Utilisation des ontologies pour la modélisation logique d'une commande en langue naturel

Laurent MAZUEL
LIP6, 104 avenue du Président Kennedy, 75016 Paris
laurent.mazuel@lip6.fr

Résumé. Dans cet article, nous nous intéressons à l'interprétation de commandes en langue naturelle pour un agent artificiel. Notre architecture repose sur une modélisation logique de la commande pour l'interprétation sémantique, qui permet de capturer la « structure fonctionnelle » de la phrase, c'est-à-dire les rôles des termes les uns par rapport aux autres. Cet article décrit une méthode d'analyse structurelle de surface qui s'appuie sur l'ontologie de l'agent pour construire cette modélisation logique. Nous définissons tout d'abord un algorithme d'ancrage des termes de la commande dans l'ontologie de l'agent puis nous montrons comment s'en servir pour l'analyse de surface. Enfin, nous expliquons brièvement comment notre modélisation peut être utilisée au moment de l'interprétation sémantique des commandes.

Abstract. In this paper, we focus on natural language interaction for artificial agents. Our architecture relies on a command logical model to enhance the semantic interpretation. It allows us to catch the « functional structure » of the user sentence, *i.e.* each terms compared to each others. This paper describes a partial structural approach which relies on the agent ontology to build a logical form of the sentence. We first define an algorithm to anchor a word from the command in the ontology and we use it to make our partial analysis. Lastly, we explain briefly how to use our model for the semantic interpretation of the user command.

Mots-clés : commande en langue naturelle, analyse structurelle de surface, modélisation logique, ontologies.

Keywords: natural language command, partial structural analysis, logical form, ontologies.

1 Introduction

Dans les applications de commandes en langue naturelle, l'utilisation d'un analyseur syntaxique basé sur des règles grammaticale fortes de la langue pose des problèmes d'efficacité (Milward, 2000; Sabouret & Mazuel, 2005). En effet, les utilisateurs emploient plus régulièrement des mots clés plutôt que des phrases bien structurées (*e.g.* « *drop object low* » ou « *take blue* »). De plus, dans le cadre d'applications réelles, la complexité, la difficulté d'écriture de règles non-spécifiques et de maintenances rendent ces types d'approches complexes à mettre en œuvre et lourdes à utiliser (Sabah, 2006). D'un autre coté, l'utilisation d'un modèle « sac de mots » est insuffisante, générant des problèmes de modélisation impossible à interpréter par la suite (par exemple, « go from London to Boston » et « go from Boston to London » sont représentées par le même sac de mots). C'est pourquoi la majorité des travaux actuels (Hobbs *et al.*, 1997;

Eliasson, 2007) cherchent à effectuer une analyse partielle (ou de surface), afin de réduire le coût de développement, augmenter la portée utilitaire et éviter les écueils des deux modélisations extrêmes précédemment décrites.

Les méthodes actuelles d'analyse de surface s'orientent ainsi vers une modélisation basée sur la logique du premier ou second ordre (Shapiro, 2000; Milward, 2000). Cette modélisation permet à la fois de s'affranchir d'une analyse syntaxique lourde et de conserver suffisamment d'information pour être applicable facilement au moment de l'analyse sémantique. Néanmoins, le défaut de ces systèmes réside dans la définition de ces prédicats, qui doit souvent se faire dans un langage contraint dépendant d'un ensemble d'axiomes logiques spécifiques (Shapiro, 2000; Sadek *et al.*, 1997). Au contraire, l'utilisation d'ontologies dans les systèmes de dialogue a pour objectif de rendre les systèmes plus indépendants de l'application. Elles sont utilisées par exemple pour l'interprétation sémantique d'une commande pour le système (Milward & Beveridge, 2003; Flycht-Eriksson, 2003) et *avant* cette interprétation pour désambiguïser les termes d'une commande (Porzel *et al.*, 2003; Resnik, 1995). Nous pensons qu'il est aussi possible d'exploiter le contenu de l'ontologie pour construire la représentation structurelle logique de la commande, ce qui permet de s'affranchir de la définition de règles dans un langage spécifique.

