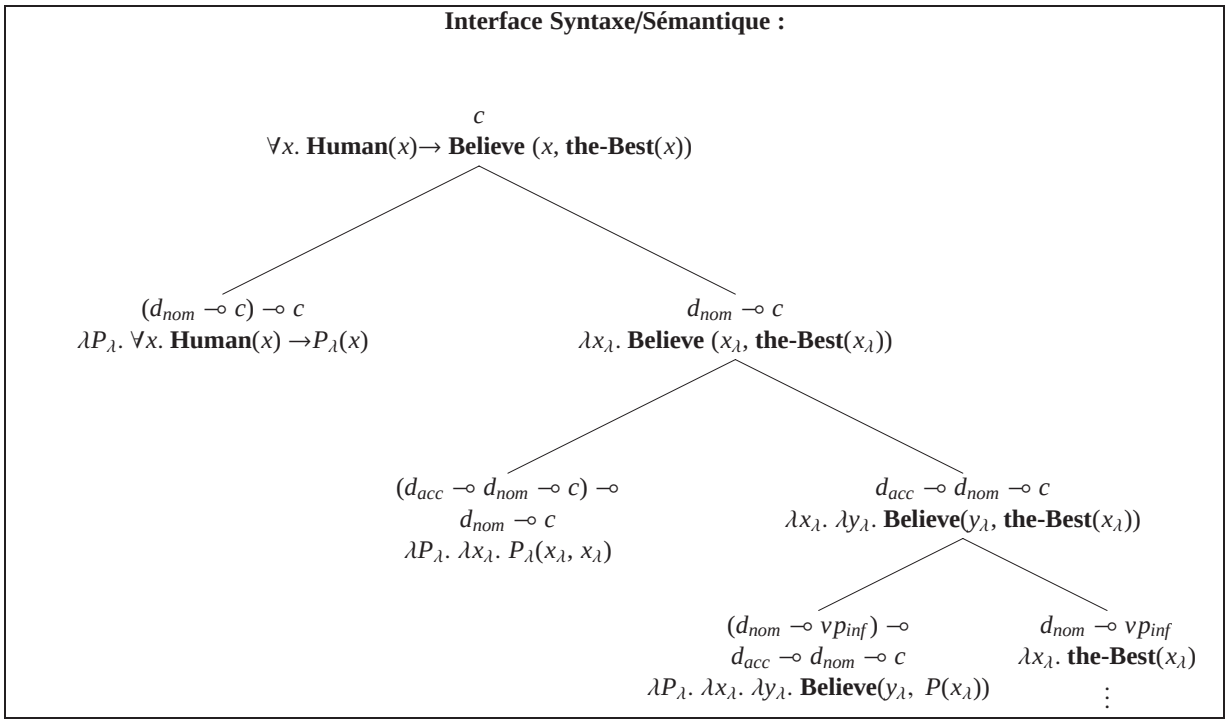
[7] Cette sémantique est adéquate lorsque le réflexif est un argument immédiat du verbe.

Nous nous intéresserons, dans ce qui suit, à la dérivation de l'énoncé le plus complexe de la liste précédente à savoir (22d). Dans cette phrase, le réflexif '*himself*' doit nécessairement être lié au sujet du verbe '*believes*' qui s'avère être un quantificateur généralisé. Même si ce réflexif porte le trait accusatif, il est sémantiquement considéré comme le sujet du groupe verbal infinitif '*to be the best*' avec lequel il forme une petite clause (i.e., *small clause*). En tenant compte de cette remarque, nous attribuons au verbe '*believes*' l'entrée lexicale suivante :

$$\vdash \left(\begin{array}{c} \lambda P_{\Phi}.\ \lambda x_{\Phi}.\ \lambda y_{\Phi}.\ y_{\Phi} \bullet \mathit{believes} \bullet x_{\Phi} \bullet P_{\Phi}(\epsilon) \\ \lambda P_{\lambda}.\ \lambda x_{\lambda}.\ \lambda y_{\lambda}.\ \mathbf{Believe}(y_{\lambda},\ P_{\lambda}(x_{\lambda})) \end{array} \right) : (d_{nom} \multimap vp_{inf}) \multimap d_{acc} \multimap d_{nom} \multimap c$$

D'après l'entrée ci-dessus, le verbe '*believes*' commence par se combiner avec le groupe verbal infinitif pour former un constituant discontinu qui a le même type abstrait qu'un verbe transitif (i.e., V_2=$d_{acc} \multimap d_{nom} \multimap c$). Le réflexif peut ainsi intégrer la dérivation par l'intermédiaire de l'entrée e_1, chose qui permet d'identifier le sujet du verbe principal avec celui du groupe verbal infinitif. Les étapes principales de la dérivation de l'énoncé (22d) sont récapitulées dans les deux figures ci-après :





En outre, notre formalisme offre la possibilité de rendre compte des réflexifs dépendants de l'objet direct de manière systématique et ce à la différence des travaux réalisés dans les logiques bi-directionnelles exigeant l'utilisation des opérateurs complexes de discontinuité [86, 83]. En effet, dans notre logique non-directionnelle, les verbes di-transitifs portent le type abstrait $V_3{=}d_{dat} \multimap d_{acc} \multimap d_{nom} \multimap c$ où l'objet indirect précéde l'objet direct dans l'ordre de sous-catégorisation[8]. Un tel type nous permet de déduire une nouvelle entrée dédiée aux réflexifs liés à l'objet :

$$e_3 \quad \vdash \left(\begin{array}{c} \lambda P_\Phi.\ \lambda x_\Phi.\ \lambda y_\Phi.\ P_\Phi(\textit{x-self},\ x_\Phi,\ y_\Phi) \\ \lambda P_\lambda.\ \lambda x_\lambda.\ \lambda y_\lambda.\ P_\lambda(x_\lambda,\ x_\lambda,\ y_\lambda) \end{array} \right) : (d_{dat} \multimap d_{acc} \multimap d_{nom} \multimap c) \multimap d_{acc} \multimap d_{nom} \multimap c$$

FIG. 41: Entrée lexicale dédiée aux réflexifs liés aux objets directs

(23) a. John talked to Bob$_i$ about himself$_i$.

b. *John talked to himself$_i$ about Bob$_i$.

L'entrée e_3 est bien adaptée à l'analyse de l'énoncé (23a) comme le montrent les deux arbres de dérivation de la figure 42. La seconde lecture de la phrase ambigüe '*John talked to Bob about himself*', où le réflexif est identifié au sujet '*John*', peut être déduite de la même façon grâce à l'utilisation de l'entrée lexicale e_2. Soulignons que dans ces deux cas, le verbe prépositionnel '*talk to ... about*' représente un constituant discontinu qui est modélisé dans notre formalisme par une seule entrée lexicale (cf. section 5.1.2).

[8] A l'opposé du type bi-directionnel inadéquat $(d_{nom} \backslash c)/d_{dat}/d_{acc}$.

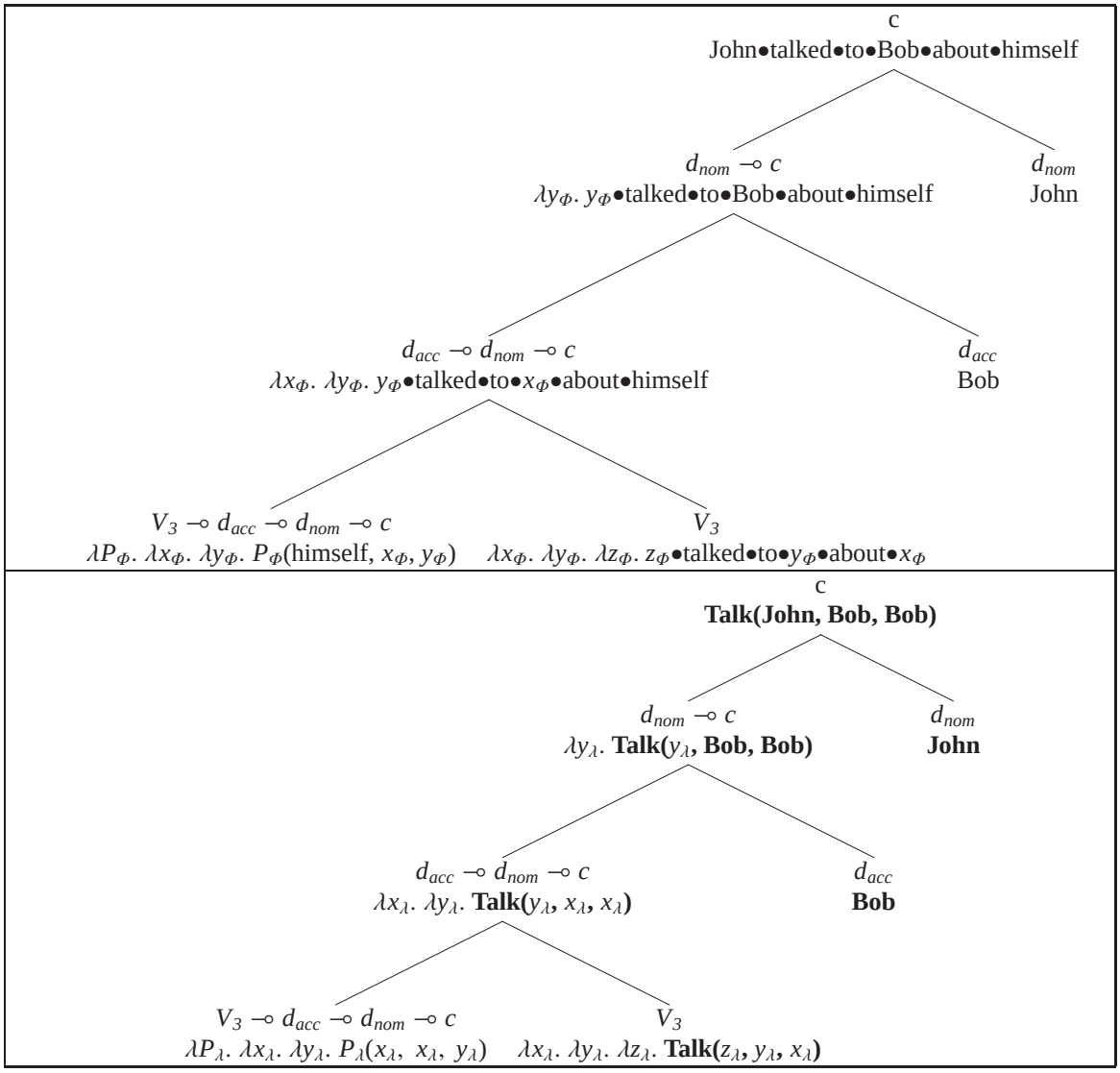


FIG. 42: Dérivation de l'énoncé 23a en utilisant l'entrée e_3

Il est intéressant de noter que le type abstrait V_3 attribué aux verbes di-transitifs bloque la conception d'une entrée lexicale liant le réflexif à un objet indirect (e.g., énoncé (23b)) puisque ce dernier sera le premier argument à fusionner avec le verbe en question. Ce blocage est dû à l'impossibilité de modifier l'ordre des arguments d'un foncteur dans le cadre de notre logique étant donné qu'elle n'est pas dotée d'un libre accès à la technique du raisonnement hypothétique.
D'autre part, l'encodage des réflexifs de la langue anglaise moyennant des entrées lexicales libres permet de respecter systématiquement la contrainte de localité énoncée par le principe *A*. En effet, les entrées libres e_1, e_2 et e_3 sont conçues pour forcer le réflexif à se combiner avec un constituant verbal qui est lexical en quelque sorte étant donné l'absence de la règle d'axiome logique ainsi que celle d'introduction de l'implication $\multimap$ dans la logique inhérente à $\mathcal{GLE}$. Ce constituant provient directement du lexique ou résulte de la fusion d'un verbe lexical avec quelques-uns de ses arguments. L'interdiction de la coréférence à longue distance est par conséquent obtenue sans efforts, à la différence des systèmes bi-directionnels décrits précédemment [54, 83]. Ainsi, l'analyse de l'énoncé (24a) ne pose aucun problème puisque le réflexif fusionne avec un élément lexical coïncidant avec le verbe transitif '*likes*'.

(24) a. John thinks Bob_i likes himself_i.
b. $^*\text{John}_i$ thinks Bob likes himself_i.
c. $^*\text{John}_i$ respects the man who likes himself_i.

En revanche, aucune analyse n'est plausible pour les énoncés (24b, 24c). En effet, il est impossible de combiner l'anaphore '*himself*' avec l'un des constituants composés '*thinks Bob likes*' ou '*respects the man who likes*' puisqu'on est incapable d'attribuer à ces derniers le type syntaxique abstrait $d_{acc} \multimap d_{nom} \multimap c$. Une telle opération nécessite impérativement le recours au raisonnement hypothétique (cf. figure 43) qui n'est pas librement accessible dans notre logique.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathit{Bob} : d_{nom}}\,Lex \quad \dfrac{\dfrac{}{\vdash \lambda x.\,\lambda y.\, y \bullet \mathit{likes} \bullet x : d_{acc} \multimap d_{nom} \multimap c}\,Lex \quad \dfrac{}{x_\Phi : d_{acc} \vdash x_\Phi : d_{acc}}\,Ax}{x_\Phi : d_{acc} \vdash \lambda y.\, y \bullet \mathit{likes} \bullet x_\Phi : d_{nom} \multimap c}\multimap E}{x_\Phi : d_{acc} \vdash \mathit{Bob} \bullet \mathit{likes} \bullet x : c}\multimap E}{x_\Phi : d_{acc} \vdash \lambda y_\Phi.\, y_\Phi \bullet \mathit{thinks} \bullet \mathit{Bob} \bullet \mathit{likes} \bullet x_\Phi : d_{nom} \multimap c}\multimap E}{\vdash \lambda x_\Phi.\,\lambda y_\Phi.\, y_\Phi \bullet \mathit{thinks} \bullet \mathit{Bob} \bullet \mathit{likes} \bullet x_\Phi : d_{acc} \multimap d_{nom} \multimap c}\multimap I$$

FIG. 43: Dérivation valide dans la logique linéaire mais illégitime dans $\mathcal{GLE}$

Le principe *A* de la théorie du liage n'est pas universel. En effet, certaines langues naturelles comportent des anaphores dites *à longue distance* qui peuvent être liées à des antécédents éloignés se trouvant à l'extérieur de leur domaine local. Ceci est le cas, à titre d'exemple, du réflexif *zi ji* de la langue chinoise, illustré par l'énoncé (25) issu de [28].

(25) Zhangsan_k renwei Lisi_j zhidao Wangwu_i xihuan $\text{ziji}_{i/j/k}$
Zhangsan renwei Lisi knows Wangwu likes self
'Zhangsan thinks Lisi knows that Wangwu likes himself'

Dans cet énoncé, le réflexif '*ziji*' peut coréférer aussi bien avec le nom propre local '*Wangwu*' qu'avec l'un des deux sujets éloignés '*Zhangsan*' et '*Lisi*'.

Comme nous l'avons précédemment vu, l'encodage des anaphores sous forme d'entrées lexicales libres force ces dernières à se combiner avec des constituants simples. Afin de prendre en charge les réflexifs à longue distance, nous avons besoin de recourir au raisonnement hypothétique, chose qui se fait uniquement en présence d'hypothèses contrôlées. Ainsi, nous proposons de modéliser ce genre de réflexifs par l'intermédiaire d'entrées liées telles que l'entrée e_4 présentée ci-après :

$$e_4 \quad \vdash \left(\begin{array}{c} \lambda P_\Phi.\ \lambda y_\Phi.\ P_\Phi(refl, y_\Phi) \\ \lambda P_\lambda.\ \lambda y_\lambda.\ P_\lambda(y_\lambda,\ y_\lambda) \end{array} \right) : (d_{acc} \multimap d_{nom} \multimap c) \multimap d_{nom} \multimap c \multimap [(x_\Phi,\ x_\lambda) : d_{acc} \vdash (x_\Phi,\ x_\lambda) : d_{acc}]$$

Fig. 44: Entrée lexicale pour les réflexifs à longue distance

L'entrée e_4 stipule qu'un réflexif à longue distance occupant la position d'objet direct se combine avec un groupe verbal (de type $d_{nom} \multimap c$) contenant une hypothèse contrôlée de type d_{acc} au lieu de fusionner directement avec une composante de type $d_{acc} \multimap d_{nom} \multimap c$ comme c'était le cas pour les entrées e_1, e_2 et e_3. Cette combinaison se fait par l'intermédiaire de la règle logique hybride $\multimap IE_c$ qui englobe une étape d'abstraction de la ressource hypothétique impliquée. L'application de cette règle permet d'identifier la sémantique du réflexif à celle du sujet attendu par le groupe verbal en jeu. Ainsi, en présence de plusieurs antécédents potentiels, différentes lectures peuvent être obtenues en fonction de l'instant où la règle $\multimap IE_c$ est appliquée. A titre d'exemple, la figure 45 montre les étapes principales de la dérivation de l'énoncé (44) produisant la coréférence du réflexif '*ziji*' avec le premier sujet enchassé '*Lisi*'. Cette lecture est obtenue en appliquant la règle logique $\multimap IE_c$ au niveau du groupe verbal composé '*zhidao Wangwu xihuan ziji*'. Les deux autres interprétations de l'énoncé (44) peuvent être engendrées de la même façon en anticipant ou en retardant le déchargement de l'hypothèse contrôlée.

L'étude proposée jusqu'à présent s'intéresse spécialement aux cas les plus fréquents où les réflexifs sont des arguments immédiats du verbe (objet direct ou indirect). Les exemples (26a, 26b, 26c) ci-après montrent qu'il est possible pour ces anaphores d'occuper des positions structurelles enchassées, soit dans l'un des arguments du verbe (cf. énoncé 26a, 26b), soit dans un adjoint (e.g., complément circonstanciel de lieu, cf. énoncé 26c).

(26) a. John$_i$ likes the picture of himself$_i$.
b. John$_i$ likes the picture of himself$_i$ taken in Bordeaux.
c. John$_i$ saw a snake near himself$_i$.
d. *Bob$_i$ thinks that John likes the picture of himself$_i$.
e. *Mary will introduce brothers of John$_i$ to himself$_i$.

A priori, l'entrée lexicale liée e_4, qui exige l'introduction d'une hypothèse contrôlée de type d_{acc}, suffit pour dériver les trois premiers énoncés. Toutefois, l'utilisation de cette entrée flexible fait sombrer le système dans une sur-génération non souhaitable en légitimant le liage de l'anaphore à un sujet se trouvant à l'extérieur de son domaine local comme c'est le cas pour l'énoncé (26d). Afin de pallier à ce problème, nous proposons de raffiner la définition de cette entrée de façon à exiger l'abstraction anticipée de l'hypothèse contrôlée. Ainsi, au lieu de décharger ladite hypothèse au niveau du syntagme verbal, l'abstraction de cette ressource hypothétique se fera à un niveau plus élémentaire coïncidant avec le constituant où le réflexif est enchassé (i.e., objet direct ou indirect,

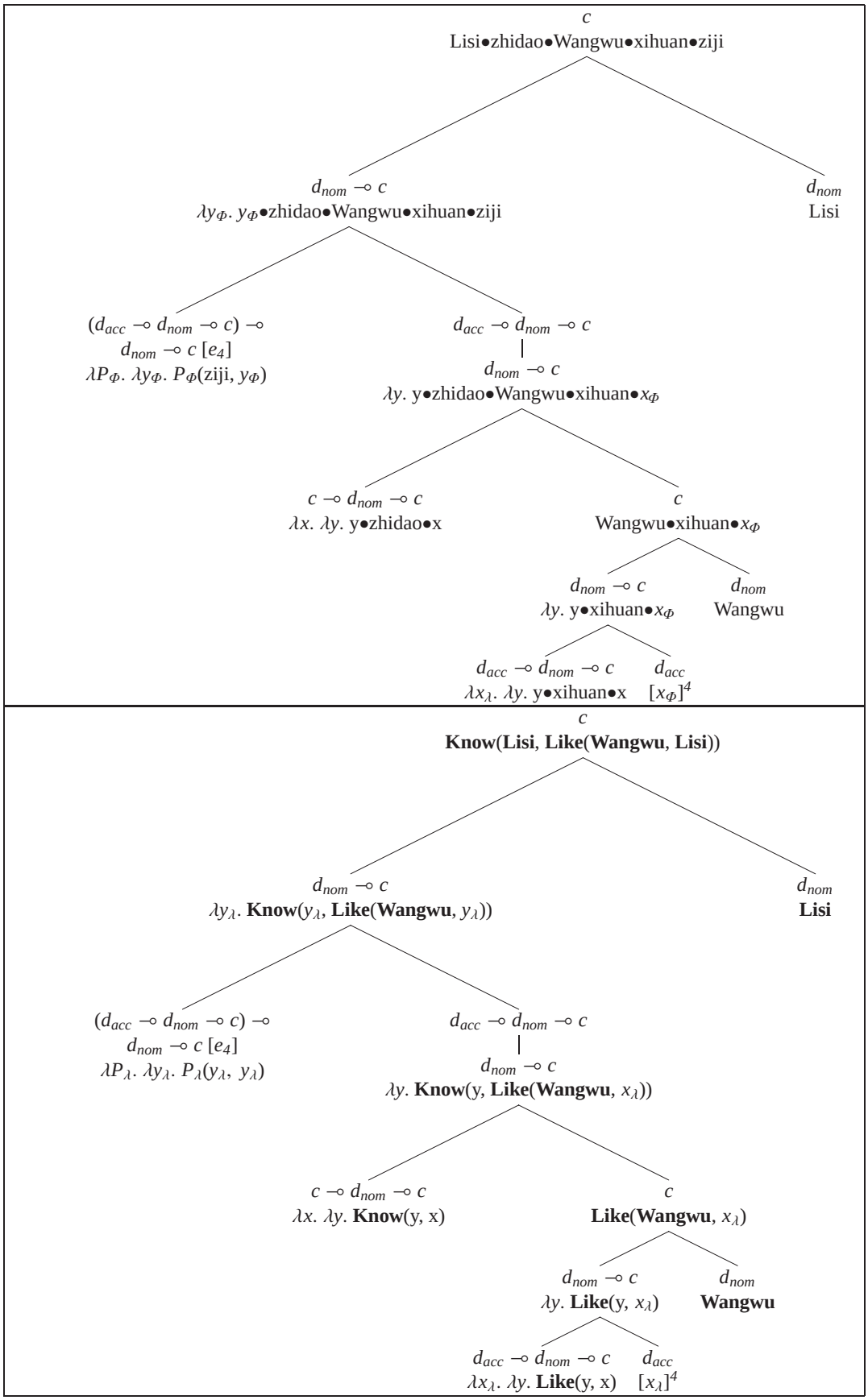


FIG. 45: Liage d'une anaphore à longue distance

adjoint). Dans le cas des réflexifs orientés sujet, nous distinguons entre deux entrées lexicales raffinées qui dépendent du type de la composante englobant l'anaphore en jeu. Ces deux entrées sont présentées ci-après : (où $V_1{=}d_{nom} \multimap c$, $V_2{=}d_{acc} \multimap V_1$, $d_{cas} \in \{d_{acc}, d_{obl}\}$ et $X{=}(x_\Phi,\ x_\lambda)$) :

e_5	$\vdash \begin{pmatrix} \lambda P_\Phi.\ \lambda Q_\Phi.\ \lambda x_\Phi.\ Q_\Phi(P_\Phi(x\text{-}self),\ x_\Phi) \\ \lambda P_\lambda.\ \lambda Q_\lambda.\ \lambda x_\lambda.\ Q_\lambda(P_\Phi(x_\lambda),\ x_\lambda) \end{pmatrix} : (d_{acc} \multimap d_{cas}) \multimap V_2 \multimap V_1 \multimap [X : d_{acc} \vdash X : d_{acc}]$
e_6	$\vdash \begin{pmatrix} \lambda P_\Phi.\ \lambda Q_\Phi.\ \lambda x_\Phi.\ P_\Phi(x\text{-}self,\ Q_\Phi,\ x_\Phi) \\ \lambda P_\lambda.\ \lambda Q_\lambda.\ \lambda x_\lambda.\ P_\lambda(x_\lambda,\ Q_\lambda,\ x_\lambda) \end{pmatrix} : (d_{acc} \multimap V_2 \multimap V_2) \multimap V_2 \multimap V_2 \multimap [X : d_{acc} \vdash X : d_{acc}]$

