Traitement de l'hétérogénéité sémantique dans les interactions humain-agent et agent-agent

Laurent Mazuel

Université Paris VI - Pierre & Marie Curie
Laboratoire Informatique de Paris 6

Thèse

présentée par

Laurent MAZUEL

pour obtenir le grade de

Docteur de l'Université Pierre & Marie Curie

Spécialité Informatique

Traitement de l'hétérogénéité sémantique dans les interactions humain-agent et agent-agent

Encadrée par Nicolas Sabouret

Soutenue le 21 novembre 2008

devant le jury composé de:

Directeur de thèse	⊕	Pr. Amal El Fallah Seghrouchni	⊕ Université Paris VI
Encadrant de thèse	⊕	MdC. Nicolas Sabouret	⊕ Université Paris VI
Rapporteurs	⊕	Pr. Jean Charlet	⊕ INSERM & DSI AP-HP
	⊕	D.R. Jean-Paul Sansonnet	⊕ LIMSI-CNRS, Orsay
Examinateurs	⊕	Pr. Jacques Malenfant	⊕ Université Paris VI
	⊕	D.R. Jérôme Euzenat	⊕ INRIA - Exmo, Grenoble
	⊕	C.R. Fabien Gandon	⊕ INRIA - Edelweiss, Sophia Antipolis

Résumé

Le thème général de cette thèse est le traitement de l'hétérogénéité sémantique dans les interactions humain-agent et agent-agent. Plus précisément, nous étudions le cas où un agent informatique muni d'un modèle de représentation de ses connaissances doit traiter des demandes envoyées par d'autres interlocuteurs, qu'il s'agisse d'utilisateurs humains ou d'agents informatiques.

La plupart des approches segmentent ce traitement en fonction de l'émetteur de la demande (humain ou agent). Nous pensons au contraire qu'il est possible de proposer un modèle d'interaction commun aux deux situations. De plus, peu de travaux proposent d'exploiter le modèle de connaissances de l'agent réceptionnant la demande pour l'interpréter. Lorsque ce modèle est exploité, seules les informations liées à la description des termes sont utilisées et les termes non compris ne sont pas pris en compte. Or, nous pensons qu'il est possible d'exploiter les relations fonctionnelles entre les concepts pour approximer leur sens, et que la compréhension d'un concept qu'il n'est pas possible d'approximer peut être résolue par l'utilisation d'un protocole de communication de clarification.

Nous présentons d'abord un algorithme d'interprétation sémantique de la commande indépendant d'un type d'interaction (humain-agent ou agent-agent). Cet algorithme considère le rapport entre « ce qui est compris » de la commande et « ce qui est possible » pour la machine. Ce rapport intervient dans un système de sélection de réponses basé sur une mesure de degré de relation sémantique. Nous proposons ensuite une telle mesure, conçue pour prendre en compte plus d'informations que la plupart des mesures actuelles. En effet, là où les distances habituelles limitent l'information utilisée aux attributs (ou propriété intrinsèque) des objets, nous considérons en plus l'ensemble des relations fonctionnelles entre ces concepts.

Nous étudions ensuite les implémentations que nous avons faites dans les cadres humain-agent et agent-agent. Pour l'implémentation humain-agent, l'une des spécificités est l'utilisation d'une langue naturelle (comme l'anglais ou le français), impliquant le besoin d'utiliser des outils de modélisation de la langue. Pour l'implémentation agent-agent, la plupart des approches partent du principe que trouver un alignement correct entre les modèles de représentations des connaissances des deux agents suffit à résoudre le problème de l'hétérogénéité sémantique. Nous montrerons que cette hypothèse est trop forte, et nous proposerons d'adapter notre architecture, en nous appuyant sur des protocoles d'interactions entre agents.

Enfin, nous présenterons l'implémentation que nous avons faite de ce système, en particulier, son intégration dans la plate-forme agent VDL que nous utilisons.

Mots-clefs : Intelligence artificielle, système multi-agents, hétérogénéité sémantique, système d'interaction homme-machine, agent introspectif

Abstract

The main purpose of this thesis is the management of semantic heterogeneity in human-agent and agent-agent interaction. We especially focus on the situation where a software agent, supplied with a knowledge representation model, has to understand requests coming from different interlocutors; either it is a human user or another software agent.

Most work in this domain either focus on human-agent interaction or agent-agent interaction. On the contrary, we suggest that it is possible to use a common interaction model for both situations. Moreover, little work proposes to make use of the knowledge representation model of the receiver agent. When this model is used, only the understood terms are take into accounts, and the others terms are ignored. On the contrary, we propose to consider the functional relations between the concepts to approximate their meanings. In addition, if the understanding of a concept cannot be approximated, we suggest that it can be resolved using a clarification communication protocol.

We first present a semantic interpretation algorithm of the request independent of the given interaction type (human-agent or agent-agent). This algorithm considers the ratio between «what is understood» of the request and «what is possible» for the machine [Mazuel et Sabouret, 2008a]. This ratio is used in an answering strategies system based upon a semantic relatedness measure. We then propose our relatedness measure, designed to take into account more information that measures currently do. Actually, current measures only consider concepts attributes, whereas our proposition also considers functional relations between these concepts [Mazuel et Sabouret, 2008b].

We then study the implementation made for the human-agent and the agent-agent interactions. For the human-agent interaction, we have to model natural language requests [Mazuel et Sabouret, 2007]. For the agent-agent implementation, the majority of work considers that solving the knowledge representation alignment problem is sufficient to solve the heterogeneity problem. We will show that this hypothesis is too strong, and we propose a new version of our architecture which relies upon an interaction protocol between agents [Mazuel et Sabouret, 2009].

English title: Semantic heterogeneity management for human-agent and agent-agent interaction

Keywords: Artificial intelligence, multi-agents system, semantic heterogeneity human-machine interaction, introspective agent

Remerciements

On a coutume de dire que pour un doctorant sa thèse est son « bébé ». Un morceau de vie, trois années consacrées exclusivement à cette obsession qui vous réveille la nuit et vous travaille à n'importe quel moment de la journée. Je suis plutôt d'accord avec cette vision des choses : vous venez de commencer la lecture des premières lignes de mon « bébé ». Néanmoins, tout le monde s'accordera à dire qu'il n'est pas possible de faire un bébé tout seul. Dans le cadre d'une thèse, ces trois ans fourmillent de petites ou grandes rencontres qui modifient plus ou moins profondément ce travail. Des rencontres sans lesquelles cette thèse ne serait pas telle que vous allez la lire. Je tiens donc à profiter de cette section pour remercier toutes ces personnes.

Tout d'abord, je remercie Nicolas Sabouret, mon encadrant de thèse. J'ai souvent dit que je n'aurai pas fait de thèse si cela n'avait pas été lui (mon DESS ne me destinant pas à une thèse), et je le redis aujourd'hui. Je le remercie d'avoir tant insisté. D'avoir publié mon travail de DESS et de m'avoir poussé à l'accompagner en conférence pour comprendre par moi-même ce qu'était la « recherche ». Je le remercie pour sa patience, même durant mes crises d'angoisse du type « je-n'y-arriverai-jamais-c'est-trop-dur » et mes crises de confiance extrêmes du genre « il-n'a-pas-signé-son-mail-c'est-que-mon-dernier-rapport-l'a-deçu ». De plus, sa vision à long terme de mon travail a été un guide où m'accrocher, me permettant de ne jamais dériver dans des « eaux troubles ». Je suis encore étonné aujourd'hui de la précision de sa vision d'il y a trois ans sur la fin de ma thèse.

Je remercie de plus Amal El-Fallah Seghrouchni, d'avoir co-signé en tant que directrice cette thèse, ainsi que d'avoir eu confiance en nos idées et de les avoir défendues si vivement dans les divers rapports et présentations d'équipes auxquelles j'ai pu assister. Je tiens ensuite à remercier Jacques Malenfant, qui, avec Nicolas, m'a poussé vers le doctorat et qui a bien voulu faire partie de ce jury de thèse. Je remercie également les rapporteurs Jean Charlet et Jean-Paul Sansonnet et éxaminateurs Jérôme Euzenat et Fabien Gandon qui ont eu le courage de relire ce manuscrit, et dont les remarques pertinentes m'ont aidé à en parfaire la rédaction. Merci d'avoir bien voulu être dans ce jury et d'avoir montré tant d'intérêt pour mes travaux.

Je remercie de manière générale les autres doctorants et post-doc ayant travaillé au LIP6 en même temps que moi, Hakim, Clément, Caroline, Miniar, Magalie, Juliette, Nina, Daniel, Katia... La bonne ambiance qui régnait lors des repas, pause-café, dans les bureaux leur est due. Je remercie également le personnel administratif du LIP6, Ghislaine, Kevin, Thierry entre autres, d'avoir toujours été très disponible et compréhensif pour gérer nos « missions » en conférences, malgré les complications que cela occasionne souvent...

Je tiens ensuite à remercier les diverses personnes et collègues rencontrés lors des conférences. Comme il est coutume de le dire, les discussions pendant la pause café et/ou le gala de la conférence sont souvent les moments où l'on apprend le plus de choses intéressantes ! Pour les doctorants, en particulier Emmanuel Blanchard, Virginie Demeure, François Bouchet, David Leray, etc. Pour les professeurs, maîtres de conférences et chargés de recherches, Fabien Gandon, Rose Dieng-Kuntz, Jean-Paul Sansonnet, Frank van Harmelen entre autres. De manière générale je remercie toutes ces personnes (citées et non citées) qui ont influé sur mon travail d'une manière ou d'une autre. J'ai une pensée particulière pour Rose Dieng-Kuntz, qui nous a quitté récemment. La seule rencontre et discussion que j'ai pu avoir avec elle m'a montré une personne profondément passionnante, sympathique, intéressante et convaincante. Ces quelques minutes de discussion ont d'ailleurs été pour moi une grande source d'idées et de motivation.

Pour finir, je tiens à remercier tous mes proches et en particulier mes parents et ma femme, Aline. Je remercie mes parents pour leur soutien de tous les jours. J'espère qu'ils sont fiers de moi et de ce travail. Enfin, je remercie ma femme de la patience et de la compréhension dont elle a fait preuve. Je sais que ce n'est pas tous les jours facile d'être marié à un thésard, en particulier quand on est seule avec les enfants (quatre...) alors que son mari est en conférence à l'autre bout du monde pendant une semaine... L'amour que je ressens pour elle et nos enfants fut l'une de mes motivations les plus importantes pour avancer. Merci d'avoir été (et d'être encore) toujours à mes cotés, dans mes bons moments comme dans mes faiblesses.

Table des matières

Introduction

« La qualité de notre communication est déterminée, non par la manière dont nous disons les choses, mais par la manière dont elles sont comprises. »

Andrew S. Grove

« Entre ce que je pense,
ce que je veux dire,
ce que je crois dire,
ce que je dis,
ce que vous voulez entendre,
ce que vous entendez,
ce que vous croyez en comprendre,
ce que vous voulez comprendre, et
ce que vous comprenez,
il y a au moins neuf possibilités de ne pas se comprendre. Mais essayons quand même... »

Bernard Werber

Le problème de l'hétérogénéité sémantique dans la communication apparaît lorsque deux entités cherchent à communiquer alors que la façon dont ils conçoivent le monde est différente. Plus précisément, les individus communiquent en s'échangeant un ensemble de *termes*. La *sémantique* représente alors le sens et la signification de ces termes impliqués dans la communication, qui est donc propre à chaque individu : on dit que les individus sont sémantiquement hétérogènes. La résolution du problème d'hétérogénéité sémantique repose ainsi sur la définition de méthodes spécifiques pour réconcilier la sémantique et identifier les ensembles de termes qui représentent un même concept. On parle d'*alignement* de termes. Ces méthodes spécifiques peuvent se décliner sous différents aspects en fonction de la manière dont on conçoit les participants à la communication. Il est donc nécessaire, avant d'aller plus loin dans les définitions des méthodes de résolution de l'hétérogénéité sémantique et d'alignement, de définir le cadre pratique de communication considéré ici.

Dans cette thèse nous nous intéresserons particulièrement aux problèmes de l'hétérogénéité sémantique dans le cadre d'une communication impliquant au moins un *agent informatique* [Ferber, 1995]. Nous définissons le terme « agent informatique » comme désignant une entité informatique :

– munie d'une représentation des connaissances du monde qu'il manipule ;

– capable de raisonnements complexes issus de l'Intelligence Artificielle ;

– capable de communiquer avec un autre agent au minimum, mais aussi avec un humain.

Un agent doit ainsi posséder un modèle de représentation des connaissances du monde dans lequel il est défini. Différents paradigmes et modèles informatiques pour modéliser les connaissances existent et ont été définis au cours des années, tel que les graphes de Sowa [Sowa, 1984], les réseaux sémantiques [Sowa et Borgida, 1991], les ontologies [Gruber, 1995] ou des formalismes logique (*e.g.* logique BDI [Bratman, 1987] ou logiques de descriptions [Baader *et al.*, 2007]). D'autre part, dans le cadre de la définition d'un modèle de connaissance pour les agents communicants, l'une des hypothèses utilisées actuellement est de ne pas considérer l'hétérogénéité structurelle. Autrement dit, chaque agent de la communication doit utiliser le même modèle pour représenter ses connaissances. Le problème consiste alors dans la gestion du contenu de cette représentation, et non plus de sa forme. En se limitant aux contraintes de communication et de structure, les travaux de la littérature [Laera *et al.*, 2007, Corby *et al.*, 2004] proposent d'utiliser comme modèle de représentation de connaissances des hiérarchies de concepts. Une hiérarchie de concepts est un ensemble de concepts organisé sous forme d'arbre selon une propriété de subsomption, de sorte qu'un concept A est le fils d'un autre concept B si A est un sous-concept de B (*e.g.* le concept de *voiture* est ainsi subsumé par le concept de *véhicule*). La recherche d'alignement précédemment évoqué correspondant alors à chercher à associer un concept d'un arbre vers un concept d'un autre arbre. Néanmoins, la représentation sous forme d'arbre de subsomption possède une expressivité limitée. En particulier, elle ne permet pas de définir des relations *fonctionnelles* entre différents concepts (*e.g.* entre *miel* et *abeille* par exemple, ou entre *essence* et *voiture*). Or, ces relations fonctionnelles permettent d'inférer des informations sur la proximité des concepts que la relation de subsomption ne peut pas modéliser. Par exemple, un *cube* possède plus d'informations en commun avec un *carré* qu'avec un *triangle*, or cette information n'est pas liée aux liens de subsomption. Pour mesurer cette quantité d'informations en commun, on utilise communément des *mesures sémantiques,* qui peuvent être de *similarité* si elles exploitent uniquement la hiérarchie, ou de *degré de relation* si elles exploitent les relations non-hiérarchiques. Ces mesures mathématiques, fournissant des valeurs numériques représentant cette quantité d'informations, sont l'un des piliers des travaux sur la sémantique dans les hiérarchies de concepts [Budanitsky et Hirst, 2006]. L'un des objectifs de cette thèse sera ainsi de montrer en quoi l'utilisation de relations non-hiérarchiques peut être intéressante pour le traitement sémantique lors de la communication.

En complément de l'alignement des termes, la prise en compte de l'hétérogénéité sémantique dans les systèmes d'interaction homme-machine ou agent-agent nécessite d'adapter les protocoles de communication. En effet, lors de la communication, les participants s'échangent des demandes (ou *requêtes*) qui ne sont pas toujours directement applicables par l'agent informatique. Par exemple, la commande peut être incomplète (*i.e.* des informations manquent pour identifier clairement l'objet du message), ou localement impossible actuellement (*i.e.* l'état du monde a changé depuis l'émission de la requête) voire totalement impossible (*i.e.* la demande ne correspond pas à ce qu'il est possible de traiter). Dans ce genre de situation, la réaction de l'agent recevant la requête est primordiale pour la bonne poursuite du dialogue. Le catalogue des réactions possibles d'un agent devant une situation peut être établi avant la mise en situation de l'agent sous la forme d'un *protocole*, expliquant pour chaque type de

requête reçu la « bonne » réaction à avoir. Si différents protocoles de réaction existent dans le cadre agent-agent ou humain-agent, aucun actuellement ne considère directement le problème de l'hétérogénéité sémantique, et donc les problèmes d'alignement et de mesures sémantique précédemment développées. De même, les requêtes partielles ou difficiles à traiter ne sont généralement pas traitées, le protocole intervenant au niveau de l'application spécifique et non à un niveau supérieur « méta » qui serait indépendant de l'application. Autrement dit, il n'existe pas actuellement de protocole permettant de gérer au niveau du langage agent (et non au niveau particulier de l'application) un échange de requêtes imparfait entre agents. L'un des objectifs de cette thèse est d'étudier comment il est possible de définir un protocole de réponse de commandes imparfaites, utilisant des algorithmes de résolution sémantique, qui ne serait dépendant que du langage agent et du modèle de représentation des connaissances, et serait ainsi indépendant du type d'application précis développé avec ce langage.

Le chapitre 1 revient sur la bibliographie traitant de ces problématiques. Elle est décomposée en trois parties. La première partie détaille les algorithmes de traitement sémantique à l'intérieur d'une hiérarchie de concepts et en particulier évoque les méthodes pour calculer le degré de relation sémantique entre deux concepts. Dans les deux autres parties, nous présentons les travaux faits autour d'une part de l'interaction humain-agent et d'autre part de l'interaction agent-agent. Nous nous efforçons de définir les avantages et inconvénients de ces approches compte tenu des axes fixés précédemment, à savoir la présence/absence de modèle de représentation des connaissances explicite, ainsi que les protocoles mis en place pour gérer les cas d'erreurs ou de commandes difficiles à interpréter.

Le chapitre 2 décrit l'architecture générale de notre système et la façon dont nous agençons les différents modules et méthodes décrits précédemment. En particulier, il décrit les algorithmes d'interprétation sémantique que nous proposons, mais limités à la gestion de la commande (algorithmes indépendant de l'interaction humain-agent ou agent-agent). Ces algorithmes sont basés sur une approche bottom-up, considérant le rapport entre « ce qui est compris » de la commande et « ce qui est possible » pour la machine. Pour utiliser automatiquement ce rapport, nous utilisons un système de sélection basé sur la mesure de degré de relation sémantique décrite au chapitre 3.

Le chapitre 3 décrit ainsi en détails les travaux que nous avons effectués pour améliorer les algorithmes de distance sémantique dans les hiérarchies de concepts. Au-delà du concept de distance sémantique présenté au chapitre 1, nous avons développé une mesure de *degré de relation sémantique* prenant en compte plus d'informations que la plupart des mesures de distance actuelles. En effet, là où les distances habituelles limitent l'information utilisée aux attributs (ou propriétés intrinsèques) des objets, nous considérons en plus l'ensemble des relations fonctionnelles entre ces concepts. Ce chapitre se conclut par une évaluation de notre travail comparativement à un certain nombre de mesures proposées dans la littérature et décrites dans le chapitre 1.

Le chapitre 4 présente l'implémentation que nous avons faite de ce noyau générique pour le cas de l'interaction humain-agent. En particulier, l'une des spécificités de ce type d'interaction est l'utilisation d'une langue naturelle (comme l'anglais ou le français), impliquant le besoin d'utiliser des outils de modélisation de la langue. De plus, le modèle de représentation des connaissances d'un utilisateur n'étant pas accessible, le traitement de l'hétérogénéité sémantique dans le cas humain-agent demande d'utiliser un certain nombre d'heuristiques en

remplacement. Nous décrirons ainsi comment nous travaillons la commande en langue naturelle pour la ramener à un modèle compatible au noyau d'algorithmes décrit au chapitre 3. L'objectif de généricité est d'autant plus difficile à réaliser dans ce cas de figure, la construction d'une phrase étant très grandement dépendante du domaine en jeu. Là où la plupart des systèmes actuels utilisent des patterns de reconnaissance, nous proposerons un algorithme qui utilise le modèle de représentation des connaissances pour trouver les dépendances dans les commandes de l'utilisateur. Ce modèle n'étant pas contraint dans la définition des relations, chaque dépendance dans une commande doit se traduire par l'utilisation d'une relation présente dans ce modèle. Nous conclurons cette section par l'évaluation que nous avons faite de notre système par des utilisateurs ordinaires, c'est-à-dire non spécialistes du problème d'hétérogénéité sémantique.

Le chapitre 5 présente l'implémentation dans le cadre agent-agent. Plusieurs formalismes de communication ont déjà été proposés pour gérer l'hétérogénéité syntaxique entre les agents. La plupart des approches partent du principe que trouver un alignement correct entre les modèles de représentations des connaissances des deux agents qui interagissent suffit à résoudre le problème de l'hétérogénéité sémantique. Or, ceci suppose que les agents définissent exactement les même concepts, mais sous une forme syntaxique différente, ou un degré de spécification différent. Nous montrons que cette hypothèse est trop forte, et nous proposons une adaptation de notre architecture bottom-up pour permettre aux agents de communiquer automatiquement pour clarifier les commandes comprises partiellement. Nous reprendrons pour cela le formalisme le plus connu (FIPA), limité aux commandes (FIPA-request) et nous montrerons comment il est possible de l'étendre pour utiliser notre système dans ce cadre.

Le chapitre 6 présente l'implémentation que nous avons faite des théories précédentes. En particulier, notre système est entièrement intégré dans la plate-forme agent VDL que nous présentons dans le chapitre 2. De plus, une librairie indépendante de gestion des mesures sémantiques dans les hiérarchies a été développée. Cette bibliothèque couvre différents formalismes de définition de hiérarchie et défini par défaut les mesures sémantiques les plus connues.

Chapitre 1

Étude de la problématique

Sommaire

1.1 Fonctionnement d'un système d'interaction

Dans cette section, nous allons chercher à expliciter et clarifier le fonctionnement général d'un système d'interaction humain-agent et agent-agent. Dans le cas le plus courant, il y a deux participants dans une interaction :

– L'émetteur : l'agent (ou l'utilisateur) désirant l'exécution d'une commande ou la réponse à une requête. Il est l'initiateur de la communication.

– Le récepteur : l'agent (ou l'utilisateur) désigné par l'émetteur pour répondre au message.

Le fait que cet agent reçoive un message n'implique pas qu'il soit capable de le traiter. D'autre part, les informations échangées entre les deux participants peuvent être partionnées en deux catégories :

– Les messages : un message est une information unique échangée entre les deux participants.
– Une interaction : une interaction est liée au traitement de la demande initiale de l'émetteur. Typiquement, une interaction est composée d'un échange de plusieurs messages.

Par exemple, supposons un cas simple où l'émetteur demande l'exécution d'une tâche A à un agent récepteur et que ce dernier exécute la commande et le confirme. Nous avons une seule interaction (liée à l'exécution de la tâche A), mais deux messages (la demande d'exécution de la tâche A et la confirmation d'exécution).

Lors de la réception du message, un agent récepteur doit effectuer plusieurs tâches pour traiter ce message (c.f. figure 1.1). Dans un premier temps, il doit identifier dans la commande reçue les concepts correspondant à des concepts qu'il connaît dans son Modèle de Représentation des Connaissances (MRC). Autrement dit, si les deux agents représentent d'une manière différente leurs connaissances, alors le message défini suivant les connaissances de l'agent émetteur devra d'abord être traduit selon les connaissances de l'agent récepteur. Alors seulement, l'agent récepteur pourra lancer le processus d'interprétation du message. Comme pour tout processus de traduction, une attention particulière doit être apportée au fait de conserver au maximum le sens des concepts. Autrement dit, un concept d'un MRC donné doit être traduit dans le MRC de l'autre agent par un concept conservant son sens et sa sémantique. Cette opération n'est pas forcément réalisable parfaitement, en fonction des manques et des différences existant entre les deux MRCs. Si un concept équivalent n'existe pas, il est ainsi nécessaire de pouvoir mesurer quel concept du MRC cible représente la meilleure approximation du sens du concept du MRC initial. Ces méthodes de mesure de score sémantique sont ainsi au cœur de l'interprétation sémantique dans les systèmes d'interactions, car ce sont elles qui fournissent les données nécessaires au bon traitement du message.

Une fois la requête exprimée selon le MRC de l'agent récepteur, l'agent doit décider comment la traiter. Il doit alors définir un ensemble de méthodes et/ou stratégies pour décider s'il doit exécuter une *action* de l'agent (*i.e.* un comportement lié à l'application spécifique pour lequel l'agent existe) et/ou construire une réponse pour l'agent émetteur (*i.e.* confirmation de l'exécution de l'action liée à la commande, clarification nécessaire suite à une demande partiellement comprise, etc.). La gestion des réponses dans un système d'interaction constitue ainsi l'un des points cruciaux de son architecture car elle détermine la fluidité de conversation, le nombre de messages échangés sur un réseau informatique ainsi que la rapidité d'exécution de la tâche demandée par l'agent émetteur.

Ce chapitre d'étude de la littérature s'articule ainsi autour de ces deux problématiques :

1. L'approximation sémantique et les scores de mesure sémantique

   Nous étudierons comment sont définies les mesures de distance/similarité sémantique dans un MRC, indépendamment de leurs applications dans le cadre d'un système d'interaction. Nous examinerons en particulier les mesures définies sur une hiérarchie de concepts, ces mesures étant les plus développées actuellement. Ceci permettra de mettre en avant les avantages et inconvénients théoriques de chaque mesure. De plus, nous

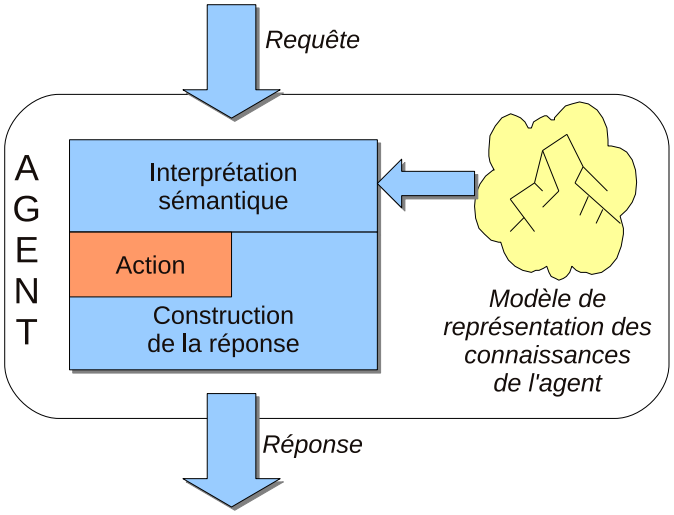

FIG. 1.1 – Schéma général simplifié d'interprétation de requête par un agent. Une action n'est pas toujours possible avec réception d'une requête, en fonction de l'ambiguïté de la commande. Le traitement des requêtes devient plus complexe si la requête n'est pas exprimée avec les termes du modèle de représentation des connaissances de l'agent.

discuterons de l'évaluation de ces mesures selon les protocoles classiques de l'ingénierie des connaissances. Ce point fera l'objet de la section 1.2.

2. La stratégie de réponse du système d'interaction.

   Nous n'avons pas trouvé dans la littérature de systèmes proposant à la fois une gestion des interactions humain-agent et agent-agent. Nous décomposerons donc cette section en deux sous-sections complémentaires :

   (a) Les systèmes d'interaction humain-agent, présentés en section 1.3 page 52.

   (b) Les systèmes d'interaction agent-agent, présentés en section 1.4 page 72.[1]

   L'objectif de cette partie est d'évaluer les propriétés théoriques de ces systèmes, ainsi que de discuter des évaluations si elles existent. Nous nous efforcerons (quand cela est possible) d'étudier particulièrement les systèmes qui utilisent un modèle de représentation des connaissances explicite et font appels aux algorithmes présentés dans la section distance/similarité sémantique.

1.2 Mesure sémantique

Cette section présente les différentes approches de la littérature autour du problème de *calcul d'un score sémantique dans un modèle de représentation des connaissances.* Nous étudierons particulièrement le cas de connaissances modélisées sous forme d'une hiérarchie de concepts, forme la plus souvent utilisée en pratique dans les systèmes d'interactions. Enfin, étant donné qu'une commande contient plusieurs concepts, il est aussi nécessaire d'étudier

[1] Dans le cadre de cette thèse, le contexte machine-machine a été étudié sous l'angle des systèmes multi-agents. D'autres applications autour de ce contexte général existent, tels que la médiation dans les services Web sémantique [Reynaud et Giraldo, 2003]. Conceptuellement, le problème de l'hétérogénéité sémantique est le même (*i.e.* un MRC par service, hétérogènes entre eux, etc.).

la question d'une distance sémantique entre ensembles de concepts, ce qui constituera la dernière partie de cette section.

1.2.1 Notion de *similarité*, de *distance* ou de *degré de relation* sémantique

Il existe trois types de mesures sémantiques, chacune possédant des propriétés différentes (résumés sur le tableau 1.1) :

La similarité sémantique : On parle de mesure de *similarité* sémantique (*i.e.* semantic *similarity*) lorsque la mesure calcule si deux concepts sont similaires sémantiquement, au sens où ils partagent des propriétés et attributs communs. Par exemple « voiture » et « vélo » sont similaires car ils possèdent tous les deux les attributs d'un « véhicule à roues ».

Le degré de relation sémantique : On parle de *degré de relation* sémantique (*i.e.* semantic *relatedness*) lorsque la mesure calcule si deux concepts sont reliés sémantiquement, au sens où ils sont liés dans leur fonction. Par exemple, « voiture » et « essence » auront un fort degré de relation sémantique.

La distance sémantique : On parle de *distance* sémantique (*i.e.* semantic *distance*) lorsque la mesure calcule si deux concepts sont distant sémantiquement. Par exemple, « voiture » et « vélo » ont une courte distance sémantique, mais aussi « voiture » et « essence ».

La similarité sémantique et le degré de relation sémantique sont intéressants à présenter ensemble car ils ont le même type de croissance : plus deux concepts sont reliés ou similaire, plus le score sémantique est grand. Autrement dit, soit $scr(x, y) \in [0, 1]$ une mesure donnant un score sémantique entre 0 et 1, alors que scr représente une similarité sémantique ou un degré de relation sémantique, nous avons $scr(x, x) = 1$. Néanmoins, le degré de relation sémantique tient compte de plus d'informations que la similarité sémantique, puisqu'il utilise aussi les relations fonctionnelles entre les concepts. Ainsi, le degré de relation sémantique peut être considéré comme une généralisation de la similarité sémantique [Resnik, 1995].

D'autre part, la courbe d'une fonction de distance sémantique est l'exact opposé de la courbe de similarité ou de degré de relation (les conversions linéaires d'une forme à l'autre sont ainsi couramment utilisées [Resnik, 1995, Jiang et Conrath, 1997]). En ce sens, en reprenant la notation $scr(x, y)$ précédente, dans une distance sémantique nous avons $scr(x, x) = 0$. L'inconvénient principal de la notation de distance est qu'elle ambiguë. En effet, elle désigne aussi bien l'inverse de la similarité sémantique que l'inverse du degré de relation sémantique. Pour éviter les confusions, nous serons donc rigoureux sur l'utilisation des termes « distance », « similarité » et « degré de relation ». La tendance générale a permis de développer les mesures de similarité en priorité, mais les travaux actuels (dont les nôtres, chapitre 3) tendent vers les mesures de degré de relation.

1.2.2 Thesaurus, réseaux sémantiques ou ontologies ?

On ne peut parler de mesure sémantique qu'en fonction d'un modèle de MRC (hiérarchique, etc.). En effet, il semble assez logique de dire que la façon de modéliser une distance doit être fonction de la représentation des connaissances utilisée. Nous détaillons ici

		Score identité $scr(x,x)$	
		$scr(x,x)=0$	$scr(x,x)=1$
MRC	Attributs	Distance	Similarité
	Toutes relations	Distance	Degré de relation

TAB. 1.1 – Terme utilisé pour désigner une mesure sémantique en fonction du type de relation utilisé dans le MRC et du score $scr(x,x)$ de l'identité (*i.e.* score d'un concept à lui-même).

les grandes sources classiques, couvrant l'ensemble des mesures de distance décrites dans la littérature.

1.2.2.1 Thesaurus

Un thesaurus est un ensemble hiérarchique de termes clés représentant des concepts d'un domaine particulier. Ils sont organisés en thèmes et possèdent des liens sémantiques entre eux : synonymie, équivalence, terme spécifique (lien vers un concept de sens plus précis), terme général (lien vers un concept de sens plus large).[2]

Le plus connu et utilisé est le Roget's Thesaurus [Kipfer et Chapman, 2001] crée par le Docteur Peter Mark Roget (1779-1869) en 1805 et accessible au public depuis 1852. L'édition originale contait 15000 mots. Elle en compte 250000 aujourd'hui. Son évolution est gérée par des linguistes et psychologues. Son système de catégories vient des travaux en philosophie de Leibniz.

Ce thésaurus est séparé en 6 classes primaires (*abstract relation, space, matter, intellect, volition* et *affections*, figure 1.2). Le but de Roget était de garder une taxonomie la plus uniforme et régulière possible. Ce thésaurus n'a pas de contraintes particulières sur la nature des liens (au contraire du modèle de représentation des connaissances WordNet, que nous présenterons section 1.2.2.3). Par exemple, le concept *money* possède les relations *usedInPlural* avec les concepts *capital* et *finance*, la relation *mediumOfExchange* avec *cash*, la relation *slang* avec *bread*, etc. Ceci lui permet d'être le thésaurus le plus riche sémantiquement, mais aussi un des plus difficile à traiter automatiquement.

La structure d'un thesaurus n'est donc pas formellement définie sur une hiérarchie (même si des travaux récents ont proposés des méthodes pour générer automatiquement une hiérarchie de concept à partir du thesaurus Roget [Kennedy et Szpakowicz, 2007]). Par contre, un thesaurus possède des niveaux (*i.e.* ensemble de concepts de même profondeur). Le Roget's Thesaurus est uniforme, c'est-à-dire que chaque niveau représente un niveau de conceptualisation équivalent.

Du fait de sa structure relationnelle complexe, peu de mesures sont définis sur un modèle de type Thesaurus. Les formules, plutôt que d'utiliser les relations, utilisent les différents niveaux pour évaluer un score sémantique entre deux concepts (tel que la distance de Jarmasz & Szpakowicz, que nous présenterons section 1.2.3.2.5).

[2] Définition du site `http://www.dicodunet.com/`.

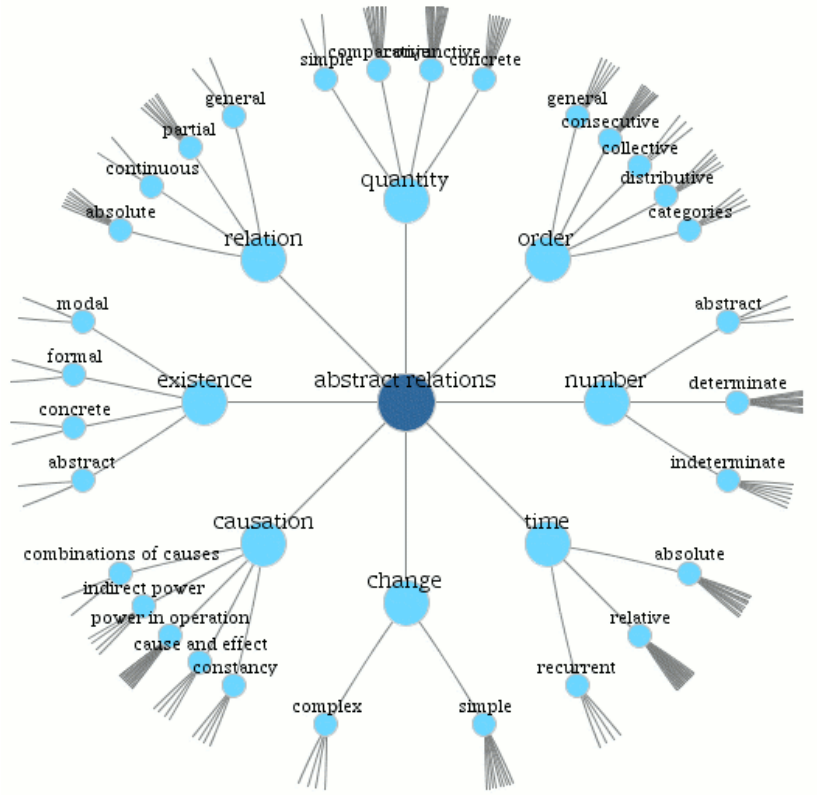

FIG. 1.2 – Vu circulaire de la classe *abstract relation* du Roget's Thesaurus. `http://www.roget.org/`.

1.2.2.2 Milieu médical

Le milieu médical est un secteur très riche en terminologie. Par exemple, les termes *heart attack*, *myocardial infarction* et parfois simplement *MI* sont parfaitement équivalents. L'inconvénient de cette profusion de termes est la difficulté à construire des systèmes informatisés et robustes d'aide au médecin (moteur de recherche clinique ou bibliographique par exemple). La communauté scientifique a donc particulièrement travaillé à produire des banques de connaissances médicales. Les deux principaux sont MeSH et SNOMED, dont nous donnons un aperçu ici.

1.2.2.2.1 MeSH

Le thésaurus MeSH (*i.e. Medical Subject Headings*) est un vocabulaire produit par l'organisme *National Library of Medicine* utilisé pour indexer, cataloguer et chercher des informations et documents à caractères biomédicals ou reliés à la santé. Par exemple, *MedLine* (branche *science et informations* de l'arborescence MeSH) permet (via le portail *PubMed*) de faire des recherches bibliographiques sur des documents médicaux. L'idée de MedLine est d'incorporer un maximum de sémantique dans leur moteur de recherche, le milieu médical étant réputé pour la profusion de termes synonymes. Dans ce sens, une partie des concepts de MeSH possède des relations de synonymie, synonymies-partielles et relationnelles, mais ce qui est insuffisant et amène les chercheurs à utiliser SNOMED ou CIM-10 (Classification

Internationale des Maladies) comme apport sémantique.

Les *descripteurs* MeSH (*i.e.* un nœud de l'arborescence MeSH) sont organisés selon 16 catégories : catégorie A pour l'anatomie, catégorie B pour les organismes vivants, catégorie C pour les maladies, catégorie D pour les médicaments et molécules médicales, etc. Chaque catégorie est divisée en plusieurs sous-catégories. Dans chaque sous-catégorie, les descripteurs sont classés hiérarchiquement sur 11 niveaux. MeSH autorise le multi-héritage. Voici un exemple d'entrée MeSH :

```
Abnormalities C16.131
    Abnormalities, Drug Induced C16.131.42
    Abnormalities, Multiple C16.131.77
        Alagille Syndrome C16.131.77.65
        Angelman Syndrome C16.131.77.95
```

Chaque descripteur est suivi par un nombre indiquant sa position dans l'arbre. Il peut être aussi suivi par un ou plusieurs nombres supplémentaires, indiquant chacun un identifiant dans le niveau de l'arbre. Les nombres n'ont pas de valeur sémantique particulière, et deux nombres se suivant ne sont pas particulièrement en relation. D'ailleurs, ces valeurs changent régulièrement en fonction des mises à jour de MeSH.

MeSH est une hiérarchie suffisamment complète pour avoir été la première source de connaissance utilisée dans une mesure de distance sémantique (celle de Rada [Rada *et al.*, 1989], section 1.2.3.2.1). Elle a néanmoins comme défaut de ne pas avoir des relations hiérarchiques « pures », dans le sens où la sémantique de certains liens n'est pas homogène à une relation de subsomption (*e.g.* dans MeSH, nous avons *Accident prevention [G03.850.110.060] is-a Accidents [G03.850.110]*, ce qui n'est pas conceptuellement correct). Son utilisation formelle sans précautions risque ainsi d'introduire des non-sens difficilement détectables.

1.2.2.2.2 SNOMED

La SNOMED (*i.e. Systematized Nomenclature of Medicine*) est une classification multi-axiale standardisée développée par le *College of American Pathologists*. Elle a été créée en 1965 comme système de terminologie sur les pathologies. Elle a depuis évolué en une version internationale en français (SNOMED 3.5 Fr) et un version « ontologique » (SNOMED CT, pour *SNOMED Clinical Terms*). Ces versions actuelles informatisées couvrent la terminologie clinique, les maladies, symptômes et procédures de soins. Son but principal est d'uniformiser les différentes terminologies utilisées pour des pathologies équivalentes.

La SNOMED internationale 3.5 contient actuellement 116000 concepts (pour environ 150000 termes en comptant les synonymes). Elle est organisée en 11 axes (Terminologie, Diagnostic, etc.), chacun de ces axes étant organisé hiérarchiquement. Il existe de plus des liens non hiérarchiques entre les axes. Par exemple, le concept « hémorragie ombilical » de l'axe *Fonction* est associé aux concepts « ombilic » (axe *Terminologie*) et « hémorragie » (axe *Morphologie*). Bien que de grande dimension, la SNOMED v3.5 n'est pratiquement pas utilisée informatiquement actuellement à cause de son format spécifique (sous forme de tableur Excel) et des erreurs de codage qu'elle contient. Elle reste néanmoins la base de connaissances médicales hiérarchisée la plus prometteuse.

La SNOMED CT est actuellement la plus grande ontologie médicale, avec près de 344000

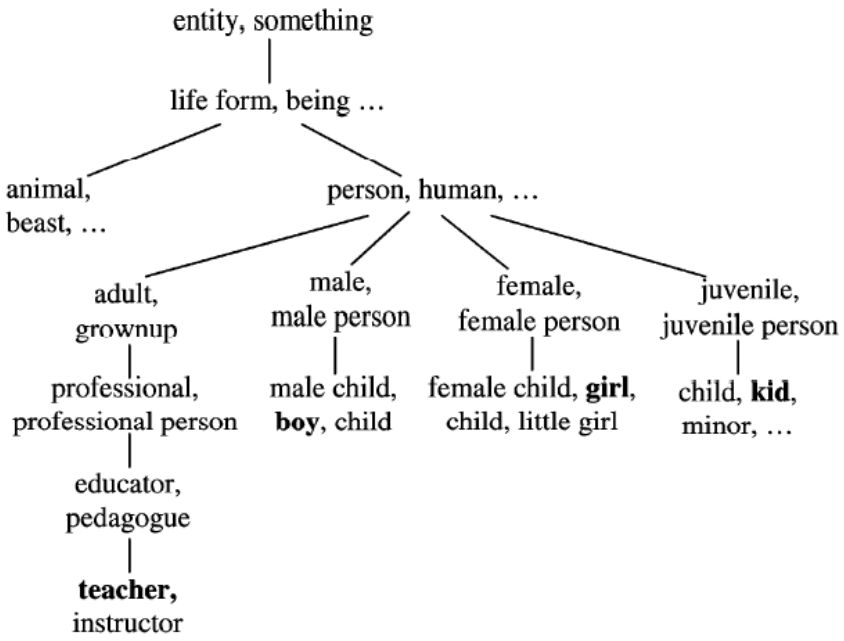

FIG. 1.3 – Exemple de hiérarchie WordNet. « ... » indique que certains mots ont été retirés pour gain de place et clarté.

concepts. Les concepts sont organisés sous forme de hiérarchie, mais beaucoup d'informations ontologiques (lien hétérogènes, post-coordinations, etc.) sont définies. Néanmoins, elle reste difficile d'accès et n'est actuellement définie qu'en anglais.

1.2.2.3 Réseau sémantique : WordNet

WordNet[3] [Fellbaum, 1998] est une base de données lexicales pour l'anglais. Les noms, verbes, adjectifs et adverbes sont organisés en *liste de synonymes* (ou *synsets*). Chaque synset représente un concept, relié par des relations venant des réseaux sémantiques. Les relations exactes de WordNet sont :

1. L'hyperonymie : La relation classique *is-a*. Par exemple, *véhicule* est un hyperonyme de *voiture*.
2. L'hyponymie : La relation inverse de l'hyperonymie, appelée aussi *subsomption*.
3. La méronymie : Est la relation de partie, ou *part-of*. Dans WordNet, elle est décomposée en trois sous-parties plus précises :
   (a) component-object : Une *branche* un méronyme d'un *arbre*.
   (b) member-collection : Un *arbre* est un méronyme d'une *forêt*.
   (c) stuff-object : L'*aluminium* est un méronyme d'un *avion*.
4. L'holonymie : La relation inverse de la méronymie, ou *has-a*. Dans WordNet, il en existe trois, inverse des relations de méronymie :

[3] `http://wordnet.princeton.edu/`

   (a) object-component : L'inverse de component-object.

   (b) collection-member : L'inverse de member-collection.

   (c) object-stuff : L'inverse de stuff-object.

5. L'antonymie : La relation *complement-of*, décrivant les opposés. Exemple : *sœur* et *frère*, *monter* et *descendre*, etc.

6. La synonymie : WordNet les associe au sein des synsets. Tous les composants d'un synset sont synonymes.

WordNet contient actuellement (version 3.0) plus de 156000 mots, 118000 synsets et environ 300000 arcs de relations. La hiérarchie représente environ 80% de la structure de WordNet. Elle est découpée en 11 catégories principales telles que : *entity, psychological feature, abstraction, shape/form, event,* etc. Les catégories ne sont pas reliées entre elles. Néanmoins, afin de pouvoir calculer des distances sémantiques entre catégories, les auteurs utilisant WordNet rajoutent toujours un nœud *thing* possédant comme hyponymes les 11 catégories.

WordNet est historiquement la base de connaissances sur laquelle les chercheurs ont le plus travaillé pour les distances sémantiques, mais en n'utilisant que la hiérarchie des concepts (section 1.2.3.2 et section 1.2.3.3). Certains chercheurs ont essayé d'utiliser d'autres relations, mais WordNet atteint ses limites. Par exemple, dans l'expérience de Finkelstein [Finkelstein *et al.*, 2001], statistiquement, une personne donne une note de 7/10 à la similarité entre « communication » et « téléphone ». Or, ces deux concepts n'ont aucun lien dans WordNet : quelle que soit la mesure utilisée, il sera toujours impossible de reproduire ce résultat.

1.2.2.4 Wikipedia

Wikipedia[4] est une encyclopédie libre et modifiable par tout utilisateur sur le Web. La version anglaise de Wikipedia contient ainsi (au 15 juillet 2008) environ 2.456.600 entrées. Les entrées sont reliées entre elles par des liens hypertextes et sont organisées dans des catégories (science, littérature, etc.). De plus, l'accès aux bases de données de Wikipedia est gratuit et les APIs de programmation sont fournies par le site.[5]

A la différence des structures de connaissances précédemment décrites, Wikipedia n'est pas constitué autour d'une hiérarchie de concepts (bien que quelques travaux récents proposent de générer automatiquement cette hiérarchie [Ponzetto et Strube, 2007]). Wikipedia est ainsi surtout utilisé comme une source de définitions de termes (*i.e. gloss-based approach*) [Strube et Ponzetto, 2006, Gabrilovich et Markovitch, 2007].

1.2.2.5 Google, Yahoo ! et les moteurs de recherche internet

Le principe d'un moteur de recherche sur internet est de retrouver à partir d'une requête à base de mots-clefs un ensemble de pages (jugées pertinentes) traitant de ces mots-clefs. Ainsi, les moteurs de recherches internet peuvent être vus comme un accès à de vastes corpus de textes et de données. De plus, pour une requête donnée, le moteur de recherche renvoie

[4] `http://fr.wikipedia.org/`
[5] `http://download.wikimedia.org/`

aussi le nombre de pages (*i.e.* nombre d'occurrences) des mots de cette requête sur internet. En utilisant cette propriété, certains auteurs ont ainsi proposé d'utiliser les connaissances indexées par Google, Yahoo !, etc. pour calculer la distance sémantique entre deux mots, basée sur la co-occurrence de termes [Cilibrasi et Vitanyi, 2006, Iosif, 2007].

1.2.2.6 Le grand absent : ontologie et taxonomie augmentée

Une taxonomie augmentée est une taxonomie de concepts (*i.e.* les concepts sont reliés par des liens de subsomption), augmentée de relations non contraintes entre ces concepts (tel que $isMadeWith$ ou $partOf$). Une taxonomie augmentée est donc beaucoup plus expressive qu'une simple hiérarchie de concepts.

Parmi les hiérarchies composées de relations diverses, les langages d'ontologie (tel que celui du W3C, OWL [Smith *et al.*, 2004]) sont plus expressifs encore que les taxonomies augmentées. Ils permettent par exemple de définir des unions/disjonctions de classes, des contraintes de cardinalité, etc.

Dans le cadre d'un système d'interaction, il est nécessaire de modéliser les connaissances sous la forme d'un de ces deux modèles ; les MRC décrits précédemment tel que WordNet ou Wikipedia ne sont en effet pas utilisables, car :

- Un système d'interaction demande une modélisation des termes spécifiques à son domaine [Dzikovska *et al.*, 2003, Corby *et al.*, 2004, Milward et Beveridge, 2003]. Ainsi par exemple, dans le domaine de l'aviation, « appareil » et « avion » seront considérés comme des synonymes, ce qui n'est pas le cas dans le cadre général.
- Un système d'interaction a besoin de modéliser des relations hétérogènes entres les concepts, et pas uniquement un lien hiérarchique de subsomption [Steichen *et al.*, 2006, Gandon *et al.*, 2008]. Par exemple, le lien entre les concepts *cube* et *carré* n'est pas l'ordre de la subsomption, alors que cette information peut être importante pour interpréter une requête (notamment pour déterminer qu'un *cube* est plus lié à la notion de *carré* qu'à celle de *triangle*).

À l'instar des autres systèmes d'interaction, notre système exige que chaque agent informatique soit relié à une taxonomie augmentée. Or, nous discuterons section 1.2.3.4 du fait que les distances sémantiques actuelles ne sont pas suffisantes pour exploiter la structure d'une taxonomie augmentée, ce qui nous a conduit à concevoir notre propre proposition de mesures, que nous présenterons dans le chapitre 3.

1.2.3 Formules théoriques de mesure sémantique

Les formules sémantiques classiques [Budanitsky et Hirst, 2006] sont les mesures de similarité sémantique. Ces mesures sont basées sur une hiérarchie de concepts uniquement. Nous étudierons dans cette section les différents types de mesures de similarité sémantique (par longueur de chemin puis par théorie de l'information), puis nous introduirons en section 1.2.3.4 les mesures de degré de relation sémantique qui exploitent toutes les relations (et non uniquement la hiérarchie) afin de montrer que les modèles actuels sont actuellement insuffisants.

1.2.3.1 Notations

Cette section permet de fixer un certain nombre de notations que nous utiliserons pour décrire les formules sémantiques de cette section :

- c_1 et c_2 : Les *concepts* dont nous chercherons la distance. Un concept fait partie d'un *synset, i.e.* un ensemble de termes dont au moins un des sens représente ce concept.
- $w_1$et w_2 : Les *mots* dont nous cherchons la distance. Un mot est peut être associé à plusieurs concepts (ou plusieurs synsets), plus ou moins probable en fonction du contexte (*avocat*, *canard*, etc.).
- $s(w_1) = \{c_1^1, ..., c_k^1\}$ et $s(w_2) = \{c_1^2, ..., c_m^2\}$: Les *synsets* contenant les mots w_1 et w_2. Le mot w_1 est attaché à k concepts distincts, et w_2 à m concepts distincts.
- C_1, C_2, W_1 et W_2 : Les ensembles (ou nuages) de concepts et de mots dont nous chercherons la distance.
- $dist_{XXX}$ et sim_{XXX} : Notation pour la distance et la similarité sémantique. Nous remplacerons XXX par l'acronyme du nom des auteurs.
- $ccp(c_1, c_2)$: Le nœud de la hiérarchie qui est le plus proche parent commun de c_1 et c_2 (*i.e. closest common parent*, notation de Zhong [Zhong *et al.*, 2002]). Cette formulation est évidemment plus courante dans les systèmes dont les connaissances sont représentées uniquement sous forme de hiérarchie, mais elle est aussi souvent utilisée dans les systèmes possédant d'autres types de relations, la relation de hiérarchie ayant en général un traitement spécial prioritaire.
- *root* : Le nœud racine de la hiérarchie. Il représente le concept le plus général, englobant tous les concepts. Il est souvent nommé *thing* dans la hiérarchie de concepts (*root* étant le nom plus « calculatoire » lié à la structure arborescente d'une hiérarchie).
- $depth(c)$: La profondeur du nœud c dans la hiérarchie avec $depth(root) = 0$.
- $sp(c_1, c_2)$: Le plus court chemin entre c_1 et c_2 *en suivant la hiérarchie* (*i.e. shortest path*). Ce plus court chemin est un ensemble de couples de concepts, un couple représentant une arête et l'ensemble de ces arêtes le chemin.
- $len(c_1, c_2) = |sp(c_1, c_2)|$: La longueur du plus court chemin entre c_1 et c_2.
- $MAX = \max_c depth(c)$: La hauteur maximale de la hiérarchie.

Il est intéressant de noter que par construction, le plus court chemin entre deux concepts dans une hiérarchie passe toujours par leur plus proche parent commun.[6] Cette propriété, bien que rarement directement nommée, permet de rapprocher certaines formules de distance à première vue radicalement différentes.

La différence entre un *mot* et un *concept* est très importante. Un mot est associé à plusieurs concepts, mais cette dépendance est liée à la langue utilisée. Par exemple, en anglais le terme *rock* possède au moins deux concepts (*musique* et *rocher*) alors que le terme français *rocher* ne posera pas cette ambiguïté. Par contre, un concept est indépendant de la langue. Le concept *voiture* possédera un mot dans différentes langues mais désignera toujours le même objet.

Or, les mesures de la littérature n'évaluent pas la similarité entre deux mots mais entre deux concepts. Ainsi en pratique, il est nécessaire de faire le lien entre une similarité entre

[6] Cette hypothèse n'est valable que dans le cas où la hiérarchie n'autorise pas de multi-héritage. Dans le cas du multi-héritage, elle ne reste valable que si il n'existe qu'un seul et unique chemin hiérarchique entre les concepts cibles.

mots et entre concepts. Le consensus actuel ([Resnik, 1995]) est de considérer le *max* des similarités entre concepts comme représentatif de la similarité entre mots :

$$sim(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} sim(c_1, c_2)$$

avec $sim(c_1, c_2)$ une fonction quelconque de similarité entre concepts. Nous parlerons donc toujours de similarité entre concepts, sous-entendue qu'en pratique il faut utiliser la formule ci-dessus pour passer des mots aux concepts.

Néanmoins, comme le montre [Resnik, 1995], cette formule générale peut engendrer des mauvais calculs de similarités. Dans son exemple, Resnik explique la similarité étonnement forte entre *tabac* (*i.e. tobacoo*) et *cheval* (i.e. *horse*) dans WordNet. *Horse* est un mot d'argot utilisé comme synonyme de *heroin* qui est une drogue (*i.e. drug*). Hors le tabac est une drogue. Néanmoins, ce problème est en pratique assez rare et spécifique des sources de connaissances génériques. En effet, dans un MRC se limitant à un domaine en particulier, il ne peut pas se produire (car le domaine est trop spécifique pour offrir plusieurs sens à un même mot du vocabulaire).

1.2.3.2 Longueur du chemin (edge-based approach)

La première catégorie de mesure sémantique est celle qui se base sur les longueurs de chemins. La source de connaissance est vue comme un arbre (au sens théorie des graphes) et les formules utilisent la longueur des chemins (pondérés ou non) pour déterminer la distance entre deux concepts. Cette idée part d'une intuition naturelle : dans une hiérarchie sémantique, plus deux concepts sont éloignés physiquement, plus ils sont éloignés sémantiquement.

1.2.3.2.1 Rada

Roy Rada fut le premier à envisager la distance entre concepts comme une distance métrique [Rada *et al.*, 1989]. Il utilise comme source de connaissances la hiérarchie MeSH. En ce sens, il affirme que toute distance sémantique se doit de respecter les hypothèses de bases d'une distance métrique, à savoir si $f(x, y)$ est une distance métrique :

1. Existence du zéro : $\forall x, f(x, x) = 0$
2. Symétrie : $\forall (x, y), f(x, y) = f(y, x)$
3. Positivité : $\forall (x, y), f(x, y) \geq 0$
4. L'inégalité triangulaire : $\forall (x, y, z), f(x, y) + f(y, z) \geq f(x, z)$

Les trois premiers points sont admis par toutes les mesures actuelles. En revanche, l'inégalité triangulaire est souvent discutée et finalement rarement respectée. En effet, si les arêtes sont pondérés, alors elle n'est souvent plus valide [Zhong *et al.*, 2002]. De même, la présence d'héritage multiple invalide cette propriété.

Roy Rada définie sa mesure de distance simplement :

$$dist_{RAD}(c_1, c_2) = len(c_1, c_2)$$

où *len* est la fonction donnant la longueur du plus court chemin entre c_1 et c_2.

Première distance sémantique proposée, la distance de Rada reste une référence mais elle ne tient pas compte du fait que la position d'une arête dans la hiérarchie (profondeur, densité des concepts locaux, etc.) influe sur la « force » sémantique de cette arête, comme nous allons le montrer dans les sections suivantes.

1.2.3.2.2 Sussna

L'approche de [Sussna, 1993] est basée sur la même idée que celle de Zhong (voir section 1.2.3.2.3 suivante) ou Wu et Palmer (voir section 1.2.3.2.4) :

> Soit deux couples de concepts séparés par le même nombre d'arêtes (*i.e.* même longueur du plus court chemin). Alors les concepts du couple le plus profond (*i.e.* le plus éloigné de la racine) sont les plus proches sémantiquement.

Ceci ajoute une hypothèse au fait que la distance dépend uniquement de la distance entre les concepts. On obtient ainsi que même à distance fixe dans le graphe, la distance sémantique peut changer. Cette hypothèse se justifie par le fait que plus un nœud est profond, plus il est spécialisé donc plus il est représentatif d'une notion précise (idée que l'on retrouve à la base des mesures sur la théorie de l'information, section 1.2.3.3). La formule de distance de Sussna est donc dépendante de la profondeur des nœuds dans l'arbre en plus de la distance en nombre de nœuds. De plus, il cherche à différencier les différents types de relation. Pour cela, il attribue pour chaque relation r un poids (ou une plage de poids $[min_r, max_r]$) en fonction du type de relations qu'elle représente :

1. Pour une antonymie : $min_r = max_r = 2,5$.
2. Pour une hyperonymie, hyponymie, holonymie et méronymie[7] : $min_r = 1$ et $max_r = 2$.
3. Pour une synonymie : $min_r = max_r = 0$.

Il définie alors le TSF (*i.e. type-specific fanout*) d'une arête typant une relation r entre deux concepts x et y. Le TSF est censé représenter le *degré de dilution* de ce type d'arête pour le concept x *i.e.* plus un concept possède d'arêtes partantes du même type, moins ce type fournit d'informations sur ce nœud et inversement. Il est calculé de la façon suivante :

$$w(c_1 \rightarrow_r c_2) = max_r - \frac{max_r - min_r}{n_r(c_1)}$$

avec $n_r(c_1)$ la fonction comptant le nombre d'arêtes du type r partant de c_1. A noter que Sussna considère que les relations entre concepts ne sont pas symétriques : dans la plupart des cas, deux relations inverses n'ont pas le même poids (ou la même plage de poids) et donc si r' est l'opposé de r, $w(c_1 \rightarrow_r c_2) \neq w(c_2 \rightarrow_{r'} c_1)$.

A partir des poids, Sussna définit la distance entre *deux concepts directement reliés* par[8] :

$$dist'_{SUS}(c'_1, c'_2) = \frac{w(c_1 \rightarrow_r c_2) + w(c_2 \rightarrow_{r'} c_1)}{2 \times min[depth(c_1), depth(c_2)]}$$

[7] L'expérience montre que l'ajustement de ce paramètre dans cette plage de valeurs ne permet pas de gain significatif de performance.

[8] Dans la formule initiale, le dénominateur est juste $2d$ où d est la profondeur de l'*arête* (c'_1, c'_2). Pour calculer la profondeur d'une arête, prenons en exemple un cas particulier. Si x est un nœud relié avec *root*, alors la profondeur de $(root, x)$ est de 0, ce qui constitue le minimum des deux profondeurs et s'étend récursivement.

Ceci est en fait un poids associé à chaque couple (x, y). Pour calculer la distance entre 2 concepts quelconques c_1 et c_2, la formule considère le plus court chemin entre c_1 et c_2, noté $sp(c_1, c_2)$. La formule est la somme des distances $dist'$ entre deux concepts consécutifs sur tous les concepts du plus court chemin :

$$dist_{SUS}(c_1, c_2) = \sum_{(x',y') \in sp(c_1,c_2)} dist'_{SUS}(x', y')$$

Bien que cette formule utilise en théorie différents types de relation (et non uniquement la hiérarchie), elle n'a pas été évaluée sur ce point en pratique. De plus, à partir de la version 1.5 de WordNet, cette formule n'est plus efficace (apparemment dû au fait que les niveaux de WordNet sont plus homogènes qu'auparavant) [Budanitsky et Hirst, 2006]. Cette mesure reste néanmoins intéressante à connaître, car c'est la première à introduire l'idée que la profondeur des concepts peut jouer un rôle sur la distance.

1.2.3.2.3 Zhong

Zhong travaille sur les moteurs de recherche sémantique. Comme Sussna, Zhong utilise le principe de profondeur qui veut que la profondeur d'un nœud est représentative de la spécificité d'un concept. Pour ce faire, Zhong définie un coefficient (le *milestone*) pour chaque nœud n de sa hiérarchie [Zhong *et al.*, 2002]. Cette valeur décroît en fonction de la profondeur et rentre en compte dans la formule finale :

$$milestone(n) = \frac{1}{k^{depth(n)+1}}$$

où k est un paramètre permettant d'intensifier ou de diminuer la rapidité d'évolution du *milestone* en fonction de la profondeur. En pratique, k est toujours instancié à 2 (tel qu'utilisé entre autre dans Corese [Corby *et al.*, 2004]).

La distance entre deux concepts c_1 et c_2 est alors définie par le milestone de c_1 et c_2 et par celui de leur plus proche parent commun $ccp(c_1, c_2)$. On obtient alors :

$$dist_{ZHO}(c_1, c_2) = 2 * milestone(ccp(c_1, c_2)) - (milestone(c_1) + milestone(c_2))$$

On remarque qu'en fait cette formule calcule une distance entre c_1 et $ccp(c_1, c_2)$ (avec le calcul $milestone(ccp(c_1, c_2) - milestone(c_1))$ puis une distance entre c_2 et $ccp(c_1, c_2)$ (avec le calcul $milestone(ccp(c_1, c_2) - milestone(c_2))$ et les additionne, permettant de faire l'analogie entre cette distance et une distance métrique pondérée.

1.2.3.2.4 Wu et Palmer

[Wu et Palmer, 1994] travaillent sur la traduction automatique de l'anglais vers le Mandarin. Leur source de connaissance est WordNet. Leur formule est assez proche de l'idée de Zhong, à savoir l'utilisation du plus proche parent des deux concepts x et y et de sa profondeur.

La distance sémantique s'énonce :

$$dist_{WP}(c_1, c_2) = \frac{len(c_1, ccp(c_1, c_2)) + len(c_2, ccp(c_1, c_2))}{len(c_1, ccp(c_1, c_2)) + len(c_2, ccp(c_1, c_2)) + 2 \times depth(ccp(c_1, c_2))}$$

Cette distance utilise encore une fois l'hypothèse de profondeur de Sussna. En effet, si l'on fixe l'ensemble des longueurs entre deux concepts (*i.e.* $len(c_1, ccp(c_1, c_2))$ et $len(c_2, ccp(c_1, c_2))$ sont alors constants), la distance diminue au fur et à mesure que la profondeur augmente (*i.e.* le fait que $depth(ccp(c_1, c_2))$ soit présent au dénominateur diminue la distance quand la profondeur augmente). Cette mesure est ainsi sûrement l'une des plus précises dans la catégorie des mesures à longueur de chemin, mais nous verrons dans les sections suivantes que les mesures utilisant la théorie de l'information obtiennent encore de meilleurs résultats.

1.2.3.2.5 Jarmasz & Szpakowicz

Cette distance est basée sur le thesaurus Roget's (voir section 1.2.2.1). La distance est calculée en fonction du niveau dans la hiérarchie du plus proche parent des deux concepts (*i.e.* le *ccp*) [Jarmasz et Szpakowicz, 2003]. C'est une méthode discrète, renvoyant un nombre pair entre 0 et 16. Il existe donc 9 situations possibles :

$dist_{PEN} =$

- 0 : Même groupe
- 2 : Même paragraphe
- 4 : Même classe de langage
- 6 : Même entête
- 8 : Même groupe d'en-tête
- 10 : Même sous-section
- 12 : Même section
- 14 : Même classe
- 16 : Dans le thésaurus

Ainsi, plus un niveau est profond, plus son score est élevé. Autrement dit, la distance entre deux concepts dépend d'un coefficient calculé en fonction de la profondeur du plus proche parent des deux concepts (le *ccp*).

1.2.3.3 Théorie de l'information (node-based approach)

Ces mesures sont les plus utilisées actuellement et considérées comme les plus efficaces pour la similarité entre concepts. Elles reposent sur le principe d'une fonction de pondération appliquée à chaque nœud : la fonction IC. La description du principe de cette fonction et des différentes méthodes pour la calculer sont donnée dans la prochaine section. Les sections suivantes présentent les mesures les plus classiques de la littérature utilisant cette fonction.