Dans cet article, nous proposons de définir une méthode d'analyse structurelle de surface pour construire une modélisation logique de la commande basée sur l'étude des concepts et des relations définis dans l'ontologie de l'agent. Notre analyse s'appuie sur un ancrage des termes de la commande dans l'ontologie (nous nous plaçons dans le cadre de l'hypothèse de connectivité sémantique de Sadek (Sadek *et al.*, 1997), qui suppose que tous les concepts de toutes commandes apparaissent dans l'ontologie). En fonction des rôles des termes dans l'ontologie (relation ou classe), nous construisons une représentation de la commande sous forme de prédicats (correspondant aux relations) et d'arguments (instances de classes).

La section suivante présente brièvement notre système d'interprétation de commandes en langue naturel. Nous décrivons l'architecture principale et l'articulation entre les différents composants. La section 3 décrit plus précisément l'ancrage des termes utilisateurs à l'ontologie, l'algorithme de construction logique de la commande et l'interprétation sémantique.

2 Architecture du système de commande en langue naturel

Notre architecture est basée sur le modèle classique des « modules réseaux communicants » (Allen *et al.*, 2000; Seneff, 2002). Cette structure permet le backtrack entre les différents composants ainsi que les réponses anticipées en fonction de l'état du dialogue (figure 1). Nous donnons dans cette section les grandes lignes des modules de l'architecture, en gardant les détails de l'analyse logique et l'interprétation sémantique (comme illustration de l'utilisation de notre analyse) pour la section 3.[1]

[1] L'architecture est définie plus en détails dans (Mazuel & Sabouret, 2006). Elle est utilisée pour la définition d'agents conversationnels sur le web : `http://www-poleia.lip6.fr/~sabouret/demos`.

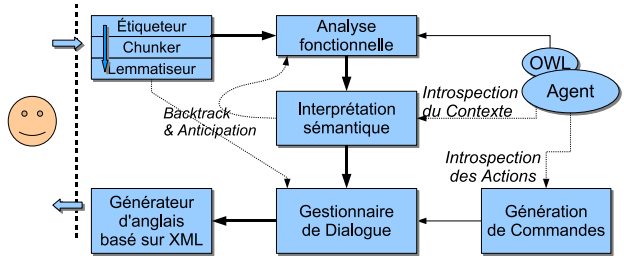

FIG. 1 – Architecture générale

2.1 Analyse morphologique et lexical

Notre module morphologique et lexical est basé sur la bibliothèque d'outils OpenNLP[2]. Nous utilisons les modules *Maximum-Entropy Tokenizer* et *Chunker*, l'étiqueteur et le lemmatiseur basé sur WordNet. L'étiqueteur, le tokenizer et le chunker sont entraînés sur des données anglaises du Wall Street Journal et du corpus Brown. Le dernier modèle proposé est annoncé à 96% d'étiquetage correct sur des données hors base d'apprentissage. Une étude comparative avec le TreeTagger[3] sur quelques exemples tirés de notre application n'a pas montré de pertes très significatives. Le lemmatiseur basé sur WordNet permet la découverte des mots composés de la commande, dans la mesure où le terme existe en tant qu'un des mots d'un *synset* (*e.g.* « *dark red* », « *extra large* »). Nous n'avons pas utilisé le module de résolution d'anaphore de OpenNLP, car elles n'apparaissent que très rarement dans une commande (à la différence de textes longs ou de dialogues).[4]

2.2 Principe de la génération de commandes formelles

Notre système de compréhension des commandes en langue naturelle repose sur une approche ascendante (*i.e.* bottom-up) comme il est possible d'en voir dans (Paraiso & Barthès, 2004). Cette approche utilise une liste *préétablie* de compétences (formelles) et essaye de relier la commande en LN à (au moins) une compétence. Cependant, elles présentent des problèmes d'efficacité en pratique (*e.g.* écriture des compétences, difficulté d'évolution, etc.) qui font que nous utilisons actuellement une version *ascendante générative* basée sur une analyse du code de notre agent (Mazuel & Sabouret, 2006). Notre modèle agent, appelé VDL, permet en effet un accès *à l'exécution* à l'ensemble du code et de son état courant (Sabouret & Sansonnet, 2001). L'algorithme de génération des commandes formelles est inspiré des travaux sur la validation de logiciel par l'analyse des préconditions d'activation d'une action.

