HAL open science

Toward a knowledge representation model dedicated to the semantic analysis of the sentence

Jules Vanier, Christian Bassac, Patrick Henry, Renaud Marlet, Christian Retoré

To cite this version:

Jules Vanier, Christian Bassac, Patrick Henry, Renaud Marlet, Christian Retoré. Toward a knowledge representation model dedicated to the semantic analysis of the sentence. [Rapport de recherche] 2006, pp.46. inria-00084245v3

HAL Id: inria-00084245

https://inria.hal.science/inria-00084245v3

Submitted on 17 Jul 2006 (v3), last revised 19 Jul 2006 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Toward a knowledge representation model dedicated to the semantic analysis of the sentence

Jules Vanier — Christian Bassac — Patrick Henry — Renaud Marlet — Christian Rétoré

N° ????

Juillet 2006 (rapport rédigé en Octobre 2005)

Thème SYM

Rapport de recherche

ISSN 0249-6399

Toward a knowledge representation model dedicated to the semantic analysis of the sentence

Jules Vanier[*][†] , Christian Bassac [*][‡] , Patrick Henry [*][§] , Renaud Marlet [*][¶] , Christian Rétoré [*][‖]

Thème SYM — Systèmes symboliques
Projet SIGNES

Rapport de recherche n° ???? — Juillet 2006 (rapport rédigé en Octobre 2005) — 43 pages

Abstract: The present work deals with the study of natural languages semantics. It's about an accurate automation dedicated reformulation of Pustejovsky's Generative Lexicon. We provide an original lexical model being both descriptive of the internal semantic structure of words and applicable in an automatized way following a classical syntactic analysis, for instance that of categorial grammars'. Our formalization of the composition rules and our algorithms allowing to use them are illustrated through application to the treatment of metonymy and ambiguous uses of the french preposition "avec".

(This report was written in October 2005 but was not published until July 2006.)

Key-words: generative lexicon, composition rules, generative mecanisms, categorial grammars, Montague's semantics

[*] Projet SIGNES, INRIA Futurs (Bordeaux)
[†] Stagiaire de recherche, INRIA
[‡] Université de Bordeaux 3
[§] LaBRI / CNRS
[¶] LaBRI / INRIA
[‖] LaBRI / Université de Bordeaux 1

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,

Vers un modèle de représentation des connaissances dédié à l'analyse sémantique de la phrase

Résumé : Ce travail se situe dans le cadre de l'étude de la sémantique du langage naturel. Il s'agit d'une reformulation précise et automatisable du Lexique Génératif de Pustejovsky. Nous proposons un modèle lexical simultanément descriptif de le structure sémantique interne des mots et applicable de manière automatique à la suite d'une analyse syntaxique classique, par exemple celle des grammaires catégorielles. Notre formalisation des règles et nos algorithmes qui permettent de les appliquer sont illustrés par le traitement de la reconstruction métonymique et des usages ambigus de la préposition "avec".

(Ce rapport a été rédigé en octobre 2005 mais n'a été publié qu'en juillet 2006.)

Mots-clés : lexique génératif, règles de composition, mécanismes génératifs, grammaires catégorielles, sémantique de Montague

1 Introduction

L'un des desseins de la linguistique computationnelle est de traduire les phrases du langage naturel en énoncés logiques exploitables informatiquement. La traduction suivante est un exemple de ce qui est attendu :

> Tout barbier se rase lui-même.
> $\forall x barbier(x) \Rightarrow rase(x,x)$

Un tel objectif relève de l'extraction de la sémantique du langage naturel. Ainsi, la première question qui se pose lorsqu'on s'intéresse à un tel problème est celle du lieu de stockage des informations sémantiques. Plusieurs approches sont possibles à ce sujet et celle dans le cadre de laquelle notre travail se situe est celle d'une sémantique lexicalisée.

Considérer que la sémantique d'un mot est encodée dans l'entrée lexicale correspondante est une idée ancienne dont l'application pratique et effective remonte au début des années 70 avec l'article fondateur de Richard Montague "The Proper Treatment of Quantification in Ordinary English" [10]. Ce mode de traitement de la sémantique des langages naturels, basé sur des outils très formels comme le calcul syntaxique de Lambek et le λ-calcul, permet de traduire en énoncés logiques des fragments du langage assez complets (incluant la quantification, la temporalité, l'intentionnalité...).

Si cette approche est très élégante mathématiquement et se prête bien à des applications automatiques, elle reste néanmoins cantonnée à l'extraction de la sémantique superficielle du langage. En effet, la sémantique de Montague ne rend pas pleinement compte de la compositionnalité du langage. Par exemple, les phénomènes de spécification des verbes légers, illustrés ci-après, sont traités par l'introduction de multiples entrées lexicales pour un même mot :

1. faire$_1$ à manger = cuisiner
2. faire$_2$ la sieste = dormir
3. ...

Une telle approche de la polysémie semble mal rendre justice aux possibilités des langages naturels dont l'une des particularités, que ne partagent pas les langages hors contexte par exemple, est la possibilité d'obtenir, à partir d'un ensemble fini de mots, une infinité de constructions et donc de sens.

C'est afin d'analyser les utilisations originales d'un mot dans un contexte nouveau et inattendu en s'appuyant sur un modèle lexical rendant compte de la compositionnalité du langage que James Pustejovsky a construit le Lexique Génératif (LG) dans les années 90 (voir [11]). Le LG propose en premier lieu d'expliquer la polysémie logique et ses nombreux avatars tels que l'alternance verbale, l'alternance nominale et l'existence de différentes formes syntaxiques ou aspectuelles pour un même mot. Les quelques exemples suivants, tirés de [2], illustrent ce phénomènes :

1. **alternance verbale**
   (a) je commence la symphonie (TRANSITIF)
   (b) la symphonie commence (INTRANSITIF)

2. **alternance nominale**
   (a) j'emporte mon repas (NOURRITURE)
   (b) j'ai dormi pendant le repas (ÉVÈNEMENT)
3. **différences dans la forme syntaxique**
   (a) je commence le livre (+NP)
   (b) je commence à lire le livre (+VP)
4. **différences aspectuelles**
   (a) cuisiner un soufflé (ACCOMPLISSEMENT)
   (b) cuisiner des tomates (PROCESSUS)

Bien qu'ayant une grande puissance descriptive, qui justifie pleinement l'engouement qu'il suscite parmi les linguistes, le LG reste assez vague sur le fonctionnement des mécanismes expliquant les compositions de constituants de la phrase. De fait, la plupart des travaux de la communauté LG s'appliquent à affiner la modélisation lexicale pour rendre compte d'un nombre croissant de phénomènes linguistiques sans proposer de manière formelle une vision computationnelle des choses. Ainsi, il n'existe à notre connaissance qu'une seule implémentation d'un système d'analyse sémantique inspiré du LG, due à Moy Gupta [5].

C'est dans le projet de contribuer à combler cette lacune du LG que s'inscrit notre travail. Notre volonté est d'exploiter à la fois les travaux très formels mais au pouvoir descriptif limité issus de la sémantique de Montague, et les propositions ambitieuses, et parfois obscures du point de vue du traitement automatique, du LG. Nous pensons qu'il serait profitable de parvenir à articuler au sein d'un modèle commun de représentation des connaissances des informations exploitables par des outils qui ont fait la preuve de leur efficacité et des informations sémantiques dont l'usage par la communauté LG prouve qu'elles permettent de décrire de nombreuses particularités des langages naturels.

Ce travail propose en premier lieu une présentation succinte du formalisme des grammaires catégorielles et de la sémantique de Montague afin d'en exposer à la fois l'efficacité systématique et les limites en termes de description. Nous faisons ensuite un court exposé du formalisme du LG tel que présenté dans sa version originale [11]. Cet exposé se veut révélateur du manque de descriptions précises du fonctionnement des mécanismes explicatifs de la compositionnalité du langage dont souffre le LG.

Nous proposons par la suite notre propre version du LG qui consiste en la reformulation du modèle antérieur de manière à permettre d'articuler notre modèle lexical avec l'analyse syntaxique. Nous détaillons en outre quelques mécanismes de composition et présentons un nouveau mécanisme génératif permettant de traiter la reconstruction métonymique. Un tel phénomène apparaît dans les constructions comme :

Mon vélo est crevé

qui signifie que "La chambre à air de la roue de mon vélo est crevée". Enfin, nous proposons une amorce d'analyse du traitement des ambiguïtés dues au rattachement prépositionnel et en particulier au double emploi syntaxique de la préposition "avec" :

1. modificateur de verbe : “Jean regarde l’homme avec des jumelles” lu comme “Jean utilise des jumelles pour regarder l’homme”
2. modificateur de nom : “Jean regarde l’homme avec des jumelles” lu comme “Jean regarde l’homme détenant des jumelles”

2 Etat de l’art

Notre travail se veut situé en un point de convergence possible de deux formalismes classiques en linguistique computationnelle : les grammaires catégorielles et le lexique génératif. Nous présentons par la suite le premier au travers d’une application de la sémantique de Montague ([10]) et le deuxième par une description succinte des travaux fondateurs en la matière de James Pustejovsky ([11]).

2.1 Grammaires catégorielles et sémantique de Montague

Cette brève présentation du formalisme des grammaires catégorielles et de la sémantique de Montague s’articule autour de l’étude de la phrase : “Jean commence un livre”. La simplicité de l’exemple ainsi qu’un soucis de concision nous poussent à simplifier grandement ce formalisme et à ne pas entrer trop avant dans les détails. Le lecteur interessé trouvera des présentations plus détaillées dans des ouvrages généraux comme [6] ou [3].

2.1.1 Analyse syntaxique : grammaires catégorielles

La première phase de l’analyse d’une phrase en langage naturel est l’analyse syntaxique. Elle consiste en la détermination de la structure de la phrase et des relations (syntaxiques) qui existent entre ses constituants. Une telle analyse, dans le formalisme des grammaires catégorielles, repose sur des règles de réécriture des catégories syntaxiques. L’ensemble de ces règles de réécriture définit la grammaire de notre langage (ou plus précisemment fragment de langage).

Dans le cas de la phrase “Jean commence un livre”, le lexique minimal est le suivant :

Forme phonologique	**Catégorie syntaxique**
`Jean`	*NP*
`commence`	*TV*
`un`	*Det*
`livre`	*N*

Les catégories syntaxiques qui y sont introduites sont de deux natures. Ou bien elles sont primitives, ou bien elles ne sont que des écritures désignant une catégorie composée. Les catégories primitives sont au nombre de trois :

- *NP* : catégorie des groupes nominaux (*noun phrase*)
- *N* : catégorie des noms communs

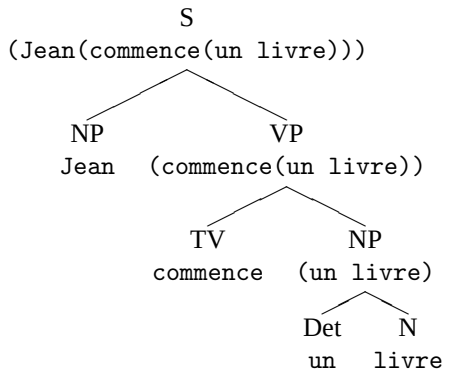


Figure 1: Exemple d'arbre d'analyse syntaxique

- S : catégorie des phrases du langage

Deux opérateurs de composition sont définis sur ces catégories : $/$ et $\backslash$. Si A et B sont des catǵories alors A/B et $A\backslash B$ sont aussi des catégories.

La grammaire associée au fragment du français présenté dans le lexique jouet ci-dessus est basée sur les règles de réécriture suivantes :

$$\begin{array}{cccc} (A/B) & B & \rightarrow & A \\ B & (B\backslash A) & \rightarrow & A \end{array}$$

Ce qui permet d'exprimer les catégories usuelles en fonction des catégories primitives :

- VP : catégorie des groupes verbaux est une écriture simplifiée de $NP\backslash S$
- TV : catégorie des verbes transitifs est une écriture simplifiée de VP/NP
- Det : catégorie des déterminants (articles,...) est une écriture simplifiée de NP/N

L'exploitation de nos règles grammaticales et du lexique permet de construire un arbre d'analyse syntaxique tel que décrit en figure 1.

Cette première analyse permet d'exclure certaines constructions comme "Jean livre commence". Rien n'est à ce point précisé de la sémantique de la phrase étudiée.

2.1.2 Traduction du français en forme logique : sémantique de Montague

Le langage vers lequel sera ici traduit le langage naturel étudié est une logique d'ordre supérieur. Le processus de traduction en lui-même fera appel au λ-calcul dont nous ne détaillons pas le fonctionnement. Notons simplement que nous ne présentons pas ici le cas d'une traduction en logique intentionnelle.