1.2.3.3.1 Quantité d'informations d'un concept, la fonction IC

Les mesures basées sur la théorie de l'information [Shannon et Weaver, 1948] reposent sur un calcul de la *quantité d'information* (*information content, IC(c)*) que représente un concept c. Ce poids IC doit être recalculé à chaque changement de la base de connaissance. La première méthode proposée (et la plus classique encore actuellement) est la méthode de [Resnik, 1995]. Pour Resnik, il faut d'abord estimer la *probabilité* d'un concept en comptant ses occurrences dans un corpus de texte (historiquement le *Brown Corpus of American English*). Si un concept c apparaît, il lui rajoute une occurrence ainsi qu'à tous les concepts ancêtres (exemple de conséquence, le nœud racine a forcement une probabilité de 1). Un

nœud a ainsi une probabilité plus faible s'il est profond, ce qui reprend l'hypothèse du Sussna sur l'importance de la profondeur du nœud (section 1.2.3.2.2). En effet, plus un nœud est profond plus il est spécifique, donc il semble logique que sa probabilité d'apparition soit plus faible. L'avantage de la méthode basée sur corpus est que la probabilité des nœuds n'est pas uniforme en fonction de la profondeur. Ce qui est à priori plus informatif sur la quantité d'information qu'apporte un concept. Ainsi, la probabilité d'un concept quelconque c est :

$$p(c) = \frac{\sum_{n \in words(c)} freq(n)}{N}$$

avec $words(c)$ l'ensemble des mots subsumant le concept c et N le nombre de mots présents dans le corpus et dans l'ontologie. Cette formule reste actuellement la référence de calculs de probabilités d'un concept sur corpus.[9]

Ensuite, Resnik reprend le principe de la théorie de l'information : plus un concept est probable, moins le fait qu'il soit présent donne de l'information, donc la quantité d'information est dépendante logarithmiquement de la probabilité [Shannon et Weaver, 1948]. Autrement dit, la formule $IC(c)$ des Resnik est donnée par :

$$IC_{RESNIK}(c) = -\log(p(c))$$

L'inconvénient majeur de cette méthode réside dans l'obligation d'avoir un corpus pour calculer la probabilité. Récemment, Nuno Seco a proposé une formule permettant de calculer la probabilité d'un concept en se basant sur la structure de la hiérarchie (en tenant compte de la profondeur et de la densité) [Seco *et al.*, 2004]. L'idée est simple et part de l'intuition de Resnik selon laquelle plus un concept est profond, plus il est spécifique et donc plus il véhicule d'informations. Seco remarque, que plus un concept est profond, moins il a de sous-classes (globalement et non localement), d'où son idée de calculer la quantité d'information d'un concept, sans corpus, en fonction du nombre de sous-classes. Le nombre de sous-classes est, par construction, fonction de :

- La profondeur du nœud : en effet, le comptage des sous-classes étant ici récursif, plus un nœud est haut dans la hiérarchie, plus il possède d'enfants qui ont aussi des enfants, etc. A l'inverse, plus un nœud est profond, moins il possède de sous-classes. Ceci correspond bien à un identifiant de la spécificité du concept.
- La densité : un sous-arbre comportant énormément de fils en moyenne sera considéré comme plus dense sémantiquement. Autrement dit, deux nœuds « frères » seront plus proches si ils possèdent beaucoup de « frères » et inversement.

Sa formule est ainsi la suivante :

$$IC_{SECO}(c) = 1 - \frac{\log(hypo(c) + 1)}{\log max_{WN}}$$

[9] Notons néanmoins que divers travaux ont cherché à l'améliorer, notamment en tenant compte de la répartition de l'apparition des mots en fonction des différents documents, l'idée étant qu'un concept apparaissant dans un seul document est plus informatif qu'un concept apparaissant dans tous les documents [Richardson et Smeaton, 1995, Sayed *et al.*, 2008].

où $hypo(c)$ représente le nombre de sous-classes de c, max_{WN} le nombre total de concepts dans la base (indice WN car Seco utilise WordNet).

Notons qu'en pratique, le calcul de la quantité d'information d'un concept est fortement dépendant de la structure de l'ontologie considérée. En effet, la méthode de Resnik par exemple est une méthode « bottom-up », au sens où le calcul du score d'un concept dépend des occurrences des concepts subsumés. De même, l'algorithme de Seco est un algorithme bottom-up. D'autre part, la méthode de Zhong (section 1.2.3.2.3) de calcul du « milestone » peut être généralisée pour le calcul de la quantité d'information du concept. La pondération de Zhong devient alors une méthode « top-down », car le score n'utilise que la profondeur d'un concept pour calculer sa quantité d'information. Dans le cadre de cette thèse, nous n'étudierons pas les détails de choix pour la formule IC, cette étude ayant été faite dans la thèse récente d'Emmanuel Blanchard [Blanchard, 2008].

Il est important de comprendre que la fonction IC est un outil nécessaire pour définir une mesure sémantique basée sur la théorie de l'information, mais indépendant de ces formules sémantiques. En ce sens, les mesures sémantiques seront définies sur la notion de fonction IC, mais pas sur une version de cette fonction en particulier. Ainsi en pratique, une mesure sémantique basée sur la théorie de l'information est un couple composé du choix d'une fonction IC et du choix d'une mesure sémantique. Pour évaluer l'apport d'une fonction IC par rapport à une autre, on fixe une mesure sémantique donnée, puis on change la fonction IC pour évaluer un nouveau résultat. Par exemple, l'évaluation comparative faite par Seco de sa fonction IC est donnée en section 1.2.4 (en fixant les mesures de Resnik, Lin et Jiang & Conrath).

1.2.3.3.2 Resnik

Si Resnik a proposé le concept de fonction IC, il a aussi proposé la première formule exploitant cette fonction [Resnik, 1995]. Il travaille uniquement sur la hiérarchie de WordNet. L'idée de Resnik est de dire que si deux concepts sont proches sémantiquement, alors le plus proche parent commun des deux est proche d'eux et donc sa quantité d'information est un bon indicateur. En effet, si deux nœuds n'ont en commun qu'un nœud très haut dans la hiérarchie, alors ils n'ont que peu d'attributs en commun, donc une similarité faible. La mesure de similarité de Resnik est définie ainsi :

$$sim_{RES}(c_1, c_2) = IC(ccp(c_1, c_2))$$

Le principal défaut de la mesure de Resnik est qu'il ne considère pas les quantités d'information des concepts c_1 et c_2. Il ne tient ainsi pas compte de la longueur du chemin à ce ccp et ainsi de la profondeur des concepts c_1 et c_2.

1.2.3.3.3 Leacock et Chodorow

Leacock et Chodorow cherche à éviter le calcul de la fonction IC, mais en gardant le concept sur la théorie de l'information [Leacock et Chodorow, 1998] :

$$sim_{L\&C}(c_1, c_2) = -\log\left[\frac{len(c_1, c_2)}{2 \times MAX}\right]$$

avec *len* la fonction renvoyant la longueur du plus court chemin dans le graphe entre $c_1$et c_2 et MAX la pronfondeur maximale de la hiérarchie de la source de connaissances.

Le dénominateur $2 \times MAX$ représente la longueur maximale d'un chemin dans une hiérarchie de concepts (au pire cas, on fait le voyage d'une des feuilles les plus profondes vers la racine, puis on continue vers une autre feuille parmi les plus profondes). Ainsi, le contenu du logarithme est normalisé dans l'intervalle $[0, 1]$ (comme la probabilité utilisée par le calcul de Resnik de la fonction IC). Ainsi, l'idée sous-jacente à la mesure de Leacock et Chodorow est d'approximer la probabilité en tenant compte de la longueur du chemin. Néanmoins, comme nous en avons discuté dans les sections précédentes, utiliser seulement la longueur du chemin est incorrect, car on ne tient pas compte de la densité ou la profondeur des concepts.

1.2.3.3.4 Jiang et Conrath

Héritier de Rada et Resnik, Jiang et Conrath définissent leur mesure de similarité comme « similarité basée sur les longueurs de chemins avec la théorie de l'information comme facteur de décision » [Jiang et Conrath, 1997]. Ils s'appuient sur les idées suivantes :

- La force d'un lien : La force d'un lien correspond à son degré d'importance dans le graphe. Elle est proportionnelle à la probabilité conditionnelle d'avoir une instance d'un concept fils c_i sachant que l'on a l'instance de son concept parent p, autrement dit : $P(c_i/p) = \frac{P(c_i \cap p)}{P(p)} = \frac{P(c_i)}{P(p)}$. Suivant la théorie de l'information, ils définissent la *force du lien* LS ainsi :

$$LS(c_i, p) = -\log P(c_i/p) = IC(c_i) - IC(p)$$

- La densité local, la profondeur du nœud et son type : Ils définissent le poids $wt(c, p)$ d'une arête entre un fils c et un parent p :

$$wt(c,p) = \left(\beta + (1-\beta) \times \frac{\overline{E}}{E(p)}\right) \left(\frac{depth(p)+1}{depth(p)}\right)^{\alpha} LS(c,p)T(c,p)$$

  avec $E(p)$ le nombre d'arête du type (c, p) partant de p, $\overline{E}$ la densité moyenne de la hiérarchie et $T(c, p)$ un facteur lié au type de lien. Les paramètres α ($\alpha \geq 0$) et β ($0 \leq \beta \leq 1$) contrôlent le degré d'importance à apporter à la profondeur du nœud et la densité. Ces contributions deviennent négligeables quand α tend vers la valeur 0 et β tend vers la valeur 1.

Reprenant alors l'idée du chemin le plus court dans la hiérarchie, la distance de Jiang et Conrath est :

$$dist_{JC}(c_1, c_2) = \sum_{c \in sp(c_1,c_2) \setminus ccp(c_1,c_2)} wt(c, parent(c))$$

Si l'on choisit les paramètres simplifiés $\alpha = 0$, $\beta = 1$, la mesure de distance $dist_{JC}$ de Jiang & Conrath est alors définie par :

$$dist_{JC}(c_1, c_2) = \sum_{c \in sp(c_1,c_2) \setminus ccp(c_1,c_2)} [IC(c) - IC(parent(c))] \times T(c, parent(c))$$

Comme toute mesure de similarité, elle se limite à la relation de subsomption, mais elle offre la possibilité de prendre en compte des valuations différentes pour chaque arête de l'arbre des concepts (les $T(c, parent(c))$). Cette fonctionnalité n'a jamais été prise en compte à notre connaissance. Nous avons cherché à utiliser cette propriété dans notre formule que nous définirons dans le chapitre 3. Dans l'article initial [Jiang et Conrath, 1997], l'évaluation fut en effet effectuée pour un poids d'arête constant ($\forall(c,p),\ T(c,p)=1$). Depuis, les systèmes utilisant la formule de Jiang & Conrath se servent de la version simplifiée sans poids sur les arêtes :

$$dist_{JCsimple}(c_1, c_2) = (IC(c_1) + IC(c_2)) - 2 \times IC(ccp(c_1, c_2))$$

1.2.3.3.5 Lin, universal similarity measure

Dans son article, [Lin, 1998] cherche à définir une mesure de similarité indépendante du type de ressource ou du domaine, sur une base mathématique démontrable. En posant informellement $common(A, B)$ la fonction décrivant les attributs communs à A et B et $description(A, B)$ la somme des descriptions complètes des objets A et B, il énonce comme axiomes initiaux :

Axiome 1 : La similarité entre A et B est liée aux informations qu'ils ont en commun de telle sorte que plus il y d'information, plus les concepts sont similaires. On notera $I(common(A, B))$ cette quantité.

Axiome 2 : La similarité entre A et B est liée à la quantité de différences entre les concepts de telle sorte que plus il y a de différences, moins les concepts sont similaires. On notera $I(description(A, B)) - I(common(A, B))$ cette différence.

Axiome 3 : La similarité est de la forme d'une fonction f paramétrée par les informations en commun et les différences : $sim(A, B) = f(I(common(A, B), I(description(A, B)))$

Axiome 4 : Si A et B sont identiques, alors leur similarité est parfaite et égale à 1. Autrement dit, $\forall x\ f(x, x) = 1$ (*i.e.* les informations en communs correspondent à leurs descriptions)

Axiome 5 : La similarité entre A et B si $I(common(A, B)) = 0$ (*i.e.* ils n'ont rien en commun) est de 0. $\forall y > 0, f(0, y) = 0$.[10]

Axiome 6 : Supposons que les deux objets A et B puissent être vus suivant plusieurs perspectives. La similarité peut être calculée suivant les différentes perspectives. La similarité entre deux objets à plusieurs perspectives est la moyenne des similarités de chaque perspective pondérée par la part d'informations que représente cette perspective dans la description des objets :

$\forall x_1 \leq y_1, x_2 \leq y_2$:

$$f(x_1 + x_2, y_1 + y_2) = \frac{y_1}{y_1 + y_2} f(x_1, y_1) + \frac{y_2}{y_1 + y_2} f(x_2, y_2)$$

La formule suivante est une formule simple qui répond à tous ces axiomes :

$$sim(A, B) = \frac{I(common(A, B))}{I(description(A, B))}$$

[10] La cas $y = 0$ est ici impossible et explique l'inégalité strict. En effet, tout concept qui n'est pas une racine a une description et donc la fonction $I(description(A, B))$ ne peut pas être nulle.

En considérant la théorie de l'information énoncée précédemment, la quantité d'information $I(common(A, B))$ est égale à $-\log P(common(A, B))$. On obtient la formule finale de Lin :

$$sim(A, B) = \frac{\log P(common(A, B))}{\log P(description(A, B))}$$

Selon Dekang Lin, il suffit de spécifier le calcul de la probabilité en fonction du domaine pour avoir sa propre mesure de similarité. Il donne comme exemples dans [Lin, 1998], une similarité entre chaîne de caractères, entre deux mots en se basant sur un corpus et enfin entre deux concepts d'une taxonomie.

Lin propose dans son article original une mesure de similarité entre deux concepts WordNet. Pour cela, il définie la quantité d'informations en commun par 2 fois la quantité d'information du plus proche parent des deux ($ccp(c_1, c_2)$)[11] et la quantité d'informations de la description par la somme des descriptions. La probabilité d'un concept est calculée suivant la méthode de Resnik décrite section 1.2.3.3.2. Ceci nous donne la formule de similarité :

$$sim_{LIN}(c_1, c_2) = \frac{2 \times \log p(ccp(c_1, c_2))}{\log p(c_1) + \log p(c_2)}$$

Ce qui peut se réécrire en utilisant la notation IC de Resnik :

$$sim_{LIN}(c_1, c_2) = \frac{2 \times IC(cpp(c_1, c_2))}{IC(c_1) + IC(c_2)}$$

1.2.3.4 Mesures de degré de relation sémantique

Ces mesures sont aussi appelées *mesure relationnelle* [Turney, 2006]. En effet, elles doivent être capables de capturer les liens sémantiques entre des couples tels que *essence-voiture* ou encore *abeille-miel* [Resnik, 1995]. Notre mesure, que nous définirons dans le chapitre 3, se range dans cette catégorie de mesure.

Nous présenterons tout d'abord le problème de la validité sémantique d'un chemin non-hiérarchique. En effet, en utilisant différents types de relation, il n'est pas évident que n'importe quel chemin (au sens théorie des graphes) ait un sens sémantiquement. Cette notion est parfois utilisée comme un outil dans les mesures de degré de relation sémantique. Ensuite, nous présenterons différentes formules considérant de degré de relation sémantique.

1.2.3.4.1 Problème de la validité sémantique d'un chemin non-hiérarchique

L'une des questions importantes lorsque l'on cherche à calculer un score de degré de relation sémantique en utilisant la vision « théorie des graphes » d'une ontologie, est la validation sémantique d'un chemin non-hiérarchique. En effet, ces mesures considèrent différents types de relations (*i.e.* relations non hiérarchique) tel que $hasPart$, $madeWith$, etc. Ainsi, le modèle de connaissance n'est plus un arbre mais un graphe, donc le chemin entre deux concepts donnés n'est plus unique et il existe donc une multitude de chemins entre eux. Cependant, tous ces chemins ne sont pas *sémantiquement corrects* [Aleksovski *et al.*, 2006b,

[11] Les informations en commun sont en double dans les deux concepts, ce qui justifie le facteur 2.

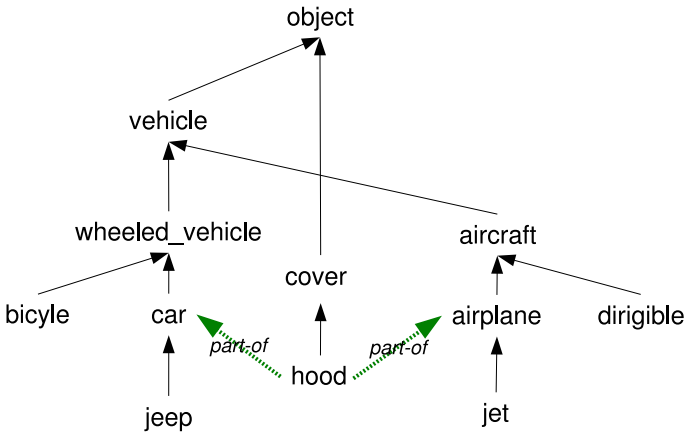

FIG. 1.4 – Exemple d'arbre avec relations

Hirst et St-Onge, 1998]. Il est donc nécessaire d'avoir un ensemble de méthodes et/ou d'heuristiques afin de filtrer les chemins incorrect sémantiquement.

Par exemple, dans les travaux de [Aleksovski *et al.*, 2006b], un chemin est considéré « correct sémantiquement » si et seulement si il n'y a pas d'arête hiérarchique utilisée après une arête non-hiérarchique. Ainsi, le chemin $\{cover \overset{includes}{\longrightarrow} hood \overset{part\text{-}of}{\longrightarrow} airplane \overset{is\text{-}a}{\longrightarrow} aircraft\}$ tiré de la figure 1.4 est incorrect pour les hypothèses d'Aleksovski puisque le lien hiérarchique $(airplane \overset{is\text{-}a}{\longrightarrow} aircraft)$ suit la relation non-hiérarchique $(hood \overset{part\text{-}of}{\longrightarrow} airplane)$.

D'autre part, [Hirst et St-Onge, 1998] proposent sûrement les travaux les plus complets à ce jour sur les contraintes à adopter pour calculer un chemin sémantique correct. Ils dégagent deux types de relations entre deux concepts, la *relation forte* et la *relation standard* (*i.e. medium-strong* ou *regular*). Ces relations sont définies de la manière suivante :

- Deux mots sont reliés par une relation forte si une des conditions suivantes est vérifiée :
  1. Ils font partis du même synset (*e.g. human* et *person*)
  2. Ils sont associés par deux synsets différents connectés par une relation d'antonymie (*e.g. precursor* et *successor*)
  3. Un des mots est un mot composé contenant l'autre mot (*e.g. school* et *private school*)
- Deux mots sont reliés par une relation standard s'il existe un *chemin sémantiquement acceptable* connectant deux synsets associés à ces deux mots. Un chemin est *acceptable* si il ne contient pas plus de cinq liens et rentre dans un des huit patterns décrits dans [Hirst et St-Onge, 1998]. Ces patterns sont construits suivant des expériences et des théories en psychologie suivant les notions de généralisation, spécialisation et coordination. Ils tiennent compte des changements de *direction* du chemin dans le réseau, une direction étant parmi les trois suivantes :
  1. Ascendant (*i.e. upward*).
  2. Descendant (*i.e. downward*).
  3. Horizontal.

Relation	Direction
Also see	Horizontal
Antonymy	Horizontal
Attribute	Horizontal
Cause	Descendante
Entailment	Descendante
Holonymy	Descendante
Hypernymy	Ascendante
Hyponymy	Descendante
Pertinence	Horizontal
Similarity	Horizontal

TAB. 1.2 – Les directions H-D-A de Hirst et St-Onge en fonction du type de relation dans WordNet.

Les directions sont affectées en fonction du type de liens dans WordNet (voir tableau 1.2). Les huit patterns finalement définis sont A, AD, AH, AHD, D, DH, HD, H. Ces patterns peuvent être résumés par les deux hypothèses suivantes :

- R1 : Aucune direction ne doit précéder un lien Ascendant ;
- R2 : Un chemin sémantiquement correct ne doit avoir qu'un seul changement de direction.

  Cette règle possède néanmoins une exception :
- R2' : Un lien Horizontal est permis comme transition entre un lien Ascendant et un lien Descendant.

Il est intéressant de constater que la règle R1 de Hirst & St-Onge contient l'hypothèse d'Aleksovski (la relation hiérarchique étant une relation Ascendante). Pour exemple d'utilisation de ces règles, le chemin entre *jeep* et *jet* sur la figure 1.4 est sémantiquement correct (pattern AD) pour Hirst & St-Onge.

Les sections suivantes présentent quelques formules de degré de relation sémantique. Ces formules sont définies dans les taxonomies augmentées de relations hétérogènes (et sont ainsi parfois amenées à utiliser les hypothèses de chemin que nous venons de présenter).

1.2.3.4.2 Hirst et St-Onge

Hirst et St-Onge [Hirst et St-Onge, 1998] basent ainsi le calcul de leur mesure sur les patterns de chemins décrits à la section précédente. Une fois qu'un chemin sémantiquement correct entre deux concepts est trouvé, alors le degré de relation sémantique de Hirst & St-Onge est défini par :

$$sim_{HIR}(c_1, c_2) = C - len_{HIR}(c_1, c_2) - k \times turns(c_1, c_2)$$

où C et k sont des constantes (dans [Hirst et St-Onge, 1998] elles sont fixées à $C = 8$ et $k = 1$), $turns(c_1, c_2)$ le nombre de changements de direction dans le chemin associé et $len_{HIR}(x, y)$ le plus court chemin entre les deux mots *en considérant toutes les relations* du tableau 1.2.

Cette mesure est une adaptation de la mesure de Rada pour tenir compte des relations non hiérarchiques. En ce sens, elle a les mêmes défauts, à savoir la non prise en charge de la

densité ou de la profondeur des concepts ainsi que la non utilisation d'une pondération (du type de la fonction IC) pour les nœuds.

1.2.3.4.3 Thieu & Steichen, une extension de la mesure de Jiang & Conrath

La mesure de Jiang & Conrath présentée lors de la section 1.2.3.3.4 est initialement prévue pour modéliser une distance sémantique hiérarchique. Elle est ainsi définie comme la somme des poids de chaque arête sur le plus court chemin entre les deux concepts cibles. Néanmoins, cette formule prévoit une pondération pour l'arête, capacité qui n'a jamais été exploitée directement par les auteurs qui l'ont simplifiée lors de leur évaluation en choisissant un poids unique de 1 (*i.e.* le poids n'a alors aucune importance). Ainsi, dans [Thieu *et al.*, 2004], les auteurs ont essayé de profiter de cette propriété pour étendre la mesure de Jiang & Conrath au degré de relation sémantique, en utilisant des poids différents en fonction des types de liens.

Pour rappel, la valeur dans un chemin donné d'une arête entre deux sommets c et p (p étant le concept subsumant c dans la hiérarchie) selon Jiang & Conrath est définie par :

$$wt(c,p) = \left(\beta + (1-\beta) \times \frac{\overline{E}}{E(p)}\right) \left(\frac{depth(p)+1}{depth(p)}\right)^{\alpha} LS(c,p)T(c,p)$$

avec $T(c,p)$ le poids associé à l'arête $\{c,p\}$ et $LS(c,p)$ étant la différence des quantités d'information IC des deux concepts :

$$LS(c,p) = |IC(c) - IC(p)|$$

La distance de Jiang & Conrath est alors définie comme la somme des $wt(c,p)$ sur le plus court chemin entre les concepts voulus. Cette formule ne peut pas être utilisée immédiatement comme mesure relationnelle, à cause de l'utilisation du poids IC. En effet, ce poids est calculé hiérarchiquement (section 1.2.3.3.1) et calculer la différence entre deux valeurs de la fonction IC sans tenir compte de la hiérarchie (*i.e.* entre deux concepts reliés par une relation hétérogène) n'a ainsi pas de sens. Il est donc nécessaire de définir une nouvelle fonction $LS(c,p)$ pour le cas où le lien entre c et p n'est pas hiérarchique.

[Thieu *et al.*, 2004] propose ainsi une nouvelle définition de la fonction $LS(c,p)$ basée sur une moyenne des IC des ancêtres du concept fils :

$$LS(c,p) = \frac{\sum_{c_i \in Ancestor(c)} |IC(c) - IC(c_i)|}{|Ancestor(c)|}$$

tel que $Ancestor(c)$ représente l'ensemble des concepts subsumant le concept c (ce qui inclut le concept p). Le point intéressant de cette formule est qu'elle n'est pas symétrique : elle n'utilise que le concept c, qui est le fils du concept p. Ceci induit donc pour ces auteurs que le degré de relation sémantique n'est pas une mesure symétrique, ce qui semble logique étant donné que rien n'affirme initialement qu'une relation hétérogène possède un inverse.

Cette formule n'a été évaluée que sur des contextes très particuliers et des ontologies construites par ses auteurs (il existe ainsi deux évaluations publiées à notre connaissance [Thieu *et al.*, 2004, Steichen *et al.*, 2006]), ce qui rend difficile voire impossible une interpré-

tation générale de ces résultats. De plus, il semble difficile de justifier l'emploi de la moyenne des valeurs de la fonction IC pour considérer une relation hétérogène, étant donné que la fonction IC a une construction totalement hiérarchique. D'ailleurs, les auteurs ne justifient pas du point de vue théorique cette nouvelle proposition de la formule $LS(c, p)$ dans leurs articles.

1.2.3.4.4 Cho

La formule de [Cho *et al.*, 2003] reprend une partie des idées exposées dans les mesures de similarité précédentes avec quelques hypothèses nouvelles, à savoir :

- L'importance du plus proche parent commun (le *ccp*)
- Les différents types de liens (hiérarchie, méronymie, etc.) sont importants et leur force doit être déterminée par une pondération.
- La distance entre deux nœuds consécutifs n'est pas forcement équivalente pour tous les nœuds.
- Plus la profondeur est importante, plus deux nœuds consécutifs sont proches.
- Un modèle de représentation des connaissances n'ayant pas une structure uniforme, il faut tenir compte de la densité des concepts.

Ils proposent ainsi la formule suivante, dans laquelle chacun des paramètres précédent possède une pondération propre.

$$sim_{CHO}(c_i, c_j) = IC(ccp(c_i, c_j)) \times \left[D_{i \to j} \times \sum_{\{c_k, c_l\} \in sp(c_i, c_j)} W_{k \to l} \times d_{k \to l} \times depth(c_k) \right]$$

avec $D_{i \to j}$ « le facteur de distance » entre c_i et c_j, $W_{k \to l}$ la fonction poids associée au type de lien entre c_k et c_l et $d_{k \to l}$ la « fonction densité » entre c_k et c_l.

Comme on l'observe, le défaut majeur de cette formule est quelle utilise énormément de poids différents et qu'aucune piste n'est donnée pour évaluer leurs valeurs respectives. L'évaluation de l'article est basée sur le test de Miller & Charles, qui n'est pas le test idéal pour une mesure de relation sémantique (voir section 1.2.4.1). Les résultats trouvés sont d'ailleurs inférieurs aux résultats obtenus par les mesures de similarités classique, et montrent peu d'apport de l'approche.

1.2.3.4.5 Yang

La formule de [Yang et Powers, 2005] se base sur le principe d'une formule paramétrée à base d'apprentissage supervisé. La structure générale de la formule est pondérée et basée sur le comptage du nombre d'arêtes sur le chemin le plus court. Les auteurs introduisent ainsi une formule paramétrée, et proposent d'utiliser des similarités rentrées à la main pour évaluer les différents paramètres. La formule est donc un simple produit de coefficients en fonction de la longueur du plus court chemin :

$$sim_{YANG}(c_1, c_2) = \begin{cases} \alpha_t \prod_{i=1}^{len(c_1, c_2)} \beta_{t_i} & \text{Si } len(c_1, c_2) < \gamma \\ 0 & \text{Sinon} \end{cases}$$

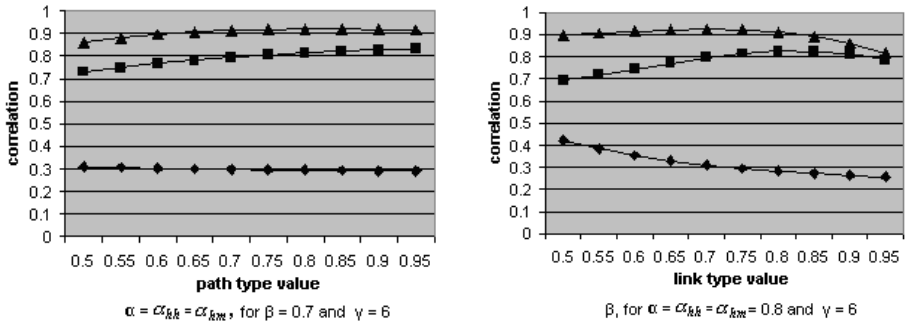

FIG. 1.5 – La recherche des coefficients de Yang et Powers.

où les paramètres correspondent :

- $len(c_1, c_2)$: la longueur du plus court chemin entre c_1 et c_2.
- α_t : un facteur lié au type d'arête t (t pouvant être un lien hiérarchique *is-a* ou meronymique *part-of*).
- β_t : Un facteur de profondeur lui aussi paramétré en fonction du type de lien.
- γ : un seuil au delà duquel la similarité est suffisamment négligeable pour être nulle. Pour l'auteur, il représente « les limitations du système cognitif humain ».

L'objectif ensuite est de prendre un ensemble de valeurs proposées par des humains, puis de les rentrer dans le système pour trouver les paramètres qui rendent le mieux les résultats humains (figure 1.5).

De la même manière que Cho, l'évaluation donne des résultats discutables. S'ils arrivent en effet sur leur base d'apprentissage à reproduire le score des humains, ils ne proposent pas de tests sur un autre ensemble de couples, ce qui réduit fortement « la preuve » de leur formule.

1.2.3.5 Conclusion

Nous pouvons synthétiser les différentes distances dans le tableau 1.3. Soulignons que :

- Le paramètre *plus court chemin* peut apparaître sous trois formes :
  1. Direct : La mesure de distance tient compte alors du plus court chemin direct entre les deux concepts.
  2. ccp direct : La mesure de distance ne calcule pas le plus court chemin directement, mais en fonction de la distance de chacun des deux concepts au plus proche parent.
  3. ccp : La mesure de distance ne tient pas compte du plus court chemin, mais seulement de l'existence du plus proche parent et de certains de ses paramètres (profondeur, densité, etc.).
- Toutes les formules utilisant un precalcul de probabilité basé sur un corpus tiennent par construction compte de la profondeur, puisque plus un concept est profond, plus sa probabilité est grande en moyenne (avec probabilité de la racine à 1).
- La densité est prise en compte si la formule contient des références aux liens et à leur quantité et/ou type autour des deux concepts. La distance Hirst et St Onge, sans en parler directement, prend en compte le nombre de changements de direction (*i.e.* le

	Mesures	Precalcul	Modèle initial	Plus court chemin	Profondeur	Densité
Longueur du chemin	Rada		MeSH	direct		
	Sussna		WordNet	direct	+	+
	Zhong		Graphes Conceptuels	ccp direct	+	
	Wu & Palmer		WordNet	ccp direct	+	
	Jarmasz & Szpakowicz		Roget's Thesaurus	direct		
Théorie Information	Resnik	IC	WordNet	ccp	+	
	Leacock & Chodorow		WordNet	direct		
	Jiang & Conrath	IC	WordNet	ccp direct	++	+
	Lin	IC	WordNet	ccp	+	
Degré de relation	Hirst & St Onge		WordNet	direct		
	Thieu & Steichen	IC	Ontologie médical	ccp	++	+
	Cho	IC et Coeffs.	WordNet	ccp et direct	+	+
	Yang	Coeffs.	WordNet	N/A	+	

TAB. 1.3 – Résumé des formules de distance sémantique. Le symbole « + » rend compte de l'importance de ce facteur (colonne) dans la mesure considérée (ligne).

nombre de changements de type d'arêtes), ce qui constitue une forme d'intégration de la densité.

A noter qu'une catégorisation équivalente des mesures sémantiques a été proposée par Emmanuel Blanchard dans [Blanchard *et al.*, 2005]. Dans sa thèse de doctorat, il propose ainsi un modèle générique permettant de considérer une grande partie des mesures présentées précédemment comme des spécialisations d'un modèle de similarité unique [Blanchard, 2008].

1.2.4 Évaluation

Cette section a pour but de présenter les différentes évaluations (de la littérature ou faites à l'occasion de cette thèse) de ces différentes mesures. Nous commencerons d'abord par présenter quelques protocoles de test classiques. Puis nous présenterons les évaluations effectuées par les auteurs de mesures eux-mêmes. Enfin, nous présenterons notre propre évaluation synthétique, refaite à l'occasion de cette thèse sur les mesures les plus classiques de la littérature.

1.2.4.1 Le protocole : le test de Miller et Charles

La première étude sur une évaluation de la distance sémantique des humains a été réalisée par [Rubenstein et Goodenough, 1965]. Cette étude a reporté les "jugements synonymiques" (*i.e. synonymy judgement*) de 51 personnes sur un ensemble de 65 couples de mots. Les personnes devaient noter leur "jugement" sur une échelle de 0.0 à 4.0. La moyenne de tous les volontaires donne un vecteur de 65 valeurs numérique représentant (si les hypothèses statistique sont respectées) la façon dont un utilisateur ordinaire calcule la similarité entre les mots (*i.e.* un représentant de la similarité personnelle d'un groupe d'individu).

Par la suite, [Miller et Charles, 1991] ont refait l'expérience sur un sous ensemble de 30 couples de mots tiré de l'ensemble initial (10 notés en 3 et 4, 10 notés entre 1 et 3 et 10 noté entre 0 et 1). 38 personnes ont participé au test. Miller et Charles utilisent le coefficient de corrélation pour déterminer la corrélation entre leurs résultats et les résultats de Rubenstein et Goodenough. Ce coefficient permet de calculer la corrélation entre deux vecteurs de valeurs. La corrélation est un nombre entre -1 et 1 défini tel que :

- Proche de -1 : les deux vecteurs sont inverses,
- Autour de 0 : les deux vecteurs sont indépendants,
- Proche de 1 : les deux vecteurs sont dépendants.

Soit deux vecteurs de longueur k $\{x_1, x_2, ..., x_k\}$ et $\{y_1, y_2, ..., y_k\}$, alors le facteur de corrélation de Pearson est définie formellement de la façon suivante :

$$\rho = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Miller et Charles obtiennent une corrélation de $0,91$. Le calcul de la corrélation entre deux vecteurs de valeurs données par des utilisateurs ordinaires a été refait plusieurs fois (*e.g.* [McHale, 1998, Budanitsky et Hirst, 2006, Jiang et Conrath, 1997, Resnik, 1995]) et ne descend jamais en dessous de $0,88$ (valeur de Jiang et Conrath). La quasi totalité des travaux présentés dans la section 1.2.3 finissent par une évaluation par rapport aux 30 couples choisis par Miller et Charles (la plupart comparent aussi à certaines autres mesures classiques, comme celle de Rada ou plus régulièrement maintenant celle de Resnik). Avant 2000, les évaluations basées sur WordNet n'utilisaient que 28 des 30 couples de Miller et Charles, car le mot *woodland* n'apparaissait pas encore.

Ce protocole et ses résultats a permis de fixer une base de travail pour l'ensemble de la communauté de recherche sur la distance sémantique. En effet, **si une mesure de distance atteint le coefficient de $0,91$, alors elle sera représentative de la distance réelle d'un jugement humain**. En effet, cette distance sera dans la marge d'erreur acceptable entre deux évaluations humaines. Il ne sera donc pas possible de faire la différence entre le résultat produit par un humain et le résultat produit par la machine.

Cependant, le test de Miller & Charles est basé sur le « jugement synonymique », il est loin d'être évident qu'il soit le meilleur choix possible pour tester une mesure de degré de relation sémantique. La plupart des couples de mots choisis n'ont ainsi pas de relation fonctionnelle entre eux. En effet, il était explicitement demandé aux personnes participant à l'expérience d'évaluer la *synonymie* entre les mots. De plus, seuls quelques couples (tel que « journey - car » ou « furnace - stove ») semblent pouvoir être reliés par des relations fonctionnelles.

Il est d'ailleurs intéressant de remarquer que dans les tests récents basés sur la similarité (tel que [Budanitsky et Hirst, 2006]), ces couples sont ceux qui diffèrent le plus du score de référence de Miller & Charles. Malgré tout cela, étant donné que « la similarité sémantique représente un cas spécial de degré de relation sémantique » [Resnik, 1995], une mesure de degré de relation sémantique doit au minimum fonctionner sur le test de Miller & Charles, même si le test ne permet pas alors de valider les aspects relationnels de la mesure.

1.2.4.2 WordSimilarity-353

Comme nous le disions précédemment, le test de Miller & Charles n'est pas le test le plus adapté pour tester les relations fonctionnelles présentes entre deux concepts. Pour tester une mesure de degré de relation sémantique, il est donc nécessaire de trouver un nouvel ensemble de test plus adapté. Le test le plus pertinent, à notre connaissance, à ce sujet est le test WordSimilarity-353[12] [Finkelstein *et al.*, 2001] qui propose majoritairement des couples connectés relationellement (*e.g.* « computer - keyboard », « telephone - communication », etc.).

Le test WordSimilarity-353 est un test proposé par Finkelstein pour pallier les problèmes du test de Miller & Charles. En particulier les couples de Miller & Charles sont surtout construits pour les tests de similarité et ne contiennent que peu de couples pouvant exploiter des relations fonctionnelles. Ce test est ainsi constitué de 353 couples de mots, mais il inclut l'ensemble de Miller & Charles, ce qui implique qu'il n'y a en pratique « que » 323 nouveaux couples de mots.

Il est en réalité composé d'une fusion de deux ensembles de tests. Le premier ensemble contient 153 couples (dont les couples de Miller & Charles) et a été noté par 13 personnes. Le deuxième ensemble contient 200 couples de mots, et a été évaluée par 16 personnes. Tous les sujets, dans les deux expériences, possèdent un niveau d'anglais « proche de la langue maternelle ». Les instructions étaient d'évaluer *le degré de relation sémantique* des couples de mots en donnant une note entre 0 (mots non reliés) et 10 (mots très reliés ou identiques).

Ce test a été effectué à l'université de Jerusalem, ce qui explique qu'un certain nombre (faible néanmoins) de couples soient très orientés par les évènements politiques du pays (« Jerusalem - Israel », « Jerusalem - Palestinian », « Arafat - terror », « Arafat - peace », etc.). Certains auteurs soulignent ainsi le manque de pertinence dans le choix des couples de ce test [Strube et Ponzetto, 2006]. Nous pensons pour notre part qu'étant donné le faible nombre et l'aspect minoritaire de ces couples « polémiques », le test reste néanmoins valable et intéressant à étudier.

1.2.4.3 Synthèse des évaluations de la littérature

Le tableau 1.4 résume les différentes valeurs de corrélation trouvées (sur WordNet pour le test de Rubenstein & Goodenough et le test de Miller & Charles) en fonction des auteurs faisant l'évaluation, à savoir Budanitsky [Budanitsky et Hirst, 2006], Jarmasz et Szpakowicz avec la bibliothèque Pedersen [Jarmasz et Szpakowicz, 2003], Patwardhan et Pedersen avec la bibliothèque Pedersen aussi [Patwardhan et Pedersen, 2006], Jiang et Conrath [Jiang et Conrath, 1997], Lin [Lin, 1998].

[12] `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html`

	Miller & Charles					Rub. & Good.		
Mesures	J-C	Lin	Jar.	Pat.	Bud.	Jar.	Pat.	Bud.
Rada	0,600	–	–	–	–	0,787	–	–
Wu & Palmer	–	0,803	–	–	–	–	–	–
Jarmasz & Szpakowicz	–	–	**0,878**	–	–	**0,818**	–	–
Resnik	0,794	0,795	0,787	0,72	0,774	0,800	0,72	0,778
Leacock & Chodorow	–	–	0,832	0,74	0,816	0,852	0,77	0,838
Jiang & Conrath	**0,828**	–	0,696	0,73	0,850	0,731	0,73	0,781
Lin	–	**0,834**	0,846	0,70	0,829	0,834	0,72	0,819
Hirst & St Onge	–	–	0,689	–	0,744	0,732	–	0,786

TAB. 1.4 – Résumé des corrélations entre humain et machine sur WordNet avec le test de Rubenstein & Goodenough et le test de Miller & Charles. Les lignes correspondent à l'auteur de la formule et les colonnes à l'auteur de l'évaluation (par ordre chronologique de publication). Dans la plupart des cas, les noms apparaissent aussi bien en ligne qu'en colonne, car les auteurs de formules recalculent les corrélations dans leur article sur les formules connues. Les valeurs en gras sont les valeurs obtenues lorsqu'un auteur évalue sa propre mesure.

	McHale	
Mesures	WordNet	Roget's
Resnik	0,791	0,790
Rada	0,664	0,886
Jiang et Conrath	0,828	0,791

TAB. 1.5 – Comparaison entre Roget's et WordNet par McHale, ou comment montrer que WordNet n'est pas toujours la solution à tous les problèmes.

	Seco	
Mesures	IC_{RES}	IC_{SEC}
Resnik	0,77	0,77
Lin	0,80	0,81
Jiang et Conrath	0,81	0,84

TAB. 1.6 – Comparaison entre IC_{SEC} et IC_{RES}. Les résultats sont sensiblement équivalents et montrent que la formule IC_{SEC} sans corpus est efficace.

Mesures	Corrélation		
	Rubenstein & Goodenough	Miller & Charles	WordSimilarity-353
Rada	0,638	0,638	0,249
Resnik	0,804	0,804	0,375
Leacock & Chodorow	0,839	0,779	0,346
Wu & Palmer	0,786	0,737	0,300
Lin	0,836	0,836	0,377
Jiang & Conrath	0,880	0,880	0,362
Hirst & St-Onge	0,847	0,847	0,380

TAB. 1.7 – Corrélation de Pearson pour les mesures les plus classiques de la littérature selon notre bibliothèque de calcul.

Le tableau 1.5 présente une comparaison de trois distances sémantiques classiques faite dans [McHale, 1998] en changeant WordNet par le Roget's Thesaurus, afin d'étudier l'efficacité des formules en elles-mêmes et leurs pertinences sur un changement de base de connaissances.

Seco a défini une nouvelle méthode pour calculer la quantité d'informations d'un concept ([Seco *et al.*, 2004], voir section 1.2.3.3.1). Il a ensuite comparé trois mesures de distances sémantiques en fonction de l'utilisation de sa formule (IC_{SEC}) et de celle de Resnik (IC_{RES}). Les résultats sont donnés dans le tableau 1.6.

1.2.4.4 Évaluation faite dans le cadre de cette thèse

Afin de pouvoir évaluer en toute objectivité la proposition de mesure de degré de relation sémantique que nous proposons dans le cadre de cette thèse (et dont les détails théoriques seront donnés chapitre 3), nous avons donc recalculé l'ensemble des corrélations pour un ensemble de mesures significatives, en utilisant notre propre bibliothèque de mesure sémantique, développée au cours de cette thèse (les détails d'implémentation seront donnés dans le chapitre 6).

Pour calculer nos scores sémantiques, nous avons utilisé comme modèle de représentation des connaissances la sous-partie *nom* de la base lexical WordNet 3.0 [Fellbaum, 1998]. Pour le test WS-353, nous avons retiré 9 couples de mots lorsqu'au moins l'un des mots du couple n'existait pas comme nom dans WordNet (*e.g.* le couple « fighting-defeating »). La quantité d'information IC d'un concept a été calculée selon la formule de Seco [Seco *et al.*, 2004]. Les références humaines de Rubenstein & Goodenough ainsi que celle de Miller & Charles sont celles calculées dans [Budanitsky et Hirst, 2006].

Nous avons considéré pour notre évaluation 6 mesures de similarité sémantique classique ([Rada *et al.*, 1989], [Resnik, 1995], [Leacock et Chodorow, 1998], [Wu et Palmer, 1994] ainsi que [Lin, 1998] et [Jiang et Conrath, 1997]) et une mesure de degré de relation sémantique ([Hirst et St-Onge, 1998]).

Les résultats des corrélations sont donnés dans le tableau 1.7.

1.2.4.5 Discussion des résultats

Le premier point important à noter est les écarts pour une même formule en fonction des évaluations (c.f. tableau 1.4 et tableau 1.7). Ces écarts s'expliquent :

- Par les évolutions de WordNet. Par exemple, l'introduction de nouveaux mots (l'extension du test de Miller et Charles de 28 à 30, grâce à l'introduction dans WordNet de *woodland* en 2000), la réorganisation du réseau sémantique dans la version 1.7, des différences dans les corpus d'apprentissage pour les mesures nécessitant un precalcul ([Budanitsky et Hirst, 2006, Jarmasz et Szpakowicz, 2003]).
- Par le choix de la fonction IC. La méthode utilisée pour calculer la quantité d'information d'un concept fait automatiquement varier les résultats de toutes les mesures sémantiques l'utilisant.

Néanmoins, malgré ces écarts, les ordres de grandeurs et de performance restent les mêmes. Les résultats du tableau 1.7 permettent donc de comparer objectivement la qualité des mesures entre elles.

L'étude du tableau 1.5 nous permet de conclure que la structure du modèle des connaissances utilisée modifie de manière significative les résultats. Par exemple, l'uniformité entre niveaux permet de réduire l'écart type des moyennes des distances à partir d'un concept donné. Autrement dit, si l'on prend un concept c au hasard et une certaine valeur de distance v, alors les concepts C tels que $\forall c' \in C, dist(c, c') = v$ sont repartis de manière plus uniforme (*i.e.* ont un nombre d'arêtes très proche). Cette uniformité baisse les résultats des distances comme Jiang et Conrath car elles sont justement construites pour être robustes à la non-uniformité.

De manière logique, les mesures sémantiques de similarité tendent à échouer au test WordSimilarity-353 (*i.e.* avoir une corrélation très faible, c.f. tableau 1.7). Néanmoins, les mauvais résultats généraux sur WS-353 s'expliquent aussi par le test en lui-même et son application à WordNet. En effet, tel que déjà énoncé dans [Strube et Ponzetto, 2006], le test WS-353 contient beaucoup de couples connectés entre eux par des liens de « sens commun » qui ne peuvent pas exister dans WordNet (*e.g.* « popcorn - movie », etc.). Pour cette raison, nous pensons qu'il sera très difficile d'aller au delà de la limite $0.35 - 0.4$ sur ce test en utilisant WordNet comme base de connaissance.

Finalement, les mesures de similarité les plus efficaces selon les évaluations sont les mesures de Lin et de Jiang & Conrath. Il est intéressant de noter que la conception de ces deux mesures est radicalement différente. D'un coté, la mesure de Lin considère la quantité d'information partagée entre les concepts. Ainsi, si deux concepts ont comme *ccp* la racine de la hiérarchie, alors la similarité de Lin entre ces deux concepts est nulle. A l'inverse, Jiang & Conrath considère la longueur du chemin entre deux concepts. Ainsi, le seul cas de similarité nulle est alors le cas limite de deux feuilles de la hiérarchie ayant la racine comme *ccp*.

1.2.5 Les distances entre ensembles de concepts

Une mesure sémantique définit une distance entre deux concepts, mais ne permet pas de calculer une distance entre deux ensembles de concepts. Pourtant, obtenir une distance entre ensembles de concepts à partir d'une distance concept-à-concept peut être intéressant, notamment pour calculer la distance entre deux ontologies afin d'optimiser un alignement futur

qui serait plus coûteux [Euzenat, 2008], ou encore dans le cadre des systèmes d'interaction homme-machine, pour apparier une commande de l'utilisateur avec une commande formelle du système [Eliasson, 2007]. L'objectif étant alors de trouver quelle est la commande formelle du système la plus proche de celle de l'utilisateur. C'est cette dernière problématique qui nous intéressera plus particulièrement.

Nous donnons dans cette section les quelques mesures connues pour calculer la distance entre deux ensembles de concepts à partir d'une mesure de distance concept-à-concept. La mesure que nous nous proposons d'utiliser sera définie dans le chapitre 3.

1.2.5.1 La similarité de Jaccard

La similarité de Jaccard est l'une des similarités les plus simples entre deux ensembles d'éléments directement comparables. Elle consiste à calculer le rapport entre l'intersection de deux ensembles par rapport à leur union. Autrement dit, soit A et B deux ensembles, alors la similarité de Jaccard est définie par :

$$sim_{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

L'avantage de cette formule est qu'elle est très simple à utiliser. Elle ne permet néanmoins pas d'utiliser directement une distance continue entre deux concepts (*i.e.* l'intersection s'interprète en pratique comme une distance binaire $\{0, 1\}$ d'égalité entre concepts). En considérant une distance sémantique de co-occurrences, cette mesure est le principe de base de toutes les mesures basées sur les moteurs de recherche internet (section 1.2.2.5) [Strube et Ponzetto, 2006, Gabrilovich et Markovitch, 2007].

1.2.5.2 La distance de Minkowski

La distance de Minkowski est une généralisation paramétrée de la distance de Manhattan, de la distance euclidienne ainsi que de la distance de Chebychev. Plus qu'une distance entre ensembles, elle est ainsi une distance entre vecteurs de valeurs et suppose ainsi que les ensembles sont de même taille et que chaque élément a un seul élément de comparaison possible. Elle est définie selon un paramètre p permettant d'influer sur l'importance d'une comparaison donnée sur le score global. Ceci permet de choisir si les valeurs particulièrement différentes de la moyenne ont une influence forte ou faible sur le résultat final (plus p est petit, plus les valeurs particulières sont importantes et inversement). La formule est alors définie par :

$$dist_{Minkowski}(A, B) = \left(\sum_{i=1}^{K} W_i \times |a_i - b_i|^p \right)^{1/p}$$

tel que K est la cardinalité des vecteurs et W_i une pondération permettant d'influer sur l'importance de la dimension i. En envisageant A et B comme des ensembles de concepts et non plus comme des vecteurs, il est facile de remplacer la différence $|a_i - b_i|$ par un calcul de similarité sémantique entre les concepts a_i et b_i. Néanmoins, le fait que les vecteurs doivent avoir la même cardinalité et soient ordonnés (*i.e.* une seule permutation d'appariement des deux ensembles) est une limitation sérieuse, car cette hypothèse n'est pas valide en pratique

sur les ensembles de concepts.

A noter qu'en prenant $p = 1$ et $\sum_{i=1}^{K} W_i \leq 1$, on obtient alors une moyenne pondérée entre les termes et bornée dans l'intervalle $[0, \sum_{i=1}^{K} W_i]$. A l'inverse, si $p \to \infty$, alors $dist_{Minkowski}(A, B) \to \max_{i \in [1,K]} |a_i - b_i|$, ce qui fais évoluer dans ce cas la distance vers la recherche de l'écart le plus grand (et non proche de la moyenne).

En prenant le temps nécessaire pour modéliser les vecteurs, il est possible d'utiliser cette distance pour aligner des ontologies. Par exemple, cette mesure a été utilisée dans le milieu médical, pour de l'appariement d'ontologies médicales en vue d'une meilleure interprétation d'un diagnostic [Bousquet *et al.*, 2000].

1.2.5.3 Similarité trigonométrique

La similarité à base de cosinus consiste à calculer le cosinus de l'angle entre deux vecteurs de valeurs. Plus ce cosinus est faible, plus les vecteurs sont similaires. Inversement, plus ce cosinus est grand (1 pour l'orthogonalité), moins les vecteurs sont similaires. Cette propriété est souvent utilisée dans les algorithmes de recherche de textes, à partir de vecteurs représentant le contenu d'un texte. Cette similarité est alors définie comme :

$$sim_{cos}(A, B) = \frac{A.B}{\|A\| \, . \, \|B\|}$$

Là encore se pose le problème de l'adaptation d'un vecteur de valeurs à un ensemble de concepts. D'autant plus qu'il n'est pas évident ici d'intégrer le calcul de distance sémantique entre deux concepts donnés. Ce type de mesure est ainsi plus couramment utilisé pour du calcul sémantique basé sur la co-occurrence de termes en fixant les dimensions sur les différents termes possibles [Patwardhan et Pedersen, 2006, Banerjee et Pedersen, 2003].

1.2.5.4 La distance de « lien moyen »

La distance lien moyen consiste simplement à calculer la moyenne des scores de distance concept-à-concept en considérant chaque permutation des deux ensembles de concepts A et B. Plus formellement, la distance entre deux ensembles de concepts est définie par :

$$dist_{moyen}(V_1, V_2) = \begin{cases} 0 & \text{Si } V_1 = V_2 \\ \frac{1}{|V_1||V_2|} \sum_{u \in V_1} \sum_{v \in V_2} dist(u, v) & \text{Sinon} \end{cases}$$

C'est la première mesure que nous présentons qui est directement applicable pour calculer la distance sémantique entre deux ensembles de concepts à partir d'une mesure concept-à-concept. A noter que cette distance fut d'ailleurs aussi proposée par [Rada *et al.*, 1989] (section 1.2.3.2.1) comme extension ensembliste à sa distance concept-à-concept.

Le problème important de cette mesure est du au fait de l'utilisation de *toutes* les permutations de couples de concepts. Une des conséquences directes et qu'il est ainsi nécessaire d'imposer la propriété de minimalité $dist_{moyen}(A, A)$, ce qui n'est pas naturelle.

1.2.5.5 La distance de Hausdorff

Définie initialement pour le calcul de distance pour la reconnaissance de formes, la distance de Hausdorff s'applique facilement à deux ensembles de concepts. La distance est alors définie comme la distance maximale d'un point d'un sous-ensemble au point le plus proche de l'autre sous-ensemble. Plus formellement, elle est définie comme :

$$dist_{Hausdorff}(A,B) = \max\left\{\max_{a\in A}\min_{b\in B} dist(a,b), \max_{b\in B}\min_{a\in A} dist(a,b)\right\}$$

Le problème de cette formule est alors l'inverse de celui de la distance de lien moyen, sa valeur est celle d'un seul couple de distance et ne tient pas compte de l'agrégation de plusieurs valeurs pour affiner le résultat.

1.2.5.6 La distance de couplage maximal de poids minimal

L'objectif pour éviter les problèmes rencontrés dans les mesures précédentes est d'utiliser un couplage maximal pour calculer la distance, afin d'optimiser l'utilisation des permutations. A ce titre, [Valtchev, 1999] propose une mesure tenant compte du couplage maximal. Un couplage maximal de poids minimal entre deux ensembles de concepts A et B est un couplage maximal $M \subseteq A \times B$, tel que pour tout autre couplage $M' \subseteq A \times B$,

$$\sum_{\langle p,q\rangle\in M} dist(p,q) \leq \sum_{\langle p,q\rangle\in M'} dist(p,q)$$

La distance entre les ensembles de concepts est alors définie par :

$$dist_{Valtchev}(A,B) = \frac{\sum_{\langle p,q\rangle\in M} dist(p,q) + \max(|A|,|B|) - |M|}{\max(|A|,|B|)}$$

L'avantage de cette distance est quelle tient compte des permutations possibles entre les différents éléments des ensembles A et B. Néanmoins, l'évaluation a montré que cette distance était très lente, due justement à ce calcul de permutation (à l'inverse des mesures tel que celle de Hausdorff décrite précédemment) [Euzenat, 2008]. Il sera donc nécessaire en pratique de trouver un juste milieu en fonction de l'ordre de grandeur des ensembles A et B et du degré de précision attendue.

A ce titre, nous présenterons section 2.3.2 notre proposition de mesure entre nuages de concepts. Notre proposition repose aussi sur une recherche du couplage maximal, mais l'ensemble des permutations possibles est réduit par l'utilisation d'une structure d'arbre de dépendance entre les concepts des deux nuages. Le principe de l'arbre de dépendance est d'assigner des dépendances hiérarchiques entre les concepts des deux nuages. Lorsqu'une permutation est proposée, alors les propriétés de subsomption entre couples de concepts doivent être respectées.

1.2.6 Conclusion

Cette section a permis de faire un état de l'art sur les mesures sémantiques (de distance/similarité ou de degré de relation) existantes. Beaucoup de travaux ont étudié la similarité sur WordNet. Finalement, ces travaux s'appliquent mal aux systèmes d'interactions homme-machine ou agent-agent, qui utilisent des ontologies spécifiques de l'application mise en jeux. Les protocoles de tests restent à définir et il n'y a pas de mesure efficace « universellement » actuellement. En pratique même, plusieurs études ont montré qu'il est difficile de choisir une bonne distance sémantique :

- A cause de la différence de jugement humain. Il a été démontré que chaque utilisateur possède une fonction de jugement sémantique différente, qui rend ainsi un choix unique de mesure pour un système donné difficile et complexe [Bernstein *et al.*, 2005].
- A cause des différents types d'application. Une même personne jugera différemment la similarité sémantique en fonction de l'application utilisée [Gandon *et al.*, 2008], ainsi qu'en fonction du contexte initial à l'application. Par exemple, la lecture d'un texte influence chez un utilisateur un certain type de contexte pour les mots ambigus et modifie ainsi les résultats des évaluations sémantiques postérieures [Klebanov et Shamir, 2006].
- A cause de la structure de l'ontologie. L'homogénéité en niveau de l'ontologie, sa densité moyenne et son nombre de concepts influent automatiquement sur la distance moyenne des plus courts chemins et la distribution des pondérations attachées aux concepts [Blanchard, 2008].

Établir une mesure sémantique « universelle » semble ainsi totalement impossible. Néanmoins, les travaux sur les mesures sémantiques ont permis d'étendre le domaine des connaissances autour du sujet et d'apporter des outils permettant de construire les mesures adaptées. Les mesures de similarités basées sur les hiérarchies ont ainsi permis de développer les notions de quantités d'informations, de densité et d'importance de la profondeur qui améliorent de manière générale les résultats, même si ils peuvent être optimisés dans le cadre d'applications particulières. Néanmoins, peu de travaux proposent une prise en charge des relations hétérogènes, alors qu'elles offrent un plus non négligeable dans beaucoup d'applications (tel que la désambiguïsation de texte ou la commande en langue naturelle) [Corby *et al.*, 2004]. Lorsque ces mesures existent, les évaluations sont incomplètes, voir limitées à des cas très particuliers et difficilement adaptables en dehors du cadre de définition. De plus, ces mesures ont été définies sur WordNet et n'utilisent donc que la relation de méronymie *part-of*, ce qui rend encore plus difficile leur évolution correcte vers un modèle plus complet.

D'autre part, le calcul de distance entre deux ensembles de concepts est un problème ouvert et difficile à maîtriser, rendant nécessaire une conception préalable du rapport en précision attendue et cardinalité des ensembles à considérer. De plus, bien que cela n'ait pas été évoqué dans cette section, le choix d'une mesure concept-à-concept peut aussi avoir un impact sur le choix et les résultats d'une mesure ensemble-à-ensemble, rendant d'autant plus difficile le bon paramétrage en fonction de l'application souhaitée.

La section suivante présente les systèmes d'interaction homme-machine existants. Dans la mesure du possible, lorsque l'un des systèmes utilise une mesure sémantique, aussi primitive qu'elle soit, nous ferons le lien avec les mesures de la section précédente.

1.3 Systèmes de dialogue homme-machine

Cette section présente un aperçu des différents systèmes d'interaction homme-machine existants. Nous insisterons particulièrement sur la capacité de ces systèmes à être réutiliser (modèle de connaissance séparé, gestion de l'hétérogénéité sémantique indépendante de l'application, etc.).

1.3.1 Systèmes historiques

1.3.1.1 Eliza & Alice

Les travaux de Joseph Weizenbaum sont les premiers du genre dans l'interaction homme-machine en langage naturel [Weizenbaum, 1966]. Eliza est un programme sensé prendre le rôle d'un « psychiatre ». L'approche utilise des méthodes de pattern-matching et de recherche de mot-clefs. La réponse se fait en fonction des mots-clefs par « des phrases à trous » (*template-based approach*) à compléter.

Par exemple, si l'utilisateur tape la phrase « I am very unhappy these days » alors Eliza active la règle de traitement `(I AM 0)`, signifiant que la phrase est constituée des mots « I » suivi « AM » suivi d'un certain nombre de mots. Le système utilise alors le template « How long have been X » et répond « How long have you been unhappy these days ». Il est intéressant de noter qu'Eliza ne fait pas d'analyse sémantique. Ainsi, à l'affirmation « I am NLIHJY », Eliza répondra « How long have you been NLIHJY ? ».

Un autre exemple illustre les règles de transformations d'Eliza. Si l'utilisateur tape la phrase « It seems that you hate me », il active la règle (`0 YOU 0 ME`). Il utilise alors les règles de réassemblages « YOU » vers « I » et « ME » vers « YOU » ainsi que le template « WHAT MAKES YOU THINK I X YOU » et répond donc « What makes you think I hate you ».

Eliza est ainsi composé de mot-clefs auxquels sont associés des règles de décomposition, puis sont associés des règles de réassemblages et des templates. Ainsi, un script Eliza est une suite de clauses du type :

```
(K (D 1)(R 1,1) (R 1,2) ... (R 1,m1))
   (D 2)(R 1,1) (R 1,2) ... (R 1,m1))
    ...
   (D 2)(R 1,1) (R 1,2) ... (R 1,m1))
)
```

Où K est le mot-clef à considérer, D_i la i-ème règle de décomposition de K et $R_{i,j}$ la j-ème règle de réassemblages ou templates.

De plus, Eliza garde une pile des sujets évoqués de manière à pouvoir relancer sur des anciens mots entrés si la phrase de l'utilisateur ne permet pas un usage pertinent de l'ensemble de règles.

Exemple de dialogue :

> How do you do. Please state your problem.
> >> I spend too much time with computers.
> Why do you mention computers ?
> >> You are one.

Does it please you to believe I am one ?
>> Yes.
You seem to be quite positive.
>> I suppose I am.
How long have you been ?
>> 31 years.
I'm not sure I understand you fully.
>> Join the club.
Please go on.
>> You don't make too much sense yourself.
We were discussing you – not me.

À l'époque, Eliza permit de montrer qu'il était possible d'avancer sur la voix du dialogue en langage naturel. Néanmoins, l'approche mots-clefs, ne faisant aucune analyse sémantique, implique régulièrement un dialogue décalé par rapport aux intentions de l'utilisateur. Pour illustrer ce problème, reprenons l'exemple précédent sur la phrase de l'utilisateur « I suppose I am ». Eliza reconnaît la règle `(I AM 0)`, mais étant donné qu'il n'y a rien après « I am », Eliza répond « How long have you been ? », réponse hors-contexte. De plus, les connaissances font parties intégrante du système, ce qui le rend difficile à adapter à d'autres domaines.

Plus récemment, Richard Wallace a défini ALICE[13], un système plus moderne basé sur le système de pattern de Eliza. Le langage pour définir les patterns est nommé AIML (Artificial Intelligence Markup Language) et est basé sur XML. Un exemple de règles AIML :

```
<category>
  <pattern>*</pattern>
  <that>WHAT CAN I CALL YOU</that>
  <template>
    <think><set_personality>average</set_personality><setname/></think>
    Nice to meet you <getname/>.
    <srai>GET NAME GENDER</srai>
  </template>
</category>

<category>
  <pattern>GET NAME GENDER</pattern>
  <template>
    <condition name ="name">
      <li value="JEB"><think><set_gender>male</set_gender></think></li>
      <li value="ANDRE"><think><set_gender>male</set_gender></think></li>
      <li value="ALLI"><think><set_gender>female</set_gender></think></li>
      <li value="ABBY"><think><set_gender>female</set_gender></think></li>
      <li><br>Are you a man or a woman ?</li>
    </condition>
  </template>
</category>
```

[13] Vous pouvez communiquer avec Alice : `http://alice.pandorabots.com`

Le langage permet de définir des croisements entre règles dans le cas de commandes ne correspondant pas directement aux patterns, ce qui est une amélioration par rapport à Eliza. Néanmoins les systèmes restent très proches.

1.3.1.2 Shrdlu (Terry Winograd)

Terry Winograd introduit dans [Winograd, 1972] le monde de cube BLOCKS dirigé par un programme (Shrdlu) capable de dialoguer en anglais avec un utilisateur. L'auteur décrit lui-même son système par :

> The system answers questions, executes commands, and accepts information in an interactive English dialog... The system contains a parser, a recognition grammar of English, programs for semantic analysis, and a general problem solving system... It can remember and discuss its plans and actions as well as carrying them out... Knowledge in the system is represented in the form of procedures, rather than tables of rules or lists of patterns.

Shrdlu est basé sur trois idées principales :

1. Le monde réduit : l'utilisation d'une modélisation limitée aux formes géométriques, couleurs, positionnement, etc. limite le nombre de concepts différents à environ 50.
2. La mémoire : Shrdlu garde en mémoire l'ensemble du dialogue, afin de :
   (a) Faire de la résolution de référence. Exemple : « put the green cone on the red block » puis « take the cone off ».
   (b) Poser des questions d'historique. Exemple : « did you pick up anything before the cone ? ».
3. Une modélisation du monde à base de règles à respecter (*i.e.* contraintes sur les actions) : Ce qui permet de poser des questions sur les actions possibles ou non, ou de savoir pourquoi une action demandée n'est pas réalisable.

Exemple de dialogue avec Shrdlu :

> Person : PICK UP A BIG RED BLOCK.
> Computer : OK.
> Person : GRASP THE PYRAMID.
> Computer : I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
> Person (changing their mind) : FIND A BLOCK WHICH IS TALLER THAN THE ONE YOU ARE HOLDING AND PUT IT INTO THE BOX.
> Computer : BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.
> Computer : OK.
> Person : WHAT DOES THE BOX CONTAIN ?
> Computer : THE BLUE PYRAMID AND THE BLUE BLOCK.
> Person : WHAT IS THE PYRAMID SUPPORTED BY ?
> Computer : THE BOX.
> Person : HOW MANY BLOCKS ARE NOT IN THE BOX ?
> Computer : FOUR OF THEM.

