Une bibliothèque Coq pour le traitement des langues naturelles

Houda ANOUN[1]

1: LaBRI & Bordeaux 1
anoun@labri.fr

Résumé

Cet article montre comment le Calcul des Constructions Inductives (**CCI**), qui est l'assise de l'assistant de preuves **Coq**, peut contribuer à l'étude des langues naturelles. Cette étude concerne particulièrement les grammaires catégorielles et l'interface syntaxe/sémantique.

1. Logiques pour le traitement des langues naturelles

1.1. Introduction

L'utilisation de la logique pour le traitement des langues naturelles a été introduite en 1958 par le mathématicien **Joachim Lambek** [Lambek58] dans le système déductif qui porte son nom. Différentes extensions ont été proposées depuis lors pour adapter le calcul de Lambek à l'analyse de phénomènes divers et complexes de la langue naturelle, ceci a donné naissance aux grammaires *multimodales*. Les différents formalismes proposés sont caractérisés par leur richesse, leur grand degré de paramétrabilité, leur capacité générative raisonnable et leur interface syntaxe/sémantique. Néanmoins, ces formalismes sont très complexes : leur utilisation et manipulation s'avère une tâche ardue pour les novices.
Plusieurs outils d'analyse ont été réalisés pour automatiser les preuves en logique multimodale parmi lesquels l'analyseur **Grail** implémenté par Richard Moot en **prolog** [Moot02]. Le problème de recherche de démonstration en logique multimodale étant indécidable, l'outil **Grail** ne peut manipuler qu'une sous-classe de cette logique où la recherche de preuves devient décidable mais d'une très grande complexité [Moot02]. En outre, cet analyseur ne peut effectuer des dérivations qu'une fois que les paramètres de la grammaire sont fixés.
Il serait donc intéressant d'avoir un outil qui prenne en charge toutes les classes de grammaires sans aucune restriction, qui offre aux chercheurs la possibilité d'en étudier les propriétés génériques tout en leur permettant d'effectuer interactivement leurs dérivations pour mieux appréhender le fonctionnement de la logique multimodale. L'assistant de preuves **Coq** [CoqManual, CoqArt04], basé sur la logique d'ordre supérieur et l'isomorphisme de **Curry Howard**, s'avère bien adapté à tous ces besoins.
Dans ce qui suit, nous présenterons quelques définitions de base concernant les grammaires catégorielles et leurs extensions multimodales ; par la suite nous décrirons brièvement les différentes composantes de l'atelier $\mathcal{ICHARATE}$ dédié à la formalisation de ces grammaires en **Coq**.

1.2. Préliminaires

Il existe une grande variété de formalismes utilisés dans le domaine de la linguistique computationnelle, entre autres les grammaires *minimalistes* [Chomsky95], les grammaires d'arbres adjoints (*TAG*)

[Joshi75] et les grammaires *catégorielles de type logique* [Lambek58, Moortgat97].
On s'intéresse particulièrement aux grammaires catégorielles étant donné qu'elles entretiennent un lien étroit avec la sémantique. Ces dernières grammaires se distinguent des *PSG*[1] de la hiérarchie de Chomsky par leurs règles de production qui sont indépendantes des terminaux et qui sont formalisées par un système logique, c'est pour cette raison que de telles grammaires sont dites universelles et lexicalisées[2]. Les systèmes déductifs représentant les logiques des grammaires catégorielles ne manipulent pas directement les mots de la langue choisie mais des représentants de ces derniers nommés *catégories* qui ne sont autres que des *types* encapsulant le comportement syntaxique des mots. Ces logiques offrent une interface syntaxe/sémantique grâce à la correspondance de **Curry-Howard** qui établit un lien entre les *formules* et les *types* ainsi qu'entre les *preuves* et les *programmes* vérifiant une certaine spécification. La fonction qui associe à chaque mot de la langue un ensemble fini de *formules* syntaxiques est appelée le *lexique*.

Malgré cette familiarité avec la logique intuitionniste, il s'avère que cette dernière n'est point adaptée au traitement des langues naturelles. En effet, on a plutôt besoin d'une logique sensible aux ressources car le nombre de ressources ainsi que leur ordre est très important dans notre cas. Les règles de permutation, d'affaiblissement et de contraction qui sont appliquées librement dans la logique intuitionniste (car elles font partie des règles de base) doivent être éliminées et remplacées par des règles structurelles dont l'application est contrôlée afin d'éviter le problème de sur-génération.
On retrouve ces critères dans la logique linéraire introduite par Jean-Yves Girard en 1987 [Girard87] et particulièrement dans une de ses variantes non commutatives nommée la logique **L** qui a été définie par Joachim Lambek en 1958 [Lambek58] et qui fera l'objet de ce qui va suivre.
Les types considérés dans la logique **L** 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** qui permet de représenter les syntagmes nominaux, par exemple '*Houda*' et '*la logique*', **n** qui représente les noms communs, par exemple '*logique*' et **s** qui n'est autre que la catégorie syntaxique associée aux phrases syntaxiquement correctes.
La seconde classe est formée de catégories composées à partir des formules atomiques en utilisant des connecteurs logiques. Le connecteur '$\Rightarrow$' de la logique intuitionniste est scindé en deux ($/$, $\backslash$) étant donné l'importance de l'ordre dans notre cas. L'opérateur '$\bullet$' est considéré comme le dual de '$/$' et '$\backslash$'. Intuitivement, la formule $\mathbf{A/B}$ (resp. $\mathbf{B\backslash A}$) représente une entité qui attend une expression de type $\mathbf{B}$ à sa droite (resp. à sa gauche) pour former une expression de type $\mathbf{A}$, par contre, la formule $\mathbf{A{\bullet}B}$ représente la concaténation de deux expressions de types respectifs $\mathbf{A}$ et $\mathbf{B}$. Ainsi, on associe au verbe transitif *ignore* la catégorie syntaxique $\mathbf{(np\backslash s)/np}$ car ce dernier attend un syntagme nominal à sa droite (l'objet) et un autre à sa gauche (le sujet) pour former une phrase.