Le principe général de l'approche ascendante générative est d'apparier les termes de la commande utilisateur avec les commandes formelles (*i.e.* notées *évènements* en VDL) générées, qui correspondent aux commandes que l'agent est capable de traiter. Cet appariement est le résultat

[2] `http://opennlp.sourceforge.net/`

[3] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger`

[4] En fait, elles apparaissent uniquement lors des dialogues avec l'utilisateur (exemple : « prend le carre vert », « ok », « pose le en haut »). C'est alors le gestionnaire de dialogue qui est responsable de leur résolution (cf. section 2.3).

de l'interprétation sémantique, dont nous parlerons brièvement en section 3.4. A l'issue de cette interprétation sémantique, chaque évènement est associé à un score d'appariement évaluant la proximité de l'événement avec la commande de l'utilisateur. L'objectif de cet article n'est pas de présenter l'algorithme de calcul de ce score (le lecteur intéressé le trouvera dans (Mazuel & Sabouret, 2006)), mais de présenter l'analyse structurelle de surface qui le rend possible.

2.3 Le gestionnaire de dialogue

A l'issue de l'interprétation sémantique le gestionnaire de dialogue utilise le score d'appariement pour déterminer la stratégie de dialogue. Nous utilisons pour cela un système de seuil inspiré de celui proposé par Patty Maes (Maes, 1994) qui permet de faire la différence entre les commandes parfaitement comprises, les commandes incertaines et les commandes non-comprises. Nous avons en plus pris en compte le cas des commandes possibles ou impossibles dans l'état courant de l'agent (Mazuel & Sabouret, 2006).

Pour répondre à l'utilisateur, le gestionnaire de dialogue utilise un générateur d'anglais qui transforme une réponse formalisée en VDL en une phrase anglaise. L'algorithme actuel est très simple et ne produit pas des réponses grammaticalement correctes, mais donne suffisamment d'informations (*i.e.* de mots clefs) pour aider l'utilisateur à reformuler sa commande. Notre objectif à long terme est d'utiliser un générateur performant basé sur XML et les ontologies.

Par exemple, dans le cas d'une ambiguïté, le gestionnaire de dialogue propose à l'utilisateur l'ensemble des commandes possibles dans le contexte courant et utilise le générateur d'anglais pour transformer les commandes formalisées :

```
- I want to go to Boston today.
- Your command is imprecise. I can either :
  - Go Boston with flight is AF1345 and departure time is 8h47
  - Go Boston with flight is AA6543 and departure time is 10h34
```

3 Analyse fonctionnelle logique

Nous décrivons dans cette section comment nous construisons un modèle de la commande de l'utilisateur sous la forme d'un ensemble de prédicats. Nous décrirons d'abord le modèle d'ontologie utilisé, l'algorithme d'ancrage d'un mot dans l'ontologie, puis enfin la construction complète de la modélisation logique de la commande.

Dans la suite de l'article, nous noterons St l'ensemble des chaînes de caractères et pour tout ensemble E, nous noterons $\mathcal{P}(E)$ l'ensemble des sous-ensembles de l'ensemble E.

3.1 Modèle de l'ontologie

Dans notre modèle, l'ontologie d'un agent[5] est un couple $\mathcal{O} = \langle \mathcal{C}, \mathcal{R} \rangle$ dans lequel :

– $\mathcal{C}$ est l'ensemble des concepts (ou *classes*). Un concept représente un ensemble d'objets réuni par les mêmes propriétés. Tout concept $c \in \mathcal{C}$ est caractérisé par un *label* l_c (nous nous

[5]Nous utilisons Jena et OWL pour l'implémentation.

limiterons à un unique label pour simplifier, mais il peut y en avoir plusieurs dans le cas de synonymie, à la manière des synsets de WordNet).
– $\mathcal{R}$ est un ensemble de relations *binaires*. Chaque relation $r \in \mathcal{R}$est caractérisée par un label de relation l_r et un ensemble de couples $E_r \subset \mathcal{C}^2$.