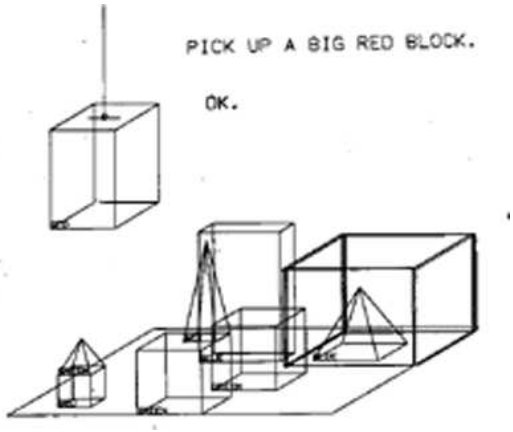

FIG. 1.6 – L'interface de Shrdlu

Person : IS AT LEAST ONE OF THEM NARROWER THAN THE ONE WHICH I TOLD YOU TO PICK UP ?

Computer : YES, THE RED CUBE.

etc.

Shrdlu est le premier à chercher à décomposer le dialogue en différentes étapes (modélisation syntaxique suivant des règles de grammaire anglaise, analyse sémantique, résolveur de problème). L'approche n'est donc plus exclusivement à base de mots-clefs et permet ainsi d'identifier les actes de langages (requêtes, commandes, confirmation, etc. même si Winograd ne parle pas d'actes de langages).

Le principal inconvénient est que Shrdlu est complètement lié au domaine du monde BLOCKS. En effet, les connaissances du monde sont codées directement dans les règles. Il est ainsi impossible de l'étendre à un autre domaine ou à d'autres contraintes. De plus, le système n'a pas de représentation des actions qu'il effectue (il se contente d'exécuter les règles).

1.3.2 Systèmes à base de plans et règles

L'évolution suivant les systèmes précédents de dialogue homme-machine moderne intégrait des plans de dialogue. Plus précisément, ils utilisaient la théorie des plans partagés [Grosz et Sidner, 1986]. Sur cette base, nous donnerons les systèmes existant de dialogue à base de plans, à savoir Circuit Fix-It Shop, Collagen et Trains.

1.3.2.1 La reconnaissance de plan

Tout d'abord, rappelons la définition d'un plan et surtout de la reconnaissance de plan, qui est la base des systèmes de dialogue utilisant la théorie des plans partagés. Pour pouvoir définir un plan, il faut :

1. Une représentation du monde sous forme d'ensemble d'états possibles.
2. Un ensemble d'actions transformant un état e en un état f.

Un plan est alors une liste ordonnée d'actions à effectuer, permettant d'obtenir à partir d'un état initial e un état final, ou but, f. On parlera de planification lorsque l'on cherche

à construire un plan à partir de l'état initial et de l'ensemble des actions possibles. Dans le cas des systèmes de dialogues, les plans représentent les différentes étapes d'un dialogue précis entre l'utilisateur et la machine. Chaque étape change l'état mental de l'agent (et, normalement, de l'humain). Les plans sont donc définis à l'avance, et il n'y a pas nécessité d'effectuer de la planification. Les plans donnés à l'avance à un système sont, en général, nommés des recettes [Pollack, 1990]. Par contre, à partir d'une suite d'interactions donnée, il est intéressant pour un système informatique de pouvoir calculer comment réagir à un certain nombre d'échanges avec l'utilisateur, *i.e.* de reconnaître parmi la base des recettes fournies la « recette plan » intégrant les échanges humain-machine précèdent, afin de déduire la suite logique du discours. Cette activité, la reconnaissance de plan, est au cœur de ce type de systèmes de dialogue. Elle est actuellement fortement liée pour le dialogue à la théorie du discours en plan partagée [Grosz et Sidner, 1986]. Nous ne détaillerons pas ce formalisme, car il n'est pas l'objet principal de cette thèse.

1.3.2.2 Le système Circuit Fix-it Shop

Le système Circuit Fix-it Shop [Smith *et al.*, 1992, Smith *et al.*, 1995, Smith, 1998] est un système d'aide à l'utilisateur pour la réparation d'un circuit électrique. Les auteurs se basent sur la théorie du discours partagé [Grosz et Sidner, 1986]. Le système est écrit en Prolog. Le principe du système est une sorte de démonstrateur mathématique. Le système tente de démontrer le prédicat but et, en cas de problème ou d'informations incomplètes, coopère avec l'utilisateur. D'un point de vue utilisateur, le système le suit dans sa démarche de réparation en lui demandant de vérifier les points critiques associés à une erreur.

Les auteurs proposent un système de modélisation des connaissances utilisateurs. Par contre, ils ne cherchent pas à modéliser son intention dans le dialogue, mais ses connaissances réelles sur la tâche en court. En fonction de certains énoncés de l'utilisateur ou de réponses à des questions du système, la modélisation permet de connaître les compétences du réparateur sur le circuit afin d'éviter les questions redondantes, ou sur des sujets où le réparateur n'a pas besoin d'aide. En pratique, ces informations permettent au démonstrateur d'avancer plus facilement sur les points considérés comme connus (les prédicats Prolog sont ajoutés à la base des axiomes).

Déroulons maintenant les principes précédents sur un exemple simplifié à partir de celui présenté dans [Smith *et al.*, 1995]. Supposons que l'objectif du système soit de « tester le circuit numéro 2 » et que ce but soit modélisé par le prédicat $CircuitTest2(V)$ ou V est le voltage résultat du test. Supposons de plus que le système possède les règles d'inférence suivante :

$CircuitTest2(V) \longleftarrow set(knob, 10), measureVoltage(121, 34, V)$

$set(X, Y) \longleftarrow find(X), adjust(X, Y)$

Supposons encore que le modèle de l'utilisateur par le système soit :

$find(knob)$ (*i.e.* le système croit que l'utilisateur sait où est l'interrupteur)

$usercan(adjust(knob, X))$ (*i.e.* le système croit que l'utilisateur peut modifier l'état de l'interrupteur)

$usercan(measureVoltage(X, Y, Z)$ (*i.e.* le système croit que l'utilisateur sait mesurer une tension)

La première action du système va donc être de démontrer $set(knob, 10)$. Pour cela, il déroule la règle $set(X, Y)$ avec les paramètres $X = knob$ et $Y = 10$. La démonstration du prédicat $find(knob)$ ne pose pas de problème car le système croit que l'utilisateur sait où est l'interrupteur. La démonstration du prédicat $adjust(X, Y)$ ne peut être faite pas le système, mais l'utilisateur en a la possibilité. Le système lance donc un dialogue :

Système : Put the knob to one zero

Le système se place alors en attente de réponse dans l'un des quatre cas suivants :

$question(location, knob)$
$question(ACTION, howToDo)$
$assertion(knob, status, 10)$
$assertion(ACTION, done)$

Si l'utilisateur répond « ok, it's done », l'assertion $assertion(ACTION, done)$ est envoyée au système qui considère l'action $adjust(knob, 10)$ comme démontrée. Si l'utilisateur répond « where is the knob ? », alors c'est la question $question(location, knob)$ qui est envoyée au système. Le système retire la croyance $find(knob)$ de sa base de croyance de l'utilisateur et lance l'action $find(X)$ pour aider l'utilisateur à trouver l'interrupteur. Une fois le prédicat *adjust* démontré, le prédicat *set* est démontré et le système passe à la démonstration du prédicat $measureVoltage$, etc.

Néanmoins, son système de représentation des connaissances est en fait une représentation de l'état courant des connaissances sur le circuit. Ainsi, si le coté démonstrateur mathématique s'utilise facilement dans le cadre d'un circuit électronique, l'écriture des règles d'inférences Prolog dans le cas d'autre domaines devient automatiquement plus complexe voire impossible. Certains auteurs lui reprochent d'ailleurs le temps d'adaptation nécessaire à un utilisateur pour communiquer avec le système [Allen *et al.*, 1996]. Les modules sont de plus difficilement réutilisables.

1.3.2.3 Le système COLLAGEN

Collagen [Rich *et al.*, 2001] est une bibliothèque fournissant des méthodes de gestion des dialogues collaboratifs. Le système utilise une nouvelle fois la théorie des plans partagés de [Grosz et Sidner, 1986] pour proposer à l'utilisateur la meilleure alternative à un choix donné. A l'inverse de la planification, la reconnaissance de plan cherche à découvrir le plan que suit un utilisateur donné à partir de la liste des actions qu'il a effectués ainsi que d'une liste de plans de références (*i.e.* recettes) applicables dans le contexte donné (c.f. figure 1.7). Un exemple de recette décrivant l'enregistrement d'un programme sur un magnétoscope dans le langage Collagen :

```
public recipe RecordRecipe achieves RecordProgram {
  step DisplaySchedule display ;
  step AddProgram add ;
  optional step ReportConflict report ;
  constraints {
    display precedes add ;
    add precedes report ;
    add.program == achieves.program ;
```

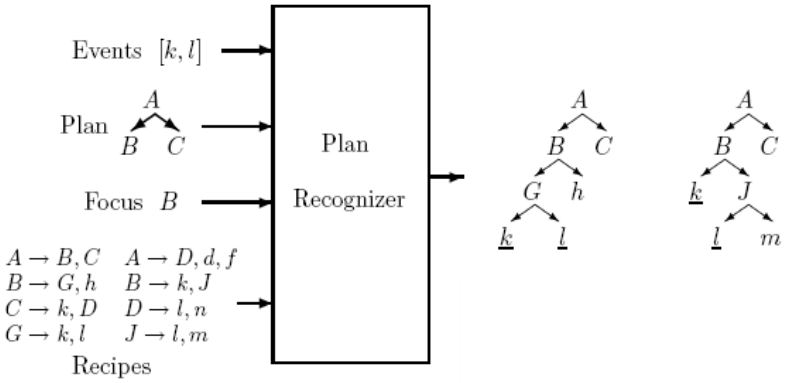

FIG. 1.7 – Un exemple de reconnaissance de plan dans Collagen. Les évènements reçus de l'utilisateur sont k et l. Le plan reconnu actuellement est l'arbre ABC avec A comme racine, le focus est actuellement en B. Selon la liste des recettes (*recipes*) actuelles, il y a deux plans possibles pour continuer l'interaction.

```
        report.program == achieves.program;
        report.conflict == add.conflict;
    }
}
```

La première ligne indique que la bonne exécution de la recette implique le bon enregistrement du programme (mot-clef « achieves »). La recette considère les actions de programmation (« Schedule ») et de validation de la programmation (« AddProgram »). Ainsi, les contraintes indiquent par exemple que la programmation doit précéder la validation.

Ce système est fortement réutilisable car les connaissances sont données par la description des plans du système grâce à la définition d'un langage de programmation spécifique. Il existe ainsi plusieurs applications différentes utilisant le moteur Collagen. Ceci permet de généraliser les algorithmes de reconnaissance de plan qui sont totalement indépendants de l'application utilisée en pratique.

Par contre, Collagen n'utilise pas de représentation séparée du domaine. Toutes les connaissances doivent être dans les plans. Ceci empêche tout traitement sémantique des commandes et toute interprétation approchée. A ce titre, les dialogues avec Collagen sont souvent pré-tapés et l'interaction est souvent très directive (*i.e.* à base d'ordre) [Rich *et al.*, 2001].

1.3.2.4 Artimis

Artimis est un système développé par France Telecom pour travailler directement en reconnaissance vocale, l'objectif final étant d'équiper des serveurs vocaux de météo, pages jaunes, etc.[14][Sadek *et al.*, 1997, Sadek, 1999]

Leur approche se base sur les agents cognitifs BDI. Un agent BDI (pour *Belief Desire Intention*) est un modèle d'agent cognitif basé une logique particulière, la logique BDI

[14] Animation de présentation du système sur le site de France Telecom : `http://www.francetelecom.com/sirius/rd/fr/galerie/dialogue_intelligent/index.php`

[Bratman, 1987]. Le modèle BDI classique a ici été étendu pour modéliser les croyances incertaines. Ainsi, un agent Artimis a des croyances (B pour *Belief*), des croyances incertaines (U pour *Uncertainty*), des désirs (C pour *Choices*) et des intentions (I pour *Intention*). Une application est ainsi modélisée par un ensemble de règles en logique (l'approche est assez proche du système Circuit Fix-It de la section 1.3.2.2, mais le modèle logique est plus complet). Le système possède lui-même un ensemble d'axiomes de base.

Formellement, Artimis utilise les quantificateurs logiques habituels, puis $B(i,p)$, $U(i,p)$, $C(i,p)$ et $I(i,p)$ pour « i croit que p est vraie », « i n'est pas sur que p soit vraie », « i veut que p soit vraie » et « i a l'intention de réaliser p ». Ainsi, l'axiome de base « Si un agent a l'intention d'être dans une situation où une action est exécutée, alors l'agent a l'intention de faire en sorte que l'action soit réalisable, sauf s'il croit déjà qu'elle est réalisable » se traduit en :

$$\models I(i, Done(a)) \Rightarrow B(i, Feasible(a)) \vee I(i, B(i, Feasible(a)))$$

Une application Artimis doit posséder un ensemble d'axiomes de ce type ainsi qu'un ensemble de règles spécifiques à l'application. Ensuite, la commande est modélisée en logique Artimis comme point de départ des inférences nécessaires pour obtenir le but. Par exemple, la phrase "I'd like ? weather forecast for ? Lannion" (? représente des mots incompris lors de la reconnaissance vocale) sera modélisé par :

$$< u, Informs(s, I(u, Bref(u, x\, numtel(x) \wedge \exists y\, server(y) \wedge number(x, y) \\ \wedge topic(y, weather\text{-}forecast) \wedge domain(y, lannion)) >$$

Autrement dit, « L'utilisateur u informe (*i.e.* $informs$) le système s qu'il veut (*i.e.* $I(u, ...)$) savoir (*i.e.* $Bref(u, x...)$) le numéro de téléphone d'un serveur Météo à Lannion ».

Comme les systèmes précédents, Artimis a l'avantage de posséder un langage spécifique de description des actions et règles de l'application, mais il n'y a pas de base de connaissances clairement séparée. Ceci a d'ailleurs amené David Sadek à formuler l'hypothèse de *connectivité sémantique* selon laquelle tous les concepts énoncés dans une commande de l'utilisateur bien construite sont forcement définis dans le modèle des connaissances de l'application. Ceci illustre le fait que *tous* les concepts d'une commande doivent être clairement définis et qu'il n'existe pas d'algorithme d'approximation sémantique.

1.3.2.5 Le système TRAINS-95

TRAINS [Allen *et al.*, 1996] est un projet dirigé par James F. Allen et Len Schubert. L'objectif du projet est de développer un système de dialogue homme-machine robuste (*robust interpretation of speech in the presence of recognition error*). Les problèmes clairement énoncés que l'équipe de TRAINS cherche à résoudre sont les suivants :

1. Fonctionner en temps réel (ou au moins temps acceptable pour un utilisateur humain en phase dialogue).
2. Pas d'entraînement pour utiliser le système, pas de contraintes de vocabulaire.
3. Un protocole d'évaluation objectif basé sur la réalisation de la tâche (évaluation globale) et non sur la compréhension d'une phrase en particulier (évaluation locale).

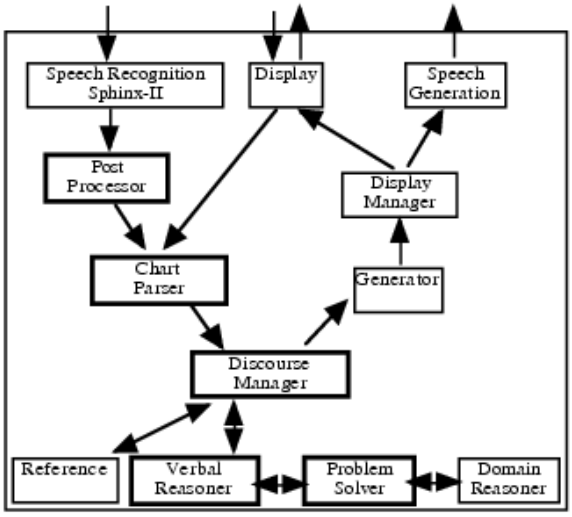

FIG. 1.8 – L'architecture du système TRAINS-95

L'utilisateur interagit avec le système vocalement ou par le clavier. Le feed-back de l'utilisateur est basé sur des phrases préenregistrées (*simple template-based system*). Un parseur cherche alors à modéliser les actes de langages présents dans la phrase. Par exemple pour la phrase[15] :

Okay now I take the last train go from Albany to is

On obtient la décomposition suivante :

1. CONFIRM(ok)
2. TELL(now I take the last train)
3. REQUEST(go from Albany)

La fin de la phrase (*to is*) est ignorée par le parseur car il la considère comme inexploitable en termes d'acte.

Par la suite, un ensemble de règles basées sur la construction des actes permet de déduire la commande de l'utilisateur pour le système.

Ce système est un des premiers à proposer un vrai dialogue à l'utilisateur, impliquant que la commande voulue peut-être discutée sur un échange de plusieurs phrases (*une* commande n'est pas *une* phrase). Ainsi, le protocole d'évaluation a évolué vers la réalisation de la tâche et non plus du nombre de phrases bien comprises. En effet, le système peut demander confirmation d'une commande partiellement comprise.

Un des principaux défauts de TRAINS, avoué par Allen lui-même, est le fait qu'il ne soit pas réutilisable. En effet, les connaissances liées à l'interaction de TRAINS sont profondément liées au système, qui ne peut pas évoluer directement vers d'autres domaines. C'est une des raisons fondatrices du projet suivant de James F. Allen, TRIPS (que nous évoquerons section 1.3.3.3).

[15] Phrase issue du module de reconnaissance vocale et comportant donc quelques erreurs de transcription.

1.3.3 Systèmes à base d'ontologies

Les systèmes plus récents utilisent maintenant une ontologie pour augmenter la performance des modules d'interprétation sémantique. Cette ontologie décrit le domaine du système. Elle est utilisée différemment en fonction du système, avec une prise en compte du fait de pouvoir réutiliser le système très variable (nous verrons ainsi certains systèmes qui définissent un formalisme d'ontologie tellement spécifique, que le système ne devient pas plus réutilisable qu'un système classique).

1.3.3.1 Paraiso & Barthes

Le système de Paraiso et Barthes ([Paraiso *et al.*, 2004, Paraiso et Barthès, 2004]) est une architecture pour agents conversationnels. Il se positionne par rapport au système Trains (il détecte aussi des actes de langages, *inform, question* et *answer*), mais il utilise pour l'interprétation sémantique WordNet et une ontologie de tâche.

WordNet est utilisé pour enrichir les modèles sémantiques avec les synonymes uniquement (pas d'utilisation de distance sémantique telle que celle présentée la section 1.2.3). L'ontologie de tâche est définie suivant un formalisme très spécifique. Elle représente une *liste de compétences*, correspondant à la liste de ce que l'agent peut faire. Le schéma d'une compétence est le suivant :

```
Compétence : et : <liste de mots clés>
             ou : <liste de mots clés>
Paramètres obligatoires : pi : <liste de mots clés>
                          qi : <question pour pi>
                          ti : <type de champ>
                          si : statut <T/F>
Paramètres optionnels : pi : <liste de mots clés>
                        qi : <question pour pi>
                        ti : <type de champ>
                        si : statut <T/F>
Action : <script> Autorisation = Yes/No
Statut de l'action : <executed, pending, running, read, suspended>
```

En résumé, bien qu'utilisant une ontologie pour séparer les connaissances du code l'agent, l'approche est précâblée. La liste des compétences est écrite à la main et apparie la liste des mots-clefs pour en déduire l'action (*i.e.* script) à déclencher. Il s'agit ainsi d'un formalisme pour décrire les appariements entre un ensemble de commandes de structure proches (mais différentes) à une entrée spécifique d'actions.

1.3.3.2 Le système SNePS

Le système SNePS[16] (pour *Semantic Network Processing System*) est un projet ancien dédié initialement à l'étude de la structure des réseaux sémantiques pour le raisonnement [Shapiro, 2000, Shapiro et Kandefer, 2005, Shapiro, 1996, Rapaport, 2000].

[16] http://www.cse.buffalo.edu/sneps/

```
(V (CAT V T ; The next word must be a verb.
       (SETR VB (FINDORBUILD LEX (^(GETR *)))) (SETR TNS (GETF TENSE))
       (TO COMPL)))
(COMPL ; Consider the word after the verb.
   (CAT V (AND (GETF PPRT) (OVERLAP (GETR VB) (GETA LEX- 'BE)))
      ; It must be a passive sentence.
       (SETR OBJ (GETR SUBJ)) (SETR SUBJ NIL) (SETR VC 'PASS)
       (SETR VB (FINDORBUILD LEX (^(GETR *)))) (TO SV))
   (CAT ADJ (OVERLAP (GETR VB) (GETA LEX- 'BE))
      ; A predicate adjective.
       (SETR ADJ (FINDORBUILD LEX (^(GETR *)))) (TO SVC))
   (JUMP SV T))
```

FIG. 1.9 – Un exemple de règle pour construire un arbre ATN

Les connaissances sont modélisées sous la forme d'un réseau sémantique propositionnel (*propositional semantic network*) dont la différence avec les réseaux sémantiques se situe dans le fait qu'un concept est toujours un nœud. Les connaissances du monde sont incorporées dans celle de l'agent (le monde n'a pas d'existence autre que celle de l'agent).

Son système de raisonnement se base sur ATMS [De Kleer, 1984] et s'appelle SWM* (parfois noté SNePS-LOG [Shapiro, 2000]). SNePS-LOG est une extension d'une logique de niveau supérieur acceptant les variables sur les noms de prédicats. SNePS redéfinit un système complet d'opérateur n-aire (*and-or*, *thresh*, ...) permettant d'alléger l'écriture de contraintes sur un ensemble de propositions (par exemple, "exactement un" parmi un ensemble à trois éléments est long à écrire avec des opérateurs binaires $< et, ou >$ classiques, donc inefficace au niveau du traitement).

Son module de modélisation des commandes est basé sur les arbres ATN [Shapiro, 1982] le principe étant de construire un arbre SNePS à partir de règles de grammaires strictes. L'avantage principal de cette approche est que les règles permettent de comprendre *et* de générer l'anglais :

1. Anglais vers SNePS : Règles de grammaire expliquant la structure de l'arbre (parseur classique)
2. SNePS vers Anglais : Calcul à partir de la structure de l'arbre des règles de grammaires qui aurait pu le produire et génération en remplaçant les catégories de la règle (verbe, sujet, ...) par le contenu des nœuds SNePS.

Son module utilise aussi un lémmatiseur capable de passer d'un mot à sa forme lémmatisée avec son étiquette mais aussi capable à partir d'un couple {étiquette, forme lémmatisée} de donner la forme conjuguée. Ce qui termine la génération. Le principal inconvénient des arbres ATN pour la modélisation d'une commande est la difficulté d'écriture des règles dans un formalisme très complexe qui rend les règles spécifiques et difficilement réutilisables (c.f. figure 1.9).

SNePS possède les avantages et les inconvénients du projet Artimis : la logique fournit un cadre de travail très strict et très efficace pour l'interprétation et les inférences, mais le modèle de représentation des connaissances est clairement intégré dans le système et le rend

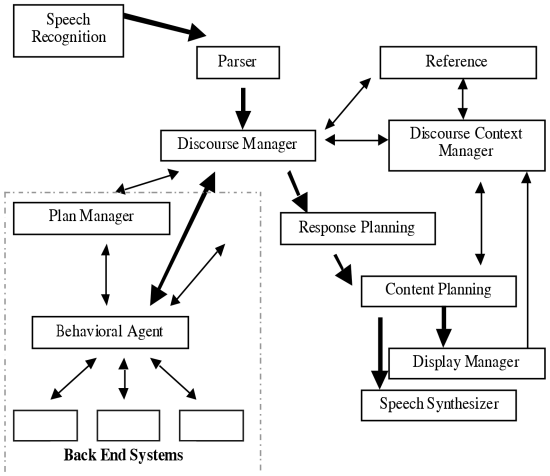

FIG. 1.10 – "Generic Dialog Shell" de TRIPS

```
(define-type LF_Vehicle
 :parent LF_Manufactured_object
 :sem (phys-obj (form objet)(origin artifact)
                (objet-function vehicle)
                (mobility movable)))
```

FIG. 1.11 – Un exemple d'ontologie LF : le concept *vehicle*

difficilement évoluable vers d'autres applications.

1.3.3.3 Le système TRIPS

Le projet TRAINS a finalement abouti au projet TRIPS [Ferguson et Allen, 1998]. TRIPS change de domaine (gestion de ressources de déplacement de type hélicoptère, camion, etc. dans le cas d'une planification d'évacuation d'une île menacée par un ouragan). Le moteur de dialogue reste donc équivalent (et reste TRAINS), mais le résolveur de problème deviens plus complexe. TRIPS ajoute aussi l'acte de langage "What-If" permettant d'émettre une planification conditionnelle.

Cependant, TRIPS cherche à réparer le principal défaut de TRAINS et se veut « le plus générique possible » [Allen *et al.*, 2000]. Ainsi, il existe plusieurs systèmes de domaines différents utilisant le même moteur (TRIPS-PACIFIA [Ferguson et Allen, 1998], TRIPS-CpoF, TRIPS-Monroe, TRIPS-AMC, TRIPS-Kitchen).

La figure 1.10 montre l'architecture du moteur générique de TRIPS. Les composants hors de la zone en pointillée sont complètement *indépendants* de l'application. La gestion du feedback est toujours *template-based* et à réécrire avec le *Behavioral Agent*. La communication est normalisée selon KQML, même entre les modules génériques. L'avantage de KQML est d'intégrer dans sa structure les actes de langages, qui font qu'un message KQML est directement un acte de langage. Certains modules (*Speech Recognizer* et *Parser)* peuvent être optimisés pour le domaine de l'application en cours.

Plus récemment dans TRIPS, Dzikovska [Dzikovska *et al.*, 2003] introduit la notion d'on-

```
(define-lf-to-kr-transform transport-transform-trips
  :preconditions (( :obligatory theme))
  :pattern ((?spec ?ev LF_Motion :agent ?a :theme ?t :intrument ?vh)
            -> ((TYPE ?ev LOAD)
                (actor ?ev ?a) (cargo ?ev ?t) (vehicle ?ev ?vh))))
```

FIG. 1.12 – Exemple d'ontologie KR pour l'action « transporter ». Ce pattern ne fonctionne que si une place est disponible sur un cargo pour le chargement.

```
(SPEECHACT sa1 SA_REQUEST :content e123)
(F e123 LF_Filling*load :agent pro1 :theme v1 :goal v2)
(IMPRO pro1 LF_Person :context-rel +YOU+)
(THE v1 LF_FOOD*orange)
(THE v2 LF_VEHICLE*truck)
```

FIG. 1.13 – Un exemple de modèle de la phrase pour la commande « load the oranges into the truck »

tologie pour augmenter encore la généricité du système. Elle définit deux ontologies, l'ontologie LF et l'ontologie KR (un exemple de morceau d'ontologie LF est donnée dans la figure 1.11 et un exemple d'ontologie KR dans la figure 1.12). L'ontologie LF est une ontologie syntaxique qui décrit les liens entre différents concepts alors que l'ontologie KR est une ontologie de lien avec les termes de l'application. L'ontologie LF est construire pour être « aussi générale que possible, une structure hiérarchique relativement plate et induite par les besoins linguistiques » à partir d'une version étendue de l'ontologie FrameNet [Johnson et Fillmore, 2000]. L'ontologie KR est « spécifique du domaine, organisée par rôles pour être efficace en inférence ». En pratique, l'ontologie LF définit énormément d'informations en double de l'ontologie KR, mais sous forme différente. De plus, les langages d'ontologies sont extrêmement spécifiques (dans l'exemple de la figure 1.11, chaque objet de l'application doit définir s'il peut bouger ou non, sa « mobilité »). De plus l'ontologie est un ensemble de pattern d'interaction branché directement sur les bonnes actions, ce qui ne constitue pas une interprétation sémantique au vu du nombre de règles nécessaires pour brancher une action (figure 1.12). Le modèle de la commande de TRIPS est ainsi très complet, mais très dépendant du système de règle de l'ontologie LF (figure 1.13).

TRIPS introduit une architecture générique très reprise actuellement. Par contre, les formalismes d'ontologie fournis sont très ad-hoc et très difficile à manipuler. La règle de la figure 1.12 montre un exemple simple qui pourtant est très complexe à écrire. De plus, le modèle n'utilise aucunes règles d'approximation sémantique à part les branchements « à la main » définis dans l'ontologie LF.

1.3.4 Systèmes à base d'ontologies et de mesure sémantique

Les systèmes les plus récents utilisent une ontologie, mais aussi des algorithmes de distances sémantiques. En effet, afin d'augmenter la généricité des systèmes, il est important d'accepter le fait que la définition des actions ne peut prendre en compte toutes les formes d'interaction et tout le vocabulaire pour déclencher cette action. Ainsi, les systèmes récents

font reposer leur interprétation sémantique sur les ontologies et surtout une mesure dans cette ontologie. Nous allons étudier ces utilisations dans ces derniers, en constatant qu'elles sont en pratique assez simples et loin de meilleures mesures telles que celles proposées dans la section 1.2.3.

1.3.4.1 SmartKom

Le système SmartKom[17] [Wahlster, 2003] a pour objectif de définir des agents conversationnels multi-modaux avec reconnaissance vocale. Ils utilisent pour cela une ontologie séparée [Gurevych *et al.*, 2003] pour la désambiguïsation de concepts et ont défini une mesure de « score contextuel » appelé OntoScore [Porzel *et al.*, 2003]. Le modèle d'ontologie est un modèle de taxonomie augmentée avec 750 concepts pour 200 relations différentes. Ainsi, l'ontologie est vue comme un graphe. SmartKom définit donc une mesure de distance sémantique sur un graphe avec différents types de relations. Soit deux concepts c_1 et c_2. Soit $\{r_1, ..., r_n\}$ l'ensemble des relations de l'ontologie et $\{w_1, ..., w_n\}$ des poids associés (0 pour la relation *is-a* et 1 pour les autres relations). Soit $P(c_i, c_j) = \{p_1, ..., p_m\}$ l'ensemble des m chemins possibles dans le graphe pour relier les concepts c_i et c_j (cet ensemble est calculé en utilisant l'algorithme de Dijkstra). Alors, le score d'un chemin p_k quelconque est donné par la formule :

$$scr(p_k) = \sum_{i=0}^{n} a_i w_i$$

où a_i représente le nombre de fois où la relation r_i apparaît dans le chemin p_k. Cette formule de score d'un chemin permet maintenant de déterminer la distance sémantique entre deux concepts quelconque c_i et c_j par la formule :

$$dist(c_i, c_j) = \min_{p_k \in P(c_i, c_j)} scr(p_k)$$

D'un point de vue distance sémantique, cette formule n'est pas homogène à celles que nous avons vues dans la section 1.2.3. En effet, en attribuant le poids de la hiérarchie à 0, tout chemin n'utilisant *que* la relation de hiérarchie, quel que soit le nombre de nœuds, aura un score $scr(p_k)$ à 0. Entre deux concepts éloignés, le chemin le plus court doit donc toujours posséder au moins une relation, afin que l'algorithme de Dijkstra élimine le chemin hiérarchique. *Ainsi, le chemin le plus court dans le graphe n'est que rarement le chemin qui donnera le score le plus faible par l'utilisation directe de la formule.* Le filtrage qu'ils mettent en place par l'algorithme de Dijkstra est donc primordial pour faire fonctionner cette mesure. De plus, leur utilisation de Dijkstra montre que l'ontologie ne contient que des relations positives (*i.e.* pas d'antonymie par exemple). Ces remarques impliquent que l'ontologie doit être très dense relationnellement et ne contenir que des relations positives. Il est ainsi difficile de faire évoluer cette formule dans un contexte différent.

Cette formule a pour finalité de servir à calculer un score contextuel pour un ensemble de concepts. Ce score contextuel calcule le degré de « densité » contextuelle de cet ensemble. Autrement dit, ce score permet d'étudier si les termes reconnus par le module de reconnais-

[17] `http://www.smartkom.org/`

sance vocale sont proches contextuellement. L'objectif étant de trouver, parmi les différentes propositions du module de reconnaissance vocale, la plus reliée contextuellement et de considérer que c'est la commande prononcée par l'utilisateur. Un ensemble de concepts sera considéré comme « contextuellement proche » si la distance moyenne entre chacun de ses concepts deux à deux est faible. Autrement dit, cette formule calcule la distance pour tous les couples de concepts possibles, puis normalise sur le nombre d'associations possibles. Soit, CR un ensemble de concepts, alors cette formule est définie de la façon suivante :

$$score(CR) = \frac{\sum_{\{c_i,c_j\} \in CR, c_i \neq c_j} dist(c_i, c_j)}{|CR|^2 - |CR|}$$

Le fait de calculer une distance entre tous les concepts revient à remplir une matrice carré $|CR| \times |CR|$ (qui possède donc $|CR|^2$ cases). La diagonale ($|CR|$ cases) n'ayant pas de sens sémantiquement (*i.e.* distance entre un concept et lui-même), elle n'est pas calculée et n'intervient donc pas dans la normalisation. La normalisation est donc bien $|CR|^2 - |CR|$.

Cependant, ce score contextuel n'est utilisé que pour désambiguïser les propositions du module de reconnaissance vocale et non pour l'interprétation sémantique des commandes à proprement parler. De même, l'ontologie n'est pas utilisée pour l'interprétation sémantique de la commande et son appariement avec les capacités. Pour cela, les différentes possibilités d'interprétation sémantique sont décrites à la main dans un langage spécifique appelé M3L. Ce langage a l'avantage de permettre la fusion des modalités dans la compréhension des commandes comme la réponse du système (ce langage permet de définir une interaction du type « quel est cet objet » avec un mouvement de l'utilisateur vers un objet). La conséquence est que SmartKom est très pauvre pour l'interprétation sémantique et se résume à une liste de patterns d'interaction possible, paramétrée par un modèle de la phrase obtenue par une méthode template.

1.3.4.2 HOMEY Project

Le projet HOMEY est un projet médical financé par l'institut Cancer Research UK. La partie interaction homme-machine est gérée par D.Milward [Milward et Beveridge, 2003]. Pour son système, il propose un ensemble de « bons critères » pour utiliser les taxonomies de concepts dans les systèmes d'interaction modernes. En ce sens, son système est assez ad-hoc d'un point de vue interaction en général mais propose des solutions d'interprétations sémantiques intéressantes.

Par exemple, utiliser les ontologies pour comprendre les réponses sous-spécifiées. Les réponses sous-spécifiées sont les réponses à des questions du système qui sont trop générales par rapport aux réponses attendues. Par exemple :

Système : Do you have a family history of chronic disease ? [réponses attendues : cancer des poumons, leucémie, sarcome, etc.] *(Avez-vous des antécédents familiaux de maladies chroniques ?)*

Patient : Yes, cancer. *(Oui, cancer)*

Système : What type of cancer ? *(Quel type de cancer ?)*

Patient : Lung cancer *(Cancer du poumon)*

Dans cet exemple, le patient répond de manière trop générale au système. Le système réduit alors le choix des réponses possibles et reformule pour avoir une réponse plus précise. La compréhension de réponses sur-spécifiées est aussi possible :

Système : Have you had any chronic disease ? [réponses attendues : cancer, hypertension, diabète, etc.] *(Avez-vous une maladie chronique ?)*

Patient : Leukomia *(Leucémie)*

Ici, la clarification n'est pas nécessaire car le système comprend grâce à l'utilisation de la hiérarchie que la réponse est une sur-spécification d'une des réponses attendues (*i.e.* la leucémie est un cancer du sang).

En étudiant le modèle de Milward, sa gestion de la sur/sous-spécificité correspond à une utilisation de la mesure de distance sémantique de Rada [Rada *et al.*, 1989] : une recherche du chemin le plus court dans le graphe amenant à un concept attendu. La recherche de Milward consiste donc à énumérer les concepts attendus et à chercher dans cette liste celui qui a la distance la plus proche du concept utilisé par l'utilisateur.

Cette application de la distance de Rada est actuellement intégrée dans les autres systèmes du projet HOMEY. Quelle que soit la mesure, l'objectif reste ainsi toujours d'énumérer les concepts attendus, puis de choisir celui qui minimise la distance sémantique au concept de l'utilisateur.

1.3.4.3 Corese

Corese est un moteur de recherche sémantique [Corby *et al.*, 2004]. Chaque document est indexé par un ensemble de concepts. L'interprétation sémantique consiste alors à associer les termes de l'utilisateur avec les concepts manipulés par l'agent artificiel en utilisant cette ontologie. Cette approche permet de concevoir des systèmes robustes et capables de comprendre des commandes ou des requêtes imprécises ou de proposer des alternatives à l'utilisateur. De plus, Corese est le seul système d'interaction à notre connaissance qui utilise explicitement l'une des mesures de distance sémantique de la littérature : celle de Zhong [Zhong *et al.*, 2002] (section 1.2.3.2.3). Soit c_1 et c_2 deux concepts de l'ontologie, alors la distance est définie par :

$$dist(c_1, c_2) = \sum_{c \in sp(c_1,c_2), c \neq c_1, c \neq c_2} \frac{1}{2^{depth(c)}}$$

où $sp(c_1, c_2)$ représente le cours chemin entre c_1 et c_2 et $depth(c)$ la profondeur du concept c dans la taxonomie (*i.e.* la distance du plus court chemin entre le concept c et la racine de la taxonomie).

Le système Corese est sans doute l'un des systèmes les plus efficaces actuellement. Néanmoins, sa distance sémantique est définie sur une hiérarchie de concepts, alors que l'un de ses concepteurs énonce le besoin de relations hétérogènes dans les ontologies [Gandon *et al.*, 2008].

1.3.4.4 Cederic

Finalement, le système Cederic [Eliasson, 2007] utilise aussi une ontologie séparée. Il utilise une mesure de similarité sémantique pour l'interprétation des commandes. Cette

```
(school has
  (instance-of (building))
  (plural (schools))
  (weight (2)))
```

FIG. 1.14 – Un exemple du langage KM pour le concept « school ». Le même modèle définit la taxonomie, la syntaxe et les poids de l'interprétation sémantique.

mesure repose sur une pondération des nœuds de l'ontologie. Soit w_c l'ensemble des concepts reliés à une capacité du système et w_i l'ensemble des concepts de la commande de l'utilisateur, alors la mesure de distance sémantique est définie de la façon suivante :

$$sim(w_c, w_i) = \left(\sum_{x \in w_c} \sum_{y \in w_i} p(x,y) - weight(y) \right) - uncovering(w_c, w_i)$$

avec $weight(y)$ le poids du concept défini dans l'ontologie, $uncovering(w_c, w_i)$ le nombre de concepts différents entre w_c et w_i et $p(x,y)$ la fonction définie par :

$$p(x,y) = \begin{cases} 3 & \text{Si } x \text{ est la même mot que } y \\ 2 & \text{Si } x \text{ est lié à } y \\ 0 & \text{Sinon} \end{cases}$$

Cette distance permet de gérer l'approximation sémantique au travers de la fonction $p(x,y)$ et du cas « est lié à », ce qui fait de Cederic un des systèmes les plus avancés dans le domaine. De plus, il est capable de définir l'ensemble w_c des concepts qui représentent une capacité et donc d'utiliser la sémantique des actions.

Cependant, l'ontologie est définie selon un langage spécifique, le langage KM (figure 1.14). Or, ce langage définit dans la même structure, les actions, le domaine de l'application (taxonomie), le modèle d'interaction et la syntaxe. Ceci signifie que l'ontologie *est* l'application et ne décrit pas l'application. Ainsi, définir une nouvelle application dépendant du même domaine demande une révision complète de l'ontologie. Enfin, le processus d'attribution des poids par concept est totalement aléatoire et géré par le concepteur. Dans le cas d'application complexe, ceci devient une tâche difficile car le moindre réglage peut déséquilibrer l'ontologie.

1.3.5 Conclusion

Le tableau de la figure 1.8 page 71 résume les différents systèmes que nous venons d'étudier. Nous avons pu constater que les systèmes d'interaction homme-machine ont énormément évolué depuis leurs débuts. Les recherches actuelles ont amené à définir des systèmes de plus en réutilisables et paramétrables. On constate cette évolution à plusieurs niveaux :

1. L'interprétation sémantique. Comme souligné par Allen [Allen *et al.*, 1996], deux méthodes sont applicables : la descendante (*i.e. top-down*) et l'ascendante (*i.e. bottom-up*) :

   (a) L'approche descendante consiste à *construire* une commande formelle à partir de la commande en langue naturelle en considérant uniquement les contraintes structurelles imposées par le modèle formel du langage (*e.g.* [Sadek *et al.*, 1997,

Shapiro, 2000]). Elle est couramment utilisée en dialogue, en situation où il est difficile de prévoir à l'avance l'étendue des commandes ou de déterminer ce qu'il est possible d'effectuer à l'instant courant.

(b) L'approche ascendante (*i.e.* bottom-up) classique utilise une liste *préétablie* de compétences (ou de capacités) et essaye de relier la commande en langue naturelle à une de ces compétences (*e.g.* [Paraiso *et al.*, 2004, Dzikovska *et al.*, 2003]). Cette approche permet au développeur d'écrire des algorithmes génériques, au sens où ils dépendent uniquement du formalisme de la liste de compétences. Un système bottom-up est dynamique si sa liste de compétences est générée dynamiquement au cours de l'exécution du système.

Finalement, les systèmes les plus génériques sont les systèmes *bottom-up dynamique* :

(a) Le principal défaut de l'approche top-down est lié à la difficulté de définir des règles génériques de transformation dans le système. Ainsi, les systèmes actuels définissent un grand nombre de règles spécifiques de l'application, afin de réduire au maximum la création « de commandes formelles impossibles » (évènements ne donnant lieu à aucune action car une des préconditions de l'action est fausse). Par exemple, Shapiro [Shapiro, 2000] utilise ces évènements impossibles pour construire des nouvelles préconditions et les ajouter à la base de connaissance. Dans la plupart des autres systèmes, les évènements impossibles ne sont pas traités et les réponses engendrées par le système se résument à « je ne comprends pas votre commande ».

(b) Dans l'approche bottom-up classique, le principal défaut réside dans les listes de compétences (*e.g.* [Dzikovska *et al.*, 2003, Paraiso *et al.*, 2004]). En effet, elles sont définies de manière statique, le système n'a aucune conscience de ce qu'il est possible ou non de faire dans l'état courant. En pratique, ces listes doivent décrire toutes les situations de dialogues possibles (en tenant compte des erreurs possibles de l'utilisateur) ainsi que la traduction en requête formelle.

(c) Dans les récents travaux à base de *bottom-up dynamique* (*e.g.* [Eliasson, 2007]), seul un ensemble minimal de compétences est fourni à l'avance. Si une commande de l'utilisateur ne s'apparie pas directement ou complètement à une des compétences existantes, le système tente de décrire une nouvelle compétence en se basant sur les éléments présents dans les compétences proches. Le système que nous défendons dans cette thèse est de même un système bottom-up dynamique.

Par ailleurs, dans l'évaluation que nous avons faite de ces trois méthodes, nous avons trouvé des résultats cohérents avec les conclusions précédentes [Mazuel et Sabouret, 2006, Mazuel et Sabouret, 2008a].

2. La base de connaissance. Il est établi que les systèmes doivent maintenant reposer sur une ontologie du domaine pour fonctionner. Son expressivité, ainsi que la méthode utilisée pour extraire les informations, sont encore un point d'étude très ouvert. Si les premiers systèmes utilisaient une taxonomie simple, les études montrent qu'il est nécessaire de développer des modèles intégrant des relations, et bien sûr des méthodes de calcul sémantique intégrant ces relations. Ces idées sont encore neuves et n'ont été

que peu exploitées. Il n'existe ainsi pratiquement pas de mesures sémantiques exploitant les relations et celles existantes le sont dans un cadre très strict et mal évalué. L'un des objectifs principaux de notre système est ainsi d'utiliser une taxonomie augmentée de relations pour l'interprétation et, par la même occasion, de définir des mesures sémantiques sur ce modèle.

3. Le langage de description des actions/capacités du système. Avoir un langage introspectif permet ainsi de pouvoir récupérer le contenu du code dynamiquement pendant l'exécution de l'agent. Ceci évite d'avoir à anticiper toutes les interactions à l'avance et donc de réécrire deux fois le code de l'agent : une fois le code *exécutable* et une fois pour le code *sémantique* d'appariement sémantique. Le langage le plus abouti à ce niveau est le langage Cederic, qui permet de replannifier dynamiquement à partir d'un ensemble de plans de bases quand une commande est légèrement différente de celle prévue. Dans notre système, nous utilisons le langage VDL (décrit en détail dans la section 2.2.1). Ce langage est proche de Cederic, mais nous travaillons directement à partir des actions primaires pour un plan et non en replannifiant. Ceci nous évite la définition de plans à l'avance. L'utilisation d'un langage fortement introspectif nous place ainsi dans les systèmes les plus avancés sur ce point.
4. Le modèle de la commande. La plupart des systèmes utilisent un modèle de la commande à base de règles strictes à redéfinir pour chaque système. Cet ensemble de règles est très lourd à définir et difficilement évoluable d'une application à l'autre. Nous pensons au contraire qu'il est possible de construire un analyseur de surface, *i.e.* un analyseur qui ne construit pas un modèle complet de la phrase mais qui ne déduit que les dépendances entre termes. Nous utilisons la structure de l'ontologie pour déduire les règles de construction de l'analyseur de surface. Ainsi, il n'est plus nécessaire de spécifier explicitement les règles de transformation.

De plus les systèmes dont nous avons parlé ne définissent pas la notion de commandes impossibles ou mal comprises. En effet, si la commande de l'utilisateur est incomplète ou imprécise, comment lancer un dialogue de clarification ? En pratique, les systèmes utilisent une action « clarification » et considèrent que l'utilisateur a lancé cette action. Cette technique amène énormément de réécriture à chaque changement d'application. Nous avons ainsi défini, ce qui à notre connaissance n'as pas été fait dans la littérature, un système générique de gestion de réponses qui ne nécessite pas l'écriture de règles particulières pour les cas limites, telle que les commandes imprécises ou incomplètes.

La section suivante présente un aspect complémentaire aux travaux que nous venons de présenter, car nous étudierons cette fois le cas de l'interaction agent-agent.

	Interaction	Interprétation	Modèle de connaissances	Introspection	Mesure sémantique
Eliza/Alice	Dialogue	Pattern Matching			
Shrdlu	Commande Question	Pattern Matching			
Circuit Fix-It Shop	Commande Assistance	Logique			
Collagen	Assistance	Pattern Matching			
Artimis	Commande	Logique Top-Down			
Trains	Commande	Bottom-up			
Paraiso Barthes	Commande	Bottom-up	X		
SNePS	Commande	Logique Top-Down	SNePS-LOG		
Trips	Commande	Bottom-up	LR	KR	
SmartKom	Commande	Bottom-up	Taxonomie augmentée	M3L	Rel.
Milward	Assistance	Bottom-up	Taxonomie		Hie.
Corese	Question	Bottom-up	Taxonomie	RDF	Hie.
Cederic	Commande	Bottom-up dynamique	KM	Cederic	Hie.

TAB. 1.8 – Un résumé des différents systèmes d'interaction. (« Rel. » pour mesure relationnelle et « Hie. » pour mesure hiérarchique).

Quelques commentaires et rappels pour une meilleure compréhension de ce tableau :

1. Les différents types de modèles de connaissances : une taxonomie augmentée est une taxonomie (*i.e.* une hiérarchie de concepts) augmentée avec des relations non contraintes.
2. Les capacités d'introspection : elles sont différentes en fonction du système. En pratique, le système doit définir un langage spécifique pour décrire les actions du système (KR, M3L, VDL, etc.) qui permettent l'analyse dynamique pour l'interprétation sémantique. Certains de ces langages sont plus introspectif que d'autres, ce qu'il est difficile de faire ressortir sur un tableau. Ainsi les langages KR et M3L sont des langages de descriptions d'interaction plus que des langages de descriptions d'actions. En ce sens, malgré des traitements spécifiques, ils se rapprochent du pattern-matching classique. Les langages Cederic et VDL (notre modèle de langage) permettent une introspection dynamique sur les actions en elles-mêmes. C'est cette capacité qui permet de définir un système bottom-up dynamique.

1.4 Communication agent-agent

Nous considerons ici les agents informatiques au sens des *agents cognitifs* [Ferber, 1995]. Plus particulièrement, un agent cognitif désigne une entité informatique :

- munie d'une représentation des connaissances du monde qu'il manipule (son MRC) ;
- capable de raisonnements complexes issus de l'Intelligence Artificielle ;
- capable de communiquer avec les autres agents présents au sein d'un *système multi-agents* (SMA).

Dans un système multi-agent ouvert, les agents sont amenés à communiquer pour résoudre les tâches et buts qui leurs sont assignés. Or, l'hypothèse même d'environnement ouvert rend impossible la définition à priori d'une seule ontologie partagée par tous les agents [Bylander et Chandrasekaran, 1987]. Ainsi, les recherches actuelles s'orientent vers un modèle où chaque agent possède sa propre ontologie [Laera *et al.*, 2007, Morge et Routier, 2007]. Néanmoins, comme nous l'avions évoqué au début de cet état de l'art, si un agent A communique avec un agent B, il va utiliser sa propre ontologie pour écrire ses messages. L'agent B recevra alors un message formulé selon les termes de l'ontologie de l'agent A, ce qui ne permet pas d'interpréter le message. Le problème de l'hétérogénéité dans les agents se situe donc dans la confrontation des ontologies des deux agents et la définition de méthodes pour les réconcilier.

Cette section propose un aperçu des différents travaux de la littérature traitant du problème de l'hétérogénéité sémantique dans la communication agent-agent. Dans une première partie, nous évoquerons le formalisme FIPA-ACL, qui est actuellement le plus couramment utilisé pour modéliser des interactions entre agents. Nous ferons ensuite un aperçu des principes de base de l'alignement d'ontologie (indépendamment de son application dans les SMA). Puis nous présenterons les différents types d'approches de résolution du problème d'hétérogénéité sémantique en elles-mêmes : les approches à base d'ontologie de référence ou globale (utilisant une troisième ontologie en médiation, ou construisant une troisième ontologie globale), les approches à base de négociation sémantique (définissant un protocole de communication pour construire et valider les alignements) et les approches à base d'optimisation d'alignement (cherchant à optimiser un alignement préexistant grâce aux protocoles de communication).

1.4.1 La communication dans les SMA

1.4.1.1 Principe de la théorie des actes de langages

La théorie des actes de langage considère que parler, c'est agir et non uniquement décrire quelque chose [Austin, 1962]. Austin distingue ainsi plusieurs types « d'actes », en particulier les actes *illocutoires*. Un acte illocutoire est un acte que l'on accomplit uniquement en disant quelque chose à cause du sens induit par les mots utilisés. Par exemple, dans la phrase « Je passe te chercher à 19H », la personne énonçant cette phrase prend l'engagement de venir chercher la personne qui entend la phrase. Il effectue donc un acte d'engagement.

Ainsi, un acte illocutoire peut être associé à différents types d'action. Les travaux de [Searle, 1969] ont ainsi étudié la répartition de ces différents types d'action. Un acte de langage illocutoire sera ainsi associé à un contenu (*e.g.* « aller chercher la personne X à

Paramètre	Rôle du paramètre
performative	Donne le type d'acte communicatif du message
sender	L'émetteur du message
receiver	L'agent réceptionnant le message
reply-to	Le nom de l'agent à qui adresser les réponses éventuelles (et non l'agent émetteur)
content	Le contenu du message
language	Le type de syntaxe formelle utilisée pour écrire le message
encoding	L'encodage (si nécessaire) utilisé pour écrire le contenu du message
ontology	L'ontologie pour l'interprétation sémantique du contenu du message
protocol	Le nom du protocole d'interaction dans lequel est défini ce message
conversation-id	Un identifiant de conversation
reply-with	Identifiant à utiliser pour le message de réponse
in-reply-to	Identifiant du message initiant l'envoie de ce message
reply-by	Le temps limite qu'accepte l'agent émetteur de ce message pour obtenir une réponse

TAB. 1.9 – Présentation des paramètres recommandés dans un message selon le formalisme FIPA ACL.

19H ») et un *performatif* désignant le type d'acte effectué (*e.g.* un « engagement » dans l'exemple précédent).[18]

Dans le cadre de la communication entre machines, le principe s'applique de la même manière. Une machine cherche à communiquer avec une autre pour demander l'exécution d'une tâche, pour décrire son environnement, pour répondre à un autre message, etc. Ainsi, les messages échangés entre des entités informatiques obéissent implicitement aux mêmes règles et possèdent des performatifs et des contenus. La section suivante présente le formalisme FIPA-ACL qui propose un formalisme informatique de la théorie des actes de langages.

1.4.1.2 Actes de langages et SMA : le formalisme FIPA-ACL

Le standard FIPA-ACL[19] (ACL pour *Agent Communication Language*) créée par la FIPA[20] (FIPA pour *Foundation for Intelligent Physical Agents*) est une proposition de normalisation pour la structure des messages échangés dans les SMA. Cette normalisation propose une structuration des messages basée sur une liste d'attribut (tableau 1.9). Ce formalisme repose sur la théorie des actes de langage. L'un des attributs correspond donc au performatif échangé et un autre au contenu du message.

FIPA propose 22 performatifs de base[21] et décrit leur sémantique standard (afin que tous agents répondant au formalisme FIPA comprennent sans ambiguïté le sens du message). Par

[18] De l'anglais *perform*, qui signifie « réaliser, accomplir »
[19] `http://www.fipa.org/specs/fipa00061/`
[20] `http://www.fipa.org/`
[21] FIPA Communicative Act Library Specification : `http://www.fipa.org/specs/fipa00037`

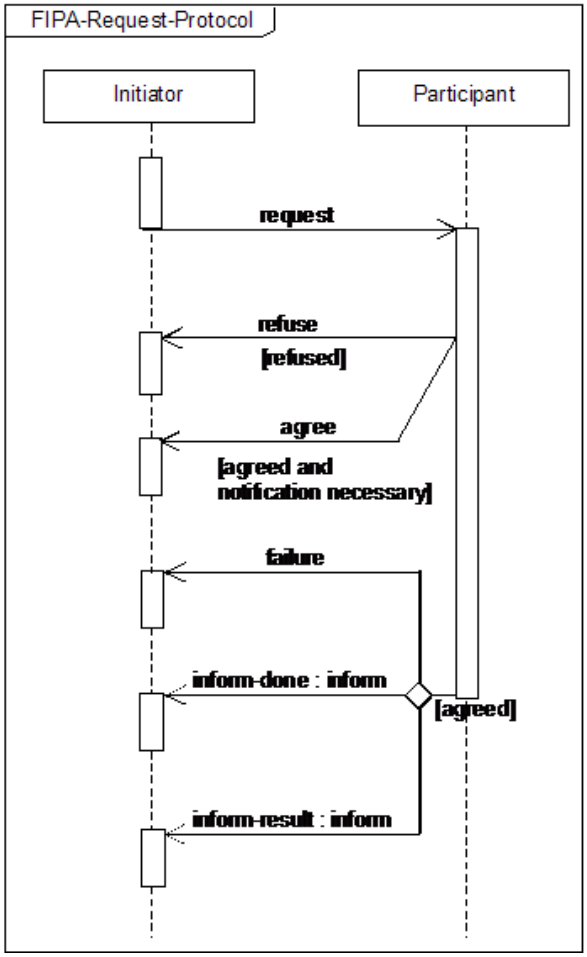

FIG. 1.15 – Un exemple de formalisation d'un protocole simple : le protocole *FIPA-request*

exemple, le performatif *inform* véhicule l'information que le contenu associé au message est une assertion. De plus, les conditions d'utilisation de ces performatifs sont définies au travers de préconditions et d'effets formalisés en logique BDI [Bratman, 1987]. Par exemple, pour le performatif *query-if*, les préconditions sont que l'agent émetteur i ne doit pas savoir la valeur de vérité d'une affirmation ϕ et doit croire qu'un autre agent j peut fournir l'information sur la valeur de ϕ. Les effets de ce performatif sont que l'agent i reçoit de j la valeur de ϕ (vraie ou fausse).

D'autre part, FIPA propose aussi un certain nombre de propositions de protocoles d'interactions. Un protocole d'interaction est la description d'une série ordonnée de messages que peuvent s'échanger les agents (classée selon leur performatif) pour accomplir un objectif. Par exemple, le protocole *request*[22] propose de normaliser une suite d'échanges entre agents en vue de l'exécution d'une requête. Les protocoles s'expriment sur des diagrammes de séquences UML (figure 1.15), représentant les envoies de messages et l'ensemble des solutions possibles pour chaque type de messages [Odell *et al.*, 2001]. Ainsi, dans le protocole *FIPA-request*, la réception d'un message *request* autorise cinq performatifs de réponses : *refuse* si l'agent refuse

[22]FIPA Request Interaction Protocol Specification : `http://www.fipa.org/specs/fipa00026`

la requête, *inform-done* pour informer que la requête a été traitée, etc. Le protocole *request* présenté en exemple est un protocole simple à deux messages maximum, mais certains protocoles sont beaucoup plus complexes et couvrent beaucoup plus de messages, proposant même parfois des boucles.

L'un des attributs de la normalisation FIPA concerne le modèle de représentation des connaissances utilisé par l'agent émetteur. En effet, chaque agent d'un système possède sa propre ontologie. Pour interpréter un message, un agent récepteur doit ainsi tenir compte de l'ontologie de l'agent émetteur. La section suivante présente le principe de l'alignement d'ontologie, qui est en général utilisé comme prémisse aux méthodes de gestion de l'hétérogénéité sémantique que nous commencerons à présenter à la section 1.4.3.

1.4.2 Aperçu des méthodes pour l'alignement d'ontologie

Le problème *d'alignement d'ontologies* consiste à trouver un ensemble d'associations entre un ensemble de concepts d'une ontologie A et un ensemble de concepts d'une ontologie B. Plus formellement, étant donné deux ontologies A et B, un alignement consiste à trouver une fonction d'association m (pouvant être de différents types, tel que « équivalent », « est plus général que », etc.) entre un concept $a \in A$ et un concept $b \in B$ tel que si $a = m(b)$ alors a et b sont considérés sémantiquement associés par la relation m.

Cette section n'a pas pour vocation de fournir un descriptif complet et exhaustif des différents modèles proposés dans la littérature (le lecteur intéressé trouvera des états de l'art complets et détaillés [Kalfoglou et Schorlemmer, 2005, Euzenat et Shvaiko, 2007] et de manière générale sur le site dédié à l'alignement d'ontologie[23]), mais un aperçu des différents types de méthodes et d'approches qu'il est possible de mettre en place. Dans le cadre de cette thèse, nous n'étudierons pas de nouvelles solutions à ce problème, mais nous nous situerons au contraire comme utilisateur des méthodes d'alignement existantes comme outils pour la gestion de l'hétérogénéité sémantique.

Plusieurs types d'approches ont été proposés dans la littérature et peuvent être segmentés en trois catégories (c.f. figure 1.16) [Kalfoglou et Schorlemmer, 2005] :

- Les méthodes structurelles, qui reposent sur la structure en graphe de l'ontologie et les labels des concepts ;
- Les méthodes à base d'instances, qui utilisent les ensembles d'instances communs aux deux ontologies pour inférer les équivalences entre les concepts ;
- Les méthodes par ontologie référente, qui utilisent une troisième ontologie médiatrice pour réconcilier les deux ontologies.

La section suivante présente d'abord l'alignement lexical (*i.e.* ou *ancrage* des termes) qui est systématiquement utilisé comme phase préliminaire d'alignement. Ensuite, nous présenterons succinctement les trois approches de base, l'alignement structurel, l'alignement à base d'instance et l'alignement à base d'ontologie de référence.

1.4.2.1 Ancrage lexical

L'ancrage lexical est la recherche d'un alignement en ne se basant que sur les labels des concepts dans les deux ontologies. Il est, dans la majeure partie des situations, l'étape

[23] `http://www.ontologymatching.org`

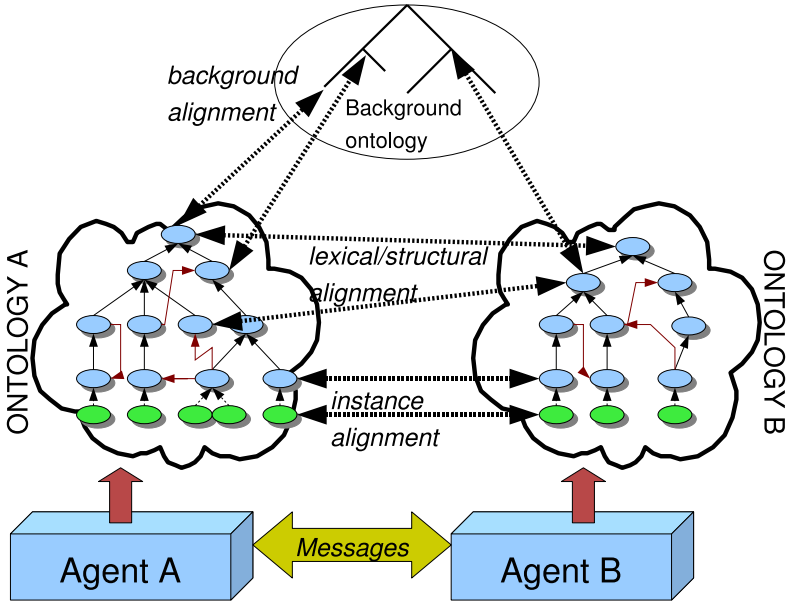

FIG. 1.16 – L'alignement d'ontologie est un ensemble d'équivalences entre des nœuds des deux ontologies. Ce schéma représente les trois solutions classiques autour desquels s'articulent les solutions pratiques actuelles : l'alignement à base d'instances, l'alignement structurel et l'alignement à base d'ontologie référente.

préliminaire à tout alignement entre deux ontologies, les étapes suivantes servant à raffiner et préciser l'ancrage lexical.

Il existe plusieurs techniques pouvant être utilisées, chacune avec ses avantages et ses inconvénients. Premièrement, les outils classiques de traitement automatique des langues tels que la lemmatisation (qui réduit un terme à sa forme singulière ou infinitive, tel que *innovantes* vers *innovant* ou encore *choisies* vers *choisir*) ou l'utilisation des abréviations, suffixe/préfixe (par exemple *net* pourra être considéré comme équivalent à *network*, *ID* équivalent à *PID*). Néanmoins, ces approches ont des limites qui nécessitent ainsi les étapes supplémentaires décrites dans les sections suivantes pour corriger cet ancrage. Par exemple, la lemmatisation peut être ambiguë (*left* est-il le verbe *leave* ou l'adjectif *left* ?), l'approche préfixe/suffixe est très dépendante du langage (par exemple, *hotel* peut être apparié à *hot* et *word* peut être apparié à *sword*). Ces erreurs impliquent le besoin de vérification et de validation qui ne fait de cette étape qu'une étape préliminaire dans la recherche d'un alignement entre deux ontologies.

Une approche complémentaire pouvant être rangée dans la catégorie lexicale est l'utilisation de mesure de distance syntaxique, telles que les distances d'édition (distance de Hamming ou distance de Levenshtein). Par exemple, la distance de Levenshtein calcule le nombre minimum d'opérations nécessaires (insertion, délétion ou remplacement de lettre) pour passer d'une chaîne de caractères à une autre. Cette distance est souvent utilisée pour de la correction orthographique, car elle permet de calculer le mot le plus « proche » du mot tapé. L'avantage intéressant de cette méthode est quelle permet de reproduire à moindre coût la fonction d'un lemmatiseur (retirer un pluriel en « s » ne coûte qu'une opération de délétion) et d'être résistante aux problèmes d'orthographe, d'utilisation de majuscule, etc. Néanmoins

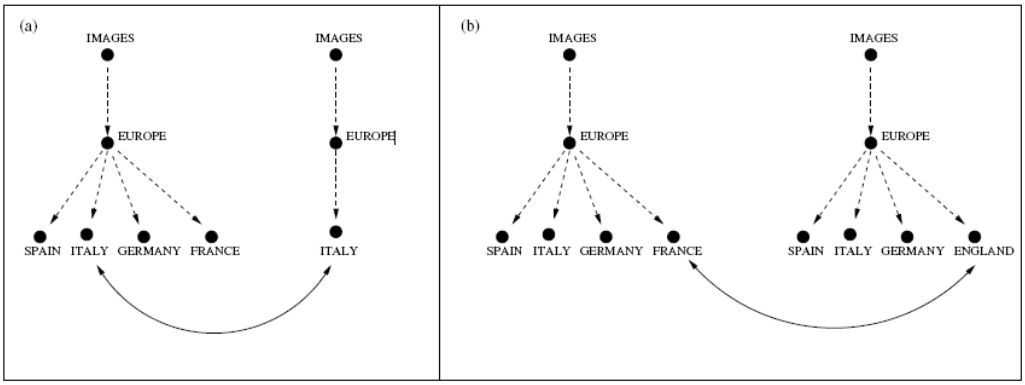

FIG. 1.17 – Exemple d'erreur possible lors d'un alignement totalement structurel [Bouquet *et al.*, 2003].

certains problèmes demeurent, tels que l'association *word* - *sword* qui possède un très bon score de Levenshtein puisqu'il n'y a qu'une lettre à modifier.