Les règles d'inférence de la logique **L** manipulent des séquents de la forme ($\Gamma \vdash \mathbf{A}$). Le jugement ($\Gamma \vdash \mathbf{A}$) signifie que la formule syntaxique $\mathbf{A}$ peut être déduite du contexte Γ qui est formé d'une liste de formules regroupées entre elles par le connecteur structurel '(,)'. Citons à titre d'exemple certaines de ces règles décrites dans le style de la déduction naturelle :

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

Dans ces deux règles, les lettres grecques représentent des contextes alors que les lettres latines symbolisent des types syntaxiques. Intuitivement la première règle d'élimination du '$/$' signifie que si une expression de type $\mathbf{A/B}$ est concaténée à une expression de type $\mathbf{B}$ alors on obtient finalement une entité de type $\mathbf{A}$. La seconde règle, quant à elle, permet d'introduire une technique dite de *raisonnement hypothétique* : afin de prouver qu'une expression est de type $\mathbf{A/B}$, il suffit de montrer

[1]Phrase structure grammars
[2]C'est Noam Chomsky qui a conjecturé en 1955 l'existence d'une grammaire universelle innée et partagée par toutes les langues

qu'en concaténant cette expression avec un élément quelconque de type **B** on obtient une entité de type **A**.
En définissant les règles de production sous forme d'un système déductif, l'analyse syntaxique d'une phrase coincide avec la preuve d'un théorème dans la logique considérée. Ainsi, une phrase $m_1 m_2 ... m_n$ est engendrée par notre grammaire ssi :

$$\forall\, i\, \in\, \{1..n\},\, \exists\, t_i\, \in\, lex(m_i) \mid t_1,\, t_2, ...,\, t_n\, \vdash\, s$$

où *lex* est la fonction qui représente le lexique.

Le système **L** assure l'associativité globale du connecteur structurel '(,)'. Ceci est néfaste linguistiquement étant donné la sur-génération qu'il peut causer. En effet, on peut facilement vérifier que la phrase agrammaticale '*(*La pomme que Marie dort et Jean mange est rouge*)' [3] est dérivable en **L** [Moortgat97]. D'autre part, ce dernier est très rigide dans la mesure où aucune commutativité n'est permise, ce qui empêche l'analyse de plusieurs phénomènes linguistiques tels l'extraction médiane[4] et les dépendances croisées [Moortgat97, Moortgat99]. Afin de pallier la première faiblesse, Lambek a introduit un système **NL** où le connecteur '(,)' est privé d'associativité globale, et c'est à partir de ce système qu'on pourrait définir différentes variantes en ajoutant un ensemble de règles structurelles qui permettent la gestion des ressources dans le contexte.

Les différents systèmes logiques de Lambek ont maints avantages parmi lesquels l'interface syntaxe/sémantique. En effet, le calcul de Lambek entretient un lien étroit avec la sémantique de **Montague** [Gamut91]. Par analogie avec la logique intuitionniste et la correspondance de **Curry-Howard**, chaque dérivation peut être représentée par un λ-terme étant donné qu'à chaque règle de déduction du système correspond une règle de typage du λ-calcul simplement typé. A titre d'exemple l'élimination du '/' coincide avec l'application fonctionnelle et l'introduction du '/' avec l'abstraction :

$$\frac{\Gamma \vdash f : A/B \quad \Delta \vdash y : B}{\Gamma, \Delta \vdash (f\ y) : A}\ /E \qquad\qquad \frac{\Gamma, x : B \vdash f : A}{\Gamma \vdash \lambda x.f : A/B}\ /I$$

Le sens d'une construction grammaticale peut alors être calculé directement à partir de son arbre de dérivation syntaxique et ce en fonction des sémantiques partielles des mots qui la composent : c'est la base du principe de compositionnalité de Frege [Carpenter97, Gamut91].

Malgré son élégance, la logique de Lambek reste inadéquate pour l'analyse des langues naturelles. En effet, les différentes variantes de cette logique souffrent de rigidité extrême (le cas de **NL**) ou de flexibilité exagérée (cas de **NLP** où la commutativité des ressources est globale). La rigidité de tels systèmes permet de limiter leur capacité générative : en effet, il a été prouvé que les grammaires de Lambek sont faiblement équivalentes aux **CFG** [Pentus93] alors que certaines constructions linguistiques -telles les dépendances croisées en Hollandais- sont sensibles au contexte. En revanche une flexibilité irréaliste permet de sombrer dans une sur-génération non désirable. Plusieurs systèmes ont été proposés par la suite pour remédier à ces problèmes en étendant la logique précédente entre autres par Hepple [Hepple90], Morrill [Morrill94] et Moortgat [Moortgat97]. C'est la dernière proposition qui sera l'objet de cet article.

1.3. Logique Multimodale de Moortgat

Afin de limiter les faiblesses du calcul pur de Lambek, **Michael Moortgat** a défini un système logique étendu pouvant comprendre un ensemble de règles structurelles appliquées localement si certaines contraintes sont vérifiées. On peut ainsi étendre la capacité générative du calcul de Lambek

[3]Par convention, les linguistes notent les phrases agrammaticales en les préfixant par *

[4]Extraction d'un élément se trouvant au milieu d'une expression. Par exemple, on ne peut analyser la phrase '*Le savant que j'aime énormément est modeste*' dans la logique **L**

tout en évitant au maximum le problème de sur-génération.
L'extension de Moortgat a été faite en introduisant deux connecteurs unaires ($\Box$, $\Diamond$) et en indexant chaque opérateur logique par un mode de composition, ce qui nous permet d'obtenir une famille de connecteurs logiques ($/_i$, $\backslash_i$, $\bullet_i$, $\Box_j$, $\Diamond_j$) qui cohabitent. C'est ainsi qu'on passe d'une logique monomodale à une logique multimodale.
Les différents modes partagent le même noyau de règles d'inférence de base, la différence réside dans l'application des règles structurelles. En effet, on est capable grâce à ces modes de composition de définir des règles structurelles qui s'appliquent localement en présence de modes spécifiques. A titre d'exemple, on peut remplacer l'associativité globale dans **L** par une associativité restreinte attribuée uniquement au connecteur structurel '$(\ , \ )^a$'.
Les opérateurs unaires peuvent aussi contribuer à restreindre davantage le champ d'application des règles structurelles. Ainsi, on peut par exemple supposer que la commutativité n'a lieu qu'en présence d'une formule décorée par '$\Diamond_c$'.
L'utilisation des opérateurs unaires pour restreindre l'applicabilité des règles structurelles s'inspire fortement de l'utilisation des exponentiels en logique linéaire où la duplication et la contraction des ressources se font seulement en présence de formules décorées par les opérateurs (! , ?) [Girard87].
Outre ce qui a été cité, les opérateurs unaires peuvent aussi servir à coder différents traits tels les traits morphosyntaxiques (genre, nombre etc) [Heylen99]. Une formule indexée par $\Diamond_i$ ou $\Box_i$ représentera une formule distinguée portant le trait **i**. A titre d'exemple, $\Diamond_{sg}$np représente un syntagme nominal singulier alors que la formule $\Diamond_{pl}$np sera associée aux syntagmes nominaux pluriels. Un tel codage de traits nous permet ainsi d'éviter d'engendrer une phrase telle que **(Les enfants dort)* où on est incapable d'unifier le trait pluriel du sujet effectif '*les enfants*' et le trait singulier du sujet attendu par le verbe '*dort*'.