Par soucis de simplification, nous identifierons l_r et l_c respectivement au concept c et à la relation r, et nous noterons ainsi abusivement $\mathcal{C}$ et $\mathcal{R}$ les ensembles de labels de concepts et de relations. Nous noterons $\mathcal{L} = \mathcal{C} \cup \mathcal{R}$. Enfin, nous noterons $\langle c_1, r, c_2 \rangle \in \mathcal{O}$ lorsque les concepts c_1 et c_2 sont reliés par la relation r.

Soulignons que l'ontologie de domaine d'un agent contiendra non seulement les relations usuelles d'hyperonymie (isa) et de meronymie ($partof$), mais aussi des relations plus spécifique du domaine comme $isLargerThan$ ou $leftOf$.

3.2 Ancrage d'un mot dans l'ontologie

Soit W l'ensemble ordonné $w_1, ..., w_n$ des mots utilisés dans la commande. L'ancrage dans l'ontologie consiste à trouver le label l_c ou l_r « le plus proche » pour chaque mot w_i. Notre algorithme se décompose en trois étapes :

1. La simplification morphologique.
2. La recherche des « approximations sémantiques ».
3. L'ancrage proprement dit.

La simplification morphologique consiste à unifier l'écriture des mots ou des groupes de mots (accents, minuscule/majuscule, remplacement des espaces par « _ », *etc*). Par exemple, le terme *bigger* de la commande peut correspondre aux labels *bigger-than*, *is-bigger* encore *biggerThan* selon la notation adoptée dans l'ontologie. Nous ne détaillerons pas le calcul de cette fonction que nous noterons $app_m : St \longrightarrow \mathcal{P}(\mathcal{L})$. Elle prend en entrée un terme de la commande et renvoie la liste de candidats morphologiquement proche parmi les labels présent dans l'ontologie.

La recherche des « approximations sémantiques ».consiste à trouver l'ensemble des termes de l'ontologie les plus proches sémantiquement d'un mot de la commande, en utilisant des mesures de similarité sémantique comme décrites dans (Budanitsky & Hirst, 2006). Nous ne faisons pas ici d'interprétation sémantique de la commande dans le contexte de l'application (nous ne sommes pour l'instant que dans l'analyse structurelle), mais nous cherchons les concepts représentant le mieux les mots utilisés par l'utilisateur. Cette démarche est équivalente aux travaux visant à désambiguïser l'ensemble des concepts reconnus pour un mot d'une commande pour ne choisir que le plus représentatif du contexte de la phrase[6] (Porzel *et al.*, 2003; Resnik, 1995). Par un exemple, dans la commande « buy a place for the Pink Floyd show at the cheapest price », le terme « cheapest » est proche du label de relation *lowerThan* et le le terme « show » du label de concept *concert*[7].

[6]Il n'est pas forcément évident que les phrases employées au sein de notre application correspondent exactement aux sens enregistrés dans WordNet, surtout lorsqu'il s'agit d'un domaine technique (Resnik, 1995). Néanmoins, nous ne nous servons pas de WordNet pour l'interprétation sémantique mais pour *aider* à retrouver les mots de l'utilisateur dans l'ontologie. Ainsi, si le domaine est technique, l'ontologie le sera aussi et la plupart des termes utilisateurs seront retrouvés directement (ou par simplification morphologique). Nous n'avons d'ailleurs pas constaté en pratique de faux-sens à ce niveau.

[7]Ces exemples sous-entendent que « cheapest » et « show » ne sont pas définie dans l'ontologie et n'ont pas d'équivalent morphologique.

