Intégration de la sémantique dans les Grammaires d'Interaction

Hassen Ben Zineb

Intégration de la sémantique dans les Grammaires d'Interaction

MÉMOIRE

présenté et soutenu le 25 Juin 2007

pour l'obtention du

Master en Informatique de l'Université Henri Poincaré – Nancy 1

(Spécialité Traitement Automatique des Langues)

par

Hassen BEN ZINEB

Composition du jury

Guy PERRIER	Professeur, Nancy 2
Dominique MÉRY	Professeur, Nancy 1
Claude GODART	Professeur, Nancy 1
Noëlle CARBONELL	Professeur, Nancy 1
Didier GALMICHE	Professeur, Nancy 1

Encadrant : Bruno GUILLAUME — Chargé de Recherche, Inria

Résumé

Le présent travail est une étude qui s'inscrit dans le cadre d'un stage de recherche pour l'obtention du diplôme Master 2 de recherche en informatique pour le traitement automatique des Langues. L'étude s'intitule *"Intégration de la sémantique dans les grammaires d'interaction"*. Il s'agit d'étudier l'interface syntaxe/sémantique dans les formalismes grammaticaux et de s'inspirer de cette étude pour proposer une solution qui introduit la sémantique dans le système syntaxique des grammaires d'interaction. Cette solution est implémentée dans un logiciel d'analyse syntaxique baptisé LEOPAR.

Mots-clés: grammaires d'interaction, interface syntaxe/sémantique, représentation sémantique, λ-calcul.

Abstract

This work is done in the context of the Masters in Computer Science, specialization in "Natural Language Processing (NLP)". It concerns the *"Integration of semantics in Interaction Grammars (IG)"* and studies how the syntax/semantics interface used for other formalisms can be adapted to IG. The solution proposed is implemented in a parser called LEOPAR.

Keywords: Interaction Grammars, syntax/semantics interface, semantic representation, λ-calculus

Remerciements

C’est avec un grand plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance.

Je tiens à remercier vivement le tuteur de mon stage Monsieur Bruno GUILLAUME pour sa disponibilité, son soutien et ses formations qu’il m’a fourni et de m’avoir dirigé et aidé dans mon travail.

Je voudrais également remercier Monsieur Guy PERRIER le responsable du Master TAL, ainsi que tous les membres de l’équipe CALLIGRAMME pour m’avoir préparé l’environnement matériel et logiciel de travail et pour leur coopération.

J’adresse aussi une attention toute particulière à Madame Nadine BEURNÉ pour sa disponibilité à la résolution de mes problèmes administratifs.

Par ailleurs, je remercie tous les enseignants, les personnels et tous mes collègues en Master TAL.

Enfin, j’aimerais saisir cette occasion pour remercier les membres du jury tout en espérant qu’ils y trouvent les qualités de clarté et de motivation qu’ils attendent.

Je dédis ce modeste travail
À mes parents,
À mes frères et sœurs,
À mes professeurs et tuteurs,
À mes amis et tous ceux que j'aime

Table des matières

Table des figures

Introduction

Le traitement automatique des langues (TAL) a pour objectif de traiter des énoncés exprimés dans une langue dite "naturelle". Pour pouvoir traiter automatiquement ces données, il faut être capable d'expliciter les règles de la langue, de les représenter dans des formalismes opératoires et calculables, et les implémenter à l'aide de programmes. L'élaboration de systèmes opérationnels nécessite de la recherche fondamentale, notamment en matière d'analyse et de génération de texte. L'analyse d'un texte se base sur son analyse syntaxique et sémantique.

Dans ce cadre, les grammaires d'interaction ont été conçues pour modéliser à la fois la syntaxe et la sémantique des langues naturelles. Elles s'inspirent de deux formalismes différents de grammaires formelles : les grammaires catégorielles et les grammaires d'arbres adjoints. Des grammaires catégorielles, les grammaires d'interaction reprennent l'idée que les syntagmes sont des ressources consommables et qu'il y a une dualité entre celles-ci qui s'exprime dans le principe de composition. Dans un même temps, les grammaires d'interaction introduisent un assouplissement considérable dans le formalisme en ayant recours à la notion de description d'arbres.

L'originalité du formalisme de grammaires d'interaction est d'utiliser à la fois des données sous-spécifiées pour décrire les structures syntaxiques et sémantiques des mots d'une phrase et un système de polarités qui permet de contraindre l'analyse. Les objets syntaxiques de base sont des descriptions d'arbres polarisées qui spécifient partiellement des arbres syntaxiques.

Les grammaires d'interaction sont implémentées dans un logiciel intitulé LEOPAR qui est développé par l'équipe CALLIGRAMME. Ce logiciel permet une analyse syntaxique détaillée mais l'aspect sémantique n'est pas encore implémenté.

Pour l'ajout de la sémantique dans les grammaires d'interaction, il faut passer par deux étapes importantes : le choix d'un formalisme pour la représentation sémantique et la description de l'interface syntaxe/sémantique.

Une proposition donnée par G. Perrier, dans [Per05], consiste à intégrer la sémantique dans le formalisme de grammaires d'interaction à travers un niveau supplémentaire dont les structures sont des graphes acycliques orientés et non des arbres localement ordonnés et une fonction simple qui fait le lien entre les deux niveaux parallèles (syntaxique et sémantique). Cette façon de procéder a l'avantage de proposer beaucoup de souplesse dans l'articulation entre les deux niveaux. Cependant son implémentation demande beaucoup de développement.

Une autre possibilité pour avoir une bonne interface syntaxe/sémantique est d'utiliser le résultat de l'analyse syntaxique pour produire dans un second temps l'analyse sémantique. Cette idée est utilisée dans [GK03] avec une représentation en "Hole Semantics" et dans [Gar06b] avec trois représentations

sémantiques différentes. Ici, il s'agit de s'inspirer des ces études pour les appliquer aux grammaires d'interaction.

Ce rapport détaille une solution suivant ce plan. Dans un premier chapitre, une étude de l'état de l'art présente les grammaires d'interaction et les études de la sémantique dans des autres formalismes. Une solution adoptée est détaillée théoriquement dans un deuxième chapitre. Quelques détails pour l'implémentation de cette solution sont donnés dans le troisième chapitre ainsi que quelques résultats obtenus par LEOPAR. Enfin, le présent rapport se termine par une conclusion générale.

Chapitre 1

Etat de l'art

1.1 Introduction

Ce chapitre décrit, en premier lieu, les grammaires d'interaction en passant par leurs originalités et leurs performances en analyse. En second lieu, une description de la sémantique dans les autres formalismes grammaticaux sera présentée par deux études différentes.

1.2 Les Grammaires d'Interaction (GI)

Le formalisme des grammaires d'interaction utilise un système de contraintes pour décrire la bonne formation d'une phrase. Ces contraintes reposent sur deux notions clés : celle de description d'arbres et celle de polarité.

1.2.1 Polarisation

1.2.1.1 Traits polarisés

Un trait est un champ de données caractérisé par un nom et associé à une valeur. Pour un trait polarisé, en plus d'une valeur, on associe une polarité, pour indiquer éventuellement si c'est une ressource consommable ou un besoin. Une polarité peut être positive (notée $\rightarrow$), négative (notée $\leftarrow$) ou neutre. Il est nécessaire dans la pratique de distinguer les traits qui sont initialement neutres de ceux qui sont issus d'une neutralisation d'un trait positif et d'un négatif. Dans le premier cas, on note la polarité $=$ et dans le deuxième, on la note $\leftrightarrow$. De plus, pour définir les opérations de façon plus uniforme, on ajoute une cinquième polarité$\perp$ qui correspond à l'échec de l'unification des polarités. Le résultat de l'unification de deux polarités p et q se note $p+q$ et est donné par le tableau suivant :

p \ q	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$	$\perp$
$\leftarrow$	$\perp$	$\leftrightarrow$	$\leftarrow$	$\perp$	$\perp$
$\rightarrow$	$\leftrightarrow$	$\perp$	$\rightarrow$	$\perp$	$\perp$
$=$	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$	$\perp$
$\leftrightarrow$	$\perp$	$\perp$	$\leftrightarrow$	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

Les traits négatifs représentent des ressources attendues, les traits positifs des ressources disponibles et les traits neutres des propriétés morpho-syntaxiques qui ne se comportent pas comme des ressources consommables.

Exemples :
- trait négatif : $\langle cat \leftarrow np \rangle$,
- trait neutre : $\langle cat \leftrightarrow np \rangle$,
- trait positif : $\langle funct \rightarrow obj \rangle$.

1.2.1.2 Structures de traits polarisés

Une structure de traits polarisés est une fonction partielle qui associe à un nom de trait t un couple formé d'une polarité et d'une valeur qui est soit une valeur atomique soit une disjonction de valeursatomiques.

Exemple : la structure de traits du mot "*Jean*" est de la forme suivante

$$\begin{bmatrix} pers & = & 2|3 \\ num & = & sg \\ gen & = & m \\ funct & \leftarrow & ? \\ cat & \rightarrow & np \end{bmatrix}$$

- le signe " $=$ " dans la polarité d'un trait indique que ce dernier est neutre par définition,
- le signe "?" dans la valeur d'un trait indique que cette valeur peut être n'importe quel élément appartenant à l'ensemble des valeurs de ce trait. Dans le cas de cet exemple, cela veut dire que la fonction du mot *Jean* dans une phrase peut être quelconque (subj, obj, ...),
- le trait *cat* est un trait positif qui donne la valeur np.