Rappelons que dans le cas de la logique **L**, le contexte n'était autre qu'une liste de formules étant donné que le connecteur structurel '(,)' était globalement associatif. Dans notre cas, le contexte considéré sera plus structuré qu'une simple liste plate. En effet, à chaque famille de connecteurs logiques on associe un connecteur structurel correspondant. Dans le cas du calcul pur de Lambek, notre logique était basée sur une seule famille de connecteurs binaires ($/$, $\backslash$, $\bullet$) à laquelle est associé l'opérateur structurel binaire '(,)'. En logique multimodale, on ajoute un deuxième connecteur structurel unaire '$\langle \ \rangle_i$' associé à la famille d'opérateurs logiques ($\Diamond_i$, $\Box_i$). Les contextes considérés alors auront la forme d'arbres binaires où les noeuds internes coincident avec les opérateurs structurels '$\langle \ \rangle_i$' et '$(\ , \ )^i$'; les feuilles du contexte, quant à elles, seront -suivant notre besoin- soit des formules soit des formules étiquetées par certaines informations (mots, variables, λ-termes...) [Kurtonina95].
On présente dans la figure 1 l'ensemble des règles de la logique multimodale formalisée sous forme de déduction naturelle. C'est ce dernier formalisme qui sera utilisé dans le reste de cet article.

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

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

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

$$\frac{< \Gamma >_j \vdash a : A}{\Gamma \vdash^{\cap} a : \Box_j A}\ \Box_i I \qquad \frac{\Gamma \vdash a : A}{< \Gamma >_j \vdash^{\wedge} a : \Diamond_j A}\ \Diamond_j I \qquad \frac{(\Delta_1\ R\ \Delta_2) \quad \Gamma[\Delta_2] \vdash x : C}{\Gamma[\Delta_1] \vdash x : C}\ Struct_R$$

FIG. 1 – Règles de la logique multimodale

Dans ce cas, les règles de déduction manipulent des jugements de typage de la forme $\Gamma \vdash x$:A, où Γ est un contexte dont les feuilles sont des formules étiquetées par des variables formelles et (x :A) est une formule préfixée par un λ-terme x qui encapsule la sémantique de la dérivation. Les variables libres de x ne sont autres que les étiquettes décorant les feuilles du contexte Γ.
Certaines règles telle $\bullet_i$**E** font intervenir des constructions de la forme $\Gamma[\Delta]$. Dans ce cas, $\Gamma[\ ]$ représente un contexte linéaire [Huet97] qui définit l'emplacement où sera appliquée la règle en question.
Notre système logique assure un lien étroit avec la sémantique comme on peut le remarquer : à chaque règle syntaxique correspond une construction du λ-calcul simplement typé étendu par l'utilisation des opérateurs d'intentionnalité ($^{\wedge}$, $^{\cup}$, $^{\cap}$, $^{\vee}$). L'apport de tels opérateurs utilisés dans la logique intentionnelle sort du cadre de cet article, le lecteur intéressé pourra trouver plus de détails sur ce sujet dans [Gamut91].
Le λ-terme x représentant la sémantique dérivationnelle d'un séquent ($\Gamma \vdash x$:A) pourra être interprété dans le cadre de la sémantique de **Montague** pour calculer la sémantique d'une expression et ce en fonction des sémantiques partielles des mots qui la composent. Ceci se fait en substituant chaque variable libre dans x représentant un mot m_i par la sémantique lexicale attribuée à ce dernier mot par le lexique.

La dernière règle du système multimodal de la figure 1 représente l'application d'une règle structurelle **R** qui réécrit le contexte Δ_1 en Δ_2. Notre grammaire multimodale est paramétrée par l'ensemble des règles structurelles à utiliser. Ainsi, si notre grammaire supporte l'associativité locale en présence du mode **a**, la règle structurelle **L1(a)** ci dessous sera ajoutée aux règles d'inférence de notre système.

$$\frac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3)^a)^a] \vdash x : C}{\Gamma[((\Delta_1,\Delta_2)^a,\Delta_3)^a] \vdash x : C}\ L1\ (a)$$

En se plaçant dans un système multimodal, on est capable de définir des règles structurelles faisant intervenir plusieurs modes à la fois : il s'agit de *postulats d'interaction*. Un exemple de telles règles est L1$\diamond$(i, j) qui est une restriction de la règle d'associativité **L1(i)** (cf. Fig 2).

$$\frac{\Gamma[(\Delta_1,(\Delta_2,<\Delta_3>_j)^i)^i] \vdash x : C}{\Gamma[((\Delta_1,\Delta_2)^i,<\Delta_3>_j)^i] \vdash x : C}\ L1\diamond\ (i,\ j) \quad \frac{\Gamma[((\Delta_1,<\Delta_3>_j)^i,\Delta_2)^i] \vdash x : C}{\Gamma[((\Delta_1,\Delta_2)^i,<\Delta_3>_j)^i] \vdash x : C}\ P1\diamond(i,\ j)$$