L'idée de Montague dans son papier fondateur [10] est d'établir un morphisme entre les catégories syntaxiques présentées en section 2.1.1 et l'ensemble de ses types sémantiques. Soient e et t deux objet distincts de notre langage logique. Les types sémantiques sont alors définis de la manière suivante :

- e et t sont des types
- si α et β sont des types, alors $\alpha \rightarrow \beta$ est un type
- rien d'autre n'est un type

Le type t est celui des valeurs de vérité, tandis que le type e est celui des entités. L'opérateur $\rightarrow$ introduit ci-dessus fonctionne de façon analogue aux opérateurs sur les types syntaxiques $/$ et $\backslash$. Les règles de composition suivantes définissent son comportement :

$$\frac{\Gamma \vdash \alpha \quad \Delta \vdash \alpha \rightarrow \beta}{\Gamma, \Delta \vdash \beta} \rightarrow_e$$

$$\frac{\Gamma \vdash \alpha \rightarrow \beta \quad \Delta \vdash \alpha}{\Gamma, \Delta \vdash \beta} \rightarrow_e$$

Ainsi, une propriété (prédicat unaire) est de type $e \rightarrow t$. Dans le tableau suivant, la notation A^* désigne l'image de la catégorie syntaxique A par le morphisme évoqué en début de section :

(Catégorie syntaxique)*	=	**Type sémantique**
S^*	=	t
NP^*	=	e
N^*	=	$e \rightarrow t$
$(A\backslash B)^* = (B/A)^*$	=	$A^* \rightarrow B^*$

Les constantes logiques usuelles sont typées comme suit :

Constante	**Type sémantique**
$\exists$	$(e \rightarrow t) \rightarrow t$
$\forall$	$(e \rightarrow t) \rightarrow t$
$\wedge$	$t \rightarrow (t \rightarrow t)$
$\vee$	$t \rightarrow (t \rightarrow t)$

Des constantes logiques sont utilisées pour dénoter des objets décrits dans le lexique et une forme logique est associée à chacun d'entre eux. L'ensemble des constantes qui interviennent dans cet exemple est $\{jean_e, commence_{e \rightarrow e \rightarrow t}, livre_{e \rightarrow t}\}$. Les λ-expressions permettent de rendre compte du fait que la sémantique de certains objets, les verbes par exemple, n'est pas complète. En effet, un verbe transitif privé de son sujet ou de son complément n'a pas de sens dont on puisse interroger la valeur de vérité. Cela revient à dire qu'il ne constitue pas une phrase valide pour le langage étudié.

Le lexique est complété par les informations sémantiques présentées ci-dessus :

Forme phonologique	**Type sémantique**	**Traduction**
`Jean`	e	$jean$
`commence`	$e \to (e \to t)$	$\lambda y.\lambda x.commence(x,y)$
`un`	$(e \to t) \to ((e \to t) \to t)$	$\lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
`livre`	$e \to t$	$\lambda x.livre(x)$

Le type sémantique de `un` est différent de ce à quoi on pourrait s'attendre. En effet, la catégorie *Det* est plus complexe qu'il n'y paraît. Le formalisme des grammaires catégorielles présenté ci-dessus n'est pas suffisant pour traiter les objets de cette catégorie syntaxique. C'est une des raisons qui rend nécessaire l'utilisation du calcul de Lambek.

Le calcul syntaxique de Lambek exploite, en plus des opérateurs $/$ et $\backslash$ des grammaires catégorielles, un opérateur $\bullet$ (non commutatif) caractérisé par :

$$A\backslash(B\backslash X) = (B \bullet A)\backslash X \quad (X/A)/B = X/(B \bullet A)$$

Les règles du calcul de Lambek sont les suivantes (voir [13] pour les détails) :

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma,\Delta \vdash B}\ \backslash_e \qquad \frac{A,\Gamma \vdash C}{\Gamma \vdash A\backslash C}\ \backslash_i\ \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta,\Gamma \vdash B}\ /_e \qquad \frac{\Gamma,A \vdash C}{\Gamma \vdash C/A}\ /_i\ \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash A \bullet B \quad \Gamma,A,B,\Gamma' \vdash C}{\Gamma,\Delta,\Gamma' \vdash C}\ \bullet_e \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta,\Gamma \vdash A \bullet B}\ \bullet_i$$

L'arbre d'analyse obtenu en utilisant ces règles est alors celui décrit en figure 2. On en déduit l'ordre d'application des λ-termes. Pour la partie gauche de l'arbre :

$$\begin{array}{c}
\big(\lambda y.\lambda x.commence(x,y)(u)\big)(jean) \\
\downarrow_\beta \\
\big(\lambda x.commence(x,u)\big)(jean) \\
\downarrow_\beta \\
commence(jean,u) \\
\text{abstraction sur } u \\
\lambda u.commence(jean,u)
\end{array}$$

Pour la partie droite de l'arbre :

$$\begin{array}{c}
\lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))(\lambda y.livre(y)) \\
\downarrow_\beta \\
\lambda Q.\exists x(\lambda y.livre(y)(x) \wedge Q(x)) \\
\downarrow_\beta \\
\lambda Q.\exists x(livre(x) \wedge Q(x))
\end{array}$$

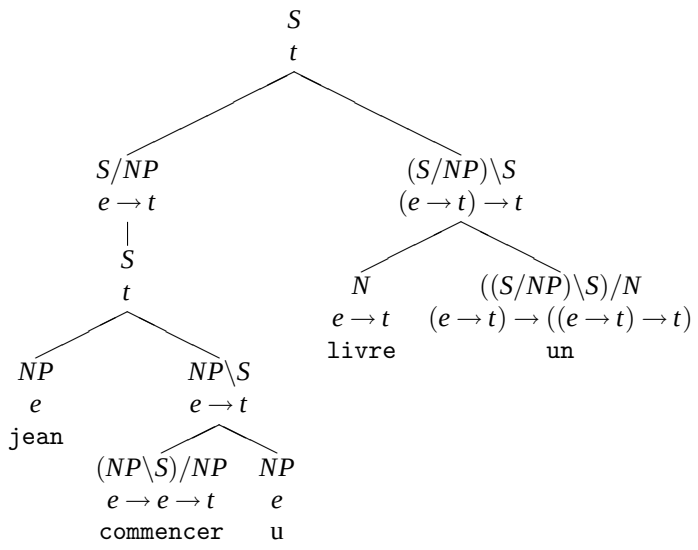


Figure 2: Arbre d'analyse obtenu grâce au calcul de Lambek

On applique le résultat précédent à l'expression obtenue pour la partie gauche de l'arbre :

$$\begin{array}{c}\lambda Q.\exists x(livre(x) \wedge Q(x))(\lambda u.commence(jean,u)) \\ \downarrow_\beta \\ \exists x(livre(x) \wedge \lambda u.commence(jean,u)(x)) \\ \downarrow_\beta \\ \exists x(livre(x) \wedge commence(jean,x))\end{array}$$

La traduction sémantique de la phrase en langage naturel "Jean commence un livre" est

$$\exists x(livre(x) \wedge commence(jean,x))$$

2.1.3 Conclusion

L'approche fondée sur les grammaires catégorielles et la sémantique de Montague permet de traduire la phrase "Jean commence un livre" par $\exists x(livre(x) \wedge commence(jean,x)$. Or quel sens donner au prédicat $commence(jean,x)$ sachant que x représente un livre? Notre volonté est d'obtenir des traductions plus proche de

$$\begin{array}{c}\exists x(livre(x) \wedge commence(jean, lire(jean,x)) \\ \text{ou} \\ \exists x(livre(x) \wedge commence(jean, écrire(jean,x))\end{array}$$

La traduction sémantique que fournit les méthodes décrites dans cette section est assez superficielle et ne permet pas, en l'état, de rejeter des constructions comme "un nuage avec un couteau" ou "regarder avec un marteau". Certains formalismes ce penchent plus en détail sur le contenu sémantique profond des items lexicaux; c'est le cas du Lexique Génératif.

2.2 Le Lexique Génératif

Cette section présente brièvement la version originale du Lexique Génératif (LG) de [11]. Des descriptions plus complètes de ce formalisme sont proposées dans [2] et [7]. Nous nous contentons ici d'un exposé qui se veut synthétique des principes du LG.

L'originalité de ce formalisme réside principalement dans la volonté de traiter de la même manière les éléments du lexique et les entités linguistiques résultant de la composition de ces éléments. Ainsi, toutes les unités du langage (morphèmes, mots, phrases) partagent une structure lexicale complexe commune. Ces représentations lexicales peuvent se composer et être manipulées par des *mécanismes génératifs* faisant émerger de la structure le sens du mot en contexte. En cela, le LG tente de cerner la compositionnalité du langage.

L'une des ambitions du LG est de proposer un mode de représentation des connaissances grâce auquel il n'est pas nécessaire d'introduire une entrée lexicale par sens de mot pour obtenir des analyse proches de la réalité linguistique. En effet, une propriété intéressante des langages naturels est justement qu'il est possible d'obtenir, à partir d'un ensemble fini de définitions des mots, une infinité de phrases, et donc de sens. Traiter cette propriété par l'énumération exhaustive des sens

possibles d'un mot en fonction des constructions linguistiques auxquelles il peut participer est à notre sens peu représentatif du fonctionnement des langages naturels et peu robuste lorsqu'on enrichit le lexique utilisé et donc les constructions possibles.

2.2.1 Structure de représentation des entités linguistiques

Dans le LG, les mots sont décrits par une structure de traits typés. Cette structure est organisée en trois niveaux de représentation :

- Structure argumentale (ARGSTR) : permet de spécifier le nombre et le type sémantique des paramètres intervenant dans la définition complète de la sémantique de l'item lexical. Ces arguments se distinguent par leur contribution à la sémantique du mot selon trois classes :
  - les arguments propres (ARG) : obligatoirement réalisés en surface (au niveau syntaxique) ou dénotant l'objet décrit.
  - les arguments par défaut (D-ARG) : pas nécessairement réalisés en surface mais participant à la sémantique du mot.
  - les arguments cachés (S-ARG) : sémantiquement contenus dans la sémantique de l'item lexical.
- Structure événementielle (EVENSTR) : permet de représenter le ou les événements lexicalement dénotés par l'item.
- Structure de qualia : permet d'une part d'établir des relations entre les éléments de la structure argumentale et de la structure événementielle, et d'autre part de spécifier le rôle sémantique de ces éléments. Ce niveau de description d'un item lexical correspondant aux modes d'explication d'Aristote, il est structuré selon quatre rôles :
  - Rôle formel : ce qui distingue l'objet dénoté par l'item lexical au sein d'un domaine plus vaste.
  - Rôle agentif : l'origine, la cause ou les facteurs ayant joué un rôle lors de la création de l'objet.
  - Rôle télique : fonction, but ou raison d'être de l'objet.
  - Rôle constitutif : relation entre l'objet, ses différentes parties et les autres objets dont il est constitutif.

Les figures 3 et 4 illustrent cette structure. Il est important de remarquer que toutes les variables introduites sont typées. La notation $x : \alpha$ signifie ainsi "x est de type sémantique α". De nombreux mécanismes propres au LG reposent sur l'utilisation d'une hiérarchie de ces types (existence d'une relation d'ordre notée $\preceq$).