Unification de deux structures de traits : soit deux structures de traits S et T :

$$S = \begin{bmatrix} f_1 & p_1 & x_1 \\ f_2 & p_2 & x_2 \\ \vdots & \vdots & \vdots \\ f_n & p_n & x_n \\ f'_1 & p'_1 & x'_1 \\ \vdots & \vdots & \vdots \\ f'_k & p'_k & x'_k \end{bmatrix} \quad T = \begin{bmatrix} f_1 & q_1 & y_1 \\ f_2 & q_2 & y_2 \\ \vdots & \vdots & \vdots \\ f_n & q_n & y_n \\ f''_1 & q''_1 & y''_1 \\ \vdots & \vdots & \vdots \\ f''_l & q''_l & y''_l \end{bmatrix}$$

telles que les traits $(f'_i)_{1 \leq i \leq k}$ ne sont pas dans T et les traits $(f''_i)_{1 \leq i \leq l}$ ne sont pas dans S.

Le résultat d'unification de S et T, noté $S + T$, est défini par :

$$S+T=\begin{bmatrix} f_1 & p_1+q_1 & z_1 \\ f_2 & p_2+q_2 & z_2 \\ \vdots & \vdots & \vdots \\ f_n & p_n+q_n & z_n \\ f'_1 & p'_1 & x'_1 \\ \vdots & \vdots & \vdots \\ f'_k & p'_k & x'_k \\ f''_1 & q''_1 & y''_1 \\ \vdots & \vdots & \vdots \\ f''_l & q''_l & y''_l \end{bmatrix}$$

où les valeurs $(z_i)_{1\leq i\leq n}$ sont définies par l'intersection des ensembles de valeurs atomiques x_i et y_i. Les deux structures de traits S et T sont unifiables que si tous leurs traits sont unifiables c'est-à-dire : $\forall 1 \leq i \leq n$ on a $\begin{cases} p_i+q_i \neq \perp \\ et \\ z_i \neq \varnothing \end{cases}$

1.2.2 Description d'arbres et traits partagés

Une description d'arbres est un ensemble de nœuds et de relations d'ascendance, de parenté et de précédence entre ces nœuds. Les nœuds représentent des syntagmes (éventuellement vides) et les relations expriment les dépendances entre ces syntagmes. Les propriétés morpho-syntaxiques de ces syntagmes sont décrites par des structures de traits. De cette façon, une description d'arbres peut être vue comme une spécification caractérisant un ensemble d'arbres syntaxiques et chacun de ces arbres apparaît comme un modèle particulier de cette description.

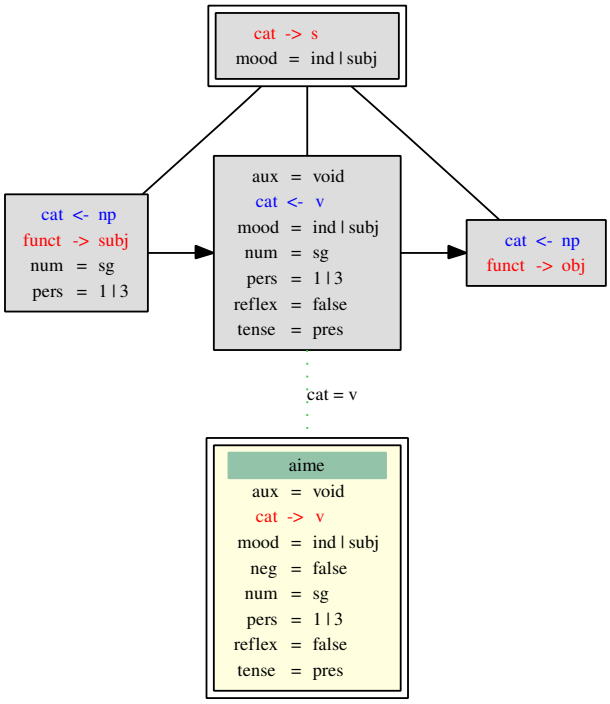

FIG. 1.1 – Un exemple de description d'arbres.

La Figure 1.1 représente une description d'arbre pour le verbe aimer conjugé avec la première ou la troisième personne du singulier de l'indicatif ou du subjonctif.

Dans des structures de traits différentes d'un même arbre, il peut exister des traits de même nom qui partagent une même valeur. En effet, l'information se propage d'un nœud vers un autre pour satisfaire des contraintes morpho-syntaxiques (accord, temps,...). Par exemple, dans la Figure 1.1 le verbe exige un sujet qui soit la première ou la troisième personne du singulier. Donc les valeurs attribuées aux traits *num* et *pers* du nœud de sujet doivent être les mêmes que ceux du verbe.

Pour résoudre ce problème de partage, les GI présentent un système de coréference qui consiste en l'affectation d'un indice à la valeur partagée qui réfère vers une valeur ou un ensemble des valeurs (présenté par environnement dans [GB03]). Ces indices sont présentés dans la Figure 1.2 par des entiers entre "<" et ">".

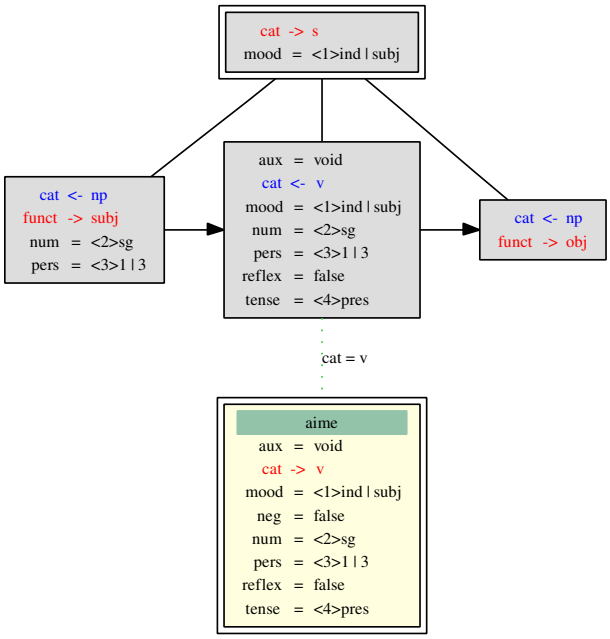

FIG. 1.2 – Une description d'arbres avec des indices de partage.

1.2.3 L'analyse dans les GI

L'analyse est la partie qui s'intéresse à l'étude des règles qui servent à expliquer d'une part, l'ordre des mots dans la phrase et, d'autre part, les relations qui existent entre les éléments qui la composent.
Les notions présentées dans les deux sous-sections précedentes forment les composantes de base dans le processus d'analyse d'une phrase par les GI. En effet, les grammaires d'interaction sont lexicalisées. Chaque item lexical est associé à un ensemble de descriptions d'arbres élémentaires. Analyser une phrase consiste tout d'abord à associer chaque mot de cette phrase à une description élémentaire issue du lexique pour former son étiquetage syntaxique. L'union des descriptions composant un étiquetage complété par des relations de précédence exprimant l'ordre des mots dans la phrase va former une description d'arbre, unique point de départ de l'algorithme de l'analyse syntaxique électrostatique[GB03]. Cet algorithme consiste à itérer l'opération de neutralisation de traits opposés pour spécifier progressivement la description initiale. L'analyse réussit si elle s'achève par un arbre complètement spécifié où tous les traits ont été neutralisés.

Les polarités servent alors à guider l'analyse syntaxique. Cet algorithme est implémenté dans un anlyseur syntaxique baptisé LEOPAR[1].
Un autre aspect important pour le TAL est l'analyse sémantique des langues naturelles. Ce point n'est pas oublié par les GI. En effet, il existe une étude detaillée dans [Per05] qui propose l'intégration de la sémantique dans les GI par l'ajout au formalisme des GI un niveau supplémentaire qui s'appuie sur les mêmes principes fondamentaux que le niveau syntaxique sauf que les structures sont des graphes acycliques orientés et non des arbres localement ordonnés. Mais elle n'est pas encore implémentée.

1.3 La sémantique dans les langues naturelles

La représentation sémantique correspond à une formule logique associée à une expression, et donne des informations sur le sens de celle-ci. Disposer d'une représentation du sens d'une phrase est un enjeu important, par exemple dans un but d'extraction d'information.
Depuis Montague (1974), les travaux menés en sémantique visent la construction d'une représentation sémantique combinée au procédé d'élaboration de la structure syntaxique. De nombreuses avancées ont été faites dans ce sens pour plusieurs formalismes grammaticaux tels que les grammaires catégorielles, les grammaires fonctionnelles lexicales (LFG) et les grammaires d'arbre adjoints (TAG). Sachant que les GI sont des descendantes de grammaires catégorielles et des TAG, il sera donc préferable d'étudier les essais qui ont visé l'intégration de la sémantique à ces deux formalismes.