1.4.2.2 Approches structurelles

Les approches structurelles sont basées sur une comparaison des graphes de concepts représentants les ontologies. Basées en général sur un appariement lexical préliminaire, ces méthodes vérifient la validité d'un alignement obtenu en vérifiant les nœuds déclarés autour du concept visé [Breitman *et al.*, 2005, Bouquet *et al.*, 2003]. Ainsi, dans le système CATO [Breitman *et al.*, 2005], les auteurs utilisent un algorithme appelé *TreeDiff* pour calculer « la plus grande sous-structure commune entre les arbres ».

Par exemple, les méthodes structurelles permettent d'aligner les termes *PC* et *Personal Computer* en se basant sur la classification de ces deux concepts dans les deux ontologies (ce qu'un alignement lexical ne pourrait pas faire). Néanmoins, tel qu'il apparaît sur la figure 1.17, l'alignement structurel peut présenter des erreurs si l'information est classée de manière similaire dans les deux ontologies, mais qu'elle n'est pas équivalente. Ce problème est d'autant plus présent que les ontologies ne partagent que peu de concepts.

1.4.2.3 Approches à base d'instances

L'objectif de ces méthodes est de déterminer un alignement en utilisant les instances communes entre les deux ontologies. Ainsi, si deux concepts de deux ontologies partagent exactement et précisément les mêmes instances, alors ces deux concepts peuvent être considérés équivalents. Par exemple, dans [Ichise *et al.*, 2003], les auteurs essayent d'aligner les hiérarchies de catégories de Google et Yahoo ! en se basant sur la distribution des sites internes comme instances (une instance étant identifiée par l'URL d'un site). Dans les travaux de [van Diggelen *et al.*, 2006c], la recherche d'instances communes et d'instances non-communes entre deux concepts permet de déterminer des relations d'équivalence et de subsomption (si l'un des deux ensembles d'instances contient entièrement l'autre). Par exemple, sur la figure 1.18, les instances du concept *roadVehicle* sont incluses dans les instances du concept

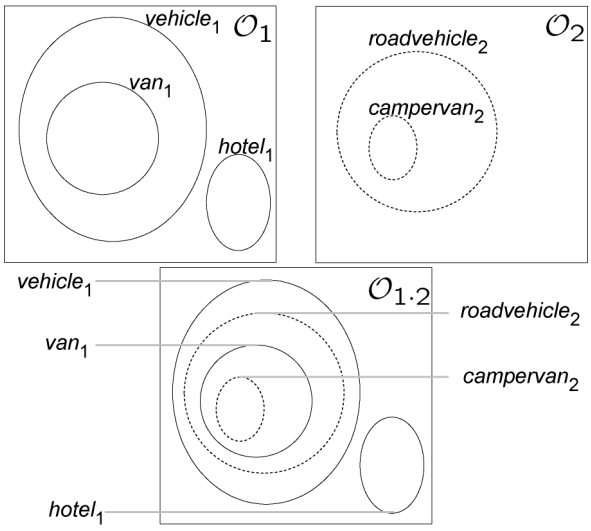

FIG. 1.18 – L'alignement à base d'instance peut aussi permettre de construire des ensembles de subsomption en plus d'ensemble d'équivalence [van Diggelen *et al.*, 2006c].

vehicle alors que l'inverse est faux. Il est ainsi possible de déduire que *roadVehicle* est une sous-classe de *vehicle*.

Néanmoins, l'inconvénient principal de cette approche est la détection des instances. Par exemple, dans les travaux de [Ichise *et al.*, 2003], seuls 10% des sites ont pu être correctement identifiés comme étant les mêmes instances, dû à des formulations diverses des URLs de sites. Dans les travaux de [van Diggelen *et al.*, 2006c], il est ainsi difficile de conclure si l'intersection des ensembles d'instances n'est pas complet pour déterminer qui est une sous-classe ou super-classe de qui.

1.4.2.4 Approches à base d'ontologie référente

Les approches à bases d'ontologie référente sont basées sur l'utilisation d'une troisième ontologie comme ontologie médiatrice du processus d'alignement [Aleksovski *et al.*, 2006b, Bouquet *et al.*, 2003][24]. L'avantage principal de cette méthode est qu'elle est très efficace et robuste dans les cas où les structures des ontologies sont très différentes ou si les instances ne sont pas disponibles [Aleksovski *et al.*, 2006a]. Ainsi, l'une des hypothèses principales (qui est aussi par la même occasion l'un des principaux inconvénients de l'approche) est d'avoir accès à une ontologie médiatrice avec suffisamment d'informations pour pouvoir ancrer les termes des deux ontologies à aligner. L'ancrage des deux ontologies vers l'ontologie médiatrice est en général un simple ancrage lexical.

L'une des difficultés majeures une fois les ontologies ancrées sur l'ontologie médiatrice est d'exploiter le contenu de cette ontologie et en particulier de considérer les chemins sémantiquement corrects (notion que nous avions déjà défini dans la section 1.2.3.4.1). Par exemple, sur la figure 1.19, suivre le lien *isMoreGeneral* permet aux auteurs de trouver un alignement entre les concepts « Aorta thoracalis dissection » (ontologie A) et « Dissection of artery »

[24]Les travaux de [Bouquet *et al.*, 2003] sont essentiellement structurels (section 1.4.2.2). Néanmoins, ils utilisent aussi WordNet comme médiateur pour résoudre certaines ambiguïtés.

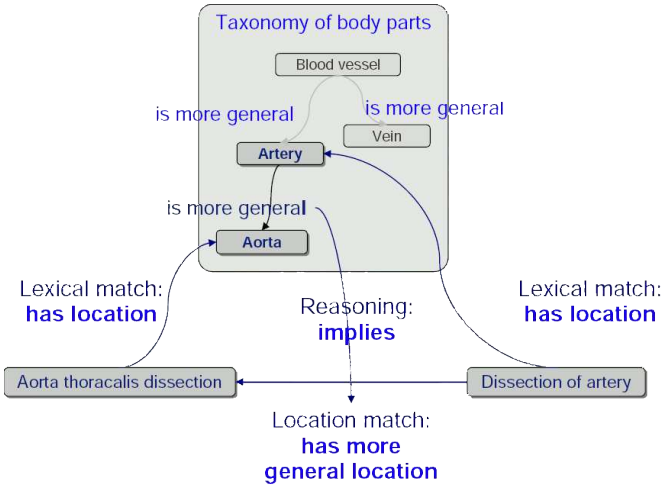

FIG. 1.19 – Exemple d'alignement obtenu grâce à l'utilisation d'une ontologie référence [Aleksovski *et al.*, 2006b].

(ontologie B).

1.4.2.5 Conclusion

Il n'y a pas de méthode d'alignement d'ontologies meilleure qu'une autre, mais des méthodes plus adaptées à certains types d'ontologies que d'autres (*e.g.* ontologie avec/sans instances, ontologie fortement/faiblement structurée, etc.). Le problème d'alignement d'ontologies est un problème qui commence à être bien maîtrisé. Ainsi, plusieurs livres traitant de l'état de l'art global commencent à apparaître et montrent la grande cohérence et complétude du domaine actuellement [Kalfoglou et Schorlemmer, 2003, Euzenat et Shvaiko, 2007].

Les progrès de l'alignement sont suffisamment avancés pour qu'ils puissent être utilisés comme outils pour la résolution de problèmes plus complexes, ne mettant en œuvre l'alignement d'ontologie que comme une partie de la solution. Par exemple, dans la communication agent-agent, l'alignement entre les ontologies des agents peut être maintenant affecté à un service externe d'alignement d'ontologies, sans que le fonctionnement de ce service ne rentre dans la description de la solution pour la communication [Laera *et al.*, 2007]. C'est ce point de vue que nous avons aussi choisi de suivre et que nous utiliserons dans la description de nos travaux.

Néanmoins, n'importe quel système d'alignement d'ontologie n'est pas directement applicable à la communication dans les SMA. En effet, ces systèmes doivent respecter certaines hypothèses :

- L'alignement entre les ontologies doit être calculable *à la volée* et *en temps limité*. Tout d'abord, il doit être *à la volée* car l'hypothèse d'un SMA ouvert ne supporte pas la possibilité que les alignements puissent être calculés à l'avance, avant le démarrage du SMA. Ensuite, il doit être *en temps limité* pour ne pas ralentir les temps de communication entre agents, qui risquerait de poser des problèmes de synchronisation entre agents.
- Les agents qui communiquent partagent des buts et des capacités communes. En pra-

tique, cette hypothèse implique que l'intersection de deux ontologies de deux agents n'est pas vide. Ainsi, nous supposerons qu'il existe toujours un alignement acceptable entre deux ontologies de deux agents. Cependant, il n'est pas possible d'affirmer que les agents posséderont exactement les même concepts au même niveau de spécialisation. Par exemple, l'une des ontologies peut n'avoir qu'une seule classe pour décrire le concept *publication scientifique*, alors qu'un autre agent ne définirait pas ce concept mais directement les concepts *publications dans une revue, publication dans une conférence,* etc.

- L'alignement entre les ontologies doit être *automatique* (alors que la plupart des travaux proposent des approches semi-automatiques). Ainsi, les agents doivent évaluer et valider l'alignement sans la décision d'un expert humain.

La prochaine section présente quelques systèmes représentatifs des travaux du domaine sur la problématique de l'hétérogénéité sémantique entre agents. Un moyen de classifier ces différents travaux est de considérer leur gestion du *Weak Alien Problem*. Le *Weak Alien Problem* (ou *WAP*) fait référence au fait que pour communiquer, deux agents doivent impérativement avoir une partie de leurs connaissances en commun [Sansonnet et Valencia, 2004].[25] Cette hypothèse communément admise peut s'exprimer sous différentes formes et nous permet de classifier les approches de la littérature en trois parties :

- Les approches à ontologie référente, dans lesquelles les agents ont accès à une ontologie de référence « publique » partagée ou cherchent à construire cette ontologie partagée. Cette ontologie partagée représente les connaissances communes aux agents.
- Les approches à base de négociation sémantique, dans lesquelles les agents suivent un protocole de communication particulier pour obtenir les équivalences entre les termes des deux ontologies.
- Les approches basées uniquement sur de la recherche d'alignement entre les ontologies des deux agents. Dans ce type d'approche, la recherche d'un alignement optimal suffit à gérer le problème.

Ces trois parties font l'objet des trois prochaines sections.

1.4.3 Système Multi-Agents et ontologie référente/globale

1.4.3.1 Stuckenshmidt & Timm

Dans [Stuckenschmidt et Timm, 2002], le fait que deux agents voulant communiquer aient forcément une intersection de leurs connaissances non nulle se traduit par l'existence d'une ontologie de référence « publique » accessible par tous les agents. Ainsi, chaque agent possède sa propre ontologie privée ainsi qu'un alignement (fait à la main avant l'exécution du système) vers l'ontologie de référence. Néanmoins, à la différence des approches classiques à ontologie de référence, l'objectif n'est pas ici de déduire un alignement entre les ontologies privées en utilisant les alignements vers l'ontologie de référence, mais de traduire toute commande d'un agent exprimé dans son ontologie privée en utilisant les termes de l'ontologie de référence. L'agent récepteur du message possédant aussi un alignement vers l'ontologie de référence peut alors traduire la commande en sa propre ontologie privée.

[25] Le *Strong Alien Problem* (*SAP*), dans lequel les agents ne partagent aucune connaissances n'est, par définition, pas solvable.

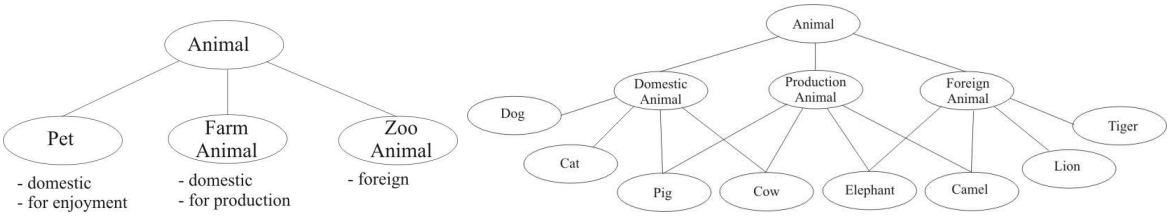

FIG. 1.20 – Exemple d'une ontologie privée d'un agent à gauche et d'une ontologie publique à droite. L'agent doit connaître un alignement entre ces deux ontologies avant son exécution [Stuckenschmidt et Timm, 2002].

Plus formellement, l'ensemble de l'approche est basé sur la logique propositionnelle. Par exemple, supposons les ontologies publiques et privées décrites sur la figure 1.20, alors l'alignement suivant est un exemple d'alignement qu'il est possible de donner avant l'exécution de l'agent :

$$\begin{aligned} ZooAnimal &\sqsubseteq \neg DomesticAnimal \\ Pet &\sqsubseteq DomesticAnimal \wedge \neg ProductionAnimal \\ FarmAnimal &\sqsubseteq DomesticAnimal \wedge ProductionAnimal \end{aligned}$$

Alors, en utilisant un système simple d'inférence logique, il est possible de traduite la requête utilisant les termes de l'ontologie privée : « Give me information about *Farm-Animal* and *Animal* that are not *Pets*. » en la requête utilisant les termes de l'ontologie publique : « Give me information about *Cows* and *Pigs* and *Animals* that are not *Domestic-Animal* ».

L'avantage de cette proposition est qu'elle propose une solution simple pour l'hétérogénéité sémantique basée des inférences logiques de subsomption (donc rapide à traiter). Néanmoins, elle repose aussi sur l'hypothèse forte qu'il existe une ontologie de référence et que chaque agent du système possède un alignement vers celle-ci. Pour éviter le problème d'une très grande ontologie de référence pour tous les agents, l'auteur argumente qu'un agent peut connaître plusieurs alignement vers plusieurs ontologies de référence et que l'ontologie de référence sera alors choisie en fonction de celles connues par les deux agents communiquant. Le problème de la définition d'un alignement préalable se pose alors d'autant plus qu'il y a un grand nombre d'ontologies de références.

1.4.3.2 Valencia et Sansonnet

Le modèle de Valencia et Sansonnet repose sur l'utilisation d'outils de topologie algébrique pour résoudre le problème de l'hétérogénéité sémantique dans la communication [Sansonnet et Valencia, 2004, Valencia et Sansonnet, 2004]. Plus précisément, la base de connaissances des agents est modélisée sous la forme d'un modèle simpliciel. Un modèle simpliciel repose sur la notion de *simplexe*. Soit $s = \{a_0, a_1, ..., a_p\}$ un ensemble géométriquement in-

Tb_i	*titi*	*coco*	*milou*	*paloma*	*carla*
yellow	1	0	0	0	0
green	0	1	0	0	1
white	0	0	1	1	0
wings	1	1	0	1	0
tail	0	0	1	0	1
hair	0	0	1	0	0

$\Longrightarrow$

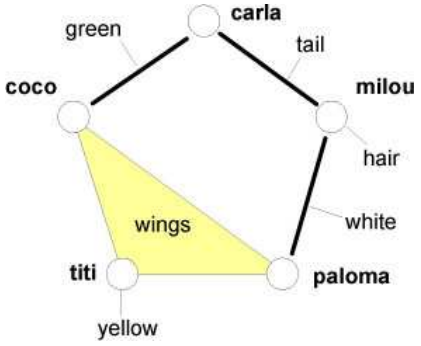

FIG. 1.21 – Conversion d'une relation entre instances et attributs vers une représentation simpliciale.

dépendant dans R^n. Le *p-simplexe* géométrique engendré par s (noté $\langle s_p \rangle$) est défini par :

$$\langle s_p \rangle = \sum_{i=0}^{p} \lambda_i a_i \text{ tel que } \sum_{i=0}^{p} \lambda_i = 1 \text{ avec } \forall i \in [1,p]\,,\ \lambda_i \in]0,1[$$

Ainsi, un 0-simplexe est un point, un 1-simplexe une arête, etc. Une collection de simplexes (*i.e.* un *complexe simplicial*) peut être utilisée comme modèle de représentation des connaissances. En effet, il est possible d'écrire des règles de conversions permettant de traduire toutes relations binaires d'un ensemble A vers un ensemble B en un simplexe (*e.g.* figure 1.21). Valencia a ainsi proposé dans sa thèse un ensemble complet de règles pour traduire une base de connaissances formalisée en logique de description (DL, [Franz Baader *et al.*, 2003]) vers un complexe simplicial [Valencia, 2000].

Deux agents possèdent ainsi deux complexes simpliciaux comme représentation des connaissances (que nous noterons KB_A pour l'agent A et respectivement KB_B pour le deuxième agent B). En considérant l'hypothèse du WAP décrite précédemment, les auteurs partent du principe qu'il existe une partie de chaque complexe simplicial qui est commune aux deux agents. Cette partie (appelée *germe*[26] et notée G) est calculée dynamiquement en utilisant des méthodes d'alignement morphologique telles que celles présentées dans la section 1.4.2.1. Ainsi, chaque agent a une partie *cachée* de sa base de connaissances (appelé $KB_{A,c}$ pour l'agent A et respectivement $KB_{B,c}$ pour l'agent B) et une partie commune qui correspond à la partie G.

Comme nous l'évoquions précédemment, une requête est exprimée selon la base de connaissances de l'agent émetteur A. Ainsi, il peut exister des concepts qui ne sont définis que dans la partie cachée $KB_{A,c}$ de la base de connaissances de l'agent A. Pour que l'agent receveur B comprenne ces concepts, les auteurs proposent de suivre un chemin sémantique de la partie cachée $KB_{A,c}$ de l'agent émetteur A vers la partie commune G, puis de suivre un chemin sémantique de la partie commune G vers la partie cachée $KB_{B,c}$ de B. La définition de ce chemin (*i.e.* une *q-chaîne* dans le vocabulaire des complexes simpliciaux) se fait en deux étapes. Premièrement, une q-chaîne est calculée entre la partie cachée $KB_{A,c}$ de A et la partie commune G. Sachant qu'avec le modèle simpliciel un concept est représenté par un

[26] D'après le terme anglais *ground* utilisé notamment pour parler des ontologies de références : *background ontology*.

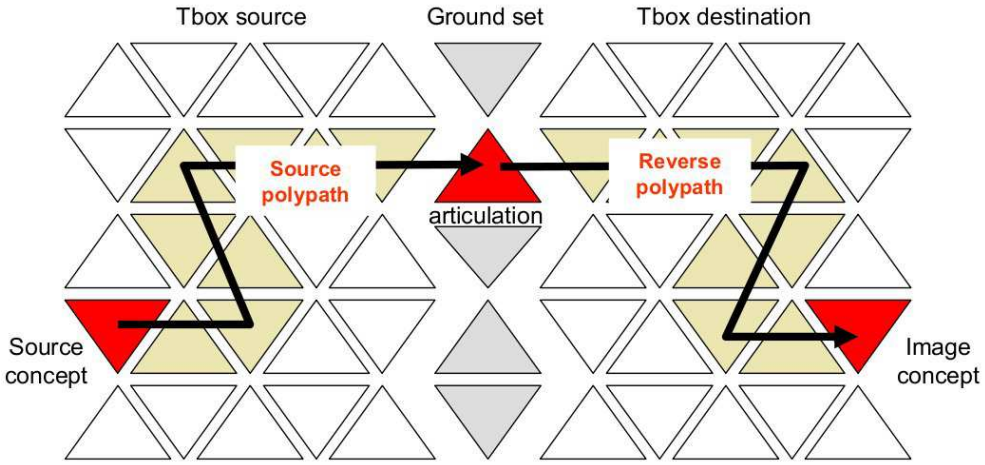

FIG. 1.22 – Exemple de q-chaîne entre deux bases de connaissances (notées ici T_{Box}) en considérant la partie commune G (noté *ground set)* en utilisant des complexes simpliciaux de dimension 2.

complexe simplicial, la construction de la q-chaîne suit deux contraintes, choisies pour limiter la quantité d'informations perdue par la modélisation de cette chaîne :

1. Le passage d'un complexe simplicial à un autre doit se faire en maximisant le nombre de faces communes ;
2. La q-chaîne doit être la plus courte possible, tout en respectant la première contrainte.

Une fois cette première chaîne calculée, il faut continuer la chaîne de la partie commune G vers la partie cachée $KB_{B,c}$ de B. Les auteurs proposent alors de suivre une q-chaîne symétrique à la première q-chaîne (cette opération est illustrée sur la figure 1.22). La nouvelle q-chaîne doit ainsi (dans la mesure du possible) :

1. Avoir des faces de transition entre complexes simpliciaux de même cardinalité que dans la première q-chaîne ;
2. Avoir la même longueur que la première q-chaîne.

Le concept désigné à l'arrivée dans la partie cachée $KB_{B,c}$ de B par la q-chaîne est jugé similaire au concept de départ de la partie cachée $KB_{A,c}$ de A. En considérant cet algorithme sur l'ensemble d'une requête, les auteurs peuvent ainsi traduire la requête formulée selon la base de connaissances KB_A en une requête selon la base de connaissances KB_B (et ainsi gérer le problème d'hétérogénéité sémantique).

L'originalité de cette approche est qu'elle utilise un modèle de topologie algébrique pour représenter les connaissances, ce qui permet de mettre en place de nouvelles solutions pour la recherche d'alignement entre concepts. De plus, ce formalisme permet de modéliser toutes bases de connaissances (des logiques de descriptions en passant par la logique des défauts [Valencia, 2000]). Néanmoins, la méthode de recherche de chemin sémantique par analogie pose les mêmes difficultés que les travaux sur l'alignement structurel rappelés dans la section 1.4.2.2. En effet, rien n'indique lorsque l'on suit un chemin purement structurel qu'il ait un sens sémantiquement (la figure 1.17 que nous avions présentée en est un exemple). De plus, l'algorithme mis en place pour la construction de cette q-chaîne est fortement combinatoire et

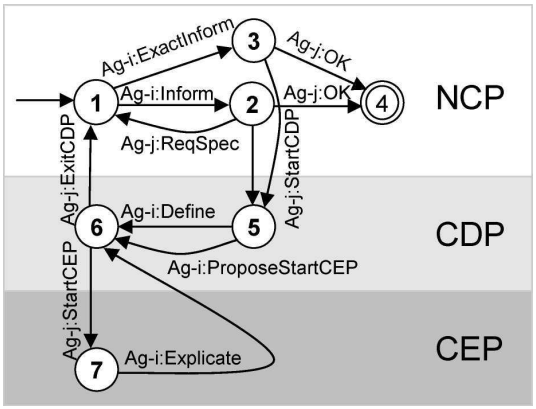

FIG. 1.23 – Protocole d'interaction du système Anemone. Le protocole est séparé en trois couches en fonction de la difficulté à interpréter un concept. La couche NCP (*i.e.* Normal Communication Protocol), CDP (*i.e.* Concept Definition Protocol) et CEP (*i.e.* Concept Explication Protocol).

l'exploration complète de toute les solutions est difficile [Sansonnet et Valencia, 2005]. Pour résoudre ce problème de temps de calcul, les auteurs proposent d'utiliser des modèles de telle sorte que la part commune G des deux bases de connaissances soit grande (pour limiter la longueur du chemin dans la partie cachée) ou d'utiliser des heuristiques pour couper certaines recherches (ce qui pose toujours le problème de la qualité et de l'optimalité de la solution atteinte).

1.4.3.3 Anemone (van Diggelen)

Le système Anemone est un système à mi chemin entre une approche par négociation sémantique (approches basées sur la description d'un protocole de communication, développées dans la section 1.4.4 prochaine) et la création d'une ontologie globale. Ainsi, l'objectif du système Anemone (*i.e.* AN Effective Minimal Ontology Negotiation Environment) [van Diggelen *et al.*, 2006a] est de proposer un protocole de communication agent permettant la création d'une unique ontologie partagée utilisable pour communiquer sans problème d'hétérogénéité sémantique.

Le protocole de communication entre agents est défini en trois niveaux en fonction de la difficulté à obtenir un accord pour l'alignement de deux concepts donnés (figure 1.23) :

- Le niveau NCP (*i.e.* Normal Communication Protocol). Ce niveau correspond à un échange classique de messages entre les agents. Si un concept n'est pas compris, l'agent demande une définition du concept (*i.e.* performatif *StartCDP*) et nous passons au niveau suivant
- Le niveau CDP (*i.e.* Concept Definition Protocol). Un agent arrive dans ce niveau de communication s'il a besoin d'une définition pour un concept qu'il ne comprend pas. Une définition donnée par un agent A correspond à un alignement que A connaît avec d'autres concepts d'autres ontologies (*e.g.* l'agent A peut ainsi envoyer un message $define(equiv(food, nutrition))$ si son concept $food$ correspond au concept $nutrition$ d'une autre ontologie. Ceci peut permettre à l'agent récepteur, en fonction des aligne-

ments qu'ils connaissaient déjà, de déduire le sens du concept qu'il n'avait pas compris. Si l'agent ne comprend pas la définition, alors il demande une explication du concept (niveau suivant, performatif *StartCEP*).

- Le niveau CEP (*i.e.* Concept Explication Protocol). Ce niveau gère la situation où l'agent ne comprend pas un concept ni sa définition. Dans ce cas, l'alignement est résolu par une méthode de comparaison d'instances, telle que nous l'avons déjà discutée dans la section 1.4.2.3 sur l'alignement d'ontologie.

L'ensemble de l'architecture de communication est pensé pour qu'un groupe de n agents construise une ontologie commune de communication, ce qui explique l'importance dans leurs travaux d'expliquer un concept en utilisant les équivalences trouvées par rapport aux ontologies des autres agents. De même, une partie de l'algorithme repose sur le choix de la bonne explication sachant qu'il peut exister $m < n$ explications possibles (la bonne explication est alors celle qui est statistiquement le plus souvent utilisée par l'ensemble des agents).

L'avantage d'Anemone est de proposer une solution réellement multi-agent alors que la plupart des autres solutions focalisent en particulier sur un problème de communication à deux agents. Néanmoins, l'approche propose la construction d'une ontologie unique et commune, ce qui à long terme pose des problèmes de dynamicité et de révision de connaissances. En effet, les connaissances des agents sont amenées à être révisées avec le temps, et les alignements calculés ne seront donc plus valables.

Finalement, les approches reposant sur la construction ou l'utilisation d'une troisième ontologie permettent de gérer la communication sur un grand groupe d'agents (en particulier si tous les agents utilisent la même ontologie de référence) ou d'éliminer le problème d'alignement (les concepts communs suffisent alors à assurer la liaison entre concepts spécifiques des deux ontologies). Néanmoins, ces approches posent le problème de la dynamicité de l'ontologie (qui devrait varier au cours du temps) ou encore de la définition d'une ontologie globale imposante devant couvrir tous les domaines de tous les agents.

1.4.4 Système Multi-Agents et négociation sémantique

1.4.4.1 Morge & Routier

L'une des hypothèses principales des travaux de [Morge et Routier, 2007] est qu'il n'est pas envisageable de considérer comme systématiquement possible d'aligner les ontologies des agents en jeux. Le principal problème qu'ils voient dans l'alignement est l'impossibilité de garantir qu'il sera correct ou complet. Or, si l'alignement est imparfait la communication est en général totalement impossible (ce point constitue d'ailleurs l'une de nos hypothèses de travail dans le cadre de cette thèse).

Ainsi, les auteurs pensent qu'il faut gérer le problème de l'hétérogénéité sémantique directement en cours de communication, par un protocole adapté de négociation sémantique. L'agent émetteur (le *customer*) envoie des requêtes à un agent de service récepteur (le *provider*). Chacun des agents peut utiliser un certain nombre de performatif (*question, request, assert, propose, refuse, reject, unknow, concede, challenge* et *withdraw*) dans un certain ordre (tableau 1.10) pour argumenter sa vue du monde et ses croyances personnels.

L'exemple présenté dans l'article est une résolution d'un problème d'hétérogénéité similaire au « diamant de Nixon » [Touretzky, 1986] (figure 1.24). Les deux agents ont la même

Performatif	Réponse pour négociation	Réponse pour abandon
$question(\phi)$	$assert(\phi)$ $assert(\neg\phi)$	$unknow(\phi)$
$request(\phi(x))$	$propose(\phi(a))$	$unknow(\phi(x))$
$assert(\Phi)$	$challenge(\phi(x)), \phi \in \Phi$	$concede(\Phi)$ $reject(\phi), \phi \in \Phi$
$challenge(\phi)$	$assert(\Phi), \Phi \vdash \phi$	$withdraw(\phi)$
$unknow(\Phi)$	$\emptyset$	$\emptyset$
$concede(\Phi)$	$\emptyset$	$\emptyset$
$refuse(\Phi)$	$\emptyset$	$\emptyset$
$withdraw(\Phi)$	$\emptyset$	$\emptyset$

TAB. 1.10 – Performatifs de [Morge et Routier, 2007] et réponses autorisées.

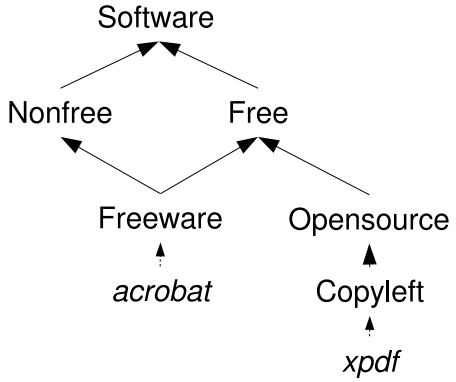

FIG. 1.24 – Ontologie présentée dans l'article de [Morge et Routier, 2007].

ontologie, mais ils ont des préférences sur la validité des arêtes. Par exemple, l'agent 1 va considérer que l'assertion $Nonfree(x) \sqsupseteq Freeware(x)$ est plus vraie que l'assertion $Free(x) \sqsupseteq Freeware(x)$ et inversement pour l'agent 2. Ainsi, lorsque l'agent 2 propose l'assertion $Free(acrobat)$ (basée sur ses préférences sur le rôle des arêtes), l'agent 1 la refuse. La négociation se termine ainsi par le choix du logiciel *xpdf* qui ne pose pas d'ambiguïté.

La force majeure de cette approche est de proposer un protocole de communication basé sur des performatifs agents pour résoudre le problème de l'hétérogénéité sémantique. En effet, le problème d'un alignement incorrect est un problème qui est trop peu pris en compte dans la littérature, qui soutient souvent que résoudre l'alignement résout entièrement le problème. Néanmoins, le défaut majeur de cette approche réside dans la simplicité de la situation, du modèle et de son manque d'évaluation. De plus, les agents possèdent une ontologie quasiment identique et les auteurs considèrent que l'échange d'axiomes logiques ne pose pas de problème d'interprétation. Il apparaît ainsi clairement qu'ignorer totalement le problème de l'alignement d'ontologies n'est pas non plus une solution.

1.4.4.2 Hisene2

Le système HISENE2 [Garruzzo et Rosaci, 2006] est un système très proche du système Anemone présenté à la section 1.4.3.3. L'objectif est là encore de proposer un protocole de

négociation sémantique permettant à deux agents de se mettre d'accord sur les équivalences entre les termes de leurs ontologies. Pour cela, les agents construisent des *explications* des concepts qui leurs sont inconnus en utilisant le protocole de négociation et l'ontologie des autres agents. L'une des originalités de l'approche est qu'elle prend en compte la notion de *réputation* permettant d'influer sur le cours de la négociation sémantique en fonction de la *confiance* qu'un agent porte en un autre agent (nous ne présenterons pas ce modèle de réputation ici car ce n'est pas l'objet de ce travail).

Un concept de l'ontologie d'un agent HISENE est simplement défini en fonction d'un ensemble de termes. Par exemple, le concept *Book* peut être défini par l'ensemble des termes $(title, author, price)$. Une *explication* d'un concept pour un agent peut être vue comme un tuple $e = \langle E, C, ea, c\rangle$ tel que E est un ensemble de concepts de l'ontologie expliquant e, C est l'ensemble des agents étant d'accord avec cette explication, ea l'agent ayant proposé cette explication et $c \in [0, 1]$ le score de *confiance* accordée à cette explication. Par exemple, soit le concept *Volume* de l'ontologie d'un agent a qui serait ambiguë pour les autres agents du système. Il pourrait exister deux explications de ce concept :

- $\langle\{Book, Sheets\}, \{a, b\}, b, 0.9\rangle$ signifiant que *Volume* est lié à la notion de livre, acceptée par les agents a et b et fournie par l'agent b. Cette explication a un degré de confiance de 0.9.
- $\langle\{Quantity, Amount\}, \{a, d, e\}, e, 0.5\rangle$ signifiant que *Volume* est lié à la notion de quantité, acceptée par les agents a, d et e et fournie par l'agent e. Cette explication a un degré de confiance de 0.5.

Un message de négociation sémantique est un message de la forme $\langle i, j, ia, p, T, C\rangle$ tel que i soit l'émetteur du message, j soit l'agent qui réceptionne le message, ia soit l'agent attendant l'explication[27], p soit le performatif du message, T le timeout de validité du message et C le contenu du message. Les performatifs utilisés sont classiques du protocole FIPA-request[28], bien que portant un autre nom (*SN_ QUERY, SN_ RESPONSE, SN_ INFORM, SN_ UNKNOW*). Le performatif correspond à la négociation sémantique est le performatif *SN_ FEEDBACK*, qui permet à un agent de demander une résolution sémantique d'un concept non compris. A noter qu'un agent peut ne pas connaître d'explication pour un concept mais être intéressé par l'explication une fois qu'un autre agent l'aura produite. Il peut alors se noter dans le message comme demandant l'explication (ce qui justifie l'utilisation du paramètre ia).

La création d'une explication est simplement l'utilisation des termes décrivant ce concept dans son ontologie. Par exemple, supposons qu'un agent a cherche à expliquer le concept *Book* décrit précédemment comme $(title, author, price)$, que les agents d et e soient intéressés par l'explication. L'agent a peut ainsi fournir une explication $\langle\{title, author, price\}, \{d, e\}, a, \_\rangle$ au concept *Book*. A noter que le score de confiance n'est pas calculé par l'agent a (car l'agent qui produit une explication ne doute pas de son explication), mais par l'agent récepteur qui évaluera ce score en fonction de la réputation de l'agent a.

Le modèle HISENE permet la prise en compte d'un facteur de *confiance* dans le choix d'un alignement (ou explication) entre des concepts de deux ontologies, notion que nous

[27] Le protocole HISENE permet aux agents de faire transiter un message d'explication. Ainsi, l'émetteur d'un message de négociation n'est pas forcément l'agent souhaitant obtenir l'information.

[28] `http://www.fipa.org/specs/fipa00026/`

utilisons spontanément lorsque quelqu'un nous explique un concept. Ce protocole ne propose pas particulièrement de solution pour le calcul de ces explications (ontologie sans structure, etc.), mais l'approche est facilement adaptable à des systèmes plus complexes d'hétérogénéité sémantique.

Les approches à base de négociation sémantique proposent de résoudre le problème d'hétérogénéité sémantique en ne passant que par un protocole d'interaction. Si communiquer pour s'expliquer des concepts est une méthode efficace, il apparaît qu'elle est surtout applicable lorsque les agents possèdent des ontologies très similaires ou faiblement structurées. Cette hypothèse sur la structure des ontologies est très forte et limite l'utilisation de ce type d'approche sur des agents très hétérogènes.

1.4.5 Système Multi-Agents et optimisation d'alignement

1.4.5.1 Laera & Tamma

Comme nous l'avons vu section 1.4.2.5, un alignement est fortement dépendant de la manière dont est construite une ontologie. Par exemple, son nombre d'instances limitera ou non l'utilisation des méthodes par instances. L'objectif des travaux de [Laera *et al.*, 2007] est de chercher à optimiser l'alignement des ontologies de deux agents en tenant compte du fait que l'agent sait comment est construite son ontologie. L'agent possède ainsi une liste de préférences sur les méthodes à utiliser pour aligner son ontologie avec un autre agent (*e.g.* un agent aura une préférence pour les alignements par instances si son ontologie en contient beaucoup). En tenant compte des préférences des deux agents ainsi que d'un ensemble de correspondances établi par un service extérieur d'alignement d'ontologies, il est ainsi possible de valider ou de refuser un certain nombre de propositions d'alignements.

Plus formellement, soit la définition classique d'un alignement entre deux concepts de deux ontologies différentes $\mathcal{O}$ et $\mathcal{O}'$ comme un tuple $m = \langle e, e', n, R \rangle$, tel que :

- $e \in \mathcal{O}$ et $e' \in \mathcal{O}'$ sont des entités (classe, relation ou instance) des deux ontologies dont il existe une correspondance,
- R est la relation de correspondance (*e.g.* équivalence, subsomption, etc.) entre e et e',
- $n \in [0, 1]$ est le degré de validité de cette correspondance.

Par exemple, dans le cas de deux ontologies décrivant des données bibliographiques, il est possible d'avoir un alignement du type $m = \langle auteur, auteur\_article, 0.65, \equiv \rangle$ signifiant que le concept *auteur* de la première ontologie est considéré équivalent au concept $auteur\_article$ de la deuxième ontologie avec un degré de certitude de 0.65. De même, les alignements $\langle personne, auteur, 0.7, \equiv \rangle$ et $\langle article, publication, 0.45, \equiv \rangle$ sont aussi des alignements possibles.

De plus, le service d'alignement a l'obligation de fournir une justification pour chacune des propositions d'alignement qu'il fait. Par exemple, pour justifier un alignement du type $m = \langle auteur, auteur\_article, 0.65, \equiv \rangle$, le service d'alignement pourra proposer une justification ayant la forme $a = \langle \langle superclass\,(auteur)\,, superclass\,(auteur\_article) \rangle\,, 0.67, \equiv, ES \rangle$ signifiant qu'il pense l'alignement m correct avec une probabilité de 0.67 car en utilisant la structure des ontologies (*i.e. ES=External Structure*) il est apparu que *auteur* est une superclasse de $auteur\_article$. Le service peut par ailleurs fournir des justifications négatives pour expliquer pourquoi il refuse certains alignements. Il est ainsi possible d'avoir dans le même

ensemble de justifications deux justifications contraires.

Nous avons à ce stade de la communication :

- deux agents ayant un ordre de préférence sur les méthodes d'alignements, ces préférences apparaissant dans les justifications ;
- une liste de candidats d'alignements associée à une liste de justifications.

En utilisant un système proche de la résolution de contraintes (basé sur les dépendances entre justifications et les préférences des agents), il est maintenant possible de trouver le sous-ensemble de justifications qui maximise les préférences de chaque agent. Les alignements qui sont alors validés sont ceux qui possèdent au moins une justification.

L'évaluation décrite dans l'article montre des résultats globalement meilleurs que les approches classiques d'alignement d'ontologies (*i.e.* alignement direct sans négociation entre agents). Néanmoins, dans le cas où deux agents possèdent des ensembles de préférences totalement différents, l'ensemble des alignements proposés est refusé (*i.e.* une justification est toujours impossible pour au moins un des deux agents), ce qui implique qu'il est dans ce cas impossible d'aligner les ontologies, alors que les outils classiques réussissent correctement. Cette évaluation permet ainsi de conclure que dans un SMA ouvert, la situation d'un alignement entre deux agents va varier entre deux extrêmes : soit l'alignement n'est pas optimal mais est complet et imparfait, soit l'alignement est optimal mais est incomplet (ces deux cas représentent d'ailleurs l'hypothèse de travail que nous avons considérée dans notre proposition pour la communication agent-agent présentée dans le chapitre 5).

D'autre part, l'un des pré-requis à l'utilisation de cette approche est que le service d'alignement d'ontologie *doit* fournir un ensemble de justifications, ce que finalement assez peu d'outils proposent actuellement.

1.4.5.2 Atencia & Schorlemmer

Les travaux de [Atencia et Schorlemmer, 2007] s'intéressent particulièrement à la dynamicité de la recherche d'un alignement. En particulier, à contre-pied de la plupart des travaux précédents, ils pensent qu'un alignement ne peut être valable qu'en fonction du contexte où il est construit et qu'il faut ainsi le remettre en question à chaque fois que le contexte change. Pour motiver cette pensée, ils prennent un exemple classique d'orientation droite/gauche. Prenons deux personnes dans une même pièce. Supposons de plus qu'elles partagent le même modèle du monde (*i.e.* la même ontologie) qui décrit l'orientation de manière usuelle gauche/droite. Supposons maintenant qu'elles soient de telle sorte qu'elles soient face à face. Alors, un objet situé à gauche de la première personne sera à droite de la deuxième. Supposons maintenant qu'elles soient côte à côte. Cette fois-ci, un objet à droite de la première personne sera aussi à droite de la seconde. Ce petit exemple permet d'apprécier que même avec la même ontologie, il n'est pas forcement évident d'éviter une ambiguïté. Ainsi, un alignement peut varier dynamiquement avec le temps.

Pour attaquer ce problème de dynamicité, les auteurs reprennent le formalisme de la théorie *des flux d'informations* de [Barwise et Seligman, 1997]. Nous ne donnerons ici qu'un aperçu du principe de cette théorie, car elle est complexe et la décrire entièrement n'apporterait par d'informations complémentaires pour comprendre le principe de l'approche de Atencia & Schorlemmer. Soit un ensemble d'états S représentant les états possibles du monde.

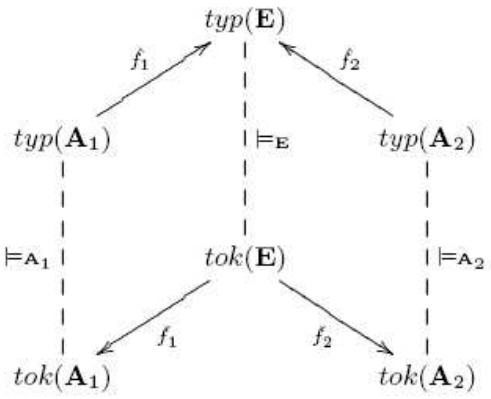

FIG. 1.25 – Modélisation selon la théorie de [Barwise et Seligman, 1997] du problème d'hétérogénéité sémantique. Chaque agent a sa propre perception du monde (fonction $\check{f}$), il la modélise selon son ontologie (fonction $\models$), cette représentation étant un modèle du monde réel (fonction $\hat{f}$)

Alors, chaque agent possède sa propre façon de modéliser l'état de ce monde. On nomme cette modélisation dépendante de l'agent une *classification*. On notera ainsi $tok(A_i)$ l'ensemble des perceptions du monde faites par l'agent A_i. Puis, l'agent A_i utilise une certaine syntaxe pour décrire cette modélisation, que l'on notera $typ(A_i)$. Le vocabulaire $typ(A_i)$ peut ainsi être assimilé à l'ontologie de l'agent A_i. La difficulté de l'hétérogénéité peut ainsi être modélisée selon la théorie des *flux d'informations* comme un problème pour rapprocher les modélisations *typ*, sachant qu'elles proviennent du même modèle de l'état du monde (figure 1.25).

Lorsque deux agents communiquent, il est donc nécessaire qu'ils se communiquent aussi leurs visions globales du monde, et non uniquement l'objet de la requête. Les deux agents savent qu'ils communiquent au sujet du même état du monde. En comparant les deux modélisations (au moyen d'un système d'inférences), les agents sont alors capable de distinguer quelles sont les termes hétérogènes qui désignent les même objets. Une fois ce travail d'identification effectué, les agents sont capables d'inter-opérer.

L'avantage de ce travail est de considérer la dynamicité de l'interaction et ainsi de poser la question ouverte de la validité en termes de temps d'un alignement construit entre deux agents. Néanmoins, ce travail repose sur l'hypothèse que les agents doivent s'envoyer la représentation complète du monde, afin que l'agent récepteur puisse déduire les alignements par inférence. Or, il semble difficilement concevable que les agents s'envoient systématiquement leurs représentations entières du monde. De plus, pour pouvoir utiliser une inférence, cela suppose qu'un certain nombre de termes du monde sont connus des deux agents. En effet, il n'y a d'inférence logique possible si les deux modélisations du monde ne partagent pas au moins une partie des prédicats.

1.4.6 Conclusion

Dans les approches à base d'ontologie de référence « publique » [Burstein *et al.*, 2003, Stuckenschmidt et Timm, 2002], un ancrage est ainsi effectué statiquement (et souvent semi-

automatiquement) avant le démarrage de l'agent entre les concepts de l'ontologie « privée » de chaque agent, et l'ontologie de référence. La résolution des problèmes de communication passe alors par la résolution combinée de l'ensemble des ancrages de chaque agent avec l'ontologie de référence. La faiblesse de ce type d'approche réside dans l'hypothèse de la présence systématique dans tout environnement d'au moins une ontologie de référence pré-ancrée sur l'ontologie privée de tous les agents.

Les approches à base de construction d'une ontologie globale [van Diggelen *et al.*, 2006a] ont l'avantage de très bien s'étendre aux problématiques du multi-agent, alors que les autres approches ne focalisent que sur une communication à deux agents. En effet, tout gain dans la construction d'une ontologie générale est directement utilisable par les autres agents du système, même s'ils n'ont pas encore cherché à communiquer. Néanmoins, cette approche pose des problèmes de mise à jour de cette ontologie, qui est ainsi de moins en moins dynamique et ouverte au fil du temps. De plus, dans un SMA ouvert, rien ne suppose que les agents réussissent à se mettre d'accord sur la définition de l'ensemble des concepts d'une ontologie générale. Enfin, tous ces agents doivent travailler sur la même tâche puisqu'ils partagent la même ontologie, or en pratique un SMA est constitué d'agents très différents, dont la création d'une ontologie commune n'aurait pas de sens.

D'autre part, dans le cadre de la négociation sémantique [van Diggelen *et al.*, 2006a, Morge et Routier, 2007, Garruzzo et Rosaci, 2006], proposer un protocole permet de tenir compte de la dynamicité de l'interaction et des capacités cognitives des agents. Les travaux actuels se sont surtout focalisés sur la négociation et l'obtention d'un accord sur la définition et les explications des concepts. Dans ces travaux, il est ainsi obligatoire que tout concept ait une définition par rapport à l'ontologie de l'autre agent. Or, les ontologies des agents sont différentes et le fait que leur intersection ne soit pas vide n'implique pas qu'elles définissent les mêmes concepts. Il est donc possible que certains concepts ne puissent pas être expliqués. Ce problème n'est pas évoqué et est renvoyé à la résolution d'un alignement d'ontologie lorsque la négociation n'est pas possible.

Finalement, le dernier type d'approche travaille directement au niveau de l'alignement entre les ontologies de deux agents donnés [Laera *et al.*, 2007, Atencia et Schorlemmer, 2007, Trojahn *et al.*, 2008]. Les algorithmes et protocoles proposés cherchent alors à construire l'alignement (ou à l'optimiser) en utilisant les capacités d'interactions et de raisonnements d'agents cognitifs complexes. Le modèle d'alignement demandé est alors particulier, afin de correspondre aux besoins des algorithmes mis en places. Par exemple, dans les travaux de [Laera *et al.*, 2007] le service d'appariement doit proposer des justifications aux propositions d'alignement. Autre exemple, dans les travaux de [Atencia et Schorlemmer, 2007], l'agent doit envoyer sa modélisation du monde en même temps que sa requête, afin que l'agent récepteur puisse inférer un alignement par comparaison des modèles du monde.

Dans la plupart des travaux sur l'interopérabilité d'agents, l'attention est particulièrement portée sur la gestion de l'alignement entre les ontologies des agents, sur la construction d'une ontologie commune ou sur des protocoles de communication pour désambiguïser des requêtes. Peu de travaux ont focalisé sur les problèmes pouvant survenir à l'étape suivante qu'est l'interprétation sémantique des requêtes. En effet, l'ensemble de ces travaux considèrent qu'être d'accord sur le sens des concepts suffit à pouvoir communiquer correctement, indépendamment du fait que certains concepts peuvent être mal compris ou non directement

traduisibles.

1.5 Bilan sur les systèmes d'interactions

Les systèmes d'interactions humain-agent et agent-agent représentent deux communautés de recherche bien distinctes. Ainsi, il n'y a que peu de travaux permettant de faire la jonction entre ces deux approches de l'interaction. Les quelques travaux récents à mi-chemin entre les deux thématiques sont représentés par les Agents Conversationnels Animés (ACA) [Cassel *et al.*, 2000] (*e.g.* l'agent SmartKom décrit dans la section 1.3.4.1 est un des meilleurs d'exemple d'ACA). Ces entités sont à la fois des agents, mais ils sont en plus conçus pour communiquer avec un utilisateur humain. Néanmoins, les ACA n'ont d'agent que le principe d'une réflexion cognitive. En pratique, ces modèles cognitifs ne sont utilisés que pour la communication avec un utilisateur ordinaire. Ainsi, il n'existe pas de travaux d'ACA directement dans le domaine des SMA.

Pourtant, lorsque l'on étudie les deux types d'interactions, certaines problématiques apparaissent dans les deux systèmes. Il est ainsi possible de généraliser un certain nombre de critères pour les systèmes d'interactions en général, basée sur l'étude des systèmes humain-agent d'une part et des systèmes agent-agent d'autre part :

- La capacité d'introspection. Nous avons vu que les systèmes d'interaction homme-machine s'orientent de plus en plus vers des systèmes capables d'introspection et d'analyse de leur propre code ou connaissances. Ceci permet de concevoir des systèmes qui sont plus réutilisables, car les algorithmes de traitement sont dépendant du formalisme de définition du code (ou des connaissances) et non de l'application en elle-même (Corese section 1.3.4.3 ou encore Cederic, section 1.3.4.4). Or, dans les Systèmes Multi-Agents, la capacité d'introspection et d'analyse est une des hypothèses de base de la construction même d'un agent [Ferber, 1995]. Ainsi, un agent doit être capable de manière autonome d'interpréter des messages, et donc de les analyser contextuellement à ses capacités. Une grande partie des approches de résolution du problème de l'hétérogénéité sémantique dans les SMA utilisent ainsi des protocoles de communications spécifiques de négociation sémantique (section 1.4.4). Finalement, il apparaît qu'un système d'interaction doit être pourvu de capacités d'introspection et d'analyse de ces capacités pour tendre vers la généricité.
- L'ontologie. De même que précédemment, si l'ontologie est devenue une hypothèse de base dans la communication agent-agent (au point que la plupart des articles considèrent la résolution du problème d'alignement comme la réponse au problème de l'hétérogénéité sémantique, section 1.4), elle n'apparaît que depuis peu dans la communication humain-agent. La problématique est ici légèrement différente néanmoins, car « l'ontologie d'un humain » n'est pas accessible par la machine. Ainsi, les systèmes d'interactions homme-machine doivent soit définir des ontologies plus complexe pour élargir le champ de compréhension de la machine (TRIPS, section 1.3.3.3), soit trouver une ontologie générale et la définir comme MRC de l'humain (WordNet par exemple dans les travaux de Paraiso & Barthès, section 1.3.3.1).
- L'approximation sémantique. Les distances sémantiques sont nettement plus utilisées dans le cadre de l'interaction homme-machine (*e.g.* Corese, section 1.3.4.3). Cela est dû à l'absence d'un modèle de connaissances pour « ontologie de l'humain », qui se traduit par des commandes à vocabulaire plus riche et donc un besoin sémantique plus

élevé. Néanmoins, cette problématique apparaît aussi dans la communication agent-agent, mais dans une moindre mesure. Dans ce cas, elle apparaît régulièrement comme un outil pour évaluer la distance sémantique entre deux ontologies entières, ou par exemple pour déterminer s'il est pertinent de lancer une communication entre deux agents donnés [Maedche et Staab, 2002, Euzenat et Shvaiko, 2007].

Notons pour finir sur l'approximation sémantique que tous les modèles de représentation des connaissances des systèmes d'interactions présentés (qu'ils soient humain-agent ou agent-agent) sont des hiérarchies de concepts. Or, dans le cadre de système d'interaction, utiliser une mesure exploitant les relations fonctionnelles entre les concepts peut améliorer l'interprétation sémantique [Gandon *et al.*, 2008]. Par exemple, « taper sur l'ordinateur » pourra se comprendre comme « taper sur le clavier de l'ordinateur ». Néanmoins, comme nous en avons discuté lors de la section 1.2, il n'existe pas de mesure de degré de relation sémantique efficace et validée actuellement qui puisse être intégrée dans un système d'interaction. C'est pourquoi nous proposerons dans le chapitre 3 une nouvelle mesure de degré de relation sémantique, afin de l'appliquer dans notre système d'interaction.

Dans le cadre de cette thèse, nous avons cherché à développer un système d'interaction pour un agent dont le problème d'hétérogénéité sémantique serait géré automatiquement par un protocole d'interaction adapté. Notre approche n'est pas une approche à base de négociation sémantique (malgré la présence d'un protocole), car nous ne cherchons pas à apprendre l'ontologie de l'autre agent. En revanche, nous cherchons à calculer dynamiquement la meilleure stratégie de réponse en fonction du degré de compréhension estimée de la requête. Le calcul de ce degré se fait dynamiquement en utilisant les capacités d'introspection de notre système, ainsi que des méthodes d'approximations sémantiques définies sur le MRC de l'agent. En fonction du résultat de ce calcul, l'agent choisit une réponse (et le performatif de cette réponse) la plus adaptée pour résoudre l'ambiguïté sémantique de la requête.

L'originalité de notre travail est que notre système est lié au principe d'interaction entre deux entités et n'est donc pas dépendant du type précis d'interaction (humain-agent ou agent-agent). En ce sens, nous avons développé des algorithmes et méthodes de conversions pour permettre à notre approche de fonctionner aussi bien en interaction humain-agent qu'en interaction agent-agent.

Dans le chapitre 2, nous présenterons plus précisément le système de stratégie de réponse que nous proposons. Nous commencerons par un descriptif des aspects du langage VDL. Le langage agent VDL, proposé par Nicolas Sabouret dans le cadre de sa thèse [Sabouret, 2002], est un langage agent introspectif qui permet une analyse du code de l'agent à l'état courant. Puis, nous continuerons par la définition des stratégies de réponses et de communication mises en place. Dans le chapitre 3, nous présenterons la mesure sémantique que nous utilisons dans notre système, qui est une mesure de degré de relation sémantique. Dans le chapitre 4, nous présenterons la version humain-agent de notre architecture, et nous garderons la version agent-agent pour le chapitre 5. Enfin, le chapitre 6 présentera succinctement le travail d'implémentation que nous avons effectué des théories que nous aurons présentées.

Chapitre 2

Gestion des réponses et interprétation sémantique

Sommaire

L'objectif final de cette thèse est de proposer un système de gestion de l'hétérogénéité sémantique applicable aussi bien pour les interactions humain-agent que agent-agent. Ce système repose un traitement sémantique des MRC des agents en jeux. Ainsi, une partie de l'information est extraite du MRC de l'agent récepteur et quantifiée à l'aide d'une mesure sémantique, tandis que les informations manquantes dans la requête (et qui ne peuvent être trouvées à l'aide d'une mesure) seront demandées à l'émetteur (que ce soit un agent ou un humain) pour clarification, vérification ou reformulation. Nous discuterons au cours de ce chapitre de l'importance d'utiliser une mesure sémantique puissante pour réduire la quantité d'informations ayant besoin d'être clarifiées. En particulier, les mesures classiques actuelles

ne permettent pas d'utiliser les relations fonctionnelles entre les concepts, alors qu'elles sont particulièrement importantes dans le cadre de système d'interaction.

Ce chapitre décrit le cœur algorithmique de ce système. La mesure sémantique que nous proposons pour considérer les relations fonctionnelles entre les concepts sera quant à elle présentée chapitre 3.

2.1 Architecture générale du système d'interaction

L'architecture de notre système est décomposée en trois parties :

1. Le module de traitement sémantique, qui est le cœur du système de traitement de l'hétérogénéité sémantique que nous proposons dans cette thèse (c.f. figure 2.1). Ce module est l'objet des sections suivantes de ce chapitre. Il traite de la stratégie de réponses, de sélection et de gestion de l'ambiguïté sémantique. Ces modules reposent sur une mesure sémantique pour calculer les distances entre les concepts mis en jeux. Les méthodes et algorithmes développés sont indépendants du type d'interaction (*i.e.* interaction humain-agent ou agent-agent). Ainsi, les entrées et sorties de ce module sont normalisées et il est nécessaire de faire appel à deux autres modules :
2. Le module de normalisation humain-agent (c.f. figure 2.2). Ce module a pour objectif de modéliser une commande en langue naturelle dans un formalisme analysable par l'agent, ainsi que de générer une réponse en langue naturelle à partir d'une réponse formelle de l'agent. Ce module fera l'objet du chapitre 4.
3. Le module de normalisation agent-agent (c.f. figure 2.3). Deux agents communiquant n'ayant pas la même ontologie, les requêtes envoyées ne peuvent être directement interprétées. Ce module a ainsi pour objectif de traduire une commande formelle exprimée dans une ontologie en une commande exprimée dans une autre ontologie, en perdant le moins de sémantique possible. Ce module fera l'objet du chapitre 5.

Ainsi, dans ce chapitre, nous présentons la première partie de l'architecture, qui est le cœur de notre système de traitement de l'hétérogénéité sémantique. Ce système repose sur le principe des systèmes *bottom-up génératif* (c.f. section 1.3). Le système de la littérature poussant le plus loin ce principe est le système Cederic [Eliasson, 2007]. Dans ces travaux, seul un ensemble minimal de compétences est fourni à l'avance (sous forme de plan). Si une commande de l'utilisateur ne s'apparie pas directement ou complètement à une des compétences existantes, le système tente de décrire un nouveau plan en se basant sur les éléments présents dans les plans associés.

Nous proposons d'aller encore plus loin, et d'adopter une approche *bottom-up constructive* basée sur l'analyse des préconditions. Toutes les capacités de l'agent sont alors calculées dynamiquement suivant un ensemble d'actions primaires réalisables par le système. Notre approche utilise les informations contextuelles (obtenues de l'agent à l'exécution) pour déterminer automatiquement quelles compétences peuvent être activées par l'agent dans l'état courant. Ce problème a été longuement étudié pour la validation de logiciel (*e.g.* [Botella *et al.*, 2002]) et montre des résultats intéressants pour la génération de cas de test. Ces travaux peuvent être adaptés à la génération de capacités si nous considérons qu'une

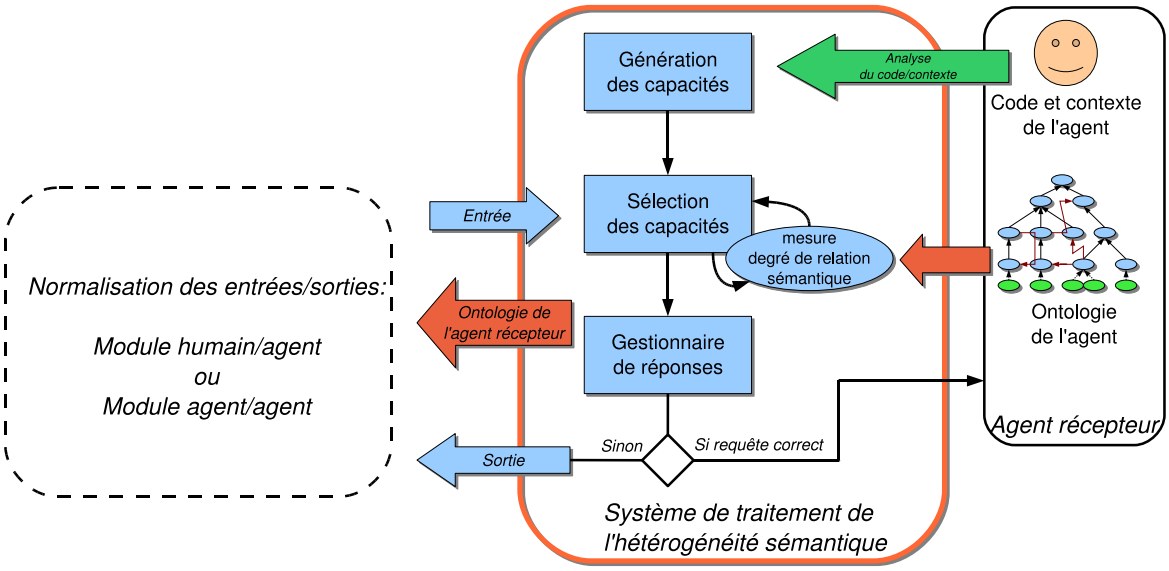

FIG. 2.1 – Architecture générale du système de traitement de l'hétérogénéité sémantique. La partie normalisation est dépendante du type d'interaction : humain-agent (figure 2.2) ou agent-agent (figure 2.3).

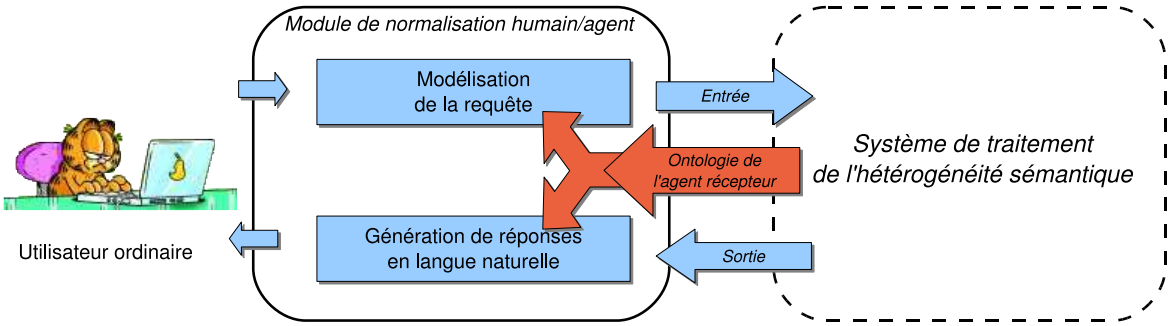

FIG. 2.2 – Module de normalisation humain-agent. Le système de traitement de l'hétérogénéité sémantique est présenté figure 2.1.

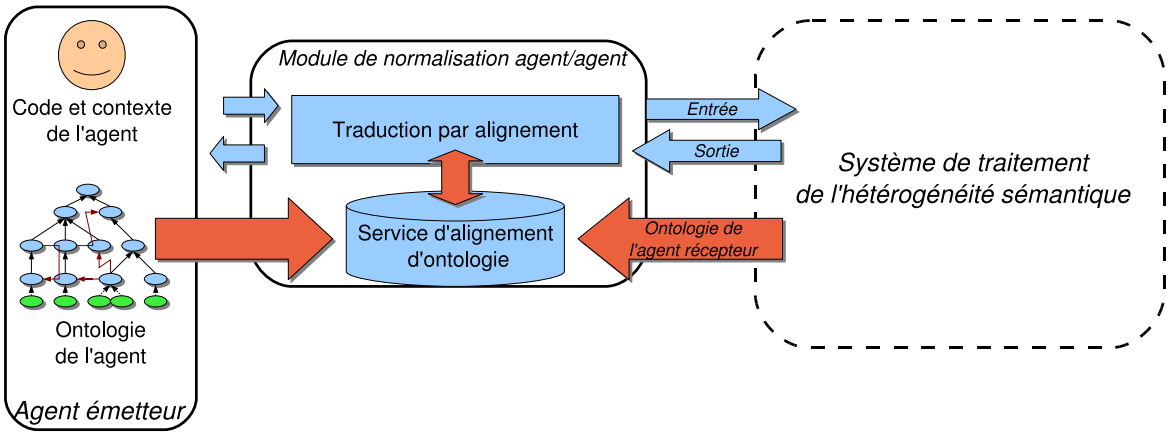

FIG. 2.3 – Module de normalisation agent-agent. Le système de traitement de l'hétérogénéité sémantique est présenté figure 2.1.

précondition a pour rôle de *tester* les messages. Ainsi, notre système construit d'abord la liste des capacités possibles, d'un point de vue agent.

La section 2.2 décrit l'algorithme de génération de commandes et donne un aperçu du modèle VDL d'agent que nous utilisons. La section 2.3 présente le système de réponses que nous utilisons, en détaillant d'abord le modèle normalisé de requête (utilisé par le module humain-agent et par le module agent-agent), le calcul de l'appariement entre une capacité et la requête (en utilisant l'ontologie de l'agent), puis la stratégie de sélection en elle-même. La section 2.4 illustre ce système avec un exemple simple de traitement de requête.

2.2 Génération des capacités d'un agent VDL

Cette première section donne les éléments importants sur le langage VDL, puis la section suivante décrit la génération de capacités en tant que telle.

2.2.1 Modèle d'actions en VDL

Nos agents sont programmés dans le langage de programmation VDL (*View Design Language,* [Sabouret et Sansonnet, 2003])[1]. Le modèle VDL est basé sur la réécriture d'arbre XML : la description de l'agent est un arbre XML dont les nœuds représentent soit des données, soit des actions. L'agent réécrit son arbre à chaque pas d'exécution en suivant certains nœuds dont la balise correspond à un mot-clef spécifique. Ce modèle permet aux agents d'accéder *pendant leur exécution* à la description de leurs actions afin de raisonner (par exemple pour planifier, analyser leur comportement, répondre à des questions sur leur fonctionnement, etc. [Sabouret et Sansonnet, 2001]).

Les agents possèdent deux types d'actions :

1. La pro-action : le comportement autonome d'un agent indépendant de toute interaction ;
2. La réaction : les actions liées à l'arrivée d'une commande, ou plus généralement d'interaction avec l'extérieur.