FIG. 2 – Quelques règles structurelles

Finalement on définit le langage engendré par une grammaire multimodale $\mathcal{MG}$ comme suit :

$$m_1, m_2, ..., m_n \in \mathcal{L}(\mathcal{MG})\ ssi\ \forall\ i \in \{1..n\}\ \exists\ t_i \in lex(m_i)\ \exists\ \Gamma \in\ Tree(t_1, t_2, ...t_n) \mid \Gamma \vdash s$$

où *Tree*(t_1,...,t_n) représente l'ensemble des contextes ayant comme feuilles t_1,...,t_n et ce dans cet ordre.
Afin de mieux appréhender le fonctionnement de ces grammaires, nous présentons ci dessous un exemple concret d'analyse d'une phrase. La phrase qu'on désire analyser est la suivante : '*Le cœur a des raisons que la raison ignore totalement*'. Linguistiquement, cette phrase présente un phénomène intéressant à savoir celui de l'extraction médiane [Carpenter97]. Ce phénomène est concrétisé par l'extraction de l'objet du verbe *ignore* dont la position n'est pas périphérique[5] étant donné qu'il se trouve entre un verbe (*ignore*) et un adverbe (*totalement*).
La partie cruciale de l'analyse est celle de la subordonnée relative '*que la raison ignore totalement*', c'est cette dernière qu'on décrira dans ce qui suit.
On commence par présenter un petit lexique (cf. Fig 3) qui associe à chaque mot du fragment à

[5]Ni à l'extrême gauche ni à l'extrême droite du fragment à analyser

analyser une formule syntaxique adéquate ainsi qu'une représentation sémantique sous forme de λ-terme.

que	toute	raison	ignore	totalement
$(\mathrm{n}\backslash_a \mathrm{n})/_a(\mathrm{s}/_a\Diamond_c\Box_c \mathrm{np})$	$(\mathrm{s}/_a(\mathrm{np}\backslash_a \mathrm{s}))/_a \mathrm{n}$	n	$(\mathrm{np}\backslash_a \mathrm{s})/_a$ np	$(\mathrm{np}\backslash_a \mathrm{s})\backslash_a(\mathrm{np}\backslash_a \mathrm{s})$
$\lambda P\lambda Q\lambda x.P(^{\cup\wedge}x) \wedge Q(x)$	$\lambda P\lambda Q.\forall\ x,P(x) \Rightarrow Q(x)$	**raison**	**ignore**	**totalement**

FIG. 3 – Lexique

Notre analyse se fera en incluant dans notre système multimodal de base les deux postulats d'interaction L1$\Diamond$(a, c) et P1$\Diamond$(a, c)(cf. Fig 2) où **a** est un mode de composition binaire et **c** un mode unaire.
La figure 4 montre les étapes principales de l'analyse syntaxique. Pour plus de lisibilité on omet le calcul de la sémantique dérivationnelle qui sera donné plus loin.

$$\dfrac{\dfrac{\dots}{(toute,raison)^a \vdash s/_a(np\backslash_a s)}\,/_aE \qquad \dfrac{\dfrac{}{totalement \vdash (np\backslash_a s)\backslash_a(np\backslash_a s)}\,Ax \qquad \dfrac{\dfrac{\dfrac{}{\Box_c np \vdash \Box_c np}\,Ax}{<\Box_c np>_c \vdash np}\,\Box_c E}{(ignore,<\Box_c np>_c)^a \vdash np\backslash_a s}\,/_aE}{((ignore,<\Box_c np>_c)^a,totalement)^a \vdash (np\backslash_a s)}\,\backslash_a E}{((toute,raison)^a,((ignore,<\Box_c np>_c)^a,totalement)^a)^a \vdash s}\,/_aE$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{((toute,raison)^a,((ignore,<\Box_c np>_c)^a,totalement)^a)^a \vdash s}{((toute,raison)^a,((ignore,totalement)^a,<\Box_c np>_c)^a)^a \vdash s}\,P1\Diamond(a,\ c)}{(((toute,raison)^a,(ignore,totalement)^a)^a,<\Box_c np>_c)^a \vdash s}\,L1\Diamond(a,\ c)}{(((toute,raison)^a,(ignore,totalement)^a)^a,(H:\Diamond_c\Box_c np))^a \vdash s}\,\Diamond_c E}{((toute,raison)^a,(ignore,totalement)^a)^a \vdash s/_a\Diamond_c\Box_c np}\,/_aI}{(que,((toute,raison)^a,(ignore,totalement)^a)^a)^a \vdash n\backslash_a n}\,/_aE$$

FIG. 4 – Dérivation syntaxique d'une subordonnée relative

L'étape principale de l'analyse est celle qui permet de placer la formule '$\Diamond_c\Box_c$ np' introduite par la règle '$/_aI$' en position d'objet direct du verbe *ignore*. Ceci se fait en appliquant les postulats cités dans la figure 2 qui assurent la restructuration du contexte. L'application de ces postulats s'avère possible étant donné que la formule représentant l'objet extrait est décorée par l'opérateur de contrôle $\Diamond_c$ qui se transforme en $<>_c$ grâce à la règle de base $\Diamond_c$E.
L'intérêt majeur des grammaires catégorielles est l'interface syntaxe/sémantique. En se basant sur cette interface aisée (cf. Fig 1) on déduit de l'arbre d'analyse de la figure 4 que la sémantique dérivationnelle de cette clause relative est la suivante :
(que (λx. (toute raison) ((totalement (ignore x))))).
L'interprétation de ce λ-terme obtenue en substituant chaque variable libre par sa sémantique lexicale permet d'obtenir la sémantique suivante sous forme normale :
λQ λx. Q(x) $\wedge$ $\forall$ y,**raison**(y) $\Rightarrow$ ((**totalement** (**ignore** (x)))(y).

Nous remarquons que la logique multimodale partage plusieurs points communs avec la logique intuitionniste notamment la notion de types, la correspondance de Curry-Howard etc. Toutefois, ces deux logiques sont aussi très différentes dans la mesure où la première est beaucoup plus ardue à appréhender étant donné son grand degré de paramétrisation. Faciliter l'étude et la compréhension de ces logiques complexes est l'une de nos motivations pour la réalisation de l'atelier $\mathcal{ICHARATE}$.