Une autre particularité du LG est la notion de type pointé. Un tel type sémantique permet d'expliquer le comportement d'un mot comme "livre" qui dénote tantôt un contenu informationnel, tantôt un objet physique. Un opérateur sur les types sémantiques noté $\bullet$ permet de modéliser cette

$$\left[\begin{array}{ll} \texttt{couteau} & \\ \text{ARGSTR} & \left[\begin{array}{lcl} \text{ARG}_1 & = & x : \textit{couteau} \\ \text{D-ARG}_1 & = & y : \textit{humain} \\ \text{D-ARG}_2 & = & z : \textit{matière} \\ \text{D-ARG}_3 & = & p : \textit{lame} \end{array}\right] \\ \text{EVENSTR} & \left[\begin{array}{lcl} \text{D-E}_1 & = & e_1 : \textit{transition} \\ \text{D-E}_2 & = & e_2 : \textit{transition} \end{array}\right] \\ \text{QUALIA} & \left[\begin{array}{lcl} \text{FORMAL} & = & x \\ \text{AGENT} & = & \textit{fabriquer}(e_1, y, x) \\ \text{TELIC} & = & \textit{couper}(e_2, x, z) \\ \text{CONST} & = & \textit{partie_de}(p, x) \end{array}\right] \end{array}\right]$$

Figure 3: Représentation LG classique pour `couteau`

$$\left[\begin{array}{ll} \texttt{marcher} & \\ \text{ARGSTR} & \left[\begin{array}{lcl} \text{ARG}_1 & = & x : \textit{individu_animé} \end{array}\right] \\ \text{EVENSTR} & \left[\begin{array}{lcl} \text{E}_1 & = & e_1 : \textit{process} \end{array}\right] \\ \text{QUALIA} & \left[\begin{array}{lcl} \text{AGENT} & = & \textit{marcher}(e_1, x) \end{array}\right] \end{array}\right]$$

Figure 4: Représentation LG classique pour `marcher`

$$\left[\begin{array}{ll} \texttt{livre} & \\ \text{ARGSTR} & \left[\begin{array}{lcl} \text{ARG}_1 & = & x : \textit{info} \\ \text{ARG}_2 & = & y : \textit{objet_physique} \\ \text{D-ARG}_1 & = & z : \textit{humain} \\ \text{D-ARG}_2 & = & p : \textit{humain} \end{array}\right] \\ \text{EVENSTR} & \left[\begin{array}{lcl} \text{D-E}_1 & = & e_1 : \textit{transition} \\ \text{D-E}_2 & = & e_2 : \textit{processus} \end{array}\right] \\ \text{QUALIA} & \left[\begin{array}{lcl} \text{FORMAL} & = & \textit{contient}(y, x) \\ \text{AGENT} & = & \textit{écrire}(e_1, z, x) \\ \text{TELIC} & = & \textit{lire}(e_2, p, x) \end{array}\right] \end{array}\right]$$

Figure 5: Représentation LG classique pour `livre`

propriété en associant à la variable dénotant l'objet "livre" le type $information \bullet objet_physique$. La structure correspondante est illustrée en figure 5.

Certaines constructions dans lesquelles intervient une coprédication, comme dans la phrase "Ce livre est rouge et subversif", sont avantageusement modélisées par l'utilisation des types pointés. Dans la phrase citée, l'objet "livre" est à la fois "rouge", ce qui suggère sa nature d'objet physique matériel, et "subversif", ce qui se rapporte au monde des idées, de l'information. En considérant qu'un livre est de type simple (par opposition à pointé) *objet_physique* ou *information*, on analyse la phrase précédente comme non valide. Les cas traités par le type pointés relèvent souvent de l'ambuguïté logique. On trouvera une illustration de ces traitements dans [9].

2.2.2 Mécanisme Génératifs

Les mécanismes génératifs sont les opérations qui permettent d'élire les aspects d'un mot qui sont effectivement exprimés dans un contexte donné. L'opération consistant en l'application d'un constituant du langage à un autre est nommée application de fonction. Ce mécanisme est décrit comme la détermination de la structure commune aux deux mots la plus générale (au sens d'une relation d'ordre á définir sur les structures de qualia). Outre l'application de fonction classique, il existe trois classes de mécanismes génératifs :

- La Coercion[1] de type : autorise à contraindre le type d'un item afin de permettre l'application de fonction classique.
- La Co-composition : permet de rendre compte de l'existence de plusieurs applications de fonction distinctes pour un même item (fonctionnel donc généralement verbal), selon le complément effectivement présent.
- Le Liage sélectif : décrit les modifications ciblées (restreintes à une partie de la structure) qu'un adjectif apporte à un nom.

Les mécanismes de coercion sont systématiquement déclenchés dans la mesure ou le typage raffiné des variables, qui revient à définir des types plus spécifiques que le type *e* des entités de la sémantique de Montague, interdit certaines compositions qui devraient être considérées comme valides. En effet, plus l'ensemble des types sémantiques est vaste, et la description par conséquent précise, plus rares sont les situations où le type attendu par un verbe, par exemple, se trouve être exactement le type du constituant avec lequel il doit se composer. De manière générale, la coercion de type est définie de la façon suivante dans [11] :

> *A semantic operation that converts an argument to the type which is expected by a function, where it would otherwise result in a type error.*

Il existe deux mécanismes de coercion :

[1] Le nom anglais de ce mécanisme est *coercion*, qui se traduit en coercition, mais nous préférons adopter la convention de nommage admise dans la communauté LG.

$$\begin{bmatrix} \texttt{cuisiner} & \\ \text{ARGSTR} & \begin{bmatrix} \text{ARG}_1 & = & x : \textit{humain} \\ \text{ARG}_2 & = & y : \textit{matière} \end{bmatrix} \\ \text{EVENSTR} & \begin{bmatrix} \text{E}_1 & = & e_1 : \textit{processus} \end{bmatrix} \\ \text{QUALIA} & \begin{bmatrix} \text{AGENT} & = & \textit{acte_cuisiner}(e_1, x, y) \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \texttt{soufflé} & \\ \text{ARGSTR} & \begin{bmatrix} \text{ARG}_1 & = & x : \textit{nourriture} \\ \text{D-ARG}_1 & = & y : \textit{matière} \\ \text{D-ARG}_2 & = & z : \textit{humain} \\ \text{D-ARG}_3 & = & w : \textit{humain} \end{bmatrix} \\ \text{EVENSTR} & \begin{bmatrix} \text{E}_1 & = & e_1 : \textit{process} \\ \text{E}_1 & = & e_2 : \textit{process} \end{bmatrix} \\ \text{QUALIA} & \begin{bmatrix} \text{FORMAL} & = & x \\ \text{AGENT} & = & \textit{acte_cuisiner}(e_1, z, y) \\ \text{TELIC} & = & \textit{manger}(e_2, w, x) \\ \text{CONST} & = & y \end{bmatrix} \end{bmatrix}$$

Figure 6: Un cas de co-spécification

- Le *subtype coercion* qui exploite une relation d'ordre sur les types sémantiques pour réaliser une accomodation de type (un type plus spécifique que le type attendu est accepté grâce à ce mécanisme).
- Le *true complement coercion* qui autorise la composition avec certains aspects d'un mot (rôles de la structure de qualia) plutôt qu'avec la seule variable qui le dénote (ce mécanisme permet d'expliquer que la construction "commencer un livre" signifie "commencer à lire un livre" ou "commencer à écrire un livre").

Le mécanisme de co-composition traite de la polysémie verbale. Plusieurs phénomènes linguistiques sont couverts par ce mécanisme, comme par exemple la co-spécification qui explique certaines différences aspectuelles. Ainsi, le verbe "cuisiner" est compris comme un processus lorsque son objet est "des carottes", disons, alors qu'il désigne un accomplissement lorsqu'il se compose avec "un soufflé". La co-composition permet d'expliquer ce comportement. L'exemple de la figure 6 modélise le cas de co-spécification évoqué ci-dessus. Lors de la composition, l'identité des rôles agentifs des deux constituants ($Q_A(\texttt{cuisiner}) = Q_A(\texttt{soufflé})$) déclenche le mécanisme de co-composition qui entraine :

1. Le verbe `cuisiner` s'applique à son objet
2. L'objet `soufflé` co-spécifie le verbe

3. La composition des structures de qualia résulte en l'émergence d'un sens dérivé du verbe où le rôle formel du complément devient le rôle formel de tout le constituant (VP) nouvellement formé

L'emergence de sens dérivé est le produit d'une opération nommée *qualia unification* qui exploite l'existence supposée d'une relation d'ordre sur les structures de qualia et qui consite en la détermination du plus grand minorant commun aux deux structures (*unique greatest lower bound*). La co-composition explique aussi d'autres phénomènes linguistiques tels que la spécification des verbes légers ("faire", par exemple).

Le liage sélectif est le mécanisme génératif qui rend compte du comportement des adjectifs. Son nom fait référence à la tendance qu'ont certains adjectifs à ne modifier qu'un seul des aspects du mot auquel ils s'appliquent. Par exemple, "un bon couteau" est un couteau qui coupe bien ou un couteau de bonne facture. De la même façon, dans "un bon dactylographe" et "un bon fromage", l'adjectif `bon` revêt des sens très différents. Les adjectifs sont des entités linguistiques complexes et ce mécanisme est par conséquent difficile à cerner. Les travaux de [2] permettront au lecteur intéressé d'approfondir le sujet.

2.2.3 Conclusion

Le Lexique Génératif propose une modélisation du langage à fort pouvoir descriptif mais dont les mécanismes restent peu automatiques. En effet, si les définitions de la portée et de l'effet des mécanismes génératifs sont claires, il n'existe pas, à notre connaissance, d'écriture algorithmique de leur fonctionnement. En revanche, la qualité de description et d'explication des phénomènes linguistique que fournit le LG est très prometteuse.

2.3 Bilan de l'existant

Les deux visions du langage présentées dans cette section nous semblent très complémentaires. En effet, le formalisme des grammaires catégorielles et la sémantique de Montague proposent les traitements automatiques qui manquent au LG tandis que ce dernier fournit les informations sémantiques permettant de tendre aux interprétations que nous disions souhaiter en section 2.1.3.

La raison d'être de ce travail est une reformalisation du LG dans l'optique d'obtenir un système plus automatique de composition des items lexicaux permettant d'extraire de phrases en langage naturel une interprétation sémantique fine. Nos objectifs sont, entre autres, de parvenir à désigner comme non valide sémantiquement l'expression "regarder avec un marteau" et de donner à la phrase "Jean commence le livre" une interprétation mettant en exergue l'activité liée au "livre" qui est effectivement commencée.

3 Une reformalisation du lexique génératif

L'approche du Lexique Génératif développée dans [11] se veut très formelle et exploitable à la fois par les linguistes et les informaticiens de la communauté du TAL. Il nous semble cependant que certains points restent obscurs, en particulier en ce qui concerne la modélisation de la compositionnalité

du langage (c'est-à-dire les règles de composition et le fonctionnement des mécanismes génératifs). Notre travail a donc consisté en une reformulation du LG dans l'optique d'une implémentation prochaine. Cette section présente notre version du LG, formalisme lexical et méthodes de composition.

3.1 Hypothèses

Notre travail se base sur certaines hypothèses, correspondant à des sujets de recherche annexes, auxquelles nous ferons appel par la suite et que nous présentons ici.

3.1.1 Analyse syntaxique et quantificateurs

Notre travail se concentre sur une reformalisation du LG et ne détaille par conséquent pas certains outils requis pour l'analyse du langage. En particulier, nous supposerons disposer d'un outil d'analyse capable de fournir une certaine structure syntaxique indiquant l'ordre de composition des constituants de la phrase et gérant la portée des quantificateurs. En effet, la quantification est une question complexe de l'analyse des langages (le typage contre intuitif de l'article `un` présenté en section 2.1.2 le souligne) et mobilise à elle seule de nombreux travaux. Nous supposons néanmoins avoir accès à une structure syntaxique dans laquelle les quantificateurs *remontent* l'arbre d'analyse comme cela est illustré par la structure associée à la phrase "Tous les chats mangent une souris" dans la figure 7.

Deux lectures sont possibles :

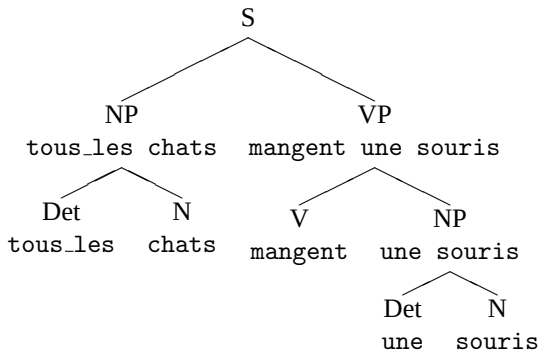
$$\forall x, \exists y[souris(y) \wedge (chat(x) \Rightarrow mange(x,y))]$$
$$\text{ou}$$
$$\exists y, \forall x[souris(y) \wedge (chat(x) \Rightarrow mange(x,y))]$$

mais nous ne traiterons pas de ce problème. Notre analyse, partant des feuilles, se termine au niveau indiqué par ★. Le mécanisme qui permet de conserver un lien entre un quantificateur et l'item quantifié (représenté naïvement par la décoration ($^{*}chats$) dans l'arbre) n'est pas détaillé ici.