Dans le cadre de cette thèse sur l'interaction, nous nous limiterons aux réactions[2]. En VDL, les réactions sont activées par des *évènements,* c'est-à-dire des nœuds XML envoyés à l'agent en guise de commande. Ils sont la représentation formelle (*i.e.* en VDL) des requêtes. Les réactions décrivent comment ces messages doivent être traités. L'objectif de notre système est ainsi de construire des évènements VDL à partir d'une requête reçue (envoyée par un utilisateur ou un autre agent).

En VDL, comme dans la plupart des modèles de représentation des actions, nous représentons une action par un tuple $r = \langle nom, P, E\rangle$ où nom est le nom de l'action, P l'ensemble des préconditions de l'action et E son ensemble d'effets. Nous pouvons définir quatre types de préconditions pour une réaction $r \in R$, R étant l'ensemble des réactions possibles :

- $\mathcal{P}_e(r)$ est l'ensemble des préconditions d'évènement. Elles sont utilisées pour aiguiller une certaine forme d'évènement vers une certaine classe de réactions, ou pour rejeter les évènements globalement mal formés. Leur interprétation dans le cadre des arbres

[1] `http://www-poleia.lip6.fr/~sabouret/demos`

[2] Nous utiliserons ainsi indifféremment dans cette thèse les termes *action* et *réaction.*

```
<action>
  <name>Prendre un objet</name>
  <event><take><object/></take></event>
  <guard><not>
    <equals><get><battery/></get><value>0</value></equals>
  </not></guard>
  <guard><exists><evt-get><object/></evt-get></exists></guard>
  <guard><near><evt-get><object/></evt-get></near></guard>
  <put>
    <path><robot-hand/></path>
    <evt-get><object/></evt-get>
  </put>
</action>
```

FIG. 2.4 – Exemple de code d'action en VDL. Les différents ensembles de préconditions sont calculés en fonction de la présence de certains mots clefs dans la précondition.

XML utilise la subsomption pour savoir si l'évènement peut être correct pour au moins une réaction. Par exemple, tous les évènements de type « temperature », seront traités par l'action associée spécifiquement aux effets sur la température.

- $\mathcal{P}_s(r)$ est l'ensemble des préconditions de structure. Elles sont utilisées pour vérifier la syntaxe précise d'un message et assurer que la réaction possède toutes les informations nécessaires pour s'exécuter sans erreurs. Les préconditions de $\mathcal{P}_s(r)$ ne dépendent donc pas de l'état courant de l'agent, mais uniquement de la structure détaillée de l'évènement. Par exemple, un message désignant une réservation d'hôtel doit toujours comporter un numéro de réservation.
- $\mathcal{P}_c(r)$ est l'ensemble des préconditions de contexte. Ces préconditions ne dépendent que du contexte courant de l'agent (*e.g.* un robot ne pourra pas prendre un objet si sa batterie est vide, figure 2.4).
- $\mathcal{P}_{cs}(r)$ est l'ensemble des préconditions contextuelles-structurelles, *i.e.* préconditions dépendante de l'évènement *et* du contexte courant de l'agent. Pour exemple, un robot ne peut pas attraper un objet, spécifié dans l'environnement de l'agent *et* en paramètre dans le message, si celui-ci n'est pas à portée.

Nous noterons $\mathcal{P}_e = \cup_{r\in\mathcal{R}}\mathcal{P}_e(r)$. Pour tout $e \in \mathcal{P}_e$, nous noterons $R_e(e) = \{r \in \mathcal{R} | e \in \mathcal{P}_e(r)\}$ l'ensemble des réactions qui peuvent traiter l'évènement e (*i.e.* dont l'exécution donnera lieu à des effets). Nous noterons également $\mathcal{P}(r) = \mathcal{P}_e(r) \cup \mathcal{P}_s(r) \cup \mathcal{P}_c(r) \cup \mathcal{P}_{cs}(r)$ l'ensemble de toutes les préconditions associées à la réaction r.

D'autre part, dans le modèle VDL, l'agent possède un certain nombre de concepts personnels (*i.e.* qui ne sont pas accessible par d'autres agents) que l'on notera des *concepts VDL*. Ces concepts sont présents dans le code XML de l'agent soit comme étiquette (*i.e.* *tag*), comme attributs ou contenu texte. Nous noterons $\mathcal{C}_{VDL}$ l'ensemble de tous les concepts VDL. Ainsi, un évènement VDL est constitué d'un ensemble (ordonné sous forme d'arbre XML) de concepts VDL. Une précondition utilise les mots-clefs du langage et ces concepts pour tester l'évènement.

De plus, chaque agent est associé à un MRC. Par construction, ce MRC doit définir au

moins un concept équivalent pour chaque concept VDL. En notant $\mathcal{C}_{MRC}$ l'ensemble des concepts de l'ontologie, il existe donc une fonction injective, que nous noterons map_{VDL} : $\mathcal{C}_{VDL} \rightarrow \mathcal{C}_{MRC}$, de l'ensemble des concepts VDL $\mathcal{C}_{VDL}$ de l'agent vers l'ensemble des concepts $\mathcal{C}_{MRC}$ définis dans le MRC. Ainsi, chaque concept VDL n'est relié qu'à un seul concept du MRC. Pour simplifier la définition des algorithmes, nous ne ferons pas la différence entre l'utilisation d'un concept VDL et l'utilisation d'un concept du MRC, ceci évitant un appel systématique à la fonction map_{VDL} à chaque utilisation d'un concept VDL.

A ce stade de la présentation de notre travail, il n'est pas nécessaire de pousser plus loin la formalisation du MRC, et nous n'utiliserons donc que l'ensemble $\mathcal{C}_{MRC}$ sans préciser la structure sous-jacente. Une formalisation plus précise sera faite en préambule des algorithmes sémantiques de traitement sur l'ontologie, chapitre 3.

2.2.2 Génération des capacités possibles : l'ensemble $\mathcal{E}$

La propriété d'introspection des agents VDL permet de générer automatiquement leurs listes de *capacités* à l'instant courant [Mazuel et Sabouret, 2008a]. Une *capacité* est un paramétrage d'une action de l'agent, permettant de réaliser un effet sur son environnement ou une modification de son état interne.[3] Prenons l'exemple d'un robot dans un entrepôt possédant les actions classiques $deplacer(x, y)$, $prendre(o)$ et $poser(o)$ (nous prenons le cas où une position dans l'entrepôt est représentée par une case de coordonnées cartésiennes (x, y)). Alors, ce robot possédera autant de capacités qu'il existe de cases accessibles pour lui par l'action $deplacer(x, y)$. De plus, si ce robot est positionné prêt d'une caisse, il aura toujours les capacités de déplacement mais aussi une capacité $prendre(caisse)$ correspondant à l'action *prendre*. Ainsi, selon notre définition, pour une action donnée il existe plusieurs capacités. Notre objectif est alors d'être capable de générer automatiquement les capacités d'un agent, en analysant son code pendant l'exécution.

L'idée de la génération des capacités est d'utiliser les préconditions structurelles ($\mathcal{P}_s$ et $\mathcal{P}_{cs}$) comme un ensemble de contraintes sur les évènements pour compléter les squelettes d'évènements fournis par les préconditions d'évènement $\mathcal{P}_e$. Nous noterons $\mathcal{P}(E)$ l'ensemble des sous-ensembles de E et Υ l'ensemble des arbres VDL possibles. Pour tout $e \in \mathcal{P}_e$, nous noterons $refine(e, r) \in \mathcal{P}(\Upsilon)$ les évènements (c'est-à-dire les capacités) obtenus à partir du squelette e et de la réaction $r \in R_e(e)$ (en n'analysant donc que $\mathcal{P}_s$ et $\mathcal{P}_{cs}$) par l'utilisation de notre algorithme de génération des cas de test. L'algorithme complet pour la méthode $refine$ est trop long pour être présenté ici et nuirait à la compréhension du principe général. Il repose fortement sur la sémantique opérationnelle du modèle VDL. Il est basé sur une interprétation récursive des termes VDL associée à un certain nombre de règles pour chaque mot-clé du langage. Un exemple succinct et simple est présenté sur la figure 2.5 pour en illustrer le mécanisme. L'algorithme complet et détaillé est présenté dans l'annexe A page 195.

Soit $eval : \Upsilon^2 \rightarrow \{\top, \bot\}$ la fonction d'évaluation booléenne d'une précondition structurelle (*i.e.* utilisant la structure d'un évènement) telle que : $\forall p \in \mathcal{P}_s(r) \cup \mathcal{P}_{cs}(r), \forall e \in \mathcal{P}(\Upsilon), eval(p, e) = \top$ ssi la précondition p est valide par rapport à l'évènement e et à l'état courant de l'agent. On pose par analogie la fonction $eval : \Upsilon \rightarrow \{\top, \bot\}$ qui calcule la valid-

[3] En VDL, chaque capacité correspond à un évènement XML différent (mais une action traite généralement toute une catégorie d'évènements).

Une partie du code de l'agent

```
<a1>
  <b>7</b>
  <c>8</c>
</a1>
<a2>
  <b>9</b>
  <c>10</c>
</a2>
```

Deux préconditions de la réaction r :

```
<event><take/></event>
<ctx-str>
  <exist>
    <get>
      <evt-get><take/></evt-get>
    </get>
    <b/>
  </exist>
</ctx-str >
```

Deux évènements générés par ces préconditions ($refine(e, r)$) :

```
<take><a1/></take>
<take><a2/></take>
```

FIG. 2.5 – Un exemple de génération d'évènements. La production de $refine(e, r)$ utilise les préconditions d'évènements (introduites par le tag *event*) et les préconditions de contexte-structure (introduites ici par le tag $ctx - str$)

ité d'une précondition contextuelle (*i.e.* indépendante de la structure d'un évènement) telle que : $\forall p \in \mathcal{P}_c(r), eval(p) = \top$ ssi la précondition p est valide par rapport à l'état courant de l'agent. Nous définissons alors l'ensemble $\mathcal{E}$ des capacités possibles de la façon suivante :

$$\mathcal{E} = \left\{ refine(e, r), \forall e \in \mathcal{P}_e, \forall r \in R_e(e) \quad | \quad \begin{array}{l} \forall p \in \mathcal{P}_s(r) \cup \mathcal{P}_{cs}(r), eval(p, refine(e, r)) = \top, \\ \forall p \in \mathcal{P}_c(r), eval(p) = \top \end{array} \right\}$$

2.2.3 Génération des capacités actuellement impossibles : l'ensemble $\mathcal{F}$

Par défaut, une approche générative ne produit que des évènements qui valident intégralement les préconditions (évènements possibles). Néanmoins, l'utilisateur fait parfois des requêtes impossibles contextuellement mais qui pourraient l'être dans un autre état, et donc qui doivent être comprises. Par exemple, si un utilisateur demande l'adresse d'un magasin de chaussures dans sa ville, le fait que cette ville ne contienne pas de magasin de chaussures n'empêche pas la compréhension de la requête (cela empêche juste de répondre à la requête).

Une évaluation préliminaire de notre système basée uniquement sur l'algorithme génératif de l'ensemble $\mathcal{E}$ nous a montré l'importance de la prise en compte de ces évènements impossibles [Mazuel et Sabouret, 2008a].[4] Nous proposons donc dans cette section une solution pour générer un ensemble $\mathcal{F}$ de *capacités actuellement impossible.*

Pour qu'un évènement soit *actuellement impossible* et donc possible dans un autre état de l'agent, il faut que le contexte de cet agent à l'état courant ne permette pas l'activation de la réaction associée à cet évènement. Autrement dit, soit r la réaction associée à l'évènement e, alors cet évènement est actuellement impossible si les préconditions de r utilisant le contexte

[4] Dans cette évaluation (que nous expliquerons plus en détail section 4.3 page 147), nous avons demandé à plusieurs utilisateurs humains d'évaluer les réponses d'un agent conversationnel. Or, nous avons montré que l'absence de réponses aux requêtes impossibles pénalise l'avis général de l'utilisateur sur l'agent.

sont fausses. Il existe deux type de préconditions de contextes : les ensembles $\mathcal{P}_c$ et $\mathcal{P}_{cs}$. Finalement, pour générer des évènements actuellement impossibles nous utilisons une méthode de construction similaire à l'ensemble $\mathcal{E}$, mais nous filtrons les évènements par rapport à la faisabilité de leur contexte.

L'ensemble $\mathcal{F}$ des capacités actuellement impossibles est alors défini par :

$$\mathcal{F} = \left\{ refine(e,r), \forall e \in \mathcal{P}_e, \forall r \in R_e(e) \quad | \quad \begin{array}{l} \forall p \in \mathcal{P}_s(r), eval(p, refine(e,r)) = \top, \\ \forall p \in \mathcal{P}_{cs}(r), eval(p, refine(e,r)) = \bot, \\ \forall p \in \mathcal{P}_c(r), eval(p) = \bot \end{array} \right\}$$

De plus, pour tout $e \in \mathcal{F}$, nous notons $np(e)$ l'ensemble des préconditions fausses empêchant l'évènement d'être possible. Comme $\mathcal{F}$ est l'ensemble des évènements impossibles, $\forall e \in \mathcal{F}$, $np(e) \neq \emptyset$. L'ensemble est défini de la façon suivante :

$$np(e) = \bigcup_{r \in R_e(e)} \{\{p \in \mathcal{P}_s(r) \cup \mathcal{P}_{cs}(r) | eval(p,e) = \bot\} \cup \{p \in \mathcal{P}_c(r) | eval(p) = \bot\}\}$$

L'algorithme de calcul de réponses présenté dans la section suivante repose sur l'hypothèse qu'un agent est capable de produire la liste de ses capacités à l'instant courant. Néanmoins, l'approche que nous allons décrire n'en devient pas pour autant dépendante du langage agent VDL. Elle est dépendante de la présence d'une liste de capacités pour l'agent à l'instant courant. Ainsi, le langage agent utilisé doit posséder un système de gestion de ces capacités (même si la liste des capacités est définie de manière statique, comme cela se fait dans de nombreuses approches bottom-up de la littérature [Paraiso *et al.*, 2004]).

2.3 Calcul de la stratégie de réponse

Le gestionnaire de réponses (GR) est responsable des dialogues avec l'utilisateur pour l'acceptation d'une commande, mais aussi pour la gestion des commandes incomplètes ou imprécises. Les paramètres d'entrées de notre GR sont les ensembles d'évènements $\mathcal{E}$ et F créés par le module de génération de la section précédente.

2.3.1 Formalisation des arbres évènements et requête

Comme nous l'illustrons sur la figure 2.1, les requêtes reçues d'un utilisateur humain ou d'autres agents sont modélisées de manière unique, afin que les algorithmes de sélection soient les mêmes dans le cadre de l'interaction humain-agent et dans le cadre de l'interaction agent-agent.

Chaque requête est modélisée sous forme d'un arbre de concepts de l'ontologie. La structure en arbre est intéressante car elle permet d'être compatible avec la structure en arbre XML d'une capacité, et simplifie ainsi le calcul de l'appariement requête/capacité. Les requêtes sont modélisées par des arbres simples, sans les notions XML d'attributs ou de contenu texte (*i.e.* les concepts des arbres requêtes sont toujours situés dans les tags des nœuds de

l'arbre).

Tout comme les capacités, les arbres des requêtes sont construits en fonction des concepts présents dans l'ontologie de l'agent. Ainsi, un arbre requête est en fait un ensemble de concepts de l'ontologie ordonné sous forme d'arbre. Il est important de noter que l'arbre de la requête n'est pas dépendant d'une structure en arbre qui serait présente dans l'ontologie. En effet, l'arbre de la requête ne représente pas une relation de subsomption mais une relation de dépendance. Par exemple, le concept *rouge* peut être un fils dans l'arbre de requête du concept *Ferrari,* sans que cela implique une relation d'héritage entre ces deux concepts.

Comme toute opération de modélisation, il est possible de perdre de l'information durant son processus. Or, l'estimation de cette perte est importante pour estimer la possibilité d'une erreur d'interprétation. Les modules de normalisation fournissent ainsi une *fonction de qualité de modélisation* $Q(R)$ qui rend une valeur dans l'intervalle $[0, 1]$ pour estimer la qualité de la modélisation (*i.e.* estimer la quantité d'information perdue par le processus de modélisation). Pour un score de 1, la modélisation sera considérée sans aucune perte et inversement pour un score de 0. Le calcul de la fonction $Q(R)$ est dépendant du module de normalisation choisi et sera donc donné dans le chapitre 4 pour l'interaction humain-agent et dans le chapitre 5 pour la modélisation agent-agent.

2.3.2 Calcul du score d'appariement

Le score d'appariement entre une requête et une capacité nous fournit un indicateur sur le fait que la requête doit être associée ou non à cette capacité. Pour ce faire, le GR (gestionnaire de réponse) calcule le score d'appariement entre chacun des évènements de $\mathcal{E}$ et $\mathcal{F}$ et la requête reçue. Plus ce score est élevé, plus l'alignement est fort et donc plus la capacité peut être considérée comme une bonne représentation en VDL de la requête (et inversement). Pour calculer ce score, nous devons tenir compte :

1. Des approximations introduites pendant la modélisation de la requête. Nous utiliserons pour cela le résultat de la fonction $Q(R)$ décrite à la fonction précédente.
2. Des « distances sémantiques » entre un concept de la capacité et un concept du modèle de la requête. Pour cela nous utiliserons une mesure sémantique (section 1.2) définie sur l'ontologie de l'agent.

D'autre part, pour le calcul des distances sémantiques, toutes les mesures que nous avons définies dans la section 1.2 peuvent être utilisées. Nous noterons ainsi $sem_{ONT}(x, y)$ la similarité (ou le degré de relation) sémantique choisie pour mesurer le rapprochement sémantique entre deux concepts x et y de l'ontologie.

Nous pensons que l'utilisation d'une mesure de *similarité* sémantique n'est pas suffisante pour appréhender le cas des interactions. En effet, dans l'évaluation préliminaire que nous avons effectuée de l'interaction humain-agent (évaluation que nous présenterons section 4.3), il n'y avait qu'une mesure sémantique simple et les résultats ont montré des manques d'interprétation sémantique qui ne pouvaient être résolus que par les mesures de degré de relation sémantique. C'est pourquoi, dans cette thèse, nous avons défini une nouvelle mesure sémantique, calculant le degré de relation entre les concepts (présentée dans le chapitre 3).

Il est important de noter que la conception des algorithmes présentée dans ce chapitre n'est pas dépendante du choix d'une mesure en particulier, mais que ce sont les résultats et

les performances du système qui seront affectés par ce choix.

L'objectif ici est de calculer le score d'appariement entre l'arbre représentant la requête de l'utilisateur (que nous noterons A_{req}) et un arbre représentant une capacité du système (que nous noterons A_{cap}). Chaque concept VDL de $\mathcal{C}_{VDL}$ étant associé à un concept du MRC de $\mathcal{C}_{MRC}$ par la fonction map_{VDL} (c.f. section 2.2.1), une capacité A_{cap} peut ainsi être traduite en un arbre de concepts de $\mathcal{C}_{MRC}$. De plus, une requête A_{req} est directement modélisée par un arbre de concepts de $\mathcal{C}_{MRC}$. Ainsi, calculer l'appariement entre A_{req} et A_{cap} peut aussi être vu comme le calcul d'un appariement entre deux nuages de concepts du MRC. Nous présenterons d'abord le cas simple de l'appariement entre deux nuages de concepts (en considérant que A_{req} et A_{cap} ne sont *que* deux nuages de concepts du MRC), puis nous présenterons comment avec une modification mineure nous pouvons tenir compte de leur structure arborescente pour améliorer le résultat.

Appariement entre nuages de concepts

Plusieurs méthodes ont été proposées pour calculer un score entre nuages de concepts, en fonction des critères à mettre en avant dans le contenu des deux ensembles de concepts (c.f. section 1.2.5 page 47). Nous avons choisi de définir notre fonction de score comme non symétrique, dans le sens où l'ensemble des concepts de la requête est jugé plus important que l'ensemble des concepts de la capacité. En effet, si tous les concepts de la requête sont dans une capacité, alors cette capacité représente correctement cette requête. L'inverse n'est pas vrai, car si une requête est plus large qu'une capacité donnée, considérer la requête comme proche de la capacité serait supposer que les informations en trop dans la requête sont inutiles.

D'autre part, nous évaluons toutes les permutations possibles d'appariement entre A_{req} et A_{cap}. Nous noterons $\Omega(A_{req}, A_{cap})$ la fonction qui renvoie l'ensemble des permutations des couples d'appariement entre A_{req} et A_{cap} (sans autoriser l'utilisation double d'un concept). Par exemple, si A_{cap} est une capacité contenant les concepts $\{take, red, object\}$ et A_{req} une requête contenant les concepts $\{take, form\}$, alors $\Omega(\{take, red, object\}, \{take, form\})$ renvoie l'ensemble des 6 permutations :

1. $\{(take : take), (red : form)\}$,
2. $\{(red : take), (object : form)\}$,
3. $\{(take : form), (red : take)\}$,
4. etc.

Grâce à cette fonction, nous pouvons calculer le score de chaque permutation de couples de concepts. Une fois le score de chacune des permutations calculé, nous choisissons le maximum des scores obtenus comme résultat. Pour une permutation donnée, nous calculons la moyenne des scores obtenus par notre mesure pour chaque couple de la permutation, puis nous pondérons ce résultat par le score $Q(A_{req})$ donné par le système de modélisation des requêtes. Ainsi nous pouvons définir la fonction *app* ainsi :

$$app(A_{req}, A_{cap}) = Q(A_{req}) \times \frac{\max\limits_{S \in \Omega(A_{req}, A_{cap})} \left(\sum\limits_{x:y \in S} sem_{ONT}(x, y) \right)}{|A_{req}|}$$

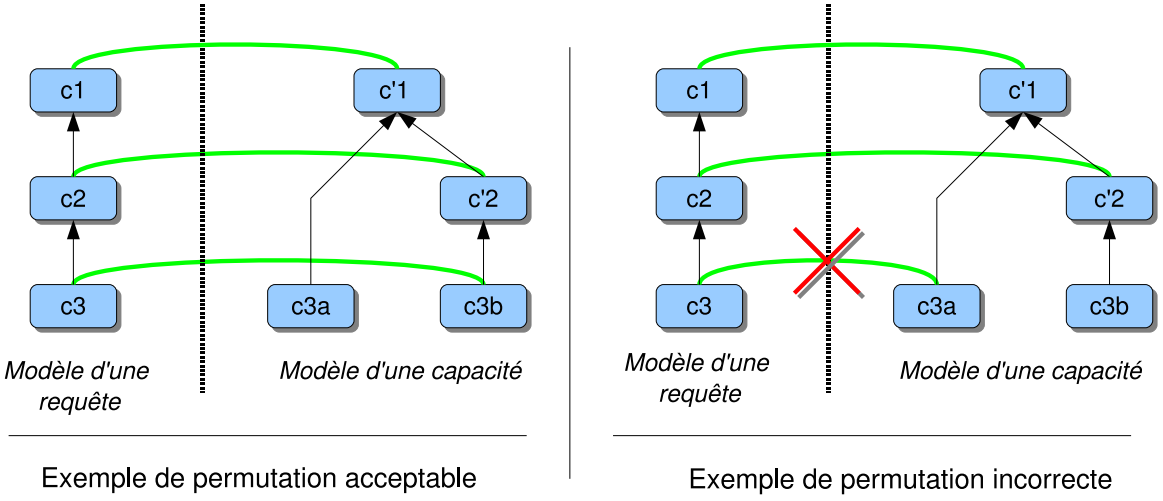

FIG. 2.6 – Filtrage des permutations de la fonction Ω en fonction de la position des concepts dans l'arbre. Dans le premier exemple, la permutation est acceptable car tout couple est en relation de dépendance avec un autre couple. Dans le deuxième exemple, le couple $(c_3 : c_3a)$ n'est pas dépendant du couple $(c_2 : c'_2)$, donc la permutation est incorrecte.

Utilisation des arbres de dépendances

Pour tenir compte de la structure en arbre de la requête et de la capacité, nous considérons un filtrage dans l'énumération des permutations $\Omega(A_{req}, A_{cap})$. Autrement dit, la structure en arbre permet de supprimer certaines permutations. L'idée est de considérer les relations de subsomption entre couples de concepts pour une permutation donnée (c.f. figure 2.6). Ainsi, les couples de concepts doivent être dépendants deux à deux pour que la permutation soit acceptable. Plus formellement, soit $P \in \Omega(A_{req}, A_{cap})$ et S_1, S_2,...,S_n les couples de la permutation P et $(x_i : y_i)$ les concepts dans le couple S_i. Soit la relation $\prec$ entre deux concepts d'un modèle, tel que $x \prec y$ implique que x est plus spécifique que (*i.e.* est subsumé par) le concept y dans l'arbre du modèle. Alors, une permutation est jugée acceptable si $\forall\,(i,j) \in [1,n]^2, i \neq j$, alors $x_i \prec x_j \Leftrightarrow y_i \prec y_j$.

2.3.3 Sélection des capacités candidates

Soit $\mathcal{E}$ l'ensemble des capacités d'un agent à l'instant courant (capacités générées dynamiquement ou fournies statiquement, section 2.2.2) et A_{req} un arbre représentant une requête. Alors, il est possible de construire le sous-ensemble $\mathcal{E}(A_{req})$ de $\mathcal{E}$ tel que $\mathcal{E}(A_{req})$ contienne les capacités maximisant le score $app(A_{req}, e)$, avec $e \in \mathcal{E}$. En effet, s'il existe un sous-ensemble de $\mathcal{E}$ avec un score $app(A_{req}, e)$ très élevé, alors les capacités de ce sous-ensemble représentent l'interprétation la plus probable de la requête A_{req}. À l'inverse, si les scores sont faibles, alors il est peu probable que la requête représente l'une des capacités courantes de notre agent. Plus formellement, nous définissons $\mathcal{E}(A_{req})$ tel que :[5]

[5]En pratique, nous n'utilisons pas directement le max, mais nous avons un seuil de tolérance statique permettant d'accepter les évènements étant à une distance réduite du max. Ce seuil est actuellement empiriquement à la valeur $0,05$. Notre objectif à long terme est de pouvoir calculer dynamiquement ce seuil en fonction de la répartition des scores dans l'ensemble $\mathcal{E}$.

$$\mathcal{E}(A_{req}) = \left\{ e \in \mathcal{E} | app(A_{req}, e) = \max_{e' \in \mathcal{E}} app(A_{req}, e') \right\}$$

D'autre part, le score d'appariement entre la requête A_{req} et chacun des éléments de $\mathcal{E}(A_{req})$ est identique pour tous les couples. Ainsi, nous noterons $p_{\mathcal{E}}$ le score unique d'appariement du sous-ensemble $\mathcal{E}(A_{req})$:

$$p_{\mathcal{E}} = \max_{e' \in \mathcal{E}} app(A_{req}, e')$$

Réciproquement, nous noterons $\mathcal{F}(A_{req})$ l'ensemble des capacités actuellement impossibles de score maximum et $p_{\mathcal{F}}$ ce score. En d'autres mots, le module de gestion des réponses ne prend en considération que les meilleurs évènements comparativement au modèle de la requête parmi ceux générés dans les ensembles $\mathcal{E}$ et $\mathcal{F}$.

2.3.4 Stratégie du gestionnaire des réponses et performatifs

La décision du choix du performatif à utiliser en réponse à une requête d'un agent est dépendante de deux seuils définis dans l'intervalle $[0, 1]$. Ces seuils s'approchent dans leur utilisation des seuils « tell me » et « do it » de Patty Maes [Maes, 1994]. Le seuil p_{min} (*tell me*) est la valeur minimale pour qu'une requête puisse être considérée *probablement comprise* comme une capacité (*i.e.* avec un doute). Le seuil p_{max} (*do it*) est la limite par delà de laquelle la requête est considérée comme étant *comprise avec certitude,* tel que la requête est une représentation fidèle de la meilleure capacité générée[6].

La réponse donnée par notre système dépend de la position de $p_{\mathcal{E}}$ et $p_{\mathcal{F}}$ par rapport à p_{min} et p_{max}. Nous différencions 5 stratégies de réponses différentes pour l'agent récepteur du message. Chaque stratégie est représentée par un *performatif* différent (c.f. section 1.4.1). Le performatif d'un message permet en effet d'identifier le rôle sémantique d'un message (*e.g.* requête, confirmation, échec, etc.). Ainsi, à chaque performatif est associé une sémantique et une structure de message différente. Le choix d'un performatif en particulier permet ainsi à l'agent ayant reçu la requête d'exprimer la manière dont il a compris cette requête. La stratégie de choix que nous adoptons est la suivante :[7]

1. Si $p_{\mathcal{E}} \geq p_{max}$ et $p_{\mathcal{F}} \leq p_{\mathcal{E}}$ et $|\mathcal{E}(A_{req})| = 1$, la requête est considérée comme correctement comprise par l'agent. De manière naturelle, ce cas ne posant pas de difficulté particulière, nous utiliserons le performatif classique de FIPA *agree*.
2. Si $p_{min} < p_{\mathcal{E}}$ et $p_{\mathcal{E}} < p_{\mathcal{F}}$, l'agent cible croit que la requête qu'il a reçu n'est pas possible (*i.e.* $p_{\mathcal{E}} < p_{\mathcal{F}}$). Il envoie donc à l'agent initial la liste $\mathcal{E}(A_{req})$ des évènements possibles les plus proches de la requête reçue. Pour cela, nous introduisons le performatif *suggest* indiquant : 1) que le message initial n'est pas exécutable et 2) que le contenu du message retour est un ensemble de capacités possibles jugées proches du message initial.
3. Si $p_{\mathcal{F}} \leq p_{\mathcal{E}}$ et que :

[6] Nous adoptons actuellement les valeurs (calculées empiriquement) $p_{min} = 0.3$ et $p_{max} = 0.85$.

[7] Nous avons réutilisé les performatifs qui existaient déjà dans la liste des performatifs FIPA 00037 `http://www.fipa.org/specs/fipa00037` lorsque leurs sémantiques correspondaient à la sémantique que nous souhaitions pour nos propres performatifs (voir section 1.4.1.2)

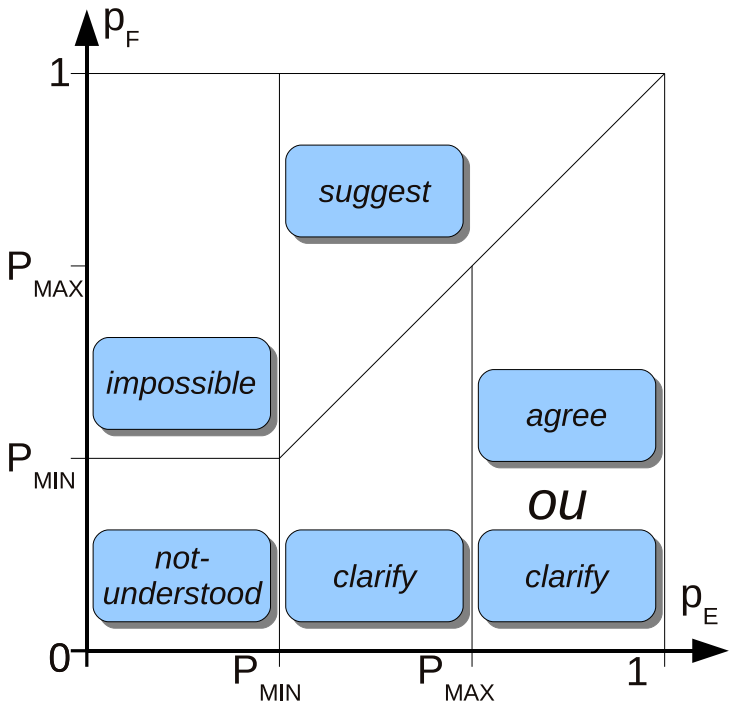

FIG. 2.7 – Représentation schématique du choix du performatif en fonction des valeurs de $p_{\mathcal{E}}$, $p_{\mathcal{F}}$, p_{min} et p_{max}. Le *ou* est un « ou exclusif », le choix étant décidé par la cardinalité de l'ensemble $\mathcal{E}(S)$.

(a) soit $p_{max} \leq p_{\mathcal{E}}$, mais $|\mathcal{E}(A_{req})| > 1$

(b) soit $p_{min} < p_{\mathcal{E}} < p_{max}$

alors, les capacités impossibles peuvent être ignorées, mais l'agent n'est pas sûr d'avoir correctement compris la requête ($p_{\mathcal{E}} < p_{max}$) ou alors il y a trop de requêtes candidates ($|\mathcal{E}(A_{req})| > 1$). Autrement dit, l'agent récepteur possède une liste de capacités possibles candidates, mais ne peut pas procéder à une exécution. L'agent fait donc une demande de clarification à l'agent A en lui notant l'ensemble des capacités possibles correspondant le plus à la requête reçue (*i.e.* l'agent envoie l'ensemble $\mathcal{E}(A_{req})$ à l'agent émetteur). Pour cela, nous introduisons le performatif *clarify*.

4. Si $p_{\mathcal{E}} \leq p_{min}$ et $p_{min} \leq p_{\mathcal{F}}$, l'agent récepteur pense avoir compris la commande reçue, mais elle est impossible. L'agent doit donc informer l'agent initial que sa commande a bien été comprise, mais n'est pas applicable actuellement. Nous introduisons, pour notifier cette situation, le performatif *impossible*.

5. Si $p_{\mathcal{E}} \leq p_{min}$ et $p_{\mathcal{F}} \leq p_{min}$, l'agent récepteur n'a pas été capable d'interpréter correctement la requête de l'agent initial. Nous reprenons alors performatif FIPA *not-understood*.

La figure 2.7 illustre graphiquement la répartition des différentes stratégies dans l'espace $[0,1]^2$ en fonction des valeurs du couple $(p_{\mathcal{E}}, p_{\mathcal{F}})$.

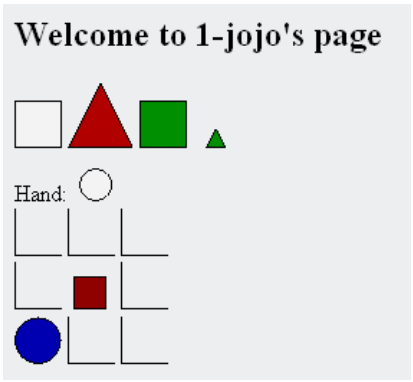

FIG. 2.8 – Un état initial pour l'agent Jojo

2.4 Illustration du principe sur un exemple

Cette section présente un exemple complet de traitement de sélection de capacité dans notre système. Pour ce faire, nous allons présenter un agent simple, puis nous allons modéliser le traitement d'une requête.

2.4.1 L'agent "jojo"

L'agent Jojo[8] est un agent simple inspiré du monde de cube de Winograd [Winograd, 1972]. Cet agent a deux actions possibles : prendre un objet et poser un objet. Le monde dans lequel est défini l'agent contient 7 objets, ainsi qu'une grille de taille 3×3 dans laquelle l'agent peut poser un objet qu'il a pris. Un objet est caractérisé par sa forme ($shape \in \{square, triangle, circle\}$), sa couleur ($color \in \{red, green, blue, white\}$) et sa taille ($size \in \{tiny, small, medium, big\}$). Une position est un emplacement dans la grille, noté par un couple défini dans $\{upper, center, lower\} \times \{right, middle, left\}$. Selon ces notations, une capacité formelle VDL peut être par exemple :

```
<take><object>
  <shape>square</shape>
  <color>red</color>
  <size>medium</size>
</object></take>
```

qui représenterait la demande « prend l'objet dont la forme est un carré, dont la couleur est rouge et dont la taille est moyenne ».

2.4.2 Contexte courant et exemple de requête

Supposons que l'agent Jojo soit dans l'état donné sur la figure 2.8. L'agent a un cercle blanc dans sa main. Il y a deux objets posés dans la grille (un cercle bleu dans la case en bas à gauche et un carré rouge au centre). Supposons que l'agent ait reçu la requête A_{req} suivante :

[8] Le code VDL complet de l'agent est donné dans l'Annexe B page 211.

```
<drop>
  <sphere/>
  <line>
    <lower/>
  </line>
</drop>
```

qui représente la demande « pose le cercle sur la ligne du bas ». Nous allons étudier comment notre système propose de traiter cette requête.

2.4.3 Génération des capacités

Premièrement, le système calcule l'ensemble $\mathcal{E}$ (les capacités possibles) et l'ensemble $\mathcal{F}$ (les capacités actuellement impossibles). Dans le contexte courant de notre exemple, seul les capacités associées à l'action « poser » sont possibles, car l'agent tient déjà un objet dans sa main et ne peut donc pas prendre quelque chose. De plus, deux cases de la grille sont déjà occupées, donc les capacités associées à l'action « poser » dans ces cases sont actuellement impossibles. Finalement, l'ensemble $\mathcal{E}$ contient 8 éléments :

1. Sept capacités correspondant aux sept cases libres dans la grille.
2. Une capacité pour poser l'objet à l'extérieur de la grille.

De même, l'ensemble $\mathcal{F}$ contient 9 capacités impossibles :

1. Deux capacités correspondant aux deux cases occupées de la grille
2. Sept capacités correspondant à l'action de prendre l'un des sept objets.

À titre d'exemple, une capacité générée de l'action « prendre un objet » possède la forme suivante :

```
<take><object>
  <shape>square</shape>
  <size>medium</size>
  <color>white</color>
</object></take>
```

D'autre part, une capacité générée de l'action « poser dans une case » est de la forme :

```
<drop><position>
  <line name=''upper''/>
  <row name=''left''/>
</position></drop>
```

2.4.4 Calcul des appariements maximaux

Il est maintenant possible de calculer les ensembles $\mathcal{E}(A_{req})$ et $\mathcal{F}(A_{req})$ qui maximisent l'appariement entre les capacités générées et la requête A_{req} présentée précédemment. Prenons la capacité A_{cap} suivante pour présenter un exemple de calcul de sélection :

```
<drop><position>
  <line name="lower"/>
  <row name="center"/>
</position></drop>
```

La fonction Ω fournit ici $\frac{7!}{(7-4)!} = 840$ permutations (*i.e.* nombre d'arrangements sans répétition sachant qu'il y a 4 concepts dans A_{req} et 7 concepts dans A_{cap}). Plusieurs de ces couples ne seront pas évalués grâce aux contraintes imposées par la structure en arbre. Par exemple, les permutations contenant les couples $(lower_{req}, lower_{cap})$ et $(line_{req}, row_{cap})$ seront invalidés, sans qu'il soit nécessaire de calculer les similarités sémantiques entre ces concepts. Finalement, la permutation permettant d'obtenir le meilleur score est la permutation $\{(drop_{req}, drop_{cap}), (line_{req}, line_{cap}), (lower_{req}, lower_{cap}), (sphere_{req}, position_{cap})\}$. Notons que si les trois premiers couples ont dans cet exemple une similarité sémantique parfaite, le concept $sphere_{req}$ ne possédait pas de concept sémantiquement proche dans la capacité finale. Le choix du concept $position_{cap}$ (qui n'est pas particulièrement proche du concept $sphere_{req}$) est justifié par le fait que dans les mesures classiques un concept quelconque est souvent plus proche d'un concept abstrait quelconque que d'un concept spécifique quelconque (le chemin hiérarchique entre deux concepts donnés passant nécessairement par un concept plus abstrait). En supposant ici que $sem_{ONT}(sphere, position) = 0.2$, alors le calcul est :

$$p_{\mathcal{E}} = \frac{1 + 1 + 1 + 0.2}{4} = 0,8$$

Finalement, une fois l'ensemble des calculs effectués, les seules capacités de $\mathcal{E}$ composant l'ensemble final $\mathcal{E}(A_{req})$ sont :

```
<drop><position>
  <line name="lower"/>
  <row name="center"/>
</position></drop>
```

```
<drop><position>
  <line name="lower"/>
  <row name="right"/>
</position></drop>
```

car ce sont les deux seules associées à la ligne du bas dans l'environnement courant (correspondant aux actions de « poser en bas au milieu » et de « poser en bas à droite »).

De manière similaire, il n'y a qu'un évènement impossible qui maximise le score d'appariement $p_{\mathcal{F}}$ dans l'ensemble $\mathcal{F}(A_{req})$ (correspondant à la case du bas gauche, déjà occupé) :

```
<drop><position>
  <line name="lower"/>
  <row name="left"/>
</position></drop>
```

Nous avons donc de même que $p_{\mathcal{F}} = 0,8$.

2.4.5 Réponse du système

Des calculs précédents nous obtenons donc $p_{\mathcal{E}} = 0,8$, $p_{\mathcal{F}} = 0,8$, $|\mathcal{E}(A_{req})| = 2$ et $|\mathcal{F}(A_{req})| = 1$. En utilisant les seuils par défaut $p_{min} = 0.3$ et $p_{max} = 0.85$, nous arrivons dans la situation 3 correspondant au performatif *clarify*. Cette stratégie ignore les évènements impossibles, et l'ensemble $\mathcal{F}$ n'est donc pas pris en compte. L'agent envoie donc un message

avec le performatif *clarify* contenant la liste des 2 capacités possibles de l'ensemble $\mathcal{E}$. Sémantiquement, cela signifie qu'il existe une ambiguïté pour répondre à la requête et que cette ambiguïté a été réduite à un choix entre deux capacités. Du point de vue de la génération en langue naturelle, nous verrons chapitre 4 que cela peut se traduire par une réponse du type :

> Your command is imprecise, please give information.
> I can either :
> - drop position line name is lower row is center
> - drop position line name is lower row is right

Notons que l'algorithme a complètement ignoré la case en bas à gauche, car elle était impossible *à l'instant courant*. Cette case pourrait être proposée plus tard dans l'exécution de l'agent si le contexte change.

2.5 Conclusion

Nous avons défini dans ce chapitre un algorithme génératif fondé sur l'introspection du code de l'agent à l'exécution. Il permet d'analyser les préconditions des actions de l'agent afin d'en déduire la liste des évènements possibles et impossibles à l'état courant. Cette méthode a l'avantage d'être générique, puisqu'elle s'adapte à n'importe quel agent écrit en VDL.

Ce système repose sur un calcul d'appariement entre une requête de l'utilisateur et une capacité du système. Ce calcul repose lui-même sur l'utilisation d'une mesure sémantique sur l'ontologie de l'agent. Nous pensons que l'utilisation d'une mesure de *similarité* sémantique n'est pas suffisante pour appréhender le cas des interactions. En effet, dans l'évaluation préliminaire que nous avons effectuée de l'interaction humain-agent (évaluation que nous présenterons section 4.3), il n'y avait qu'une mesure sémantique simple et les résultats ont montré des manques d'interprétation sémantique qui ne pouvaient être résolus que par les mesures de degré de relation sémantique. Ainsi, nous avons développé dans le cadre de cette thèse une mesure de ce type, afin de l'utiliser pour calculer l'appariement requête/capacité. La description de cette formule fait l'objet du prochain chapitre.

La modélisation de la requête est dépendante du type d'interaction initiale (humain-agent ou agent-agent), mais elle repose sur un formalisme commun de représentation des requêtes. Ainsi, nos algorithmes de sélection ne sont dépendants que de ce formalisme. Le chapitre 4 présente la modélisation humain-agent alors que le chapitre 5 présente la modélisation agent-agent.

Il n'est pas possible de faire directement une évaluation de la partie du système de traitement de l'hétérogénéité sémantique présentée dans ce chapitre. En effet, il est nécessaire pour cela d'avoir un système complet, soit en interaction humain-agent, soit en interaction agent-agent. C'est pourquoi l'évaluation du système humain-agent sera présentée au chapitre 4 section 4.3 alors que l'évaluation du système agent-agent sera présentée au chapitre 5 section 5.4.

Chapitre 3

Mesure de degré de relation sémantique dans une taxonomie augmentée

Sommaire

Si la similarité sémantique considère la quantité d'information partagée entre les attributs des concepts, le degré de relation sémantique permet de prendre aussi en compte les relations fonctionnelles entre ces concepts. L'une des conséquences directes est que dans les modèles à base de graphes, la similarité n'exploite que le squelette hiérarchique. Par exemple, les concepts « voiture » et « essence » ont un faible degré de similarité, mais un fort degré de relation, ce qui n'est pas modélisable avec une simple taxonomie [Resnik, 1995]. Cependant, le calcul automatique d'un degré de relation sémantique est considéré comme étant plus difficile que le calcul d'une similarité sémantique. Pour cette raison, la plupart des travaux de la littérature (que nous avons présentés en section 1.2) reposent sur des hiérarchies de subsomptions (tel que MeSH [Rada *et al.*, 1989] ou WordNet [Budanitsky et Hirst, 2006]). Les travaux récents sur le degré de relation sémantique ont tenté d'adopter un autre point de vue en se basant sur l'utilisation de définitions (*i.e. gloss-based approach* [Patwardhan et Pedersen, 2006,

Strube et Ponzetto, 2006]) selon l'idée qu'une définition contient l'ensemble des termes reliés à un concept. Une autre méthode est d'utiliser un moteur de recherche sur le web, avec l'idée que les co-occurrences de termes ne sont pas spécifiques des types de liens (hiérarchiques ou relationnels). Les mesures ainsi définies modélisent alors un degré de relation sémantique [Cilibrasi et Vitanyi, 2006, Iosif, 2007].

Cependant, il n'y a pas de travaux, depuis ceux de [Hirst et St-Onge, 1998], qui ont repris le problème du degré de relation sémantique sur un modèle de représentation des connaissances basé sur les graphes (tel que les réseaux sémantiques ou les ontologies réduites à la hiérarchie et aux propriétés des objets), qui est le modèle de représentation des connaissances le plus utilisé pour les systèmes d'interprétation (qu'ils soient humain-agent ou agent-agent). C'est pourquoi, nous pensons qu'il est intéressant de calculer un degré de relation sémantique sur un modèle à base de graphe. De même, dans les travaux sur l'alignement d'ontologie, il peut être intéressant d'être capable de calculer avant la recherche d'un alignement le degré de rapprochement sémantique entre ces ontologies, pour évaluer à quel point l'alignement peut être utile ou non [Euzenat et Shvaiko, 2007, Maedche et Staab, 2002]. En pratique, dans tous ces travaux, seule une hiérarchie de concepts est considérée, alors que la littérature avance un besoin pour l'interprétation sémantique qui ne peut être résolu par une hiérarchie [Eliasson, 2007, Gandon *et al.*, 2008, Hau *et al.*, 2005]. Par exemple, dans une de nos premières évaluations n'utilisant qu'un modèle hiérarchique simple [Mazuel et Sabouret, 2008a], certains concepts ne pouvaient être reliés alors que les utilisateurs les considéraient comme équivalents (*e.g.* « *form* » et « *object* »).

Quelques travaux récents ont proposé des mesures de degré de relation sémantique sur des langages de haut niveau basés sur la logique, tel que OWL-Lite [Smith *et al.*, 2004]. Par exemple, dans [Hau *et al.*, 2005], les auteurs tiennent compte des relations non-hiérarchiques mais aussi des intersections, disjonctions de classes, etc. Cependant, ces approches sont restées à l'état de théorie et n'ont jamais été évaluées ni même implémentées dans un système réel. Nous pensons de plus qu'avant d'arriver à des étapes aussi complexes, définir (et valider) une mesure valable sur les ontologies limitées aux propriétés objets est déjà une première étape difficile et présentant des questions non résolues.

Dans ce chapitre, nous proposons notre propre mesure de degré de relation sémantique sur une hiérarchie de concepts augmentée de relations hétérogènes. Notre objectif est de respecter les propriétés suivantes :

- De considérer un ensemble de « patrons de chemins sémantiquement corrects » comme guide dans le graphe (tel que nous l'avons discuté en section 1.2.3.4.1 page 36)
- D'utiliser les théories utilisables sur les mesures de similarité pour les étendre au degré de relation (tel que la théorie de l'information).
- De pouvoir calculer le poids sémantique d'un chemin non-hiérarchique.

Nous présenterons d'abord le modèle de représentation des connaissances que nous considérons comme base pour définir notre mesure. Nous présenterons ensuite notre mesure puis une évaluation de cette mesure. Enfin, nous terminerons par une étude de la complexité des algorithmes permettant de la calculer.

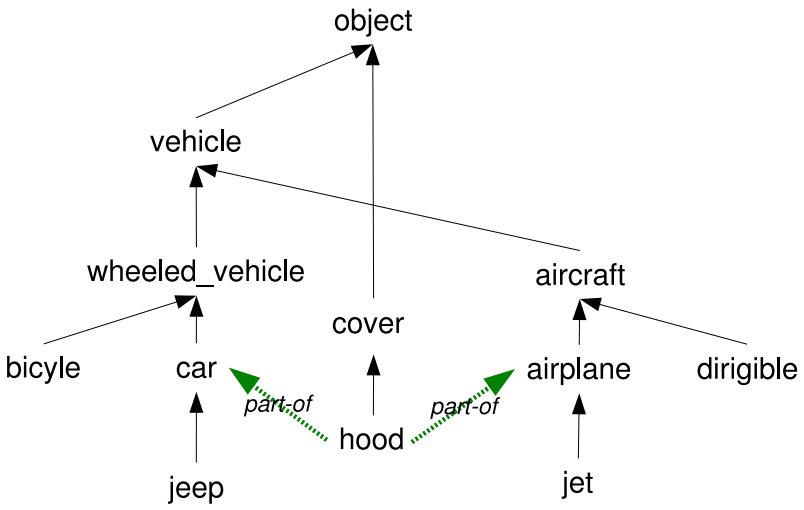

FIG. 3.1 – Exemple de hiérarchie avec relations (tiré de WordNet 3.0). Les flèches noires représentent les relations de subsomption *is-a*.

3.1 Modèle des connaissances

Nous utiliserons comme modèle de représentation des connaissances une taxonomie de concepts (telle que nous l'avions évoquée et utilisée dans les formules sémantiques présentées dans le chapitre 1), mais augmentée de relations non-hiérarchiques pouvant apparaître entre n'importe quelles paires de concepts (telle que la relation *part-of* par exemple, mais aussi d'autres types de relations non contraints comme $madeWith$, $inverseOf$, etc.).

Plus formellement, notons $\mathcal{P}(E)$ l'ensemble des sous-ensembles de l'ensemble E. Alors, nous pouvons définir une *taxonomie augmentée* $\mathcal{TA}$ comme un tuple $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$ tel que :

- C est l'ensemble de concepts de la taxonomie $\mathcal{TA}$ (représentant les nœuds dans une taxonomie de concepts).
- R est une énumération des types de relations non-hiérarchiques existant dans cette taxonomie $\mathcal{TA}$.
- $L \in \mathcal{P}\left(C \times (R \cup \{is\text{-}a, includes\}) \times C\right)$ est l'ensemble des arêtes du graphe (*i.e.* les relations entre paires de concepts). Par exemple, en OWL, les relations hiérarchiques $\{is\text{-}a, includes\}$ sont modélisées par la relation *owl :subClassOf* et les relations non-hiérarchiques par des sous-classes de *owl :ObjectProperty*.
- $\mathcal{F} : C \rightarrow \mathbb{R}^+$ est une fonction associant un poids à un concept de la taxonomie $\mathcal{TA}$ (par exemple la fonction IC des travaux sur la théorie de l'information, section 1.2.3.3.1) ;
- $\mathcal{G} : R \rightarrow \mathbb{R}^+$est une fonction associant un poids à un type de relation présent dans la taxonomie $\mathcal{TA}$.

Par exemple, sur la taxonomie $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$ de la figure 3.1, nous avons $car \in C$, $part\text{-}of \in R$, $\langle hood, is\text{-}a, cover \rangle \in L$ et $\langle hood, part\text{-}of, car \rangle \in L$. De plus, selon la définition de Seco (section 1.2.3.3.1), la quantité d'information d'un concept feuille est de 1.0, ce qui peut se traduire dans l'exemple par $\mathcal{F}(jeep) = IC(jeep) = 1.0$. Dans la suite de ce chapitre, nous conservons l'idée que le poids d'un concept représente sa quantité d'information IC et

nous utiliserons donc uniquement cette fonction comme pondération des concepts.

Nous aurons parfois besoin de séparer explicitement l'ensemble des arêtes hiérarchiques des arêtes non-hiérarchiques. Nous définissons donc une partition de l'ensemble L de la façon suivante :

- $L_H = \left\{\langle x, h, y\rangle \in L, (x, y) \in C^2, h \in \{is\text{-}a, includes\}\right\}$: l'ensemble des arêtes hiérarchiques ;
- $L_R = \left\{\langle x, r, y\rangle \in L, (x, y) \in C^2, r \in R\right\}$: l'ensemble des arêtes non-hiérarchiques.

Pour illustrer ces définitions sur les exemples précédents, nous avons donc $\langle hood, is\text{-}a, cover\rangle \in L_H$ et $\langle hood, part\text{-}of, car\rangle \in L_R$.

Dans un modèle de représentation des connaissances sous forme de graphe, les relations existent habituellement avec leurs inverses. Par exemple, si la relation *is-a* a comme inverse la relation *includes*, la relation *part-of* a la relation *has-part*, etc. Dans les modèles associés aux mesures sémantiques actuelles, la distinction n'est pas faite entre ces deux relations inverses. Ainsi, dans le cas de mesures de similarités, un lien hiérarchique possédera le même poids qu'il soit traversé dans le sens *is-a* ou dans le sens *includes*.

Dans le cadre de taxonomie augmentée de relations non-hiérarchiques, nous avons décidé de conserver cette simplification du problème et ainsi de considérer deux hypothèses :

1. L'inverse d'une relation existe systématiquement et sera noté r^{-1} :

$$\forall r \ \in R, \exists r^{-1} \in R | \forall (x, y) \in C^2, \langle x, r, y\rangle \in L \Rightarrow \left\langle y, r^{-1}, x\right\rangle \in L$$

2. Le poids d'une relation inverse $r^{-1} \in R$ est le même que celui de la relation $r \in R$:

$$\forall r \in R,\ \mathcal{G}(r) = \mathcal{G}(r^{-1})$$

Nous considérerons donc dans la suite de cette section que le graphe associé à la taxonomie augmentée est un *graphe orienté* de telle sorte que les arcs non-hiérarchiques soient valués, mais que pour chaque relation entre deux concepts il existe une relation inverse de poids équivalent. De même, pour simplifier l'écriture, nous ne représenterons qu'un seul lien entre deux concepts et non les deux liens inverses (sur la figure 3.1 nous n'avons ainsi fait apparaître que les relations *part-of* et non les relations *has-part*, alors qu'elles existent). Du point de vue du vocabulaire pour les algorithmes que nous présenterons, nous utiliserons donc celui des graphes orientés (*arc*, *chemin, circuit,* etc.).

Ces deux hypothèses impliquent une symétrie complète entre deux concepts pour une relation donnée. Ainsi, la mesure que nous définirons est une mesure symétrique :$\forall (c_1, c_2) \in C^2$, $dist(c_1, c_2) = dist(c_2, c_1)$.[1]

3.2 Description de notre mesure de degré de relation

Note préliminaire : La mesure que nous présentons d'abord dans cette section est analogue à une distance (*i.e.* une mesure de « non-relation » sémantique) : plus le score est faible,

[1] Cette hypothèse est couramment admise, même si il existent quelques mesures non symétrique dans la littérature [Ventresque *et al.*, 2008]. Néanmoins, ces mesures ne sont pas évaluées sur ce critère car il n'existe pas de tests proposant de le prendre en compte.

plus les concepts sont reliés. Néanmoins, nous terminerons cette section en proposant une conversion de cette formule de distance en formule de degré de relation (plus le score est faible, moins les concepts sont reliés) en nous basant sur les méthodes proposées par [Resnik, 1995, Jiang et Conrath, 1997].

Notre travail reprend l'idée développée dans le cadre des hiérarchies avec la théorie de l'information (suivant laquelle deux arêtes hiérarchiques ne représentent pas la même quantité d'information) mais en l'étendant aux arêtes non-hiérarchiques. Nous présenterons d'abord comment nous calculons le poids d'une arête donnée ou plus généralement d'un *chemin de type unique*. Nous expliquerons ensuite comment calculer le poids d'un *chemin de type multiple*, avant de finalement présenter notre mesure de degré de relation sémantique.

3.2.1 Chemin de type unique

Nous appelons *chemin de type unique* (*i.e.* CTU) un chemin tel que toutes les arêtes de ce chemin soient du même type. Par exemple, sur la figure 3.1, le chemin $\{jeep \xrightarrow{is\text{-}a} car \xrightarrow{is\text{-}a} wheeled\_vehicle \xrightarrow{is\text{-}a} vehicle\}$ est un chemin de type unique, constitué uniquement du type *is-a*. Pour calculer le poids W d'un CTU $path_X(x, y)$ de type X, nous différencions deux cas : soit le type du chemin est hiérarchique (X représente la relation *is-a* ou *includes*) soit il ne l'est pas. Les deux sous-sections suivantes décrivent le calcul du poids en fonction de ces deux situations.

Notons que même si deux relations r et r^{-1} sont symétriques, la définition d'un CTU ne permet pas qu'elles soient présentes en même temps dans le même chemin. Par exemple, le chemin $\{jeep \xrightarrow{is\text{-}a} car \xrightarrow{is\text{-}a} wheeled\_vehicle \xrightarrow{includes} bicycle\}$ n'est pas un CTU mais une concaténation de deux CTU : les chemins $\{jeep \xrightarrow{is\text{-}a} car \xrightarrow{is\text{-}a} wheeled\_vehicle\}$ et $\{wheeled\_vehicle \xrightarrow{includes} bicycle\}$.

3.2.1.1 Chemin hiérarchique

La valeur sémantique d'un chemin hiérarchique est un des points les plus étudiés dans la littérature sur les mesures sémantiques. Nous avons donc décidé de reprendre l'une des formules de la littérature. Comme nous en discutions en section 1.2.4.5 page 47, les meilleurs mesures actuellement sont les mesures de Lin et de Jiang & Conrath. Nous avons choisi la formule de Jiang & Conrath, car elle est basée sur le calcul d'un chemin dans un graphe, puis sur le calcul de la valeur sémantique de ce chemin (à l'inverse de la mesure de Lin qui n'utilise pas la structure en graphe). Or, nous pensons que pour tenir compte des liens hétérogènes entre les concepts il faut choisir une formule basée sur les chemins pouvant emprunter ces liens.

Par rapport à la présentation que nous avons faite de la mesure de Jiang & Conrath section 1.2.3.3.4, nous pouvons synthétiser cette mesure de la façon suivante :

$$dist_{JC}(c_1, c_2) = \sum_{\{x,y\} \in sp(c_1,c_2)} LS(x, y) \times T(x, y)$$

où $sp(c_1, c_2)$ représente le plus court chemin hiérarchique entre c_1 et c_2, $LS(x, y)$ représente le poids associé à l'arête $\{x, y\}$ et $T(x, y)$ le poids associé au *type* de l'arête $\{x, y\}$. En

pratique, cette mesure n'a été définie que hiérarchiquement, en supprimant le poids d'arête (*i.e.* $\forall (x,y) \in C^2, \langle x, is\text{-}a, y\rangle \in L \Rightarrow T(x,y) = 1$). D'autre part, le poids d'une arête est défini en fonction de la différence entre les poids des deux concepts :

$$LS(c_1, c_2) = |IC(c_1) - IC(c_2)|$$

Nous réutiliserons donc cette définition qui correspond à nos hypothèses de travail. Ainsi, si X est une relation hiérarchique, nous poserons que le poids d'un CTU $path_X(x,y)$ entre deux concepts x et y est :

$$W(path_{X \in \{is\text{-}a, includes\}}(x,y)) = |IC(x) - IC(y)|$$

3.2.1.2 Chemin relationnel

Si le type de relation X du CTU n'est pas hiérarchique, nous ne pouvons pas utiliser la quantité d'information IC propre à chaque nœud, car cette formule est calculée suivant la structure hiérarchique de la taxonomie et ne tient pas compte des relations existant entre les concepts. Cette section présente ainsi une nouvelle proposition pour calculer le poids d'un CTU non-hiérarchique. Cette formule est basée sur les idées suivantes :

1. Chaque type de relation X est associé à un poids T_X représentant le « coût sémantique » de cette relation. L'idée de ce poids est de considérer que certaines relations apportent moins d'informations et donc leur coût sémantique doit être plus grand (la mesure que nous cherchons à définir étant une distance, une relation qui n'apporte pas d'information doit augmenter cette distance et donc avoir un grand coût, et inversement). Du point de vue pratique, ce coût sémantique peut être vu comme le coût maximal d'un CTU de longueur infinie. Ainsi, pour toute relation X il existe une valeur limite L, telle que pour un poids $T_X < L$, suivre ce type de relation apporte de l'information, pour un poids $T_X = L$, l'information est la même que suivre un lien hiérarchique et pour $T_X > L$ suivre ce type de relation alourdit un chemin sémantique.[2] Ce système de poids permet de différencier les différents types de relations d'une taxonomie augmentée. Par exemple, dans la plupart des systèmes la relation de méronymie aura un poids $T_{hasPart}$ inférieur à 1 et la relation d'antonymie aura, au contraire, un poids $T_{antonymy}$ très supérieur à 1.
2. La valeur des coûts sémantique d'un chemin étant donné son poids T_X et sa longueur $|path_X(c_1, c_2)|$ doit respecter les conditions suivantes :
   (a) Elle doit croître avec la longueur du chemin ;
   (b) Elle doit être bornée de manière à ce qu'un chemin de longueur infinie corresponde bien au coût T_X ;
   (c) Elle doit être construite de telle manière que la différence de coût sémantique en deux longueurs successives (*i.e.* entre un chemin de longueur l et un chemin de longueur $l+1$) diminue au fur et à mesure que la longueur augmente. Autrement

[2] La valeur précise de cette limite L est dépendante de la manière dont sont calculés les poids hiérarchiques, autrement dit de la fonction IC.

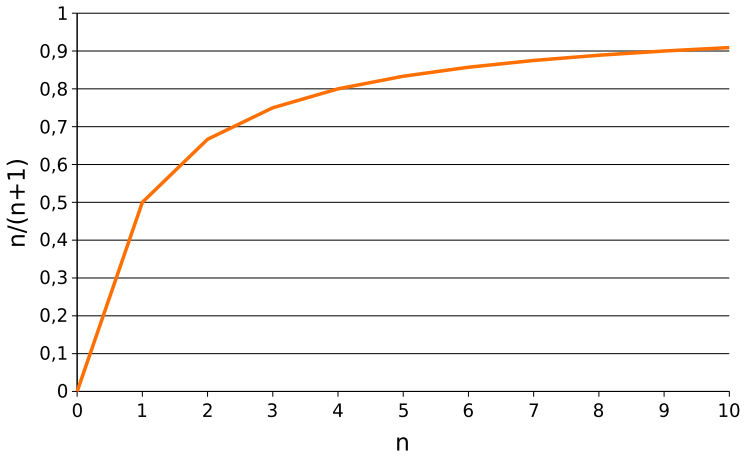

FIG. 3.2 – Évolution de la fonction $\frac{n}{(n+1)}$. L'avantage de cette fonction dans notre situation est que sa limite en l'infini est finie et que sa forme présente une structure proche d'une fonction logarithmique.

dit, la fonction doit avoir une forme logarithmique (si $f(x)$ est une fonction à forme logarithmique, alors $\lim_{x\to\infty}(f(x+1)-f(x))=0$)[3]. L'idée sous-jacente à cette hypothèse est qu'il est plus coûteux de passer d'un chemin de longueur 1 à un chemin de longueur 2 (*i.e.* la longueur du chemin double) qu'un chemin de longueur 15 à un chemin de longueur 16 (*i.e.* la longueur globale du chemin évolue peu). De plus, l'évolution logarithmique est à la base des mesures sur la théorie de l'information qui ont montré leurs efficacités [Resnik, 1995, Budanitsky et Hirst, 2006]. Par ailleurs, [Zhong *et al.*, 2002] avait déjà expérimenté ce principe en proposant une pondération $\frac{1}{2^x}$ (section 1.2.3.2.3)

Pour respecter les hypothèses, nous proposons la formule suivante basée sur la fonction à forme logarithmique $n/n+1$ (voir figure 3.2). Ainsi, nous définissons :

$$W(path_X(x,y)) = T_X \times \left(\frac{|path_X(x,y)|}{|path_X(x,y)|+1}\right)$$

3.2.2 Chemin de type multiple

Considérons maintenant un chemin quelconque entre deux concepts x et y dans une ontologie. Nous noterons ce chemin un *chemin de type multiple* (*i.e.* CTM) car il peut contenir plusieurs types de relations différents. Ce chemin peut être décomposé en une liste ordonnée de n CTU telle que :

$$path(x,y) = path_{X_1}(x,z_1) \oplus path_{X_2}(z_1,z_2) \oplus ... \oplus path_{X_n}(z_{n-1},y)$$

Néanmoins, il existe plusieurs décompositions possibles pour un chemin donné $path(x,y)$.

[3] L'utilisation directe de la fonction logarithme n'est pas possible, car elle ne respecte pas la condition (b), liée à l'existence d'une borne maximale.

Par exemple, pour le chemin $path(jeep, object) = \{jeep \xrightarrow{is\text{-}a} car \xrightarrow{has\text{-}part} hood \xrightarrow{is\text{-}a} cover \xrightarrow{is\text{-}a} object\}$ (c.f. figure 1.4) il existe deux décompositions en CTU possibles : la décomposition en trois CTU $\{\{jeep \xrightarrow{is\text{-}a} car\}, \{car \xrightarrow{has\text{-}part} hood\}, \{hood \xrightarrow{is\text{-}a} cover \xrightarrow{is\text{-}a} object\}$ et la décomposition en quatre CTU $\{\{jeep \xrightarrow{is\text{-}a} car\}, \{car \xrightarrow{has\text{-}part} hood\}, \{hood \xrightarrow{is\text{-}a} cover\}, \{cover \xrightarrow{is\text{-}a} object\}$.