Pour cette recherche, nous avons choisi d'utiliser la formule de Jiang & Conrath (Jiang & Conrath, 1997) appliquée aux calculs de probabilités définies par N. Seco (Seco *et al.*, 2004). Cette formule calcule sur WordNet un score de similarité sémantique compris entre $[0, 1]$. Nous ne détaillerons pas cette formule ici car ce n'est pas l'objectif de cet article. Elle a été plusieurs fois évalué et présente les meilleurs résultats actuels en la matière (Budanitsky & Hirst, 2006). Nous noterons $sim_{JC}(w_1, w_2)$ le score de similarité sémantique entre les mots $w_1 \in St$ et $w_2 \in St$.

Nous noterons $app_s : St \longrightarrow \mathcal{P}(\mathcal{L})$ la fonction calculant l'ensemble des labels les plus proches du terme de l'utilisateur. Nous la définissons de la façon suivante :

$app_s(w) =$

$$\begin{cases} \emptyset & \text{si } max^w_{sim} < t_o \\ \{l \in \mathcal{L} \text{ tq } sim_{JC}(l, w) = max^w_{sim}\} & \text{sinon} \end{cases}$$

avec $t_o \in [0, 1]$ le ***seuil d'acceptabilité*** et la similarité maximum $max^w_{sim} = \max_{l \in \mathcal{L}} sim_{JC}(l, w)$. Le seuil d'acceptabilité t_o permet de décider si l'appariement est acceptable ou non[8]. La similarité max^w_{sim} est le score maximal obtenu pour le mot w lors du calcul de similarité sur l'ontologie. Autrement dit, $app_s(w)$ donne l'ensemble des concepts de l'ontologie de similarité maximale avec w.

Ainsi, nous pouvons définir l'ancrage $\mathcal{A} \subset St \times \mathcal{L}$ des mots $w_1, ..., w_n$ dans l'ontologie $\mathcal{O}$:

$\mathcal{A} =$

$$\bigcup_{w \in W} \begin{cases} \bigcup\limits_{l \in app_m(w)} \langle w, l \rangle & \text{Si } app_m(w) \neq \emptyset \\ \bigcup\limits_{l \in app_s(w)} \langle w, l \rangle & \text{Sinon} \end{cases}$$

Soulignons qu'un même terme peut être ancré à plusieurs labels de l'ontologie, donc à plusieurs concepts et/ou relations.

Soulignons aussi que l'interprétation sémantique (cf. section 3.4) utilise les scores calculés à cet étape par sim_{JC} pour déterminer l'imprécision globale de la commande. Cette imprécision est ensuite utilisée par le gestionnaire de dialogue pour déterminer la meilleure stratégie de dialogue.

3.3 Construction des prédicats

Notre objectif est de définir une modélisation logique qui capture la structure fonctionnelle de la phrase, c'est-à-dire de construire un ensemble de prédicats représentant les relations entre les concepts (au sens de l'ontologie $\mathcal{O}$) tels qu'ils sont exprimées dans la commande. Par exemple, dans « *the big object next to the book* », l'utilisateur exprime une relation « *next-to* » entre « *big object* » et « *book* ».

Pour cela, chaque terme est considéré du point de vue de son ancrage dans l'ontologie : si c'est une relation, nous la modéliserons sous la forme d'un prédicat et nous devons rechercher ses arguments dans la commande parmi les autres termes/concepts. En adoptant une représentation

[8] Actuellement et empiriquement, la valeur du seuil d'acceptabilité t_o est de 0.7.

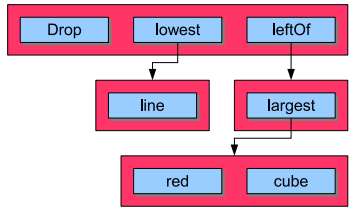

FIG. 2 – Modélisation de « drop on the lowest line, left of the largest red cube »

arborescente des prédicats, les nœuds des arbres sont les termes de la commande. Les termes qui sont des labels de concepts sont représentés par des feuilles. Les termes qui sont des labels de relations sont représentés par des nœuds dont les fils sont les arguments de la relation dans la commande de l'utilisateur. Par exemple, dans la phrase « *drop on the lowest line, left of the largest cube* », « *drop* », « *line* », « *red* » et « *cube* » sont des feuilles, « *lowest* » aura comme fils « *line* ». Nous obtenons alors le résultat présenté dans sur figure 2.