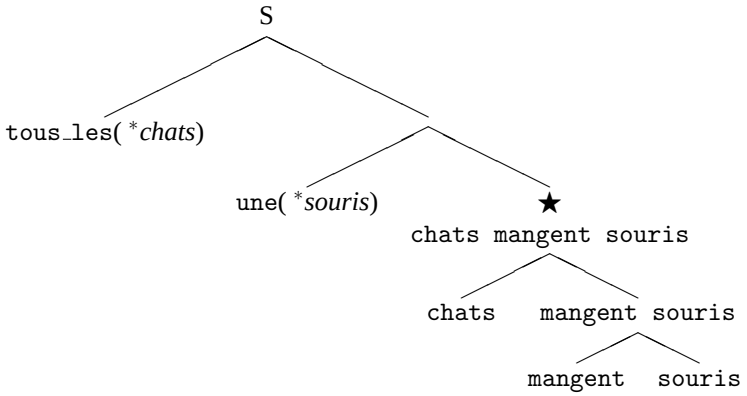
3.1.2 Hiérarchie des types et distance sémantique

Outre ce premier outil, il nous arrivera dans ce qui suit de faire référence à une hiérarchie des types sémantiques ainsi qu'à une notion de *distance sémantique*. Nous avons choisi d'indentifier l'ensemble des types sémantiques à l'ensemble des prédicats de notre langage de représentation. En effet, le typage d'une variable revient toujours à évaluer la vérification d'une certaine propriété par cette variable. Ce que nous considérons comme disponible est une collection de relations d'ordre (une par arité de prédicat) sur les types sémantiques. Chacune de ces relations est supposée posséder un plus grand élément correspondant au typage de la sémantique de Montague pour les prédicats concernés.

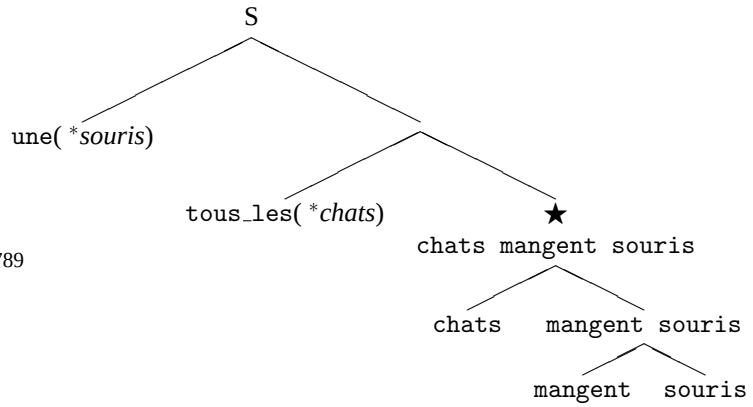
Outre cette hiérarchie de types, nous ferons appel à une distance (notée d) définie sur tout couple de prédicats du langage de représentation. Nous supposons que cette distance prend ses valeurs dans $\mathbb{R} \cup \{+\infty\}$. La valeur de la distance entre deux prédicats représente la communauté de sens qui


S
NP
tous_les chats
VP
mangent une souris
Det
tous_les
N
chats
V
mangent
NP
une souris
Det
une
N
souris

↓


S
tous_les(*chats)
une(*souris)
★
chats mangent souris
chats
mangent souris
mangent
souris

ou


S
une(*souris)
tous_les(*chats)
★
chats mangent souris
chats
mangent souris
mangent
souris

existe entre les deux objet représentés. Nous considérons que lorsque la distance est finie, il existe un proximité linguistique entre les sens des deux prédicats. Par exemple, la distance entre *manger* et *dévorer* est finie, tandis que $d(\textit{regarde}, \textit{martèle}) = \infty$.

3.1.3 Lexique et ontologie

Notre dernière hypothèse porte sur le lexique. Nous supposerons qu'il existe un lexique écrit selon le formalisme que nous exposons ci-après. Nous ne proposons donc pas notre propre lexique mais plutôt les contraintes à respecter lors de sa construction afin qu'il soit compatible avec les mécanismes que nous présentons.

Qui plus est, nous supposons que les éléments de notre lexique sont organisés au sein d'une structure complexe telle qu'une ontologie (ou plusieurs le cas échéant; voir [4] pour la définition précise d'une ontologie) ou taxonomies. Les travaux de [1] et [12] donnent une idée de la structure hiérarchique que nous présupposons.

3.2 Modèle d'item lexical

Notre modèle de représentation des connaissances sémantiques est très proche de celui proposé par Pustejovsky. Nous en conservons les trois niveaux de description tout en modifiant légèrement leur structure interne. Les figures suivantes présentent des exemples de notre modèle lexical. La signification du contenu de nos structures LG est détaillée plus loin et les exemples prendront tout leurs sens après lecture des sous-sections suivantes. Les mots "couteau", "beurrer", "commencer" et "livre" sont modélisés comme illustré en figures 8 à 11. Ces quelques exemples exploitent la plupart de nos apports au LG. Les paragraphes suivants détaillent chacune de ces modifications.

3.2.1 Les statuts des variables

Ce que nous nommons statut d'une variable représente le rôle précis qu'elle occupe dans la sémantique du mot. Chaque variable possède ainsi une collection de statuts qui peuvent être de quatre formes :

- *self* indique que la variable dénote l'objet décrit par l'item.
- *dot* indique que la variable est la projection d'un des aspects pointés de l'objet.
- $\emptyset$ est le statut vide (qui s'avère utile pour définir les mécanismes de composition voir section 3.3).
- *R@T* où *R* représente le rôle thématique local que joue la variable vis-à-vis du trait *T* (parmi *formal*, *agentif*, *result* et *trigger*). Le rôle thématique local (par opposition aux rôles thématiques de Chomsky) *R* peut être agent, patient ou adjoint (selon les sens habituels de ces termes).

Les satuts peuvent avoir valeur de contrainte. Par exemple, afin de traiter le cas de verbes comme `finir`, on peut exiger qu'une variable présente dans la sous-structure REQ ne soit liable qu'à une

$$
\begin{bmatrix}
\texttt{couteau} & & & & & & & & \\
\text{ARGSTR} & \begin{bmatrix}
\text{ARG} & \begin{bmatrix} \begin{bmatrix} \textit{self} \\ \textit{patient@agentif} \\ \textit{agent@result} \\ \textit{patient@trigger} \end{bmatrix} & = & x & : & \textit{artefact} \end{bmatrix} \\
\text{D-ARG} & \begin{bmatrix} \begin{bmatrix} \textit{agent@agentif} \end{bmatrix} & = & y & : & \textit{personne} \\ \begin{bmatrix} \textit{agent@trigger} \end{bmatrix} & = & z & : & \textit{personne} \\ \begin{bmatrix} \textit{patient@result} \end{bmatrix} & = & w & : & \textit{matière} \end{bmatrix}
\end{bmatrix} \\
\text{EVENSTR} & \begin{bmatrix}
\text{E} & \begin{bmatrix} \begin{bmatrix} \textit{reify@formal} \end{bmatrix} & = & e_1 & : & \textit{état} \\ \begin{bmatrix} \textit{reify@agentif} \end{bmatrix} & = & e_2 & : & \textit{transition} \\ \begin{bmatrix} \textit{reify@result} \end{bmatrix} & = & e_3 & : & \textit{processus} \\ \begin{bmatrix} \textit{reify@trigger} \end{bmatrix} & = & e_4 & : & \textit{processus} \end{bmatrix} \\
\text{ARG-ST} & \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_w \end{bmatrix}
\end{bmatrix} \\
\text{QUALIA} & \begin{bmatrix}
\text{FORMAL} & = & \begin{bmatrix} \textit{couteau}(e_1, x) \end{bmatrix} \\
\text{AGENT} & = & \begin{bmatrix} \textit{fabriquer}(e_2, y, x) \end{bmatrix} \\
\text{TELIC} & = & \begin{bmatrix} \begin{bmatrix} \text{RESULT} & = & \textit{couper}(e_3, x, w) \\ \text{TRIGGER} & = & \textit{utiliser}(e_4, z, x) \end{bmatrix} \\ \vdots \end{bmatrix} \\
\text{CONST} & = & \begin{bmatrix} \text{PART-OF} & = & \begin{bmatrix} \texttt{lame} \\ \texttt{manche} \end{bmatrix} \\ \text{MADE-OF} & = & \begin{bmatrix} \texttt{acier} \end{bmatrix} \\ \vdots & = & \vdots \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 8: Notre représentation lexicale pour `couteau`

$$\left[\begin{array}{ll}
\texttt{beurrer} & \\
\text{ARGSTR} & \left[\begin{array}{ll}
\text{ARG} & \left[\text{REQ}\ \left[\begin{array}{l}\left[\ \textit{agent@agentif}\ \right] = x : \textit{humain} \\ \left[\ \textit{patient@agentif}\ \right] = y : \textit{objet_physique}\end{array}\right]\right] \\
\text{S-ARG} & \left[\begin{array}{l}\left[\begin{array}{c}\textit{adjoint@agentif} \\ \vdots\end{array}\right] = z : \textit{tartinable} \\ \text{DEFAULT} = \textit{beurre}\end{array}\right]
\end{array}\right] \\
\text{EVENSTR} & \left[\begin{array}{ll}
\text{E} & \left[\left[\begin{array}{c}\textit{self} \\ \textit{reify@agentif}\end{array}\right] = e : \textit{processus}\right] \\
\text{ARG-ST} & \left[\begin{array}{l}\varepsilon_x \\ \varepsilon_y \\ \varepsilon_z\end{array}\right]
\end{array}\right] \\
\text{QUALIA} & \left[\text{AGENT} = \left[\ \textit{beurrer}(e,x,y,z)\ \right]\right]
\end{array}\right]$$

Figure 9: Notre représentation lexicale pour `beurrer`

$$\left[\begin{array}{ll}
\texttt{commencer} & \\
\text{ARGSTR} & \left[\text{ARG}\ \left[\text{REQ}\ \left[\left[\ \textit{agent@agentif}\ \right] = x : \tau\ \right]\right]\right] \\
\text{EVENSTR} & \left[\begin{array}{ll}
\text{E} & \left[\begin{array}{ll}\text{REQ}\ \left[\left[\ \textit{patient@agentif}\ \right] = e : \xi\ \right] \\ \left[\left[\begin{array}{c}\textit{self} \\ \textit{reify@agentif}\end{array}\right] = e_1 : \pi\right]\end{array}\right] \\
\text{ARG-ST} & \left[\ \varepsilon_x\ \right]
\end{array}\right] \\
\text{QUALIA} & \left[\text{AGENT} = \left[\ \textit{commencer}(e_1,x,e)\ \right]\right]
\end{array}\right]$$

Figure 10: Notre représentation lexicale pour `commencer`

$$
\begin{bmatrix}
\texttt{livre} & & & & & & & \\
\text{ARGSTR} & \begin{bmatrix}
\text{ARG} & \begin{bmatrix}
\begin{bmatrix} \textit{self} \\ \textit{patient@agentif}_1 \\ \textit{patient@result} \end{bmatrix} & = & x & : & \textit{info} \bullet \textit{objet_physique} \\
\begin{bmatrix} \textit{self} \\ \textit{dot} \\ \textit{patient@agentif}_2 \end{bmatrix} & = & y & : & \textit{info} \\
\begin{bmatrix} \textit{self} \\ \textit{dot} \end{bmatrix} & = & z & : & \textit{objet_physique}
\end{bmatrix} \\
\text{D-ARG} & \begin{bmatrix}
\begin{bmatrix} \textit{agent@agentif}_1 \end{bmatrix} & = & w & : & \textit{personne} \\
\begin{bmatrix} \textit{agent@agentif}_2 \end{bmatrix} & = & u & : & \textit{personne} \\
\begin{bmatrix} \textit{agent@result} \end{bmatrix} & = & v & : & \textit{personne}
\end{bmatrix}
\end{bmatrix} \\
\text{EVENSTR} & \begin{bmatrix}
\text{E} & \begin{bmatrix}
\begin{bmatrix} \textit{reify@formal} \end{bmatrix} & = & e_1 & : & \textit{état} \\
\begin{bmatrix} \textit{reify@agentif}_1 \end{bmatrix} & = & e_2 & : & \textit{processus} \\
\begin{bmatrix} \textit{reify@agentif}_2 \end{bmatrix} & = & e_3 & : & \textit{processus} \\
\begin{bmatrix} \textit{reify@result} \end{bmatrix} & = & e_4 & : & \textit{processus}
\end{bmatrix} \\
\text{ARG-ST} & \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_w \\ \varepsilon_u \\ \varepsilon_v \end{bmatrix}
\end{bmatrix} \\
\text{QUALIA} & \begin{bmatrix}
\text{FORMAL} & = & \begin{bmatrix} \textit{livre}(e_1, x) \\ \text{PROJ} \quad = \begin{bmatrix} \textit{O_Elab}(y,x) \\ \textit{O_Elab}(z,x) \end{bmatrix} \end{bmatrix} \\
\text{AGENT} & = & \begin{bmatrix} Q_{A1} & \textit{écrire}(e_2, w, x) \\ Q_{A2} & \textit{relier}(e_3, u, y) \end{bmatrix} \\
\text{TELIC} & = & \begin{bmatrix} \begin{bmatrix} \text{RESULT} & = & \textit{lire}(e_4, v, x) \\ \text{TRIGGER} & = & \emptyset \end{bmatrix} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 11: Notre représentation lexicale pour `livre`

$$\begin{bmatrix} \texttt{couteau} & & & & & & \\ \text{ARG} & \begin{bmatrix} \begin{bmatrix} \textit{self} \\ \textit{patient@agentif} \\ \textit{agent@result} \\ \textit{patient@trigger} \end{bmatrix} & = & x & : & \textit{artefact} \end{bmatrix} \\ \text{D-ARG} & \begin{bmatrix} \begin{bmatrix} \textit{agent@agentif} \end{bmatrix} & = & y & : & \textit{personne} \\ \begin{bmatrix} \textit{agent@trigger} \end{bmatrix} & = & z & : & \textit{personne} \\ \begin{bmatrix} \textit{patient@result} \end{bmatrix} & = & w & : & \textit{matière} \end{bmatrix} \end{bmatrix}$$