Fig. 46: Entrées lexicales pour les réflexifs enchassés

L'entrée liée e_5 modélise le comportement des réflexifs enchassés dans un complément d'objet du verbe (de type d_{cas}) alors que l'entrée e_6 s'applique en présence de réflexifs encastrés dans des adjoints qui jouent le rôle syntaxique de modifieurs de groupes verbaux (de type $V_2 \multimap V_2$). A titre d'exemple, l'utilisation de l'entrée e_5 est illustrée par la dérivation de l'énoncé (26a) présentée dans la figure 47. Dans un but de concision, nous représentons la sémantique de l'article '*the*' moyennant l'opérateur de description définie

ι[9], où $\iota(\lambda x.\ P(x))$ (noté usuellement $\iota x.\ P(x)$) dénote l'unique individu vérifiant la propriété P [26]. D'autre part, nous utilisons une relation binaire **Of** qui est vérifiée par deux individus ssi le premier possède le second dans un certain sens.
En suivant les étapes de déduction à partir des feuilles, nous remarquons que le déchargement de l'hypothèse contrôlée se fait immédiatement après la construction du complément d'objet où le réflexif '*himself*' sera enchassé. Quand l'anaphore retrouve sa position finale, l'objet direct ainsi construit (i.e., '*the picture of himself*') se comporte syntaxiquement de la même façon que l'entrée e_1 en permettant uniquement la combinaison avec des verbes transitifs simples. Notre entrée raffinée introduit ainsi l'utilisation du raisonnement hypothétique de manière contrainte qui bloque toute tentative de liage à longue distance comme c'est le cas pour l'énoncé (26d).

Finalement, il est intéressant de noter que même s'il est permis à un réflexif d'occuper une position imbriquée, il est interdit pour l'antécédent d'être enchassé dans un des arguments du verbe, comme l'illustre l'agrammaticalité de l'énoncé (26e). Notre approche respecte systématiquement cette contrainte puisque la sémantique lexicale de toutes les entrées conçues identifie l'anaphore avec un argument direct du prédicat mis en jeu.

Traitement des pronoms personnels

A la différence des réflexifs, les dépendances entre les pronoms personnels et leurs antécédents sont non bornées. En outre, ces derniers peuvent apparaître dans des contextes arbitraires et être séparés par des constituants de diverses natures comme le montrent les exemples ci-après :

(27) a. John$_i$ thinks he$_{i/j}$ is smart.
b. John$_i$ likes the book that he$_{i/j}$ wrote.
c. *He$_i$ thinks John$_i$ is smart.

[9] L'opérateur ι porte le type sémantique (e→t)→ e.

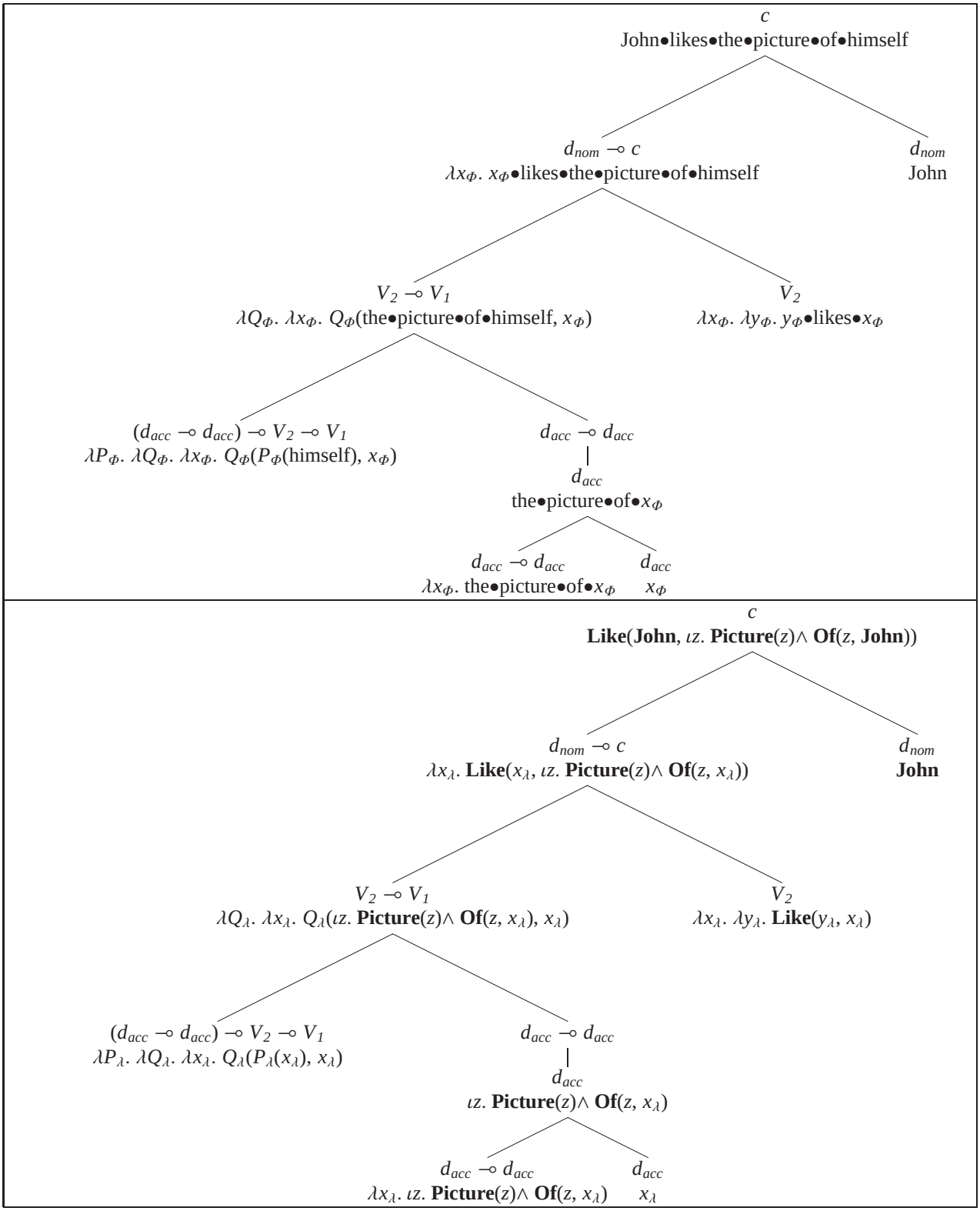


FIG. 47: Dérivation simplifiée de l'énoncé 26a

Ainsi, au lieu d'exprimer le partage de ressources entre un pronom et son antécédent par le seul intermédiaire de la sémantique lexicale dudit pronom, nous optons pour la réflexion de cette non-linéarité sur la syntaxe abstraite grâce à l'utilisation de l'exponentielle.
Dans la présente étude, nous nous interésserons particulièrement à l'emploi anaphorique des pronoms, leur utilisation libre peut être exprimée aisément par une entrée indépendante telle que la suivante (où la sémantique du pronom *he* prend la forme d'une variable libre x_λ qui sera instanciée au niveau du discours) :

$$\vdash \left(\begin{array}{c} he \\ x_\lambda \end{array} \right) : d_{nom}$$

En outre, nous nous focaliserons uniquement sur les exemples grammaticaux qui sont soumis aux principes de la théorie de liage exigeant du pronom d'être c-commandé par son antécédent.
Par analogie avec les travaux de Kayne [64], nous considérons dans notre approche que le pronom et son antécédent constituent une seule unité lexicale encodée sous la forme d'une entrée non-linéaire. Nous supposons que le lexique est doté d'une règle de production permettant de former une entrée non-linéaire à partir de chaque entrée (linéaire ou libre) associée à un groupe nominal ou un déterminant en lui rattachant un pronom adéquat. Ainsi à titre d'exemple, nous formalisons le constituant double [John, he] de Kayne [64] par l'entrée liée ci-après :

$$e_1 \quad \vdash \left(\begin{array}{c} \lambda P_\Phi.\ John \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda.\ P_\lambda(\mathbf{John}) \end{array} \right) : (!d_{nom} \multimap c) \multimap c \ \not\ni \ \begin{array}{l} ([(x_{\Phi 1}, x_{\lambda 1}) : d_{nom} \vdash (x_{\Phi 1}, x_{\lambda 1}) : d_{nom}], \\ [(x_{\Phi 2}, x_{\lambda 2}) : d_{nom} \vdash (he, x_{\lambda 2}) : d_{nom}]) \end{array}$$

Cette entrée est liée à deux hypothèses contrôlées, chose qui justifie la présence de " !" dans son type principal. La première hypothèse est réalisée par un axiome logique qui servira à occuper le site de l'antécédent. Par contre, la seconde hypothèse est modélisée par un axiome extra-logique représentant le pronom '*he*' qui porte une forme phonétique prononcée. Le phénomène de liage résulte de la contraction de ces deux hypothèses de contrôle suivie de leur déchargement simultané par la tête de leur entrée associée. Une telle procédure permet d'obtenir de manière directe les lectures liées des énoncés (27a, 27b). La figure 48 illustre son application sur l'exemple (27a). Notons qu'il est indispendable de respecter l'ordre d'introduction des hypothèses contrôlées afin de satisfaire la pré-condition ‡ légitimant l'application de la règle hybride $\multimap IE_c$ (cf. Fig. 29). Par conséquent, la dernière hypothèse de la liste l_{hyps}, à savoir celle qui représente le pronom *he*, doit intégrer la dérivation en premier et occuper ainsi la position sujet de la subordonnée complétive. La première hypothèse de l_{hyps} est introduite ultérieurement, elle remplit la position sujet du verbe principal '*thinks*' qui c-commande tous les constituants de la complétive en question. L'agrammaticalité de l'énoncé (27c) qui transgresse le principe *C* est ainsi expliquée par la violation de la contrainte ‡ résultant de l'inversion de l'ordre d'introduction des hypothèses contrôlées liées à l'entrée e_1. Soulignons que la représentation arborescente de la dérivation nous permet de vérifier aisément la validité de la condition ‡ par la simple comparaison des profondeurs des feuilles respectives associées aux hypothèses contrôlées en jeu. En effet, plus une hypothèse est introduite tôt plus la distance séparant le moment de son introduction de sa phase d'abstraction est grande.

Notre approche peut facilement être adaptée pour rendre compte des adjectifs possessifs (e.g., *his*, *her*) dont le comportement partage plusieurs points communs avec celui des pronoms personnels. Afin d'analyser une phrase telle que (28a), une première solution consiste à considérer une entrée non-linéaire qui représente le constituant double [everybody, his•mother].

(28) a. Everybody$_i$ loves his$_i$ mother.

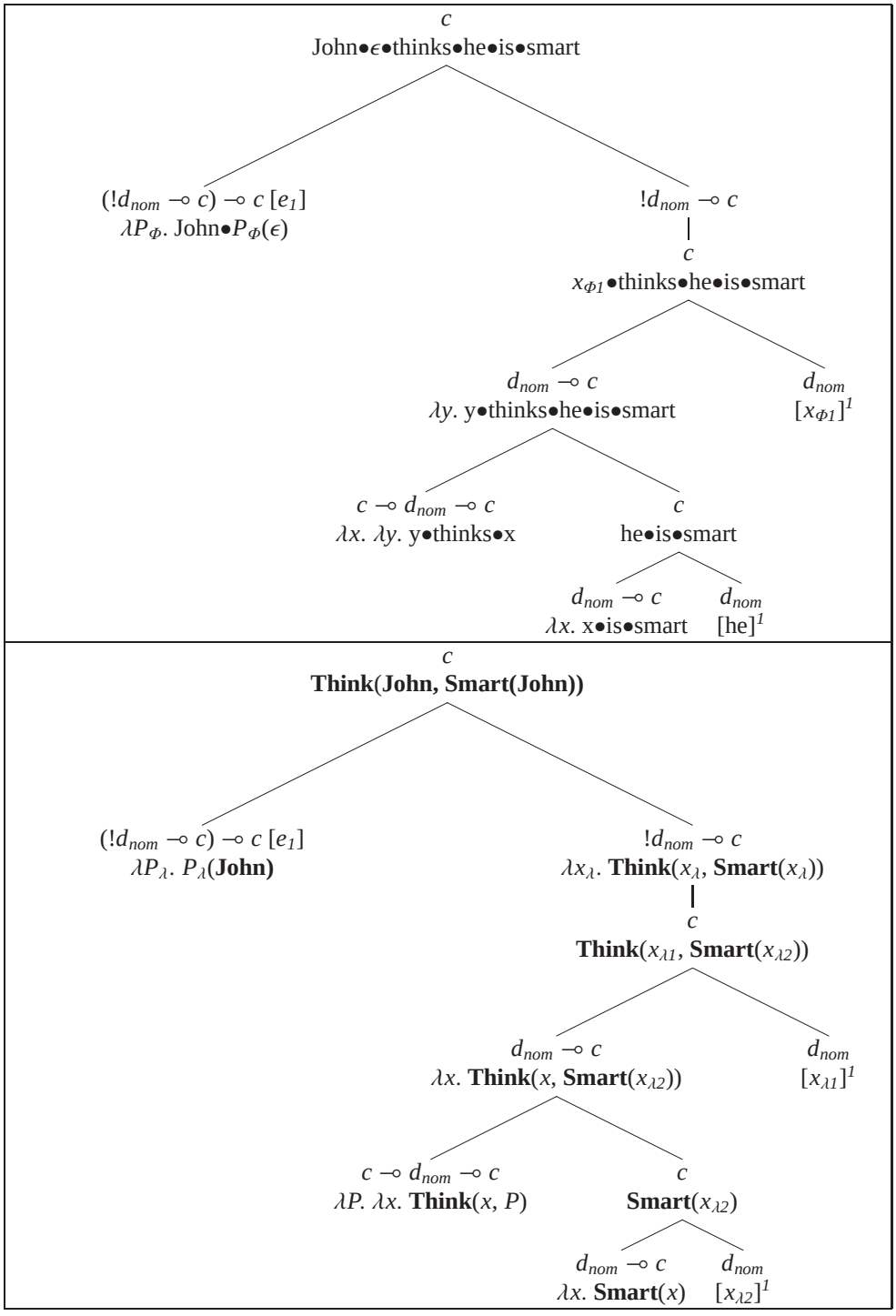


FIG. 48: Dérivation de l'exemple (27a)

b. *His$_i$ mother loves everybody$_i$.

Toutefois cette solution ne s'avère pas adéquate puisqu'elle nécessite de former exhaustivement toutes les entrées non-linéaires associées aux composants [everybody, his•x] et ce pour tout nom commun *x*. Or, il est théoriquement possible de construire une infinité de noms communs en modifiant les noms communs lexicaux par des adjectifs ou des subordonnées relatives (e.g., *his nice mother, his nice mother who likes logic*, etc). Cette première solution n'est pas réalisable puisqu'elle transgresse l'hypothèse de finitude du lexique.
Afin de résoudre ce problème, nous proposons de simuler tout adjectif possessif (e.g., *my*, *your*, *his*, *her*) par la combinaison d'un pronom génitif (e.g., *mine*, *yours*, *his*, *hers*) avec le possessif *'s* (e.g., '*his friend*' peut être vu comme une forme contractée de '*his's friend*', i.e., '*a friend of his*'). L'énoncé (28a) sera par conséquent considéré comme la forme simplifiée de (29) :

(29) Everybody$_i$ loves his$_i$'s mother.

Il est ainsi plus pertinent de considérer que l'hypothèse contrôlée attachée à '*everybody*' est réduite au pronom possessif génitif '*his*'. Ceci permet de remplacer la liste infinie de composants ∀x. [everybody, his•x] (où '*his*' dans ce cas est un adjectif possessif) par un seul constituant double [everybody, his] modélisé simplement par l'entrée e_2 ci-après :

$$e_2 \quad \vdash \left(\begin{array}{c} \lambda P_\Phi.\ everybody \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda.\ \forall x.\ \mathbf{Human}(x) \Rightarrow P_\lambda(x) \end{array} \right) : (!d \multimap c) \multimap c \multimap \begin{array}{c} ([(x_{\Phi 1}, x_{\lambda 1}) : d \vdash (x_{\Phi 1}, x_{\lambda 1}) : d_{nom}], \\ [(x_{\Phi 2}, x_{\lambda 2}) : d \vdash (his, x_{\lambda 2}) : d_{gen}]) \end{array}$$

Dès lors, on peut facilement construire une dérivation de (29) comme le montre la figure 49. Cette dérivation converge puisque la condition ‡ est remplie : l'hypothèse contrôlée associée au pronom génitif '*his*' a été introduite avant celle correspondant à son antécédent. Remarquons que le changement de cet ordre viole la contrainte ‡, ce qui permet de bloquer la dérivation de (28b).

L'étude proposée jusqu'à présent ne tient pas compte du principe *B*. En effet, il n'est pas difficile de constater que l'encodage du constituant double [John, him] de la même façon que l'entrée e_1 permettrait de lier, à tort, le pronom *him* à un antécédent se trouvant à l'intérieur de son domaine local dans les énoncés (30) ci-après.

(30) a. Bob$_j$ thinks John$_i$ likes him$_{j/*i}$.
b. *John$_i$ believes him$_i$ to be the best.
c. *John$_i$ likes him$_i$.