Nous définissons la décomposition minimale $T_{min}(path(x, y))$ comme la décomposition qui minimise la cardinalité n de la décomposition de $path(x, y)$ en CTU. Cette décomposition est construite selon la nature transitive ou non-transitive des arêtes du CTM :

- Si une arête donnée du CTM est du type d'une relation X qui n'est pas transitive (*e.g.* $isLocatedAt$ ou $madeWith$), alors un chemin de longueur 2 de ce type de relation n'a conceptuellement pas de sens. Donc, chaque arête d'un CTM associée à une relation non transitive correspond à un CTU de longueur 1.
- Sinon, si la relation X donnée est transitive (*e.g.* *is-a*, *part-of*), alors toutes les arêtes du type X aux extrémités font partie du même CTU. Autrement dit, si l'on considère une sous-partie $path_t(a, b)$ du chemin $path(x, y)$ telle que toutes les relations entre a et b sont transitives, pour tout élément successif de l'ensemble $T_{min}(path_t(a, b))$ les relations sont différentes (*i.e.* $\forall i \in [1, n-1]$, $X_i \neq X_{i+1}$).[4] Cette propriété peut être utilisée pour construire en pratique cette factorisation $T_{min}(path(x, y))$, car en parcourant le chemin les changements de types de relations (et uniquement à ce moment là) induisent la coupure en CTU.

Par exemple, sur la figure 1.4, $T_{min}(path(jeep, object)) =$

$$\left\{ \begin{array}{l} \left\{jeep \xrightarrow{is\text{-}a} car\right\}, \\ \left\{car \xrightarrow{has\text{-}part} hood\right\}, \\ \left\{hood \xrightarrow{is\text{-}a} cover \xrightarrow{is\text{-}a} object\right\} \end{array} \right\}$$

Remarquons que $\left\{jeep \xrightarrow{is\text{-}a} car\right\}$ et $\left\{hood \xrightarrow{is\text{-}a} cover \xrightarrow{is\text{-}a} object\right\}$ sont deux CTU disjoints de $T_{min}(path(x, y))$, même s'ils ont le même type de relation, car ils n'ont pas de concepts en commun.

Nous proposons finalement comme coût sémantique pour un CTM de considérer la somme des poids des CTU composant la factorisation minimale $T_{min}(path(x, y))$. Ainsi, le poids du chemin $path(x, y)$ est défini par :

$$W(path(x, y)) = \sum_{p \in T_{min}(path(x,y))} W(p)$$

Nous pouvons illustrer la formule précédente sur le chemin $path(jeep, object)$ précédent

[4] Preuve par l'absurde : Soit $path(x, y)$ un CTM composé uniquement de relations transitives. Supposons que $T_{min}(path(x, y))$ se décompose en n CTU tel que $\exists i \in [1, n-1]$, $X_i = X_{i+1}$. Ceci implique que le chemin $path_{X_i}(z_i, z_{i+1}) \oplus path_{X_{i+1}}(z_{i+1}, z_{i+2})$ est un chemin de type unique. Donc, il existe une factorisation de $path(x, y)$ avec $n - 1$ CTU, ce qui est plus petit que la factorisation dite minimale $T_{min}(path(x, y))$. Nous arrivons à une contradiction sur la minimalité de $T_{min}(path(x, y))$. Finalement, pour la factorisation minimale $T_{min}(path(x, y))$, nous avons la propriété $\forall i \in [1, n-1]$, $X_i \neq X_{i+1}$.

avec[5] $IC(jeep) = 1.0, IC(car) = 0.68, IC(hood) = 1.0, IC(cover) = 0.55, IC(object) = 0.08$ et comme poids pour la relation *part-of* $TC_{has\text{-}part} = 0.4$. Les poids des trois CTU composant ce chemin sont :

- $W(\{jeep \xrightarrow{is\text{-}a} car\}) = |IC(jeep) - IC(car)| = 0.32$
- $W(\{car \xrightarrow{has\text{-}part} hood\}) = 0.4 \times \frac{1}{1+1} = 0.2$
- $W(\{hood \xrightarrow{is\text{-}a} cover \xrightarrow{is\text{-}a} object\}) = |IC(hood) - IC(object)| = 0.92$

Finalement, la somme de ces différents coûts permet d'obtenir le poids final de ce chemin allant de *jeep* à *object* : $W(path(jeep, object)) = 0.32 + 0.2 + 0.92 = 1.44$. Notons bien que le calcul que nous venons de faire ne considère *qu'un* chemin entre *jeep* et *object*. Nous verrons dans la section suivante qu'il faut considérer l'ensemble des chemins entre deux concepts (et donc l'ensemble des poids de chaque chemin) pour calculer notre mesure finale.

Nous pouvons remarquer que la définition d'un poids pour un CTM est cohérente avec la distance de Jiang & Conrath. En effet, dans le cas d'un CTM hiérarchique entre deux concepts c_1 et c_2, il y a deux types de relations uniquement, la relation montante *is-a* entre c_1 et le *ccp* des deux concepts et la relation descendante *includes* du *ccp* au concept c_2. On obtient ainsi comme poids pour ce type de chemin :

$$\begin{aligned} W(path(c_1, c_2)) &= W(path_{is\text{-}a}(c_1, ccp(c_1, c_2)) + W(path_{includes}(ccp(c_1, c_2), c_2)) \\ &= |IC(c_1) - IC(ccp(c_1, c_2))| + |IC(ccp(c_1, c_2)) - IC(c_2)| \\ &= IC(c_1) + IC(c_2)) - 2 \times IC(ccp(c_1, c_2)) \end{aligned}$$

ce qui correspond bien à la distance de Jiang & Conrath (section 1.2.3.3.4 page 34).

3.2.3 Mesure finale

Comme nous en discutions dans la section 1.2.3.4.1 page 36, en considérant un modèle sémantique plus riche que la hiérarchie de concept, il existe une grande quantité de chemins possibles entre deux concepts donnés, alors que tous ces chemins n'ont pas forcément un sens sémantiquement. Il est donc nécessaire de filtrer l'ensemble des chemins entre deux concepts donnés pour ne garder que ceux qui sont corrects sémantiquement.

Pour calculer les chemins corrects sémantiquement, nous avons décidé d'utiliser la définition de [Hirst et St-Onge, 1998] . Ces patrons de chemin ont été validés et sont récurrents dans de nombreux travaux, ce qui amène à penser qu'ils sont une bonne approximation de la notion de chemin sémantiquement correct. Soit deux concepts c_1 et c_2. Nous noterons $\pi(c_1, c_2)$ l'ensemble des chemins élémentaires (*i.e.* ne contenant pas de circuit) entre les concepts c_1 et c_2. De plus, nous noterons $HSO : \pi(c_1, c_2) \rightarrow \mathbb{B}$ la fonction à affectation booléenne (*i.e.* $\mathbb{B} = \{true, false\}$) permettant de décider (selon les patrons de Hirst & St-Onge) si un chemin est sémantiquement correct ou non. Notre distance finale correspond alors au poids minimal W parmi tous les chemins sémantiquement corrects entre les concepts c_1 et c_2. Plus

[5] Les cinq valeurs IC données ici en exemple sont des valeurs réelles calculées par la formule de Seco [Seco *et al.*, 2004] sur WordNet 3.0.

formellement, nous obtenons :

$$dist(c_1, c_2) = \min_{\{p \in \pi(c_1,c_2) | HSO(p)=true\}} W(p)$$

Prenons pour exemple (toujours sur la même figure) le calcul de distance entre les concepts *jeep* et *object*. Le chemin considéré à la section précédente (passant par un *part-of*) est un chemin sémantiquement correct, donc la valeur 1.44 trouvée compte dans le calcul du minimum. Le chemin hiérarchique entre ces deux concepts est $\{jeep \xrightarrow{is\text{-}a} car \xrightarrow{is\text{-}a} wheeled\_vehicle \xrightarrow{is\text{-}a} vehicle \xrightarrow{is\text{-}a} object\}$ et son coût sémantique est alors obtenu par : $|IC(jeep) - IC(object)| = 0.92$. Ainsi, dans cet exemple, le chemin relationnel n'apporte pas d'information et le score final est : $dist(jeep, object) = \min\{1,44; 0,92\} = 0,92$. Ce résultat était attendu, car *object* étant un concept très général il n'est pas relié de manière sémantiquement intéressante avec des concepts spécifiques. Au contraire, si nous considérons maintenant la distance entre *jeep* et *hood*, il existe uniquement deux chemins sémantiquement corrects : le chemin $\{jeep \xrightarrow{is\text{-}a} car \xrightarrow{has\text{-}part} hood\}$ (correspondant à un poids de $0,52$) et le chemin totalement hiérarchique (en utilisant le *ccp object*, correspondant à un poids de $1,50$). Dans ce dernier exemple, c'est le chemin passant par une relation qui donne le meilleur résultat : $dist(jeep, hood) = \min\{1,50; 0,52\} = 0,52$.

Nous pouvons remarquer que le chemin hiérarchique entre deux concepts c_1 et c_2 (correspondant à un calcul de similarité) est toujours un chemin sémantiquement correct. Notre distance considérera donc toujours les chemins relationnels proportionnellement au chemin hiérarchique. De ce point de vue, nous respectons la définition du degré de relation sémantique de [Resnik, 1995] telle qu' « une mesure de degré de relation sémantique est une généralisation d'une mesure de similarité sémantique ». De même, comme nous l'avons démontré à la section précédente, nous évaluons un chemin hiérarchique de manière cohérente avec le calcul de Jiang & Conrath. Ainsi, nous pouvons conclure que notre formule est une proposition de généralisation relationnelle de la formule de Jiang & Conrath. Cette propriété a des conséquences directes sur la valeur finale de notre formule. En effet, le résultat de Jiang & Conrath faisant toujours partie de l'ensemble des valeurs considérées pour obtenir notre distance, nous avons la propriété suivante :

$$\forall (c_1, c_2) \in C^2,\ dist(c_1, c_2) \leq dist_{JC}(c_1, c_2)$$

Comme conséquence, si la fonction IC est bornée sur l'intervalle $[0, IC_{max}]$ (ce qui est le cas avec la formule de [Seco *et al.*, 2004] avec $IC_{max} = 1.0$), le poids d'un chemin hiérarchique peut être borné dans l'intervalle $[0, 2 \times IC_{max}]$ ($2 \times IC_{max}$ étant la valeur maximale de la distance de Jiang & Conrath, entre deux feuilles ayant comme *ccp* la racine de la taxonomie). Ainsi, notre distance peut être, si nécessaire, convertie en une mesure de degré de relation sémantique (homogène à la progression d'une similarité) en utilisant une conversion linéaire classique [Resnik, 1995, Jiang et Conrath, 1997] :

$$rel(c_1, c_2) = 2 \times IC_{max} - dist(c_1, c_2)$$

3.3 Évaluation

L'objectif de cette évaluation est d'étudier la pertinence de nos propositions (poids, chemin non-hiérarchique, validité des chemins sémantiques, etc.) pour le calcul du degré de relation sémantique. Dans cette section, nous présentons l'évaluation faite sur WordNet en utilisant deux ensembles de test tirés de la littérature et présentés précédemment dans la section 1.2.4.

3.3.1 Protocole de test

Nous avons utilisé pour tester notre mesure sémantique le protocole de Miller & Charles et le protocole WordSimilarity-353. Le premier test est basé sur la notion « jugement synonymique », ce qui implique qu'il n'est pas le meilleur choix pour tester une mesure de degré de relation sémantique. Néanmoins, notre mesure doit obtenir de bons résultats sur ce test, étant donné qu'une mesure de similarité sémantique est un cas particulier de mesure de degré de relation sémantique. Il était donc nécessaire de trouver, en plus du test de Miller & Charles, un autre test permettant d'évaluer les capacités relationnelles des mesures de degré de relations sémantiques. Le test le plus pertinent, à notre connaissance, à ce sujet est le test WordSimilarity-353[6] [Finkelstein *et al.*, 2001] qui propose majoritairement des couples connectés relationellement. De manière logique, les mesures sémantiques de similarité tendent à échouer à ce test (*i.e.* avoir une corrélation très faible). Notre mesure devrait néanmoins être capable de donner de bons résultats sur ce test, mais nous verrons que le choix d'une ontologie est aussi un point critique qui peut limiter les résultats.

Pour calculer nos scores sémantiques, nous avons utilisé comme MRC la sous-partie *nom* de la base lexical WordNet 3.0 [Fellbaum, 1998].[7] Nous ne considérerons dans notre test que les relations non-hiérarchiques transitives *part-of* de WordNet.[8] Pour le test WS-353, nous avons retiré 9 couples de mots lorsqu'au moins l'un des mots du couple n'existait pas comme nom dans WordNet (*e.g.* le couple « fighting-defeating »).

Ne pouvant anticiper une valeur initiale correcte pour le poids $T_{part\text{-}of}$, nous avons effectué l'évaluation pour un ensemble de valeurs allant de 0 (*i.e.* lien non-hiérarchique gratuit) à 1.5 (*i.e.* lien non-hiérarchique trop coûteux pour être intéressant).

De plus, la quantité d'information IC d'un concept a été calculée selon la formule de [Seco *et al.*, 2004]. Nous avons considéré pour notre évaluation quatre mesures de similarité ([Rada *et al.*, 1989], [Resnik, 1995], [Lin, 1998] et [Jiang et Conrath, 1997]) et une mesure de degré de relation sémantique ([Hirst et St-Onge, 1998]). [9]

Les résultats des corrélations sont donnés dans le tableau 3.1. L'évolution du facteur de corrélation en fonction du poids attribué à $T_{part\text{-}of}$ est donnée dans la figure 3.3.

[6] `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html`

[7] Bien que la base de connaissance WordNet ne soit pas une ontologie, elle reste néanmoins la base le plus facile d'accès et large (en termes de concepts).

[8] WordNet considère trois relations de méronymie (*meronym_member, meronym_part, meronym_substance*) et trois d'holonymie (*holonym_member, holonym_part, holonym_substance*).

[9] Pour comparer objectivement les facteurs de corrélation, nous les avons tous recalculé suivant la même API (présentée chapitre 6) avec WordNet 3.0.

	Corrélation	
Mesures	Miller & Charles	WordSimilarity-353
Rada	0,638	0,249
Resnik	0,804	0,375
Lin	0,836	0,377
Jiang & Conrath	0,880	0,362
Hirst & St-Onge	0,847	0,380
Notre mesure avec $T_{part\text{-}of} = 0.4$	0,902	0,400

TAB. 3.1 – Corrélation de Pearson pour les mesures de Rada, Resnik, Lin, Jiang & Conrath, Hirst & St-Onge et notre approche pour le poids donnant les valeurs optimales ($T_X = 0.4$).

3.3.2 Discussions

Pour les deux tests, notre mesure dépasse les résultats de corrélation des autres mesures. De plus, à notre connaissance, c'est la première fois qu'une mesure basée uniquement sur la structure en graphe de WordNet atteint une corrélation de 0.4 sur le test WS-353 (voir [Strube et Ponzetto, 2006] pour la dernière évaluation connue).

Puisque que le test de Miller & Charles est majoritairement basé sur la notion de similarité, la plupart des résultats que nous obtenons pour ces couples utilisent un chemin hiérarchique pur (*i.e.* pas de contribution d'arêtes non-hiérarchiques). Ainsi, une partie importante des résultats est équivalente aux résultats de la mesure de Jiang & Conrath (comme la théorie le prévoyait). Néanmoins, quelques résultats sont améliorés grâce à l'utilisation de liens non-hiérarchiques. Par exemple, pour le couple « furnace-stove », le *ccp* (*i.e.* plus proche parent commun) hiérarchique est « artifact », ce qui est une très faible information. En utilisant les relations, notre mesure a pu considérer le chemin « furnace *has-part* grate *part-of* stove » qui utilise une propriété fonctionnelle commune aux deux objets. Ce chemin étant identifié en utilisant les hypothèses de chemin sémantique de Hirst & St-Onge, leur mesure trouve aussi un meilleur score sur ce couple que les mesures de similarité. Néanmoins, leur résultat souffre de l'absence de pondération appliquée aux arêtes. Notre mesure tire ses avantages des deux paradigmes : la recherche de chemin relationnel sémantiquement correct et l'approche par la théorie de l'information pour pondérer les arêtes et ainsi affiner les résultats.

Remarquons de plus que ni notre mesure ni celle de Hirst & St-Onge n'ont pu trouver de chemin pertinent pour le couple « journey-car ». Ceci s'explique par le lien particulier entre ces deux concepts, qui ne correspond pas à une relation de type méronymie/holonymie telle que celles définies dans WordNet. Ceci met en évidence les limites de WordNet en tant que modèle relationnel et affirme le besoin d'ontologies spécifiques d'un domaine pour pouvoir utiliser les capacités des mesures relationnelles.

Comme c'était attendu, les mesures de degré de relation sémantique (Hirst & St-Onge et la notre) obtiennent de meilleurs résultats que les mesures de similarités sur le test WS-353, puisque ce test contient majoritairement des couples liés relationellement. Les mesures relationnelles peuvent trouver des liens entre des mots (*e.g.* « keyboard-computer », etc.) là où les mesures de similarité ne peuvent utiliser que des liens hiérarchiques. De plus, les patrons de Hirst & St-Onge permettent d'invalider certains chemins qui sont courts dans le

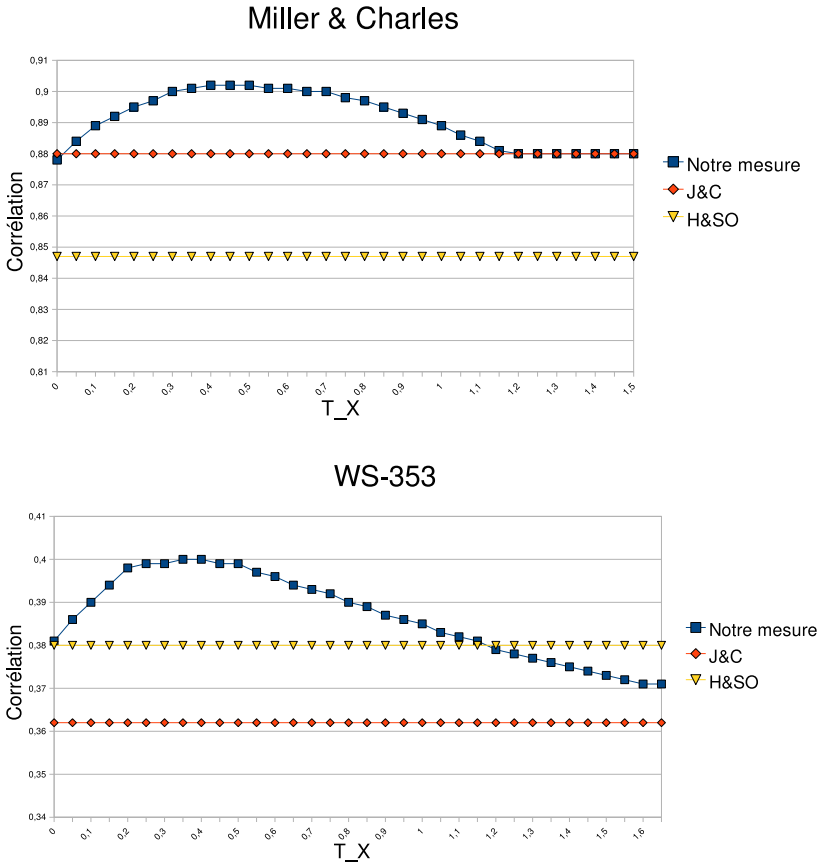

FIG. 3.3 – Facteur de corrélation de Pearson par rapport à l'évolution du paramètre T_X pour le test de Miller & Charles et le test WS-353. La corrélation maximale a été obtenue dans les deux cas avec le paramètre $T_X = 0.4$.

graphe, mais ne représentent pas un chemin sémantiquement correct (figure 3.4). Néanmoins, là encore certains couples ne sont pas connectés dans WordNet et aucun chemin ne peut être trouvé par notre mesure (*e.g.* « telephone-communication »). Ceci s'explique encore par le manque de relation dans WordNet. De plus, comme énoncé par [Strube et Ponzetto, 2006], le test WS-353 contient beaucoup de mots qui sont reliés par des liens de « sens commun » qui ne peuvent pas exister dans WordNet (*e.g.* « popcorn-movie », etc.). Ceci explique pourquoi il est si difficile d'obtenir de bonnes corrélations avec WordNet sur ce test. Ainsi, nous pensons qu'il serait très difficile d'aller au-delà de la limite $0.35 - 0.4$ sur le test WS-353 en utilisant WordNet comme modèle de connaissances. Seul un modèle riche contenant des relations sémantiques variées permettrait d'atteindre de bons degrés de corrélation.

3.4 Algorithmique et complexité

Il est légitime, à partir du moment où notre MRC n'est plus une taxonomie mais un graphe, de se poser la question de la complexité et de la mise en œuvre pratique de notre formule. Cette section a ainsi pour but d'étudier la complexité de notre mesure et de présenter

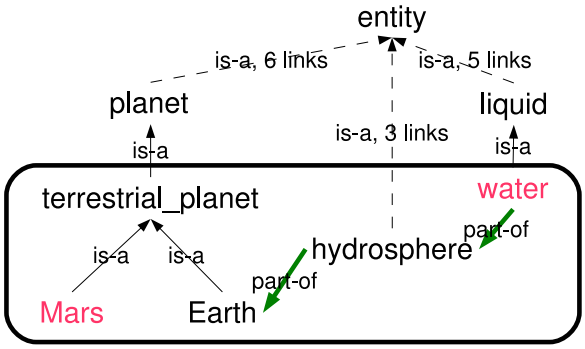

FIG. 3.4 – Exemple de chemin sémantiquement incorrect pour les patrons de Hirst & St-Onge. Du concept « water » à « Mars », le chemin le plus court (du point de vue théorie des graphes) n'est pas valable car le lien $(terrestrial\_planet, includes, Mars)$ n'est pas autorisé.

l'algorithme que nous utilisons en pratique pour calculer cette distance, afin de montrer que ce calcul est réalisable en temps acceptable.

3.4.1 Complexité théorique naïve

La structure de notre mesure peut être modélisée par l'algorithme simple suivant :

Algorithme 3.1 Algorithme simple « naïf »

ENTRÉES: $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$, $(c_{init}, c_{cible}) \in C^2$
SORTIES: $d \in \mathbb{R}^+$
1: $d \leftarrow 2 \times IC_{max}$
2: **pour tout** $c \in \pi(c_{init}, c_{cible}) | HSO(c) = true$ **faire**
3: $\quad d \leftarrow \min\{d, W(c)\}$
4: **fin pour**
5: **renvoyer** d

Dans cet algorithme, la complexité associée au calcul du coût sémantique d'un chemin (ligne 3) est en $\mathcal{O}(1)$. En effet, pour calculer le poids $W(c)$ d'un chemin c nous avons deux formules en fonction du type de liens :

- Si le chemin est hiérarchique, le coût de ce chemin est déterminé par la fonction $W(path_{X \in \{is\text{-}a, includes\}}(x, y)) = |IC(x) - IC(y)|$. L'appel à la fonction IC est en $O(1)$ et donc le poids d'un CTU hiérarchique est en $\mathcal{O}(1)$.
- Si le chemin est relationnel, le coût est déterminé par la fonction $W(path_X(x, y)) = T_X \times \left(\frac{|path_X(c_1, c_2)|}{|path_X(c_1, c_2)|+1} \right)$. Là encore, le coût est en $\mathcal{O}(1)$.[10]

Donc, de manière générale, quel que soit le chemin considéré le calcul de la fonction de poids W est de l'ordre de $\mathcal{O}(1)$.

Le coût de la boucle (ligne 2) est lui moins évident. En effet, si nous étudions le pire cas dans lequel le graphe associé est complet, l'énumération de l'ensemble des chemins possibles

[10] Nous supposerons ici que calculer la cardinalité d'un ensemble est en $\mathcal{O}(1)$. Autrement dit, la valeur associée à cette cardinalité n'est pas calculée en temps réel mais est stockée sous forme de cache. La complexité mémoire perdue est minime étant donné qu'elle ne représente que la taille d'un entier. La plupart des langages de programmation ont maintenant fait ce choix.

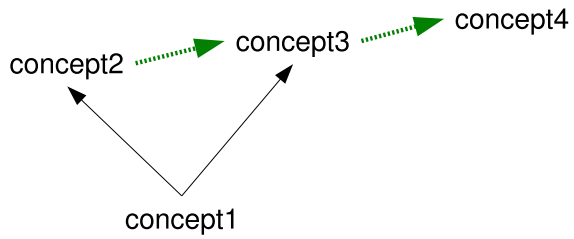

FIG. 3.5 – Exemple du problème d'allocation d'un poids unique à une arête. Dans cet exemple, il n'est pas possible d'attribuer un poids unique à l'arête $\{3,4\}$, car ce poids change en fonction de l'arête antécédente dans le chemin (l'arête $\{2,3\}$ ou l'arête $\{1,3\}$) .

entre deux concepts donnés est égal à $\sum_{i=0}^{n-2} A_{n-2}^{i}$, avec n le nombre de sommets du graphe et A_m^k le nombre d'arrangement sans répétitions à k éléments parmi un ensemble à m éléments.[11] Ainsi, la complexité de cette énumération est de l'ordre de $\mathcal{O}(n^{n-1})$. Les opérations intérieures de la boucle d'énumération étant en $\mathcal{O}(1)$, ceci implique que la complexité finale de l'algorithme 1 est de l'ordre de $\mathcal{O}(n^{n-1})$, ce qui n'est pas utilisable en pratique. La section suivante présente ainsi l'algorithme optimisé que nous avons utilisé pour pouvoir calculer facilement notre mesure.

3.4.2 Algorithmique optimisé

3.4.2.1 Principe de l'algorithme optimisé

Si la complexité de l'algorithme précédent est importante, il est néanmoins possible d'envisager un certain nombre d'optimisation afin de le rendre efficace en temps. Ces optimisations sont basées sur deux hypothèses :

- Il est simple et peu coûteux en complexité de calculer le score du chemin hiérarchique entre les deux concepts. En commençant par calculer ce score, nous obtenons ainsi une borne maximale que ne dépassera jamais notre distance (comme nous l'avons justifiée section 3.2.3).
- Toutes les arêtes peuvent être associées à un poids sémantique. En conséquence, le problème d'énumération des chemins peut être transformé en recherche du chemin de poids minimal (les pondérations des arêtes étant positives) entre deux concepts dans un graphe.

Cette deuxième hypothèse implique que plutôt que de calculer l'ensemble des chemins, puis les énumérer, nous allons calculer le score sémantique de plusieurs chemins partiels en parallèle, tant que nous n'aurons pas trouvé un chemin liant les deux concepts. Pour ce type de recherche, l'algorithme que nous allons utiliser est l'algorithme de Dijkstra[12], algorithme le plus classique pour ce type de problème, en l'adaptant à notre situation.

La principale difficulté est que le poids d'un chemin non-hiérarchique est dépendant d'un chemin complet de longueur n, et non d'une somme des arêtes (c.f. section 3.2.1.2). Pour

[11] $A_m^k = \frac{m!}{(m-k)!}$

[12] Il n'est pas nécessaire d'envisager des versions plus complexe, tel que l'algorithme de Ford-Bellman, car les poids sont ici toujours positifs.

une arête donnée non-hiérarchique, sa contribution sémantique sera différente en fonction du type de l'arête précédente dans le chemin (figure 3.5). Ainsi, là où l'algorithme de Dijkstra ne mémorise qu'une liste de nœuds pouvant être explorés, nous mémoriserons une liste de chemins partiels pouvant être continués. Ainsi, il pourra exister plusieurs chemins partiels terminant par le même concept, mais devant être considérés séparément car le chemin pour arriver à ce concept sera différent. Par exemple, sur la figure 3.5, les deux chemins partiels $\{1,3\}$ et $\{1,2,3\}$ commençant tous les deux au sommet 1 et terminant tous les deux au sommet 3 doivent être mémorisés séparément.

La section suivante présente un certain nombre de notations et fonctions annexes utiles pour décrire notre utilisation de l'algorithme de Dijkstra.

3.4.2.2 Notations et fonctions annexes

Nous définissons dans cette section un certain nombre de notations et de fonctions annexes utiles pour décrire l'algorithme principal.

Structure formelle des chemins

La structure d'un chemin a déjà été évoquée dans la section 3.2.1. Un chemin de type multiple est composé d'une liste ordonnée de plusieurs chemins de type unique, eux-mêmes composés d'une liste ordonnée de triplets $\langle x, r, y\rangle \in L$. Nous avons besoin de définir un certain nombre de fonctions de manipulation de base sur les chemins, pour simplifier l'écriture de l'algorithme final. Soit $\mathcal{P}^*(L)$ l'ensemble des chemins possibles selon l'ensemble des arêtes L dans la taxonomie $\mathcal{TA}$ et $P \in \mathcal{P}^*(L)$ un CTM, alors les méthodes suivantes sont accessibles :

- $P.lastCTU \rightarrow \mathcal{P}^*(L)$: renvoie le dernier CTU du chemin P. On utilisera régulièrement le cardinal de ce CTU, correspondant à sa longueur : $|P.lastCTU|$
- $P.lastType \rightarrow R \cup \{is\text{-}a, includes\}$: renvoie le type de relation composant le dernier CTU du chemin C
- $P.lastNode \rightarrow C$: renvoie le dernier nœud de ce chemin

Comme nous l'avons évoqué dans la section précédente, il n'est pas possible de caractériser la valeur d'une arête unique, car elle changera en fonction du type de l'arête précédente. Deux chemins distincts finissant au même nœud ne sont pas forcément substituables. Il est donc nécessaire d'être capable d'évaluer si deux chemins peuvent se substituer ou non, pour le processus de mise à jour de l'algorithme de Dijkstra. Nous pouvons définir la fonction $subs : \mathcal{P}^*(L) \times \mathcal{P}^*(L) \rightarrow \mathbb{B}$ qui détermine si deux chemins sont substituables ou non. Soit $(P_1, P_2) \in \mathcal{P}^*(L) \times \mathcal{P}^*(L)$, alors la fonction $subs$ est définie :

$subs(P_1, P_2) =$

$$\begin{cases} true & \begin{cases} \text{Si} & (P_1.lastNode = P_2.lastNode) \\ \wedge & \left(\begin{array}{l} (|P_1.lastCTU| = |P_2.lastCTU| \wedge P_1.lastType = P_2.lastType) \\ \vee (P_1.lastType \in L_H \wedge P_2.lastType \in L_H) \end{array} \right) \end{cases} \\ false & \text{Sinon} \end{cases}$$

La complexité de ces formules est de l'ordre de $\mathcal{O}(1)$ et leurs utilisations n'influeront donc pas sur la complexité de l'algorithme final.

Arbre binaire de recherche

La variable notée *tree* dans l'algorithme représentera une structure de stockage sous forme d'arbre binaire de recherche. Cette structure a l'avantage de proposer des algorithmes d'insertion, de suppression et de recherche en $\mathcal{O}(\log n)$, où n est ici le nombre de nœuds de l'arbre de recherche. Nous utiliserons les méthodes suivantes appliquées à un arbre :

- $tree.add(X)$: cette fonction insère une cellule X dans l'arbre. La cellule X doit être associée à une valeur numérique permettant son classement dans l'arbre.
- $tree.remove(X)$: cette fonction retire une cellule X de l'arbre. La valeur numérique associée à X permet de retrouver directement la cellule.
- $tree.first$: cette fonction renvoie et supprime de l'arbre la cellule classée en premier, en se basant sur la valeur numérique associée aux cellules.

Dans les algorithmes suivants, nous classerons dans l'arbre des chemins $P \in \mathcal{P}^*(L)$. Nous utiliserons alors comme valeur numérique de classement le poids $W(P)$ du chemin P.

Accès aux voisins

La fonction $voisins : C \rightarrow \mathcal{P}(L)$. Renvoie les arêtes voisines à un concept de départ. La fonction peut s'écrire simplement :

$$voisins(c) = \{a \in L, r \in R, x \in C | a = \langle c, r, x \rangle\}$$

3.4.2.3 Algorithme final

L'algorithme 2 représente notre algorithme final optimisé.

La ligne 1 représente le calcul de la borne hiérarchique évoquée précédemment. Ainsi, dans la boucle de mise à jour (ligne 13), un nouveau chemin sera mémorisé uniquement s'il est inférieur à la borne maximale hiérarchique. Cette limite est particulièrement efficace pour les concepts qui sont proches hiérarchiquement, car dans cette situation beaucoup de chemins seront coupés très tôt, ce qui dirige l'exploration.

La mise à jour des arbres tient compte de la méthode utilisée pour atteindre un nœud. Ainsi, quand un nouveau chemin est construit, il remplace tous les chemins de poids supérieur qui peuvent être substitués par lui (ligne 15-18).[13]

L'algorithme de Dijkstra est reconnu pour être d'une complexité de l'ordre $\mathcal{O}(m+n\log n)$ (avec n les sommets et m les arêtes) si la structure de calcul du prochain sommet minimal est en $\mathcal{O}(\log n)$ (tel qu'un arbre de recherche ou une structure par tas de Fibonacci). C'est une complexité pire-cas qui est loin d'être la complexité réelle en pratique, dans la mesure où le calcul ne tient pas compte des chemins sémantiquement corrects. En effet, en considérant le pire-cas, tous les chemins du graphe sont sémantiquement corrects et l'appel à la fonction HSO ne réduit pas l'espace de recherche. Ainsi en pratique lors de l'évaluation présentée

[13] A noter qu'en pratique, l'utilisation d'une table de symboles (tel qu'une table de hachage) permet d'éviter l'utilisation d'une boucle sur les chemins connus (ligne 14) et optimise encore l'algorithme. Cette version n'est pas présentée ici pour une meilleure compréhension globale.

Algorithme 3.2 Algorithme optimisé de notre mesure de degré de relation sémantique

Entrées: $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$, $(c_{init}, c_{cible}) \in C^2$
Sorties: $d \in \mathbb{R}^+$

1: $limit \leftarrow dist_{JC}(c_{init}, c_{cible})$ // La borne maximale est la distance de J&C
2: $tree \leftarrow \{\langle c_{init}, \emptyset, c_{init} \rangle\}$ // Avec $\emptyset$ la relation de coût 0.
3: **tantque** *tree* n'est pas vide **faire**
4: $P \leftarrow tree.first$
5: $c_{tmp} \leftarrow P.lastNode$
6: **si** $c_{tmp} = c_{cible}$ **alors** // On a trouvé le concept, fin.
7: **renvoyer** $W(P)$
8: **finsi**
9: $D \leftarrow voisins(c_{tmp})$
10: **pour tout** $\langle c_{tmp}, r, c_{voisin} \rangle \in D$ **faire**
11: $P_{tmp} \leftarrow P \cup \langle c_{tmp}, r, c_{voisin} \rangle$
12: $w_{tmp} \leftarrow W(P_{tmp})$
13: **si** $HSO(P_{tmp}) \wedge (w_{tmp} < limit)$ **alors** // Test limite et patrons de HSO
14: **pour tout** $P \in tree$ **faire** // Boucle de mise à jour
15: **si** $subs(P, P_{tmp}) \wedge (w_{tmp} < W(P))$ **alors**
16: $tree.remove(P)$
17: **finsi**
18: $tree.add(P_{tmp})$
19: **fin pour**
20: **finsi**
21: **fin pour**
22: **fin tantque**
23: **renvoyer** *limit* // Ce retour n'intervient que si l'on sort de la boucle principale, et donc qu'aucun chemin non hiérarchique intéressant n'a été trouvé.

dans ce chapitre, les temps de calculs de cette version de l'algorithme étaient comparables aux temps de calculs appliqués aux mesures de similarités sémantiques.

3.5 Conclusion

Dans ce chapitre nous avons décrit notre mesure de degré de relation sémantique entre deux concepts d'une taxonomie augmentée de relations non-hiérarchiques. Cette mesure mélange les patrons de chemin de [Hirst et St-Onge, 1998] pour le filtrage des chemins sémantiquement corrects et l'approche à base de théorie de l'information de [Resnik, 1995]. Nous avons testé nos hypothèses sur WordNet en utilisant le test de Miller & Charles [Miller et Charles, 1991] ainsi que le test WordSimilarity-353 [Finkelstein *et al.*, 2001] et nous avons montré que notre mesure obtenait de meilleurs résultats que les autres mesures existantes dans la littérature.

Par ailleurs, l'évaluation met en évidence le manque de relations non-hiérarchiques dans WordNet, tel que cela était déjà mentionné dans [Hirst et St-Onge, 1998]. Par exemple, dans WordNet, il n'y a pas de chemins relationnels entre les mots des couples « journey-car » ou encore « telephone-communication ». Ceci nous amène à conclure que pour utiliser toute la puis-

sance des mesures de degré de relation sémantique sur les ontologies, nous avons besoin d'une ontologie *réelle* d'un domaine. Nous avons donc tenté de récupérer des ontologies à l'aide du moteur de recherche d'ontologie Swoogle[14], de récents résultats montrant qu'une recherche automatique pouvait être intéressante [Finin *et al.*, 2005, Gracia *et al.*, 2006, Sabou *et al.*, 2006]. Néanmoins, nous n'avons pas été en mesure de trouver d'ontologie permettant de tester des concepts aussi généraux que ceux de Miller & Charles ou de WS-353. Pour cette raison, l'un de nos objectifs attenant à cette thèse est de tester l'impact de notre mesure sur le taux de performance de notre application d'agents conversationnels [Mazuel et Sabouret, 2008a], sachant que dans notre architecture, chaque agent est muni de sa propre ontologie décrivant son propre domaine d'activité. Ces résultats nous permettraient de conclure définitivement sur le passage à l'échelle de notre approche sur différents types d'application. Notons de plus que l'utilisation du *part-of* comme unique relation hétérogène dans notre évaluation ne nous a pas permis de tester la pertinence de notre mesure dans le cas de relations non-hiérarchiques non transitives (voir section 3.2.1.2).

D'autre part, nous ne pouvons pas savoir à l'avance si un type de relation représente, pour le développeur de l'ontologie, une relation de rapprochement sémantique. Néanmoins, il semble parfaitement réalisable de faire appel à un expert du domaine pour fixer un ensemble initial de poids T_X. L'expert modélise alors à quel point une relation X donnée représente (ou non) un rapprochement sémantique. Une fois un choix initial (empirique ou selon un expert) effectué, nous pensons qu'il serait possible d'affiner automatiquement ce poids. Dans le cadre humain-agent par exemple, pour éviter les non-sens, un utilisateur peut confirmer au système la validité des déductions sémantiques faites grâce à notre mesure de degré de relation, lorsqu'elle utilise des relations non-hiérarchiques (*p. ex.* Utilisateur : « peux-tu taper sur l'ordinateur pour moi ? », Système : « Par *ordinateur*, vouliez-vous désigner ici le terme *clavier* ? »). Dans le cadre agent-agent, la bonne exécution (confirmée par l'agent émetteur) d'une requête contenant des concepts amenant à utiliser des relations non-hiérarchiques peut permettre à l'agent ayant interprété la requête de conclure sur la validité ou non du poids utilisé. Nous pensons que ces confirmations peuvent servir de base pour un algorithme d'apprentissage par renforcement [Sutton, 1988, Watkins et Dayan, 1992] dont le but serait de comprendre quelles relations amènent à des résultats sémantiques cohérents si elles sont utilisées dans la mesure. Cet algorithme, dont le développement restera une perspective de cette thèse, aurait ainsi en charge d'apprendre les pondérations T_X optimales.

Enfin, notre objectif à long terme serait de proposer (et d'évaluer) une mesure de degré de relation sémantique reposant sur un modèle de représentation des connaissances plus complexe et complet que notre modèle taxonomique avec relations. Par exemple, [Hau *et al.*, 2005] ont proposé une mesure de similarité basé sur une extension de la mesure paramétrable de Lin [Lin, 1998] pour le langage OWL-Lite, qui considère restriction, cardinalité, intersection, etc. Hélas, cette mesure n'a jamais été ni évaluée, ni même implémentée et reste une ébauche théorique. Nous pensons que, en s'appuyant sur ce type de travaux, il est possible d'étendre notre mesure pour modéliser les relations complexes entre les concepts, telles que les intersections ou les disjonctions.

[14] `http://swoogle.umbc.edu/`

Chapitre 4

Application à la communication humain-agent

Sommaire

Ce chapitre traite du module de normalisation des entrées/sorties humain-agent (voir le chapitre 2, et plus particulièrement la figure 2.2 page 97). Ce chapitre est décomposé en deux parties disjointes :

- La modélisation d'une requête reçue de l'utilisateur. Cette partie, traitant de la construction de l'arbre de dépendance des concepts, sera traitée lors de la section 4.1.
- La génération de réponses en langue naturelle. L'objectif étant là de traduire les performatifs et arbres VDL reçus en une phrase en langue naturelle. Cette partie fait l'objet de la section 4.2.

Enfin, la section 4.3 terminera ce chapitre par une évaluation préliminaire du système appliqué à l'interaction humain-agent.

4.1 Modélisation de la commande en langue naturelle

Comme nous le disions dans les chapitres précédents, dans les applications de commandes en langue naturelle, l'interprétation sémantique est souvent le cœur du système et sa con-

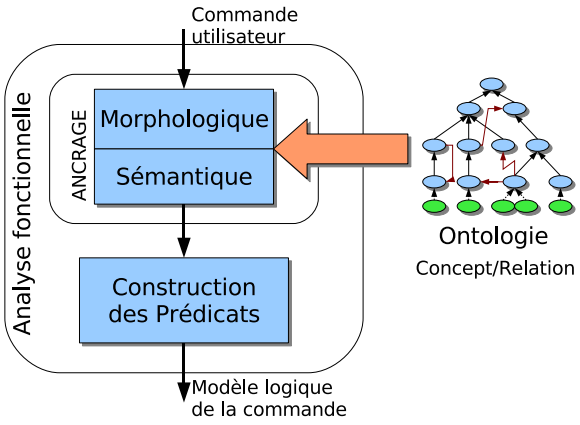

FIG. 4.1 – Module d'analyse fonctionnelle

ception impacte très fortement les modules de Traitements Automatique des Langues (TAL) et de modélisation de la commande de l'utilisateur [Dzikovska *et al.*, 2003, Wahlster, 2003]. Ainsi, beaucoup de modèles ne proposent que des méthodes spécifiques à base de règles assez lourdes à définir. Quelques systèmes essaient d'utiliser un analyseur de surface basé sur la logique du premier ou second ordre [Shapiro, 2000, Milward, 2000] pour réduire la définition de ces règles. 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].

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. De plus, le modèle est plus facile à utiliser pendant l'interprétation sémantique, étant défini par rapport au MRC prévu initialement à cet effet.

Dans les sections suivantes, nous définissons une méthode d'analyse structurelle de surface pour construire une modélisation logique de la commande. Cette analyse est basée sur l'étude des concepts et des relations définies dans l'ontologie de l'agent. Notre analyse s'appuie sur un ancrage des termes de la commande dans le MRC (nous nous plaçons dans le cadre de l'hypothèse de connectivité sémantique de Sadek [Sadek *et al.*, 1997], qui suppose que tous les termes de toute commande apparaissent dans le MRC). En fonction des rôles des termes dans le MRC (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 décrit plus précisément l'ancrage des termes utilisateurs dans le MRC (ancrage morphologique et sémantique). La section 4.1.2 présente l'algorithme de construction logique de la commande (figure 4.1). Enfin, la section 4.1.3 décrit le module d'interprétation des relations et la manière dont l'introspection de l'agent aide pour la résolution contextuelle.

4.1.1 Recherche des concepts équivalents dans le MRC et fonction $Q(R)$

4.1.1.1 Analyse morphologique et lexical

Notre module morphologique et lexical est basé sur la bibliothèque d'outils OpenNLP[1]. 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[2] sur quelques exemples tirés de notre application n'a pas montré de pertes significatives. L'un des avantages important obtenu en utilisant le lemmatiseur basé sur WordNet est qu'il permet la découverte de concepts composés de plusieurs mots (*e.g.* "*dark red*", *"extra large"*).

Nous utilisons un analyseur d'actes de langage [Searle, 1969] pour déterminer le performatif du message de l'utilisateur. Cet analyseur nous permet de distinguer les affirmations, les commandes, les questions ouvertes, les questions à réponse oui/non et les phrases négatives.[3] Son fonctionnement est basé sur la définition d'un automate de la langue par acte reconnaissable. Si une phrase active l'un des automates (ou plusieurs automates), alors le performatif associé à cet automate est associé à la phrase. La reconnaissance des affirmations permet de valider les demandes de confirmations du système (performatif *clarify,* chapitre 2). Le traitement des questions posées à l'agent est développé dans l'architecture VDL, mais ne fait pas l'objet de cette thèse.

4.1.1.2 Ancrage d'un mot dans l'ontologie

Dans notre modèle, le MRC est une hiérarchie augmentée de relations et se modélise sous la forme d'un tuple $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$ (voir section 3.1). L'ensemble C représente l'ensemble des concepts du MRC, et R l'ensemble des relations non hiérarchiques entre les concepts.

Chaque relation $r \in R$ est caractérisée par un label de relation l_r. De même, chaque concept est caractérisé par un label l_c. Par soucis de simplification, nous identifierons l_r et l_c respectivement au concept c et à la relation r, et nous garderons ainsi abusivement les notations C et R pour les ensembles de labels de concepts et de relations aussi bien que pour les ensembles de concepts et de relations en eux-mêmes. Nous noterons de plus $\mathcal{L} = C \cup R$ la liste de tous les labels possibles dans le MRC. Finalement, nous noterons St l'ensemble des chaînes de caractères et $\mathcal{P}(E)$ l'ensemble des parties de l'ensemble E.

Soit W l'ensemble ordonné $w_1, ..., w_n$ des termes utilisés dans la commande. L'ancrage dans le MRC 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 : l'appariement morphologique, la recherche des « approximations sémantiques » et l'ancrage proprement dit.

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

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

[3] Nous remercions à ce sujet les étudiants de Master Julien Heliot et Dia Al Jrab qui ont contribué à développer cet analyseur. Nous présenterons plus en détails ce travail en section 6.3.2 page 174.

4.1.1.2.1 L'appariement morphologique

L'appariement 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 le MRC.

Pour cela, nous utilisons une distance classique entre chaîne de caractères, la distance de Levenshtein (ou distance d'édition). Cette distance compte le nombre de modifications nécessaire pour passer d'une chaîne de caractères à une autre chaîne de caractères. En particulier, elle considère le comptage des trois opérations basiques suivantes :

- La substitution d'un caractère par un autre ;
- L'ajout d'un caractère ;
- Le retrait d'un caractère.

Par exemple, la distance entre le mot « chien » et le mot « chat ». Pour passer de « chien » à « chat », il faut remplacer le « i » par un « a », remplacer le « e » par un « t » et retirer le « n ». Nous avons réalisé trois opérations, la distance de Levenshtein entre « chien » à « chat » est donc de 3. Notons bien que cette distance n'est pas une distance sémantique. La distance entre « voiture » et « automobile » est ainsi très grande alors que la distance entre « professeur » et « processeur » est très faible. Nous noterons $dist_{EDIT} : St \times St \rightarrow \mathbb{N}$ cette formule.

En pratique, nous n'utilisons pas directement cette formule, car elle n'est pas normalisée. Ainsi, la distance entre « cat » et « cats » est la même qu'entre « ordinateur » et « ordinateurs », ce qui n'est pas pertinent. Nous utilisons donc une version normalisée $dist_{N-EDIT} : St \times St \rightarrow [0,1]$:

$$dist_{N-EDIT}(w_1, w_2) = \frac{dist_{EDIT}(w_1, w_2)}{length(w_1) + length(w_2)}$$

avec $length : St \rightarrow \mathbb{N}$ la fonction renvoyant le nombre de caractères du mot en paramètre.

Nous pouvons maintenant détailler la fonction $app_m : St \longrightarrow \mathcal{P}(\mathcal{L})$ qui renvoie la liste des labels du MRC morphologiquement proches de la chaîne de caractère en paramètre. Pour cela, nous utilisons un seuil d'acceptabilité t_{morpho} au-delà duquel le rapprochement morphologique est jugé incorrect.[4] Ensuite, nous considérons l'ensemble des labels maximisant le score d'approximation morphologique. Nous noterons app^*_m le score d'appariement minimal qu'il est possible d'avoir avec une chaîne de caractères donnée :

$$app^*_m(w) = \min_{l \in \mathcal{L}} dist_{N-EDIT}(w, l)$$

La fonction app_m est alors définie de la façon suivante :

$$app_m(w) \quad = \quad \begin{cases} \emptyset & \text{Si } app^*_m(w) > t_{morpho} \\ \{l \in \mathcal{L} | dist_{N-EDIT}(w, l) = app^*_m(w)\} & \text{Sinon} \end{cases}$$

Néanmoins, l'ancrage morphologique ne tient pas compte de la sémantique des concepts et certains ancrages qui seraient corrects peuvent ne pas être trouvés. C'est pourquoi, si l'ensemble des concepts trouvés morphologiquement est vide, nous lançons une recherche

[4] Empiriquement, le seuil est actuellement fixé à $t_{morpho} = 0,1$.

sémantique sur le MRC. Cette recherche sémantique fait l'objet de la section suivante.

4.1.1.2.2 La recherche de « approximations sémantiques »

La recherche des « approximations sémantiques » consiste à trouver l'ensemble des termes du MRC les plus proches sémantiquement d'un mot de la commande, en utilisant des mesures de similarité sémantique comme décrites dans [Budanitsky et Hirst, 2006] et le chapitre 1. 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 de désambiguïsation contextuelle de [Porzel *et al.*, 2003, Resnik, 1995]. Par 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 terme « show » du label de concept *concert*[5].

Pour cette recherche, nous utilisons la base lexical hiérarchisée WordNet [Fellbaum, 1998]. Nous avons choisi d'utiliser la formule de [Jiang et Conrath, 1997] appliquée aux calculs de probabilités définis dans [Seco *et al.*, 2004]. Cette formule calcule un score de « similarité sémantique » compris entre $[0, 1]$ (section 1.2.3.3.4). 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 :

$$app_s(w) = \begin{cases} \emptyset & \text{Si } app^*_{sem}(w) < t_{sem} \\ \{l \in \mathcal{L} | sim_{JC}(l, w) = app^*_{sem}(w)\} & \text{Sinon} \end{cases}$$

avec $t_{sem} \in [0, 1]$ le *seuil d'acceptabilité sémantique*[6] (comparable dans le principe au seuil morphologique t_{morpho} précédent) et la similarité maximum $app^*_{sem}(w) = \max_{l \in \mathcal{L}} sim_{JC}(l, w)$ (comparable dans le principe à la similarité app^*_m. Autrement dit, $app_s(w)$ donne l'ensemble des concepts de l'ontologie de similarité sémantique maximale avec w.

4.1.1.2.3 Ancrage final et fonction de qualité $Q(R)$

A partir des deux fonctions précédentes, nous pouvons définir la fonction $anchor : St \rightarrow \mathcal{P}(\mathcal{L})$ qui permet d'associer un mot d'une commande utilisateur avec une liste potentielle de labels (représentant soit un concept soit une relation) du MRC. La fonction est alors :

$$anchor(w) = \begin{cases} app_m(w) & \text{Si } app_m(w) \neq \emptyset \\ app_s(w) & \text{Sinon} \end{cases}$$

Autrement dit, nous avons décidé de privilégier les résultats de l'alignement morphologique avant les résultats de l'alignement sémantique.

Comme nous l'avons discuté lors du chapitre 2, la modélisation des termes d'une commande utilisateur produit une incertitude qu'il faut prendre en compte. Nous avions à ce

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

[6] Actuellement et empiriquement, la valeur du seuil d'acceptabilité t_{sem} est de 0.7.

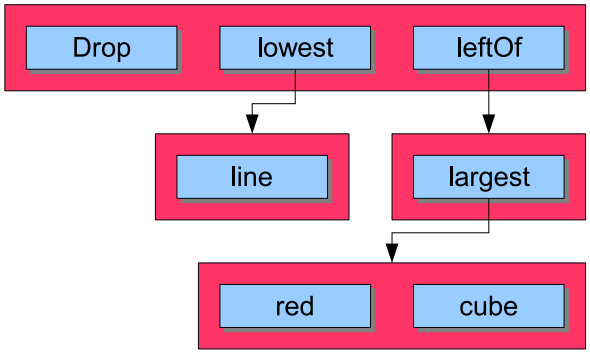

FIG. 4.2 – Modélisation de « drop on the lowest line, left of the largest red cube »

sujet introduit la fonction $Q(R)$ dans la section 2.3.1 page 102, mais sans pouvoir la détailler. Dans le cadre de la modélisation humain-agent, soit A_{req} une modélisation de la commande utilisateur et R la commande en langue naturelle de l'utilisateur. Alors, la fonction $Q(R)$ dans le cadre de l'interaction humain-agent est définie de la façon suivante :

$$Q(A_{req}) = \frac{\sum_{w \in R} \begin{cases} 1 & \text{Si } app_m(w) \neq \emptyset \\ app^*_{sem}(w) & \text{Sinon} \end{cases}}{|R|}$$

Autrement dit, nous considérons qu'un ancrage morphologique ne modifie pas la qualité de modélisation du concept, mais qu'un ancrage sémantique est incertain et que cette incertitude doit être prise en compte. Au final, nous utilisons la moyenne de ces différents scores.

4.1.2 Construction des prédicats

Notre modélisation logique doit capturer la structure fonctionnelle de la phrase. Autrement dit, nous devons construire un ensemble de prédicats représentant les relations entre les concepts (au sens du MRC) tels qu'ils sont exprimés 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 le MRC : 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 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 sur la figure 4.2.

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.

L'un des avantages 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*[7]. 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/valeur(s) (chaque prédicat représentant un nœud, et ses valeurs les fils du nœud). La fonction τ est définie récursivement par l'algorithme suivant :

$\tau(S) =$

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

avec $S = \{c_1, c_2, ..., c_n\}$, $\forall i \in [1, n]$, $c_i = \{s_{i,1}, s_{i,2}, ..., s_{i,k_i}\}$ et $\tau(\emptyset) = \tau(\{\emptyset\}) = \emptyset$

Autrement dit, l'arbre S_a est obtenu en transformant chaque relation de la commande en nœud dont les fils sont les termes restants 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 le traitement de l'exemple suivant par la fonction $\tau(S)$: « *bring me the book written by Frank Herbert which succeeds Dune* » est chunkée en :

> [VP bring :VB] [NP me :PRP] [NP the :DT book :NN] [VP written :VBN] [PP by :IN] [NP Frank :NNP Herbert :NNP] [NP which :WDT] [VP succeeds :VBZ] [NP Dune :NNP]

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

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

$$S = \{\{bring\}, \{book\}, \{written\text{-}by\}, \{Frank, Herbert\}, \{succeeds\}, \{Dune\}\}$$

Nous obtenons alors :

$$\tau(S) = \{bring, book, written\text{-}by(Frank, Herbert), succeeds(Dune)\}$$

4.1.3 Interprétation des relations

Les relations modélisées dans la requête peuvent être de deux types :

- Des références relatives à la situation de l'agent. Une référence relative désigne une partie du contexte de l'agent en utilisant l'état courant de cet agent. L'exemple le plus intuitif correspond au positionnement géographique (*e.g.* « à droite » est une référence relative, car elle ne peut être interprétée qu'en connaissant la position à l'instant courant).
- Des dépendances entre termes (*e.g.* « $to(Boston)$ » est une relation de dépendance).

Pour pouvoir interpréter correctement la requête, un agent doit résoudre préalablement toutes les références *relatives* (*i.e.* dépendante du contexte) existant dans la requête.[8] Cette section présente les méthodes que nous mettons en place pour interpréter ces références relatives.

L'interprétation des relations de la commande consiste ainsi à remplacer chaque relation par un ensemble de concepts du MRC. Par exemple, traduire « *à gauche de la boulangerie* » par une expression non relative du type « *le 32 avenue Guy Moquet* ». Il est ainsi nécessaire de considérer :

1. Les connaissances de l'ontologie. Elles fournissent les définitions des relations (domaine de départ, d'arrivée, instances, etc.)
2. Le contexte courant de l'agent (*i.e.* ses connaissances sur son environnement). Par exemple, l'interprétation de « *à gauche de la boulangerie* » prendra une signification différente en fonction de la position de l'agent dans son environnement au moment de l'interprétation (*e.g.* « *32 avenue Guy Moquet* », « *17 avenue du Président Kennedy* », etc.).

Cette section se découpe donc en deux parties. D'abord, nous expliquerons comment est modélisé le contexte de notre agent et quelles méthodes sont applicables à son analyse. Ensuite, nous expliquerons l'algorithme de transformation de relations qui utilise le MRC, la méthode d'analyse du contexte et une relation de la commande pour obtenir un ensemble résultat de concepts du MRC.

4.1.3.1 Introspection du contexte de l'agent

Nos agents sont programmés en VDL, langage qui permet l'analyse du code *à l'exécution* (section 2.2.1). Pour analyser le modèle que l'agent fait de son environnement, nous avons besoin d'une méthode adaptable à tous les agents (*i.e.* dépendante uniquement du langage VDL). Le modèle VDL étant un modèle arborescent, les concepts sont répartis à des emplacements et des profondeurs très différents. Ainsi, il n'est pas possible de donner une mesure

[8] Les dépendances hors contexte entre termes constituent l'arbre de dépendance dont nous avons détaillé l'utilisation en section 2.3.2 page 105.

```
<grid>
  <line name=upper>
    <row name=left>
      <object>
        <shape>cube</shape>
        <color>green</color>
      </object>
    </row>
    <row name=center>
      <empty/>
    <row>
  </line>
</grid>
```

FIG. 4.3 – Exemple de modélisation XML de l'environnement par un agent VDL.

de distance sémantique entre les concepts VDL. En effet, le code VDL d'un agent combine généralement la représentation des connaissances de l'agent sur le monde ainsi qu'un besoin de manipulation de l'arbre (dans le cadre des effets des actions) par des opérateurs VDL. Pour analyser le code, nous utiliserons donc une heuristique basée sur l'utilisation courante constatée du langage par les programmeurs (convention d'écriture due à la structure d'arbre) :

> Plus la distance dans l'arbre VDL entre deux nœuds est faible, plus les deux nœuds sont reliés contextuellement.

Autrement dit, si un concept a est plus proche en terme de chemin dans l'arbre d'un concept c que le concept b, alors le concept a est plus proche contextuellement que b du concept c. Sur la figure 4.3, le concept *empty* est ainsi plus proche du concept *center* que du concept *left*. Nous pouvons étendre cette hypothèse, qui ne couvre que la comparaison de 2 concepts à un troisième, à la comparaison de deux nuages de concepts entre eux, pour déterminer lequel est le plus « dense » contextuellement par rapport à l'autre :

> Soit deux nuages de concepts de l'agent N_1 et N_2, si la profondeur du concept le plus profond subsumant tous les concepts de N_1 est supérieur à la profondeur du concept le plus profond subsumant tous les concepts de N_2, alors N_1 est plus « dense » que N_2.

Ce calcul reprend l'idée des mesures sémantiques utilisant le *ccp* entre les concepts (c.f. section 1.2). Par exemple sur la figure 4.3, l'objet *cube* est positionné actuellement dans la ligne *upper* et la colonne $left$. Supposons que les deux nuages à considérer soient $\{upper, left, cube\}$ et $\{upper, right, cube\}$. Pour $\{upper, left, cube\}$, le concept le plus subsumant est *line*, qui est de profondeur 1. Pour $\{upper, right, cube\}$, le concept le plus subsumant est *grid*, qui est de profondeur 0. Ainsi, notre heuristique conclue que les concepts $\{upper, left, cube\}$ sont plus proches dans le contexte actuel que les concepts $\{upper, right, cube\}$. L'étendu de la structure arborescente du code contribue au résultat induit par cette hypothèse.

La profondeur n'étant pas un indicateur objectif entre diffèrents agents (dû à différentes implémentations possibles), cette mesure contextuelle ne peut être utilisée que pour comparer

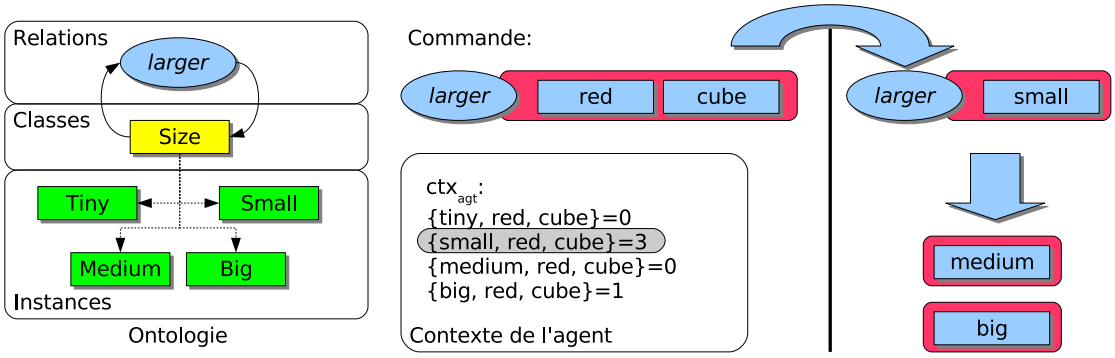

FIG. 4.4 – Exemple d'interprétation de la relation $larger(red, cube)$, qui peut être modélisée à partir d'une commande du type « *pick an object which is larger than the red cube* ». Les paramètres « *red* » et « *cube* » sont d'abord interprétés comme la taille « *small* » après étude du contexte de l'agent. Puis le prédicat $larger(small)$ peut être interprété comme l'un des deux concepts « *medium* » ou « *big* ». Les valeurs numériques de ctx_{agt} sont donnés pour l'exemple, l'agent n'étant pas défini ici.

différents nuages de concepts *du même contexte courant* entre eux et non pas pour les évaluer individuellement.

Dans les sections suivantes, nous noterons $ctx_{agt}(W)$ la fonction prenant en paramètre un ensemble W de concepts de l'agent et renvoyant la profondeur du concept le plus profond subsumant tous les concepts de l'ensemble W.

4.1.3.2 Transformation des relations

Le module de transformation est chargé d'établir une équivalence entre une relation obtenue dans la modélisation de la commande et un ensemble de concepts de l'ontologie. Il utilise pour cela la fonction $ctx_{agt}(W)$ et la définition de la relation dans l'ontologie. Dans notre exemple précédent (« *à gauche de la boulangerie* »), la fonction $ctx_{agt}(W)$ de la section précédente sert à traduire « *la boulangerie* » en « *30 avenue Guy Moquet* » après analyse du contexte courant de l'agent. Ensuite, la transformation des relations se base sur la définition de « *à gauche* » dans l'ontologie, pour déduire que « *à gauche du 30 avenue Guy Moquet* » doit être transformé en « *32 avenue Guy Moquet* ». Ce module se décompose en deux étapes :

1. L'identification du paramètre par rapport au domaine de la relation. Si le paramètre de la relation n'est pas directement une instance du domaine de la relation dans l'ontologie, il faut l'interpréter en utilisant l'analyse du contexte de l'agent pour faire le lien (sur la figure 4.4, « red,cube » est ainsi traduit par « *small* », le domaine de la relation « *larger* » dans l'ontologie étant l'ensemble d'instances « *tiny* », « *small* », « *medium* »[9]).
2. La réduction de la relation. Une fois que l'on a l'assurance que le paramètre est bien du domaine de la relation définie dans l'ontologie, on peut transformer cette relation on considérant son codomaine (sur la figure 4.4, le codomaine contient les instances « *small* », « *medium* », et « *big* ». Les instances reliées à « *small* » par la relation « *larger* » sont « *medium* » et « *big* »).

[9] « *Big* » ne fait pas parti du domaine car il n'est jamais sujet d'un triplet dans le MRC. Sémantiquement parlant, « *big* » n'as pas de taille plus grande que lui. Par contre il fait parti du codomaine.

L'identification n'est nécessaire que si le paramètre ne peut pas être directement réduit à la classe de définition de la relation. Ce cas demande donc une résolution suivant le contexte de l'agent. L'objectif est de chercher parmi toutes les valeurs possibles parmi les instances du domaine de la relation celle qui maximise le score de contexte (au sens de la fonction ctx_{agt}) en utilisant les termes du paramètre. Pour reprendre notre exemple, si la classe « *Size* » est définie par quatre instances « *Tiny* », « *Small* », « *Medium* » et « *Big* », nous devons chercher parmi les quatre ensembles de mots $\{Tiny, Red, Cube\}$, $\{Small, Red, Cube\}$, $\{Medium, Red, Cube\}$ et $\{Big, Red, Cube\}$ lequel maximise le score ctx_{agt} (pour les besoins de l'exemple, nous supposerons que l'ensemble maximal est $\{Small, Red, Cube\}$ avec un score de 3).

Formellement, soit $r \in R$ une relation du MRC. Nous noterons $domain(r)$ le domaine de la relation et $range(r)$ son codomaine. En reprenant la notation du chapitre 3 de l'ensemble L (ensemble des arêtes triplets de relation entre concepts), nous noterons $\{c_1, r, c_2\} \in L$ lorsque les concepts $c_1 \in domain(r)$ et $c_2 \in range(r)$ sont reliés par la relation r . Soit $\tau(S)$ la modélisation logique d'une commande et $rel(e_1, ..., e_n) \in \tau(S)$ un prédicat appartenant à cette relation. Soit la fonction $maxctx(rel, W)$ qui calcule le score maximal de contexte que l'on peut obtenir avec la relation rel sur l'ensemble de mots W définie de la façon suivante :

$$maxctx(rel, W) = \max_{x \in domain(rel)} ctx_{agt}(\{x\} \cup W)$$

Dans notre exemple, nous avons $maxctx(larger, \{red, cube\}) = 3$.