1.3.1 L'ambiguïté de portée

Il existe des situations ou une analyse syntaxique correspond à plusieurs analyses sémantiques. Par exemple, l'ambiguïté de portée correspond à une situation où deux quantificateurs peuvent être interprétés comme ayant une portée supérieure l'un par rapport à l'autre. Un exemple célèbre d'ambiguïté de portée est donné par la phrase :

$$\textit{Tout homme aime une } \textit{femme}.$$

l'ambiguïté de cette phrase provient de la présence de deux quantificateurs dont les portées sont interprétés de deux façons :

1. pour chaque homme, il existe une femme, pour laquelle, l'homme est amoureux. Ce cas est présenté par la formule logique :

$$\forall x.man(x) \rightarrow (\exists y.woman(y) \wedge loves(x,y))$$

2. tous les hommes aiment la même femme. Cela correspond à la formule logique :

$$\exists y.woman(y) \wedge (\forall x.man(x) \rightarrow loves(x,y))$$

1.3.2 La sémantique plate en TAG

La sémantique plate utilise une logique fondée sur celle présentée dans [Bos96], augmentée de variables d'unification[Gar06b]. Cette logique présente certains avantages tels que la réduction des ambiguïtés de portée (via un procédé de sous-spécification), ou la composition des formules élémentaires lors de dérivations (via l'utilisation du procédé d'unification de traits).

[1] Voir : www.loria.fr/equipes/calligramme/leopar

Le principe de la sémantique plate appliquée aux grammaires TAG consiste à associer des formules logiques du premier ordre aux arbres élémentaires, formules qui vont être composées[2] lors de l'analyse de phrases par combinaison d'arbres.
Les objets utilisés dans cette représentation sont les suivants :

- des constantes individuelles (notées sous formes de chaînes de caractères),
- des constantes individuelles d'unification (notées a, b, c, ...),
- des variables individuelles d'unification (notées x_i),
- des constantes de label d'unification (notées l_i),
- des variables de label d'unification (notées s_i),
- des constantes de portées (notées h_i),
- des relations n-aires prenant en paramètre des constantes individuelles[3], des variables individuelles ou des constantes de portée,
- et des relations de portée (notées $\geq$).

Les formules utilisées sont de deux types suivant qu'elles comportent des variables d'unification ou non : formules d'unification et formules saturées.
Formellement, la définition d'une formule d'unification en logique L_U est la suivante :

Définition 1 *Soient l une constante ou variable de label, h une constante de portée, i_1,...,i_n des variables ou constantes individuelles ou constantes de portée, et R^n une relation n-aire, alors :*

1. $l : R^n(i_1, \ldots, i_n)$ *est une formule* L_U.
2. $h \geq l$ *est une formule* L_U.
3. ϕ,ψ *est une formule* L_U *ssi* ϕ *est une formule* L_U *et* ψ *est une formule* L_U.
4. *ce sont les seules formules* L_U.

Expression en langue naturelle	**Formule L_U associée**
un	$l : \exists(x, h_1, h_2)$, $h_1 \geq s_1$, $h_2 \geq s_2$
chien	$l : Chien(x)$
un chien aboie	$l_0 : \exists(x_1, h_1, h_2)$, $h_1 \geq l_1, h_2 \geq l_2$, $l_1 : Chien(x_1)$, $l_2 : Aboyer(x_1)$

FIG. 1.3 – Exemples de représentation sémantique en logique L_U

Un intérêt fort de cette logique est la sous-spécification des relations de portée permettant de représenter les ambiguïtés sous forme « factorisée ». En L_U, cette sous-spécification est représentée à l'aide de l'opérateur $\geq$ symbolisant une relation de portée au sens large entre deux prédicats.

Définition 2 *Soit ϕ une formule saturée (i.e. sans variable d'unification). Alors la relation de portée entre une constante de portée et une variable de label $\geq_\phi$ est définie par :*
$\forall k, k', k'' \in L_\phi \cup H_\phi$[4]

1. $k \geq_\phi k$.
2. $k \geq_\phi k'$ *si* $k \geq k'$ *est dans* ϕ.
3. $k \geq_\phi k''$ *si* $k \geq_\phi k'$ *et* $k' \geq_\phi k''$.

[2] Cette composition correspond en pratique à une conjonction des formules élémentaires.
[3] Ces constantes peuvent être d'unification ou non.
[4] L_ϕ et H_ϕ représentent respectivement l'ensemble des variables de label et celui des constantes de portée.

4. *si* $\exists k : \tau$ *avec* $\tau(..,k',..)$ *alors* $k \geq_\phi k'$ *et* $\neg(k' \geq_\phi k)$.
5. *si* $\exists R^n(..,k,..,k',..) \in \phi$ *alors* $\neg(k \geq_\phi k')$ *et* $\neg(k' \geq_\phi k)$.
6. *ce sont les seules représentations de* $\geq_\phi$.

La propriété 4 exprime le fait que, si une variable de label apparaît comme argument d'une relation, alors cette variable a une portée moindre que la constante de label étiquetant la relation en question.
A ce stade nous n'avons pas abordé la question de la résolution des ambiguïtés contenues dans une formule sous - spécifiée. Cela passe par l'emploi d'une fonction injective $P : H_\phi \rightarrow L_\phi$. Notons ϕ' la formule L_U résultante du remplacement dans ϕ de toutes les constantes de portée $k \in H_\phi$ par $P(k)$. La fonction P réalise un branchement (plugging en anglais) entre une constante de portée et une variable de label. La résolution d'une ambiguïté de portée correspond à la détermination de tous les branchements possibles pour une formule saturée ϕ :

Définition 3 *P est un branchement possible pour une formule saturée* ϕ ***ssi*** $\forall k,k' \in L_\phi$ *: si* $k \geq_{\phi'} k'$ *alors soit* $k = k'$*, soit* $\neg(k' \geq_{\phi'} k)$.

Afin de clarifier ces notions,l'exemple (1) a pour formule L_U associée (2) :

(1) Tout homme aime une femme.

(2) $l_0 : \forall(x_1,h_1,h_2)$, $h_1 \geq l_1, h_2 \geq l_2$, $l_1 : Homme(x_1), l_2 : Aimer(x_1,x_2)$, $l_3 : \exists(x_2,h_3,h_4)$, $h_3 \geq l_4$, $h_4 \geq l_2$, $l_4 : Femme(x_2)$

Il s'en suit les branchements acceptables suivants :

1. $\{h_1 \mapsto l_1$, $h_2 \mapsto l_2$, $h_3 \mapsto l_4$, $h_4 \mapsto l_0\}$, ce qui correspond à la formule $l_0 : \forall(x_1,l_1,l_2)$, $l_1 : Homme(x_1), l_2 : Aimer(x_1,x_2)$, $l_3 : \exists(x_2,l_4,l_0)$, $l_4 : Femme(x_2)$ (i.e. tous les hommes aiment la même femme),
2. $\{h_1 \mapsto l_1$, $h_2 \mapsto l_3$, $h_3 \mapsto l_4$, $h_4 \mapsto l_2\}$, ce qui correspond à la formule $l_0 : \forall(x_1,l_1,l_3)$, $l_1 : Homme(x_1), l_2 : Aimer(x_1,x_2)$, $l_3 : \exists(x_2,l_4,l_2)$, $l_4 : Femme(x_2)$ (i.e. tous les hommes aiment une femme, pas forcément la même).

1.3.3 La sémantique de Montague : répresentation en λ-calcul typé

R. Montague a proposé dans [Mon74] que la sémantique d'une phrase soit obtenue par une combinaison du sens de chaque mot de cette phrase suivant sa structure syntaxique. Alors il propose un calcul sémantique compositionnel, dans lequel :

- à chaque mot du lexique est associée une représentation sémantique (λ-calcul typé),
- les règles syntaxiques de la grammaire sont couplées avec des règles sémantiques indiquant comment calculer le sens de la phrase en fonction du sens de ses constituants (analogie entre les catégories syntaxiques et les types sémantiques).

1.3.3.1 le λ-calcul typé

Le λ-calcul est un langage de calcul fonctionnel inventé par A. Church[Chu40]. Le syntaxe de ce langage est défini par :

Définition 4 *Soient X un ensemble infini dénombrable de variables et C un ensemble fini de constantes. On definit un* λ*-terme T par :*

$$T ::= c \quad | \quad x \quad | \quad \lambda x.T \quad | \quad TT$$

où $x \in X$ *et* $c \in C$.

L'ensemble des λ-termes typés est donc défini comme le plus petit ensemble contenant :
- les constantes,
- les variables,
- pour toute variable x de type α et tout λ-terme t de type τ, le λ-terme $\lambda x.t$ représente la fonction associant à x le λ-terme t et il est de type $\alpha \to \tau$,
- pour tous λ-termes t de type $\tau \to \sigma$ et u de type τ, le λ-terme tu représente l'application fonctionnelle de t à u et il est de type σ.

Ce langage est étendu par R. Montague pour définir les représentations sémantiques en langue naturelle. Il manipule des constantes et des variables qui peuvent référer à des prédicats ou à tout objet par la théorie des types, ce qui lui permet d'être un langage d'ordre supérieur.
Les types de base de ce langage, pour la sémantique, sont :

1. e : type des individus,
2. t : type des phrases complètes.