Afin de rendre compte de la condition *B*, Kayne suggère que le constituant double doit nécessairement se déplacer à une position intermédiaire avant de retrouver son site final [64]. D'après Kayne, cette position structurelle se trouve hiérarchiquement au-delà du sujet de la clause à laquelle appartient le pronom en jeu. Par conséquent, dans les exemples (30), la position transitoire en question se trouve au-delà du sujet '*John*', leur agrammaticalité peut ainsi être expliquée par l'impossibilité de mouvement de haut en bas.
Dans le but de formaliser la proposition de Kayne, nous avons besoin d'utiliser un axiome propre tel que $\vdash (\lambda x_\Phi.\ x_\Phi,\ \lambda x_\lambda.\ x_\lambda) : c \multimap c$ pour encoder l'hypothèse contrôlée qui représente la position transitoire délimitant le domaine local du pronom. Cet enrichissement nécessite la révision de la

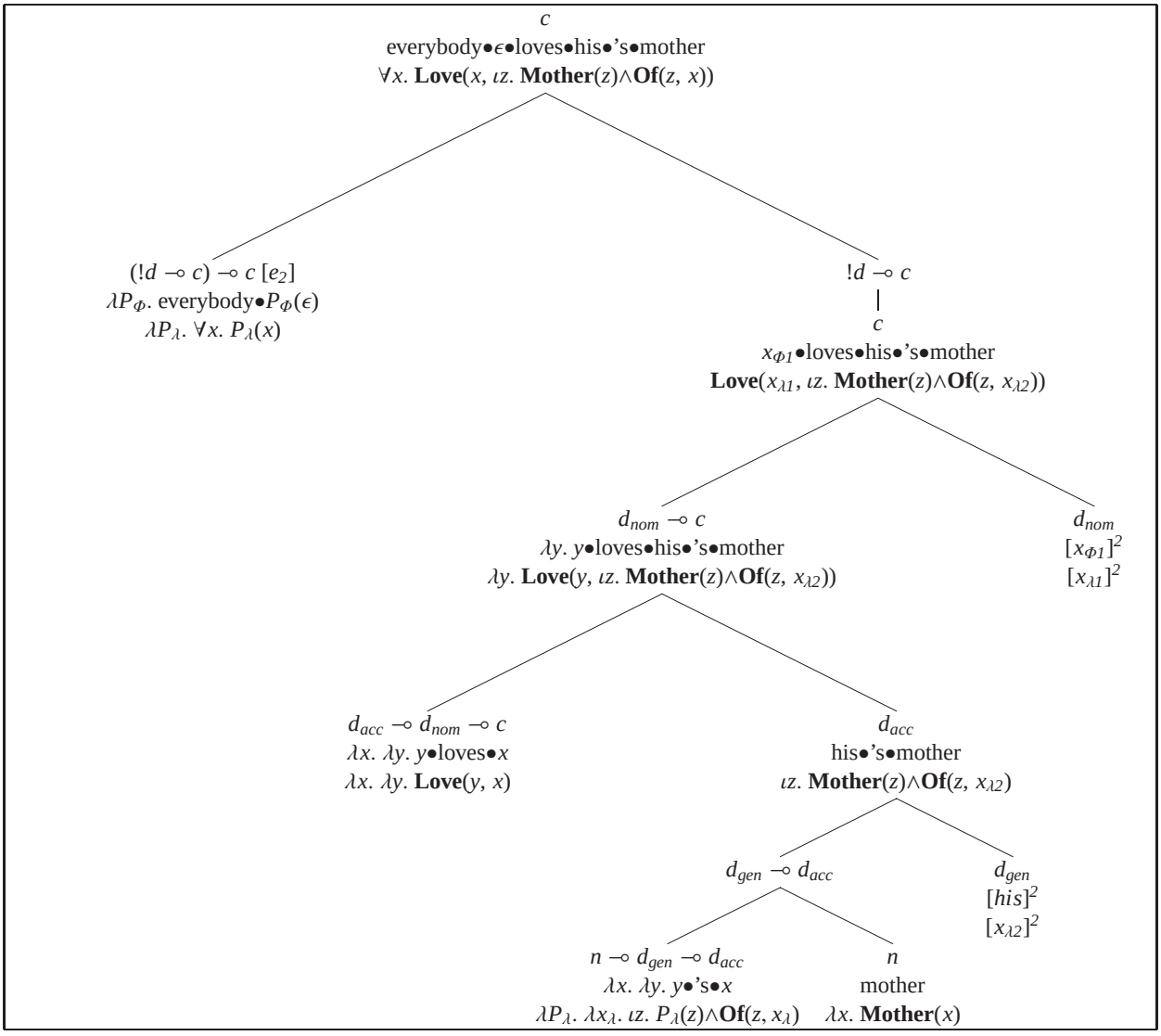


FIG. 49: Dérivation de l'énoncé (29) en utilisant l'entrée e_2

spécification initiale exigeant des hypothèses contrôlées qu'elles dépendent d'un signe variable déclaré dans le contexte. Nous proposons ainsi d'élargir le spectre de la définition 4.4.4[10] :

Définition 5.2.1 *Définition 4.4.4 révisée*
Soit e_i une entrée lexicale liée à la liste d'hypothèses l_{hyps}. Soit H un type syntaxique abstrait (i.e., le type commun pour tous les axiomes logiques contrôlés). La liste des axiomes contrôlés prend alors la forme suivante $l_{hyps}=(ax_1, ..., ax_k)$ où : $\forall i \in \{1..k\}$, $ax_i=[(x_{\Phi i}, x_{\lambda i}) : H \vdash (u_{\Phi i}, x_{\lambda i}) : H]$ ($u_{\Phi i} \in \Lambda_\Phi(\Sigma_\Phi)$ de type concret $\tau_\Phi(H)$) ou bien $ax_i=[\vdash (u_{\Phi i}, u_{\lambda i}) : H_i]$ ($H_i \in \mathcal{T}(\mathcal{A})$, $u_{\Phi i} \in \Lambda_\Phi(\Sigma_\Phi)$ de type concret $\tau_\Phi(H_i)$ et $u_{\lambda i} \in \Lambda_\lambda(\Sigma_\lambda)$ de type concret $\tau_\lambda(H_i)$).

En s'inspirant des idées de Kayne, nous encodons le constituant double [John, him] moyennant l'entrée e_3 qui est liée à trois hypothèses contrôlées : la première sert à occuper le site de l'antécédent '*John*', la seconde spécifie la position intermédiaire précédemment décrite tandis que la dernière représente le pronom personnel '*him*'.

$$e_3 \vdash \left(\begin{array}{c} \lambda P_\Phi.\ John \bullet P_\Phi(\epsilon) \\ \lambda P_\lambda.\ P_\lambda(\textbf{John}) \end{array} \right) : (!d \multimap c) \multimap c \multimap \left(\begin{array}{l} ([(x_{\Phi 1}, x_{\lambda 1}) : d \vdash (x_{\Phi 1}, x_{\lambda 1}) : d_{nom}], \\ [\vdash (\lambda y_\Phi.\ y_\Phi, \lambda y_\lambda.\ y_\lambda) : c \multimap c] \\ [(x_{\Phi 2}, x_{\lambda 2}) : d \vdash (him, x_{\lambda 2}) : d_{acc}]) \end{array} \right)$$

Notons que la position intermédiaire porte le type abstrait $c \multimap c$, elle n'apporte aucune contribution syntaxique ou sémantique étant donné que ses composantes concrètes (Φ-terme et λ-terme) coïncident avec la fonction d'identité. Le rôle essentiel de cette position est de constituer une frontière délimitant le domaine local du pronom '*him*'. Rappelons que la convergence de la dérivation est liée au respect de la contrainte ‡ qui exige des profondeurs respectives des feuilles représentant les différentes hypothèses contrôlées d'être proportionnelles à leurs rangs dans la liste l_{hyps}. Ainsi, afin de respecter cette contrainte, l'hypothèse contrôlée associée à l'antécédent (i.e., $(x_{\Phi 1}, x_{\lambda 1})$) doit être introduite en dernier, elle ne peut donc représenter le sujet de la clause qui constitue l'argument de la position transitoire. Exprimé autrement, cela signifie que la présence de ce site intermédiaire empêche le pronom '*him*' de se lier au sujet qui fait partie de sa clause locale.

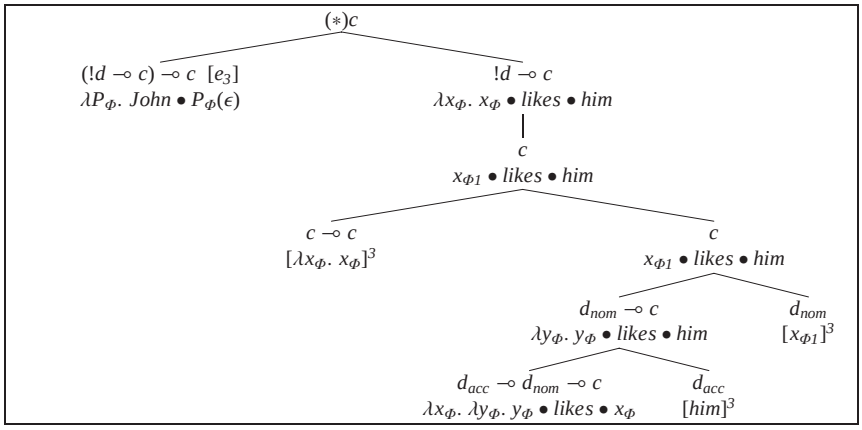


FIG. 50: Agrammaticalité de l'énoncé 30c

[10] La modification apportée à cette spécification concerne uniquement la structure des hypothèses contrôlées explicitée au début du point 2.

La figure 50 illustre la raison sous-jacente à l'agrammaticalité de l'énoncé (30c) en montrant l'impossibilité de dériver la phrase '*John likes him*' moyennant l'entrée e_3. Le blocage de la dérivation précédente est ainsi du à la violation de la contrainte ‡ puisque la profondeur de la seconde hypothèse contrôlée occupée par la position intermédiaire est inférieure à celle de la première qui représente le site de l'antécédent.

Pour les mêmes raisons précédentes, il n'est pas possible d'effectuer une dérivation de la phrase '*Bob thinks John likes him*' en utilisant l'entrée e_3 qui établit le liage entre le pronom '*him*' et le sujet local '*John*'. La figure 51 montre en revanche que le liage est totalement autorisé entre ce pronom et le sujet éloigné '*Bob*'. Cette analyse utilise une entrée liée e_3' définie de manière analogue à e_3 pour représenter le constituant double [Bob, him].

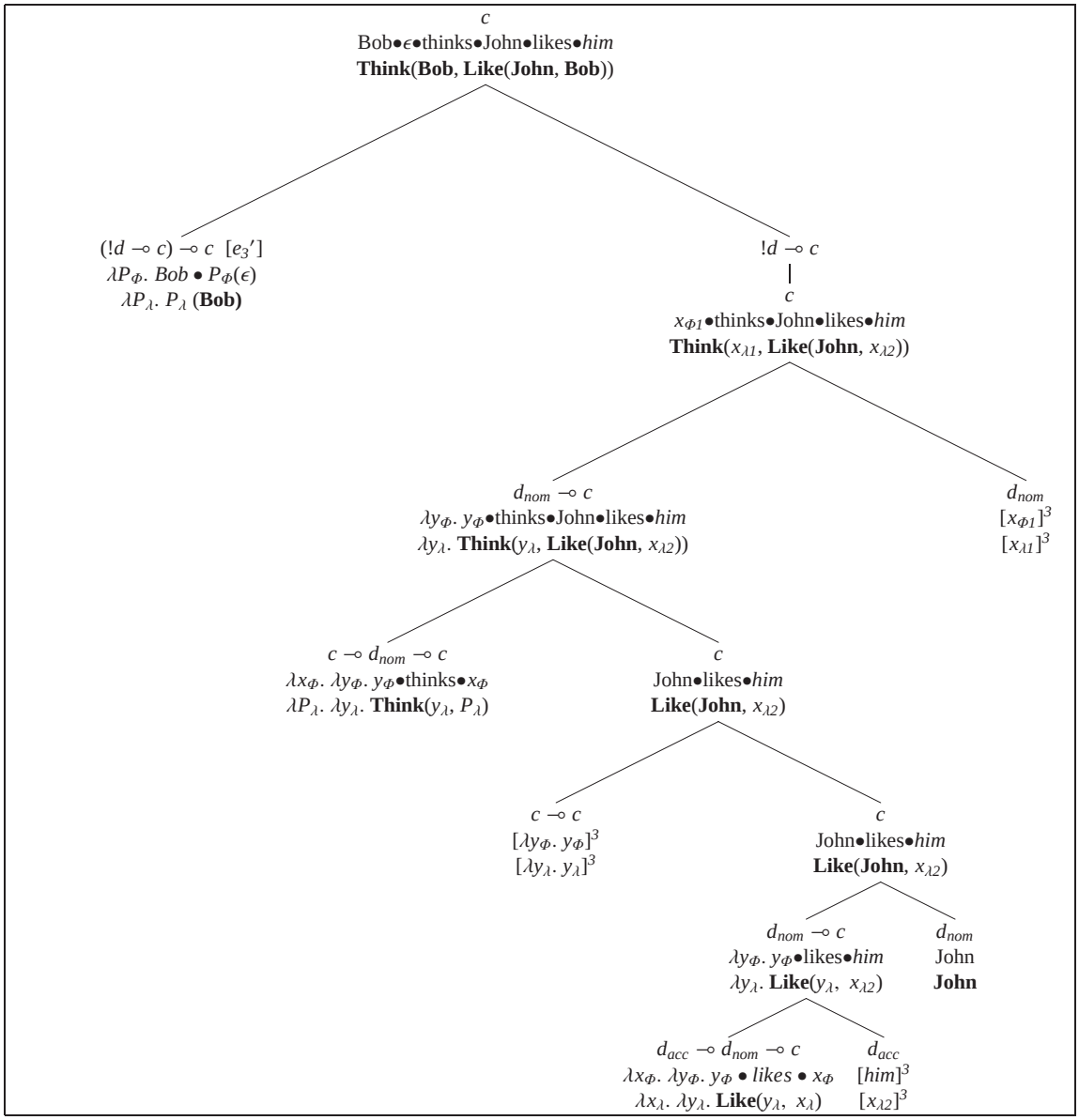


FIG. 51: Dérivation de l'énoncé (30a)

5.3 Ellipse

5.3.1 Travaux antérieurs

L'ellipse est un phénomène linguistique qui se manifeste par l'omission d'une partie d'un énoncé dans le but d'alléger la formulation. Les énoncés ci-après illustrent trois variétés de constructions elliptiques : l'ellipse du syntagme verbal (i.e., *VP-ellipse*) (31a), le *gapping* (31b) et l'*ACE* (i.e., *antecedent contained ellipsis*) (31c). Dans ces différents exemples, les éléments élidés sont soulignés ainsi que les éventuelles composantes (e.g., auxiliaires : does too, did) qui les remplacent.

(31) a. John likes golf and Mary does too.

b. John likes maths and Mary _ logic.

c. John read every book that Mary did.

Dans les deux premières variantes, l'ellipse a lieu en présence de deux clauses coordonnées de structures parallèles où la première (i.e., la source) est complète tandis que la seconde (i.e., la cible) ne l'est pas. Ainsi, dans l'énoncé (31a) par exemple, le syntagme verbal '*likes golf*' est supprimé de la clause cible, il est remplacé par l'auxiliaire '*does too*' qui partage la sémantique de son antécédent (i.e., la propriété d'aimer le golf). En revanche, dans une construction de gapping tel que (31b), l'élément élidé ne laisse aucune trace prononcée dans la clause cible, chose qui engendre la présence d'un trou (i.e., une trace phonétiquement vide) entre le sujet '*Mary*' et l'objet '*logic*'. Dans la dernière variante elliptique illutrée par l'exemple (31c), le constituant omis (un verbe transitif) est compris dans la subordonnée relative qui est enchassée dans le syntagme verbal où se trouve l'antécédent '*likes*'.

Plusieurs auteurs ont tenté de rendre compte de ce phénomène relativement complexe en utilisant différentes techniques. Ainsi, Dalrymple et ses co-auteurs proposent dans [36] une méthode de recouvrement des éléments élidés en se basant sur la sémantique formelle. Ils utilisent comme support un algorithme d'unification d'ordre supérieur afin de résoudre certaines équations logiques et trouver ainsi les interprétations pertinentes des éléments non réalisés phonétiquement. A titre d'exemple, ces derniers considèrent que la sémantique de l'énoncé (31a) prend la forme **Like**(**John**, **Golf**)$\wedge$*P*(**Mary**) où la propriété *P* est solution de l'équation *P*(**John**)= **Like**(**John**, **Golf**). Dalrymple et al. ne retiennent que la solution pertinente *P*=λx. **Like**(x, **Golf**) qui conserve le parallélisme entre le sujet de la clause source et celui de la clause cible (i.e., cette propriété est vérifiée aussi bien par '*Mary*' que par '*John*')[11].

Alors que la démarche de Dalrymple et al. est focalisée sur l'aspect purement sémantique de l'ellipse, d'autres auteurs tels que Carpenter [26], Morrill [85, 83], Hendriks [53] et Jäger [61] se sont intéressés au traitement de ce phénomène aussi bien d'un point de vue syntaxique que sémantique et ce dans le cadre des grammaires de type logique. A titre d'exemple, afin de rendre compte de la VP-ellipse, Morrill attribue à l'auxiliaire '*does+too*' une sémantique lexicale non-linéaire permettant d'encoder le partage de la propriété sous-jacente au VP élidé [85]. Toutefois, cette solution vient au détriment de la simplicité et la lisibilité du type syntaxique assigné à cet

[11] Ladite équation admet une seconde solution λx. **Like**(**John**, **Golf**) qui est rejetée par les auteurs puisqu'elle néglige totalement la contribution sémantique de l'élément *Mary* qui est le parallèle de *John* dans la clause cible.

auxiliaire[12]. Globalement, pour analyser l'énoncé (31a), l'approche de Morrill force l'entrée '*does too*' à se combiner d'abord avec le constituant discontinu (and+Mary, ϵ) pour former un modifieur de syntagme verbal (de type VP\VP) portant la sémantique λP. λx. P(**Mary**)$\wedge$P(x). C'est en fusionnant ensuite avec le VP de la clause source que le partage de ressources est effectué. L'étude de Morrill comprend aussi le traitement du phénomène de gapping [83]. Ce traitement a été ultérieurement repris et amélioré par Hendriks qui utilise un système bi-directionnel multimodal pour simuler de manière plus élégante les opérateurs discontinus généralisés [53]. L'approche de Hendriks permet ainsi d'analyser un énoncé tel que (31b) en trois étapes principales. Dans un premier temps, les deux constituants discontinus (Mary , logic)^n et (John , maths)^n sont coordonnées ensemble par l'intermédiaire de la conjonction '*and*'. Ensuite, la composante résultante -portant le type $TV\backslash_w s$[13]- est combinée avec le verbe transitif '*likes*' garantissant ainsi le partage de sa sémantique entre les deux clauses mises en jeu. Finalement, ledit verbe subit un ensemble de déplacements dus à l'interaction entre les différents modes de composition du système, chose qui lui permet d'atteindre sa position finale en occupant le point d'insertion du constituant discontinu de la clause source.

D'un autre côté, Jäger propose la résolution de certaines constructions elliptiques telles que la VP-ellipse de façon analogue à son traitement du liage des pronoms personnels. Il utilise pour cela le connecteur non-linéaire | qui se combine avec son argument sans le consommer (cf. section 5.2.1). Ainsi, Jäger assigne à la variable prononcée '*does too*' le type syntaxique (np\s)|(np\s) exprimant qu'il s'agit bien d'un syntagme verbal qui a besoin d'être précédé par un VP (son antécédent) avec qui il partagera l'interprétation sémantique. L'énoncé (31a) est donc analysé comme suit dans cette approche :

$$\dfrac{\dfrac{\dfrac{\textit{John}}{\textbf{John} : np}\ Lex \qquad \dfrac{\dfrac{\textit{likes}}{\lambda y.\ \lambda x.\ \textbf{Like}(x,y) : (np\backslash s)/np}\ Lex \qquad \dfrac{\textit{golf}}{\textbf{Golf} : np}\ Lex}{[\lambda x.\ \textbf{Like}(x,\ \textbf{Golf}) : np\backslash s]_i}\ /E}{\textbf{Like}(\textbf{John},\ \textbf{Golf}) : s}\ \backslash E \qquad \dfrac{\dfrac{\dfrac{\textit{does too}}{[\lambda x.\ x : (np\backslash s)|(np\backslash s)]_i}\ Lex}{\lambda x.\ \textbf{Like}(x,\ \textbf{Golf}) : np\backslash s}\ |E \qquad \vdots \qquad \vdots}{\lambda Q.\ \textbf{Like}(\textbf{Mary},\ \textbf{Golf}) \wedge Q : s\backslash s}\ /E}{\textbf{Like}(\textbf{Mary},\ \textbf{Golf}) \wedge \textbf{Like}(\textbf{John},\ \textbf{Golf}) : s}\ \backslash E$$

5.3.2 Traitement de l'ellipse dans $\mathcal{GLE}$

Analyse de l'ACE

Nous proposons dans ce qui suit une méthode uniforme traitant les trois types d'ellipse précédemment évoqués. Notre approche est basée sur une généralisation de l'utilisation des constituants doubles de Kayne, introduits initialement pour rendre compte des pronoms [64]. Ainsi, un énoncé elliptique tel que (31c) peut être intuitivement analysé en supposant que le verbe transitif '*read*' forme avec l'auxiliaire '*did*' un constituant double [read, did] qui sera ultérieurement séparé suite au mouvement de tête de l'antécédent :

[12]Morrill assigne à l'entrée '*does+too*' le type composé ((VP\VP)↑VP)↓(VP\VP) où VP=np\s et la sémantique non-linéaire λP. λQ. P(Q, Q)

[13]Ce type simule le type discontinu s↑TV, il représente les phrases auxquelles il manque, quelque part, un verbe transitif.

$$\textit{John \_ every book that Mary } [\textit{read},\ \textit{did}] \longrightarrow \textit{John read}_i \textit{ every book that Mary } [t_i,\ \textit{did}]$$

Le doublet [read, did] mis en jeu peut être simplement formalisé dans $\mathcal{GLE}$ en utilisant l'entrée liée suivante :

$$e_1 \quad \vdash \begin{pmatrix} \lambda x_\Phi.\ \lambda y_\Phi.\ y_\Phi \bullet \textit{read} \bullet x_\Phi \\ \lambda x_\lambda.\ \lambda y_\lambda.\ \mathbf{Read}(y_\lambda,\ x_\lambda) \end{pmatrix} :! V_2 \multimap \begin{array}{l} ([(P_{\Phi 1}, P_{\lambda 1}) : V_2 \vdash (P_{\Phi 1}, P_{\lambda 1}) : V_2], \\ [(P_{\Phi 2}, P_{\lambda 2}) : V_2 \vdash (P'_{\Phi 2}, P_{\lambda 2}) : V_2]) \end{array}$$
$$\textit{où}\ :\ V_2 = d_{acc} \multimap d_{nom} \multimap c \textit{ et } P'_{\Phi 2} = \lambda x.\ \lambda y.\ y \bullet \textit{did} \bullet x.$$

L'entrée e_1 est liée à deux hypothèses contrôlées : la première sert à occuper le site de l'antécédent (i.e., le verbe transitif '*read*') tandis que la seconde coïncide avec sa trace phonétiquement prononcée '*did*'. Précisons que cette entrée est bien formée car elle est conforme à la spécification (2.c) de la définition (4.4.4). A la différence des entrées composées encodant la relation antécédent-pronom, l'entrée e_1 est non-linéaire simple, sa tête représentera ainsi un argument (une feuille droite) dans la dérivation arborescente (cf. Fig. 33). Le partage de la sémantique nécessaire à la résolution de l'ellipse se fera suite au déchargement simultané des deux hypothèses contrôlées et ce grâce à l'application de la règle hybride IE_s (cf. Fig. 29). A ce moment, la forme phonétique de la tête de l'entrée se substituera à la place de l'hypothèse non-prononcée $P_{\Phi 1}$ et les deux variables sémantiques $P_{\lambda 1}$ et $P_{\lambda 2}$ seront remplacées par la représentation sémantique du verbe '*read*'[14]. La figure 52 donne une représentation simplifiée de l'arbre de dérivation de l'énoncé (31c). A part l'entrée e_1, l'analyse utilise deux entrées liées supplémentaires e_2 et e_3 associées respectivement au pronom relatif '*that*' et au déterminant '*every*'. Les différentes composantes de ces entrées sont synthétisées dans le tableau ci-après :

Id	Φ-terme	λ-terme	Type abstrait	l_{hyps}
e_2	$\lambda P_\Phi.\ \lambda y_\Phi.\ y_\Phi \bullet \text{that} \bullet P_\Phi(\epsilon)$	$\lambda P_\lambda.\ \lambda Q_\lambda.\ \lambda y_\lambda.\ P_\lambda(y_\lambda) \wedge Q_\lambda(y_\lambda)$	$(d_{acc} \multimap c) \multimap n \multimap n$	$([\text{X} : d_{acc} \vdash \text{X} : d_{acc}])$
e_3	$\lambda x_\Phi.\ \lambda P_\Phi.\ P_\Phi(\text{every} \bullet x_\Phi)$	$\lambda P_\lambda.\ \lambda Q_\lambda.\ \forall \text{x}.\ P_\lambda(\text{x}) \Rightarrow Q_\lambda(\text{x})$	$n \multimap (d_{acc} \multimap c) \multimap c$	$([\text{Y} : d_{acc} \vdash \text{Y} : d_{acc}])$

La dérivation de la figure 52 converge puisque l'ordre d'introduction des hypothèses contrôlées est bien respecté. En effet, la variable prononcée '*did*' est introduite en premier (dans le bloc $\mathcal{B}_1$), alors que l'hypothèse associée à l'antécédent intégre la déduction en dernier. La permutation de ces deux hypothèses bloquera l'application de la règle IE_s, ce qui permet de rendre compte de l'agrammaticalité de l'énoncé : **John did every book that Mary read*.