Nous pouvons maintenant définir la fonction de transformation $trans(rel(E))$ avec $E = \{e_1, ..., e_n\}$ qui transforme un prédicat de la commande en un ensemble de concepts du MRC :

$trans(rel(e_1, ..., e_n)) =$

$$\begin{cases} \{c\} & \text{si } (n = 1) \wedge (\{e_1, r, c\} \in L) \\ \{c | \langle x, r, c \rangle \in \mathcal{O}, ctx_{agt}(\{x\} \cup E) = maxctx(rel, E)\} & \text{sinon} \end{cases}$$

Cette réduction considère les triplets $\{sujet, predicat, object\}$ du MRC. L'identification nous ayant permis de déduire le sujet de la relation, l'ensemble résultat correspond à tous les objets ayant le sujet déduit et la relation en prédicat. (sur notre exemple, ce sont les triplets $\{small, larger, medium\}$ et $\{small, larger, big\}$ qui permettent de déduire le résultat obtenu « medium » et « big »).

Cette fonction *trans* est un prerequis important pour le gestionnaire de réponse du chapitre 2. En effet, cette fonction, en interprétant les relations, permet de transformer le modèle de la commande comprenant des références relatives à un modèle ne contenant que des références absolues, tel que sont construites les capacités de l'agent. Il permet donc d'améliorer grandement les chances de calculer correctement l'appariement entre la requête et une capacité.

Si la section 4.1 a permis de développer les entrées du système d'interaction de notre système, il est aussi nécessaire d'expliquer comment l'agent traite les sorties formelles du système pour produire des phrases en langue naturelle à l'utilisateur. La section suivante présente ainsi le module de génération de langue naturelle que nous avons utilisé, basé sur la sortie du gestionnaire de réponses présenté chapitre 2.

4.2 Générateur des réponses en langue naturelle

La génération des réponses en langue naturelle comprend plusieurs parties indépendantes. Tout d'abord, il faut traduire les propositions de capacités données par le système. Dans la même idée, il faut aussi traduire une précondition fausse renvoyée par l'agent pour expliquer ses erreurs. Ces deux problèmes de génération font en fait partie du même problème, celui de la transformation d'un arbre VDL en langue naturelle. Cette question sera traitée par la section 4.2.1.

Ensuite, il faut tenir compte du performatif renvoyé par le système pour générer le bloc de réponses (performatif compris dans l'ensemble *clarify, suggest,* etc. voir chapitre 2). Nous expliquerons ainsi en section 4.2.2 comment nous tenons compte des messages reçus (avec leur performatif et leur contenu) pour générer la réponse finale en langue naturelle du système.

4.2.1 Transformation des arbres VDL en langue naturelle

Le principe de base de notre générateur d'anglais à partir d'un arbre VDL est d'analyser récursivement l'arbre. Nous noterons $\oplus$ l'opérateur de concaténation de chaînes de caractères. Soit Υ l'ensemble des arbres VDL possibles. De plus, soit $\mathcal{P}(E)$ l'ensemble des sous-ensembles possible de E. Pour un nœud de l'arbre $node \in \Upsilon$ donné, nous noterons :

- $tag(node) : \Upsilon \rightarrow St$, la fonction renvoyant l'étiquette XML de l'arbre $node$.
- $att(node) : \Upsilon \rightarrow \mathcal{P}(St \times St)$, la fonction renvoyant la liste des attributs du nœud. Pour une entrée $a \in att(node)$, nous noterons $a.key$ l'accès à la clef et $a.val$ l'accès à la valeur associée à cette clef.
- $content(node) : \Upsilon \rightarrow St$, la fonction renvoyant le contenu texte du nœud $node$. Si ce contenu n'existe pas, alors la fonction renvoie la chaîne vide $\surd$.
- $children(node) : \Upsilon \rightarrow \mathcal{P}(\Upsilon)$, la fonction renvoyant les fils de l'arbre $node$.

La fonction $nlgen_{VDL} : \Upsilon \rightarrow St$ qui traduit un nœud VDL en une chaîne de caractère est alors définie par :

$$
\begin{aligned}
nlgen_{VDL}(n) &= tag(n) \\
&\oplus \left\{\oplus_{a \in att(n)} [a.key \oplus \text{ " is " } \oplus a.val]\right\} \\
&\oplus \begin{cases} \text{" is " } \oplus content(n) & Si\ content(n) \neq \surd \\ \{\oplus_{n' \in children(n)} nlgen_{VDL}(n')\} & Sinon \end{cases}
\end{aligned}
$$

Par exemple, l'arbre VDL suivant :

```
<take position="out">
  <shape>square</shape>
</take>
```

sera traduit par :

« *take position is out shape is square* »

Le traitement des préconditions est légèrement plus complexe, dans la mesure où une précondition contient un certain nombre de mots-clefs spécifique du langage VDL. De plus, pour

générer une précondition en langue naturelle, nous avons besoin de connaître sa sémantique logique (*i.e.* si elle doit être générée comme une précondition fausse ou une précondition vraie). Autrement dit, nous considérons une version légèrement modifiée de la fonction $nlgen_{VDL}(n)$ avec un paramètre booléen en plus : $nlgen_{VDL}(n, b) : \Upsilon \times \mathbb{B} \rightarrow St$. Nous noterons $\mathbb{B} = \{\top, \bot\}$ tel que $\top$ est la sémantique « vraie » et $\bot$ la sémantique « faux ».

La définition précise de ces règles pour chaque mot-clef ne présente pas d'intérêt conceptuel particulier et sera donc simplement donnée en Annexe C page 217. Pour présenter un exemple, supposons que la précondition suivante soit une précondition fausse fournie par la fonction $np(e)$:

```
<guard>
  <not>
    <is-a>
      <get><in-hand/></get>
      <nothing/>
    </is-a>
  </not>
</guard>
```

Alors, cette précondition sera traduite par :

« the content of in-hand is not nothing »

Le résultat n'est pas syntaxiquement correct, mais suffisant pour que les utilisateurs comprennent les propositions du système dans notre évaluation. Il est possible d'améliorer les résultats avec les travaux de générateurs basés sur XML [Bateman, 1997].

4.2.2 Interprétation des performatifs en langue naturelle

La génération en langue naturelle au niveau des performatifs est gérée par des patterns dépendants du type de performatifs. Nous réutiliserons les notations du chapitre 2, tel que A_{req} sera le modèle de la requête, $\mathcal{E}(A_{req})$ l'ensemble des capacités possibles, $\mathcal{F}(A_{req})$ l'ensemble des capacités impossibles et $np(f)$ pour $f \in \mathcal{F}(A_{req})$ l'ensemble des préconditions empêchant l'exécution de la capacité f. Ainsi, nous utilisons les patterns suivants :

Performatif « agree » Ce performatif ne possède pas de paramètres. Il valide la bonne compréhension et exécution de la requête de l'utilisateur. En ce sens, la génération de l'anglais est une simple formule de validation telle que « Yes. Something else ? » ou « Performed. ».

Performatif « clarify » Ce performatif reçoit en paramètre l'ensemble $\mathcal{E}(A_{req})$. L'affichage dépend de la cardinalité de $\mathcal{E}(A_{req})$. En effet, si $\mathcal{E}(A_{req})$ ne contient qu'une seule capacité, alors ce message s'interprète comme une demande de validation pour l'utilisateur. Sinon, c'est une demande de clarification entre plusieurs propositions. Soit e_1 l'évènement unique de l'ensemble $\mathcal{E}(A_{req})$ si $|\mathcal{E}(A_{req})| = 1$, le système génère alors la sortie :

- Si $|\mathcal{E}(A_{req})| = 1$:

  "Do you mean " $\oplus\, nlgen_{VDL}(e_1) \oplus$ " ?"

- Si $|\mathcal{E}(A_{req})| > 1$

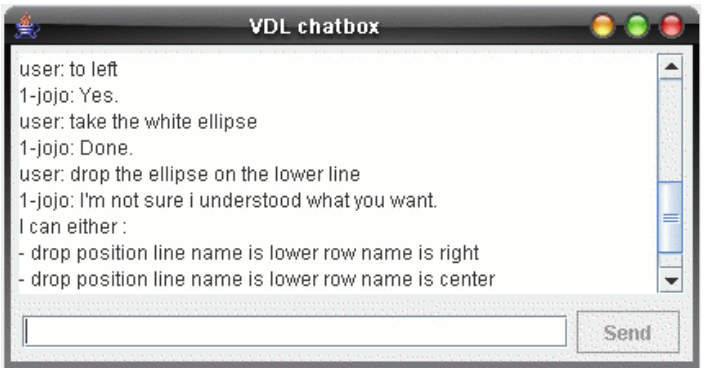

FIG. 4.5 – Exemple de génération d'anglais par notre système. L'exemple correspond à celui présenté dans la section 2.4 page 108.

$$\text{"I'm not sure I understood what you want. I can either :"}$$
$$\oplus_{e \in \mathcal{E}(A_{req})} \{\text{"- "} \oplus nlgen_{VDL}(e)\}$$

Performatif « impossible » Ce performatif reçoit en paramètre l'ensemble $\mathcal{F}(A_{req})$ et les préconditions fausses associées. L'agent répond à l'utilisateur que sa commande est impossible et affiche la liste des préconditions fausses :

$$\text{"Your command is currently impossible, because :"}$$
$$\oplus_{f \in \mathcal{F}(A_{req})} \left\{\oplus_{p \in np(f)} \{\text{"- "} \oplus nlgen_{VDL}(p, \bot)\}\right\}$$

Performatif « suggest » Ce performatif reçoit en paramètre l'ensemble $\mathcal{E}(A_{req})$, l'ensemble $\mathcal{F}(A_{req})$ et les préconditions fausses associées. L'affichage est assez proche d'une simple fusion des deux affichages précédents, à la différence près que l'ensemble $\mathcal{E}(A_{req})$ est traité comme un ensemble de propositions spontanées du système, alors que dans le performatif *clarify* il est traité comme un ensemble de propositions de traduction de la requête. Le système génère alors :

$$\text{"Your command is currently impossible, because :"}$$
$$\oplus_{f \in \mathcal{F}(A_{req})} \left\{\oplus_{p \in np(f)} \{\text{"- "} \oplus nlgen_{VDL}(p, \bot)\}\right\}$$

puis :

- Si $|\mathcal{E}(A_{req})| = 1$:
$$\text{"Therefore, I can "} \oplus nlgen_{VDL}(e_1)$$
- Si $|\mathcal{E}(A_{req})| > 1$
$$\text{"However, I suggest you the following requests :"}$$
$$\oplus_{e \in \mathcal{E}(A_{req})} \{\text{"- "} \oplus nlgen_{VDL}(e)\}$$

Performatif « not-understood » Ce performatif ne reçoit pas de paramètre. Il signifie que la requête ne correspond à rien de connu pour l'agent. L'agent signifie simplement sa non-compréhension de la requête : « Your request cannot be understood ».

Un exemple d'utilisation de ces patterns est présenté sur la figure 4.5.

4.3 Évaluation préliminaire

Cette section présente l'évaluation préliminaire que nous avons faite de ce système. En particulier, nous expliquerons d'abord pourquoi elle n'a que le statut de préliminaire, puis nous expliquerons le protocole mis en place. Enfin, nous discuterons des résultats.

4.3.1 Hypothèses et statut d'évaluation préliminaire

Évaluer un système d'interaction humain-agent n'est pas une tâche simple. En effet, par essence, l'évaluation ne peut pas être automatique, car l'un des deux interlocuteurs est un utilisateur ordinaire. De plus, il faut un nombre suffisant d'utilisateurs pour pouvoir considérer des résultats comme significatifs.

Dans le cadre de ce travail, nous avons effectué début 2006 une évaluation de la version humain-agent du système [Mazuel et Sabouret, 2008a]. Certaines des fonctionnalités du système présentées jusqu'à présent n'existaient pas encore :

- La mesure de distance sémantique. Dans cette évaluation, elle était binaire $\{0, 1\}$ et ne gérait que la synonymie entre concepts.
- Le principe de l'arbre de dépendance de la capacité et du modèle de la requête. La formule de base entre nuages de concepts (donc sans tenir compte des dépendances entre les termes) était alors utilisée. Ainsi, le modèle de prédicats présenté section 4.1.2 et section 4.1.3 n'existait pas.

A l'inverse, la plupart des caractéristiques essentielles de notre approche étaient déjà présentes au moment de cette évaluation :

- Le système de gestion des réponses avec les performatifs *clarify, suggest,* etc. En particulier, la génération d'anglais selon les performatifs et les capacités présentes ;
- Le principe de la stratégie de sélection en fonction d'un appariement entre capacité et modèle de la commande ;
- La génération automatique de capacités à partir d'une analyse dynamique du code de l'agent.

En résumé, cette évaluation avait pour rôle de justifier le cœur de notre système, à savoir le module de traitement de l'hétérogénéité sémantique par une stratégie de sélection basée sur la génération automatique de capacités. En ce sens, nous pensons que l'absence de mesure sémantique et d'arbre de dépendance ne nuit pas à l'utilité et à la pertinence de cette évaluation, mais que ces deux facteurs jouent néanmoins sur la qualité des résultats (et c'est effectivement la conclusion qui est apparue). C'est pourquoi nous présentons ici néanmoins cette évaluation, mais en la qualifiant de « préliminaire » car elle n'évalue pas le système dans l'état exact dans lequel il a été présenté jusqu'ici.

La section suivante présente le protocole donné aux utilisateurs participant à l'évaluation et le contexte agent utilisé.

4.3.2 Protocole

Notre expérience a été faite avec l'agent simple appelé Jojo[10] que nous déjà présenté dans la section 2.4. Pour rappel, cet agent est inspiré du monde de cube de Winograd

[10] Vous pouvez essayer Jojo sur la page demo : `http://www-poleia.lip6.fr/~sabouret/demos`.

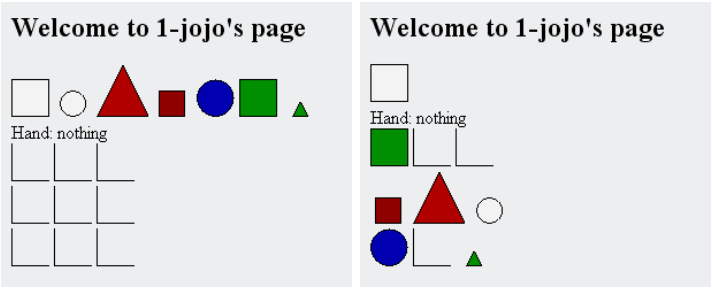

FIG. 4.6 – État initial et final du protocole

[Winograd, 1972]. De plus, il possède deux actions : « prendre un objet » et « poser un objet » sur une « grille ». Un objet est caractérisé par sa forme ($shape \in \{square, triangle, circle\}$), sa couleur ($color \in \{red, green, blue, white\}$) et sa taille ($size \in \{tiny, small, medium, big\}$). Une position est un couple parmi le produit cartésien des ensembles $\{upper, center, lower\} \times \{right, middle, left\}$.

Douze personnes ont fait l'expérience. Aucune de ces personnes n'avait utilisée le système auparavant. Ils n'avaient aucune information sur les capacités en Traitement Automatique des Langues du système. L'objectif d'une personne était d'atteindre un état particulier de l'environnement de l'agent (figure 4.6), sans limitation de temps. Après l'expérience, un questionnaire permettait aux sujets de l'expérience de noter leurs impressions, commentaires libres et donner une note entre 1 et 10 sur certains critères prédéfinis (tel que le degré de compréhension estimé de l'agent, ou sa pertinence dans ses réponses).

Les résultats attendus étaient les suivants :

1. Le système doit permettre une interaction plus souple car il est capable de faire des propositions à l'utilisateur si sa commande est imprécise.
2. Le système peut fournir des explications sur les commandes impossibles, afin d'aiguiller l'utilisateur sur les problèmes potentiels de formulation.

4.3.3 Résultats principaux

Toutes les personnes ont terminé la tâche. Le temps moyen mis pour la réaliser est de 912 secondes (avec un écart moyen de 138). La note moyenne sur 10 attribué est de $8,25$ (avec un écart moyen de 1). Il apparaît que le retour à l'utilisateur et les propositions faites par l'agent sont les points les plus importants d'après l'évaluation des questionnaires.

Une analyse plus approfondie des traces des interactions lors de l'expérience amène aux conclusions suivantes :

1. La nécessité pour l'utilisateur de savoir ce qu'attend l'agent, ce qu'il ne peut pas faire. Par exemple, lorsque l'utilisateur dit *"drop on the lower line"*, le système propose la liste des cases vides en bas de la grille (sauf lorsqu'il n'en reste qu'une). D'un point de vue initiative mixte, l'utilisateur sait, au besoin uniquement, ce que le système attend de lui comme information complémentaire pour pouvoir traiter sa commande.
2. La nécessité pour l'utilisateur de savoir pourquoi il ne peut pas le faire. C'est ce qui permet de corriger l'état de l'agent par rapport au souhait de l'utilisateur. Par exemple :

la commande *"Take the red figure"*, si la main est déjà pleine, est gérée par une réponse du type *"I can't because the hand is not empty"*. En réponse du retour contextualisé sur l'état de l'agent, les utilisateurs choisissent quasi systématiquement la proposition du système.

3. Dans le cadre de l'interaction humain-agent, les mesures sémantiques entres concepts sont très importantes. En effet, souvent un utilisateur utilise un terme plutôt qu'un autre, où propose une requête réalisable, mais énoncée avec des termes qui ne sont pas ceux de l'agent. Par exemple, un utilisateur parlera de « cube » pour désigner le concept agent « carré ». Ainsi, l'étude des traces montre que les mesures de similarité ne suffisent pas à modéliser les concepts dans un système d'interaction en langue naturelle. Cette constatation nous a permis de confirmer l'hypothèse que nous avions faite initialement sur la nécessité d'utiliser une mesure de degré de relation sémantique (chapitre 3).

4.4 Conclusion

Dans ce chapitre, nous avons proposé un algorithme de modélisation d'une commande et une méthode d'interprétation des relations qui utilise les capacités d'introspection de l'agent pour résoudre les références de contexte. La commande est modélisée sous la forme d'un ensemble de propositions logiques qui s'appuie sur l'utilisation du MRC de l'agent. 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 actuels 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 présentée repose sur un algorithme de similarité sémantique basé sur WordNet.

La génération des réponses en langue naturelle présentée est dépendante uniquement de la sémantique opérationnelle du langage agent VDL. En ce sens, tout agent VDL peut prétendre communiquer avec l'utilisateur. Néanmoins, le niveau syntaxique de ces réponses est encore à l'heure actuelle faible. Nous travaillons actuellement sur plusieurs heuristiques basées sur la catégorie syntaxique des termes présents dans le code VDL, ainsi que sur la structure hiérarchique du MRC. Par exemple, en anglais, un adjectif apparaît toujours avant le nom. Si un nœud possède un seul fils qui est un nom et que ses « frères » sont des adjectifs, alors nous pouvons penser qu'ils sont reliés. De même, si dans un arbre VDL deux nœuds fils sont subsumés dans le MRC, alors seul le nœud le plus spécifique apporte de l'information. Ces heuristiques permettent par exemple de remplacer la génération « take object shape is circle size is medium » par la phrase « take medium circle », qui est plus fluide. Nous ne présentons pas ce travail ici car il est complémentaire à cette thèse et trop récent pour que les heuristiques proposées puissent être validées, même sur la théorie.[11]

L'évaluation préliminaire de notre système présente des résultats encourageants pour la validation de l'approche humain-agent. Cependant, nous voudrions la valider sur le système

[11] Nous remercions à ce sujet les étudiants de Master Fabrice Mongkhoune et Ravaka Razafimanantsoa qui ont contribué à la réflexion et aux tests préliminaires de ces heuristiques.

actuel avec d'autres agents et MRC que ceux que nous avons utilisés jusqu'à présent, afin de montrer la portabilité de notre approche sur différents contextes.

Chapitre 5

Application à la communication agent-agent

Sommaire

Dans ce chapitre, nous proposons une solution pour l'interprétation de commandes entre deux agents dans un cadre sémantiquement hétérogènes. Notre protocole peut être vu comme une extension de FIPA-request[1] qui se focaliserait plus sur les messages non compris (*not-understood*) que sur l'engagement et la réalisation de l'action, telle que détaillés dans le protocole initial. Pour cela, à l'instar de [Laera *et al.*, 2007], nous nous appuierons sur un service d'alignement d'ontologies supposé accessible et directement utilisable par les agents (figure 5.1). Ceci nous permet la traduction, même partielle, de la requête de l'agent initiateur en utilisant les termes de l'ontologie de l'agent récepteur. Néanmoins, nous ne ferons pas d'hypothèses sur la qualité de l'alignement fourni par ce service, en admettant donc qu'il puisse être imparfait et/ou incomplet. Une fois la traduction effectuée, l'agent récepteur évalue son degré de compréhension de la requête traduite et répond à l'agent expéditeur en

[1] `http://www.fipa.org/specs/fipa00026/`

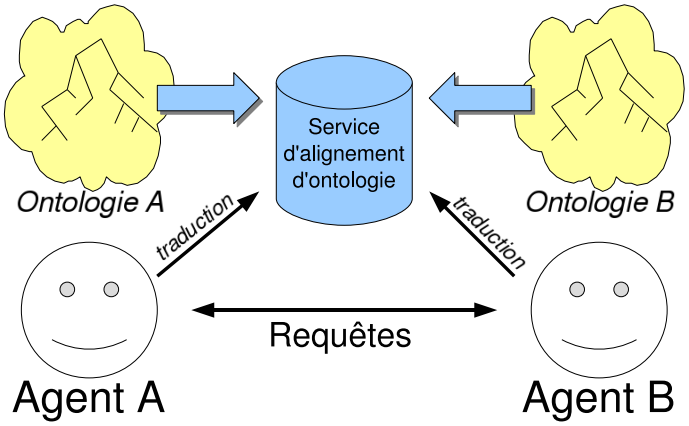

FIG. 5.1 – Traduction de requêtes par un service d'alignement d'ontologies. Les agents s'envoient des requêtes qu'ils interprètent en utilisant dynamiquement ce service.

utilisant le système décrit dans le chapitre 2 en fonction de son niveau de compréhension de la requête. Un protocole d'échange doit ainsi être décrit pour encadrer l'ensemble de ces interactions et des réactions de deux agents (alors que ce n'était pas nécessaire dans le cadre humain-agent, l'humain gérant seul son choix devant les propositions du système).

La section 5.1 présente le module de traduction des requêtes (correspondant au module de normalisation des entrées/sorties présenté dans le chapitre 2, et plus particulièrement la figure 2.3 page 97). La section 5.2 présente le protocole de communication et d'échange mis en place entre les deux agents pour comprendre la requête initiale. La section 5.3 présente un exemple simple d'application sur un échange entre deux agents, puis finalement la section 5.4 décrit l'évaluation faite de ce système.

5.1 Traduction des requêtes par alignement

Dans cette section, nous présentons le module de traduction des requêtes que nous considérons. Dans une première partie, nous détaillerons les notations utilisées pour formaliser la notion d'alignement. Ensuite, nous présenterons le principe de la traduction de requête en elle-même. Enfin, nous terminerons par la description de la fonction de qualité $Q(R)$.

5.1.1 Formalisation d'un alignement

Nous nous plaçons dans un modèle de système multi-agents ouvert, où chaque agent possède sa propre ontologie. Dans notre modèle, l'ontologie est une hiérarchie augmentée de relations et se modélise sous la forme d'un tuple $\mathcal{TA} = \langle C, R, L, \mathcal{F}, \mathcal{G} \rangle$ (voir section 3.1). L'ensemble C représente l'ensemble des concepts de l'ontologie, R l'ensemble des relations non hiérarchiques entre les concepts et L l'ensemble des liens entre les concepts.

Dans la suite de ce chapitre, nous noterons $\mathcal{P}(E)$ l'ensemble des sous-ensembles de l'ensemble E. Pour identifier l'appartenance d'une ontologie $\mathcal{TA}$ à un agent donné A, nous noterons le nom de l'agent en exposant. Par exemple, l'ensemble des concepts de l'ontologie A sera noté C^A.

Nous pouvons ainsi définir la notion d'*alignement* entre deux concepts de deux ontologies différentes $\mathcal{TA}$ et $\mathcal{TA}'$ comme un tuple $m = \langle e, e', n, \mathcal{M} \rangle$, tel que :

- $e \in \mathcal{TA}$ et $e' \in \mathcal{TA}'$ sont des entités (concept ou relation) des deux ontologies telles qu'une correspondance existe,
- $\mathcal{M}$ est la relation de correspondance (*e.g.* équivalence, subsomption, etc.) entre e et e',
- $n \in [0, 1]$ est le degré de validité de cette correspondance[2].

Nous noterons $M(\mathcal{TA}, \mathcal{TA}')$ l'ensemble des mappings entre l'ontologie $\mathcal{TA}$ et l'ontologie $\mathcal{TA}'$. Par extension, si S est un ensemble d'entités (concept ou relation) de l'ontologie $\mathcal{TA}$, alors nous noterons $M(S, \mathcal{TA}')$ l'ensemble des mappings correspondant aux entités S de l'ontologie $\mathcal{TA}$ dans l'ontologie $\mathcal{TA}'$.

Par exemple, dans le cas de deux ontologies sur la description de données bibliographiques, il est possible d'avoir un alignement du type $m = \langle auteur, auteur\_article, 0.65, \equiv \rangle$ signifiant que le concept *auteur* de la première ontologie est considéré équivalent au concept *auteur_article* de la deuxième ontologie avec un degré de certitude de 0.65. De même, les alignements $\langle personne, auteur, 0.7, \equiv \rangle$ et $\langle article, publication, 0.45, \equiv \rangle$ sont aussi des alignements possibles.

5.1.2 Construction de la nouvelle requête

Lorsqu'un agent A envoie une requête formulée selon une ontologie $\mathcal{TA}^A$ à un agent B qui utilise l'ontologie $\mathcal{TA}^B$, l'agent B doit tout d'abord traduire cette requête. Nous avons choisi de considérer (comme d'autres travaux, tels que [Laera *et al.*, 2007], section 1.4.5.1), que le SMA a accès à un service d'alignement d'ontologies. Dans cette section, nous expliquons comment l'agent B utilise ce service d'alignement d'ontologies pour traduire le contenu de la requête de l'agent A dans les termes de son ontologie $\mathcal{TA}^B$.

Soit S_A le contenu du message envoyé par un agent A à un agent B. Par construction, $S_A \subset \mathcal{P}(\mathcal{TA}^A)$ (*i.e.* S_A est un ensemble de concepts de l'ontologie de A). Le service d'alignement construit alors un ensemble de mappings $M(S_A, \mathcal{TA}^B)$. L'ensemble S_B, traduction de S_A dans $\mathcal{TA}^B$, est donc défini comme l'ensemble des concepts de $\mathcal{TA}^B$ tel qu'il existe un alignement $m \in M(S_A, \mathcal{TA}^B)$ le reliant à l'un des concepts de S_A. Plus formellement, la requête traduite $S_B \subset S(\mathcal{TA}^B)$ est obtenue par[3] :

$$S_B = \left\{ c' \in \mathcal{TA}^B | \exists \left\langle c, c', n, \mathcal{M} \right\rangle \in M(S_A, \mathcal{TA}^B) \right\}$$

Cette formule correspond simplement à la réaffectation des termes de la requête initiale en les termes d'une nouvelle requête dans l'ontologie B. En pratique, dans la plupart des travaux actuels, cette opération est jugée réalisable, sans perte d'informations (la recherche de l'alignement constituant alors le point difficile à atteindre [Laera *et al.*, 2007, van Diggelen *et al.*, 2006b]). Or, nous pensons justement que cette hypothèse est trop forte.

[2] Notons que la présence de ce degré n'est pas obligatoire, notamment si le service d'alignement choisi ne le propose pas. Dans ce cas, nous considérerons que l'alignement est correct avec certitude et son score est alors de 1.

[3] Cette formule ne considère pas le cas où le service d'alignement peut fournir plusieurs propositions d'alignement possibles pour un concept de la requête avec des scores n et des types R différents. Si un choix est nécessaire, nous privilégierons d'abord les R correspondant à une équivalence, puis l'alignement ayant le meilleure score n.

En effet, certains concepts ne seront pas forcément définis dans les deux ontologies (et seront donc manquant dans S_B), ou ne seront pas définis au même niveau de spécialisation (et la traduction dans S_B sera alors trop général et peu exploitable). C'est pour résoudre ces problèmes que nous proposons d'utiliser, après l'étape de traduction, un protocole d'interaction adapté aux traductions incomplètes ou partielles.

5.1.3 Qualité de modélisation et fonction $Q(R)$

La présence du score n pour l'alignement entre deux concepts e et e' (voir section 5.1.1) permet de considérer la qualité de cet alignement. Au niveau d'une traduction complète de requête, ces approximations permettent d'estimer la qualité de traduction de cette requête. Autrement dit, ce score peut nous permettre de modéliser la fonction $Q(R)$ qui correspond à la qualité de modélisation d'une requête (section 2.3.1 page 102). En effet, si un score d'appariement est très bon (*i.e.* proche de 1), mais que le mapping utilisé est incertain, il est logique que le score d'appariement soit plus faible (afin que la requête soit considérée comme ambiguë pour le choix de la stratégie de réponse). Ainsi, nous définirons dans le cadre agent-agent la fonction $Q(R)$:

$$Q(R) = \frac{\sum_{\langle c',c,n,\mathcal{M}\rangle \in M(R,\mathcal{TA}^B)} n}{\left|M(R,\mathcal{TA}^B)\right|}$$

Autrement dit, $Q(R)$ représente ici la moyenne des scores d'alignements mis en jeux dans le mapping M. Sur l'exemple des trois mappings de la section 5.1.1, nous obtenons $Q(R) = 0,6$.

5.2 Protocole de communication

5.2.1 Structure des messages échangés

La structure des paramètres des messages que nous adoptons pour la communication entre les agents repose sur le modèle FIPA-ACL. L'identifiant d'exécution d'une commande sera modélisé par le paramètre *conversation-id*. Les interactions multiples rentrant dans le cadre de l'exécution de cette commande seront identifiées par les paramètres *reply-with* et *in-reply-to*. Le contenu du paramètre *performative* correspondra aux performatifs classiques *request, agree,* etc. ainsi qu'aux nouveaux performatifs qui ont été décrits dans le chapitre 2.

5.2.2 Description du protocole

Dans le cadre de l'interaction humain-agent, l'agent émetteur est un être humain et c'est lui qui a la tâche de trouver la bonne stratégie de réponse en fonction du performatif répondu par l'agent récepteur (nous n'avions ainsi pas à gérer de protocole). Dans notre situation, l'agent est informatique et doit répondre seul aux propositions de l'agent récepteur. C'est pourquoi nous proposons le protocole de communication de la figure 5.2, formalisé en AUML 2.1[4], pour spécifier la réaction de l'agent émetteur.

[4] `http://www.auml.org/`

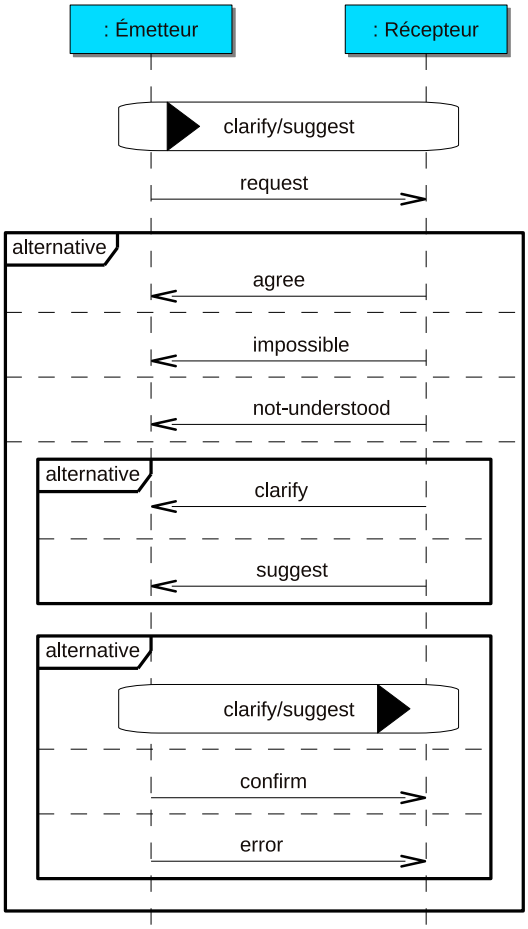

FIG. 5.2 – Protocole agent-agent de notre système au formalisme AUML 2.1

En pratique, la première partie du protocole est simple. L'agent émetteur envoie des requêtes (*i.e.* message dont le performatif est *request* et dont le contenu est un arbre VDL de requête). Ces requêtes sont traduites à la volée par le système de traduction présenté à la section précédente. La requête traduite passe alors par le système décrit dans le chapitre 2, et les réponses possibles de l'agent récepteur sont ainsi :

- *agree* : la requête a été comprise et acceptée ;
- *clarify* : la requête correspond a trop de capacités de l'agent ou l'interprétation a besoin d'être confirmée ;
- *suggest* : la requête a été reconnue comme impossible et l'agent récepteur fait des propositions proches parmi ce qu'il peut faire ;
- *impossible* : la requête a été reconnue comme impossible et l'agent récepteur n'a pas de solution proche à proposer ;
- *not-understood* : la requête n'a pas été comprise par le système.

L'agent émetteur reçoit ainsi l'un de ces 5 performatifs et doit réagir. Nous proposons les réactions suivantes :

- Si le performatif est *agree*, *impossible* ou *not-understood*, alors la conversation est terminée. Dans le cas *impossible,* l'agent émetteur reçoit de l'agent récepteur la liste des

préconditions violées rendant sa requête impossible. L'agent a ainsi les éléments nécessaires pour raisonner, comprendre le problème et proposer une nouvelle requête. Néanmoins, ce processus de raisonnement est spécifique du contexte de développement de l'agent et de son application. Dans le cadre de cette thèse, nous n'aborderons pas les processus cognitifs que peut mettre en place l'agent pour analyser ces messages et déduire le fonctionnement de l'autre agent.[5]

- Si le performatif est *clarify* ou *suggest,* alors l'agent émetteur reçoit la liste des propositions $\mathcal{E}(R)$ liées à sa requête R. Comme nous l'avons évoqué à la section précédente, nous ne considérons pas les préconditions impossibles pour l'instant. Le traitement du performatif *clarify* et du performatif *suggest* est donc le même actuellement. Ici, l'agent émetteur sait que sa requête est incluse dans l'ensemble $\mathcal{E}(R)$ qu'il vient de recevoir. L'agent cherche donc alors les capacités $C \subseteq \mathcal{E}(R)$ qui sont les plus proches de sa requête (en utilisant l'algorithme d'appariement requête/capacité présenté dans la section 103). Trois situations sont alors possibles :
  - Si $|C| = 1$, alors l'agent a réussi à trouver la capacité de $\mathcal{E}(R)$ qui correspond à sa requête. Dans ce cas, l'agent émetteur répond avec le performatif *confirm* en précisant la capacité qu'il a jugé comme étant la bonne capacité.
  - Si $|C| = |\mathcal{E}(R)|$, alors la conversation est un échec (*i.e.* l'agent émetteur n'est pas capable de réduire l'ensemble des capacités candidates). L'agent émetteur prévient l'agent récepteur que la conversation est un échec avec le performatif *error.*
  - Si $|C| < |\mathcal{E}(R)|$, alors l'agent n'as pas pu se décider, mais il a réussi à réduire l'ensemble des capacités candidates. La réaction exacte dépend de l'agent et de sa programmation. Il peut décider d'abandonner la conversation (avec le performatif *error*) ou de relancer une nouvelle requête sémantiquement équivalente à la première (mais avec des concepts différents, permettant de discriminer les capacités de C), en prévenant l'agent récepteur de sa réduction C. La conversion peut ainsi converger vers des réductions successives de l'ensemble C.

5.3 Exemple

Cette section présente un exemple simple d'application de la stratégie de communication précédemment décrite. Dans un premier temps, nous définirons un cadre pour 2 agents, leur environnement, puis la deuxième section présentera plus en détails l'interaction en elle-même.

5.3.1 Cadre de l'exemple

Supposons deux agents dans un monde de cube, du type [Winograd, 1972] (la figure 5.3 présente un exemple de monde de cube). L'agent A (resp. B) possède sa propre ontologie $\mathcal{O}^A$ (resp. $\mathcal{O}^B$) pour décrire ce monde. Supposons que les descriptions des agents de ce monde soient incomplètes, de tel sorte que :

[5] En particulier, ce sujet fait l'objet de la thèse de Shirley Hoet qui commence en octobre 2008 sous la direction de Nicolas Sabouret. Cette thèse met en œuvre des algorithmes d'apprentissages basés sur les performatifs et réactions des agents pour améliorer les interactions. Nous parlerons plus en détail de ce travail dans le dernier chapitre sur les perspectives de cette thèse, page 177.

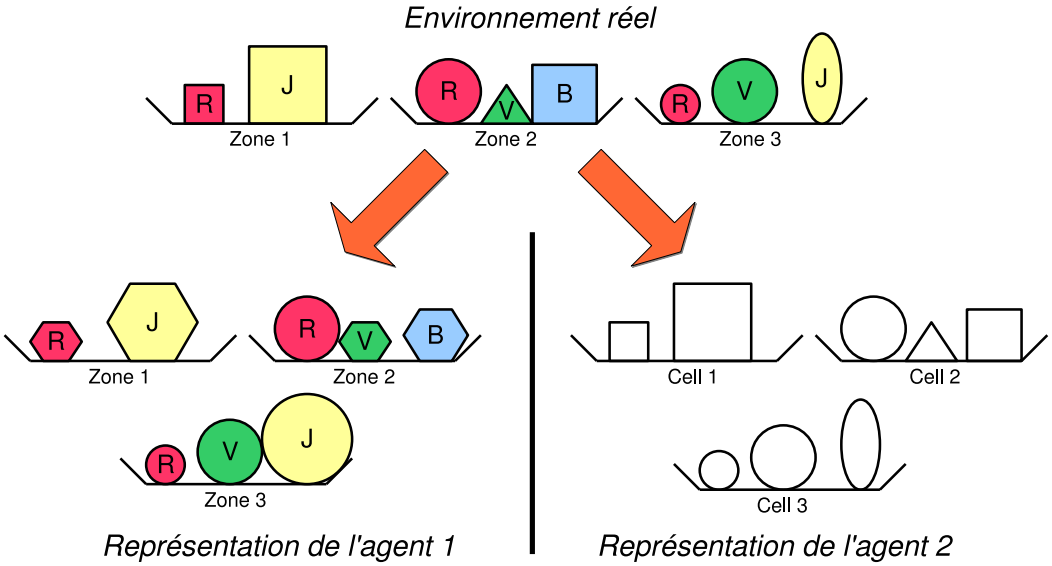

FIG. 5.3 – Exemple d'un modèle d'environnement avec formes géométriques. Le haut de l'exemple présente à titre d'illustration l'environnement tel qu'il est réellement, tandis que les deux schémas du bas présentent l'image que s'en font les agents. Les lettres dans les formes correspondent à différentes couleurs ($R \equiv rouge$, $B \equiv bleu$ et $J \equiv jaune$)

- L'ontologie $\mathcal{O}^A$ possède une description des tailles possibles pour les objets (*e.g.* « petit », « moyen », etc.), une représentation des couleurs (*e.g.* « rouge », « jaune », etc.), mais une représentation incomplète des formes sous deux types de classes uniquement, les « sphères » et les « polygones ».
- L'ontologie $\mathcal{O}^B$ possède de même une description des tailles d'objets, mais pas de représentation des couleurs. De plus, cette ontologie décrit beaucoup plus précisément les formes possibles de objets (*e.g.* « carré », « triangle », etc.)

De plus, ces deux agents peuvent situer les objets en fonction des trois zones géographique (« zone 1 », « zone 2 » et « zone 3 »). Avec ce type de modélisation, certains objets désignés explicitement (pour son contexte) par l'agent A (*e.g.* « le polygone vert »), deviennent ambiguë pour l'agent B. L'alignement entre les deux ontologies ne permet pas de résoudre le manque de couleurs dans le modèle de B. En effet, il n'existe pas dans le modèle de B d'équivalent pour la notion de « couleur ». Les concepts émis par l'agent A qui seront donc associés à la notion de « couleur » seront perdus. Or, si une requête repose sur cette notion pour se discriminer des autres dans A, elle devient alors ambiguë dans B.

La section suivante présente un exemple simple d'application de notre protocole sur ce problème. Le problème de communication peut ainsi se réduire dans ce cas précis à un calcul fait en fonction des capacités de l'agent B, comparativement à l'information partielle fournie par l'agent A.

5.3.2 Un exemple d'interaction

La figure 5.4 représente un exemple d'interaction possible de clarification en fonction du contexte d'exécution de capacités de l'agent B. Nous supposerons dans cet exemple simple

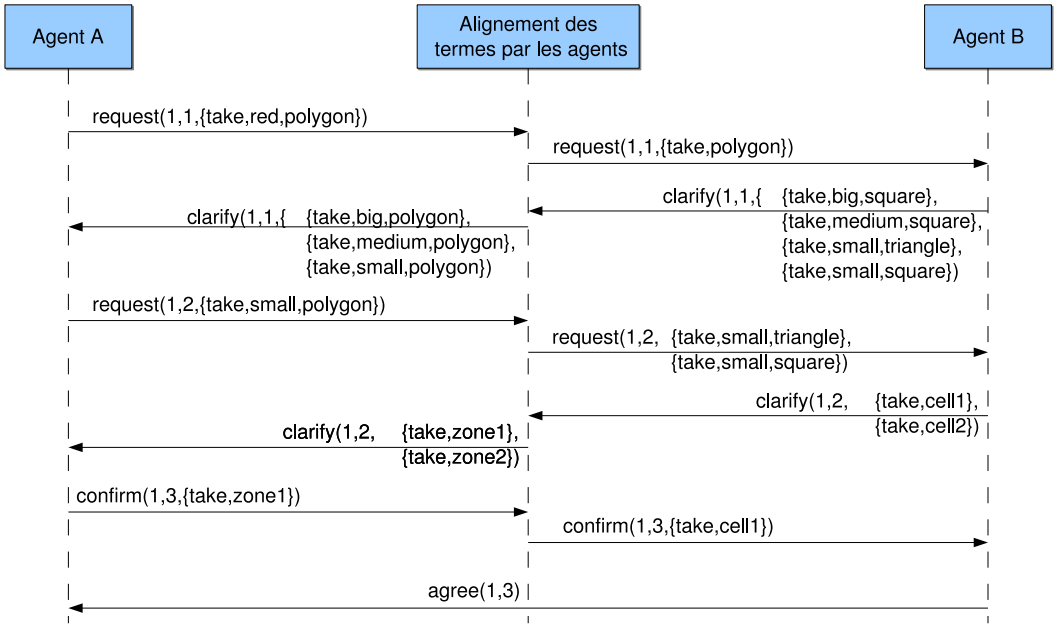

FIG. 5.4 – Exemple d'interaction selon notre stratégie d'interaction. La colonne centrale ne désigne pas un agent, mais représente la traduction opérée par les agents grâce au service d'alignement d'ontologie présent dans le système.

que pour toute requête R, alors $Q(R) = 1$.

Supposons que l'agent A envoie la requête $request(1, 1, \{take, red, polygon\})$, correspondant à la requête 1 de la conversation 1. Dans le contexte de l'agent A, cette requête ne pose aucune ambiguïté car il y a un unique polygone rouge. La requête est ici simplifiée en un ensemble de concepts $S_{A,1,1} = \{take, red, polygon\}$ de l'ontologie $\mathcal{O}^A$. Le service d'alignement d'ontologies calcule un alignement pour cette requête et transforme l'ensemble de concepts $S_{A,1,1}$ en un ensemble de concepts correspondants à l'ontologie $\mathcal{O}^B$: $S_{B,1,1} = \{take, polygon\}$.

L'agent B calcule alors ses capacités. Il en possède 8 de possibles (une par objet, voir figure 5.3). On a donc $|\mathcal{E}| = 8$. En considérant que $sim_{ONT}(polygon, square) = 0.5$ et $sim_{ONT}(polygon, triangle) = 0.5$ (en pratique, ce score est dépendant de la structure de l'ontologie), on a alors $|\mathcal{E}_{max}| = 4$, avec $p_{\mathcal{E}} = {}^{(1+0.5)}/_{2} = 0.75$. L'agent B utilise alors la stratégie 3, et donc le performatif *clarify* pour informer l'agent A de l'ambiguïté. Il envoie ainsi une réponse $clarify(1, 1, \{\{take, big, square\}, \{take, medium, square\}, \{take, small, triangle\}, \{take, small, square\}\}$. Ces 4 évènements sont sans ambiguïté pour l'agent B, car ils permettent de discriminer clairement les 4 capacités.

L'ontologie de l'agent A ne permettant pas de modéliser les formes aussi précisément, la réponse de B est traduite en $clarify(1, 1, \{\{take, big, polygon\}, \{take, medium, polygon\}, \{take, small, polygon\}\}$. Étant donné le contexte initial de l'agent A, la requête suivante de l'agent A précise la taille de l'objet : $request(1, 2, \{take, small, polygon\}$. Cette requête est retranscrite selon son équivalent avant la traduction, et correspond donc à la requête $clarify(1, 2, \{\{take, small, triangle\}, \{take, small, square\}\}$. Cette requête est encore ambiguë. L'agent B cherche alors un autre moyen de discriminer ces deux évènements et envoie la requête $clarify(1, 2, \{\{take, cell1\}, \{take, cell2\}\}$, en utilisant la position géographique qui

est discriminante dans cette situation. Cette dernière requête est ainsi traduite pour l'agent A par $clarify(1, 2, \{\{take, zone1\}, \{take, zone2\}\}$. Or, dans le contexte de A cette dernière requête suffit à confirmer l'objet prévu initialement (l'objet rouge est dans la zone 1) et l'agent répond $confirm(1, 3, \{take, zone1\})$. La requête est finalement acceptée par l'agent B (*i.e.* l'agent B envoie un performatif de confirmation $agree(1, 3)$).

Dans cet exemple, les agents ne communiquent pas systématiquement toutes les informations en leur possession pour décrire un objet. Cette situation est naturelle, puisqu'il n'est pas efficace qu'un agent envoie systématiquement l'intégralité de ces connaissances sur un objet si certaines informations peuvent suffire à le décrire. Certaines approches de la littérature utilisent pourtant ce type de méthode, mais cela introduit des lourdeurs de communication et d'interprétation difficiles à traiter (section 1.4.5.2).

5.4 Évaluation

Cette section présente l'évaluation que nous avons faite de la version agent-agent de notre système de traitement de l'hétérogénéité sémantique. Dans la section 5.4.1 nous donnerons le protocole expérimental que nous avons utilisé. Puis en section 5.4.2 nous discuterons des résultats.

5.4.1 Protocole

L'une des difficultés majeures des évaluations agent-agent réside dans la définition du SMA à considérer. En effet, il est nécessaire de définir des ontologies hétérogènes et un ensemble d'agents avec des capacités propres, les degrés d'hétérogénéités devant être quantifiables et la qualité de la communication mesurable, ce qui est très coûteux à définir. Afin de rendre possible une évaluation, nous testerons donc le protocole de communication présenté dans la section 5.2 en considérant donc un certain nombre d'hypothèses :

5.4.1.1 Construction des ontologies des agents

Nous noterons A l'agent émetteur et B l'agent récepteur. Nous noterons ainsi $\mathcal{O}^A$ et $\mathcal{O}^B$ les ontologies des deux agents. Ces modèles seront générés automatiquement selon la méthode suivante :

1. Les ontologies seront des arborescences de concepts.
2. Pour simplifier la structure des ontologies, nous supposerons qu'elle est à 2 niveaux uniquement. Le niveau 1 contient T concepts. Chacun des concepts du niveau 1 possède S concepts fils. Le niveau 2 contient ainsi globalement $T \times S$ concepts et chaque arborescence contient au final $1 + T + T \times S$ concepts.
3. Nous utilisons un paramètre P_{inter} pour définir la proportion de concepts qui seront dans l'intersection des deux ontologies $\mathcal{O}^A$ et $\mathcal{O}^B$. Par exemple, si $P_{inter} = 1$ les deux agents ont la même ontologie. Si $P_{inter} = 0.5$, alors les niveaux 1 partagent $\lfloor T/2 \rfloor$ concepts et $\lceil T/2 \rceil$ concepts sont différents. Si $P_{inter} = 0$, l'intersection des deux ontologies est nulle. Dans cette situation, il est impossible de résoudre le problème d'hétérogénéité sémantique, quelle que soit la méthode utilisée (section 1.4).

Pour construire les ontologies suivant les paramètres T, S et P_{inter} tout en respectant les définitions précédentes, nous utilisons l'algorithme suivant :

Algorithme 5.1 Construction des arborescences des agents A et B

Entrées: $T \in \mathbb{N}^*$, $S \in \mathbb{N}^*$, $P_{inter} \in [0, 1]$
Sorties: 2 arborescences de concepts

1: $nbCommunL1 \leftarrow \lfloor P_{inter} \times T \rfloor$
2: $nbDisjointL1 \leftarrow T - nbCommunL1$
3: créer 2 arborescences vides $\mathcal{O}^A$ et $\mathcal{O}^B$
4: ajouter une racine *Root* à $\mathcal{O}^A$ et $\mathcal{O}^B$
5: **pour tout** i tel que $1 \leq i \leq nbCommunL1$ **faire**
6: $L_i \leftarrow$ un label texte utilisant i
7: ajouter L_i en fils de *Root* dans $\mathcal{O}^A$ et $\mathcal{O}^B$
8: ajouter S fils à L_i dans $\mathcal{O}^A$ et $\mathcal{O}^B$
9: **fin pour**
10: **pour** $X \in \{A, B\}$ **faire** // Représente le nom de l'agent
11: **pour tout** i tel que $1 \leq i \leq nbDisjointL1$ **faire**
12: $L_i^X \leftarrow$ un label texte utilisant i spécifique à X
13: ajouter L_i^X en fils de *Root* dans $\mathcal{O}^X$
14: ajouter S fils à L_i^X dans $\mathcal{O}^X$
15: **fin pour**
16: **fin pour**
17: **renvoyer** $\mathcal{O}^A$ et $\mathcal{O}^B$

Ainsi, dans le cadre de cette génération d'ontologies, si un alignement existe, alors il est parfait. En effet, dans notre construction, soit nous utilisons le même label pour un concept dans les deux ontologies (ligne 7-8), soit nous utilisons un label totalement différent (ligne 13-14). Ainsi, les seuls alignements $m = \langle e, e', n, \mathcal{M} \rangle$ possible sont les alignements de la forme $m = \langle e, e, 1.0, \equiv \rangle$. L'une des conséquences directes est qu'un concept émis ne peut être interprété que de deux façons : soit il s'aligne parfaitement avec un concept de l'ontologie de destination, soit il n'existe aucune traduction possible. Cette hypothèse nous permet de tester notre système selon l'un des critères qui nous semble le plus important, à savoir sa capacité à s'adapter aux alignements incomplets.

5.4.1.2 Les capacités de l'agent récepteur

Les requêtes et les capacités seront des nuages des concepts. L'agent récepteur B possède N_{cap} capacités possibles composées de T concepts, telles que chacun des T concepts est un concept de niveau 2 ayant un père différent. Ceci nous permet de forcer une capacité à utiliser l'ensemble de l'ontologie. Par exemple, sur la figure 5.5, le nuage de concepts $\{L11\text{-}1, LA1\text{-}1\}$ est une capacité possible.

De plus, nous définissons un pourcentage P_{match} pour définir la proportion moyenne de concepts parmi les m concepts qui peuvent être communs entre deux capacités quelconques. Par exemple, si $P_{match} = 0$, alors chaque capacité a exactement m concepts différents. Il suffit donc d'un seul concept pour identifier de manière unique une capacité. Si $P_{match} = 0,5$, alors pour toute capacité il existe au moins une capacité qui partage $m/2$ concepts avec elles.

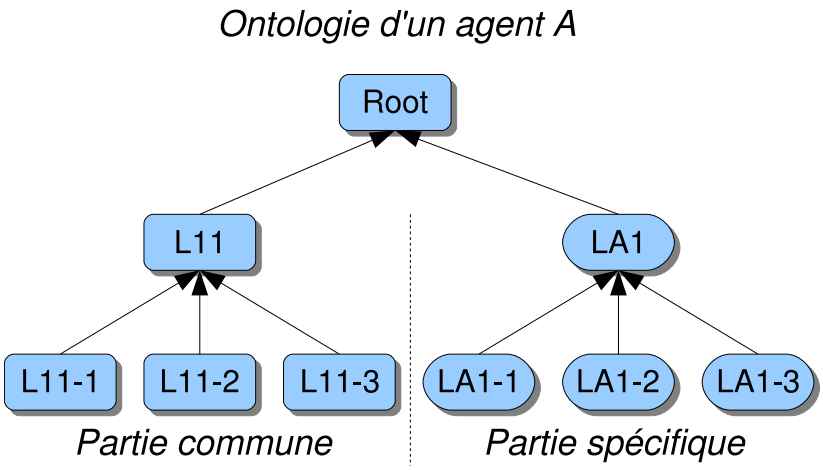

FIG. 5.5 – Exemple d'ontologie générée avec les paramètres $T = 2$, $S = 3$ et $P_{inter} = 0,5$.

Par construction, le cas $P_{match} = 1$ est impossible (toute les capacités seraient absolument identiques). La valeur maximale que nous testerons sera donc $P_{match} = 0,9$. Sur la figure 5.5 précédente, un score P_{match} de $0,5$ peut permettre de générer l'ensemble des quatre capacités $\{L11\text{-}1, LA1\text{-}1\}$, $\{L11\text{-}1, LA1\text{-}2\}$, $\{L11\text{-}2, LA1\text{-}2\}$ et $\{L11\text{-}2, LA1\text{-}1\}$.

5.4.1.3 Construction des requêtes

De manière équivalente aux capacités, les requêtes de l'agent A sont composées au maximum de T concepts, de telle sorte que chacun des T concepts soit un concept de niveau 2 ayant un père différent. Néanmoins, l'agent enverra des requêtes les plus discriminantes possibles selon son ontologie (*i.e.* possédant le moins de concept possibles pour être discriminant entre deux capacités). Par essence, la requête pourra donc contenir des concepts de la partie spécifique à l'agent A. Or, comme nous l'avons expliqué section 5.4.1.1, les alignements sont ici incomplets. Ceci impliquera qu'une requête non ambiguë pour l'agent A pourra le devenir pour l'agent B. C'est la résistance à cette propriété que nous cherchons à évaluer.

5.4.1.4 Instanciation des paramètres et critères surveillés

Nous avons ainsi la possibilité d'évaluer notre stratégie en faisant évoluer les paramètres T, S, N_{cap}, P_{inter} et P_{match}. Nous testerons les valeurs dans les plages $T \in [5, 40]$, $S \in [5, 30]$, $N_{cap} \in [5, 40]$, $P_{inter} \in [0, 1]$ et $P_{match} \in [0, 0.9]$. Ces valeurs couvrent un large éventail de situations :

- Des petites ontologies ($T = 10$ et $S = 5$, donc de 50 concepts) jusqu'aux grandes ontologies ($T = 40$ et $S = 30$, donc de 1200 concepts) avec des recouvrements variables ($P_{inter} \in [0, 1]$)
- Des agents possédant peu de capacités et telles que chaque capacité est formulée de manière unique (cas le plus simple, avec $N_{cap} = 5$ et $P_{match} = 0$) jusqu'aux agents possédant beaucoup de capacités avec beaucoup de concepts en commun entre ces capacités (cas le plus complexe, avec $N_{cap} = 40$ et $P_{match} = 0,9$).

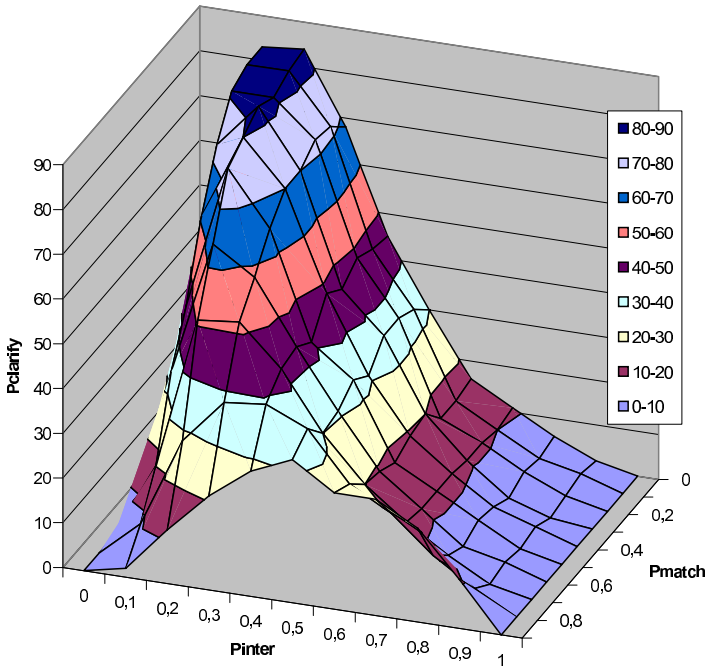

FIG. 5.6 – Diagramme 3D de la proportion $R_{clarify}$ (axe z) en fonction de l'évolution du paramètre P_{inter} (axe x) et du paramètre P_{match} (axe y) pour $T = 25$, $S = 15$ et $N_{cap} = 30$. Des coupes 2D représentants la répartition des trois proportions sont proposés sur la figure 5.7 pour $P_{inter} = 0,1$ et $P_{match} = 0,6$.

De même, pour chaque paramétrage de valeurs, nous pouvons générer aléatoirement un modèle consistant avec ces valeurs. Nous recommençons cette opération sur 1000 itérations pour avoir des résultats significatifs en moyenne.

Les résultats nous seront donnés par un calcul de proportion (en pourcentage) entre trois critères :

1. La proportion de requêtes directement traitées par l'agent récepteur (et donc n'ayant pas eu besoin de notre système). Nous noterons cette proportion R_{direct}.
2. La proportion de requêtes traitées grâce à notre système (et donc qui n'auraient pas pu être traitées normalement). Nous noterons cette proportion $R_{clarify}$.
3. La proportion de requêtes non traitées. De part la conception de cette évaluation, ces requêtes ne pouvaient pas être résolues, quel que soit le système utilisé. Nous noterons cette proportion $R_{impossible}$.

Ainsi, par définition nous aurons $R_{direct} + R_{clarify} + R_{impossible} = 100$.

5.4.2 Résultats

Tout d'abord, les résultats convergent pour un paramétrage donné. Autrement dit, les proportions décrites précédemment se figent en moyenne au bout de 600 itérations. Ceci montre que le modèle que nous utilisons pour notre protocole est cohérent et qu'il est possible d'interpréter les résultats obtenus.

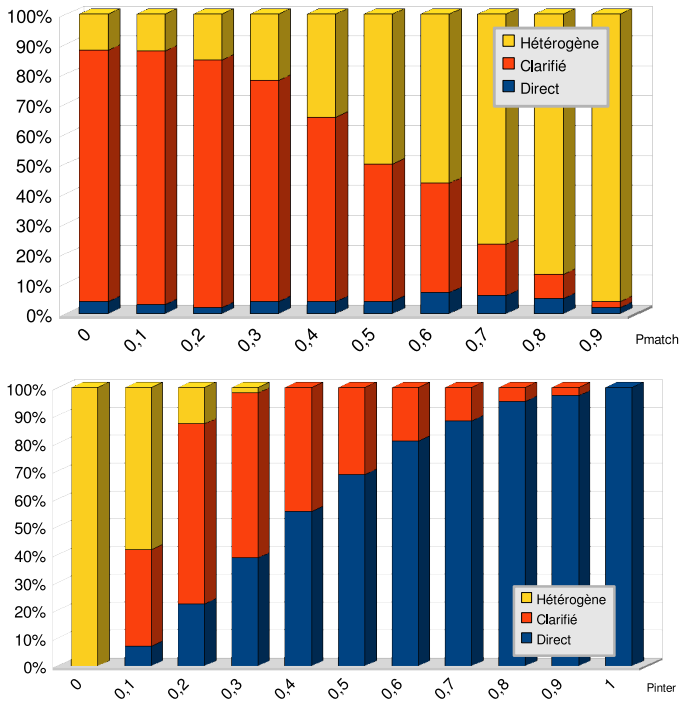

FIG. 5.7 – Répartition des trois proportions pour les paramètres $T = 25$, $S = 15$ et $N_{cap} = 30$. L'histogramme du haut présente l'évolution des proportions selon P_{match} pour $P_{inter} = 0,1$. L'histogramme du bas présente l'évolution des proportions selon P_{inter} pour $P_{match} = 0,6$. A noter que ces histogrammes sont des coupes 2D de la figure 5.6, mais complétées des autres proportions qui n'étaient pas représentées.

Les figures 5.6 et 5.7 illustrent les données recueillies. Nous pouvons d'abord discuter des résultats qui ne sont pas liés à notre approche (*i.e.* évolution de R_{direct} et $R_{impossible}$) pour mieux situer nos résultats par la suite :

- De manière attendue, le paramètre P_{inter} influe surtout sur la quantité de requêtes qu'il est possible de traiter directement (*i.e.* R_{direct}). En effet, dans le cas $P_{inter} = 0$, le problème est insoluble (*i.e.* $R_{impossible} = 100$ et $R_{clarify} = R_{direct} = 0$) et inversement dans le cas $P_{inter} = 1$ (*i.e.* $R_{direct} = 100$ et $R_{clarify} = R_{impossible} = 0$). L'évolution de R_{direct} entre ces deux valeurs extrêmes de P_{inter} est quasi-linéaire.
- Le paramètre $R_{impossible}$ présente une évolution plus complexe. Ce paramètre évolue suivant une combinaison de plusieurs facteurs. Ainsi, plus P_{inter} augmente et plus $R_{impossible}$ diminue, mais plus P_{match} augmente et plus $R_{impossible}$ augmente. Ceci s'explique par le fait que lorsque P_{inter} est faible, les ontologies des deux agents contiennent plus de concepts spécifiques. D'autre part, plus P_{match} augmente, plus les capacités ont de concepts en commun, donc plus il faut de concepts pour les discriminer. Ainsi, lorsque les capacités d'un même agent ont beaucoup de concepts en communs, mais qu'en plus les ontologies des deux agents ont peu de concepts partagés, l'hétérogénéité sémantique augmente (donc $R_{impossible}$ augmente) et les capacités sont difficilement discriminables.
- Le facteur N_{cap} augmente le nombre de capacités. Or, lorsque N_{cap} augmente, l'agent émetteur doit utiliser plus de concepts pour discriminer une unique capacité. Donc, la

requête émise contient en moyenne plus de concepts, et donc l'interaction est moins hétérogène (*i.e.* P_{direct} augmente).
- Finalement, les facteurs T et S n'influent sur le résultat qu'en fonction du paramètre P_{match}. En effet, si l'ontologie est grande (*i.e.* T et S grand) mais que les capacités possèdent beaucoup de concepts similaires (P_{match} grand), alors seule une sous-partie de l'ontologie est utilisée (ce qui a tendance à augmenter la difficulté d'interprétation).

Ces premières interprétations nous permettent de situer et d'interpréter la proportion $R_{clarify}$ de requêtes traitées par notre système :
- Notre système est particulièrement efficace lorsque les ontologies partagent peu d'informations (*i.e.* P_{inter} faible) et que les capacités sont peu semblables (*i.e.* P_{match} faible). Par exemple, pour $P_{inter} = 0,1$ et $P_{match} = 0,2$, notre système permet de résoudre 80% des requêtes (*i.e.* $R_{clarify} = 80$) alors qu'il n'y a que 2% des requêtes qui sont traitables directement (*i.e.* $R_{direct} = 2$, c.f. figure 5.6). D'autre part, le taux P_{match} n'agit sur nos résultats que parce que le problème devient plus difficile (*i.e.* $R_{impossible}$ augmente). En effet, le paramètre P_{match} n'a pas d'influence sur le taux de requête direct R_{direct} (c.f. figure 5.7).
- A l'inverse, notre système est moins utile pour certaines valeurs de paramètre, par exemple, si beaucoup de concepts sont utilisés dans la commande (dépendant du paramètre N_{cap}). De même, plus les ontologies sont semblables, moins notre système est utilisé (corollaire classique du problème d'hétérogénéité sémantique).

5.5 Conclusion

Dans ce chapitre, nous avons présenté le module agent-agent de notre système de traitement de l'hétérogénéité sémantique. Il repose sur la traduction des requêtes d'une ontologie à une autre en utilisant un service d'ontologie, puis sur un protocole de communication pour gérer automatiquement les performatifs générés par le système présenté au chapitre 5. Ce protocole d'interaction est défini directement au niveau des couches de communications, ce qui rend son utilisation transparente aux yeux des agents et permet une mise en place automatique.