A partir de la combinaison de ces deux types, tous les λ-termes du lexique seront typés. Comme exemple, le verbe transitif aimer sera considéré comme une fonction (ou prédicat) à deux arguments qui représentent son sujet et son objet :

$$\lambda xy.love(x,y) : \quad e \to e \to t$$

1.3.3.2 Analogie entre les catégories syntaxiques et les types sémantiques

R. Montague associe à chaque valeur de catégories syntaxiques un type sémantique. Ces types sont construits grâce à la définition des trois types atomiques pour le syntaxe et à la composition syntaxique dans un formalisme grammatical. Ces trois types sont le *np* (noun phrase), *s* (phrase) et *n* (noun). Ils sont associés à ces trois types sémantiques :
- $np \quad \rightsquigarrow \quad e$
- $n \quad \rightsquigarrow \quad e \to t$
- $s \quad \rightsquigarrow \quad t$

Après, pour typer un terme du lexique il faut voir la règle syntaxique qui aboutisse à la valeur de sa catégorie syntaxique. Comme exemple, soit la grammaire qui analyse la phrase "Jean aime Marie" defini par l'ensemble des règles suivantes :

1. S → NP VP
2. VP → TV NP
3. TV → aime
4. NP → Jean
5. NP → Marie

Comment trouver les types sémantiques de mots de ce lexique ?
La résolution de ce problème se fait en deux étapes :

1. chercher les types syntaxiques de ces mots en fonction de trois types atomiques *np*, *n* et *s*,
2. remplacer les types atomiques par leurs types sémantiques.

Dans cet exemple la prémière étape se fait ainsi :

1. S → NP VP $\rightsquigarrow \langle VP \rangle_{syn} = np \to s$
2. VP → TV NP $\rightsquigarrow \langle TV \rangle_{syn} = np \to \langle VP \rangle_{syn} = np \to np \to s$
3. TV → aime $\rightsquigarrow \langle aime \rangle_{syn} = \langle TV \rangle_{syn} = np \to np \to s$

4. NP $\rightarrow$ Jean $\rightsquigarrow \langle Jean \rangle_{syn} = np$
5. NP $\rightarrow$ Marie $\rightsquigarrow \langle Marie \rangle_{syn} = np$

et donc la deuxième étape donne le résultat suivant :

- $\langle aime \rangle_{sem} = e \rightarrow e \rightarrow t$
- $\langle Jean \rangle_{sem} = e$
- $\langle Marie \rangle_{sem} = e$

Ce qui verifie le type de la représentation sémantique de *aime* donné précédemment, et assure que les types des représentations de *Jean* et *Marie* sont ceux des arguments de ce verbe. Il ne reste donc que le calcul de la sémantique de la phrase "Jean aime Marie".

1.3.3.3 Calcul de la sémantique

Le calcul de la sémantique d'une phrase se fait à l'aide de l'arbre syntaxique (structuration syntaxique) et l'application des λ-termes mises comme lexique sémantique des mots. Ce calcul est detaillé par un exemple dans la Figure 1.4.

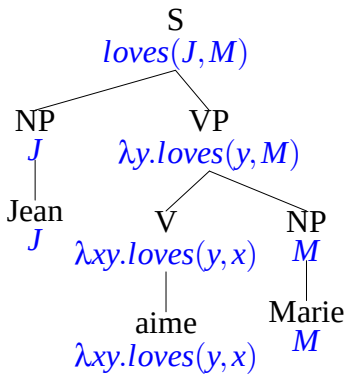

FIG. 1.4 – Calcul de la sémantique de "Jean aime Marie".

1.4 Conclusion

Aprés une brève présentation de grammaires d'interaction, le contexte de la sémantique dans le TAL est présenté à travers deux études : la sémantique sous-spécifiée et la sémantique de Montague. Il s'agit d'exploiter ces études pour intégrer la sémantique dans les GI. Cette exploitation sera détaillée et discutée dans les chapitres suivants.

Chapitre 2

Intégration de la sémantique aux GI

2.1 Introduction

Ce chapitre présente l'approche adoptée pour introduire de la sémantique dans les GI. Il détaille les étapes menant à intégrer une sémantique dans un arbre de GI.

Dans la deuxième partie du chapitre précédent, deux représentations sémantiques ont été présentées : la sous-spécifiée de J. Bos et les λ-termes de R. Montague.

Dans cette étude, la représentation de Montague a été choisie. Il est sans doute possible de faire un travail équivalent avec la représentation de Bos.

2.2 L'ajout d'un trait sémantique dans les GI

Comme est detaillé dans la première partie du chapitre précédent, le formalisme de GI manipule des descriptions d'arbres dont les nœuds sont des structures des traits polarisés. Ces structures ont des traits qui présentent les caractéristiques morpho-syntaxiques des composantes d'une phrase de la langue naturelle. Par exemple, le trait de la catégorie syntaxique, noté *cat*, caractèrise les catégories syntaxiques et il est utilisé comme le montre la Figure 2.1 pour analyser la phrase "Jean aime Marie".

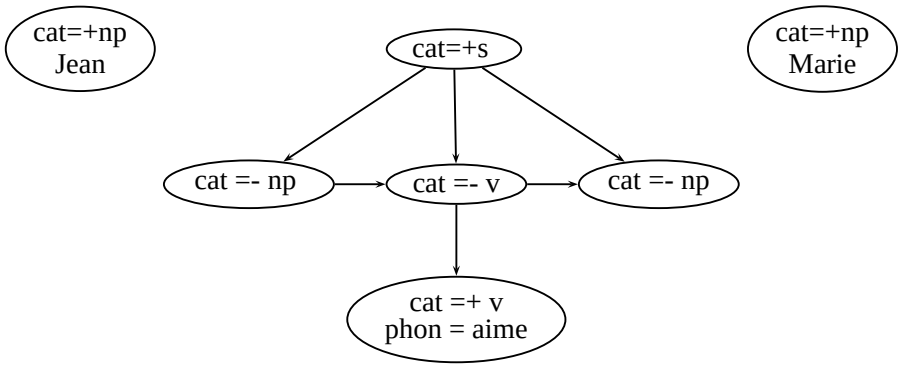

FIG. 2.1 – L'utilisation du trait *cat*.

Il est important de noter que le calcul de la sémantique d'une phrase est basé sur les compositions des structures syntaxiques. De ce point de vue, il est naturel d'ajouter un nouveau trait sémantique noté

sem qui se comporte de la même manière que celui de la catégorie syntaxique.

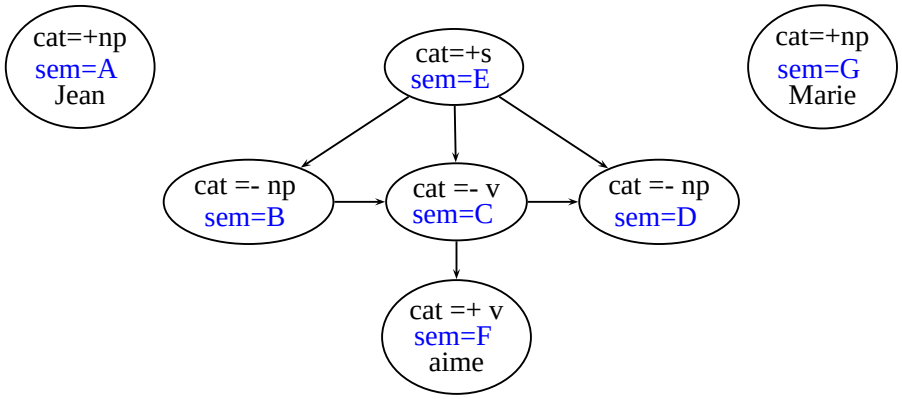

FIG. 2.2 – L'ajout d'un trait sémantique *sem*.

Donc en attribuant des valeurs au trait *sem* comme le montre la Figure 2.3, des questions ont étés posées :

1. quelle polarisation sera attribuée à ce trait ?
2. comment décrire la sémantique de la phrase à partir des ces valeurs ?

2.2.1 La neutralité du trait *sem*

Le nouveau trait ne joue pas de rôle actif et ne doit pas contraindre l'analyse syntaxique. Il utilisé pour garder une trace des unifications de l'analyse syntaxique. Donc, sa polarisation sera toujours neutre.

2.2.2 Le principe d'attribution des valeurs au trait *sem*

Genéralement, un trait ne prend que des valeurs bien définies dans un domaine fini d'éléments. Il n'est pas possible de définir à priori une liste contenant les termes de la sémantique de Montague. Pour résoudre ce problème, il faut se debarrasser de cette contrainte qui exige des valeurs statiques et rendre possible l'attribution des valeurs dynamiques comme des variables ou des fontions.

Les traits dans les GI ont des valeurs pouvant être des disjonctions de valeurs atomique et ont la possibilité de coindexer ces valeurs. Pour le trait *sem*, les valeurs sont :

– soit des variables d'unification : elles sont représentées par le mécanisme de coindexation (elles apparaissent donc sous la forme « <i> ? »),
– soit des λ-termes qui peuvent faire référence aux variables d'unification présentes dans la description d'arbres.

Dans toutes les descriptions d'arbres utilisées durant cette étude, le principe suivant est imposé pour chacun de ses nœuds :

- si $cat \rightarrow c$ alors $sem = t$ tels que t est un λ-terme de type $< c >_{sem}$(le type sémantique correspondant à c),
- si $cat \leftarrow c$ alors $sem =< i >?$ (une variable d'unification).