Figure 12: Les statuts dans la structure argumentale de notre représentation de `couteau`

variable ayant un statut donné dans le contexte de sa propre structure LG. Cette information de contrainte n'est pas stockée dans les structures LG mais fait, selon nous, partie de la définition des règles de composition et des mécanismes génératifs. Ainsi, nous pensons pouvoir définir des classes de verbes, ou réutiliser des classification pré-existantes, en fonction des contraintes sur le statut de leurs arguments. Cette étude fait partie de nos perspectives de travail. La figure 12 présente l'utilisation des statuts.

L'information contenue dans les statuts des variables est partiellement redondante avec celle que fournit la structure de qualia. En effet, l'écriture $\textit{verbe}(e,x,y)$ donne des informations sur le rôle thématique local des variables e,x,y selon la définition du prédicat *verbe*. Si nous choisissons de préserver cette redondance, c'est dans un soucis de lisibilité de notre structure de représentation des items lexicaux. Cependant, il est intéressant de remarquer que les seules informations supplémentaires introduites dans les rôles formel, agentif et télique de la structure de qualia sont les prédicats représentant ces rôles. Nous pourrions donc modéliser "couteau" comme décrit en figure 13.

Or, comme nous l'expliquons en section 3.1.2, nous identifions les types sémantiques aux prédicats de notre langage de représentation. La structure de qualia exposée ci-dessus est donc une structure de traits typés comme cela est généralement admis bien que non-explicite. Cette vision des choses donne un sens à une structure d'héritage des structures de qualia ce qui permet d'imaginer des solutions pour l'établissement réel de l'ontologie que nous supposons en section 3.1.3.

3.2.2 Les arguments cachés

Les arguments cachés (stockés dans S-ARG) sont définis comme sémantiquement contenus dans la sémantique de l'item lexical. Le verbe "beurrer" est l'exemple typique de leur raison d'être.

Le prédicat *beurrer* que nous utilisons dans le trait agentif de notre exemple prend quatre arguments. Le premier est un événement et nous reviendrons sur sa fonction. La signification de l'expression

$$\textit{beurrer}(e,x,y,z)$$

est "x beurre y avec z". Notons qu'en sémantique de Montague, on se contentera généralement d'un prédicat binaire $\textit{beurre}(x,y)$ signifiant "x beurre y".

$$
\begin{bmatrix}
\texttt{couteau} & & & & & & & & \\
\text{ARGSTR} & \begin{bmatrix}
\text{ARG} & \begin{bmatrix} \begin{bmatrix} \textit{self} \\ \textit{patient@agentif} \\ \textit{agent@result} \\ \textit{patient@trigger} \end{bmatrix} & = & x & : & \textit{artefact} \end{bmatrix} \\
\text{D-ARG} & \begin{bmatrix} \begin{bmatrix} \textit{agent@agentif} \end{bmatrix} & = & y & : & \textit{personne} \\ \begin{bmatrix} \textit{agent@trigger} \end{bmatrix} & = & z & : & \textit{personne} \\ \begin{bmatrix} \textit{patient@result} \end{bmatrix} & = & w & : & \textit{matière} \end{bmatrix}
\end{bmatrix} \\
\text{EVENSTR} & \begin{bmatrix}
\text{E} & \begin{bmatrix} \begin{bmatrix} \textit{reify@formal} \end{bmatrix} & = & e_1 & : & \textit{état} \\ \begin{bmatrix} \textit{reify@agentif} \end{bmatrix} & = & e_2 & : & \textit{transition} \\ \begin{bmatrix} \textit{reify@result} \end{bmatrix} & = & e_3 & : & \textit{processus} \\ \begin{bmatrix} \textit{reify@trigger} \end{bmatrix} & = & e_4 & : & \textit{processus} \end{bmatrix} \\
\text{ARG-ST} & \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_w \end{bmatrix}
\end{bmatrix} \\
\text{QUALIA} & \begin{bmatrix}
\text{FORMAL} & = & \begin{bmatrix} \textit{couteau} \end{bmatrix} \\
\text{AGENT} & = & \begin{bmatrix} \textit{fabriquer} \end{bmatrix} \\
\text{TELIC} & = & \begin{bmatrix} \begin{bmatrix} \text{RESULT} & = & \textit{couper} \\ \text{TRIGGER} & = & \textit{utiliser} \end{bmatrix} \\ \vdots \end{bmatrix} \\
\text{CONST} & = & \begin{bmatrix} \text{PART-OF} & = & \begin{bmatrix} \texttt{lame} \\ \texttt{manche} \end{bmatrix} \\ \text{MADE-OF} & = & \begin{bmatrix} \texttt{acier} \end{bmatrix} \\ \vdots & = & \vdots \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 13: Représentation lexicale compacte pour `couteau`

$$\begin{bmatrix} \texttt{beurrer} & & & & & \\ \text{S-ARG} & \begin{bmatrix} \begin{bmatrix} \textit{adjoint@agentif} \\ \vdots \end{bmatrix} & = & z & : & \textit{tartinable} \\ \text{DEFAULT} & = & \textit{beurre} & & \end{bmatrix} \end{bmatrix}$$

Figure 14: Modélisation d'un argument caché du verbe `beurrer`

Selon Pustejovsky, l'argument *z* n'apparaît en surface (au niveau syntaxique) que s'il est effectivement plus spécifique que l'argument caché, ce qui permet d'éviter le pléonasme "Jean beurre sa tartine avec du beurre". Par exemple, dans la phrase "Jean beurre sa tartine avec du beurre demi-sel", une information supplémentaire apparaît et l'argument *z* ne représentera pas sa valeur par défaut mais l'objet désigné par "beurre demi-sel".

Cependant, nous pensons que la contrainte imposée classiquement est trop forte. Elle consiste en effet à typer l'argument caché par son type par défaut. Or il est possible qu'un type supérieur soit acceptable dans certains contextes. En revanche, il est interessant de garder l'information sur le type par défaut pour tous les cas où la variable n'est pas réalisée en surface.

C'est pourquoi nous dissocions contrainte de type et type par défaut. Ceci est illustré par la figure 14. En cas de non rélisation de l'argument caché, on considérera que le typage par défaut, et non la contrainte de type classique, est vérifié.

3.2.3 La sous-structure REQ

Certains éléments de la structure argumentale doivent être en quelque sorte instanciés, c'est-à-dire qu'ils seront nécessairement unifiés avec une variable présente dans la structure d'un autre mot et qu'ils participeront nécessairement à la sémantique finale. Il en va ainsi des arguments représentant le sujet d'un verbe, par exemple.

Nous pensons qu'il faut stocker de telles variables dans une sous-structure particulière afin de pouvoir déterminer simplement si un constituant est sémantiquement complet ou si certaines de ses variables sont encores insaturées. C'est pourquoi nous proposons d'utiliser une sous-structure REQ (pour "éléments requis") à l'intérieur des structures contenant les arguments propres (ARG) et les événements propres (E). En effet, ces variables requises participant nécessairement à la sémantique du mot, il est naturel qu'elle fasse partie de la classe des arguments ou événements propres. Cette sous-structure est présentée en figure 15.

3.2.4 Types pointés et projections

Pour des items nominaux de type pointé, le *rôle formel* est traditionnellement une relation entre arguments dénotant chacun des aspects de l'objet. Notre optique étant moins celle d'une description exhaustive que fonctionnelle de la strucutre interne des mots, cette relation a peu d'intérêt pour nous.

$$\begin{bmatrix} \texttt{commencer} & & \\ \text{ARGSTR} & \begin{bmatrix} \text{REQ} & \begin{bmatrix} \begin{bmatrix} \textit{agent@agentif} \end{bmatrix} & = & x & : & \tau \end{bmatrix} \end{bmatrix} \\ \text{EVENSTR} & \begin{bmatrix} \text{REQ} & \begin{bmatrix} \begin{bmatrix} \textit{patient@agentif} \end{bmatrix} & = & e & : & \xi \end{bmatrix} \\ & \begin{bmatrix} \begin{bmatrix} \textit{self} \\ \textit{reify@agentif} \end{bmatrix} & = & e_1 & : & \pi \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 15: Eléments requis dans les structures argumentale et événementielle de `commencer`

En effet, l'information dont nous souhaiterions disposer est le lien entre la variable dénotant l'objet et celles dénotant chacun de ses aspects.

Des travaux récents dans le LG ([8] par exemple) proposent de rendre compte de ces liens à l'aide d'une relation dîte d'*Object Elaboration*. La sémantique d'une telle relation n'est pas spécifiée. En revanche, elle permet de lier une variable syntaxiquement réalisée à une variable en quelque sorte virtuelle qui dénote un aspect particulier d'un objet. Ainsi, "john brûle un livre" se traduit en :

$$\begin{aligned} &\exists e_1, e_2, e_3, e_z, \exists p, \exists z, \exists x, \forall y, \\ &\quad [(\textit{john}(e_1, y) \leftrightarrow x = y) \wedge \textit{brûle}(e_2, x, z) \wedge \textit{O_Elab}(z, p) \wedge \textit{livre}(e_3, p) \wedge \textit{phys_obj}(e_z, z)] \end{aligned}$$

L'ordre des quantificateurs proposé ci-dessus n'en est qu'un parmi un grand nombre de valides.

Le traitement des types pointés que nous proposons permet aussi de modéliser des objets plus complexes comme `journal`, de type $(\textit{phys_obj} \bullet \textit{info}) \bullet \textit{organisation}$, sans utiliser d'opérateur $\bullet$ dans l'espace des variables logiques comme c'est classiquement le cas. Les variables virtuelles utilisées pour représenter un des aspects de l'objet de type complexe sont stockées en tant qu'arguments propres de l'item correspondant dans la mesure où elles dénotent en quelque sorte l'objet. En outre, ces variables sont repérées comme étant la projection de l'objet selon l'un de ses aspect par le statut *dot*. L'exemple canonique de la représentation de "livre" est repris en figure 11.

3.2.5 La réification systématique

La réification consiste en l'introduction pour chaque prédicat dénotant d'une situation d'une variable représentant précisément cette situation. Cet élément est sous la portée du prédicat (qui représente le rôle) et, par convention, il occupe la première position (cela n'a pas d'importance puisque les statuts gardent trace de cette information). D'un point de vue linguistique, ce premier argument représente la situation que dénote le prédicat lui-même. Ainsi, "x construit y" sera traduit par $\textit{construit}(e, x, y)$ (par opposition à la forme classique des grammaires catégorielles $\textit{construit}(x, y)$) et une propriété de l'action de construire sera exprimée par une prédication sur e. C'est-à-dire que "x construit y rapidement" est traduit en $\textit{construit}(e, x, y) \wedge \textit{rapidement}(e)$ (au lieu de $\textit{rapidement}\big(\textit{construit}(x, y)\big)$). Cette écriture est utilisée dans la modélisation des rôles de qualia pour les exemple des figures 8 à 13.