D'une façon générale, le traitement de la variante *ACE* de l'ellipse se base sur l'emploi de doublets [V_{trans}, Aux] qui peuvent être automatiquement engendrés par une règle de production lexicale paraphrasée ci-après :

Pour toute entrée libre $e_i = (a_\Phi, a_\lambda) : V_2 \in \mathcal{LEX}$ (*associée à un verbe transitif* V_{trans}), *ajouter l'entrée* e'_i *ci-après à* $\mathcal{LEX}$:

$$e'_i \quad \vdash \begin{pmatrix} a_\Phi \\ a_\lambda \end{pmatrix} :! V_2 \multimap \begin{array}{l} ([(P_{\Phi 1}, P_{\lambda 1}) : V_2 \vdash (P_{\Phi 1}, P_{\lambda 1}) : V_2], \\ [(P_{\Phi 2}, P_{\lambda 2}) : V_2 \vdash (\lambda x.\ \lambda y.\ y \bullet \textit{Aux} \bullet x, P_{\lambda 2}) : V_2]) \end{array}$$

où Aux est une forme adéquate de l'auxilaire 'do' qui est compatible avec V_{trans} (*i.e., même temps, mode, personne...*).

[14] Rappelons que l'abstraction simultanée des deux hypothèses contrôlées implique une étape de contraction qui identifie les deux variables $P_{\lambda 1}$ et $P_{\lambda 2}$.

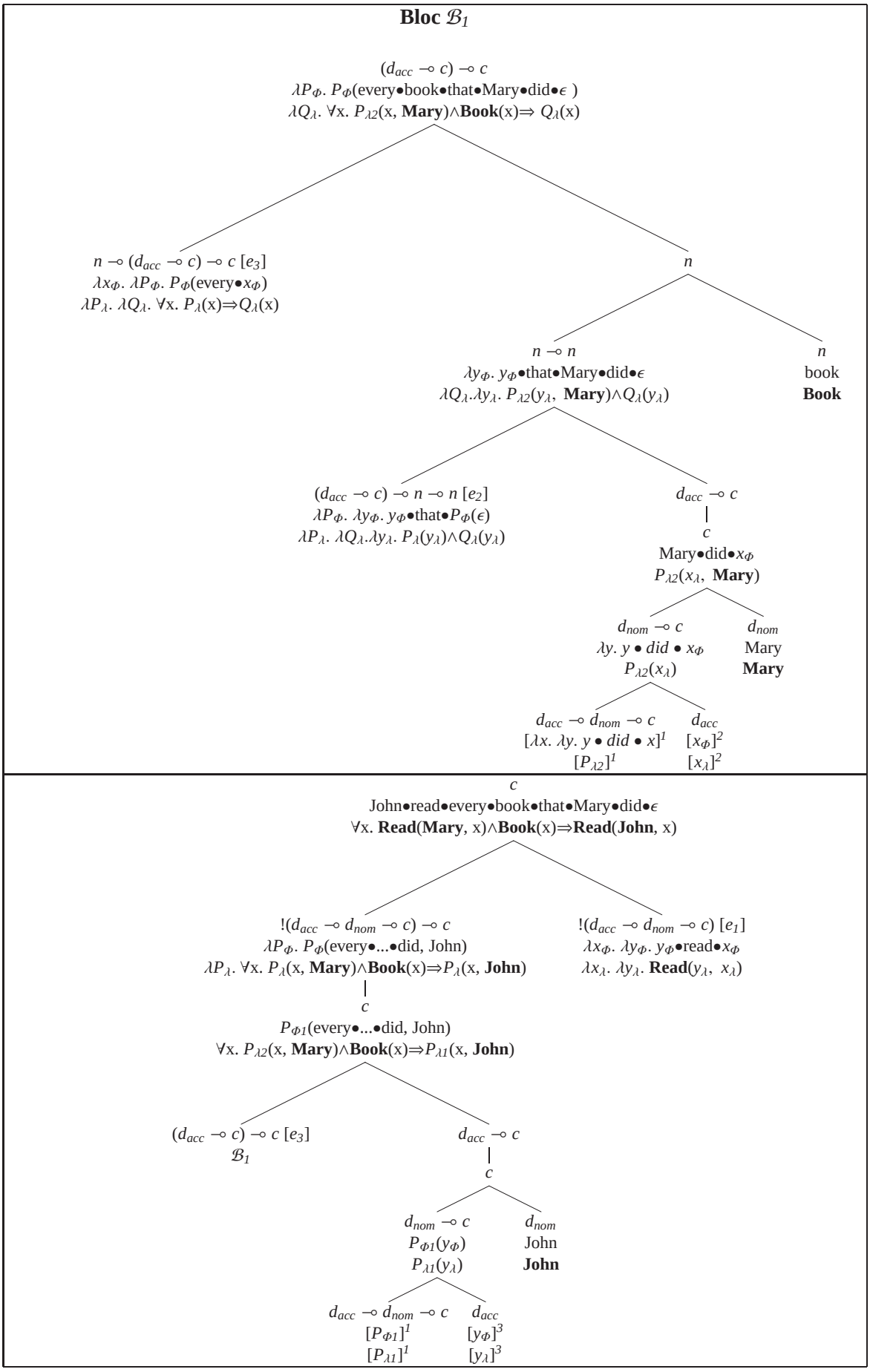


FIG. 52: Exemple d'analyse de la construction elliptique *ACE*

Analyse du gapping

Notre traitement de l'*ACE* peut être adapté sans difficulté pour prendre en charge la seconde variante d'ellipse à savoir le gapping. La construction du gapping, illustrée par l'énoncé (31b) fait intervenir le partage de la sémantique d'un verbe transitif entre la clause source où le verbe est réalisé phonétiquement et la clause cible qui ne comprend qu'une trace vide de ce dernier. Par analogie avec l'*ACE*, l'analyse de l'exemple (31b) est basée sur l'emploi d'un doublet [likes, ϵ] encodé formellement moyennant l'entrée liée suivante :

$$e_4 \quad \vdash \left(\begin{array}{c} \lambda x_\Phi.\ \lambda y_\Phi.\ y_\Phi \bullet \mathit{likes} \bullet x_\Phi \\ \lambda x_\lambda.\ \lambda y_\lambda.\ \mathbf{Like}(y_\lambda,\ x_\lambda) \end{array} \right) :!V_2 \multimap \begin{array}{l} ([(P_{\Phi 1}, P_{\lambda 1}) : V_2 \vdash (P_{\Phi 1}, P_{\lambda 1}) : V_2], \\ [(P_{\Phi 2}, P_{\lambda 2}) : V_2 \vdash (P'_{\Phi 2}, P_{\lambda 2}) : V_2]) \end{array}$$
$$\textit{où} \ :\ V_2 = d_{acc} \multimap d_{nom} \multimap c \ \textit{et}\ P'_{\Phi 2} = \lambda x.\ \lambda y.\ y \bullet x.$$

La différence entre les entrées e_1 et e_4 réside dans l'encodage de la composante phonétique de la seconde hypothèse contrôlée. Dans l'entrée e_1, le verbe élidé laisse derrière lui une trace phonétiquement prononcée, tandis que dans l'entrée e_4 la trace est vide. En effet, le Φ-terme associé à cette trace, à savoir $\lambda x.\ \lambda y.\ y \bullet x$, ne comprend aucune constante phonétique ; il n'est autre que la forme normale du terme $\lambda x.\ \lambda y.\ y \bullet \epsilon \bullet x$ où le site du verbe transitif est occupé par l'élément vide ϵ. En utilisant cette entrée liée, on est capable d'analyser l'énoncé (31b) de manière analogue à (31c) en suivant un ensemble d'étapes d'inférence synthétisées dans la dérivation simplifiée de la figure 53.

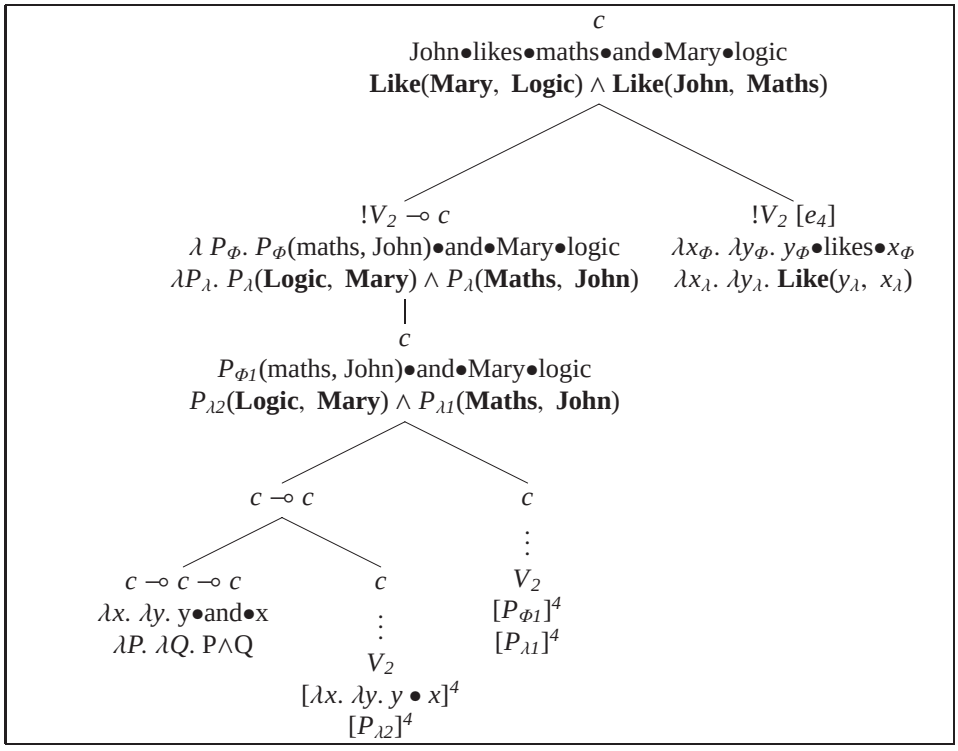


FIG. 53: Etapes principales de la dérivation de l'énoncé (31b)

Comme c'est le cas pour l'*ACE*, nous supposons que les entrées encodant les doublets [V_{trans}, ϵ]

sont automatiquement générés par une règle de production lexicale. Toutefois, il n'est pas difficile de constater que l'utilisation inappropriée de ces doublets fait sombrer le système dans une sur-génération indésirable. En effet, l'emploi du doublet [read, ϵ] à la place de [read, did] permettra d'analyser à tort la phrase agrammaticale **John read every book that Mary _*. Pour résoudre ce problème, il est nécessaire de discerner les contextes qui tolérent l'insertion de la trace vide associée au verbe élidé des contextes qui l'interdisent comme c'est le cas des ilôts (e.g., les phrases relatives). La solution proposée sera discutée et illutrée dans la section suivante.

Traitement de la VP-ellipse

Nous employons la même démarche pour traiter la VP-ellipse, conservant ainsi l'homogénéité du système. Cette fois ci, la ressource sémantique partagée n'est plus une relation binaire (de type concret $e \multimap e \multimap t$) mais un prédicat (de type $e \multimap t$) associé au syntagme verbal de la clause source. Il est ainsi nécessaire de forcer le verbe de la clause source à se combiner avec ses éventuels compléments d'objet avant de procéder à la résolution de l'ellipse. A titre d'exemple, pour analyser l'énoncé (31a), nous introduisons l'entrée e_5 qui formalise les points précédents :

$$e_5 \quad \vdash \left(\begin{array}{c} \lambda x_\Phi.\ \lambda y_\Phi.\ y_\Phi \bullet likes \bullet x_\Phi \\ \lambda x_\lambda.\ \lambda y_\lambda.\ \mathbf{Like}(y_\lambda,\ x_\lambda) \end{array} \right) : d_{acc} \multimap !V_1 \multimap \exists \quad \begin{array}{l} ([(P_{\Phi 1}, P_{\lambda 1}) : V_1 \vdash (P_{\Phi 1}, P_{\lambda 1}) : V_1], \\ [(P_{\Phi 2}, P_{\lambda 2}) : V_1 \vdash (P'_{\Phi 2}, P_{\lambda 2}) : V_1]) \end{array}$$
$$où\ :\ V_1 = d_{nom} \multimap c \text{ et } P'_{\Phi 2} = \lambda x.\ x \bullet does \bullet too.$$

Remarquons que la position de l'exponentielle, figurant dans le type principal de l'entrée, détermine la nature de l'élément partagé. En effet, dans les entrées e_1 et e_4, l'exponentielle est le connecteur le plus externe ; il englobe le type V_2 en entier. En revanche, dans l'entrée e_5, seul le sous-type $d_{nom} \multimap c$ est enveloppé par !, alors que le connecteur le plus externe coïncide avec $\multimap$. Ceci exprime que la fusion du verbe '*likes*' avec son objet direct (de type d_{acc}) est prioritaire puisqu'elle permet de construire le syntagme verbal qui sera ultérieurement partagé. L'analyse de l'énoncé (31a) illustrant le phénomène de la VP-ellipse découle directement de l'utilisation de l'entrée e_5 comme le montre la figure 54.
Comme c'est le cas pour le phénomène du gapping, les deux clauses coordonnées sont structurellement parallèles. Ainsi, les deux hypothèses contrôlées impliquées partagent la même profondeur dans leurs clauses respectives. Par conséquent, la comparaison entre leurs profondeurs globales dans la dérivation dépend uniquement de l'ordre de sous-catégorisation de la conjonction de coordination '*and*'[15]. En associant à '*and*' le Φ-terme $\lambda x.\ \lambda y.\ y \bullet and \bullet x$, nous exigeons sa combinaison avec la clause cible en premier. De ce fait, l'hypothèse contrôlée impliquée dans la clause cible intégrera l'arbre d'analyse avant celle introduite dans la clause source. Ainsi, la convergence de la dérivation n'est assurée que si la première hypothèse contrôlée est utilisée par la clause source et la seconde par la clause cible. Une telle contrainte est vérifiée par l'énoncé (31a) permettant à son analyse de converger, tandis qu'elle est violée par l'exemple agrammatical **John does too and Mary likes golf* dont la dérivation ne peut qu'être bloquée.

Même si dans la présente étude nous nous sommes focalisés sur les syntagmes verbaux dont la tête est un verbe transitif, il est intéressant de souligner que notre approche est générique et peut être aisément adaptée aux cas où ledit verbe est non-transitif (ex. 32a) ou di-transitif (ex. 32b).

[15] Rappelons que cet ordre est fixé par le lexique étant donné l'impossibilité du libre accès au raisonnement hypothétique.

(32) a. John left and Bill did too.

b. John threw flowers at Mary and Bill did too.

D'une façon générale, nous considérons que les entrées liées intervenant dans l'analyse de la VP-ellipse sont engendrées par une règle de production stipulant ce qui suit :

Pour tout verbe fini V d'arité $k+1$, *possédant une entrée libre* $e_i = (a_\Phi, a_\lambda) : d_{x_1} \multimap ... \multimap d_{x_k} \multimap d_{nom} \multimap c \in \mathcal{LEX}$, *ajouter l'entrée* e'_i *ci-après à* $\mathcal{LEX}$:

$$e'_i \quad \vdash \begin{pmatrix} a_\Phi \\ a_\lambda \end{pmatrix} : d_{x_1} \multimap ... \multimap d_{x_k} \multimap !V_1 \quad \exists \begin{array}{l} ([(P_{\Phi 1}, P_{\lambda 1}) : V_1 \vdash (P_{\Phi 1}, P_{\lambda 1}) : V_1], \\ [(P_{\Phi 2}, P_{\lambda 2}) : V_1 \vdash (\lambda x.\ x \bullet Aux \bullet too, P_{\lambda 2}) : V_1]) \end{array}$$

où $V_1 = d_{nom} \multimap c$, $\forall i \in \{1...k\}\ x_i \in \{obl, acc\}$, *et Aux est une forme adéquate de l'auxiliaire 'do' qui est compatible avec V* (*i.e., même temps, mode, personne...*).

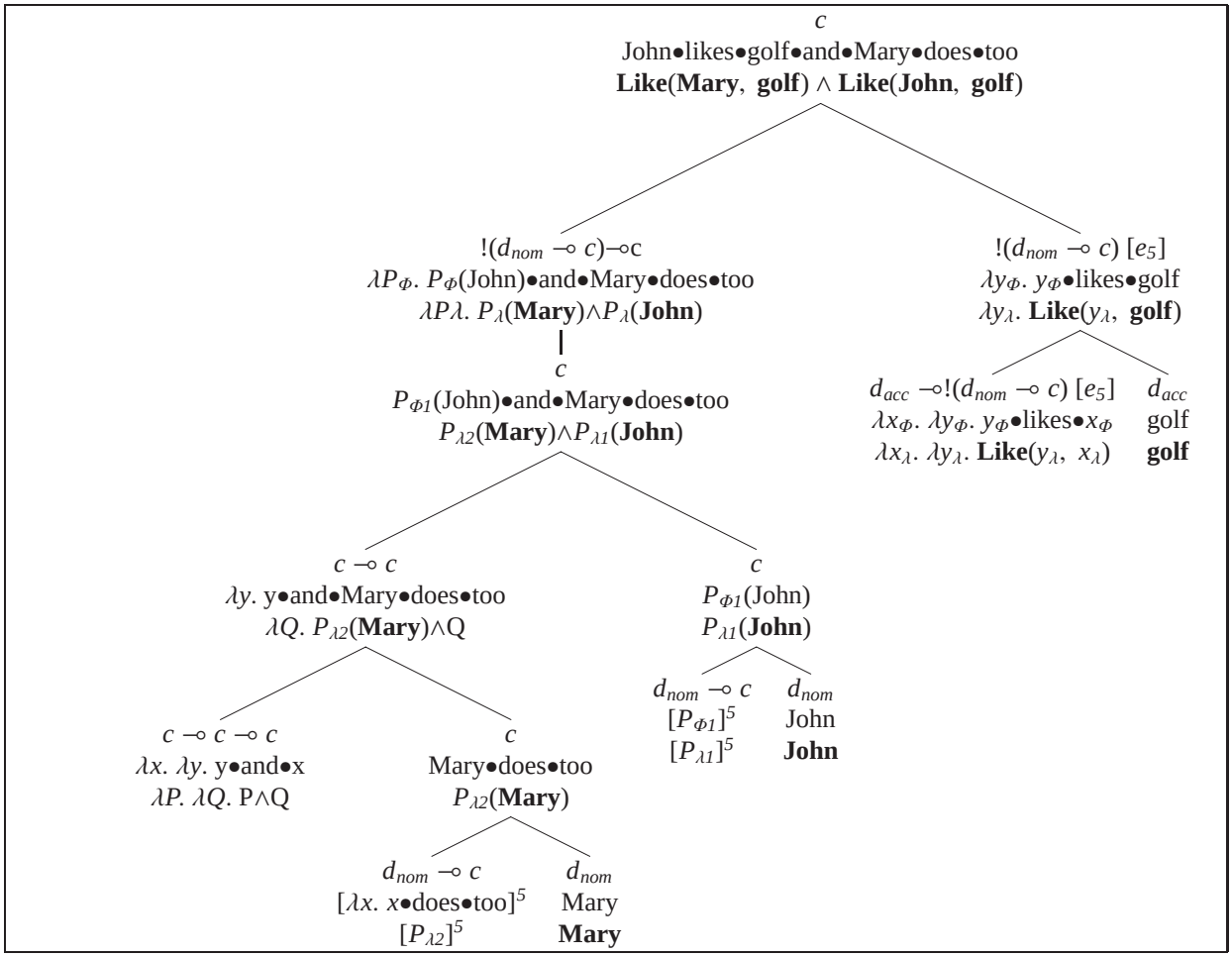


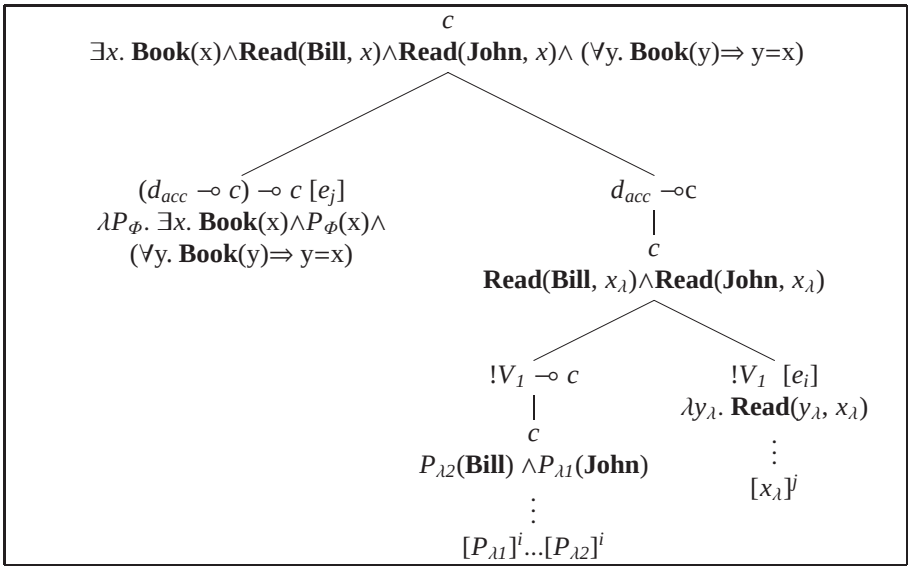
FIG. 54: Analyse de la VP-ellipse dans $\mathcal{GLE}$

Interaction de l'ellipse avec d'autres phénomènes linguistiques

Après avoir étudié le comportement basique de certaines variantes d'ellipse, il est intéressant de se tourner vers l'analyse de leur interaction avec d'autres phénomènes linguistiques.
Nous nous intéresserons notamment à l'étude de l'interaction entre l'ellipse et la quantification qui a lieu lorsque le VP élidé est constitué d'un déterminant (e.g., *a*, *the*, *every* ...) ou d'un quantificateur généralisé (e.g., *everyone*, *someone* ...) comme l'illustrent les exemples suivants :

(33) a. John read the book and Bill did too.
b. John read a book and Bill did too.
c. John gave every student a test and Bill did too.