Une conséquence de ce principe est que l'unification entre deux traits *sem* de valeurs s_1 et s_2 sera toujours possible. En effet, (s_1, s_2) contient toujours un λ-terme et une varible d'unification.

2.2.3 L'emplacement de l'information sémantique

Dans les grammaires d'interaction, une description d'arbres de la grammaire est construite pour un ensemble de mots qui se comportent de la même façon dans la composition d'une phrase. Un lexique contient les mots de la langue et ses caractéristiques morpho-syntaxiques.

L'information sémantique est divisée en deux partie selon son emplacement :

1. **les λ-termes lexicaux :** ils sont placés dans le lexique pour définir les sémantiques des mots. Ils seront consommés comme les ressources syntaxiques dans le trait *sem*. C'est la partie qui est spécifique à chaque mot. Par exemple, le fait que le prédicat associé à « aime » est « loves ».
2. **les λ-termes de composition :** ils sont placés aux nœuds racines des arbres pour définir l'ordre d'application entre les variables d'unification placés dans les autres nœuds. C'est une partie qui est commune à tous les mots associés à la même description d'arbres. Par exemple, pour tous les verbes transitifs, la sémantique s'obtient toujours par application du λ-terme lexical à l'objet puis au sujet.

2.3 Exemple de calcul de la sémantique dans les GI

Pour appliquer les notions vue sur le trait *sem*, l'exemple suivant montre le calcul de la sémantique pour la phrase « Jean aime Marie ».

Le lexique : le tableau suivant présente le lexique utilisé pour analyser la phrase.

Mot	cat	λ-terme	Type
Jean	np	J	e
Marie	np	M	e
aime	v	$\lambda yx.loves(x,y)$	$e \rightarrow e \rightarrow t$

Descriptions d'arbres : pour l'analyse de cette phrase, les arbres suivants sont mis en initiation de l'analyse syntaxique.

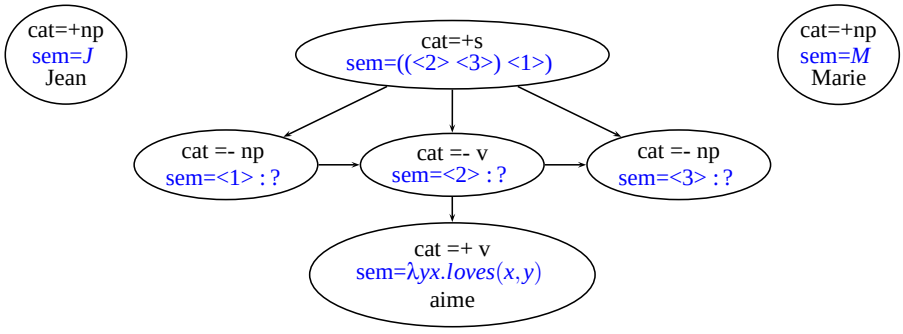

FIG. 2.3 – La description d'arbres initiale avec le trait *sem*.

Après l'interaction entre ces arbres, l'arbre donné dans la Figure 2.4 représente le résultat de l'analyse syntaxique. La valeur "((<2> <3>) <1>) est un λ-terme de composition et les valeurs J, M et $\lambda yx.loves(x,y)$ sont des λ-termes lexicaux.

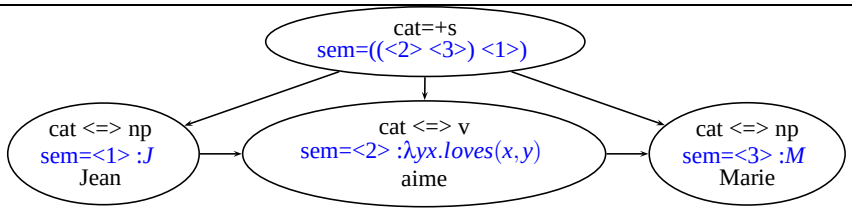

FIG. 2.4 – Le résultat de l'analyse syntaxique et sémantique.

Enfin, il faut associer à cette composition les λ-termes correspondants et faire la β-réduction pour avoir la bonne représentation sémantique de l'arbre. Dans le cas de l'exemple de "Jean aime Marie" la sémantique est calculée ainsi :

1. *sem*= ((<2> <3>) <1>)
2. *sem*= ((λ*yx*.*loves*(*x*,*y*) *M*) *J*) (association des λ-termes aux indices)
3. *sem*= (λ*x*.*loves*(*x*,*M*) *J*) (β-réduction)
4. *sem*= *loves*(*J*,*M*) (β-réduction)

2.4 Conclusion

Ce chapitre a montré que l'étude de Montague est adaptable aux grammaires d'interaction en modifiant les notions conçues pour l'analyse syntaxique. Le chapitre suivant montrera comment la méthode de ce chapitre a été mise en œuvre dans LEOPAR.

Chapitre 3

Implémentation de la sémantique dans les GI

3.1 Introduction

Les notions définies pour le GI sur l'aspect syntaxique sont déjà implémentés dans LEOPAR. Ce chapitre présente comment concrétiser l'approche du chapitre précédant dans ce logiciel.

3.2 Utilisation de la dimension sem de XMG

XMG[5](eXtensible MetaGrammar) est un compilateur de métagrammaires. Il sert à construire des arbres TAG ou des arbres GI à travers un langage unique mais il existe deux procédures différentes pour le compiler.

XMG offre deux dimensions pour la construction des arbres TAG : une syntaxique qui décrit la structuration de ces arbres et une autre sémantique qui est détaillée dans [Par07]. Les GI n'utilisent que la dimension syntaxique. Il s'agit de voir comment utiliser la dimension sémantique pour les GI.

3.2.1 Prédicats sémantiques en XMG

La représentation sémantique utilisée dans [Par07] pour le formalisme TAG est la représentation sous-spécifiée de J. Bos. La dimension sémantique de XMG utilise des conjonctions de formules E_i définies par la syntaxe abstraite suivante :

$$E := l : p(E_1, E_2, ..., E_n) \mid \neg l : p(E_1, E_2, ..., E_n) \mid E_i << E_j$$

où :

- p est un prédicat à n arguments ($n \in \mathbb{N}$),
- $<<$ est une relation de portée pour la représentation sous-spécifiée,
- l est un identifiant sémantique.

Cette dimension, en XMG, peut être liée avec la dimension syntaxique en utilisant des variables déclarées dans cette dernière. Ce mécanisme va permettre d'utiliser l'unification de l'analyse syntaxique pour réaliser l'analyse sémantique.

[5]Voir : http ://wiki.loria.fr/wiki/XMG/Documentation

3.2.2 Conversion du prédicat sémantique à la représentation de Montague

Dans la dimension syntaxique, un arbre est défini concrètement par une classe qui contient des nœuds. Chaque nœud a un ensemble de traits qui peuvent prendre des variables ou des constantes comme valeurs.

Une première étape consiste à implémenter des arbres GI avec deux traits dans chaque nœud : *cat* pour la catégorie syntaxique et *sem* pour le nouveau trait sémantique. On utilise ici que le trait *cat* pour la partie syntaxique pour simplifier la présentation. Il faudra évidemment ajouter les autres traits syntaxiques comme $funct$, *gen*, *num*,... Sauf que le premier trait prend ses valeurs d'un domaine défini et l'autre les prend dans un domaine indéfini.

Après, il s'agit d'utiliser le prédicat sémantique sans la relation de portée comme suit :

- si la valeur sémantique est une variable d'unification alors cela est donné par $sem =< i >?$ dans la dimension syntaxique,
- si la valeur sera un λ-terme t utilisant des variables d'unification $< i_1 >, < i_2 >, \ldots < i_n >$ alors cela est donné par $sem =< j >?$ dans la dimension syntaxique et par

$$\begin{bmatrix} label : sem & = & < j >? \\ pred : sem & = & t_0 \\ arg : sem & = & < i_1 >? \\ arg : sem & = & < i_2 >? \\ \vdots & \vdots & \vdots \\ arg : sem & = & < i_n >? \end{bmatrix}$$

dans la dimension sémantique avec t_0 représente un λ-terme pur tels que :

$$(t_0 \ < i_1 > < i_2 > \ldots \ < i_n >) \ \rightarrow_\beta \ t$$

La Figure 3.1 représente l'arbre GI construit pour les verbes transitifs. La boite en haut donne la sémantique de l'arbre syntaxique en bas en utilisant le prédicat sémantique comme détaillé ci-dessus.

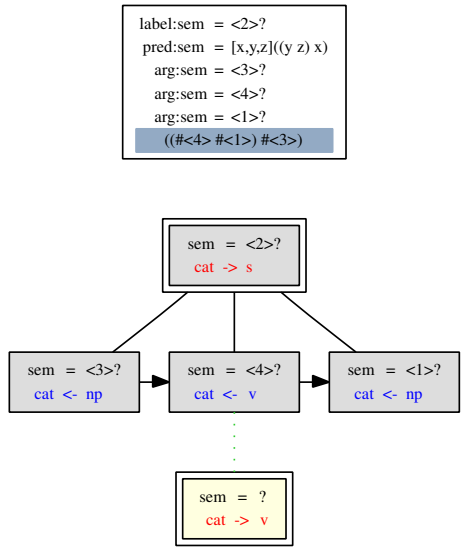

FIG. 3.1 – Description d'arbres pour les verbes transitifs.