Toute la difficulté de cette construction réside dans la capacité à déterminer quel terme est un argument de quelle relation. Idéalement, nous devrions nous appuyer sur l'analyse sémantique de la phrase et sur les définitions des relations dans l'ontologie pour identifier les instances correspondant à des arguments de l'agent, en utilisant du *backtrack* pour rechercher toutes les permutations possibles.

Mais dans un premier temps, par soucis d'efficacité, nous utiliserons l'heuristique suivante, tirée de nos observations sur les relations dans la langue anglaise :

> Les arguments d'une relation sont soit l'ensemble des termes restant dans le syntagme nominal de la relation, soit dans l'ensemble des termes du syntagme immédiatement suivant.

La force de cette heuristique est qu'elle prend aussi en compte le traitement des comparatifs et des superlatifs :

1. Si un superlatif apparaît, il l'est alors à titre d'adjectif descriptif de l'objet. Les termes de la commande reliés appartiennent donc au *même syntagme* (*e.g.* « *the biggest square* », « *the darkest big object* », etc.).
2. Si un comparatif apparaît, l'objet de la comparaison est séparé par l'utilisation d'une conjonction (« *than* », etc.) et donc dans le syntagme suivant (*e.g.* « *higher than the cube* », « *left to the current position* », etc.).

Formellement, soit S l'ensemble *ordonné* $\{c_1, c_2, ..., c_n\}$ composé de n *chunks* tel que $\forall i \in [1, n]$, $c_i = \{s_{i,1}, s_{i,2}, ..., s_{i,k_i}\}$ où les $s_{i,j}$ sont les termes de la commande utilisateur, regroupés en *chunks*[9]. La fonction $\tau : S \longmapsto S_a$ construit l'ensemble d'arbres S_a à partir de la modélisation de la commande chunkée S. Les éléments de S_a seront représentés en utilisant une notation prédicat/valeurs (chaque prédicat représentant un nœud, et ses valeurs les fils du nœud).

La fonction τ est définie récursivement par : $\tau(S) =$

[9]La commande de l'utilisateur est l'ensemble ordonné $S_{user} = \{s_{1,1}, s_{1,2}...., s_{1,k_1}, s_{2,1}, ..., s_{2,k_2}, ..., s_{n,1}, ..., s_{n,k_n}\}$.

$$\begin{cases} \{s_{1,1}\,(\tau(\{\{s_{1,2},...s_{1,k_1}\}))\} \cup \tau(\{c_2,...,c_n\}) & \text{si } (k_1 > 1) \wedge (\exists c \in \mathcal{R}.\, \langle s_{i,j}, c\rangle \in \mathcal{A}) \\ \{s_{1,1}\,(\tau(\{c_2\}))\} \cup \tau(\{c_3,...,c_n\}) & \text{si } (k_1 = 1) \wedge (\exists c \in \mathcal{R}.\, \langle s_{i,j}, c\rangle \in \mathcal{A}) \\ \{s_{1,1}\} \cup \tau(\{\{s_{1,2},...s_{1,k_1}\}, c_2,...,c_n\}) & \text{sinon} \end{cases}$$

avec $\tau(\emptyset) = \tau(\{\emptyset\}) = \emptyset$. Autrement dit, l'arbre S_a est obtenu en transformant chaque relation de la commande en nœud dont les fils sont les termes restant du chunk (lorsque $k_1 > 1$) ou les éléments du chunk immédiatement suivant lorsque la relation est le dernier élément du chunk ($k_1 = 1$). Les concepts sont systématiquement transformés en feuilles. Pour mieux comprendre cette opération, considérons l'exemple suivant : « *drop on the lowest line, left of the largest red cube* » est chunkée en :