Jusqu'à présent, nous avons formalisé le comportement des déterminants et quantificateurs généralisés par des entrées liées nécessitant l'introduction d'une hypthèse contrôlée de type d_{cas} qui sera ultérieurement déchargée au niveau d'une clause complète (de type *vp*, *ip* ou *c*)[16]. Dans de telles conditions, le déchargement des hypothèses contrôlées liées aux quantificateurs (ou quantificateurs généralisés) composant le VP-élidé se fera nécessairement en dernier après la résolution de l'ellipse. Ce fait est illustré par le schéma ci-après qui synthétise les étapes d'analyse de l'énoncé (33a).



L'approche adoptée favorise ainsi l'identité de référence entre l'objet de la clause source et celui de la clause cible. Ce partage est légitime quand il s'agit d'un syntagme nominal défini (e.g., '*the book*' dans l'énoncé 33a) mais n'est point désirable quand le VP-élidé comprend un syntagme nominal indéfini (e.g., '*a book*' dans l'énoncé 33b). En effet, si la résolution de l'ellipse se fait avant le déchargement de l'hypothèse contrôlée liée au déterminant '*a*', nous obtiendrons une lecture erronée de l'énoncé (33b) qui stipule l'existence d'un même livre lu aussi bien par '*John*' que par '*Bill*'.
Afin de rendre compte de l'identité du sens entre le VP de la clause source et celui de la clause cible, on a besoin de décharger l'hypothèse contrôlée liée au quantificateur avant d'abstraire les

[16]Le type primitif *vp* représente les clauses qui n'ont pas encore fusionné avec l'inflection.

hypothèses associées au VP elliptique. Pour cela, on introduit une nouvelle entrée lexicale e_{V_1} permettant aux quantificateurs de décharger, plus tôt, leur hypothèse contrôlée et ce au niveau du syntagme verbal.

$$e_{V_1} \vdash \left(\begin{array}{c} \lambda x_\Phi.\ \lambda P_\Phi.\ \lambda y_\Phi.\ P_\Phi(det \bullet x_\Phi, y_\Phi) \\ a_\lambda \end{array} \right) : n \multimap (d_{cas} \multimap !V_1) \multimap !V_1 \multimap (\![X : d_{cas} \vdash X : d_{cas}])$$
$$où\ :\ X = (x_\Phi,\ x_\lambda)\ et\ cas \in \{acc,\ obl\}.$$

Grâce à cette entrée, le syntagme nominal objet '*a book*' regagnera sa place au niveau du VP qui sera partagé (de type $!V_1$), par conséquent, le sens du quantificateur indéfini '*a*' sera capturé avant la résolution de l'ellipse et la sémantique du VP elliptique coïncidera alors avec le prédicat $\lambda x.\ \exists y.$ **Book**(y)∧**Read**(x,y).

Dans ce qui suit, nous proposons d'appliquer la solution précédente sur l'énoncé (33c) dans le but de construire la lecture (∃∀ ∧ ∃∀) où la portée de l'objet indirect '*a test*' est la plus large tout en restant locale à chacune des deux clauses coordonnées. La dérivation de l'énoncé (33c) qui engendre la lecture précédente est représentée dans la figure 55. Les entrées lexicales liées qui y sont utilisées sont récapitulées dans le tableau ci-dessous :

e_6	e_7	e_8
$\lambda x_\Phi.\ \lambda P_\Phi.\ \lambda y_\Phi.\ P_\Phi(\text{a}\bullet x_\Phi, y_\Phi)$	$\lambda x_\Phi.\ \lambda P_\Phi.\ \lambda y_\Phi.\ P_\Phi(\text{every}\bullet x_\Phi, y_\Phi)$	$\lambda x_\Phi.\ \lambda y_\Phi.\ \lambda z_\Phi.\ z_\Phi \bullet \text{gave} \bullet y_\Phi \bullet x_\Phi$
$\lambda P_\lambda.\ \lambda Q_\lambda.\ \lambda y_\lambda.\ \exists x.\ P_\lambda(x) \wedge Q_\lambda(x,\ y_\lambda)$	$\lambda P_\lambda.\ \lambda Q_\lambda.\ \lambda y_\lambda.\ \forall x.\ P_\lambda(x) \Rightarrow Q_\lambda(x,\ y_\lambda)$	$\lambda x_\lambda.\ \lambda y_\lambda.\ \lambda z_\lambda.$ **Give**$(z_\lambda, y_\lambda, x_\lambda)$
$n \multimap (d_{obl} \multimap !V_1) \multimap !V_1$	$n \multimap (d_{acc} \multimap !V_1) \multimap !V_1$	$d_{obl} \multimap d_{acc} \multimap !V_1$
$([(\text{X} : d_{obl} \vdash \text{X} : d_{obl}])$	$([(\text{X} : d_{acc} \vdash \text{X} : d_{acc}])$	$([(\text{P}_{\Phi 1}, \text{P}_{\lambda 1}) : V_1 \vdash (\text{P}_{\Phi 1}, \text{P}_{\lambda 1}) : V_1],$ $[(\text{P}_{\Phi 2}, \text{P}_{\lambda 2}) : V_1 \vdash (\text{did}, \text{P}_{\lambda 2}) : V_1])$

D'un autre côté, il est intéressant de souligner que notre approche permet également de rendre compte de la différence sémantique subtile entre les deux énoncés (34a) et (34b) issus de [36].

(34) a. John greeted every man when Bill did.

b. John greeted every man that Bill did.

En effet, alors que la phrase (34a) admet deux lectures différentes (31.a.1 & 31.a.2) suivant la portée du quantificateur '*every*' (locale ou globale), l'énoncé (34b), quant à lui, n'est pas ambigü et n'admet qu'une représentation sémantique unique (31.b.1).

31.a.1 **When**(∀x. **Man**(x)⇒**Greet**(**John**, x), ∀x. **Man**(x)⇒**Greet**(**Bill**, x)).
31.a.2 ∀x. **Man**(x)⇒**When**(**Greet**(**John**, x), **Greet**(**Bill**, x)).
31.b.1 ∀x. **Man**(x)∧**Greet**(**Bill**, x)⇒**Greet**(**John**,x).

Dans notre formalisme, la lecture (31.a.1) est obtenue si l'hypothèse contrôlée associée au quantificateur '*every*' est déchargée au niveau du VP (avant la résolution de l'ellipse), tandis que la lecture (31.a.2) est engendrée si l'abstraction de l'hypothèse en question a lieu au niveau de la clause entière (i.e., à la fin de la dérivation). En ce qui concerne l'exemple (34b), il représente un cas de l'**ACE** dont l'analyse est basée sur l'utilisation du doublet [greeted, did]. La sémantique à partager est réduite à celle du verbe transitif '*greeted*' et ne fait aucunement appel au quantificateur mis en jeu, chose qui explique la portée globale de ce dernier.

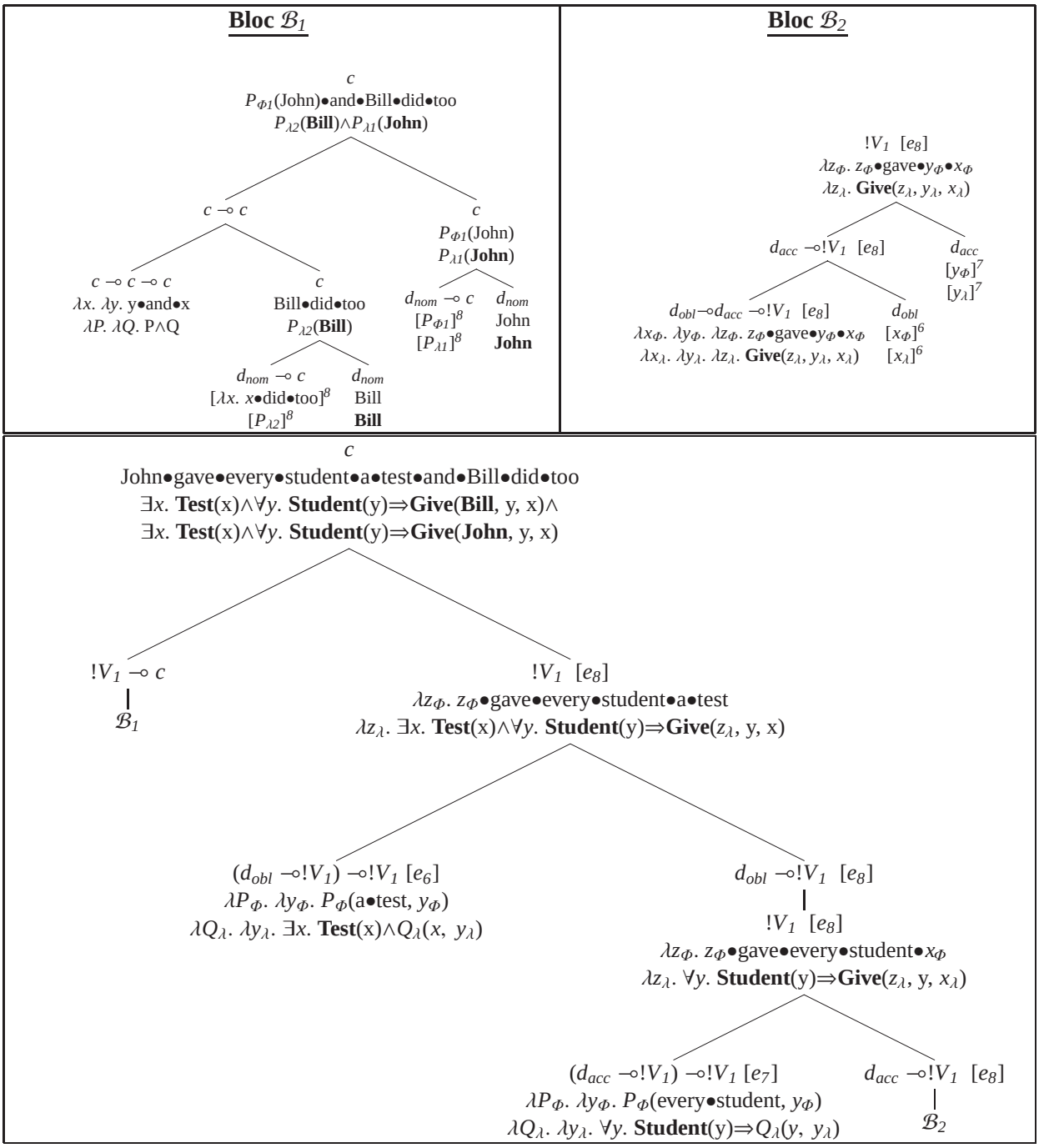


FIG. 55: Dérivation de la lecture ($\exists\forall \wedge \exists\forall$) de l'énoncé (33c)

5.4 Ilôts et extraction

5.4.1 Contributions antérieures

Les linguistes ont remarqué l'existence de certaines composantes syntaxiques qui ne permettent pas l'extraction de leurs sous-constituants (ou le permettent de manière fortement contrainte). D'après la terminologie de Ross, de tels domaines sont nommés des *ilôts* (i.e., *islands*) [98]. Les exemples ci-dessous illustrent l'interdiction d'extraction de certains ilôts :

(35) a. *the gift which John wonders [where Eric went to buy _].
b. *the man who John knows [whether Eric likes _].
c. *The book that [Mary sleeps and Eric reads _] is interesting.
d. *What sofa will he put the chair between [some table and _] ?
e. *The book that Mary hates the girl [who read _] is interesting.

Les exemples (35a & 35b) montrent l'impossibilité d'extraction des *qu-ilôts* (i.e., *wh-islands*) qui ne sont autres que les subordonnées enchassées introduites par l'intermédiaire d'un qu-mot (e.g., *where*, *whether*). D'un autre côté, les énoncés agrammaticaux (35c & 35d) illustrent l'interdiction d'extraire un élément de l'une des deux structures coordonnées. Cette contrainte, dite de structure de coordination (**CSC**, i.e., *coordinate structure constraint*), a été identifiée et caractérisée par Ross dans [98]. Enfin, le dernier exemple (35e) présente un cas de violation de la contrainte du NP complexe (**CNC**, i.e., *complex NP constraint*) qui stipule l'agrammaticalité d'extraction des clauses enchassées dans les NP (e.g., les phrases relatives) [98].
Dans l'approche transformationnelle, Chomsky a tenté d'expliquer le phénomène des ilôts en postulant l'existence de nœuds bloquants (e.g., NP et IP pour la langue anglaise) qui jouent le rôle de *barrières* et restreignent les dépendances non-bornées. L'opération de déplacement est ainsi contrainte par une condition de *sous-jacence* qui interdit à tout élément de traverser plus d'une catégorie bloquante lors de son mouvement. Une telle contrainte permet ainsi d'empêcher l'extraction des subordonnées relatives et expliquer, par exemple, l'agrammaticalité de l'énoncé (35e). En effet, le déplacement de l'objet du verbe '*read*' est prohibé étant donnée la présence de deux nœuds bloquants (i.e., IP_1 et IP_2) intervenant entre sa position initiale (t_j) et son site final ($that_j$) :

$$^{*}[CP_2\ that_j\ [IP_2\ Mary\ hates\ the\ girl\ [CP_1\ who_i\ [IP_1\ t_i\ read\ t_j]]]]$$

Notons que le mécanisme de barrières a été récemment reformulé par Chomsky dans sa théorie des phases selon laquelle les dérivations syntaxiques sont progressivement construites en plusieurs phases distinctes (i.e., structures locales) où chaque phase identifie un ilôt particulier [32]. La contrainte de sous-jacence est traduite par la *condition d'impénétrabilité de phases* qui empêche les éléments inclus dans le domaine d'une phase de participer dans les opérations des phases suivantes.

D'autre part, plusieurs auteurs, dont Morrill [82], Hepple [54] et Moortgat [77], ont suggéré de formaliser certains cas du phénomène des ilôts dans des systèmes logiques bi-directionnels en se basant sur l'utilisation de l'opérateur modal $\Box$. Ainsi, afin de prendre en charge la contrainte **CSC**, Moortgat propose d'assigner à la conjonction de coordination '*and*' un type de la forme $(X\backslash\Box X)/X$ (à la place de $(X\backslash X)/X$). Ce type raffiné permet à la conjonction de se combiner avec ses

deux éléments coordonnés formant ainsi une structure de coordination qui porte le type distingué $\Box X$. Afin de se débarasser du connecteur $\Box$, cette structure doit être enveloppée par l'opérateur structurel $< >$ qui permet de délimiter les frontières de l'ilôt en question. Par conséquent, toute ressource hypothétique encapsulée à l'intérieur de $< >$ se trouve bloquée et ne peut être déchargée. La figure 56 illustre la proposition de Moortgat et montre la raison sous-jacente à l'agrammaticalité de l'énoncé (35c) dans $L\Diamond$ (calcul de Lambek associatif enrichi par les opérateurs de contrôle).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{that \vdash (n\backslash n)/(s/np)}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{and \vdash (s\backslash\Box s)/s}\,Ax \quad \dfrac{\dfrac{\dfrac{}{Mary, sleeps \vdash s}\ ...}{Eric, reads, (x:np) \vdash s}\ ...}{and, Eric, reads, (x:np) \vdash s\backslash\Box s}\,\backslash E}{Mary, sleeps, and, Eric, reads, (x:np) \vdash \Box s}\,/E}{< Mary, sleeps, and, Eric, reads, (x:np) >\vdash s}\,\Box E}{< Mary, sleeps, and, Eric, reads >, (x:np) \vdash s}\,(*)}{< Mary, sleeps, and, Eric, reads >\vdash s/np}\,/I}{that, < Mary, sleeps, and, Eric, reads >\vdash n\backslash n}\,/E$$

FIG. 56: Agrammaticalité de l'énoncé (35c) dans L$\Diamond$

5.4.2 Formalisation des ilôts dans $\mathcal{GLE}$

Le système $\mathcal{GLE}$ autorise l'abstraction des hypothèses contrôlées indépendamment de leurs positions structurelles dans une phrase. D'un côté, une telle flexibilité est positive, car elle permet de prendre en charge l'extraction non périphérique (extraction médiane) de manière systématique, à la différence des systèmes bidirectionnels. Toutefois, l'absence totale de contraintes gérant l'opération d'extraction fait sombrer le système dans une sur-génération défavorable. En particulier, l'état actuel du formalisme tolére la dérivation de tous les exemples agrammaticaux (35) où les éléments extraits sont inclus dans des ilôts. L'objectif de cette section est d'enrichir notre système afin de tenir compte de l'interdiction d'extraction de certains contextes particuliers.

Afin de rendre compte de la construction des ilôts dans $\mathcal{GLE}$, on définit un nouvel opérateur de types abstraits $\hat{}$ tel que :

$$\forall A \in \mathcal{T}(\mathcal{A})\ \ \hat{A} \in \mathcal{T}(\mathcal{A}),\ \tau_\Phi(\hat{A}) = \tau_\Phi(A)\ et\ \tau_\lambda(\hat{A}) = \tau_\lambda(A).$$

Le comportement de cet opérateur est décrit par la règle de sous-typage ci-dessous : La règle$\hat{}S$

$$\dfrac{\Gamma \vdash (a_\Phi, a_\lambda) : \hat{A}}{\Gamma \vdash (a_\Phi, a_\lambda) : A}\ \hat{}S\ (\flat)$$

FIG. 57: Règle de sous-typage associée à l'opérateur $\hat{}$

représente une forme du principe de substitutivité qui permet de considérer les ilôts bien formés de type $\hat{A}$ comme un cas particulier de constituants de type A (i.e., $\hat{A}$ est un sous-type de A). Ce

passage n'a lieu que si la contrainte de bonne formation d'ilôts ♭ est respectée. Cette condition stipule que toutes les variables du contexte Γ proviennent d'hypothèses contrôlées prononcées. Ceci revient à dire que toutes les variables contrôlées non-prononcées introduites au cours de la dérivation de $\Gamma \vdash (a_\Phi,\ a_\lambda) : \hat{A}$ devraient être déchargées. Ainsi, le Φ-terme a_Φ doit être clos (i.e., ne doit contenir aucune Φ-variable libre).

L'introduction de ce nouvel opérateur nous permet de bloquer assez aisément la dérivation des exemples (35) précédents. Ainsi, pour exprimer la prohibition d'extraction des qu-ilôts illustrée dans (35a & 35b), il suffit d'attribuer aux qu-mots '*where*' et '*whether*' le type abstrait décoré $c \multimap \overset{\wedge}{cp}$ alors que les verbes '*wonders*' et '*knows*' gardent leur type usuel $cp \multimap d_{nom} \multimap c$. Un tel typage permet de considérer les subordonnées complétives préfixées par un qu-mot comme des clauses closes interdites à l'extraction. Afin d'intégrer la dérivation globale, il est nécessaire de convertir le type de ces clauses (i.e., $\overset{\wedge}{cp}$) en *cp* par l'intermédiaire de la règle de sous-typage $\wedge S$. L'agrammaticalité des énoncés (35a & 35b) est ainsi expliquée par l'impossibilité de vérifier la bonne formation des qu-ilôts mis en jeu. En effet, la contrainte ♭ est totalement violée dans ces deux cas puisque la forme phonétique de ces qu-ilôts contient une Φ-variable libre occupant la position objet. La figure 58 montre l'étape principale du blocage de la dérivation de l'énoncé 35b :

$$\dfrac{\dfrac{}{\vdash \lambda x.\ \lambda y.\ y \bullet knows \bullet x : cp \multimap d_{nom} \multimap c}\ Lex \qquad \dfrac{\dfrac{}{\vdash \lambda x.\ whether \bullet x : c \multimap \overset{\wedge}{cp}}\ Lex \qquad \dfrac{\dfrac{\dfrac{(whom)}{x : d_{acc} \vdash x : d_{acc}}\ Ax}{\vdots}}{x : d_{acc} \vdash Eric \bullet likes \bullet x : c}}{x : d_{acc} \vdash whether \bullet Eric \bullet likes \bullet x : \overset{\wedge}{cp}}\ \multimap E}{x : d_{acc} \vdash whether \bullet Eric \bullet likes \bullet x : cp}\ \wedge S\ (*)$$
$$\dfrac{x : d_{acc} \vdash \lambda y.\ y \bullet knows \bullet whether \bullet Eric \bullet likes \bullet x : d_{nom} \multimap c}{\vdots}\ \multimap E$$
$$\dfrac{}{\vdash \lambda z.\ z \bullet whom \bullet John \bullet knows \bullet whether \bullet Eric \bullet likes \bullet \epsilon : n \multimap n}\ \multimap IE_c$$

FIG. 58: Blocage de la dérivation de l'exemple (35b)

Comparons maintenant les deux exemples agrammaticaux (35a & 35b) avec les expressions bien formées suivantes :

(36) a. the man who John knows that Mary likes _.
b. John wonders whether he will win.