Dans ce cas la sémantique <2> ? de l'arbre (qui apparaît dans le nœud $cat \rightarrow s$) est décrite dans la dimension sémantique comme l'application du λ-terme pur $\lambda xyz.((y\ z)\ x)$ aux variables d'unification <3>, <4> et <1>. Le résultat de l'application est donné dans la partie grisée.

Remarque : Syntaxiquement, la valeur de *P* ne peut pas être écrite comme un vrai λ-terme dans XMG. Cela est résolu par l'utilisation des chaînes de caractères qui seront interprétés comme des λ-termes par un module ajouté à LEOPAR et qui sera détaillé dans la section suivante.

3.3 Calcul de la sémantique dans LEOPAR

3.3.1 Architecture de LEOPAR

Pour effectuer l'analyse d'une phrase d'un langage donné, LEOPAR doit construire toutes les descriptions d'arbres associées à ce langage. Pour cela, il suit l'architecture donnée par la Figure 3.2.

Elle consiste en trois opérations binaires appliquées sur des ressources différentes :

1. **Construction du lexique syntaxique :** c'est l'opération de fusion entre les lemmes et leurs familles pour avoir un lexique syntaxique dont les éléments sont des lemmes étiquetées par les caractéristiques de leurs familles syntaxiques,

2. **Croisement mopho-syntaxique :** c'est l'opération de croisement entre les lemmes donnés dans le lexique syntaxique fourni par l'opération précédente et les formes fléchies de ces lemmes qui sont données dans un lexique morphologique. Cette opération aboutit à un lexique morpho-syntaxique contenant les formes fléchies étiquetées par leurs familles syntaxiques,

3. **Ancrage** : c'est l'opération d'association entre le lexique fourni par l'opération de croisement et les arbres produits par XMG pour donner des descriptions d'arbres prêtes au déroulement de l'algorithme d'analyse électrostatique.

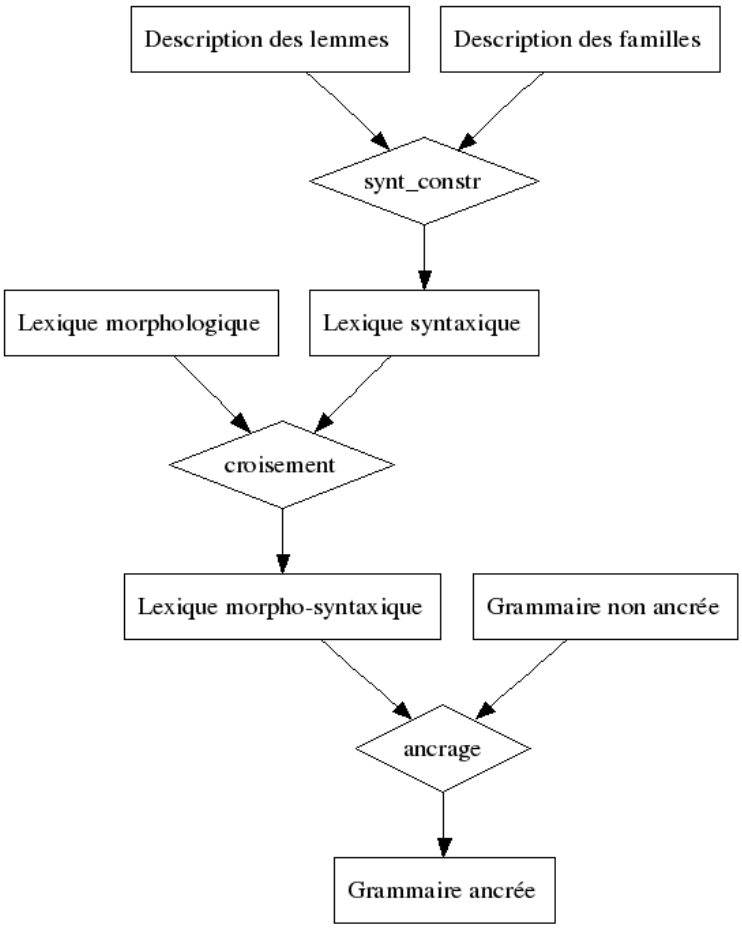

FIG. 3.2 – Architecture de LEOPAR.

Les arbres produits par XMG sont présentés au LEOPAR dans un fichier XML qui décrit les deux dimensions syntaxique et sémantique pour chaque arbre. Une modification est introduite au LEOPAR pour qu'il puisse interpréter la dimension sémantique dans ce fichier.

3.3.2 Calcul des λ-termes

Pour l'interprétation de la sémantique par LEOPAR, un module a été implémenté et ajouté au code source de LEOPAR. Il est inspiré d'un cours de Gérard Huet[Hue02] qui manipule des λ-termes pur et calcule leurs formes normales.

Une nouvelle syntaxe est définie pour les termes avec quantificateurs. Cette syntaxe remplace respectivement "λx", "∃x" et "∀x" par "[x]", "EX x" et "FA x". Donc le terme "$\lambda PQ.\exists x.(Px) \wedge (Qx)$" est représenté par "[P,Q] EX x.(P x)∧(Q x)".

Pour rendre ces λ-termes implémentables sur OCAML[6], le type t d'un terme est défini ainsi :

[6]OCAML est un langage de programmation fonctionnel. Les codes sources de LEOPAR et du module de G.Huet sont écrit en ce langage.

```
t  = Var of string
   | Lambda of string * t
   | Apply of t * t

   | Exists of string * t
   | Forall of string * t

   | And of t * t
   | Imply of t * t

   | Const of string
   | Unary of string * t
   | Binary of string * t * t
```

Deux fonctions sont implémentées : la première pour convertir un terme de type chaîne de caractères utilisant la nouvelle syntaxe en un autre de type *t* pour le calcul. La deuxième réalise l'operation inverse pour afficher le résultat du calcul.

Ces termes codés en chaîne de caractère seront facilement affichés dans les résultats de LEOPAR. Cela va être montré dans la section suivante.

3.4 Résultats

Avant de commencer la présentation des résultats, une remarque sur l'ordre des λ-termes de Montague : les exemples donnés dans les deux chapitres précédents contiennent des termes d'ordre inférieur alors que le travail qui va être présenté utilise des termes d'ordre supérieur.

3.4.1 Exemples d'entrées du lexique

Le tableau suivant donne un lexique utilisé pour tester la sémantique dans les GI.

Mot	Cat	λ-terme	Type
Jean	np	$\lambda P.(P\ \ J)$	$(e \to t) \to t$
homme	n	$\lambda x.man(x)$	$e \to t$
petit(e)	adj	$\lambda Px.small(x) \wedge (P\ \ x)$	$(e \to t) \to (e \to t)$
un(e)	det	$\lambda PQ.\exists x.(P\ \ x) \wedge (Q\ \ x)$	$(e \to t) \to ((e \to t) \to t)$
tout(e)	det	$\lambda PQ.\forall x.(P\ \ x) \wedge (Q\ \ x)$	$(e \to t) \to ((e \to t) \to t)$
aimer (1)	v (t)	$\lambda QP.(P\ \ \lambda x.(Q\ \ \lambda y.loves(x,y)))$	$((e \to t) \to t) \to ((e \to t) \to t) \to t$
aimer (2)	v (t)	$\lambda QP.(Q\ \ \lambda x.(P\ \ \lambda y.loves(y,x)))$	$((e \to t) \to t) \to ((e \to t) \to t) \to t$
dormir	v (i)	$\lambda P.(P\ \ \lambda x.sleep(x))$	$((e \to t) \to t) \to t$

Ce lexique donne deux représentations sémantiques pour chaque verbe transitif dans le but de résoudre le fameux problème sémantique de "l'ambiguïté de portée". L'ancrage de ce lexique avec les arbres GI donne les résultats suivants :

Le verbe aimer : suite aux deux représentations sémantiques de ce verbe, l'opération d'ancrage donne deux arbres qui portent chacun une sémantique comme le montre la Figure 3.3.

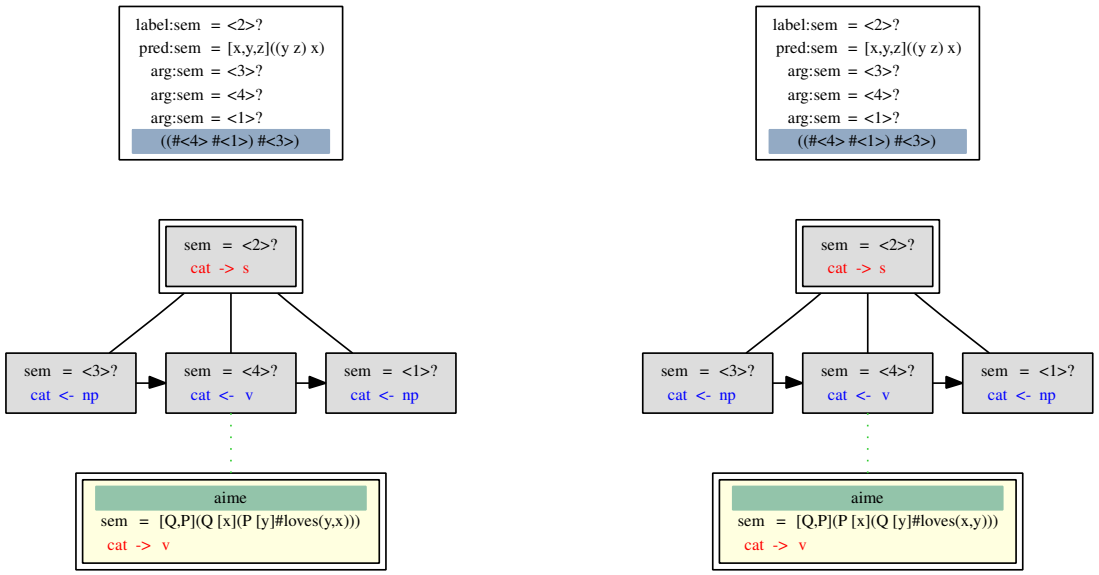

FIG. 3.3 – Les deux arbres du verbe "aimer".