Outre la réification des prédicats représentant des verbes et des adjectifs, nous proposons d'adopter la même écriture pour tous les autres prédicats. En particulier, pour les contraintes de type, l'expression

$$\begin{bmatrix} \texttt{couteau} & & & \\ \text{TELIC} & = & \begin{bmatrix} \begin{bmatrix} \text{RESULT} & = & \mathit{couper}(e_3,x,w) \\ \text{TRIGGER} & = & \mathit{utiliser}(e_4,z,x) \end{bmatrix} \\ \vdots \end{bmatrix} \end{bmatrix}$$

Figure 16: Structure interne du télique de `couteau`

$a : \alpha$ se traduit en $\exists e, \alpha(e,a)$ au lieu de $\alpha(a)$. En effet, la réification permet d'éviter d'utiliser une logique d'ordre supérieur.

Une application possible de cette réification est le traitement de la temporalité (en utilisant des relations temporelles sur les évènements plutôt que des opérateurs temporels) ou des phénomènes dits de désactivation[2](comme dans la phrase "le prisonnier s'est évadé" qui dépossède le prisonnier de sa nature même de prisonnier). Ces traitements sur la séquence des événements peuvent aussi être traités en dehors de la forme logique associée à un fragment de texte. On comprend alors l'intérêt de manipuler des objets simples et comparables plutôt que des prédicats ou expressions logiques de structures parfois très différentes.

3.2.6 La structure interne du télique

La représentation du rôle télique que nous avons choisi d'adopter possède une sous-structure interne. En effet, il n'est pas rare que les descriptions de phénomènes linguistiques à l'aide du LG fassent intervenir les notions de rôle formel ou agentif du télique. Ces dénominations nous semblent trompeuses dans la mesure où elles suggèrent que le rôle télique est lui-même constitué d'une structure de qualia. Or les deux rôles évoqués précédemment peuvent être compris comme le résultat (que nous appelons RESULT) et le déclencheur (noté TRIGGER) du télique. Cette distinction permet du rendre compte du fait qu'un "couteau" posé sur une table est certes fait pour "couper" mais ne coupe pas.

Toutes les manipulations usuellement basées sur le rôle télique doivent désormais s'intéresser au résultat du télique (RESULT). C'est en effet ce trait qui exprime réellement le télique de l'item, le déclencheur n'étant qu'une condition d'existence effective du télique. Cette structure interne est illustrée par la figure 16.

3.2.7 Le développement de la structure d'héritage

L'ontologie que nous supposions en section 3.1.3 permet de définr des liens d'héritage des structures de qualia. Dans la version classique du LG, ces liens permettent d'expliquer pourquoi un même item peut assumer plusieurs rôles agentifs ou téliques. Ceci est important afin d'interpréter correctement des expressions comme "finir son café". La représentation simplifiée de "café" dans le LG classique est telle qu'illustrée en figure 17. Sans relation d'héritage, "Jean finit son café" risque d'être

[2]Désactivation est la traduction proposée par Christian Bassac du mécanisme nommé *gating* par Pustejovsky

$$\begin{bmatrix} \texttt{café} & \\ \text{ARGSTR} & \begin{bmatrix} \text{ARG}_1 & = & x : \textit{liquide} \\ \text{D-ARG}_1 & = & y : \textit{humain} \end{bmatrix} \\ \text{EVENSTR} & \begin{bmatrix} \text{D-E}_1 & = & e_1 : \textit{transition} \end{bmatrix} \\ \text{QUALIA} & \begin{bmatrix} \text{FORMAL} & = & x \\ \text{TELIC} & = & \textit{réveiller}(e_1, x, y) \end{bmatrix} \end{bmatrix}$$

Figure 17: Représentation LG classique simplifiée pour `café`

interprété comme “Jean fini de se réveiller”. Or ce n’est pas ce que nous attendons. Cependant, le “café” est un cas particulier d’une “boisson” (ou “liquide_potable” selon l’ontologie) dont le rôle télique est précisemment d’être bu.

Afin d’éviter d’avoir à définir les règles d’héritage des structures de qualia, nous préférons développer au sein de la structure interne d’un mot tous les rôles qu’il peut assumer. Cette méthode radicale peut engendrer un grand nombre d’analyses pour une même phrase et ne règle pas le problème du café présenté ci-dessus. Pour cette raison, nous envisageons la possibilité d’ordonner les différents rôles d’un même trait de la structure de qualia (ordonner les rôles agentif, ordonner les rôles téliques...) ou de leur attribuer des scores liés à leur fréquence d’expression dans le langage. Ce dernier objectif n’est actuellement qu’une perspective d’étude.

3.2.8 Le rôle constitutif

D’un point de vue purement fonctionnel, le rôle constitutif sert à faire référence à des items externes. En effet, seules les relations tout/partie ou celles spécifiant la substance figurent dans ce rôle. Car les informations portant sur le poids, la taille ou la forme d’un objet sont généralement contenue dans le *rôle formel*. Or il est peu pratique de conserver une écriture logique de ces relations.

En effet, le formalisme LG classique stocke dans le rôle constitutif des expressions analogues à :

$$\textit{part_of}(x,y) \wedge \textit{made_of}(y,z)$$

Les variables figurant dans ces relations apparaîssent nécessairement dans la structure argumentale de l’item. Par conséquent, la forme classique du rôle constitutif ne permet pas d’établir des relations entre items mais seulement entre certains aspects de ces items.

Nous pensons que cette limitation est trop restrictive et c’est pourquoi nous proposons de modéliser le rôle constitutif d’un item par de simples listes d’items (c’est-à-dire d’entrées lexicales dans notre cas) vis-à-vis desquels il entretient une certaine relation. Nous ignorons actuellement s’il est nécessaire de distinguer plusieurs relations possibles avec d’autres items comme nous l’avons fait dans l’exemple (figure 8) de la structure du mot “couteau”. Car s’il peut être utile de faire la différence entre la relation qui lie le “couteau” à de “l’acier” et celle qui le lie à la “lame” et au “manche” en termes de description, c’est possiblement inutile d’un point de vue fonctionnel. Il semble que ce rôle permette de traiter les cas de reconstruction métonymique comme celui de la

phrase “Mon vélo est crevé” où l’on signifie que la “chambre à air” de la “roue” de mon “vélo” est à crevée. Si c’est là le seul phénomène linguistique qui exploite le rôle constitutif, on peut se contenter d’y stocker les items vis-à-vis desquels une reconstruction métonymique est possible.

3.3 Règles de composition

L’approche classique du LG consiste à considérer que la composition de deux constituants se fait selon quatre mécanismes distincts : un mécanisme d’application de fonction simple (pendant de l’application d’un λ-terme à un autre en sémantique de Montague) et les trois mécanismes génératifs présentés en section 2.2.2. Dans le cadre de notre travail, nous ne tenons pas compte du mécanisme de liage sélectif car il ne concerne que les adjectifs dont l’étude est un sujet à part entière.

Quoi qu’il en soit, nous proposons de rompre avec cette vision de la composition et de ne présenter qu’un seul mécanisme général pour la composition. Les mécanismes génératifs du LG classique interviennent au sein de ce notre mécanisme global mais ne redéfinissent pas les règles d’application de fonction comme proposé dans [11].

Le mécanisme de composition que nous présentons ici traite exclusivement des cas pouvant se ramener à une saturation de variable insaturée. C’est-à-dire que nous n’abordons pas la question de la composition des nominaux (noms composés) et que nous nous situons toujours dans un paradigme verbal/nominal (les verbaux sont les seuls constituants fonctionnels du fragment du langage que nous étudions).

Dans ce qui va suivre, nous présentons des algorithmes pour les règles de composition. Ils sont écrits à l’aide de fonctions détaillées ci-après :

- **Remarques**

  · Typographie : les *Fonctions* sont en italique et commencent par une majuscule, les *variables* sont en italique, les CONSTANTES sont en petites majuscules.

  · Les types (au sens informatique du terme) des variables utilisées ne sont pas déclarés et sont implicites.

  · Lors de la première apparition d’une variable, on suppose que l’objet est implicitement créé.

  · Les textes contenus entre /* et */ sont des commentaires.

- **Opérateurs**

  · $\in$ vérifie l’appartenance d’un objet à un vecteur.

  · $\bigcup$ signifie l’union de deux ensembles/vecteurs.

  · $\preceq$ représente la relation de précédence sur les types sémantiques (inclusion au sens de la hiérarchie de types).

  · $\neg$ indique la négation d’une expression logique.

- **Fonctions**

  · *Addvar* ajoute une variable à une structure LG. Prend une structure LG comme premier argument, une sous-structure de la structure argumentale ou événementielle comme deuxième argument, un identifiant de variable comme troisième argument, un type sémantique comme quatrième argument et un vecteur de statuts comme dernier argument. Une variable du nom donné est alors ajoutée dans la sous-structure spécifiée de la structure LG fournie. Les types et statuts de cette variables sont ceux spéciiés en entrée.

  · *Agentive* renvoie le contenu du trait agentif d'une structure de qualia. Prend une structure LG comme unique argument.

  · *Argstr* renvoie un vecteur contenant les variables de la structure argumentale d'une structure LG. Prend une structure LG comme unique argument.

  · *Const* renvoie un vecteur contenant les éléments stockés (labels de structure LG) dans le trait constitutif de la structure de qualia d'une structure LG. Prend une structure LG comme unique argument.

  · *Evenstr* renvoie un vecteur contenant les variables de la structure événementielle d'une structure LG. Prend une structure LG comme unique argument.

  · *Formal* renvoie le contenu du trait formel d'une structure de qualia. Prend une structure LG comme unique argument.

  · *Qualia* renvoie un vecteur contenant les noms des traits de la structure de qualia (parmi les rôles formel, agentif et les deux sous-traits du télique, à savoir result et trigger) d'une structure LG. Prend une structure LG comme unique argument.

  · *Module_select* sélectionne le module de construction à utiliser selon le mode de sélection choisi et, le cas échéant, d'autres critères reposant sur la distance sémantique entre les traits des structures de qualia des constituants en présence.

  · *New_var_name* génère un nouveau nom de variable (aléatoirement).

  · *Remove_last* supprime et lit le dernier élément d'un vecteur. Prend un vecteur comme unique argument.

  · *Result* renvoie le contenu du trait result d'une structure de qualia. Prend une structure LG comme unique argument.

  · *Statuses* renvoie un vecteur contenant les statuts d'une variable donnée. Prend une variable comme unique argument.

  · *Str_completion* complète les structures argumentale et événementielle d'une structure LG de sorte que toutes les variables apparaissant dans les traits de la structure de qualia y figurent. Prend une structure LG comme unique argument.

  · *Type* renvoie le type sémantique d'une variable donnée. Prend une variable comme unique argument.

  · *Unify* réalise l'unification de deux variables. Prend deux variables en entrée.

  - *Value* retourne la valeur stockée dans un trait de structure de qualia. Prend un trait d'une structure de qualia comme unique argument.

- **Autres procédures**

  - add... to... ajoute un objet à la fin d'un vecteur.
  - add_conj... to... ajoute une expression logique en conjonction d'une autre.
  - set_value_of... to... permet de stocker une valeur dans un trait d'une structure de qualia.
  - write... in ... stocke une chaine de caractères dans une variable.

- **Définition :** STATUSES est un vecteur contenant tous les rôles thématiques locaux possible pour une variable. Dans notre cas, STATUSES = [AGENT,PATIENT,ADJOINT].

3.3.1 Mécanisme général

De manière générale, la composition de deux constituants se fait suivant deux phases distinctes :

- Une phase de sélection;
- Une phase de construction du résultat.

Selection : La phase de sélection consiste en l'établissement d'une correspondance entre une variable insaturée du constituant jouant le rôle de fonction et un argument (ou un événement) de la structure interne de l'autre constituant. C'est lors de cette phase qu'interviennent les mécanismes de coercion. Le mécanisme de *subtype coercion* peut être déclenché en fin de sélection, une fois un argument choisi, afin de contraindre le type de celui-ci. Au contraire, le mécanisme de *true complement coercion* influe sur le choix de l'argument et se déclenche donc au début de la phase de sélection.