L'énoncé (36a) représente un syntagme nominal syntaxiquement correct dont la structure est identique à celle de l'exemple (35b). Néanmoins, à la différence de ce dernier, nous employons le complémenteur (i.e., *complementizer*) '*that*' qui se combine avec une clause ouverte à l'extraction. Ainsi, à l'opposé du qu-mot '*whether*', la subordonnée complétive préfixée par '*that*' ne constitue pas un ilôt. Le complémenteur '*that*' conserve alors son type abstrait non décoré : $c \multimap cp$. D'autre part, même si le qu-mot '*whether*' déclenche la formation d'un ilôt, l'énoncé (36b) reste grammatical. En effet, l'hypothèse contrôlée (i.e., '*he*') impliquée dans la construction de la clause

imbriquée '*he will win*' est prononcée. La condition de bonne formation d'ilôt ♭ est ainsi vérifiée, chose qui permet d'appliquer la règle de sous-typage $\wedge S$. La figure 59 récapitule les étapes principales de la dérivation de cet énoncé. L'analyse en question est basée sur l'utilisation d'une entrée lexicale liée formalisant le doublet [John, he] (cf. section 5.2.2).

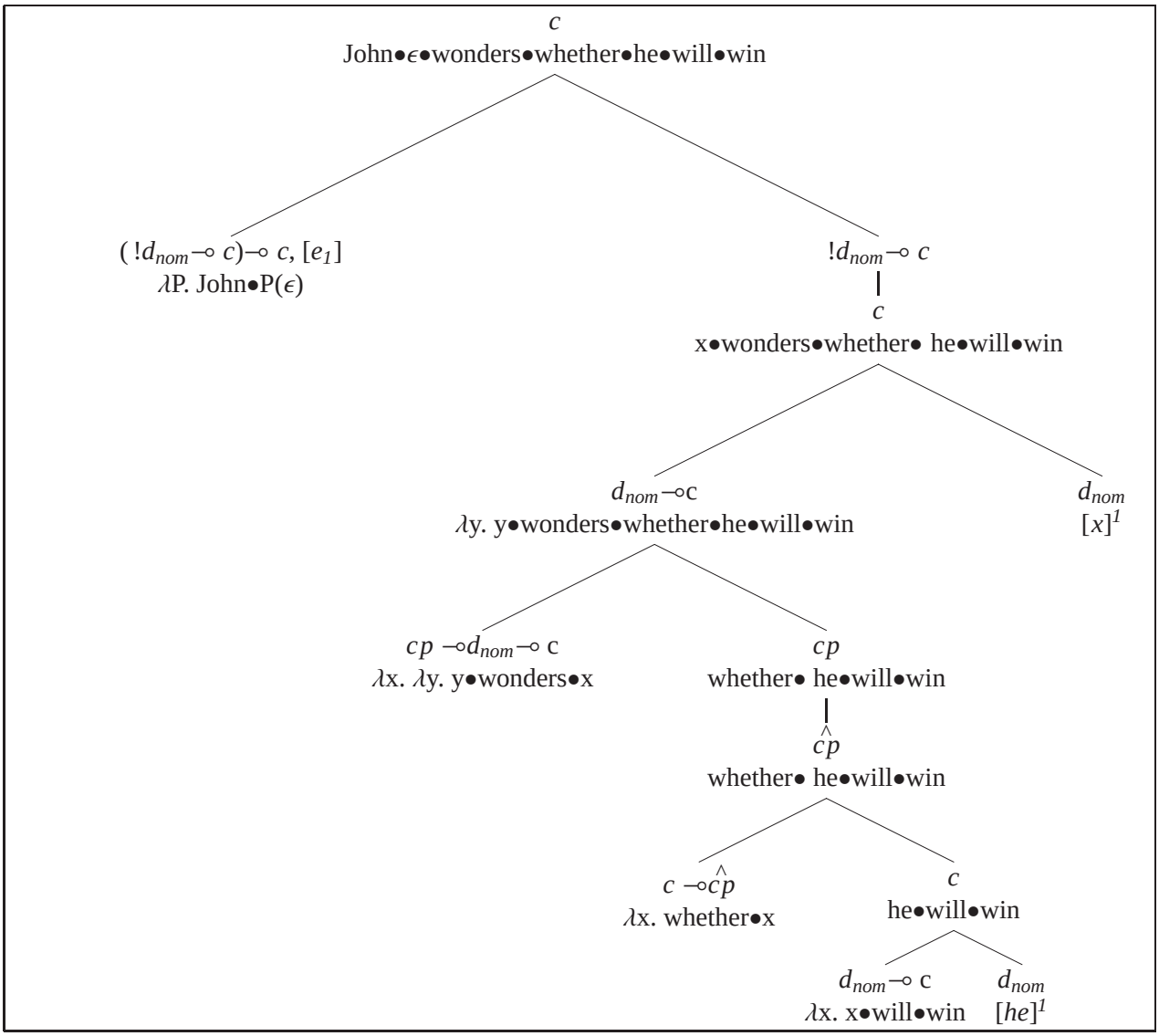


FIG. 59: Dérivation de l'énoncé (36b)

La même démarche peut être appliquée pour rendre compte des contraintes de Ross (e.g., **CSC** et **CNC**) et bloquer l'extraction des structures de coordination et des subordonnées relatives. A titre d'exemple, en associant aux pronoms relatifs (e.g., '*who*') le type raffiné $(d_{acc} \multimap c) \multimap (n \stackrel{\wedge}{\multimap} n)$, on force la formation d'un ilôt autour de la clause relative (émanant de la combinaison du pronom avec sa subordonnée). De ce fait, l'analyse de la phrase agrammaticale (35e) ne pourra pas aboutir étant donné que l'ilôt construit à partir de la phrase relative interne '*who•reads•x*' est mal formé car il contient une Φ-variable libre (i.e., l'hypothèse contrôlée *x* liée au pronom relatif '*who*').

Le traitement des pronoms relatifs proposé permet de rendre compte de l'interaction entre le phénomène d'ilôt et d'autres phénomènes linguistiques tels que l'ellipse. En effet, il n'est pas difficile de constater que la présente approche bloque systématiquement la dérivation de l'énoncé mal-formé (37a) qui s'appuie sur l'emploi du doublet [read, ϵ] alors qu'elle tolére l'analyse de la phrase correcte (37b) qui utilise, quant à elle, le constituant double [read, did]. La contrainte de bonne formation $\flat$ est uniquement respectée dans le second cas où l'élément elliptique est une variable prononcée (l'auxiliaire '*did*').

(37) a. *John read every book that Mary _.
b. John read every book that Mary did.

Finalement, il convient de préciser que notre approche permet de détecter, le plus tôt possible, la mal-formation des ilôts mis en jeu. Par exemple, l'agrammaticalité d'un énoncé composé de plusieurs subordonnées enchassées tel que (38) est repérée de manière anticipée dès la construction de la relative la plus imbriquée '*who reads _*'.

(38) *the book that Mary thinks John hopes ... Kim hates the girl [who read _].

En revanche, dans les contributions antérieures, notamment celle de Moortgat [77], la mal-formation de l'ilôt est décelée tardivement après la construction de la structure entière associée à la subordonnée relative principale (i.e., '*<Mary, thinks, John, hopes, ..., Kim, hates, the, girl, who, read, x>*').

5.5 Conclusion

Ce chapitre présente certaines applications linguistiques réalisées dans le formalisme $\mathcal{GLE}$. Nous avons particulièrement opté pour la description de quelques phénomènes complexes qui sont difficiles à traiter dans les grammaires catégorielles bidirectionnelles.
Dans un premier temps, nous avons montré que ce formalisme offre la possibilité de rendre compte du phénomène de discontinuité en permettant de regrouper les différents constituants discontinus dans la même entrée lexicale. Ensuite, nous avons présenté une formalisation logique des principes de la théorie de liage visant le traitement des anaphores et des pronoms personnels. D'une part, en contrôlant l'accès à la technique de raisonnement hypothétique, on arrive à respecter la contrainte de localité dictée par le principe *A*. D'autre part, en utilisant des entrées lexicales non-linéaires composées, qui déchargent simultanément leurs hypothèses contrôlées, on est capable de traiter les dépendances sémantiques entre les pronoms personnels et leurs antécédents tout en tenant compte du principe *B*. En outre, nous avons généralisé notre approche afin de prendre en charge le partage de ressources inhérent à différentes constructions elliptiques (VP-ellipse, **ACE**, gapping). Enfin, nous avons suggéré d'étendre le système de base par une règle de sous-typage simple et intuitive qui permet de contrôler le phénomène d'extraction et vérifier la bonne formation des ilôts.

Conclusions et Perspectives

L'exploration d'un domaine pluridisciplinaire tel que la linguistique computationnelle représente une aventure enrichissante qui est jalonnée de péripéties. Cette expérience est indubitablement instructive puisqu'elle permet de prendre conscience de la connexion entre des disciplines d'apparence éloignées telles que la linguistique et la logique. C'est dans cette orientation que s'inscrit la présente étude.
Cette thèse apporte deux contributions principales. D'un côté, nous avons proposé un nouveau formalisme non-directionnel $\mathcal{GLE}$ qui est capable de simuler les primitives du Programme Minimaliste dans un cadre logique. En outre, nous avons illustré sa capacité à décrire des phénomènes complexes tels que l'ellipse, le liage d'anaphores et les ilôts. D'un autre côté, nous avons réalisé un atelier logique $\mathcal{ICHARATE}$ dédié à la formalisation de certains modèles linguistiques avancés (grammaires catégorielles multimodales et $\mathcal{GLE}$) dans le système d'aide à la démonstration **Coq**.
Les différentes contributions issues de ces travaux ont été regroupées en deux parties. Nous allons donc conclure par un récapitulatif synthétique du contenu de chacune de ces parties et nous proposerons, au fur et à mesure, quelques perspectives de recherche qui nous paraissent intéressantes à poursuivre.

6.1 Grammaires catégorielles multimodales

Je me suis focalisée sur l'étude des grammaires catégorielles multimodales pendant les deux premières années de ma thèse. Ce travail m'a conduit à la réalisation d'un atelier logique et pédagogique $\mathcal{ICHARATE}$, comprenant un ensemble de bibliothèques **Coq** dédiées à la recherche et l'enseignement de certains formalismes de la linguistique computationnelle, dont les grammaires multimodales. L'objectif de cet outil est de faciliter la compréhension de ces modèles complexes en offrant aux utilisateurs la possibilité d'effectuer interactivement leurs analyses syntaxiques de phrases, d'obtenir leurs interprétations sémantiques et surtout d'établir des propriétés universelles valides pour des classes entières de grammaires. A la différence de l'analyseur automatique **Grail**, il est possible de tirer profit du pouvoir expressif de la logique d'ordre supérieur de **Coq** afin d'énoncer et prouver des règles dérivées génériques qui constituent des schémas de dérivations. $\mathcal{ICHARATE}$ contient déjà un catalogue de règles réutilisables modélisant le comportement de phénomènes linguistiques divers (e.g., extraction médiane, dépendances non-bornées, dépendances croisées) ainsi qu'un nombre important de méta-théorèmes décrivant certaines propriétés formelles du système (e.g., interopérabilité entre le calcul de séquents et la déduction naturelle, propriété de sous-formule dans le calcul de séquents). Ce catalogue ne représente aucunement une compilation close de théorèmes, il peut être étendu par les utilisateurs qui souhaitent explorer

les propriétés de nouvelles classes de grammaires multimodales. Cette extension est rendue facile grâce à l'utilisation d'outils spécialisés tels que les tactiques ou la technique de proof-by-pointing. En outre, quelques essais ont été menés pour enrichir la sémantique de Montague et inclure les types inductifs et les spécifications fortes de **Coq**. Il serait pertinent d'approfondir cette expérience afin de passer d'une sémantique statique se limitant au niveau de la phrase à une sémantique dynamique qui modélise le discours et le dialogue. En effet, la théorie des types aurait un bon potentiel pour développer un substitut rigoureux à la λ-DRT avec de bonnes propriétés de compositionnalité comme le prouvent les travaux de de Groote basés sur les continuations [38]. Une deuxième piste consiste à établir la communication entre **Grail** et $\mathcal{ICHARATE}$. Une telle connexion permettrait, à titre d'exemple, de déléguer la construction des dérivations syntaxiques des énoncés d'un discours à **Grail** et utiliser les résultats obtenus pour calculer les représentations sémantiques avancées de ces phrases et raisonner sur leur contenu. Enfin, du point de vue de la forme, il serait bénéfique d'améliorer l'interface utilisateur de cet atelier pour prendre en charge l'automatisation des tâches répétitives et faciliter la saisie des données (modes, mots, lexiques ...).

Outre la réalisation de l'outil $\mathcal{ICHARATE}$, je me suis penchée sur l'étude des phénomènes linguistiques inhérents à la formation des phrases nominales dans la langue arabe. Ceci m'a mené à une tentative de description de la syntaxe et la sémantique d'un certain nombre de ces phénomènes au sein de la logique multimodale [9]. L'utilisation des opérateurs unaires de contrôle s'avère adéquate pour vérifier les traits morphosyntaxiques des mots dans cette langue fortement flexionnelle (cf. traitement du phénomène d'annexation dans la section 2.3.2). Il serait intéressant d'élargir le champ de cette étude afin de prendre en charge d'autres constructions grammaticales plus élaborées (e.g., ordre des mots dans les phrases verbales). Il serait aussi instructif d'effectuer une comparaison entre la langue arabe et d'autres langues sémitiques telles que l'hébreu.

6.2 Grammaires logiques étiquetées

Dans cette seconde partie de la thèse, nous avons proposé un système non-directionnel souple et compact qui est basé sur un fragment réduit de la logique linéaire. Un tel système offre plusieurs atouts dont les suivants :

1. $\mathcal{GLE}$ autorise le recours au raisonnement hypothétique de manière contrainte et totalement guidée par le lexique. Cet avantage garantit la normalité de toutes les dérivations effectuées et permet, en outre, de prendre en charge les contraintes de localité sous-jacantes au liage des réflexifs.

2. $\mathcal{GLE}$ permet d'interpréter les transformations des Grammaires Minimalistes dans un environnement logique où le calcul de la représentation sémantique se fait, sans peine, grâce à la correspondance de Curry-Howard. Ainsi, la fusion est encodée moyennant la règle de modus-ponens ($\multimap$ E), tandis que le mouvement de constituant et le déplacement de tête sont simulés par des règles hybrides ($\multimap E_c$ et $\multimap E_s$) qui combinent une phase de raisonnement hypothétique avec une étape de fusion.

3. $\mathcal{GLE}$ est capable de rendre compte de phénomènes linguistiques avancés (e.g., discontinuité, théorie de liage, ellipse, ilôts) de manière intuitive et élégante. Le partage de ressources nécessaire pour analyser les phénomènes non-linéaires est pris en charge grâce à l'utilisation d'entrées lexicales liées qui déchargent simultanément leurs hypothèses contrôlées.

Ce travail laisse entrevoir de nombreuses pespectives prometteuses. La première d'entre elles consiste à confirmer, par une preuve rigoureuse, l'interopérabilité entre $\mathcal{GLE}$, les Grammaires Minimalistes et les Grammaires Catégorielles Multimodales. En effet, notre présentation de l'encodage logique des Grammaires Minimalistes ainsi que celle de Vermaat décrite dans [106] sont, pour l'instant, informelles.
D'un autre côté, il reste à caractériser certaines propriétés computationnelles fondamentales de $\mathcal{GLE}$. En se limitant à des entrées lexicales dont la tête comprend au moins une constante phonétique, on se garantit la décidabilité du système. Néanmoins, il reste à trouver une stratégie d'analyse efficace pour ces grammaires et établir sa complexité. Il serait également pertinent d'étudier le pouvoir expressif de $\mathcal{GLE}$. En effet, nous ne disposons, jusqu'à présent, d'aucun résultat qui spécifie les types de langages pouvant être modélisés dans $\mathcal{GLE}$. Il est toutefois clair que ces grammaires sont plus expressives que les grammaires hors contextes puisqu'elles sont capables d'engendrer des langages faisant intervenir des dépendances croisées tels que $(\{a^n b^m c^n d^m \mid \text{n, m} \geq 0\})$. Ce dernier langage est, à titre d'exemple, engendré par la grammaire $\mathcal{GLE}$ définie par le lexique suivant (où *ty* dénote le type composé $c \multimap c \multimap c \multimap c \multimap$c) :

$\vdash \epsilon : c$
$\vdash \lambda$x. λy. λz. λu. x•y•z•u : ty
$\vdash \lambda$ P. λx. λy. λz. λu. P(a•x, y, c•z, u) : ty $\multimap$ ty
$\vdash \lambda$ P. λx. λy. λz. λu. P(x, b•y, z, d•u) : ty $\multimap$ ty

Il serait aussi intéressant d'étudier la possibilité d'utiliser le tenseur de la logique linéaire $\otimes$ à la place de l'exponentielle. En effet, dans notre cadre, nous manipulons uniquement des listes finies d'hypothèses contrôlées.

D'un point de vue linguistique, notre encodage logique de la théorie de liage souffre d'une légère asymétrie. En effet, alors que les réflexifs sont représentés par des entrées libres autonomes, les pronoms personnels sont considérés comme des hypothèses contrôlées qui sont attachées à leurs antécédents. Cependant, cette asymétrie est en phase avec le phénomène de *retard de l'effet du Principe B* (i.e., *Delay of Principle B effect*) [27] selon lequel les enfants obéissent au Principe *A* bien avant le Principe *B*. Dans notre approche, ceci pourrait s'expliquer par la difficulté d'acquérir les entrées lexicales liées (les doublets ou triplets) décrivant le comportement syntaxique des pronoms.

La formalisation logique de la théorie des phases de Chomsky [32] constitue une piste prometteuse susceptible d'homogénéiser le traitement de plusieurs phénomènes linguistiques basés sur les contraintes de localité. A titre d'exemple, le Principe *A* pourrait s'exprimer, dans ce cadre, par la nécessité d'abstraire les réflexifs dans leur phase d'introduction, tandis que le Principe *B* pourrait être formulé par l'obligation de décharger les pronoms dans une phase ultérieure.
Finalement, il serait intéressant d'élargir les champs d'applications linguistiques de $\mathcal{GLE}$ en tentant de décrire d'autres phénomènes non-linéaires tels que la coordination et ses deux variantes sémantiques (distributive et non-distributive) [26] ainsi que les pronoms résomptifs [15] présents, entre autres, dans la langue arabe. Il reste également à améliorer le traitement des ilôts pour tenir compte des exceptions tolérant l'extraction de certains contextes clos notamment dans le cas des *trous parasites* (i.e., *parasitic gaps*) (e.g., '*Which book did John file* _ [*without reading* _]' ?) [103, 83].

Bibliographie

[1] K. Ajdukiewicz. Die syntaktische Konnexität. *Studies in Philosophy*, 1 :1–27, 1935.

[2] C. Alvarado. *Réflexion pour la réécriture dans le calcul des constructions inductives*. PhD thesis, Université de Paris Sud, Orsay, 2002.

[3] M. Amblard, H. Anoun, and A. Lecomte. Ellipse et coordination en grammaire logique. In *JSM : Journées Sémantique et Modélisation*, Bordeaux, 2006.

[4] A. Amerkad, Y. Bertot, L. Pottier, and L. Rideau. Mathematics and proof presentation in pcoq. Technical report, INRIA, 2001.

[5] H. Anoun. Formalisation du calcul de lambek en coq. Master's thesis, Université de Bordeaux 1 & LaBRI, 2003.

[6] H. Anoun. Icharate : Un atelier logique pour les grammaires multimodales. In *Recital : Rencontre des étudiants chercheurs en informatique pour le traitement automatique des langues*, Fes, 2004.

[7] H. Anoun. Méta-linguistique dans l'assistant de preuves coq. In *MajecStic : Manifestation des Jeunes Chercheurs en Stic*, Rennes, 2005.

[8] H. Anoun. Reasoning on multimodal grammars with the calculus of inductive constructions. In *LPAR-12 : International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Montego Bay, 2005.

[9] H. Anoun. Towards a logical analysis of nominal sentences in standard arabic. In *session étudiante d'Esslli*, Malaga, 2006.

[10] H. Anoun. Une bibliothèque coq pour le traitement des langues naturelles. *Technique et Science Informatiques*, 2007.

[11] H. Anoun and P. Castéran. A coq toolkit for lambek calculus, 2003. Contribution Coq téléchargeable à partir de http ://coq.inria.fr/contribs-eng.html.

[12] H. Anoun and P. Castéran. Icharate : A logical toolkit for multimodal categorial grammars, 2006. Contribution Coq téléchargeable à partir de http ://coq.inria.fr/contribs-eng.html.

[13] H. Anoun, P. Castéran, and R. Moot. Proof automation for type-logical grammars. Technical report, ESSLLI, 2004.

[14] H. Anoun and A. Lecomte. Logical grammar with labels. In *Formal Grammar*, Malaga, 2006.

[15] A. Asudeh. *Resumption as Resource Management*. PhD thesis, Stanford University, 2004.

[16] J. Baldridge. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2002.

[17] Y. Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29 :47–58, 1953.

[18] H. Barendregt and H. Geuvers. Proof assistants using dependent type systems. In *Handbook of Automated Reasoning*. Elsevier Science, 2001.

[19] B. Barras. A formalisation of the calculus of construction, 1997. Contribution Coq, coq.inria.fr/contribs/coq-in-coq.html.

[20] J. Benthem. Categorial grammar and lambda calculus. In D. Skordev, editor, *Mathematical Logic and Its Applications*, pages 39–60. Plenum Press, New York, 1987.

[21] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art : The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS series. Springer Verlag, 2004.