Le verbe dormir : la Figure 3.4 correspond à l'arbre d'un verbe intransitif ancré par les caractéristiques du verbe dormir conjugué avec la troisième personne du singulier.

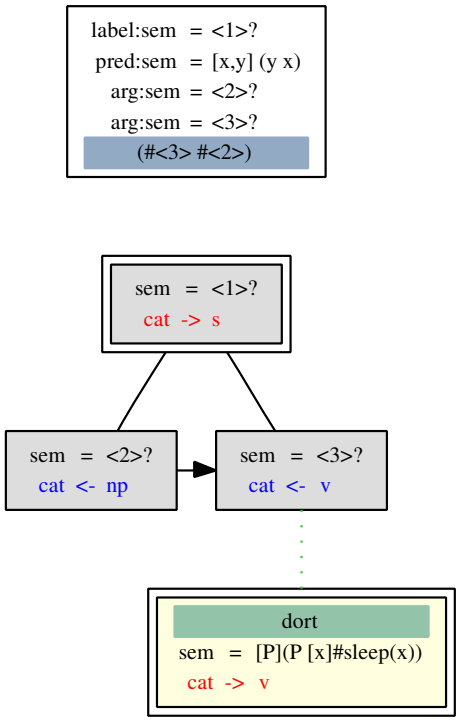

FIG. 3.4 – L'arbre du verbe "dormir".

Les déterminants avec les quantificateurs : les deux arbres de la Figure 3.5 ont la même structure syntaxique d'un déterminant et la même composition sémantique. Ces déterminants sont traduits au niveau sémantique à l'aide des quantificateurs « $\exists$ » et « $\forall$ ».

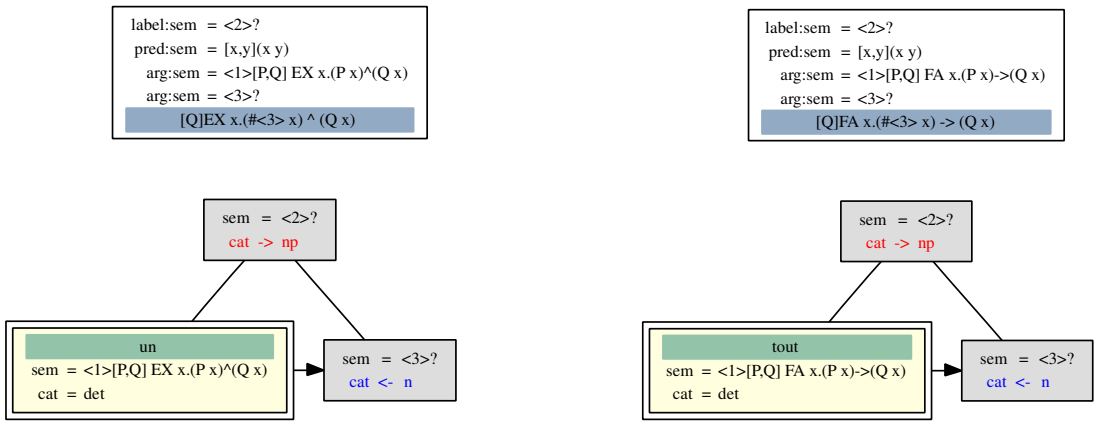

FIG. 3.5 – Les arbres de déterminants "un" et "tout".

L'adjectif petit : Le type d'adjectifs traité ici joue le rôle d'un modificateur d'un nom. Le syntagme de type *adj* prend un syntagme de type *n* pour donner un autre de type *n*. Dans la sémantique, cela est traduit comme le présente la figure 3.6 par l'application du terme donné à l'adjectif au celui donné au nom pour produire le terme de nom résultant.

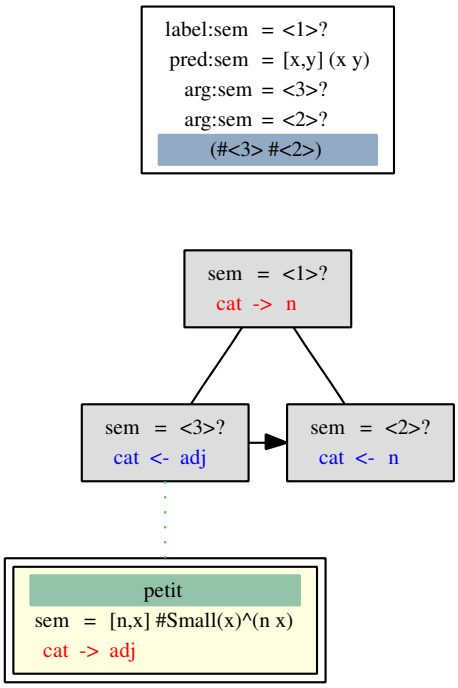

FIG. 3.6 – L'arbre de l'adjectif "petit".

3.4.2 Analyse de quelques phrases

L'utilisation des arbres présentés au dessus dans des phrases complètes donne ces résultats après l'analyse de LEOPAR.

La phrase : "Jean aime Marie."

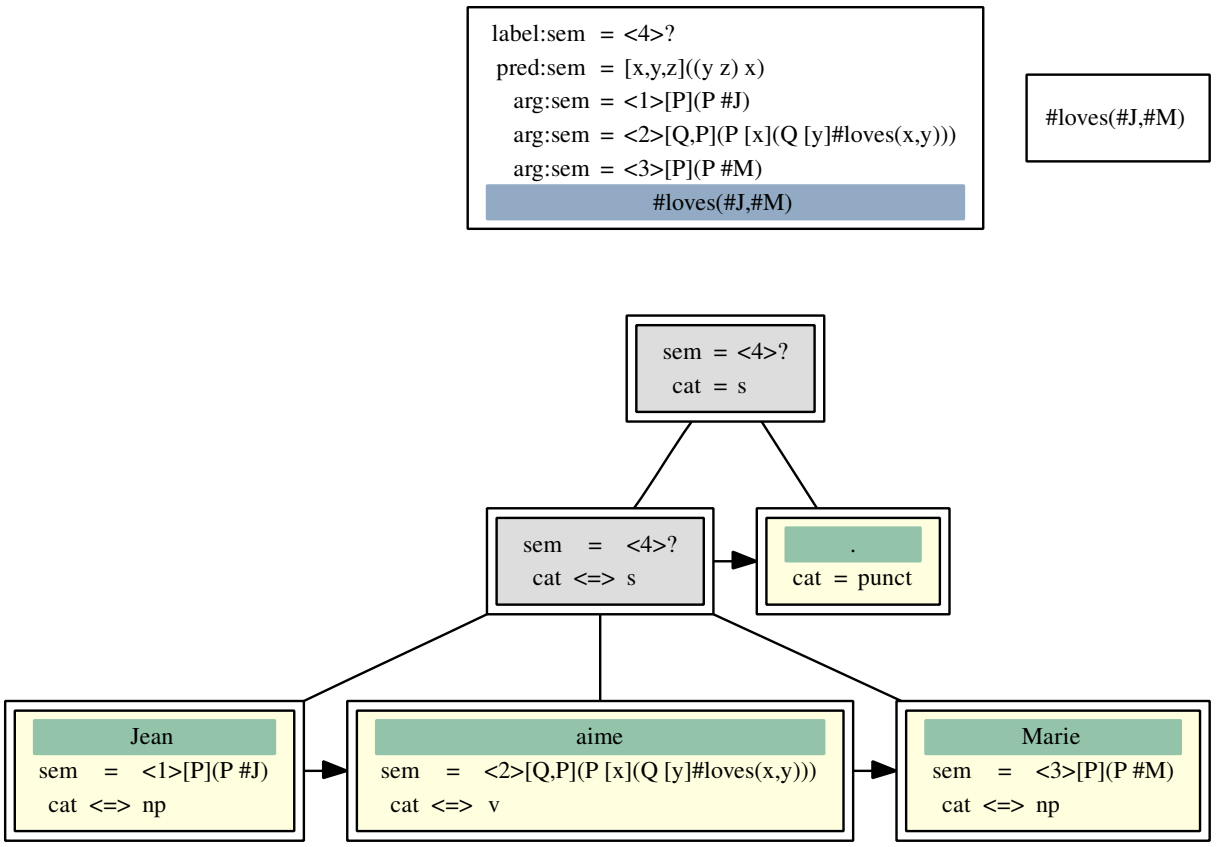

FIG. 3.7 – La sémantique de "Jean aime Marie.".