> [VP Drop :VB] [PP on :IN] [NP the :DT lowest :JJS line :NN] [? ? , :,] [NP left :NN] [PP of :IN] [NP the :DT largest :JJS red :JJ cube :NN]

Après filtrage des termes non- significatifs, nous obtenons l'ensemble d'ensembles :

$$S = \{\{drop\}, \{lowest, line\}, \{leftof\}, \{largest, red, cube\}\}$$

Nous obtenons alors $\tau(S) = \{drop, lowest(line), leftof(largest(red, cube))\}$, représenté sous forme d'arbre sur la figure 2.

3.4 Interprétation sémantique

L'analyse fonctionnelle décrite précédemment (*cf.* figure 3) permet :

1. La construction d'un ensemble d'arbres représentant la commande ;
2. L'ancrage de cet arbre, par l'ancrage de chacun de ses termes, sur l'ontologie.

Ces deux propriétés sont à la base de notre modèle d'analyse sémantique. En effet, de manière similaire, nous ancrons semi-automatiquement le code de l'agent VDL sur l'ontologie au moment de l'écriture de l'agent. Ainsi, les évènements formels construits par notre algorithme ascendant génératif, utilisant des termes issus du code VDL, sont déjà ancrées dans l'ontologie (chaque commande générée ayant un ancrage diffèrent). Nous nous retrouvons alors dans une situation proche d'un problème d'appariement d'ontologies selon une ontologie de référence (*e.g. (Aleksovski* et al.*, 2006)*). L'objectif est alors d'évaluer comparativement ses deux ancrages, afin de pouvoir décider quelles sont les commandes générées les plus proches de la commande en langue naturelle de l'utilisateur.

C'est l'ancrage des termes de la commande dans l'ontologie qui permet de se ramener à un problème (non trivial) d'alignement d'ontologies. En effet, il nous est alors possible de calculer l'alignement demandant le moins « d'effort » d'approximation entre les deux ensembles de termes ancrés et donc d'en déduire quel couple (évènement/structure de commande) est le meilleur candidat comme résultat à cette interprétation sémantique.

La modélisation logique structurée de la commande utilisateur est ensuite utilisée au moment de l'interprétation sémantique pour calculer la fermeture transitive de la relation dans le contexte courant de l'agent. Par exemple, si l'utilisateur parle d'un objet « *à côté du livre* », notre interprétation sémantique donne l'ensemble des positions correspondant à « *à côté du livre* » en fonction de la position du livre dans l'état courant.

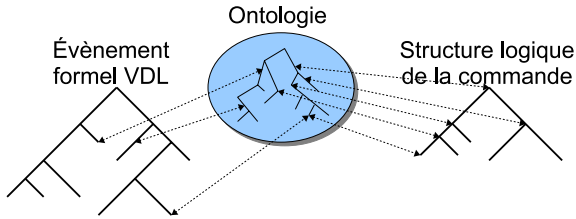

FIG. 3 – État après la modélisation logique avec un évènement formel et un seul arbre de la commande.

4 Conclusion

Dans cet article, nous proposons un algorithme de modélisation d'une commande sous la forme d'un ensemble de propositions logiques qui s'appuie sur l'utilisation de l'ontologie de l'agent. Les symboles de prédicats utilisés sont directement extrait à partir des termes de la commande en fonction de leur proximité avec les concepts de l'ontologie. Les rôles de prédicats ou arguments pour chaque terme sont choisis à partir de leur définition dans l'ontologie. Ce mécanisme ne nécessite donc pas l'utilisation d'un formalisme particulier pour définir les règles d'analyse syntaxique. La modélisation obtenue est simple à interpréter et à utiliser, en particulier pour l'interprétation sémantique de la commande. La plupart des systèmes de dialogues actuelles étant basés sur l'utilisation d'ontologies pour l'interprétation sémantique, l'approche est applicable à large échelle sur des systèmes d'implémentation diverses.