2. Atelier $\mathcal{ICHARATE}$

L'atelier $\mathcal{ICHARATE}$ est un projet en cours de réalisation qui étend et enrichit notre contribution antérieure dédiée à la formalisation en **Coq** du calcul de Lambek [AnCas04]. Notre objectif est de créer une bibliothèque **Coq** pour la logique multimodale qui permettra de faciliter la compréhension de ces dernières aux différents utilisateurs (linguistes, chercheurs ...). Notre bibliothèque actuelle est composée d'un noyau qui comprend la formalisation de trois présentations de la logique multimodale à savoir la présentation axiomatique, le calcul des séquents et la déduction naturelle qui possédent différentes règles d'inférence mais engendrent le même langage [Moortgat97]. Elle contient aussi un ensemble de règles dérivées et théorèmes génériques, un module de calcul de la sémantique ainsi qu'une banque de tactiques spécialisées dans le domaine considéré.
Dans ce qui suit, on décrira les composantes de base de cet atelier. Pour plus de lisibilité, les différentes définitions et les divers théorèmes seront présentés en utilisant les notations mathématiques classiques. Toutefois, le lecteur intéressé par la formalisation en **Coq** trouvera les sources de l'atelier à l'adresse suivante :*www.labri.fr/˜anoun/Recherche/Icharate.tar.gz.*

2.1. Représentation des Grammaires Multimodales en Coq

Le Calcul des Constructions Inductives, qui est l'assise du système **Coq**, nous permet de représenter par des types inductifs *paramétrés* aussi bien les formules, les jugements et les règles structurelles que les dérivations et le λ-calcul simplement typé. Le système **Coq** autorise la paramétrisation de manière élégante grâce à son mécanisme de sections [CoqManual, CoqArt04].
A titre d'exemple, le type *natded* associé aux déductions naturelles est un type inductif *dépendant*, il est déclaré de sorte **Set** car il représente une structure de donnée. Chaque constructeur de *natded* est associé à une règle de dérivation. La figure 5 en donne un extrait.

```
Section natDed.
......
Variables I J At W:Set.    (* modes, atomes et mots *)
Variables decI :∀ (a b:I),  {a=b}+{a≠ b}
          decJ :∀ (a b:J),  {a=b}+{a≠ b}.  (* décidabilité de l'égalité *)
Variable R: extension I J.   (* ensemble de règles structurelles *)

Inductive natded: (context I J At W) →  (Form I J At) →  Set :=
 | wd : ∀ r A,  natded (r : A) A
 | slashI : ∀ Γ  A B n i, natded (Γ , (hyp_n: B))^i  A →
                          natded  Γ A /_i B
 .....
 | struct : ∀ Γ_z Δ_1 Δ_2 C ru,  ru ∈ R →
                          apply_rule ru decI decJ Δ_1 = Some Δ_2 →
                          natded (zfill Γ_z Δ_2) C →
                          natded (zfill Γ_z Δ_1) C .
```

FIG. 5 – Quelques contructeurs du type *natded*

Nous utilisons les *zippers* de Gérard Huet [Huet97] comme types de données temporaires pour formaliser les contextes linéaires $\Gamma[\ ]$ sous forme d'arbres binaires munis d'un pointeur sur l'un de leurs sous-arbres. Par exemple, la formalisation de la règle *struct* (cf. Fig 5) utilise le contexte linéaire Γ_z dont le 'trou' est rempli par un contexte donné grâce à la fonction *zfill.*

Les règles structurelles, quant à elles, sont définies de manière abstraite comme paires de contextes symboliques dont les feuilles sont des variables. Une fonction *apply_ rule* permet de calculer à partir de chaque règle structurelle symbolique une règle concrète formalisée sous la forme d'une fonction partielle sur l'ensemble des contextes. Une telle représentation est salutaire étant donné qu'elle facilite les preuves de différentes propriétés portant sur les règles de réécriture (linéarité, linéarité à gauche).

2.2. Règles dérivées et théorèmes

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 de méta-théorèmes, qui permettent d'étudier les propriétés globales du formalisme considéré.
Dans le cas des grammaires multimodales, ce point est particulièrement important, étant donné le fort degré de paramétrisation des systèmes déductifs sous-jacents.
L'accès aisé à un tel ensemble de règles ainsi que la facilité d'en dériver de nouvelles s'avère fructueux 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 particuliers à telle ou telle classe de grammaires.

2.2.1. Règles dérivées

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

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

où $\overset{R}{\vdash}$ représente l'opérateur de déduction dans une grammaire multimodale supportant l'ensemble de règles structurelles **R**, et **cond(R)** permet de définir des contraintes devant être vérifiées par **R**.
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 [Alvarado02] ...etc. Afin d'alléger l'écriture de ces dernières, les quantifications universelles seront omises.

Règle de coupure :
La règle de coupure présentée ci dessous ne fait pas partie des règles de base de la déduction naturelle (cf. Fig 1) : c'est une règle dérivée [Moortgat97].

$$\frac{(weak\_Sahlqvist\ R)\quad \Delta \vdash A\quad \Gamma[A] \vdash C}{\Gamma[\Delta] \vdash C}\ cut$$

La classe *weak_ Sahlqvist* de règles structurelles dites de forme faible de **Sahlqvist** (règles linéaires à gauche)[Kurtonina95] est celle que nous considérons dans notre atelier. La démonstration de cette règle de coupure se fait en prouvant une règle généralisée qui prend en compte les remplacements multiples, la preuve de cette dernière s'effectue par récurrence sur les dérivations [Essli04].
Intuitivement, la règle de coupure représente l'utilisation de lemmes intermédiaires au cours d'un raisonnement. Elle est très utile et peut faciliter énormément la preuve d'autres règles dérivées. A titre d'exemple, la recherche d'une démonstration directe (qui utilise uniquement les règles de base) pour les deux règles d'isotonicité ci dessous s'avère ardue et fastidieuse.

$$\frac{A \vdash B}{A/_iC \vdash B/_iC}\ /_iIso \qquad \frac{A \vdash B}{C\backslash_i A \vdash C\backslash_i B}\ \backslash_iIso$$

En revanche, l'utilisation de la règle de coupure permet de construire, pour chacune de ces règles, une preuve simple et courte et ce assez aisément (cf. Fig.6).