Dans cette phrase, l'utilisation de deux sémantiques du verbe "aimer" donnent une même sémantique qui est $loves(J,M)$.

La phrase : "Tout homme aime une femme."
l'analyse de cette phrase consiste à tester la résolution du problème de l'ambiguïté de portée par la sémantique de Montague. Avec les deux représentations sémantiques du verbe aimer, LEOPAR donne deux solutions pour l'analyse de cette phrase. Elles sont détaillées dans la Figure 3.8.

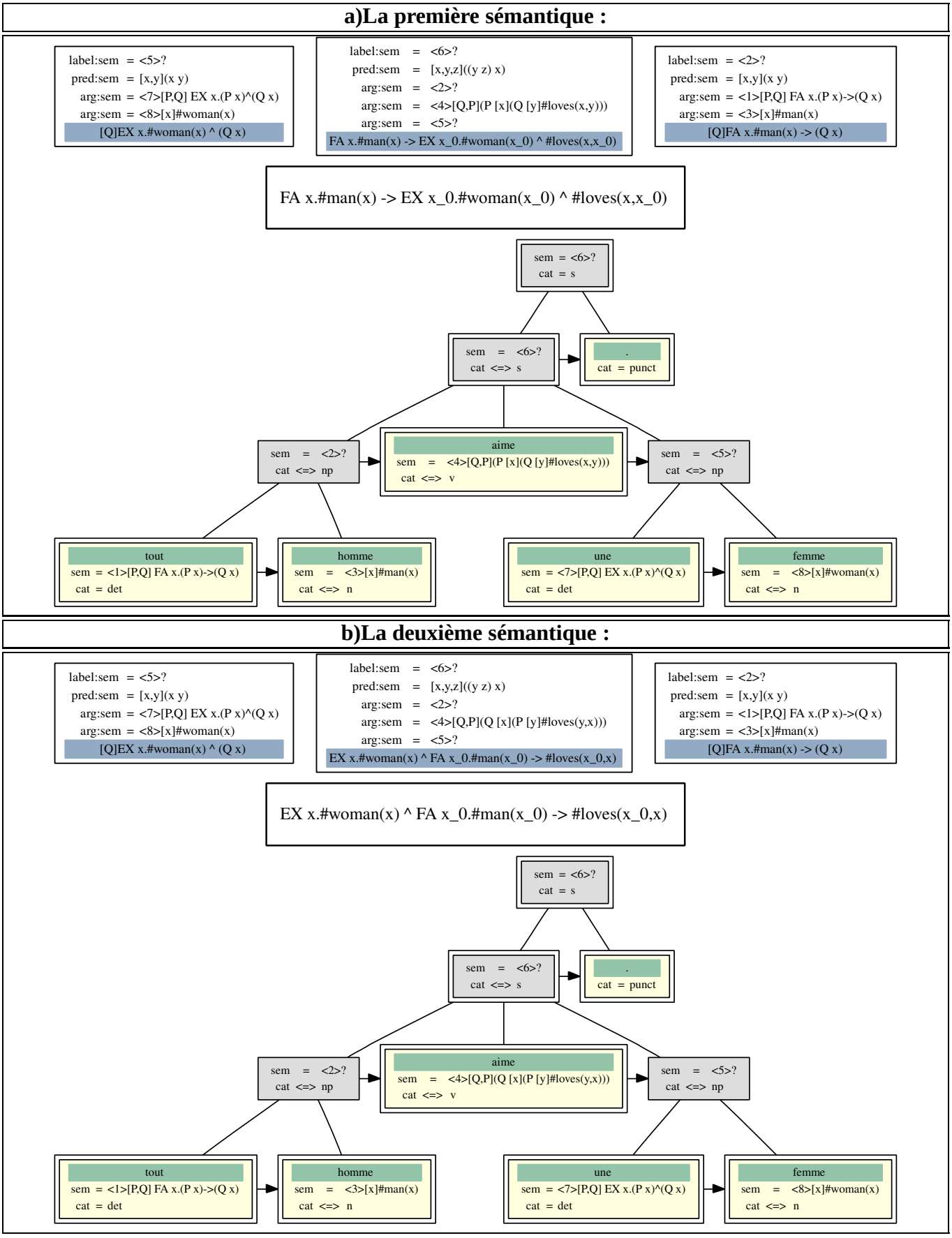

FIG. 3.8 – La sémantique de "Tout homme aime une femme".

La bonne résolution de l'ambiguïté est d'avoir une seule représentation syntaxique et toutes les représentations sémantiques correspondantes. Ici, la solution contient deux arbres syntaxiques presque identiques. Cela est issue à l'utilisation de deux arbres pour le verbe aimer.

3.5 Conclusion

Ce chapitre a illustré l'implémentation de la solution proposée à l'intégration de la sémantique dans les GI. Le début de ce chapitre a présenté comment construire une grammaire d'interaction avec une dimension sémantique. Puis le calcul de la sémantique dans LEOPAR a été detaillé. Enfin, les résultats de ce travail ont été illustrés par des figures données dans LEOPAR.

Conclusion

L'analyse syntaxique dans les grammaires d'interaction est un aspect bien étudié et implémenté dans LEOPAR. Pour lui intégrer une analyse sémantique, une solution inspirée de l'étude de R. Montague a permis d'avoir une relation entre la syntaxe et la sémantique dans les grammaires d'interaction.

Cette solution a été implémentée dans LEOPAR par des modifications dans ce logiciel et l'ajout d'un module de calcul de λ-termes qui peut être utile pour d'autres essais d'intégration de la sémantique dans les GI. Cela a montré que l'analyse syntaxique par les grammaires d'interaction peut être accompagnée par une analyse sémantique.

Le formalisme de Montague résout le problème de l'ambiguïté de portée par l'attribution de deux sémantiques aux verbes transitifs et par la suite deux arbres syntaxiques dans le cas de notre étude. Cette méthode construit deux analyses syntaxiques pour présenter les deux analyses sémantiques. Une solution à ce problème est l'extension au niveau des valeurs du trait *sem* qui permettra d'avoir une valeur sous la forme de disjonction de plusieurs λ-termes. Donc une analyse syntaxique pourra produire plusieurs analyses sémantiques.

Pour ce travail, une petite grammaire a été construite à partir d'une grammaire existante pour l'analyse syntaxique dans les grammaires d'interaction en lui ajoutant des informations sémantiques. Il serait intéressant de voir comment appliquer les mêmes principes à l'ensemble de la grammaire existante.

Pour l'instant, il n'y a pas de contrôle de type sur les λ-termes. Pour élargir la grammaire, le typage sera utile pour faciliter le développement. Pour cela on pourra utiliser un outil existant de calcul de sémantique de Montague comme Nessie[7] par exemple.

Les études présentées dans [GK03] et [Gar06b] ont montré que le même mécanisme de l'interface syntaxe/sémantique pourrait être utilisée avec différentes représentations sémantiques (sémantique de Montague, représentation sous-spécifiée). Il serait donc intéressant de faire un travail similaire à notre étude avec la représentation sous-spécifiée de J. Bos comme représentation sémantique pour les trait *sem* et ainsi trouver une seule solution syntaxique pour l'ambiguïté de portée.

[7] Nessie est un outil de construction de sémantique. Voir : http ://trac.loria.fr/ nessie/

Bibliographie

[BB05] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI, 2005.

[Bos96] J. Bos. Predicate logic unplugged. *Proceedings of the Tenth Amsterdam Colloquium*, pages 133–142, 1996.

[Chu40] A. Church. A formulation of a simple theory of types. *Journal of Symbolic Logic*, 5 :56–68, 1940.

[Gar06a] C. Gardent. Intégration d'une dimension sémantique dans les grammaires d'arbres adjoints. In *Actes de La 13ème édition de la conférence sur le TALN (TALN 2006)*, 2006.

[Gar06b] C. Gardent. Tree Adjoining Grammar, Semantic Calculi and Labelling Invariants. In *IWCS'07, Tilburg, The Netherlands*, 2006.

[GB03] G. Perrier G. Bonfante, B. Guillaume. Analyse syntaxique électrostatique. *Traitement Automatique des Langues*, 44 :3 Évolutions en analyse syntaxique, 2003.

[GK03] C. Gardent and L. Kallmeyer. Semantic construction in FTAG. In *Proceedings of the European chapter of the Association for Computational Linguistics (EACL'03), Budapest*, 2003.

[GK06] C. Gardent and E. Kow. Three reasons to adopt TAG-based surface realisation. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8), Sydney, Australia*, 2006.

[Hue02] Gérard Huet. Constructive computation theory. *Course notes on lambda calculus, University of Bordeaux I*, 2002.

[Mon74] Richard Montague. English as a formal language. *Formal Philosophy. Selected papers of Richard Montague, pages 188-221*, 1974.

[Par07] Y. Parmentier. *SemTAG : une plate-forme pour le calcul sémantique à partir de grammaires d'arbres adjoints*. PhD thesis, Nancy Université, 2007.

[Per05] Guy Perrier. La sémantique dans les grammaires d'interaction. *Traitement Automatique des Langues (T.A.L.)*, 45(3) :123 – 144, 2005.