Notre hypothèse au début de cette phase est qu'une unique variable insaturée (stockée dans REQ) du constituant verbal est désignée par l'analyse syntaxique comme devant être saturée par la composition en cours. Nous présentons en table 1 l'algorithme décrivant la procédure de sélection intervenant au début de la composition entre un constituant fonctionnel *func* et un constituant *obj* lorsque la variable *x* est pointée par la syntaxe comme devant être saturée. La variable *output* contient les modes de sélection possibles pour cette composition, c'est-à-dire, dans notre cas, une collection de triplets. Le premier élément de ces triplets contient une information sur la nature de la sélection effectuée (classique, expression d'un trait de qualia, référence métonymique...); le deuxième contient la variable sélectionnée; le troisième contient le nom de la structure LG avec laquelle le constituant fonctionnel s'est effectivement composé (différent de *obj* en cas de reconstruction métonymique). Dans un premier temps, cet algorithme effectue un parcours des structures argumentale et événementielle du constituant "objet". Ce parcours est tel qu'il permet de trier les variables dénotant l'objet. Un test sur les types des variables est alors lancé (opérateur $\preceq$). Ce test représente le mécanisme de *subtype coercion*. Si le test est vérifié, une première sélection

Table 1: Algorithme de l'opération de sélection

```
Procedure : Selection(func, x, obj)
  output = [ ]
  for all y ∈ (Argstr(obj) ⋃ Evenstr(obj)) do
    /* parcours des structures argumentale et événementielle */
    if SELF ∈ Statuses(y) then
      /*permet de trier les variables dénotant l'objet */
      if Type(y) ⪯ Type(x) then
        /* ce test représente le mécanisme de subtype coercion */
        add [ REGULAR , Name(y) , obj ] to output
      end if
    end if
  end for
  for all e ∈ Evenstr(obj) do
    /* parcours de la structure événementielle */
    for all quale ∈ Qualia(obj) do
      write REIFY"@"quale in status
      if status ∈ Statuses(e) then
        /* ce parcours correspond à la true complement coercion */
        if Type(e) ⪯ Type(x) then
          add [quale, Name(e), obj] to output
        end if
      end if
    end for
  end for
  if output == [ ] then
    for all link ∈ Const(obj) do
      /* parcours les références possible depuis le mot courant */
      for all triple ∈ Selection(func, x, link) do
        write ("⟨" obj "⟩" triple[1]) in string
        triple[1] = string
      end for
    end for
  end if
  if output == [ ] then
    return ERROR
  end if
  return output
```

valide est trouvée et la variable *output* est complété. La constante REGULAR désigne ici le mode de sélection classique (variable dénotant l'objet). On peut choisir d'insérer à ce point de l'algorithme un moyen de sortie afin de limiter le nombre de sélections possibles. Par exemple, on peut retourner *output* s'il est non vide à ce point et quitter la fonction. Dans un deuxième temps, l'algorithme effectue un parcours de la structure événementielle du constituant objet afin de sélectionner tous les évènements issus de la réification d'une situation décrivant un trait de qualia. Ce parcours de la structure événementielle correspond à la *true complement coercion*. Un test sur les types des variables est alors lancé (opérateur $\preceq$). Ce test représente le mécanisme de *subtype coercion*. Si le test est vérifié, *output* est complétée et l'information concernant le trait effectivement sélectionné est conservée comme mode de sélection (ici *quale*). Il nous semble pertinent de ne nous intéresser à la reconstruction métonymique que lorsque tous les autres modes de sélection ont échoué. C'est pourquoi l'algorithme se termine si *output* est non vide avant de passer au traitement de la reconstruction métonymique. Ce traitement consiste en le parcours des références possibles depuis le mot courant (rôle constitutif) puis en le lancement récursif de la procédure de sélection sur chaque item présent dans le rôle constitutif. Le mode de sélection est construit de telle sorte qu'il permet de garder trace de la chaine de reférence intervenant dans la reconstruction métonymique.

Il est possible qu'il faille limiter le nombre d'exécutions récursives de la procédure *Selection* car la reconstruction métonymique que suggère le recours au rôle constitutif de l'item pour permettre la sélection n'est peut être plus compréhensible dans des cas réels au delà d'un certain nombre de références. Par exemple, le mot "vélo" peut désigner la "roue" du "vélo" mais plus rarement la "tige filetée" de la "valve" de la "chambre à air". Nous pensons que la détermination du nombre de références autorisées doit faire l'objet d'une étude linguistique ultérieure et nous nous contenterons de considérer pour le moment qu'il n'existe pas de telle limite.

Il est important de noter que nous choisissons d'exploiter les informations contenues dans le rôle constitutif d'un item seulement si les autres méthodes de sélection ont échoué. En effet, il nous semble que lorsqu'il y a reconstruction métonymique aucune interprétation plus locale (c'est-à-dire exploitant exclusivement les variables contenues dans la structure des items effectivement présents) n'est sémantiquement acceptable.

Comme nous l'avons expliqué au début de cette section, le mécanisme que nous proposons inclut les mécanismes génératifs. Ainsi, les tests exploitant la relation $\preceq$ sont des utilisations implicites de *subtype coercion*, tandis que la possibilité de parcourir des variables issues de la structure argumentale aussi bien que de la structure événementielle relève du *true complement coercion*.

Construction : La phase que nous nommons de construction du résultat est celle durant laquelle ont lieu ce que PUSTEJOVSKY nomme unifications de structure de qualia. C'est lors de cette phase que peut se déclencher la *co-composition* en modifiant la façon dont l'unification est faite. Cette phase consiste en la complétion des traits de la structure de qualia couvrant la variable nouvellement saturée par des informations issues de la structure de qualia de l'item.

De par la nature multiple des mécanismes de la co-composition et la diversité des phénomènes linguistiques qu'ils traitent, il ne nous paraît pas pertinent de présenter une unique procédure de construction du résultat. L'algorithme général pour l'opération de construction est décrit en table 2. Dans cet algorithme, *func* est la structure LG du constituant fonctionnel, *x* est la variable de

Table 2: Algorithme de l'opération de construction

```
Procedure : Construction(func, x, selection_output)
  /* func est la structure LG du constituant fonctionnel */
  /* x est la variable de func sélectionnée par l'analyse syntaxique pour être saturée */
  /* selection_output est le résultat de la procédure Selection */
  /* selection_output est donc un vecteur de triplets */
  output = [ ]
  /* cette variable output contient les constructions valides pour cette composition */
  for all selected ∈ selection_output do
    /* considère chaque sélection possible */
    case = Module_select(func, x, selected)
    /* cette variable case désigne un cas de figure correspondant à un module de con-
    struction donné */
    if case == MODULE1_CASE then
      add Module1(func, x, selected) to output
    else
      if case == MODULE2_CASE then
        add Module2(func, x, selected) to output
      end if
    else
      if case == MODULE3_CASE then
        add Module3(func, x, selected) to output
      end if
      ...
    end if
  end for
  return output
```

func sélectionnée par l'analyse syntaxique pour être saturée, et *selection_output* est le résultat de la procédure *Selection*. La variable *selection_output* est donc un vecteur de triplets. On stocke dans une variable *output* les constructions valides pour cette composition. La procédure *Module_select* permet de détecter le cadre linguistique (du point de vue des phénomènes intervenant) au sein duquel a lieu la composition. Cela permet de choisir un module de construction proposant un traitement approprié de l'unification des structures de qualia. Nous présentons par la suite quelques modules de construction.

3.3.2 Module 1 : sans co-composition

Ce premier module décrit la procédure d'unification hors mécanismes de co-composition. Cette procédure rend compte de compositions simples comme celle intervenant dans l'expression "aimer marie".

Nous faisons ici deux suppositions :

1. La structure du constituant résultant de la composition est une copie de la structure du constituant fonctionnel à laquelle on fait subir des modifications minimales.

2. Seuls les rôle du constituant objet couvrant la variable selectionnée doivent être conservés.

L'algorithme décrivant la procédure est détaillé en table 3. Ici encore, *func* est la structure LG du constituant fonctionnel, *x* est la variable de *func* sélectionnée par l'analyse syntaxique pour être saturée et *selected* est un triplet toujours de la forme $[\text{REGULAR}, y, obj]$ quand ce module est sélectionné (il revient à la procédure *Module_select* de faire en sorte que ce soit le cas). Dans ce triplet, REGULAR est un simple marqueur qualifiant le type de sélection effectif (ici sélection classique), *y* est la variable sélectionnée pour saturer *x* et *obj* est la structure LG du constituant non fonctionnel intervenant dans la composition (l'objet avec lequel le constituant fonctionnel se compose effectivement). La variable *func_obj* contient le résultat de la composition. Conformément à la première des deux suppositions faites ci-dessus, ce résultat est une copie de la structure du constituant fonctionnelle à laquelle on apporte quelques modifications issues de l'autre constituant. Ces modifications sont stockées dans la variable *string*. Elles complètent la structure interne du constituant fonctionnel suite à la composition. L'algorithme sélectionne tous les traits du constituant *obj* contenant la variable sélectionnée. Les informations contenues dans chacun de ces traits sont stockées dans *string*. Puis chaque trait de la structure *func_obj* contenant la variable saturée (notée *x*) est modifié par l'adjonction des informations contenues dans *string*. La variable nouvellement saturée (ici *x*) et la variable sélectionnée (ici *y*) sont ensuite unifiées.

Nous ne détaillons pas le fonctionnement de la procédure *Str_completion* utilisée ci-dessus. En effet, cette procédure consiste simplement en la complétion des structures argumentale et événementielle de `Func_Obj` afin qu'y figurent toutes les variables utilisées dans la structure de qualia. Le renommage des variable en cas de conflit est lui aussi pris en charge par cette procédure. Notons que les variables ainsi ajoutées à la nouvelle structure se voient attribuer le statut `0` comme unique statut indépendemment des statuts qu'elles possédaient au sein de leur structure initiale.

Nous choisissons de conserver systématiquement le rôle formel de l'item car il contient les informations les plus spécifiques concernant la variable dénotant l'objet décrit. Se priver de cette

Table 3: Algorithme du module de construction sans co-composition

```
Procedure : Module1(func, x, selected)
  func_obj = func
  string = Formal(selected[3])
  for all quale ∈ Qualia(selected[3]) do
    for all θ ∈ STATUSES do
      write θ"@" quale in status
      /* les quelques traitements précédents permettent de construire un critère de
      sélection des traits contenant la variable sélectionnée */
      if status ∈ Statuses(selected[2]) then
        add_conj Value(quale) to string
      end if
    end for
  end for
  for all quale ∈ Qualia(func_obj) do
    for all θ ∈ STATUSES do
      write θ"@"quale in status
      /* les quelques traitements précédents permettent de construire un critère de
      sélection des traits contenant la variable saturée */
      if status ∈ Statuses(x) then
        quale_value = Value(quale)
        add_conj string to quale_value
        set_value_of quale to quale_value
        /* les traits couvrant la variable saturée sont complétés par les informations
        concernant la variable sélectionnée */
      end if
    end for
  end for
  Unify(x, y)
  /* la variable saturée et la variable sélectionnée sont unifiées */
  Str_completion(func_obj)
  return func_obj
```

information conduit à utiliser la contrainte de typage comme propriété la plus spécifique (au sens de la hiérarchie de types) ce qui ne nous semble pas satisfaisant.

3.3.3 Module 2 : Reconstruction métonymique

La reconstruction métonymique n'est pas expliquée par les mécanisme génératifs usuels. Nous avons proposé en section 3.3.1 un mécanisme de sélection prenant en compte la possibilité de traiter un tel phénomène linguistique et nous détaillons ici le mécanisme de construction correspondant. L'algorithme de la table 4 décrit son fonctionnemment. Comme précédemment, *func* est la structure LG du constituant fonctionnel et x est la variable de *func* sélectionnée par l'analyse syntaxique pour être saturée. La variable *selected* est toujours de la forme $[\langle obj \rangle string, y, obj]$ quand ce module est sélectionné. Dans l'expression du mode de sélection, *string* est un chaîne de caractère quelconque. La variable *metonymic_link* permet de garder trace du lien métonymique entre l'objet nommé dans la phrase et celui qui est effectivement désigné. L'algorithme permet de *remonter* la chaine de références jusqu'à l'objet effectivement désigné. La construction en elle-même est alors effectuée avec ce dernier élément de la chaine de références, quelque soit le mode de sélection qui lui est propre. La structure LG résultant de cette construction est ensuite complétée de manière à exprimer avec des prédicats la chaine de références entre objets. Ceci est réalisé par la sélection dans chacun des objets concernés d'une variable dénotant cet objet afin de donner un sens à des expressions du type $part_of(x,y)$. Ces variables sont ajoutées au fur et à mesure dans la structure argumentale ou événementielle de *func_obj*.

3.3.4 Le cas de la préposition "avec"

Nous considérons qu'il n'existe pas de structure LG pouvant représenter une préposition ou un article. De notre point de vue, ces entités linguistiques doivent être modélisées par des opérateurs sur les structures LG. Nous proposons ici une description du comportement de l'opérateur modélisant la sémantique de la préposition "avec".

Syntaxiquement parlant, "avec" peut assumer deux rôles distincts :

- introduction d'un modificateur nominal (type syntaxique $N \rightarrow N$);
- introduction d'un modificateur verbal (type syntaxique $VP \rightarrow VP$).