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

FIG. 6 – Preuve de $/_i Iso$ en utilisant *cut*

Extraction médiane :

Nous présentons dans ce qui suit deux règles dérivées utilisées pour l'analyse du phénomène de l'extraction médiane. Ces dernières sont un cas typique de règles permettant la compréhension des principes d'interaction.
La première règle constitue un cas particulier de la règle de composition mixte (X/Y , X\Z ⊢ Z/Y) dérivable dans le système **LP** (où l'associativité et la commutativité sont globales). Elle concerne uniquement la classe de grammaires multimodales supportant la règle structurelle **P1◇(i, j)**.

$$\frac{P1\Diamond(i,\ j) \in R}{(X/_iY, X\backslash_i Z)^i \overset{R}{\vdash} Z/_i\Diamond_j\Box_j Y}\ Extract1$$

Une seconde règle *Extract2* peut être déduite de la première et ce pour les grammaires supportant à la fois les deux règles structurelles **P1◇(i, j)** et **L1◇(i, j)** (cf. Fig.2).

$$\frac{P1\Diamond(i,\ j) \in R \quad L1\Diamond(i,\ j) \in R}{(Z_1, (X/_iY, X\backslash_i(Z_1\backslash_i Z_2))^i)^i \overset{R}{\vdash} Z_2/_i\Diamond_j\Box_j Y}\ Extract2$$

On peut utiliser ces règles pour effectuer une dérivation simplifiée de '*que Houda ignore totalement*' ⊢ $n\backslash_a n$ comme ceci[6] :

$$\cfrac{\cfrac{}{que \vdash (n\backslash_a n)/_a(s/_a\Diamond_c\Box_c np)}\ Ax \qquad \cfrac{}{(Houda, (ignore, totalement)^a)^a \vdash (s/_a\Diamond_c\Box_c np)}\ Extract2}{(que, (Houda, (ignore, totalement)^a)^a)^a \vdash n\backslash_a n}\ /_aE$$

FIG. 7 – Application de la règle dérivée Extract2

Dépendances non bornées :

Nous présentons dans cette section un exemple de règles dont l'application provoque une réorganisation complexe du contexte : il s'agit d'une règle permettant de traiter les dépendances non bornées. Ce phénomène linguistique a lieu quand deux éléments arbitrairement éloignés d'une phrase sont sémantiquement reliés. Ceci est illustré à titre d'exemple par la phrase suivante '*The book which every man thinks Sandra hopes Mary wrote _ is wonderful*' où '_' représente l'emplacement de l'objet extrait qui est très éloigné du pronom relatif '*which*' auquel il est lié. Après l'introduction de l'élément extrait, ce dernier se place 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

[6] On utilise le lexique 3 et on attribue au nom propre '*Houda*' la catégorie **np**

extrait comme frère droit du dernier sous-contexte dominé par le connecteur structurel $(\ ,\ )^i$ et se trouvant dans la branche la plus à droite du contexte initial.

$$\frac{L1\Diamond(i,\ j) \in R \quad \rho(\Gamma,\ < A >_j,\ i) \overset{R}{\vdash} C}{(\Gamma, \Diamond_j A)^i \overset{R}{\vdash} C}\ Dep$$

Cette règle utilise la fonction auxiliaire ρ définie récursivement par :

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

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 $(\Gamma, \Diamond_j A)^i$ et ce par la règle de réécriture $L1\Diamond(i,\ j)$.
On peut appliquer cette règle pour effectuer une dérivation de '*which every man thinks Sandra hopes Mary wrote _*'. L'élément extrait sera placé automatiquement à son endroit (objet du verbe *wrote*).

$$\cfrac{\cfrac{\cfrac{....}{((every\ ,\ man)\ ,\ (thinks\ ,\ (Sandra,\ (hopes,\ (Mary,\ (wrote\ ,\ (\_ :< \Box_c np >_c))^a)^a)^a)^a)^a}}{(((every\ ,\ man)\ ,\ (thinks\ ,\ (Sandra,\ (hopes,\ (Mary,\ wrote)^a)^a)^a)^a)^a, (\_ : \Diamond_c\Box_c np))^a \vdash s}\ Dep}{((every\ ,\ man)\ ,\ (thinks\ ,\ (Sandra,\ (hopes,\ (Mary,\ wrote)^a)^a)^a)^a)^a \vdash s/_a\Diamond_c\Box_c np}\ /_aI$$

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 section (1.3) placera l'objet extrait comme frère droit de l'adverbe *totalement*, cette position bloquera le déroulement de l'analyse.

2.2.2. Critères de non dérivabilité

Il est intéressant d'avoir des lemmes permettant de prouver formellement qu'un séquent ($\Gamma \vdash$ C) n'est pas dérivable dans une logique multimodale donnée. En effet, la recherche de démonstration en logique multimodale est d'une grande complexité [Moot02], aussi est-il généralement difficile pour un analyseur syntaxique automatique de décider la non démontrabilité d'un séquent. C'est ici qu'intervient la puissance du raisonnement par rapport au calcul : il suffit de trouver une condition nécessaire vérifiée par tous les séquents démontrables, une fois que cette condition n'est pas respectée par un séquent particulier on déduit automatiquement que ce dernier n'est pas dérivable.
On présentera dans ce qui suit un 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 :

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

Pour un contexte donné Γ, $\psi_j(\Gamma)$ est un entier relatif représentant la différence entre le nombre d'occurrences[7] des opérateurs ($\diamond_j$, $\langle\ \rangle_j$) et celui de l'opérateur $\Box_j$.
Soit *linear*$\diamond_j$ la classe de règles structurelles linéaires qui, en outre, conservent la multiplicité (le nombre d'occurrences) de l'opérateur structurel $\langle\ \rangle_j$. Les règles de la figure 2 font partie de cette classe alors que la règle K(i,j) ci dessous ne l'est pas car elle augmente le nombre d'occurences de $\langle\ \rangle_j$ après la réécriture.