La méthode d'ancrage des termes de la commande dans l'ontologie (c'est-à-dire la recherche du concept de l'ontologie le plus proche sémantiquement d'un terme donné) que nous avons présenté repose sur un algorithme de similarité sémantique basé sur WordNet. L'évaluation préliminaire de notre système actuellement en court présente des résultats encourageants. Cependant, nous voudrions la valider sur d'autres agents et ontologies que celles que nous avons utilisées jusqu'à présent, afin de montrer la généricité de notre approche.

Références

ALEKSOVSKI Z., TEN KATE W. & VAN HARMELEN F. (2006). Exploiting the structure of background knowledge used in ontology matching. In *Proc. Workshop on Ontology Matching in ISWC2006* : CEUR Workshop Proceedings.

ALLEN J., BYRON D., DZIKOVSKA M., FERGUSON G., GALESCU L. & STENT A. (2000). An architecture for a generic dialogue shell. *NLENG : Natural Language Engineering*, **6**.

BUDANITSKY A. & HIRST G. (2006). Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, **32**(1), 13–47.

ELIASSON K. (2007). Case-Based Techniques Used for Dialogue Understanding and Planning in a Human-Robot Dialogue System. In *Proc. of IJCAI07*, p. 1600–1605.

FLYCHT-ERIKSSON A. (2003). Design of Ontologies for Dialogue Interaction and Information Extraction. In *Proc. Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI'03)*.

HOBBS J., APPELT D., BEAR J., ISRAEL D., KAMEYAMA M., STICKEL M. & TYSON M. (1997). FASTUS : A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. *Finite-State Language Processing*, p. 383–406.

JIANG J. & CONRATH D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. on International Conference on Research in Computational Linguistics*, p. 19–33, Taiwan.

MAES P. (1994). Agents that reduce workload and information overload. *Communications of the ACM*, **37**(7), 30–40.

MAZUEL L. & SABOURET N. (2006). Generic command interpretation algorithms for conversational agents. In *Proc. Intelligent Agent Technology (IAT'06)*, p. 146–153 : IEEE Computer Society.

MILWARD D. (2000). Distributing representation for robust interpretation of dialogue utterances. In *ACL*, p. 133–141.

MILWARD D. & BEVERIDGE M. (2003). Ontology-based dialogue systems. In *Proc. 3rd Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI03)*, p. 9–18.

PARAISO E. & BARTHÈS J. (2004). Architecture d'une interface conversationnelle pour les agents assistants personnels. In P. PAROUBECK & J.-P. SANSONNET, Eds., *Actes de la Journée d'Etude ATALA Agental « Agents et Langue »*, p. 83–90, Paris, France : ATALA ATALA.

PORZEL R., GUREVYCH I. & MULLER C. (2003). Ontology-based contextual coherence scoring. In *Proc. of the Fourth SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan.

RESNIK P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, p. 448–453.

SABAH G. (2006). *Compréhension des langues et interaction*. Cognition et Traitement de l'Information. Hermes-Lavoisier.

SABOURET N. & MAZUEL L. (2005). Commande en langage naturel d'agents VDL. In *Proc. 1st Workshop sur les Agents Conversationnels Animés (WACA)*, p. 53–62.

SABOURET N. & SANSONNET J. (2001). Automated Answers to Questions about a Running Process. In *Proc. CommonSense 2001*, p. 217–227.

SADEK D., BRETIER P. & PANAGET E. (1997). Artimis : Natural dialogue meets rational agency. In *IJCAI (2)*, p. 1030–1035.

SECO N., VEALE T. & HAYES J. (2004). An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proc. ECAI'2004, the 16th European Conference on Artificial Intelligence*, p. 1089–1090.

SENEFF S. (2002). Response planning and generation in the MERCURY flight reservation system. In *Computer Speech and Language*, volume 16, p. 283–312.

SHAPIRO S. (2000). Sneps : a logic for natural language understanding and commonsense reasoning. *Natural language processing and knowledge representation : language for knowledge and knowledge for language*, p. 175–195.