A chacun de ces rôles, nous associons un opérateur.

Opérateur associé à la préposition `avec` rattachée à un nominal

Lors de son application à une structure LG, l'opérateur associé à la préposition `avec` déclenche les modifications décrites en table 5. Dans cette description, *noun* est la structure LG du nominal spécifié par le PP.

Il semblerait que la sous-structure RESTR puisse être complétée par l'adjonction de $e \circ_{\propto} e_{new}$ où e est de statut *reify@formel*.

Le prédicat P_τ est déterminé en fonction de τ. En effet, utiliser un unique prédicat *avec* de type $\top \rightarrow \top \rightarrow t$ (l'événement résultant de la réification du prédicat ne figure pas dans ce typage) ne permet pas d'exclure des expressions comme "un nuage avec un couteau". Nous ne connaissons

Table 4: Algorithme du module de construction pour la métonymie

```
Procedure : Module2(func, x, selected)
  tmp = selected
  metonymic_link = [ ]
  while tmp[1] matches ("⟨"var1"⟩"var2) do
    add var1 to metonymic_link
    tmp[1] = var2
    /* on remonte la chaine de références */
  end while
  func_obj = Construction(func, x, tmp)
  /* la construction est effectuée avec le dernier élément de la chaine de références */
  var = Name(x)
  repeat
    /* cette boucle permet d'exprimer avec des prédicats la chaine de références entre
    objets */
    last = Remove_last(metonymic_link)
    for all y ∈ Argstr(last) do
      if (SELF ∈ Statuses(y)) & ¬(DOT ∈ Statuses(y)) then
        z = New_var_name()
        if Type(y) ≼ EVENT then
          Addvar(func_obj, EVENSTR, z, Type(y), [∅])
        else
          Addvar(func_obj, D-ARG, z, Type(y), [∅])
        end if
        for all quale ∈ Qualia(func_obj) do
          for all θ ∈ STATUSES do
            write θ"@"quale in status
            if status ∈ Statuses(x) then
              write "partof("var","z")" in string
              quale_value = Value(quale)
              add_conj string to quale_value
              set_value_of quale to quale_value
              var = z
            end if
          end for
        end for
      end if
    end for
  until metonymic_link == [ ]
  return func_obj
```

Table 5: Algorithme de l'opérateur associé à la préposition "avec" rattachée à un nominal

Opérateur : $Avec_nom(noun)$

/* *noun* est la structure LG du nominal spécifié par le PP */
copy *noun* in *noun_avec*
for all $x \in Argstr(noun_avec)$ **do**
  if SELF $\in Statuses(x)$ **then**
    $e = New_var_name()$
    $Addvar(noun_avec, \text{D-E}, e, \text{STATE}, [\emptyset])$
    $y = New_var_name()$
    $Addvar(noun_avec, \text{ARGSTR.REQ}, y, \tau, [\emptyset])$
    /* la valeur de τ est encore inconnue */
    $quale_value = Formal(noun_avec)$
    write "P_τ("$Name(e)$","$Name(x)$","$Name(y)$")" in *string*
    add_conj *string* to *quale_value*
    set_value_of *Formal*(*noun_avec*) to *quale_value*
    return *noun_avec*
  end if
end for

pas à l'heure actuelle le mécanisme déterminant P_τ. De même, nous ne sommes pas encore à même d'expliciter le type τ.

La structure résultant de l'application de cet opérateur à la structure d'un nominal a un comportement fonctionnel puiqu'une de ses variables est insaturée. La définition du module approprié au traitement de la composition de cette structure avec celle du nominal introduit par la préposition n'est pas donnée ici car nous pensons qu'elle est intimement liée aux mécanismes de détermination de τ (et de P_τ).

Opérateur associé à la préposition `avec` rattachée à un verbal

Lors de son application à une structure LG, l'opérateur associé à la préposition `avec` déclenche les modifications décrites en table 6. Dans l'algorithme, *verb* désigne la structure LG du verbal spécifié par le PP.

Comme dans le cas précédent, la sous-structure RESTR peut être complétée par l'adjonction de la relation $e \circ_\infty e_{new}$ où e est de statut *reify@agentive*. Ici encore, nous ne savons déterminer τ ni P_τ. En revanche, nous pensons que le prédicat réifié par la variable de statut *reify@trigger* du nominal introduit par la préposition est un bon candidat pour remplacer P_τ. Si, comme pour l'autre lecture syntaxique de "avec", nous ne proposons pas de module de construction complet pour ce cas de figure, nous pouvons néanmoins en donner une condition d'application tel que décrit dans l'algorithme de la table 7. Cette condition se base sur la distance sémantique d (voir section 3.1.2) et représente la necessité qu'il existe une communauté de sens entre l'action décrite par le verbal et celle décrite dans le télique du nominal.

Table 6: Algorithme de l'opérateur associé à la préposition "avec" rattachée à un verbal

Opérateur : *Avec_verb*(*verb*)

/* *verb* est la structure LG du verbal spécifié par le PP */
copy *verb* in *verb_avec*
for all $x \in Argstr(verb)$ **do**
  if $\big(\text{AGENT@AGENTIVE} \in Statuses(x)\big) \& \big(\text{SELF} \in Statuses(x)\big)$ **then**
    $e = New_var_name()$
    $Addvar(verb_avec, \text{D-E}, e, \text{PROCESS}, [\emptyset])$
    $y = New_var_name()$
    $Addvar(verb_avec, \text{ARGSTR.REQ}, y, \tau, [\emptyset])$
    /* la valeur de τ est encore inconnue */
    $quale_value = Agent(verb_avec)$
    write "P_τ("$Name(e)$","$Name(x)$","$Name(y)$")" in *string*
    add_conj *string* to *quale_value*
    set_value_of AGENTIVE(*verb_avec* to *quale_value*
    return *verb_avec*
  end if
end for

Table 7: Algorithme partiel de construction pour un cas de rattachement prépositionnel

Procedure : *Module3*(*func*, *x*, *selected*)

/* *func* est la structure LG du constituant fonctionnel */
/* *x* est la variable de *func* sélectionnée par l'analyse syntaxique pour être saturée */
/* *selected* est toujours de la forme $[P, y, obj]$ quand ce module est sélectionné */
/* où *P* est une chaine de caractères */
if $d\big(Result(selected[3]), Agentive(func)\big) = \infty$ **then**
  /* le test précédent porte sur la distance sémantique entre deux expressions */
  return ERROR
else
  return $func_obj = Module3'(func, x, selected)$
  /* nous ignorons le fonctionnement de $Module3'$ */
end if

4 Evaluation des résultats

4.1 Problèmes couverts

Notre travail apporte un modèle de représentation des connaissances sémantiques conforme à la version classique du LG. Par conséquent, l'ensemble des questions liguistiques traitées par le LG reste abordable par l'utilisation de nos propositions.

En particulier, nous traitons efficacement les cas de composition simples correspondant à un niveau de richesse sémantique au moins équivalent à celui obtenu par les méthodes présentées en section 2.1. En effet, hors mécanisme de co-composition, nous fournissons un raffinement de la sémantique de Montague permettant de détecter des phrases sémantiquement invalides par le truchement de notre mécanisme de sélection.

En ce qui concerne une sémantique plus profonde, nous proposons avec ce travail un traitement original de la reconstruction métonymique qui, bien qu'incomplet, permet d'étendre le champ d'application (en termes de phénomènes linguistiques décrits) du LG. Notre approche des prépositions par l'usage d'opérateurs plutôt que de structures LG nous permet de proposer le traitement formel de ces objets qui était implicitement appliqué dans les travaux antérieurs.

4.2 Problèmes en suspens

Le traitement de la co-composition que nous ne faisons qu'ébaucher ici reste à faire. Nous ne sommes pas à l'heure actuelle capables de définir avec précision le fonctionnement des mécanismes rendant compte de la co-spécification qui est pourtant le cas de co-composition le plus élémentaire. Cette lacune est d'autant plus handicapante que l'intérêt principal du LG réside dans ce mécanisme génératif complexe.

La question de l'articulation de notre travail avec un outil d'analyse syntaxique approprié reste elle aussi en suspens. Pour cette raison, il ne nous a pas été possible de proposer une implémentation de notre production. Cette perspective reste néanmoins prioritaire.

5 Perspectives et conclusion

Le but de ce travail était de proposer une reformulation du *Lexique Génératif* plus utilisable dans le cadre d'une analyse automatique du langage. Nous avons, dans cette optique, proposé une reformalisation du travail de [11] à laquelle nous avons adjont quelques apports originaux (notion de *distance sémantique*, traitement de la reconstruction métonymique...). Notre volonté première en ce qui concerne cet outil est d'en réaliser une implémentation afin de tester sa validité pratique.

Ce travail se poursuivra par une étude plus poussée des classes de problèmes couverts par la co-composition afin de proposer une collection plus vaste de modules de construction. Nous pensons aussi travailler à la détermination d'un outil d'analyse syntaxique respectant les hypothèses que nous avons faites à ce sujet (en particulier en ce qui concerne le traitement de la quantification).

References

[1] BOGURAEV B., PUSTEJOVSKY J. (1993), Lexical knowledge representation and natural language processing

[2] BOUILLON P. (1997), Polymorphie et sémantique lexicale : le cas des adjectifs, *Thèse de Doctorat*.

[3] GAMUT L.T.F. (1991), *Intensional Logic and Logical Grammar, vol. 2*, Chicago, The Chicago University Press.

[4] GUARINO N. (1998), Formal Ontology and Information Systems, Actes de *FOIS'98*

[5] MOY GUPTA K., AHA D.W. (2005), Interpreting Events Using Generative Sublanguage Ontologies, Actes de *Third International Workshop on Generative Approaches to the Lexicon*.

[6] HEIM I., KRATZER A. (1998), *Semantics in Generative Grammar* Blackwell Publishers.

[7] JACQUEY E. (2004), Ambiguïtés lexicales et Traitement Automatique des Langues : Modélisation de la polysémie logique et application aux déverbaux d'action ambigus en français, *Thèse de Doctorat*.

[8] JACQUEY E. (2004), Ambiguïté lexicale et quantification : une modélisation de la polysémie logique, Chapitre 1 de *Interpréter en contexte*, Hermès.

[9] JACQUEY E. (2005), Un cas de "polysémie logique" : Modélisation de Noms d'Action en Français Ambigus entre Processus et Artefact, Actes de *Fifth International Conference on Logical Aspects of Computational Linguistics*.

[10] MONTAGUE R. (1970), The Proper Treatment of Quantification in Ordinary English, Actes de *The 1970 Stanford Workshop on Grammar and Semantics*.

[11] PUSTEJOVSKY J. (1995), *The Generative Lexicon*, Cambridge, The MIT Press.

[12] PUSTEJOVSKY J. (2001), Type Construction and the Logic of Concepts, in *The Syntax of Word Meaning*, Bouillon et Busa eds, Cambridge University Press.

[13] RETORE C. (2004), The Logic of Categorial Grammars, notes de cours

Contents

1 Introduction **3**

2 Etat de l'art **5**
2.1 Grammaires catégorielles et sémantique de Montague 5
2.1.1 Analyse syntaxique : grammaires catégorielles 5
2.1.2 Traduction du français en forme logique : sémantique de Montague 6
2.1.3 Conclusion 10
2.2 Le Lexique Génératif 10
2.2.1 Structure de représentation des entités linguistiques 11
2.2.2 Mécanisme Génératifs 13
2.2.3 Conclusion 15
2.3 Bilan de l'existant 15

3 Une reformalisation du lexique génératif **15**
3.1 Hypothèses 16
3.1.1 Analyse syntaxique et quantificateurs 16
3.1.2 Hiérarchie des types et distance sémantique 16
3.1.3 Lexique et ontologie 18
3.2 Modèle d'item lexical 18
3.2.1 Les statuts des variables 18
3.2.2 Les arguments cachés 22
3.2.3 La sous-structure REQ 24
3.2.4 Types pointés et projections 24
3.2.5 La réification systématique 25
3.2.6 La structure interne du télique 26
3.2.7 Le développement de la structure d'héritage 26
3.2.8 Le rôle constitutif 27
3.3 Règles de composition 28
3.3.1 Mécanisme général 30
3.3.2 Module 1 : sans co-composition 34
3.3.3 Module 2 : Reconstruction métonymique 36
3.3.4 Le cas de la préposition "avec" 36

4 Evaluation des résultats **40**
4.1 Problèmes couverts 40
4.2 Problèmes en suspens 40

5 Perspectives et conclusion **40**

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399