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

Par simple récurrence sur la dérivation de ($\Gamma \vdash$ C), on démontre sans difficulté le théorème suivant :

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

Un corollaire du théorème précédent est le suivant :

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

Ce corollaire est utilisé dans la définition d'une tactique de réfutation permettant de prouver, dans certains cas, la non dérivabilité d'un séquent et ce en ayant recours à un calcul simple qui se fait automatiquement par le système. Cette tactique est basée sur la technique de *réflexion* [Alvarado02] qui est très utilisée en **Coq** car elle permet de faire des raisonnements compliqués par calcul.
Appliquons maintenant ce critère de non dérivabilité sur un exemple concret. Considérons la phrase agrammaticale **(Houda dorment)*. On attribue au nom propre *Houda* la catégorie $\diamond_{sg}\mathbf{np}$ et au verbe intransitif *dorment* la formule syntaxique $\diamond_{pl}\mathbf{np}\backslash_0\mathbf{s}$. Nous allons prouver formellement qu'il est impossible d'effectuer une dérivation de **(Houda dorment)* dans la grammaire multimodale **NL**$\diamond$ qui ne supporte aucune règle structurelle, ceci revient à démontrer que :

$$\forall\, \Gamma,\ \ \Gamma \in Tree(lex(Houda),\ lex(dorment))\ \Rightarrow\ \Gamma \nvdash s$$

A priori, le contexte Γ est inconnu, on connait juste les feuilles qui le composent, il est donc impossible d'appliquer directement le critère précédent. Pour arriver à notre but, on passe par un lemme intermédiaire :

$$\forall\, \Gamma\ t_1\ t_2 ... t_n\ j,\quad \Gamma \in Tree(t_1, t_2, ..., t_n) \Rightarrow\ \ \psi_j(\Gamma) \geq \sum_{i=1}^{n} \psi_j(t_i)$$

Soit Γ un contexte structuré associé à la phrase précédente. Par un simple calcul on retrouve $\psi_{sg}(\Gamma) \geq 1$ alors que ψ_{sg}(s)=0 , le critère de non démontrabilité permet ainsi de conclure.
L'atelier comprend aussi d'autres critères de non dérivabilité tel le théorème de polarité [Essli04].

2.2.3. Méta-théorèmes

D'autres théorèmes sont prouvés dans $\mathcal{ICHARATE}$ tels l'interopérabilité entre les trois formalismes de la logique multimodale [Moortgat97]. Nous avons démontré l'équivalence entre les trois présentations multimodales en ayant recours à des fonctions certifiées qui permettent le passage d'un formalisme source à un formalisme cible supportant les règles structurelles du premier. Cette intéropérabilité 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 les deux autres formalismes sans avoir à les redémontrer. A titre d'exemple, nous avons prouvé la cohérence et la complétude de la présentation axiomatique multimodale en utilisant les modèles de **Kripke** [Kurtonina95], grâce à l'intéropérabilité nous avons pu déduire, sans aucune peine, la preuve de cohérence et de complétude pour les autres systèmes déductifs multimodaux.

[7] Une occurence peut être positive ou négative.

2.3. Calcul de la Sémantique de Montague

L'atelier $\mathcal{ICHARATE}$ prend en charge le calcul de la sémantique de **Montague**. Un module de λ-calcul simplement typé avec opérateurs intentionnels a été formalisé en **Coq** et ce en s'inspirant de la bibliothèque de Bruno Barras [Barras97]. Notre λ-calcul est polymorphe, il est paramétré par l'ensemble des constantes considérées. Ceci s'avère utile pour notre étude étant donné qu'on considére deux λ-calculs distincts : le premier pour représenter la sémantique dérivationnelle (cf. Fig1) et le second pour calculer la sémantique de **Montague**. Les types utilisés dans le premier λ-calcul ne sont autres que les types syntaxiques. En revanche, le second λ-calcul est basé sur des types sémantiques définis par **Montague** [Gamut91] à savoir deux types de base **e** (ensemble des individus) et **t** (valeurs de vérité) et des types composés à partir de ces derniers en utilisant les deux constructeurs '$\rightarrow$' (application fonctionnelle) et '$\times$'(produit cartésien).
Le calcul de la sémantique d'une expression (m_1 m_2 ...m_n) à partir d'une dérivation syntaxique et d'un lexique *lex* se déroule en trois étapes dont voici une brève description :

1. A partir d'une dérivation d'un jugement $\Gamma\vdash$ x :A associé à l'expression précédente, on extrait la sémantique *dérivationnelle* x qui est représentée sous forme d'un λ-terme simplement typé. Le contexte Γ comprend n feuilles de la forme (x_i : t_i) où x_i est une variable formelle et t_i un type syntaxique associé au mot m_i par le lexique *lex*. Ainsi, les variables libres de **x** ne sont autres que x_1, x_2..., x_n.
2. Le calcul de la sémantique de **Montague** se fait en substituant chaque variable libre x_i dans **x** par la sémantique lexicale du mot m_i. On obtient dans ce cas un nouveau λ-terme simplement typé dont les variables libres ne sont autres que les constantes sémantiques considérées [8].
3. Le λ-terme obtenu dans l'étape (2) est généralement très complexe, on passe donc par une phase de normalisation qui permet de réduire ce terme sous sa forme normale. Notre λ-calcul étant confluent et convergent, nous avons défini dans le langage **Ltac** [Delahaye01] une tactique récursive élaborée permettant de calculer automatiquement la forme normale d'un λ-terme donné et ce en appliquant une suite finie de réécritures.

2.4. Tactiques Spécialisées

$\mathcal{ICHARATE}$ est muni d'une banque de tactiques spécialisées définies dans le langage **Ltac** qui est fourni avec le système **Coq** [Delahaye01]. 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 donnée. 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 hypothètiques dans le contexte lors de l'application de règles basées sur le raisonnement hypothètique (e.g. $/_i$I, $\backslash_i$I ...). A la différence de la logique intuitionniste, les ressources introduites sont placées à un endroit bien précis du contexte (e.g. 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 *struct* (cf. Fig.1), dont les variables quantifiées universellement n'apparaissent pas toutes dans les types

[8] A titre d'exemple, les constantes sémantiques sont représentées en gras dans la figure 3

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 linéraire de la forme $\Gamma[\,]$. Ces tactiques sont paramétrées 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 (zfill Γ_z Δ_1)) et ce en fonction du chemin passé en paramètre.

Outre les tactiques qui appliquent uniquement les règles de base, nous avons défini un ensemble de tactiques sophistiquées qui utilisent des règles dérivées 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, e.g. la vérification d'une propriété portant sur l'ensemble des règles structurelles (linéarité à gauche, linéarité, ...) 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*.