[22] Y. Bertot, G. Kahn, and L. Théry. Proof by pointing. In M. Hagiya and J. C. Mitchell, editors, *Proceedings of the International Symposium on Theoretical Aspects of Computer Software*, pages 141–160, Japon, 1994. Springer-Verlag LNCS 789.

[23] R. Blachère. *Eléments de l'Arabe classique*. Maisonneuve Et Larose eds, 1994. ISBN 2706811293.

[24] R. Bonato. *An Integrated Computational Approach to Binding Theory*. PhD thesis, University of Verona, 2006.

[25] B. Carpenter. The Turing-completeness of multimodal categorial grammars. Manuscript, 1995.

[26] B. Carpenter. *Type-logical Semantics*. MIT Press, Cambridge, Massachusetts, 1998.

[27] Y-C Chien and K. Wexler. Children's knowledge of locality conditions in binding as evidence for the modularity of syntax and pragmatics. *Language Aquisition*, 1 :225–295, 1990.

[28] Dong-Ik Choi. Binding principle for long-distance anaphors. *Kansas Working Papers in Linguistics*, 22 :57–71, 1997.

[29] N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1961.

[30] N. Chomsky. *Lectures on Government and Binding*. Holland : Foris Publications, 1981.

[31] N. Chomsky. *The minimalist program*. MIT Press, 1995.

[32] N. Chomsky. Derivation by phase. *Ken Hale : A Life in Language*, 2001.

[33] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76, 1988.

[34] S. Coupet-Grimal. An axiomatization of linear temporal logic in the calculus of inductive constructions. *Journal of Logic and Computation*, 13 :801–813, 2003.

[35] H. B. Curry. Some logical aspects of grammatical structures. *Symposia on Applied Mathematics*, pages 56–68, 1961.

[36] M. Dalrymple, S. M. Shieber, and F. C. N. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14 :399–452, 1991.

[37] P. de Groote. Towards abstract categorial grammars. In *proceedings of ACL*, Toulouse, 2001.

[38] P. de Groote. Towards a montagovian account of dynamics. In *Semantics and Linguistic Theory XVI*. CLC Publications, 2006.

[39] F. De Saussure. *Cours de linguistique générale*. 1916.

[40] D. Delahaye. A tactic language for the system coq. In *Proceedings of Logic for Programming and Automated Reasoning (LPAR)*, pages 85–95. Springer-Verlag LNCS/LNAI, 2000.

[41] D. Delahaye. *Conception de langages pour décrire les preuves et les automatisations dans les outils d'aide à la preuve, Une étude dans le cadre du système Coq*. PhD thesis, Université de Paris VI, Pierre et Marie Curie, 2001.

[42] Y. Devisschere. Réalisation d'une interface utilisateur pour l'atelier icharate. Master's thesis, Université de Bordeaux 1, 2005. Mémoire co-encadré par Pierre Castéran et Houda Anoun.

[43] G. Dowek. Théorie des types. Notes du cours du Master Parisien de Recherche en Informatique, 2004.

[44] D. Dowty. Comments on the paper by bach and partee. In K.J. Kreiman and A.E. Ojeda, editors, *Papers from the Parasession on Pronouns and Anaphora*. Chicago Linguistic Society, 1980.

[45] A. Felty and L. Théry. Interactive theorem proving with temporal logic. *Journal of Symbolic Computation*, 23 :367–397, 1997.

[46] G. Frege. *The Foundations of Arithmetic : A logico-mathematical enquiry into the concept of number*. Blackwell, 1974. Traduction de J. L. Austin.

[47] L.T.F GAMUT. *Language and Meaning, Vol 2 : Intentional Logic and Logical Grammar*. University of Chicago Press, 2001.

[48] J. Y Girard. Linear logic. *Theoretical Computer Science*, 50 :1–102, 1987.

[49] J. Y. Girard, Y Lafont, and L. Regnier. *Advances in Linear Logic*. London Mathematical Society Lecture Notes. Cambridge University Press, 1995.

[50] J. Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.

[51] Michael Gordon, Robin Milner, and Christopher Wadsworth. *Edinburgh LCF : A mechanized logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[52] B. Curry Haskell and F. Robert. *Combinatory Logic I*. North- Holland, 1958.

[53] P. Hendriks. *Comparatives and Categorial Grammar*. PhD thesis, Utrecht, 1995.

[54] M. Hepple. *The Grammar and Processing of Order and Dependency*. PhD thesis, Edinburgh, 1990.

[55] M. Hepple. Discontinuity and the lambek calculus. In *15th Conference on Computational linguistics*, Kyoto, Japan, 1994.

[56] D. Heylen. *Types and Sorts Resource Logic for Feature Checking*. PhD thesis, Utrecht University, 1999.

[57] A. Heyting. *Intuitionism - an Introduction*. North-Holland, 1971.

[58] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry : Essays on combinatory logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[59] G. Huet. Functional pearl : The zipper. *Journal of Functional Programming*, 7(5) :549–554, 1997.

[60] G. Huet. Zen and the art of symbolic computing : Light and fast applicative algorithms for computational linguistics. In *Practical Aspects of Declarative Languages, 5th International Symposium*, number 2562 in Lecture Notes in Computer Science, 2003.

[61] G. Jäger. Anaphora and quantification in categorial grammar. In A. Lecomte and M. Moortgat, editors, *Logical Aspects of Computational Linguistics*, Berlin, 1998. Springer-Verlag.

[62] A. Joshi, L. Levy, and M. Takahash. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10 :136–163, 1975.

[63] A. Joshi, K. Vijay-Shanker, and D. Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Shells, S. M. Shieber, and T. Wasow, editors, *Fundational issues in natural language processing*. MIT Press, 1991.

[64] R. Kayne. Pronouns and their antecedents. In S.D. Epstein & T.D. Seely, editor, *Derivation and Explanation in the Minimalist Program*. Blackwell, 2002.

[65] H. Koopman and A. Szabolcsi. Verbal complexes. In *Current series in Linguistic Theory*. MIT Press, 2000.

[66] N. Kurtonina. *Frames and Labels. A Modal Analysis of Categorial Inference*. PhD thesis, OTS Utrecht, ILLC Amsterdam, 1995.

[67] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65 :154–170, 1958.

[68] J. Lambek. On the calculus of syntactic types. In R. Jacobson, editor, *Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics*, volume XII, pages 166–178. American Mathematical Society, 1961.

[69] A. Lecomte. Rebuilding mp on a logical ground. In *Research on Language and Computation*, pages 27–55. Kluwer Academic, 2003.

[70] A. Lecomte and C. Retoré. Extending lambek grammars : a logical account of minimalist grammars. In *39th Annual Meeting of the Association for Computational Linguistics*, pages 354–362, Toulouse, 2001.

[71] P. Lescanne. Mechanizing epistemic logic with coq. Technical report, ENS Lyon, 2004. http ://perso.ens-lyon.fr/pierre.lescanne/publications.html.

[72] The Coq development team. *The Coq reference manual*. LogiCal Project, 2007. Version 8.1.

[73] R. Montague. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press, New Haven, 1974.

[74] M. Moortgat. *Categorial investigations. Logical and Linguistic aspects of the Lambek Calculus*. PhD thesis, Université d'Amsterdam, 1988.

[75] M. Moortgat. Generalized quantification and discontinuous type constructors. In W. Sijtsma and A. van Horck, editors, *Proceedings Tilburg Symposium on Discontinuous Constituency*, De Gruyter, 1991.

[76] M. Moortgat. In situ binding : A modal analysis. In Paul Dekker and Martin Stokhof, editors, *Proceedings 10th Amsterdam Colloquium*, pages 539–549, Amsterdam, 1996. ILLC.

[77] M. Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier/MIT Press, 1997.

[78] M. Moortgat. Constants of grammatical reasoning. In Gosse Bouma, Erhard Hinrichs, Geert-Jan Kruijff, and Richard T. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*, pages 195–219. CSLI, Stanford, 1999.

[79] M. Moortgat. Labelled deduction in the composition of form and meaning. *Logic Language and Reasoning*, 1999.

[80] M. Moortgat. Meaningful patterns. In de Rijke Gerbrandy, Marx and Venema, editors, *JFAK : Essays dedicated to Johan van Benthem on the occasion of his 50th birthday*. Institute of Logic, Language and Computation, University of Amsterdam, 1999.

[81] R. Moot. *Proof Nets for Linguistic Analysis*. PhD thesis, Utrecht Institute of Linguistics OTS, Utrecht University, 2002.

[82] G. Morrill. Intensionality and boundedness. *Linguistics and Philosophy*, 13 :699–726, 1990.

[83] G. Morrill. *Type Logical Grammar : Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

[84] G. Morrill. Type-logical anaphora. Technical report, 2002.

[85] G. Morrill and J. Merenciano. Generalising discontinuity. In *Traitement automatique des langues*, volume 37, 2, pages 119–143. 1996.

[86] G. Morrill and T. Solias. Tuples, discontinuity, and gapping in categorial grammar. In *the 6th Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, 1993.

[87] R. Muskens. Language, lambdas, and logic. In *Resource Sensitivity in Binding and Anaphora*, Studies in Linguistics and Philosophy. Kluwer, 2003.

[88] R. Oehrle. Term-labeled categorial type systems. *Linguistics and Philosophy*, 17 :633–678, 1994.

[89] R. Oehrle. Some 3-dimensional systems of labelled deduction. *Bulletin of IGPL*, 3 :429–448, 1995.

[90] C. Paulin-Mohring. Inductive definitions in the system Coq - rules and properties. In M. Bezem and J.-F. Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993. LIP research report 92-49.

[91] M. Pentus. Lambek grammars are context free. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 429–433, Montreal, Canada, 1993.

[92] J. Y. Pollock. *Langage et cognition : Introduction au programme minimaliste de la grammaire générative*. Presses Universitaires de France -PUF, 1998.

[93] D. Prawitz. *Natural Deduction. A Proof-Theoretic Study*. PhD thesis, université de Stockholm, 1965.

[94] A. Radford. *Minimalist Syntax : Exploring the structure of English*. Cambridge University Press, 2004.

[95] A. Ranta. Grammatical framework, 2004. Home page : `http ://www.cs.chalmers.se/~aarne/GF/`.

[96] T. Reinhart. Coreference and bound anaphora. *Linguistics and Philosophy*, 6 :47–88, 1983.

[97] C. Retoré. The logic of categorial grammars : Lecture notes. Technical report, INRIA, 2005. http ://www.inria.fr/rrrt/rr-5703.html.

[98] J-R. Ross. Constraints on variables in syntax. In *PhD Dissertation*. MIT, 1967.

[99] K. Ryding. *A Reference Grammar of Modern Standard Arabic*. Cambridge University Press, 2005.

[100] S. M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8 :333–343, 1985.

[101] E. Stabler. Derivational minimalism. In Alain Lecomte, editor, *LACL97*, volume 1582 of *Lecture Notes in Computer Science*. Springer, 1997.

[102] E. Stabler. Remnant movement and structural complexity. In G.J. Kruijff G. Bouma, H.Hinrichs and R. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*. CSLI, Stanford, 1999.

[103] M. Steedman. *Surface Structure and Interpretation*. The MIT Press, 1996.

[104] A. Szabolcsi. Bound variables in syntax (are there any ?). In Gronendijk, Stokhof, and Veltman, editors, *Sixth Amsterdam Colloquium*, pages 331–351, 1987.

[105] A. S. Troelstra. *Lectures on Linear Logic*. CSLI Lecture Notes, Standford, 1991.

[106] W. Vermaat. Controlling movement. Minimalism in a deductive perspective. Master's thesis, Utrecht University, Utrecht, 1999.

[107] W. Vermaat. *The Logic of Variation : A cross-linguistic account of wh-question formation*. PhD thesis, Utrecht University, Utrecht, 2005.

[108] S. Wintner. Definiteness in the hebrew noun phrase. *Journal of Linguistics*, 36, 2000.

Glossaire

Nous récapitulons dans ce qui suit, la signification des différentes abréviations employées dans cette thèse.

Abréviations	Significations
AB	Grammaires catégorielles d'Ajdukiewic et Bar-Hillel
AS	Arabe Standard
CCI	Calcul des constructions inductives
CNC	Contrainte du NP-complexe
CSC	Contrainte de structure de coordination
GLE	Grammaires linéaires étiquetées
IIELL	Logique linéaire intuitionniste implicative et exponentielle
L	Logique associative et non-commutative de Lambek
LP	Logique associative et commutative de Lambek
MG	Grammaires minimalistes
MMCG	Grammaires catégorielles multimodales
MP	Programme minimaliste
NL	Logique non-associative et non-commutative de Lambek
NLP	Logique commutative et non-associative de Lambek
PSG	Grammaires à structure de phrase
TAL	Traitement automatique de langues
TLG	Grammaires de type logique

Un panorama du système Coq

L'outil **Coq** [72, 21], est un système conçu pour developper des preuves mathématiques, écrire des spécifications formelles et vérifier la correction des programmes par rapport à leurs spécifications. En se basant sur l'isomorphisme de Curry-Howard, les programmes, les propriétés et les preuves sont tous formalisés dans le même langage dit *Calcul des Constructions Inductives*, qui est une variante enrichie du λ-calcul typé. L'originalité majeure du système **Coq** réside dans sa manière de considérer tous les jugements logiques comme étant des jugements de typage. En effet, afin de démontrer une proposition P il suffit d'exhiber un λ-terme de type P (i.e., chercher un habitant du type P). La validation de la preuve ainsi construite est prise en charge par l'algorithme de vérification de types qui respecte bien le critère de de Bruijn (i.e., l'algorithme est simple et court ; il peut être facilement vérifié à la main) [18]. Ceci permet de justifier la confiance accordée à la fiabilité du système **Coq**.

B.1 Termes et types

Dans ce qui suit, nous n'envisageons pas de donner une présentation complète du Calcul des Constructions Inductives. Notre objectif est plutôt d'illustrer le pouvoir expressif de ce calcul de telle façon à faciliter la compréhension du chapitre 3 aux non-spécialistes de **Coq**.

Soit $t : T$ le jugement de typage pouvant être paraphrasé par " t est un terme bien typé de type T ". Formellement, il est nécessaire de préciser un *environnement* E et un *contexte* Γ composé des différentes déclarations des identifiants intervenant éventuellement dans t ou T, la notation complète devrait être $E[\Gamma] \vdash t : T$. Toutefois, dans un but de simplicité, nous nous contenterons d'employer la notation abrégée.
Précisons que dans le Calcul des Constructions Inductives, tout type est considéré à son tour comme un terme, et en tant que terme il a aussi un type. Le type d'un type prend la forme d'un identificateur nommé *sorte*. Dans cette annexe, nous utiliserons en particulier la sorte `Set`, qui est le type des structures de données, ainsi que `Prop`, qui est le type des propositions logiques. Les sortes `Prop` et `Set` font partie toutes les deux de la sorte `Type`[1].

[1] Il n'y'a pas de raison pour s'arrêter là. Le type de `Type` est la sorte (nommée aussi *univers*) `Type(1)`, et ainsi de suite ...

B.1.1 Recueil de termes typés

Nous présentons dans cette section une série de jugements de typage commentés qui illustre la richesse du système de types de **Coq**.

Nombres

Nous commençons par un exemple très simple de jugement de typage à savoir " `3 :nat` ", où `nat` est le type associé aux nombres naturels. Précisons qu'outre les entiers naturels, l'outil **Coq** comprend des bibliothèques spécifiques permettant la définition des entiers et des réels.

Fonctions

Soient T et T' deux types, $T \to T'$ représente le type des fonctions *totales* de T vers T'. Le terme " `fun (`x `:` T`)⇒ t'` " est de type $T \to T'$ si t' porte le type T' lorsque x est de type T. Un tel terme correspond à la notation classique $\lambda\, x : T.t'$. L'opérateur $\to$ est associatif à droite, ainsi le type $T \to T' \to T''$ représente la forme abrégée de $T \to (T' \to T'')$. A titre d'exemple, la fonction d'addition des nombres naturels, portant l'identifiant `plus` est de type `nat→nat→nat`, alors que la fonction " `fun (n :nat) ⇒ n + n` " est de type `nat→nat`[2] .

L'application d'une fonction $f : T \to T'$ à un argument $x : T$ produit un terme noté " $f\ x$ " qui est de type T'. L'application fonctionnelle est associative à gauche, ainsi l'application " $f\ x\ y$ " est une abréviation de " $(f\ x)\ y$ ".

Types Inductifs

En **Coq**, on a souvent recours à l'utilisation des *types inductifs* qui sont définis par une énumération de *constructeurs*. Chaque habitant d'un type inductif I est construit moyennant un nombre fini d'applications des constructeurs de I. Parmi les exemples simples, citons le type des valeurs booléennes qui possède deux constructeurs et le type des entiers naturels doté d'un constructeur récursif permettant de former une infinité de termes de type `nat` : `O`, `S O`, `S (S O)`, `S (S (S O))`, etc.[3]

```
Inductive bool : Set := true | false.

Inductive nat : Set :=
| O : nat  (* zero *)
| S : nat -> nat. (* successeur *)
```

Le système **Coq** associe à chaque définition inductive un ensemble de théorèmes utilisés lors d'un raisonnement par récurrence. A titre d'exemple, le principe de récurrence `nat_ind` est engendré automatiquement suite à la définition du type inductif `nat` :

[2]Comme c'est le cas pour plusieurs langages de programmation fonctionnelle, les fonctions sont "currifiées" ; en outre, la notation infixe "n+p" est une abréviation de "plus n p".

[3]Heureusement, le système **Coq** utilise les notations classiques (0, 1, 2, 3, etc.) pour représenter ces termes.

```
nat_ind : forall P : nat -> Prop,
           P O -> (forall n : nat, P n -> P (S n)) ->
           forall n : nat, P n
```

En outre, **Coq** permet l'utilisation de la technique de filtrage (i.e., *pattern matching*) afin de définir les fonctions effectuant un traitement par cas sur la valeur d'une expression dont le type est inductif. Par exemple, les opérations élémentaires sur les entiers naturels peuvent être facilement définies grâce au filtrage et à la récurrence comme ceci :

```
Fixpoint plus (n p : nat) {struct n} : nat :=
  match n with
  |  O => p
  |  S n' => S (plus n' p)
  end.

Fixpoint mult (n p : nat) {struct n} : nat :=
  match n with
  | O => O
  | S n' => plus p (mult n' p)
  end.

Eval compute in (mult 6 6).
36:nat
```

Rappelons que dans le Calcul des Constructions Inductives, les fonctions considérées sont totales, ainsi toute définition récursive devrait avoir au moins un argument strictement décroissant par rapport à un ordre bien fondé. Dans le cas de la récurrence structurelle (i.e., la relation d'ordre est celle de sous-terme), la notation `{struct n}` précise que l'argument décroissant coïncide avec `n`.

Il est intéressant de noter que les types inductifs peuvent être paramétrés notamment par d'autres types. Nous présentons ci-après la définition d'un type polymorphe associé aux arbres binaires dont les feuilles sont étiquetées par des éléments du type *A*.

```
Inductive tree (A:Set):Set :=
| leaf : A -> tree A
| bin : tree A -> tree A -> tree A.
```

Grâce à cette définition générique, on est capable de considérer des arbres décorés par des entiers naturels :“ `tree nat` ”, ou par des fonctions sur les entiers : “ `tree (Z→Z)`”, etc.

Citons aussi un autre type inductif fort utile permettant de construire des termes qui dénotent soit une valeur de type *A* soit aucune valeur du tout : il s'agit du type `option`. Un tel type est utilisé pour définir des fonctions partielles : une fonction partielle f de *A* vers *B* peut être représentée par une fonction totale qui associe à x la valeur `Some` y si $f(x) = y$ et `None` si x ne fait pas partie du domaine de f. La fonction `left_son` définie ci-dessous en est un exemple.

```
Inductive option (A:Type):Type :=
```

```
| Some : A -> option A
| None : option A.

Definition left_son (A:Set) (t: tree A): option (tree A) :=
  match t with
  | (bin  t1 _) => Some t1
  | _           => None
  end.
```

Produits dépendants

Le *produit dépendant* est une construction de type qui généralise la flèche $A \to B$. Il permet de définir des fonctions dont le type de retour dépend de la valeur de leur argument.

Par exemple, la fonction `left_son` définie ci-dessus renvoit un résultat de type `option(tree` T`)` qui dépend de l'argument T choisi. Dans le Calcul des Constructions Inductives, cette fonction porte le type ci-après qui n'est autre qu'un produit dépendant :

$$forall\ A : Set, tree\ A \ \to \ option\ (tree\ A)$$

Types dépendants

Un *type dépendant* est un type dont l'expression contient des termes arbitraires. Par exemple, " un vecteur de taille n ", " un nombre premier supérieur à n " sont des types dépendants. Ce genre de types s'avère très utile notamment pour construire des spécifications.

La définition du type polymorphe associé aux vecteurs se fait comme suit : pour tout type de donnée A, `null_vector` est un vecteur de taille 0, et pour tout entier naturel n, si $a : A$ et v est un vecteur de taille n, alors `(cons_vector` $A\ n\ a\ v$`)` est un vecteur de taille $n + 1$.

```
Inductive vector (A:Set): nat -> Set :=
| null_vector : vector A O
| cons_vector : forall (n:nat), A -> vector A n -> vector A (S n).
```

B.2 Propositions et preuves

Une *proposition* est n'importe quel type de sorte `Prop`. Le système **Coq** suit l'approche intuitionniste proposée by Heyting [57], qui remplace la question "est ce que la proposition P est vraie ?" par la question "quelles sont les preuves éventuelles de P ?". Les preuves des propositions sont représentées par des termes du λ-calcul. Par exemple, une preuve de $P \Rightarrow Q$ est une fonction qui *construit* une preuve de Q à partir d'une preuve arbitraire de P.

L'isomorphisme de *Curry–Howard* établit une correspondance entre la programmation fonctionnelle et les démonstrations en déduction naturelle [49, 52, 58]. D'un point de vue pratique, cette correspondance nous permet d'utiliser les mêmes idées et techniques aussi bien pour le raisonnement que pour la programmation.

En utilisant les types dépendants, il est facile de formaliser un *prédicat* sur un type T par une fonction de type $T \rightarrow$ `Prop`. A titre d'exemple, le prédicat < est une fonction de type `nat→nat→Prop` et le prédicat "est supérieur à 36" est ainsi représenté par l'abstraction " `fun n :nat ⇒ 36 < n` ".

B.3 Exemples

Plusieurs connecteurs logiques sont définis par des types inductifs de sorte `Prop`. A titre d'exemple, la disjonction intuitionniste est encodée par un type inductif à deux constructeurs, représentant les deux règles d'introduction de ce connecteur logique. Comme c'est le cas pour les autres connecteurs, la règle d'élimination est automatiquement engendrée par **Coq**.

```
Inductive or (A : Prop) (B : Prop) : Prop :=
  | or_introl : A -> A \/ B
  | or_intror : B -> A \/ B.

Check or_ind.
or_ind: forall A B P : Prop, (A -> P) -> (B -> P) -> A \/ B -> P
```

La quantification universelle (de premier ordre ou d'ordre supérieur) est exprimée sous la forme d'un produit dépendant. Par exemple, le principe de récurrence `nat_ind` présenté antérieurement, contient deux quantifications : une sur les entiers naturels et une autre sur les prédicats.

Remarquons que les propositions définies inductivement peuvent être vues comme des généralisations des définitions à la *Prolog* : chaque constructeur est associé à une clause.

Par exemple, la figure 60 montre les deux définitions de la relation d'ordre total ≤ respectivement en *Prolog* et en **Coq** :

```
X <= X.
X <= s Y :- X <= Y
```

```
Inductive le (n:nat): nat-> Prop :=
| le_n : n <= n
| le_S : forall m:nat, n <= m -> n <= S m.
```

FIG. 60: Définition de la relation ≤ en Prolog et en Coq

Comme pour tout type inductif, **Coq** associe à chaque prédicat défini de manière inductive un principe de récurrence. Dans le cas de ≤, le système génère automatiquement le schéma illustré par la figure 61[4].

[4]Pour plus de lisibilité, nous omettons les quantificateurs universels portant sur P, n et p.

$$\frac{n \leq p \quad P(n) \quad \forall q \;\; n \leq q \Longrightarrow P(q) \Longrightarrow P(S(q))}{P(p)} \texttt{ le\_ind}$$

FIG. 61: Schéma de récurrence associé à ≤

B.4 Tactiques et Automatisation

Le système **Coq** est basé sur une logique d'ordre supérieur, il est ainsi irréaliste d'attendre de lui une automatisation complète de la recherche de preuves. Néanmoins, étant donné que les démonstrations sont représentées par des λ-termes généralement complexes, ca serait inapproprié d'exiger de l'utilisateur de saisir lui même de tels termes.

Comme c'est le cas pour un grand nombre d'assistants de preuves qui suivent l'approche *LCF* [51], la démonstration d'un théorème ou la réalisation d'une spécification est effectuée moyennant l'emploi de *tactiques* ou *tacticielles*, qui aident l'utilisateur dans sa construction d'un habitant d'un type donné en appliquant la stratégie *diviser pour règner*.

Le problème de recherche d'un habitant d'un type donné T est nommé *but*. Trouver une *solution* à un but consiste à déterminer un terme t dont le type est T. Les tactiques sont ainsi utilisées pour remplacer un but T par une séquence finie de *sous-buts* $T_1, \ldots, T_n$, potentiellement plus faciles à résoudre, dont les solutions respectives $t_1, \ldots, t_n$ contribueront à la construction de la solution t.

Il existe un large éventail de tactiques prédéfinies en **Coq**, allant des tactiques élémentaires qui implémentent les règles logiques de base (introduction, élimination) aux procédures de décision complexes appliquées pour des fragments spécifiques (Arithmétique de Presburger, simplification des expressions dans les anneaux, semi-décision de la logique de premier ordre, etc.)

D'une façon générale, les tactiques ne sont pas censées être complètes : elles peuvent très bien échouer quand elles sont invoquées en dehors de leur domaine d'application, mais elles ne peuvent, en aucun cas, contribuer à la construction de preuves erronées.

Finalement, précisons que **Coq** est muni d'un langage de programmation pour les tactiques, nommé `Ltac` [41, 40], qui permet aux développeurs de définir des tactiques spécialisées dans un domaine particulier. Grâce à cet outil, l'impossibilité d'atteindre une "déduction complètement automatisable" est palliée par la plausibilité d'une "déduction assistée et programmable".

Grammaires Minimalistes : un tour d'horizon

Cette annexe est dédiée à une description succincte et informelle des Grammaires Minimalistes (**MG** en abrégé) ; elle est plutôt destinée aux lecteurs qui ne sont pas familiers avec ce formalisme.

C.0.1 Définitions de base

Le Programme Minimaliste (**MP** en abrégé, i.e., *minimalist program*) [31, 94, 92] est un modèle récent développé par Chomsky dans le cadre de sa grammaire générative. Il est qualifié de *minimaliste* dans la mesure où il est basé sur un ensemble restreint de primitives qui reflètent les principes de la grammaire universelle. Ce modèle a été formalisé de manière rigoureuse par Stabler et ce moyennant un système dérivationnel défini dans [101, 102]. Ledit système manipule des structures linguistiques organisées sous la forme d'arbres binaires mettant en évidence la hiérarchisation des constituants. Chaque structure arborescente admet un unique pivot lexical représentant l'élément le plus important de l'énoncé ; une telle feuille est nommée la *tête*. La tête est repérée grâce aux nœuds internes employés par les arbres binaires, à savoir, les deux pointeurs complémentaires > et <. Ainsi, le nœud < indique que la tête est située dans le sous-arbre gauche alors que > précise que c'est le sous-arbre droit qui comprend la tête en question. A titre d'exemple, le schéma de la figure 62 modélise la représentation symbolique de l'énoncé '*Houda peut parler trois langues*'. Pour une raison de compacité, les deux représentations phonétique (i.e., /*mot*/) et sémantique (i.e., (**mot**)) d'un élément lexical '*mot*' sont encapsulées dans le même terme 'mot'.

La tête de cet arbre est soulignée, elle coïncide avec l'*inflexion* du verbe. Cette dernière est une catégorie fonctionnelle utilisée pour représenter le temps et l'accord verbal, elle est considérée comme le site d'accueil des verbes modaux (e.g., *pouvoir*, *vouloir* ...) et des auxiliaires. Le sous-arbre placé à droite de la tête (le groupe verbal infinitif) est nommé *complément* alors que celui qui est situé avant la tête (le sujet) est dit *spécifieur*.
La *projection maximale* d'une tête locale η (i.e., d'une feuille quelconque) est définie comme étant le plus grand constituant (i.e., sous-arbre) qui admet η comme tête. La projection maximale représente ainsi une construction *endocentrique* dans la mesure où elle hérite de sa tête certaines propriétés syntaxiques cruciales comme le nombre ou le genre. A titre d'exemple, la projection maximale du déterminant numéral '*trois*' n'est autre que le groupe nominal '*trois langues*', par contre, la projection maximale de la tête globale '*peut*' coïncide bien avec la phrase entière.
Les grammaires minimalistes sont totalement lexicalisées, ceci revient à dire que la façon dont

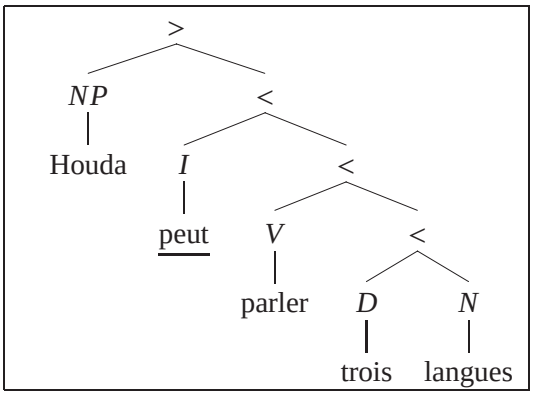


FIG. 62: Exemple de structure linguistique manipulée par les **MG**

les mots se combinent pour former des phrases est uniquement déterminée par leurs catégories syntaxiques intrinsèques. C'est le lexique qui permet d'assigner à chaque mot une séquence finie et ordonnée de traits qui seront progressivement consommés dans une dérivation.
La génération de nouvelles constructions se fait grâce à deux primitives basiques indépendantes de la langue considérée à savoir la *fusion* et le *déplacement*.

C.0.2 Opération de fusion

Intuitivement, la fusion permet d'assembler deux constituants construits antérieurement pour former une composante plus riche. Cette combinaison est déclenchée par la présence d'un trait particulier de sélection [= u] exprimant une demande d'une entité complète de type atomique [u] (e.g., = n reflète une demande d'un nom commun). Une fois les deux constituants fusionnés ensemble, les traits catégoriels impliqués (= u et u) deviennent inactifs.
Cette opération se trouve à l'origine de la combinaison d'un déterminant avec son nom produisant ainsi un syntagme nominal. Elle intervient également dans la formation d'un groupe verbal à partir d'un verbe transitif et son objet, aussi bien que dans la construction d'une phrase à partir d'un syntagme verbal et d'un sujet. Le schéma de la figure 63 illustre ces différentes fusions sur un exemple concret.

La fusion de deux arbres σ_1 et σ_2 n'est possible que si la première catégorie syntaxique active associée à la tête de σ_1 est un sélecteur (i.e., de la forme = u) dont le rôle est d'annuler le trait catégoriel u qui marque la tête de σ_2. Après la fusion, les deux traits de polarités opposées sont désactivés (barrés sur le schéma). A la fin de la dérivation, seul le trait c doit subsister pour marquer la grammaticalité de l'énoncé analysé.
La procédure de fusion se manifeste différemment suivant la nature de l'arbre σ_1. En effet, si ledit arbre est réduit à une feuille (e.g., les deux premiers cas du schéma), l'arbre σ_2 est placé en position de complément. En revanche, si l'arbre σ_1 est composé (e.g., dernier cas du schéma), σ_2 occupe plutôt la position de spécifieur. Dans les différents cas de figure, il est intéressant de noter que la tête du constituant final obtenu après fusion coïncide avec la tête de l'arbre σ_1.
L'opération de fusion admet une deuxième variante permettant de rendre compte du phénomène de *mouvement de tête*. Cette version manipule des sélecteurs forts (= U ou U =) qui sont capables d'attirer la composante phonétique occupant la tête de l'arbre σ_2 afin de l'adjoindre à la tête de

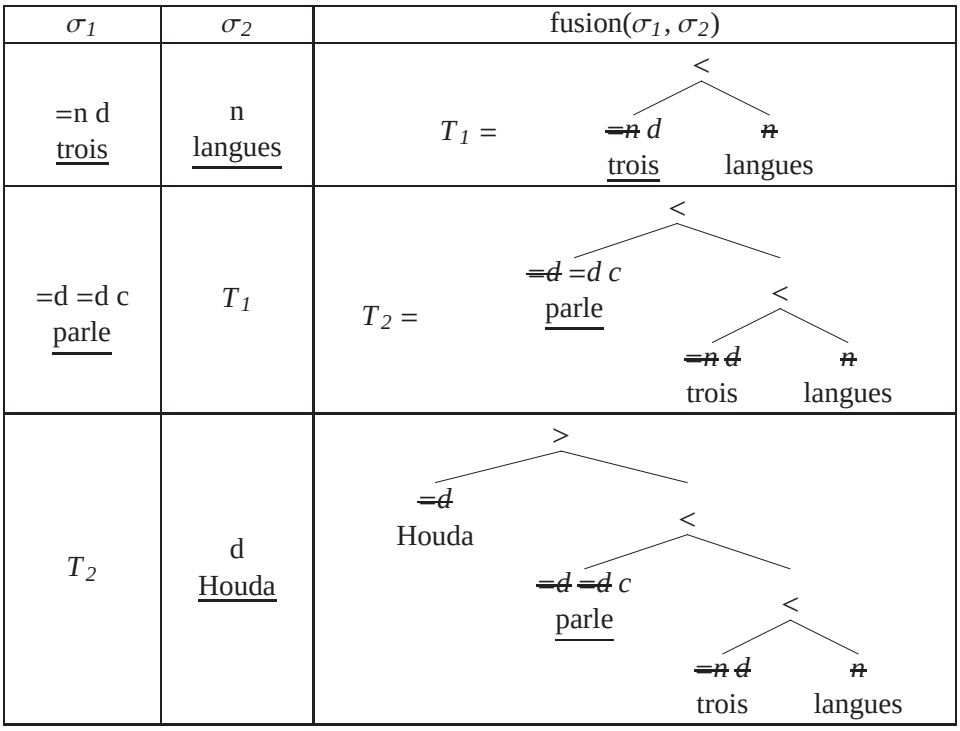


FIG. 63: Illustration du fonctionnement de l'opération de fusion

σ_1 [101]. Un tel phénomène est nécessaire pour expliquer l'inversion du verbe avec son sujet qui a lieu dans certaines phrases interrogatives telles que '*Aimes-tu la logique ?*'.

C.0.3 Opération de déplacement

Outre la fusion, les **MG** sont basées sur une seconde opération dite de déplacement. Cette opération permet d'expliquer une imperfection notoire qui caractérise les langues naturelles à savoir *la propriété de dislocation* [94]. Ladite propriété survient lorsqu'une expression est interprétée à un endroit différent de celui où elle apparait phonétiquement. Ce phénomène est, à titre d'exemple, illustré par les phrases interrogatives telles que '*Quelle langue Sam peut parler ?*' où la qu-expression (*wh*-expression) '*quelle langue*' joue toujours la fonction d'objet de l'infinitif '*parler*' même si elle est déplacée au début de la phrase. Cette dernière hypothèse est confirmée par l'absence totale de dislocation dans les questions *qu-in-situ* où les qu-éléments restent figés dans leurs positions d'origine. Le dialogue simple ci-après montre un exemple de question in-situ, dite question *écho*, qui est adressée par le locuteur (2) au locuteur (1) suite à une incompréhension, lui demandant implicitement de répéter son dernier énoncé.

Locuteur (1) : Sam peut parler le Farsi
Locuteur (2) : Sam peut parler *quelle langue* ?

La formulation du locuteur (2) soutient l'idée que la qu-expression occupait initialement la position complément du verbe infinitif '*parler*' avant de se déplacer en position frontale. Cette opération de déplacement est déclenchée par la nécessité de vérifier des traits ininterprétables représentés sous la forme d'*assignés* [-*v*] qui peuvent être annulés par des traits *assignateurs forts* [+*V*]. Ainsi, les qu-constituants portent un trait spécial [−*qu*] qui a besoin d'être vérifié par le trait [+*Qu*] associé à la tête non lexicale *c* de la proposition interrogative. Ce processus de vérification de traits engendre le déplacement du syntagme interrogatif en position spécifieur de la tête *c* et s'achève par la désactivation des traits formels opposés (-*v*, +*V*). D'après la théorie des traces de Chomsky [**?**], le constituant déplacé laisse derrière lui une *trace* co-indexée qui occupe son site d'origine (le site d'extraction). Cette trace est vue comme une *catégorie vide* qui n'a pas de réalisation phonétique mais qui hérite les propriétés sémantiques pertinentes du constituant auquel elle est liée. L'élément déplacé et ses différentes traces forment ainsi un objet discontinu qu'on nomme une *chaine*.
La figure 64 illustre l'application de l'opération de déplacement permettant d'obtenir la question qu-ex-situ précédente.

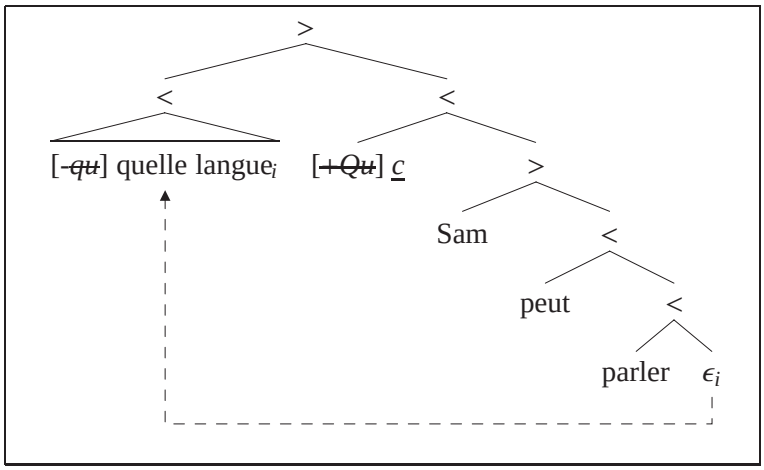


FIG. 64: Déplacement dans une question qu-ex-situ

Cette même technique est employée pour déplacer le sujet en dehors du VP-shell (i.e., la projection maximale de la tête *V* occupée par le verbe) [101, 94] et l'insérer en position de spécifieur de l'inflexion. Ce mouvement est motivé par la nécessité de vérifier le cas nominatif [-*nom*] du sujet par la tête *I* portant l'assignateur [+*NOM*] comme l'illustre la figure 65 ci-dessous.

D'une façon générale, l'opération de déplacement est contrainte par deux principes dits de *dernier recours* (i.e., last resort) et d'*avarice* (i.e., greed) [92] selon lesquels un syntagme ne se déplace qu'en cas de nécessité et ce pour vérifier son trait ininterprétable [-*v*]. La vérification d'un tel trait se fait si la tête de la structure arborescente globale est marquée par l'assignateur [+*V*]. Cette procédure entraine ainsi une réorganisation structurelle de l'énoncé qui s'effectue en déplaçant la projection maximale portant le trait [−*v*] et l'insérant en position de spécifieur de l'arbre. Une telle restructuration s'achève par la désactivation des deux traits formels complémentaires (−*v* et +*V*). Afin de se garantir un certain déterminisme lors de l'application de ce processus, on suppose qu'à tout moment de la dérivation, on ne peut pas trouver deux candidats potentiels portant le même trait [-*v*]. Cette hypothèse est une variante simplifiée du principe de *distance minimale* (i.e., shortest move condition) [92, 101].

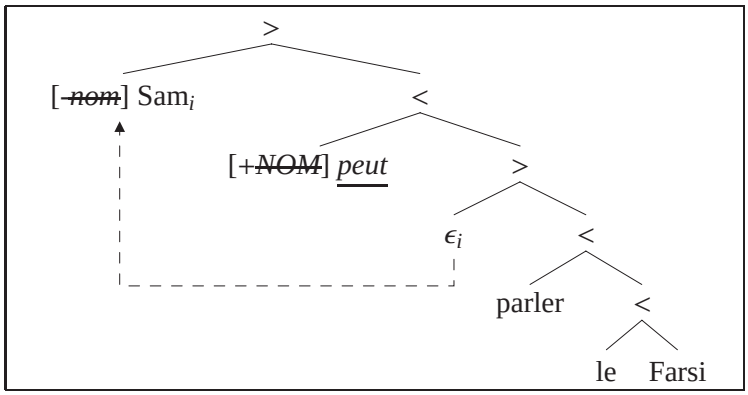


FIG. 65: Déplacement du sujet en dehors du VP-shell

L'opération de mouvement admet une deuxième variante dite *furtive* (ou invisible) permettant de déplacer uniquement la composante sémantique tout en gardant la forme phonétique dans son site d'origine. Cette version est appliquée lorsque le trait assignateur impliqué est *faible* [+*v*]. Elle est utilisée à titre d'exemple pour rendre compte des questions qu-in-situ ainsi que pour vérifier le cas accusatif du complément d'objet direct (qui reste toujours à l'intérieur du VP-shell).

Les primitives des (**MG**) ont pour objectif d'éliminer progressivement les paires de traits opposés ($= u| = U|U = , u$) et ($+V| + v , -v$). Chaque trait formel ne peut consommer qu'un seul et unique trait complémentaire. Il est donc naturel de songer à exprimer ces mécanismes dans une logique sensible aux ressources notamment la logique **IIELL**. Un tel plongement est bénéfique étant donné qu'il propose une interface syntaxe/sémantique transparente, chose qui permettra de pallier le flou planant autour du calcul de la forme logique dans les (**MG**).

Index

Approche logique des grammaires pour les langues naturelles

Résumé : Les contributions majeures de cette thèse s'articulent autour des trois axes de base de la linguistique computationnelle, à savoir la logique, la linguistique et l'informatique. Nous proposons ainsi un nouveau système non-directionnel $\mathcal{GLE}$ permettant de simuler les opérations transformationnelles du Programme Minimaliste dans un cadre logique qui fait appel au raisonnement hypothétique de manière contrôlée. La pertinence de ce formalisme est soulignée en montrant sa capacité de prendre en charge des phénomènes linguistiques complexes, nécessitant un partage contraint de ressources, tels que le liage d'anaphores ou la résolution d'ellipse. En outre, nous présentons un atelier logique $\mathcal{ICHARATE}$ destiné à la recherche et l'enseignement de la linguistique computationnelle. Cet outil est composé de bibliothèques pour l'assistant à la démonstration Coq, qui comprennent la formalisation de systèmes logiques avancés dédiés au traitement des langues naturelles, dont la logique multimodale.

Mots-clés : Linguistique computationnelle, grammaires catégorielles de type logique, grammaires minimalistes, interface syntaxe-sémantique, assistant de preuves **Coq**.

A logical approach to grammar for natural languages

Abstract : The major contributions of this thesis are articulated around the three basic axes of computational linguistics, namely logic, linguistics and computer science. We propose a new undirected system $\mathcal{GLE}$ which allows to simulate the transformational operations of Minimalist Program within a logical setting. The relevance of this formalism is underlined by showing its ability to deal with complex linguistic phenomena, requiring constrained resource sharing, such as anahora binding or ellipsis resolution. Moreover, we present a logical framework $\mathcal{ICHARATE}$ intended for the study of computational linguistics. This tool is composed of libraries, built upon the **Coq** proof assistant, which include the formalization of advanced logical systems dedicated to natural language analysis, such as multimodal logic.

Keywords : Computational linguistics, type logical grammars, minimalist grammars, syntax-semantics interface, **Coq** proof assistant.

Thèse en informatique, préparée au LaBRI
Université Bordeaux I
351 cours de la Libération
33405 Talence