3. Conclusion & Perspectives

Nous avons présenté dans cet article l'état actuel de l'atelier logique $\mathcal{ICHARATE}$. Notre objectif n'est pas de réaliser un outil automatique d'analyse pour les langues naturelles car de tels outils existent déjà [Moot02, Eijck03, Carpenter97]. Notre but est plutôt de mettre à la disposition des utilisateurs un atelier qui leur permet d'étudier et d'appréhender des formalismes logiques complexes telles les grammaires catégorielles multimodales. Grâce au Calcul des Constructions Inductives, qui est la base de **Coq**, nous avons pu définir un catalogue important de règles dérivées génériques et réutilisables qui permettent de rendre compte de divers phénomènes linguistiques (dépendances croisées, extraction médiane ...). En plus, notre atelier offre aux utilisateurs la possibilité de prouver, assez aisément, de nouvelles règles et propriétés ou d'effectuer interactivement des dérivations syntaxiques pour des grammaires données en ayant recours à la banque de tactiques implémentées pour ce besoin.

Toutefois, beaucoup de travail reste à faire, aussi bien sur le fond que sur la forme, avant d'avoir un outil autonome et complet.

Sur le fond, nous envisageons d'étendre la sémantique de **Montague** qui s'avère très limitée [9] étant donné qu'elle est basée sur le λ-calcul simplement typé. Nous comptons donc explorer la possibilité d'intégrer des types plus complexes (inductifs ou avec produits dépendants) dans cette sémantique et profiter ainsi de la richesse des types de **Coq**. D'autre part, nous nous intéresserons à la définition de méta-théorèmes pertinents portant sur la sémantique. Dans une autre direction, nous envisageons d'enrichir notre atelier en ajoutant des formalismes logiques récents dédiés au traitement des langues naturelles [Lecomte04, Muskens03]. Le but principal de cette piste est de pouvoir tester ces nouveaux modèles et étudier la possibilité d'interopérabilité entre ces différents systèmes logiques.

Du point de vue de la forme, nous comptons munir notre atelier d'une interface conviviale qui facilitera l'utilisation de l'atelier pour les non spécialistes de **Coq**. Ceci se fera en utilisant **PCoq** [Amerkad, Bertot01] qui permet d'afficher les formules mathématiques avec des notations lisibles et qui, en outre, offre des possibilités d'interaction intéressantes dont la technique de *proof-by-pointing* [Bertot94].

Remerciements

Je tiens à remercier mes deux encadrants Pierre Castéran et Paul Gloess qui m'ont soutenu dans la rédaction de cet article grâce à leurs remarques fructueuses.

[9] Parmi ses limitations, citons son incapacité de résolution des anaphores

Références

[Alvarado02] Cuihtlauac Alvarado *Réflexion pour la réécriture dans le calcul des constructions inductives* Thèse de doctorat, Université de Paris Sud, Orsay 2002.

[Amerkad, Bertot01] Ahmed Amerkad, Yves Bertot, Loic Pottier, Laurence Rideau *Mathematics and Proof Presentation in Pcoq* Rapport de recherche INRIA, 2001.

[AnCas04] Houda Anoun, Pierre Castéran *Calcul de Lambek* Bibliothèque Coq pour le calcul de Lambek, Contribution Coq, 2004.

[Barras97] Bruno Barras *A formalisation of the Calculus of Construction* Contribution Coq, coq.inria.fr/contribs/coq-in-coq.html, 1997.

[Bertot94] Yves Bertot, Gilles Kahn, Laurent Théry *Proof by Pointing* Symposium on Theoretical Aspects of Computer Software, Japon, 1994.

[Carpenter97] Bob Carpenter. *Type Logical Semantics.* Massachusetts Institute of Technology, 1997.

[Chomsky95] Noam Chomsky *The Minimalist Program* MIT Press, 1995.

[CoqArt04] Yves Bertot et Pierre Castéran *Interactive Theorem Proving and Program Development : Coq'Art : The Calculus of Inductive Constructions* Springer Verlag, 2004.

[CoqManual] The Coq Development Team *The Coq Proof Assistant, Reference Manual, Version 8.0.*

[Delahaye01] David Delahaye *Conception de langages pour décrire les preuves et les automatisations dans les outils d'aide à la preuve* Thèse de doctorat, Université Paris 6, 2001.

[Eijck03] Jan van Eijck *Computational Semantics and Type Theory* http ://homepages.cwi.nl/~jve/cs

[Essli04] Houda Anoun, Pierre Castéran, Richard Moot *Proof Automation for Type-Logical Grammars* Notes pour le cours d'ESSLI, Nancy 2004 http ://esslli2004.loria.fr.

[Gamut91] GAMUT, L.T.F *Language and Meaning, Vol 2 : Intentional Logic and Logical Grammar* University of Chicago Press, 1991.

[Girard87] Jean-Yves Girard *Linear Logic* Theoretical Computer Science, 1987.

[Hepple90] Mark Hepple *The Grammar and Processing of Order and Dependency : a Categorial Approach* PhD thesis, Edinburgh, 1990.

[Heylen99] Dirk Heylen *Types and Sorts Resource Logic for Feature Checking* PhD thesis, Utrecht,1999.

[Huet97] Gerard Huet *The Zipper* Journal Of Functional Programming, 7(5) :549-554, 1997.

[Joshi75] Aravind Joshi, Leon Levy, Masako Takahashi *Tree adjunct grammar* Journal of Computer and System Sciences, vol.21, no.2, 1975.

[Kurtonina95] Natasha Kurtonina *Frames and Labels* PhD Thesis, Utrecht, 1995.

[Lambek58] Joachim Lambek *The Mathematics of Sentence Structure* American Mathematical Monthly, 1958.

[Lecomte04] Alain Lecomte *Derivations as Proofs a Logical Approach to Minimalism* Grammaires catégorielles, 2004.

[Moortgat97] Michael Moortgat *Categorial Type Logic* Chapter 2 in Van Benthem & ter Meulen (eds) Handbook of Logic and Language, Elsevier 1997.

[Moortgat99] Michael Moortgat *Labelled Deduction in the Composition of Form and Meaning* in U.Reyle and H.J.Ohlbach eds, Logic Language and Reasoning.Kluwer, 1999.

[Moot02] Richard Moot *Proof Nets for Linguistic Analysis* PhD Thesis, Universiteit Utrecht, 2002.

[Morrill94] Glyn Morrill *Type Logical Grammar Categorial Logic of Signs* Kluwer, Dordrecht, 1994.

[Muskens03] Reinhard Muskens *Language, Lambdas, and Logic* Resource Sensitivity in Binding and Anaphora, Studies in Linguistics and Philosophy, Kluwer, 2003

[Pentus93] Mati Pentus *Lambek Grammars are context free* Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science,Montreal, 1993.