L'évaluation montre que dans certaines situations (*i.e.* en fonction des structures des deux ontologies, nombres de capacités, degré moyen d'appariement entre capacités, etc.) notre système évalue plus de 85% des requêtes, contre 5% sans notre système. En particulier, notre stratégie de réponse est particulièrement efficace si les ontologies sont fortement différentes (avec un taux de concepts partagés entre 10% et 50%), ainsi qu'en fonction du degré d'appariement entre capacités (*i.e.* plus les capacités sont proches structurellement dans le modèle de l'agent émetteur, plus il est difficile de les discriminer pour l'agent récepteur).

Cette évaluation ne constitue qu'une première étape vers une validation complète de l'architecture. En particulier, le fait qu'elle soit générée automatiquement implique qu'il est difficile d'appréhender son impact réel sur la communication dans un système existant. De même, il n'était pas concevable de générer automatiquement des ontologies ayant des relations hétérogènes (et donc pouvant mettre à profit la mesure décrite chapitre 3). En effet, cette génération aurait impliqué des hypothèses fortes sur la sémantique de ces relations, ainsi qu'une génération automatique des poids T_X.

Chapitre 6

Codage et implémentation

Sommaire

Dans le cadre de cette thèse, l'ensemble des algorithmes proposés ont été implémentés en Java 1.6 et intégré dans la plate-forme VDL. En particulier, les algorithmes de similarité sémantique présentés dans le chapitre 3 ont donné lieu au développement d'une bibliothèque autonome. Nous présentons dans cette section le travail d'implémentation effectué.

6.1 La bibliothèque JSL : JavaSimLibrary

Dans cette section nous présentons la bibliothèque JSL[1] que nous avons développée dans le cadre de cette thèse. Cette bibliothèque nous permet de définir et d'utiliser les mesures sémantiques nécessaires à l'implémentation de notre système. Cette bibliothèque est intégralement écrite en Java et est composée d'environ 10000 lignes de code.

[1] `http://www-poleia.lip6.fr/~mazuel/jsl`

6.1.1 Besoin et bibliothèques existantes

Vis-à-vis de l'implémentation de la plate-forme VDL, nous avions besoin d'une bibliothèque pour la gestion de la similarité répondant aux propriétés suivantes :

- Une implémentation en Java ;
- Le calcul de scores de similarités/distances sémantique sur une ontologie en OWL ;
- Le calcul de scores de similarités/distances sémantique sur WordNet ;
- La possibilité d'ajouter facilement nos propres mesures, afin de développer la mesure présentée dans le chapitre 3.

Quelques librairies ont déjà été proposées sur ce sujet :

- La bibliothèque WordNetSimilarity[2] de Ted Pedersen est sans doute la solution de calcul sur WordNet la plus courante et la plus utilisée pour les évaluations et applications récentes (utilisé entre autres dans [Patwardhan et Pedersen, 2006] ou encore dans [Jarmasz et Szpakowicz, 2003]). Elle propose les mesures de Resnik, Lin, Jiang-Conrath, Leacock-Chodorow, Hirst-St.Onge, Wu-Palmer, Banerjee-Pedersen, et Patwardhan-Pedersen. Une version en ligne de cette bibliothèque est aussi accessible.[3] Néanmoins, pour nos besoins cette bibliothèque n'était que difficilement utilisable car elle ne fonctionne que sur WordNet et est définie en Perl.
- La bibliothèque de Nuno Seco [Seco *et al.*, 2004] en Java, téléchargeable à la rubrique « extension » du site officiel de WordNet[4]. Même si cette fois la bibliothèque est écrite en Java, le défaut principal reste encore la limitation à l'usage de WordNet.
- La bibliothèque SOQA-SimPack[5] de l'université de Zurich [Ziegler *et al.*, 2006] a elle aussi l'avantage d'être écrite en Java. Elle implémente par défaut les mesures de Jiang et Conrath, Lin et Resnik. De plus, SimPack accepte le modèle d'ontologie OWL (par l'API Jena[6]) et SOQA (non accessible librement néanmoins) permettant l'accès à des bases hétérogènes du type WordNet, etc. Malgré les avantages évidents, les premiers tests pratiques ont montré quelques lenteurs difficilement compatibles avec les besoins de réactivité nécessaire à une application d'interaction (trop de calcul à la volée, tel que la fonction IC). De plus, il n'existe aucune base conceptuelle réutilisable pour la définition de mesure sémantique, ce qui rend la définition de nouvelles mesures complexe.

Devant ses difficultés nous avons donc décidé de développer notre propre bibliothèque, basée sur les objectifs de réalisation suivant :

- Une bibliothèque complètement Java.
- La possibilité de définir les mesures sémantique *indépendamment de l'implémentation du modèle de représentation des connaissances.* Ceci afin de pouvoir utiliser l'implémentation d'une mesure donnée mesure aussi bien sur une ontologie OWL que sur WordNet.
- Un outillage pour définir rapidement une mesure sémantique, en fournissant directement un accès simple aux notions de base, telles que la quantité d'information IC d'un

[2] `http://www.d.umn.edu/~tpederse/similarity.html`
[3] `http://marimba.d.umn.edu/cgi-bin/similarity.cgi`
[4] `http://wordnet.princeton.edu/links`
[5] `http://www.ifi.unizh.ch/ddis/simpack.html`
[6] `http://jena.sourceforge.net/`

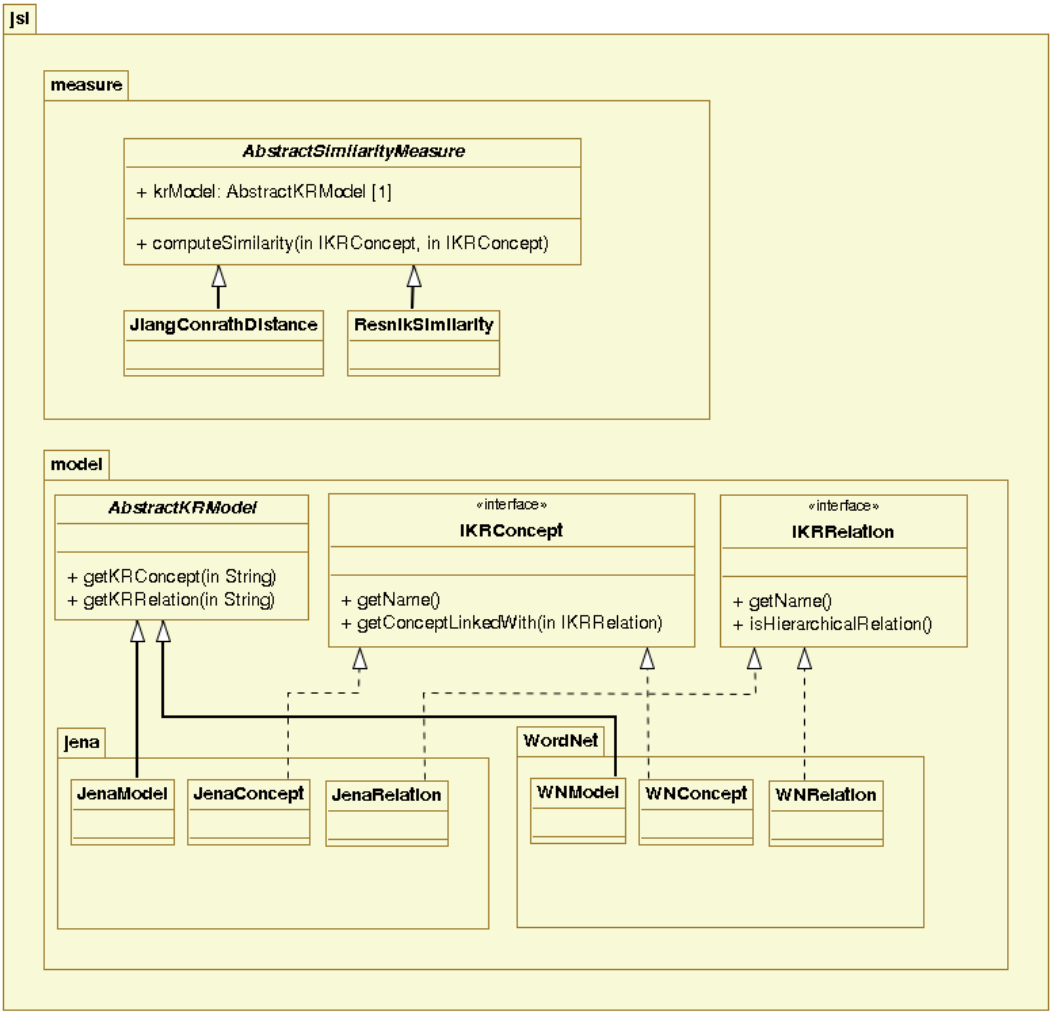

FIG. 6.1 – Diagramme de classe UML simplifié de l'architecture de la librairie JSL

concept ou encore le plus proche parent commun *ccp*.

– Des possibilités de précalculer les valeurs critiques, telles que les valeurs de la fonction IC par exemple, et d'utiliser un système de cache pour rendre l'exécution la plus courte possible (avec comme objectif une utilisation fluide dans le cadre d'un système d'interaction, tel que la réponse de l'ordinateur à une requête arrive dans un délai raisonnable).

6.1.2 Le modèle KR

L'architecture globale est construite sur un modèle de patrons, représenté en Java par des interfaces et/ou des classes abstraites (figure 6.1). Ainsi, le modèle de représentation de connaissance choisie (OWL, WordNet, etc.) n'est pas directement utilisé, mais est encapsulé dans un modèle abstrait nommé KR. Ceci permet de définir les mesures sémantiques en fonction des méthodes de l'interface KR et non en fonction des méthodes spécifiques au API des différents modèles utilisés. Ainsi, une même mesure peut être utilisable, sans réécriture de code, sur différents modèles, tant que l'intégration de ce modèle a été faite selon l'interface

KR. Actuellement, les modèles intégrés sont Jena, et Protégé-OWL API[7] pour OWL, JWNL[8] pour WordNet (jusqu'à la version 2.0) et JWI[9] pour WordNet 3.0.

Le modèle KR est un modèle de représentation des connaissances basé sur la théorie des graphes. Un modèle est alors une taxonomie de concept (taxonomie représentant les relations de subsomption entre les différents concepts) augmentée de relation non contraintes définie par l'utilisateur. Ce modèle permet de respecter les informations fournies dans WordNet, mais limite l'expressivité de OWL (nous ne considérons pas actuellement les intersections de classes, contraintes de cardinalités, etc.). Néanmoins, comme nous en avons discuté à la section 1.2, il n'existe pas de mesures qui profitent de cette expressivité actuellement et le modèle n'est ainsi pas une limitation pour l'expressivité des distances sémantiques.

Un modèle est simplement décomposé en trois classes :

- `AbstractKRModel` : La classe représentant un modèle. Cette classe possède aussi les méthodes de traitement non dépendante du modèle appliqué, tel que nous le définirons dans la suite de cette section ;
- `IKRConcept` : L'interface représentant un concept (un nœud dans l'arbre) ;
- `IKRRelation` : L'interface représentant un lien entre deux concepts (même hiérarchique).

Les méthodes d'accès à l'arbre KR à partir de l'interface `IKRConcept` sont :

- `getAllAncestor`() : Renvoie la liste des tous les ancêtres hiérarchiques du concept courant ;
- `getChildren`() : Renvoie la liste de tous les fils du concept courant ;
- `getOutcomingRelationTypes`() : Renvoie la liste des types de liens existant par du concept courant ;
- `getConceptsLinkedWith(IKRRelation rel)` : Renvoie la liste des concepts reliés au concept courant par un lien du type *rel*.

L'interface `IKRRelation` ne contient qu'une méthode en relation avec la structure de l'arbre, `isHierarchical`(), qui teste si un lien quelconque est hiérarchique ou non.

A partir de ces méthodes simples, il est possible de définir plusieurs traitements complexes, *indépendamment du modèle qui sera appliqué aux modèles KR*. En particulier :

- `getRoot()` : Renvoie le concept `IKRConcept` représentant la racine de l'arbre. Une racine est un nœud qui n'a pas de père. Une implémentation récursive suivant la méthode `getAllAncestor()` suffit donc pour implémenter cette fonction ;
- `getGlobalDepth()` : Renvoie la profondeur maximale de la hiérarchie ;
- `getDepth(IKRConcept c)` : Renvoie la profondeur du concept *c ;*
- `findCCP(IKRConcept c1, IKRConcept c2)` : Sans doute l'une des méthodes les plus importantes dans le cadre des mesures de degré de relation sémantique. Cette méthode renvoie le *ccp* (*i.e.* le plus proche commun parent) des deux concepts en paramètres ;
- `findShortestTaxonomicPath(IKRConcept c1, IKRConcept c2)` : Calcule le chemin le plus court entre les deux concepts, en ne considérant que les liens hiérarchiques ;
- `findShortestPath(IKRConcept c1, IKRConcept c2)` : Calcul le chemin le plus court entre les deux concepts, en considérant tous les liens possibles.

[7] `http://protege.stanford.edu/plugins/owl/api/index.html`
[8] `http://sourceforge.net/projects/jwordnet`
[9] `http://www.mit.edu/~markaf/projects/wordnet/`

Ces algorithmes ne sont dépendants que du modèle KR et sont donc automatiquement accessibles (sans nouveau code) à chaque ajout d'un modèle applicatif implémentant le modèle KR. Par exemple, pour la fonction `findCCP`, la fonction `getAllAncestor` peut être utilisée pour trouver tous les chemins possibles vers la racine pour chacun des concepts c_1 et c_2. Puis, pour un couple de chemins donné vers la racine (l'un partant de c_1 et l'autre partant de c_2), il suffit de trouver le concept le plus profond étant commun aux deux chemins pour trouver le *ccp*. Ainsi, l'implémentation du *ccp* n'est dépendante que du modèle KR.

6.1.3 Définition d'une mesure de similarité/distance

De même que différents algorithmes peuvent être définis directement sur le modèle KR, les mesures sémantiques sont définies aussi directement sur ce modèle, ce qui leurs permet d'être directement utilisables sur tout nouveau modèle applicatif. Toute mesure doit étendre la classe abstraite `AbstractSemanticMeasure`, qui fournit un ensemble de propriétés utilisables par toutes les formules.

6.1.3.1 Calcul de la quantité d'information d'un concept

La plupart des formules de similarité/distance sémantique actuelles reposent sur le fait que l'utilisation de la quantité d'information d'un concept (couramment noté IC). Le calcul de cette valeur n'est pas lié à la mesure en elle-même et peux être effectué selon différentes méthodes. Nous avons donc proposé de séparer le calcul de ce score afin qu'il ne soit pas géré par le programmeur d'une mesure. Ainsi, chaque classe représentant une mesure possède une instance d'une classe implémentant l'interface `IConceptInformationContent`. Cette classe décrit la méthode à utiliser pour calculer la quantité d'information d'un concept au moyen de la méthode `computeInformationContent(IKRConcept c)`. Dans la version actuelle, deux implémentations de cette interface sont proposées :

- `NunoSecoIC` : Le calcul de la quantité d'information selon la formule de Nuno Seco (voir section 1.2.3.3.1, [Seco *et al.*, 2004]) ;
- `ZhongIC` : Le calcul de la quantité d'information selon la formule de Zhong ($\left(\frac{1}{2}\right)^{depth(c)}$, voir section 1.2.3.2.3, [Zhong *et al.*, 2002]).

Pour accéder à la quantité d'information d'un concept, la classe `AbstractSemanticMeasure` contient la méthode `getIC(IKRConcept c)`. Cette méthode est reliée à un système de cache permettant de répondre directement dans le cas où ce score a déjà été calculé auparavant. Si le score n'est pas disponible dans le cache, alors la méthode `getIC(IKRConcept c)` appelle la fonction `computeInformationContent(IKRConcept c)` correspondant à son instance de l'interface `IConceptInformationContent`.

Le calcul de la fonction IC pouvant prendre du temps, nous avons inclus la classe `ICFileManager` permettant de précalculer et de sauver/charger des fichiers contenant les résultats dans le cache de la méthode `getIC(IKRConcept c)`. Ce fichier est dépendant d'un couple représenté par une formule IC et une ontologie donnée. La version actuelle de JSL est fournie avec les fichiers IC précalculés selon la formule de Nuno Seco pour WordNet 2.0 et WordNet 3.0.

6.1.3.2 Implémentation d'une mesure

Une mesure implémente ainsi la classe abstraite `AbstractSemanticMeasure` et en particulier la méthode abstraite `computeSemanticScore(IKRConcept c1, IKRConcept c2)`. Ce que nous avons pu définir précédemment nous permet de définir les mesures simplement. Par exemple, l'implémentation de la mesure de Lin (section 1.2.3.3.5) est :

```
public double computeSemanticScore(IKRConcept<T, U> c1,
                                   IKRConcept<T, U> c2) {
  // Optimisation : c1 = c2 = ccp => sim = 1.0
  if(c1.equals(c2)) return 1.0;
  IKRConcept<T, U> ccp = findCCP(c1, c2);
  if(ccp == null) { System.out.println("No CCP"); return 0.0; }
  return (2.0 * getIC(ccp)) / (getIC(c1) + getIC(c2));
}
```

La version actuelle de JSL définit pour le modèle KR les mesures de Rada, Hirst & St-Onge, Jiang & Conrath (distance et similarité par la conversion de Resnik), Leacock & Chodorow, Lin, Resnik, Wu & Palmer et notre proposition du chapitre 3.

6.2 Intégration de notre système dans la plate-forme VDL

L'ensemble du code pour le traitement de l'hétérogénéité sémantique présenté dans cette thèse a été directement intégré dans la plate-forme VDL. Ce code représente environ 12000 lignes, réparties (approximativement) de la façon suivante :

- 1000 lignes pour le générateur de capacités VDL (voir section 6.2.3).
- 7500 lignes pour la partie Traitement Automatique des Langues. Cette partie fait l'objet de la section 6.3 de ce chapitre.
- 3500 lignes pour la partie interaction ou les intégrations diverses. Nous présentons cette intégration dans cette section.

L'architecture d'intégration est présentée sur la figure 6.2. Le reste de cette section décrit chacun des composants présents sur la figure.

6.2.1 La gestion des connaissances : VDLConceptManager

Chaque agent possède une instance de la classe `VDLConceptManager`. Cette classe (qui représente l'implémentation de la fonction map_{VDL}, c.f. section 2.2.1) est une sorte de grande table de hachage faisant le lien entre les concepts VDL (tags, contenu texte ou attributs, section 2.2.1) et les concepts de l'ontologie. La recherche des équivalences se fait selon plusieurs critères :

- Les critères directs : par un fichier RDF de description des équivalences (Annexe D page 221). Ce fichier décrit explicitement certaines équivalences entre un terme VDL et un concept de l'ontologie.
- Les critères heuristiques : Dans ce cas, nous utilisons l'approximation morphologique décrite à la section 4.1.1.2 par la distance de Levenshtein.

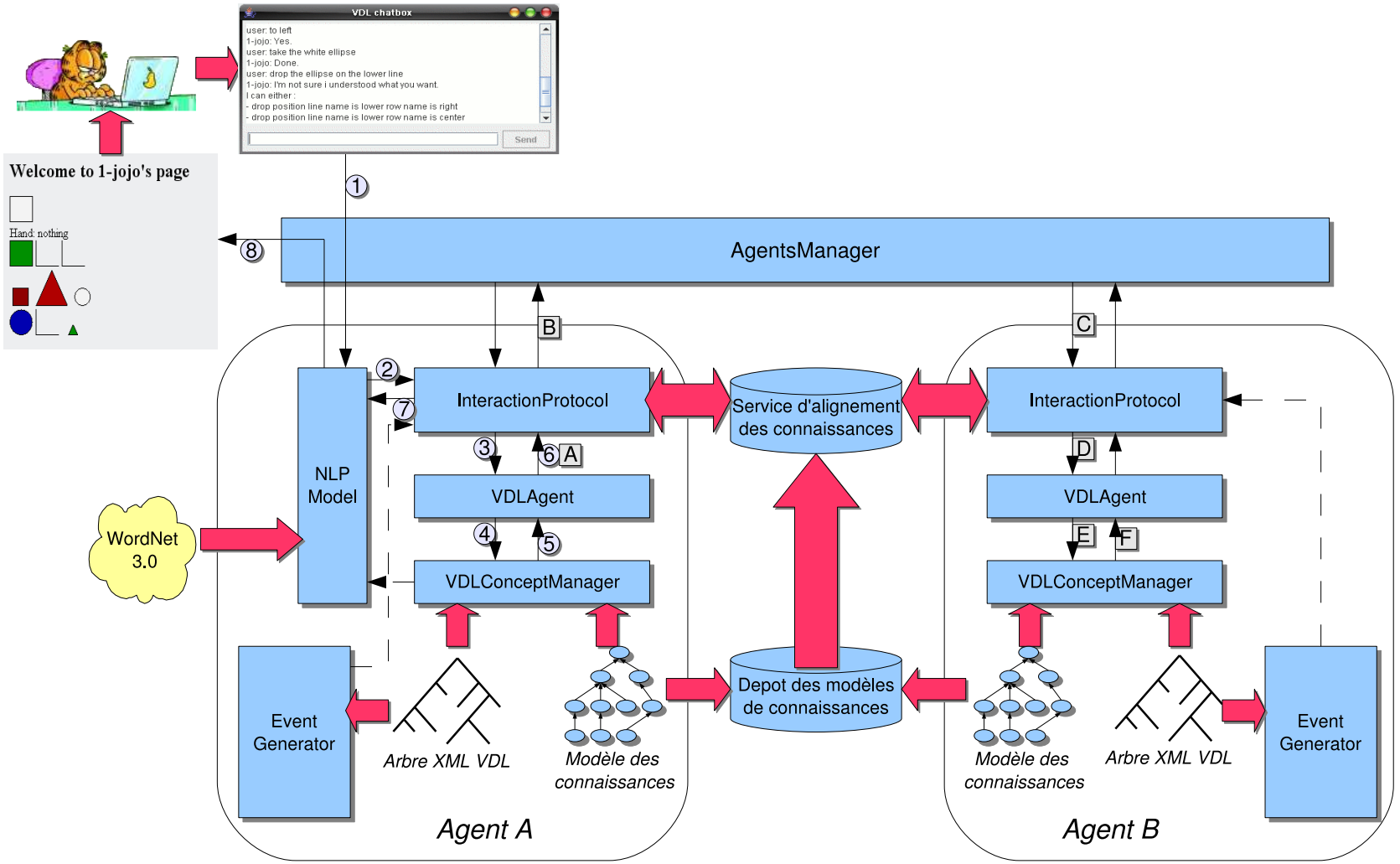

FIG. 6.2 – Présentation de l'architecture technique de l'intégration de notre système dans la plate-forme VDL. La numérotation en cercle des flèches représente le parcours classique d'une requête en langue naturelle. La numérotation alphabétique en carré des flèches représente l'envoie d'un message de l'agent A vers l'agent B. Quelle que soit la situation, tous les messages passent par la couche `InteractionProtocol` qui gère l'hétérogénéité sémantique des messages. Les flèches en pointillés représentent une utilisation du module au départ de la flèche par l'autre module.

La création de cette instance est effectuée au démarrage de l'agent. Son code XML est entièrement parcouru, et chaque terme rencontré est analysé. Tous les termes VDL doivent être, soit définis dans l'ontologie (à un appariement morphologique près), soit déclarés dans le fichier RDF.

6.2.2 L'interaction : InteractionProtocol

La couche `InteractionProtocol` gère l'ensemble des entrées/sorties de message d'un agent. Au niveau des sorties, la couche `InteractionProtocol` communique directement avec le gestionnaire d'agents pour décider du routage nécessaire. Au niveau des entrées, l'algorithme de traitement sémantique des requêtes présenté au chapitre 2 est directement intégré dans cette classe.

Cette classe est en amont de l'agent car une partie de ces traitements sont automatiques. En effet, l'interprétation d'une requête par rapport à l'ensemble des capacités possibles est traitée à ce niveau et la réponse est envoyée directement, de manière totalement transparente pour l'agent qui n'a pas eu à gérer de réaction.

6.2.3 La génération d'évènements : EventGenerator

La classe `EventGenerator` concerne le générateur d'évènements VDL (fonction $refine$, annexe A page 195). Dans la section 2.2, nous avons présenté le générateur d'évènements comme générant l'ensemble $\mathcal{G}$ de tout ce qui était possible, puis séparant cet ensemble en deux ensembles $\mathcal{E}$ et $\mathcal{F}$. En pratique, pour optimiser notre système nous pouvons directement construire les deux ensembles $\mathcal{E}$ et $\mathcal{F}$. Notre fonction est ainsi paramétrable :

```
public Map<VDLEvent, VDLNodeSet> getAgentEvents(EventsType eventsType,
                                                VDLAgent agent);
```

tel que `EventsType` est une énumération permettant de spécifier quel type de génération est demandé :

```
public enum EventsType {ALL_EVENTS, POSSIBLE_EVENTS, IMPOSSIBLE_EVENTS}
```

6.2.4 Le noyau VDL : VDLAgent et AgentsManagers

La classe `VDLAgent` représente un agent complet, avec son module de traitement des langues, ses liens vers sa couche `InteractionProtocol`, etc. Cette classe ne possède pas de traitement à proprement parler mais regroupe tous ces modules, l'ontologie et le code.

La classe `AgentsManager` représente le dépôt des agents sur une machine donnée (chaque machine d'un réseau possédera donc sa classe `AgentsManager`). Cette classe mémorise les emplacements des agents, et s'occupe ainsi des routages de messages locaux (dans la même instance de `AgentsManager` sur la même machine) ou distants (sur une autre machine, une autre instance de `AgentsManager`).

Ces classes représentent le cœur de VDL existaient avant notre travail sur la sémantique. Au cours des développements de cette thèse, nous avons intégré et ajouté diverses fonctionnalités à la base de VDL. Par exemple l'ajout d'une ontologie à un agent et sa gestion des

concepts par `VDLConceptManager` ou encore l'ajout des capacités de communication en langue naturelle. En effet, auparavant il n'était possible de communiquer avec un agent VDL qu'avec des nœuds XML.

6.3 Implémentation du Traitement Automatique des Langues

Le Traitement Automatique des Langues (TAL) constitue une part importante de code dans l'architecture. En effet, du fait de la grande diversité des requêtes possibles, il existe beaucoup de situations limites demandant un réglage précis du système. Nous décrivons dans cette section les principales modifications que nous avons du apporter à l'outil de base que nous avons utilisé pour optimiser la construction des requêtes. En particulier, nous traiterons dans une première section des améliorations apportées à notre couple étiqueteur/chunker, puis nous décrirons notre module de reconnaissances des actes de langages.

6.3.1 Étiqueteur et Chunker

Le chunking et l'étiquetage des termes d'une phrase font parties des phases du traitement d'une requête en langue naturelle qui génèrent le plus d'erreurs de modélisation. Nous avons ainsi du mettre en place une boucle de rétroaction permettant de tester diverses propositions d'étiquetages et chunking en parallèle. En effet, souvent lorsque l'étiquetage présente une erreur, le chunking n'en présente pas, car les contraintes de structures de la phrase sont suffisantes. Dans ce cas, on observe une incohérence entre l'étiquetage et le chunking. Par exemple, dans le chunk `[VP drop :NN]`, il est syntaxiquement incorrect d'avoir un syntagme verbal ne contenant qu'un nom. Il est ainsi possible de déduire que l'étiquetage est en erreur et choisir une nouvelle proposition d'étiquetage plus cohérente avec le chunking.

Pour traiter ce problème, nous utilisons un algorithme qui s'appuie sur les probabilités d'étiquetage et de chunking fournies par la bibliothèque de traitement automatique des langues que nous utilisons, OpenNLP. Notre algorithme considère ainsi d'abord l'ensemble le plus probable. Puis, nous testons la cohérence de l'ensemble étiquetage/chunking, en utilisant un ensemble de règles simples (en exploitant par exemple le fait qu'un groupe verbal soit forcément contenir un verbe). Si l'ensemble n'est pas cohérent, alors nous passons au couple suivant le plus probable et ainsi de suite jusqu'à obtenir un couple correct.

Par exemple, l'étiquetage le plus probable de la phrase « drop this object to the left » est :

```
drop :VV this :DT object :NN to :TO the :DT left :VVN
```

En utilisant notre système, une incohérence est détectée pour l'étiquette du terme *left*. le chunking identifie le groupe « the left » comme un syntagme nominal. Devant cette incohérence, notre système choisit le deuxième étiquetage le plus probable, puis reteste sa cohérence avec le résultat du chunking. Ainsi, le terme *left* est bien compris comme un nom :

```
[VP drop :VB] [NP this :DT object :NN] [PP to :TO] [NP the :DT left :NN]
```

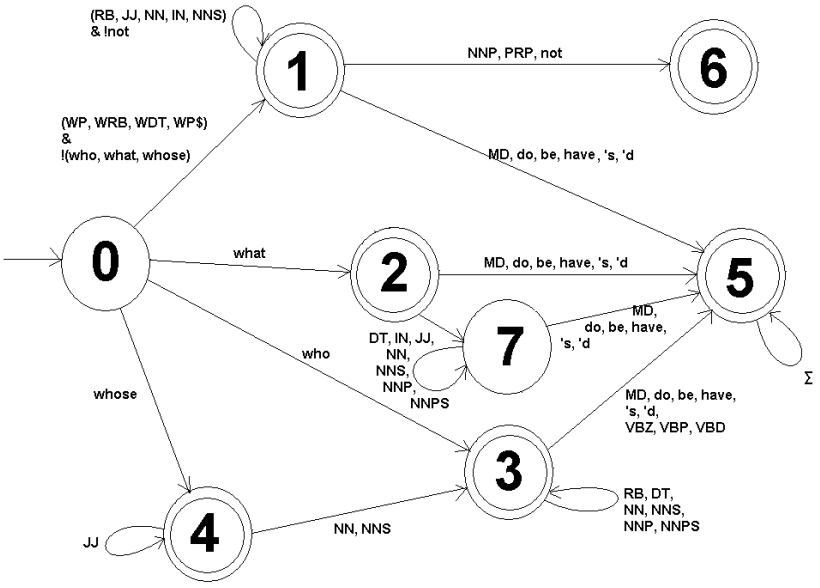

FIG. 6.3 – Un exemple d'automate pour détecter les actes de langages. Ici l'automate « question ».

6.3.2 Détection des actes de langage

Afin de spécifier le type d'interaction à enclencher avec l'agent, nous détectons les actes de langage utilisé pour une phrase en langue naturelle. Cette détection est basée sur la définition d'automates d'analyse de la phrase, les transitions étant assurées par l'étiquetage syntaxique des termes. Nous détectons actuellement les actes :

- NEGATION, pour une phrase négative ;
- ORDER, pour une commande ;
- QUESTION, pour une question simple d'ordre général ;
- YES_NO_QUESTION, pour une question à réponse binaire oui/non ;
- ACCEPT, pour une confirmation de l'utilisateur (*e.g.* « ok », « j'accepte », etc.) ;
- HI, pour une formule de politesse de type « bonjour » ;
- THANKS, pour une formule de politesse de type « merci ».

La figure 6.3 illustre la reconnaissance des actes interrogatifs. Chaque acte de langage possède son propre automate.[10]

6.3.3 Limites et perspectives d'améliorations

Les deux précédents modules nous ont permis d'améliorer de manière significative notre capacité de modélisation des requêtes en langue naturelle. Dans le premier cas, le bon étiquetage des mots permet une meilleure prise en charge de leur sémantique (un verbe n'a pas le même sens qu'un nom, même s'ils viennent de la même forme). D'autre part, détecter les actes de langages permet de mieux comprendre l'intention de l'utilisateur et de lui répondre

[10] Nous remercions encore une fois les étudiants de Master Julien Heliot et Dia Al Jrab qui ont contribué à développer ce point de l'architecture.

FIG. 6.4 – L'interface graphique de notre agent Wumpus.

avec plus d'à propos (par exemple à ne pas essayer de traiter la phrase « bonjour » comme une commande).

Néanmoins, d'autres améliorations sont encore possibles. En particulier, nous pensons qu'une gestion des références d'objets basée sur l'historique de la conversation pourrait être un plus intéressant pour augmenter l'aspect dialogique de notre système. Par exemple, dans la phrase « attrape-la et rapporte-la moi », le pronom « la » ne peut pas être interprété sans une utilisation de l'historique de conversation.

6.4 Exemples d'agents

Plusieurs agents ont été proposés pour tester et améliorer certaines capacités du système. Parmi ces agents nous trouvons :

- L'agent Wumpus. Cet agent reprend le problème maintenant classique du Wumpus [Russel et Norvig, 2003] (figure 6.4). Conformément au Wumpus, les commandes de bases sont « avancer d'un case », « tirer une flèche » et « changer de direction ». Cet agent nous a permis d'évaluer le problème de la planification à partir d'actions basique. En effet, des traitements du type « avance tant que tu ne sens pas le monstre » demande un moteur de planification. Le problème de planification se situe au-delà de cette thèse et sera abordé dans la conclusion de ce manuscrit.
- L'agent Tour de Hanoï (figure 6.5). Agent encore une fois commandé par un utilisateur humain. Le rôle conceptuel de cet agent fut d'étudier la capacité de notre système à comprendre les commandes non symétriques. En effet, sans l'arbre de dépendance, la commande « move from the first stack to the second stack » ne serait pas traitable, car un nuage de concepts perd ici l'information importante « from/to ».

6.5 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation que nous avons effectuée des formules algorithmiques décrites dans les chapitres 2 à 5. Nous avons d'abord présenté la

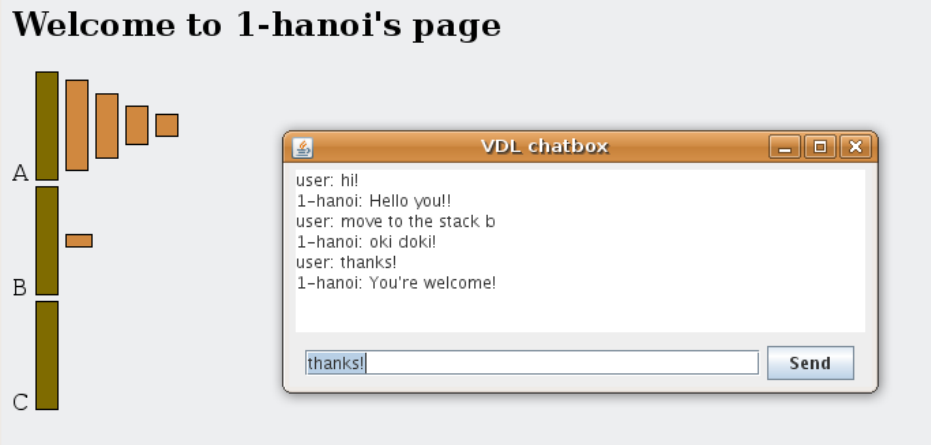

FIG. 6.5 – L'interface web de notre agent Hanoi.

bibliothèque JSL, qui nous permet de calculer notre mesure de degré de relation sémantique sur l'ontologie des agents (chapitre 3), ainsi que d'utiliser la mesure de Jiang & Conrath sur WordNet pour la désambiguïsation sémantique des commandes (section 4.1.1.2.2). Nous avons ensuite présenté l'intégration de notre système dans la plate-forme VDL, ainsi que les méthodes utilisées pour améliorer la modélisation d'une requête utilisateur. Enfin, nous avons présenté deux agents supplémentaires (en plus de l'agent « Jojo » présenté section 2.4.1) développés au cours de cette thèse pour étudier et évaluer notre système.

Conclusion

« Ce n'est pas la fin. Ce n'est même pas le commencement de la fin. Mais, c'est peut-être la fin du commencement. »

Winston Churchill

Les propositions de la littérature autour du problème de la gestion de l'hétérogénéité sémantique entre deux entités sont toujours dépendantes du type d'interaction mis en jeux (agent-agent ou humain-agent). Dans le cadre humain-agent, les solutions étudient les méthodes d'approximations sémantiques et de mesure sémantique, pour modéliser le plus de vocabulaire possible exprimé par un utilisateur humain. Dans le cadre agent-agent, la majeure partie des travaux proposent de calculer un alignement des bases de connaissances des agents (que cet alignement soit calculé par protocole d'interaction, négociation sémantique, ou autres), cet alignement suffisant alors à résoudre le problème. Or, il n'y a pas de travaux qui ont envisagé une approche combinée permettant d'utiliser les mesures sémantiques des bases de connaissances et en même temps un protocole d'interaction résistant aux alignements imparfaits.

Dans cette thèse, nous avons proposé une première initiative dans ce sens. Proposer des architectures applicables aussi bien en interaction humain-agent et agent-agent devient un problème d'actualité, puisque ces architectures s'appliquent dans les communautés mixtes humain-agent, en particulier dans le cadre de l'intelligence ambiante [Viterbo *et al.*, 2008]. Ce modèle conduit ainsi vers la définition de systèmes ouverts, intelligents et auto-adaptatifs.

Nous avons tout d'abord décrit un système de gestion de l'hétérogénéité sémantique indépendant du type précis d'interaction. Ce système se base sur une analyse dynamique des capacités de l'agent réceptionnant les messages, pour calculer un score d'appariement sémantique entre la requête reçue et ses capacités. Ce score exploite une mesure sémantique définie sur l'ontologie de l'agent cible. Enfin, le résultat de ce score permet de choisir une politique de réponse (basée sur le choix entre plusieurs performatifs) pour expliquer à l'agent qui a émis le message la situation de l'interaction.

Nous avons ensuite présenté notre proposition de mesure sémantique utilisable dans le système de score précédent. Cette mesure est une mesure de degré de relation sémantique utilisant une structure taxonomique augmentée de relations hétérogènes. Cette mesure utilise des hypothèses sur la validation et la construction de chemins sémantiquement corrects dans un graphe, puis associe des pondérations spécifiques au type d'arête rencontré pour évaluer le degré de relation sémantique entre deux concepts. Nous avons proposé un algorithme pour implémenter cette mesure qui permet d'éviter l'explosion combinatoire d'une énumération

directe des chemins possibles dans un graphe. Enfin, nous avons évalué cette mesure sur les protocoles classiques d'ingénierie des connaissances et montré qu'elle obtenait des résultats meilleurs que les mesures actuelles de la littérature.

Nous avons présenté les adaptations nécessaires au fonctionnement de notre architecture dans le cadre humain-agent. En particulier, nous avons énoncé les différents outils de traitement automatiques des langues utiles pour modéliser les requêtes d'un utilisateur en perdant le moins d'informations possibles. Pour cela, nous avons expliqué comment construire un arbre de dépendances sur les concepts de la requête qui donne une représentation de la structure logique de la commande, comme c'est le cas dans le cadre agent-agent. Nous avons de même expliqué comment réduire cet arbre de dépendances en résolvant (avant l'interprétation sémantique) les références relatives contenues dans la requête. Enfin, nous avons présenté une évaluation de cette version humain-agent qui a permis de montrer que les utilisateurs trouvent pertinent l'utilisation d'un système de clarification dans ce contexte.

Nous avons ensuite présenté la version agent-agent de notre architecture. Nous avons montré comment transformer une requête définie selon l'ontologie de l'agent émetteur en une requête définie selon l'ontologie de l'agent récepteur. Ensuite, nous avons présenté un protocole de communication permettant à deux agents autonomes d'utiliser notre architecture. Nous avons présenté une évaluation de ce système, basée sur une génération automatique des agents et des bases de connaissances sous contraintes, de manière à identifier les paramètres influençant le plus les résultats. Nous avons montré que l'approche permet de traiter jusqu'à 80% de messages en plus, messages qui seraient abandonnés car jugés ambigus par l'agent récepteur.

Nous avons enfin implémenté l'intégralité des méthodes proposées précédemment. Cette implémentation nous permet de fournir à l'issue de cette thèse une plate-forme multi-agents avec des outils de gestion de l'hétérogénéité sémantique applicables pour la communication agent-agent et humain-agent.

Cette thèse ne représente qu'un premier pas vers un système de gestion de l'hétérogénéité sémantique applicable aux SMA pouvant interagir avec des humains. En particulier, plusieurs problématiques apparaissent dans le cadre de notre approche et doivent être approfondies pour compléter ce travail.

Tout d'abord, l'un des défauts majeurs du système actuel est son impossibilité à comprendre les commandes nécessitant l'exécution de plusieurs capacités en séquence. Ceci est un problème de planification, or nous n'avons pas abordé cette problématique dans cette thèse. Plusieurs planificateurs existent pour le langage VDL (chaînage avant, Graphplan [Russel et Norvig, 2003], etc.), mais pour pouvoir utiliser un algorithme de planification il faut connaître l'état du monde souhaité à la fin de la planification. Or, une commande ne représente pas forcement un état complet du monde. Plus précisément, les commandes peuvent être partionnées en deux types :

1. Les commandes composées d'actions primaires. Exemple : « prend l'objet et pose le ici ». Cette commande s'exprime directement en un ensemble d'actions primaires. Ce cas ne nécessite pas d'appel à planification et peut se résoudre assez simplement.
2. Les commandes exprimant un *état* souhaité. Par exemple, « Mets le triangle rouge à la place du carré vert » ou encore « Je veux être à Denver mercredi pour 13h ». Dans

ce cas, il est nécessaire de modéliser l'état du monde souhaité par l'utilisateur pour pouvoir planifier la série d'actions nécessaires.

Cette dernière situation est problématique car l'approche bottom-up générative n'est alors plus utilisable. En effet, alors qu'il était possible de générer les capacités de l'agent à l'instant courant, il n'est pas envisageable de générer l'intégralité des états possibles pour un agent à partir de l'état courant. Ainsi, il semble que pour étendre notre approche et considérer plus de requêtes possibles, il faille utiliser une approche mixte combinant une approche directe top-down de modélisation d'un état du monde et une approche d'appariement bottom-up.

L'une des autres perspectives de ce travail concerne la prise en compte des autres agents logiciels existant dans l'environnement, soit par une coordination entre les services offerts [Charif et Sabouret, 2007] ou par l'utilisation d'une architecture de services Web sémantique [Reynaud et Giraldo, 2003]. En effet, dans le cadre de ce travail nous nous sommes concentré sur une interaction à deux intervenants (un émetteur et un récepteur). En particulier, l'agent récepteur de la commande cherche *seul* à effectuer la requête qu'il reçoit. Or, il est possible qu'un groupe d'agent puisse effectuer la requête, chacun effectuant des sous-parties simples impliquant la bonne exécution globale de cette requête [Charif et Sabouret, 2007]. Néanmoins, ces derniers travaux n'ont pas considéré le problème de l'hétérogénéité sémantique entre les agents du groupe, focalisant sur la coordination. Une approche hybride combinant notre proposition de résolution des ambiguïtés sémantiques à l'approche coordinatrice des SMA (ou de la médiation de services Web) pourrait être intéressante à étudier.

Finalement, notre dernière proposition de perspective concerne le fonctionnement cognitif des agents. En effet, lors du chapitre 5, nous avons supposé que l'agent émetteur avait déjà une idée de qui était possible par l'autre agent. Or, dans un SMA ouvert, les possibilités des agents entrants ne sont pas connues à priori. Il est alors nécessaire, avant d'envisager de communiquer avec un agent, d'apprendre les possibilités des autres agents du système. Cet apprentissage peut se résoudre par un protocole de communication permettant aux agents d'échanger des informations sur leurs possibilités. Une interaction de communication correspond alors à un état d'un algorithme d'apprentissage par renforcement [Sutton, 1988, Watkins et Dayan, 1992]. Mais quels sont les informations que doivent échanger ces agents ? Quel sera l'impact de cet apprentissage sur la gestion de l'hétérogénéité sémantique, puisque les informations que s'envoient ces agents seront, par essence, hétérogènes ? Cette problématique de modélisation et d'apprentissage des autres agents de l'environnement fait l'objet de la thèse de Shirley Hoet, thèse commençant au moment où se termine celle-ci.

Liste des publications effectuées

Revues internationales avec comités de lecture

- L. Mazuel, N. Sabouret. Generic command interpretation algorithms for conversational agents. Dans *Web Intelligence and Agent Systems*, (IOS press), Vol.6, N°2. Avril 2008

Chapitre de livre

- J. Viterbo, L. Mazuel, Y. Charif, M. Endler, N. Sabouret, K. Breitman, A. El Fallah Seghrouchni et J.-P. Briot. Ambient intelligence : Management of distributed and heterogeneous context knowledge. In *Context-Aware Self Managing Systems, CRC Studies in Informatics Series*. Waltenegus Dargie (eds), Chapman & Hall. à paraître, 2008.

Conférences internationales avec comités de lecture

- L. Mazuel et N. Sabouret. A communication protocol for semantic heterogeneity with incomplete ontology alignment (Short Paper). In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pp. 1–2, à paraître, 2009.
- L. Mazuel et N. Sabouret. Semantic relatedness measure using object properties in an ontology. In *7th International Semantic Web Conference (ISWC'08)*, pp. 681–694, Springer, 2008.
- L. Mazuel et N. Sabouret. Semantic relatedness in semantic networks. In *18th European Conference on Artificial Intelligence (ECAI'08)*, pp. 727–728, IOS Press, 2008.
- L. Mazuel, N. Sabouret. Modeling user's commands using ontologies. In *Workshop on Communication between Human and Artificial Agents (CHAA07, IAT07 Workshop)*, pp. 325–328, IEEE Computer Society, 2007.
- L. Mazuel, N. Sabouret. How to enhance genericity in natural language command interpretation using introspection and ontologies ?, In *The 4th Conference on Speech Technology and Human - Computer Dialogue (SpeD07)*, pp. 81–88, IEEE Romania, 2007.
- L. Mazuel, N. Sabouret. Generic command interpretation algorithms for conversational agents, In *Intelligent Agent Technology (IAT'06)*, pp. 146–153, IEEE Computer Society, 2006.
- L. Mazuel, N. Sabouret. Generic natural language command interpretation in ontology-based dialogue systems, In *Workshop on Communication between Human and Artificial Agents (CHAA06, IAT06 Workshop)*, pp. 347–350, IEEE Computer Society, 2006.

Conférences nationales avec comités de lecture

- L. Mazuel et N. Sabouret. Un modèle d'interaction pour des agents sémantiquement hétérogènes. In *16e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'08)*, pp. 233–242, 2008.
- L. Mazuel, N. Sabouret. Protocole d'évaluation d'une mesure de degré de relation sémantique. In *Premier Atelier sur les "Mesures de similarité sémantique" (EGC'08)*, pp. 77–86, 2008.
- L. Mazuel, N. Sabouret. Interprétation de commandes en langage naturel pour les agents conversationnels à base d'ontologie. In *Quatrième Journées Francophones des Modèles Formels de l'Interaction (MFI'07)*, pp. 341–348, 2007.
- L. Mazuel, N. Sabouret. Degré de relation sémantique dans une ontologie pour la commande en langue naturelle. In *Plate-Forme AFIA, Ingénierie des Connaissances 2007 (IC 2007)*, pp. 73–83.
- L. Mazuel. Utilisation des ontologies pour la modélisation logique d'une commande en langue naturel. In *Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL 2007)*, pp. 427–436.
- L. Mazuel, N. Sabouret. Vers une approche générique pour l'interprétation de commandes en langage naturel. In *Rencontre des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 2007)*, pp. 133–149.
- L. Mazuel, N. Sabouret. Une interprétation générique de commandes en langue naturelle dans les systèmes de dialogues à base d'ontologies, In *Second Workshop sur les Agents Conversationnels Animés (WACA 06)*, pp. 105-113, 2006.
- N. Sabouret, L. Mazuel. Commande en langage naturel d'agents VDL, In *Premier Workshop sur les Agents Conversationnels Animés (WACA)*, pp. 53-62, 2005.

Bibliographie

[Aleksovski *et al.*, 2006a] Aleksovski, Z., Klein, M., ten Kate, W. et van Harmelen, F. (2006a). Matching Unstructured Vocabularies using a Background Ontology. *Proceedings of Knowledge Engineering and Knowledge Management (EKAW)*, pages 182–197.

[Aleksovski *et al.*, 2006b] Aleksovski, Z., ten Kate, W. et van Harmelen, F. (2006b). Exploiting the structure of background knowledge used in ontology matching. *In Proc. Workshop on Ontology Matching in ISWC2006*. CEUR Workshop Proceedings.

[Allen *et al.*, 2000] Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L. et Stent, A. (2000). An architecture for a generic dialogue shell. *NLENG : Natural Language Engineering*, 6.

[Allen *et al.*, 1996] Allen, J., Miller, B. W., Ringger, E. K. et Sikorski, T. (1996). A robust system for natural spoken dialogue. *In ACL*, pages 62–70.

[Atencia et Schorlemmer, 2007] Atencia, M. et Schorlemmer, M. (2007). A formal model for situated semantic alignment. *Proc. of the 6th international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'07)*, pages 1270–1277.

[Austin, 1962] Austin, J. (1962). *How to Do Things with Words*. Oxford.

[Baader *et al.*, 2007] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (2007). The Description Logic Handbook - Theory, Implementation, and Applications - 2nd Edition.

[Banerjee et Pedersen, 2003] Banerjee, S. et Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. *Proc. of the 8th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 805–810.

[Barwise et Seligman, 1997] Barwise, J. et Seligman, J. (1997). *Information flow : the logic of distributed systems*. Cambridge University Press, New York, NY, USA.

[Bateman, 1997] Bateman, J. A. (1997). Enabling technology for multilingual natural language generation : the kpml development environment. *Nat. Lang. Eng.*, 3(1):15–55.

[Bernstein *et al.*, 2005] Bernstein, A., Kaufmann, E., Bürki, C. et Klein, M. (2005). How similar is it ? towards personalized similarity measures in ontologies. *In Proc. 7. Internationale Tagung Wirtschaftsinformatik*, pages 1347–1366.

[Blanchard, 2008] Blanchard, E. (2008). *Exploitation d'une hiérarchie de subsomption par le biais de mesures sémantiques*. Thèse de doctorat, Laboratoire d'Informatique de Nantes Atlantique (LINA).

[Blanchard *et al.*, 2005] Blanchard, E., Harzallah, M., Briand, H. et Kuntz, P. (2005). A typology of ontology-based semantic measures. *EMOI-INTEROP'05, Proc. Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability (at CAISE'05)*, (407–412).

[Botella *et al.*, 2002] Botella, B., Taillibert, P. et Gotlieb, A. (2002). *Test de logiciel*, volume 21, chapitre Utilisation des contraintes pour la génération automatique de cas de test structurels, pages 1163–1187. RSTI-TSI.

[Bouquet *et al.*, 2003] Bouquet, P., Serafini, L. et Zanobini, S. (2003). Semantic coordination : A new approach and an application. *In* Mylopoulos, J., Fensel, D. et Sycara, K., éditeurs : *Proc. 2nd International Semantic Web Conference (ISWC)*, volume 2870 de *LNCS*, pages 130–145. Springer Verlag.

[Bousquet *et al.*, 2000] Bousquet, C., Jaulent, M.-C., Chatellier, G. et Degoulet, P. (2000). Using semantic distance for the efficient coding of medical concepts. *In Proc. Converging Information, Technology, and Health Care (AMIA)*.

[Bratman, 1987] Bratman, M. (1987). *Intention, plans, and practical reason*. Harvard University Press Cambridge, Mass.

[Breitman *et al.*, 2005] Breitman, K., Felicíssimo, C. et Casanova, M. (2005). CATO–A Lightweight Ontology Alignment Tool. *Proc. 17th Conf. on Advanced Information Systems Engineering (CAISE'05)*.

[Budanitsky et Hirst, 2006] Budanitsky, A. et Hirst, G. (2006). Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

[Burstein *et al.*, 2003] Burstein, M., McDermott, D., Smith, D. et Westfold, S. (2003). Derivation of Glue Code for Agent Interoperation. *Autonomous Agents and Multi-Agent Systems*, 6(3):265–286.

[Bylander et Chandrasekaran, 1987] Bylander, T. et Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning : the right level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):231–243.

[Cassel *et al.*, 2000] Cassel, J., Sullivan, J., Prevost, S. et Churchill, E. (2000). *Embodied Conversational Agents*. MIT Press.

[Charif et Sabouret, 2007] Charif, Y. et Sabouret, N. (2007). Coordination in Introspective Multi-Agent Systems. *In Proc. of the International Conference on Intelligent Agent Technology (IAT'07)*, pages 412–415.

[Cho *et al.*, 2003] Cho, M., Choi, J. et Kim, P. (2003). *An Efficient Computational Method for Measuring Similarity between Two Conceptual Entities*, volume 2762/2003 de *LNCS - Time Series, Similarity, and Ontologies*, pages 381–388. Springer Berlin / Heidelberg.

[Cilibrasi et Vitanyi, 2006] Cilibrasi, R. et Vitanyi, P. (2006). Automatic Extraction of Meaning from the Web. *In Proc. IEEE International Symposium on Information Theory*, pages 2309–2313.

[Corby *et al.*, 2004] Corby, O., Dieng-Kuntz, R. et Faron-Zucker, C. (2004). Querying the Semantic Web with the CORESE search engine. *In* Press, I., éditeur : *Proc. of the ECAI'2004*, pages 705–709, Valencia.

[De Kleer, 1984] De Kleer, J. (1984). How Circuits Work. *Artificial Intelligence*, 24(1-3):205–280.

[Dzikovska *et al.*, 2003] Dzikovska, M. O., Allen, J. F. et Swift, M. D. (2003). Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. *In Proc. of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems.*

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

[Euzenat, 2008] Euzenat, J. (2008). Quelques pistes pour une distance entre ontologies. *Atelier Mesures de Similarité Sémantique - EGC'08.*

[Euzenat et Shvaiko, 2007] Euzenat, J. et Shvaiko, P. (2007). *Ontology matching.* Springer-Verlag, Heidelberg (DE).

[Fellbaum, 1998] Fellbaum, C., éditeur (1998). *WordNet, An Electronic Lexical Database* . MIT-Press.

[Ferber, 1995] Ferber, J. (1995). *Les systèmes multi-agents : Vers une intelligence collective.* InterEditions.

[Ferguson et Allen, 1998] Ferguson, G. et Allen, J. (1998). Trips : An integrated intelligent problem-solving assistant. *In Proc. AAAI/IAAI*, pages 567–572.

[Finin *et al.*, 2005] Finin, T., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A. et Peng, Y. (2005). Swoogle : Searching for knowledge on the Semantic Web. *AAAI 05 (intelligent systems demo).*

[Finkelstein *et al.*, 2001] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. et Ruppin, E. (2001). Placing search in context : the concept revisited. *In WWW '01 : Proceedings of the 10th international conference on World Wide Web*, pages 406–414, New York. ACM Press.

[Franz Baader *et al.*, 2003] Franz Baader, D. C., McGuinness, D., Nardi, D. et Patel-Schneider, P. (2003). *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press.

[Gabrilovich et Markovitch, 2007] Gabrilovich, E. et Markovitch, S. (2007). Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. *In Proc. IJCAI07*, pages 1606–1611.

[Gandon *et al.*, 2008] Gandon, F., Corby, O., Diop, I. et Lo, M. (2008). Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique. *In Premier atelier sur les mesures de similarité sémantique - EGC'08*, pages 87–96.

[Garruzzo et Rosaci, 2006] Garruzzo, S. et Rosaci, D. (2006). HISENE2 : A Reputation-Based Protocol for Supporting Semantic Negotiation. *LNCS*, 4275:949–966.

[Gracia *et al.*, 2006] Gracia, J., Trillo, R., Espinoza, M. et Mena, E. (2006). Querying the Web : a Multiontology Disambiguation Method. *In Proc. 6th International Conference on Web Engineering (ICWE06)*, pages 241–248.

[Grosz et Sidner, 1986] Grosz, B. J. et Sidner, C. L. (1986). Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204.

[Gruber, 1995] Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928.

[Gurevych *et al.*, 2003] Gurevych, I., Porzel, R., Slinko, E., Pfleger, N., Alexandersson, J. et Merten, S. (2003). Less is more : using a single knowledge representation in dialogue systems. *In Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pages 14–21, Morristown, NJ, USA. Association for Computational Linguistics.

[Hau *et al.*, 2005] Hau, J., Lee, W. et Darlington, J. (2005). A Semantic Similarity Measure for Semantic Web Services. *In Proc. Workshop on Web Service Semantics.*

[Hirst et St-Onge, 1998] Hirst, G. et St-Onge, D. (1998). Lexical chains as representation of context for the detection and correction malapropisms. *In* Fellbaum, C., éditeur : *WordNet : An Electronic Lexical Database*, chapitre 13, pages 305–332. MIT Press.

[Ichise *et al.*, 2003] Ichise, R., Takeda, H. et Honiden, S. (2003). Integrating multiple internet directories by instance-based learning. *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence (IJCAI03).*

[Iosif, 2007] Iosif, E. and Potamianos, A. (2007). Unsupervised Semantic Similarity Computation usingWeb Search Engines. *In International Conference on Web Intelligence (WI)*, pages 381–387. IEEE.

[Jarmasz et Szpakowicz, 2003] Jarmasz, M. et Szpakowicz, S. (2003). Roget's thesaurus and semantic similarity. *Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pages 212–219.

[Jiang et Conrath, 1997] Jiang, J. et Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *In Proc. on International Conference on Research in Computational Linguistics*, pages 19–33, Taiwan.

[Johnson et Fillmore, 2000] Johnson, C. et Fillmore, C. (2000). The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure. *In Proc. ANLP-NAACL*, pages 56–62.

[Kalfoglou et Schorlemmer, 2003] Kalfoglou, Y. et Schorlemmer, M. (2003). Ontology mapping : the state of the art. *The Knowledge Engineering Review*, 18(01):1–31.

[Kalfoglou et Schorlemmer, 2005] Kalfoglou, Y. et Schorlemmer, M. (2005). Ontology mapping : The state of the art. *In* Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S. et Uschold, M., éditeurs : *Semantic Interoperability and Integration*, numéro 04391 de Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany.

[Kennedy et Szpakowicz, 2007] Kennedy, A. et Szpakowicz, S. (2007). Disambiguating Hypernym Relations for Roget's Thesaurus. *LNCS*, 4629:66–75.

[Kipfer et Chapman, 2001] Kipfer, B. A. et Chapman, R. L., éditeurs (2001). *Roget's International Thesaurus, Sixth Indexed Edition.* Collins, 6 édition.

[Klebanov et Shamir, 2006] Klebanov, B. et Shamir, E. (2006). Reader-based exploration of lexical cohesion. *Language Resources and Evaluation*, 40(2):109–126.

[Laera *et al.*, 2007] Laera, L., Blacoe, I., Tamma, V., Payne, T., Euzenat, J. et Bench-Capon, T. (2007). Argumentation over Ontology Correspondences in MAS. *Proc. of the*

6th international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS'07), pages 1285–1292.

[Leacock et Chodorow, 1998] Leacock, C. et Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *In* Fellbaum, C., éditeur : *WordNet : An Electronic Lexical Database*, chapitre 11, pages 265–283. MIT Press.

[Lin, 1998] Lin, D. (1998). An information-theoretic definition of similarity. *In Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA.

[Maedche et Staab, 2002] Maedche, A. et Staab, S. (2002). Measuring similarity between ontologies. *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW)*, pages 251–263.

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

[Mazuel et Sabouret, 2006] Mazuel, L. et Sabouret, N. (2006). Generic command interpretation algorithms for conversational agents. *In Proc. Intelligent Agent Technology (IAT'06)*, pages 146–153. IEEE Computer Society.

[Mazuel et Sabouret, 2007] Mazuel, L. et Sabouret, N. (2007). Modeling user's commands using ontologies. *In Workshop on Communication between Human and Artificial Agents 2007 (CHAA07, IAT07 Workshop)*, pages 325–328. IEEE Computer Society.

[Mazuel et Sabouret, 2008a] Mazuel, L. et Sabouret, N. (2008a). Generic command interpretation algorithms for conversational agents. *Web Intelligence and Agent Systems (IOS press)*, 6(2).

[Mazuel et Sabouret, 2008b] Mazuel, L. et Sabouret, N. (2008b). Semantic relatedness measure using object properties in an ontology. *In 7th International Semantic Web Conference (ISWC'08)*, pages 681–694. Springer.

[Mazuel et Sabouret, 2009] Mazuel, L. et Sabouret, N. (2009). A communication protocol for semantic heterogeneity with incomplete ontology alignment (Short Paper). *In Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 1–2.

[McHale, 1998] McHale, M. L. (1998). A Comparison of WordNet and Roget's Taxonomy for Measuring Semantic Similarity. *CoRR*, cmp-lg/9809003.

[Miller et Charles, 1991] Miller, G. et Charles, W. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

[Milward, 2000] Milward, D. (2000). Distributing representation for robust interpretation of dialogue utterances. *In ACL*, pages 133–141.

[Milward et Beveridge, 2003] Milward, D. et Beveridge, M. (2003). Ontology-based dialogue systems. *In Proc. 3rd Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI03)*, pages 9–18.

[Morge et Routier, 2007] Morge, M. et Routier, J.-C. (2007). Debating over heterogeneous descriptions. *Applied Ontology, Special issue on Formal Ontology for Communicating Agents*, 2(3-4):333 – 349.

[Odell *et al.*, 2001] Odell, J., Parunak, H. et Bauer, B. (2001). Representing Agent Interaction Protocols in UML. *Agent-Oriented Software Engineering*, pages 121–140.

[Paraiso et Barthès, 2004] Paraiso, E. et Barthès, J. (2004). Architecture d'une interface conversationnelle pour les agents assistants personnels. *In* Paroubeck, P. et Sansonnet, J.-P., éditeurs : *Actes de la Journée d'Etude ATALA Agental « Agents et Langue »*, pages 83–90, Paris, France. ATALA, ATALA.

[Paraiso *et al.*, 2004] Paraiso, E., Barthès, J. A. et Tacla, C. A. (2004). A speech architecture for personal assistants in a knowledge management context. *In Proc. European Conference on AI (ECAI)*, pages 971–972.

[Patwardhan et Pedersen, 2006] Patwardhan, S. et Pedersen, T. (2006). Using WordNet Based Context Vectors to Estimate the Semantic Relatedness of Concepts . *In Proc. of the EACL 2006 Workshop Making Sense of Sense - Bringing Computational Linguistics and Psycholinguistics Together.*

[Pollack, 1990] Pollack, M. (1990). Plans as complex mental attitudes. *Intentions in Communication*, pages 77–103.

[Ponzetto et Strube, 2007] Ponzetto, S. et Strube, M. (2007). Deriving a Large Scale Taxonomy from Wikipedia. *Proceedings of the 22st National Conference on Artificial Intelligence*, pages 1440–1445.

[Porzel *et al.*, 2003] Porzel, R., Gurevych, I. et Muller, C. (2003). Ontology-based contextual coherence scoring. *In Proc. of the Fourth SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan.

[Rada *et al.*, 1989] Rada, R., Mili, H., Bicknell, E. et Blettner, M. (1989). Development and Application of a Metric on Semantic Nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30.

[Rapaport, 2000] Rapaport, W. J. (2000). How to pass a turing test : Syntactic semantics, natural-language understanding, and first-person cognition. *Special Issue on Alan Turing and Artificial Intelligence, Journal of Logic, Language, and Information 9*, pages 467–490.

[Resnik, 1995] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *In 14th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 448–453.

[Reynaud et Giraldo, 2003] Reynaud, C. et Giraldo, G. (2003). An Application of the Mediator Approach to Services over the Web. *In Special track "Data Integration in Engineering", Proceedings of CE'2003*, pages 209–216.

[Rich *et al.*, 2001] Rich, C., Sidner, C. L. et Lesh, N. (2001). Collagen : Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–26.

[Richardson et Smeaton, 1995] Richardson, R. et Smeaton, A. F. (1995). Using WordNet in a knowledge-based approach to information retrieval. Rapport technique CA-0395, Dublin, Ireland.

[Rubenstein et Goodenough, 1965] Rubenstein, H. et Goodenough, J. (1965). Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

[Russel et Norvig, 2003] Russel, S. et Norvig, P. (2003). *Artificial Intelligence : A Modern Approach (Second Edition)*. Prentice Hall.

[Sabou *et al.*, 2006] Sabou, M., d'Aquin, M. et Motta, E. (2006). Using the Semantic Web as Background Knowledge for Ontology Mapping. *In Proc. Workshop on Ontology Matching in ISWC2006*. CEUR Workshop Proceedings.

[Sabouret, 2002] Sabouret, N. (2002). *Étude de modèles de représentation, de requêtes et de raisonnement sur le fonctionnement des composants actifs pour l'interaction homme-machine*. Thèse de doctorat, Université Paris-Sud.

[Sabouret et Sansonnet, 2001] Sabouret, N. et Sansonnet, J. (2001). Automated Answers to Questions about a Running Process. *In Proc. CommonSense 2001*, pages 217–227.

[Sabouret et Sansonnet, 2003] Sabouret, N. et Sansonnet, J. (2003). Un langage de description de composants actifs pour le web sémantique. *Revue RI*^{3}, 3(1):9–36.

[Sadek, 1999] Sadek, D. (1999). Design considerations on dialogue systems : From theory to technology - the case of artimis. *In ESCA Workshop Interactive Dialogue in Multi-modal Systems*, pages 173–187.

[Sadek *et al.*, 1997] Sadek, D., Bretier, P. et Panaget, E. (1997). Artimis : Natural dialogue meets rational agency. *In IJCAI*, pages 1030–1035.

[Sansonnet et Valencia, 2004] Sansonnet, J.-P. et Valencia, E. (2004). Modèle simpliciel pour l'hétérogénéité sémantique entre agents informationnels. *In* Hermes-Lavoisier, éditeur : *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'04)*, pages 271–284.

[Sansonnet et Valencia, 2005] Sansonnet, J.-P. et Valencia, E. (2005). Terminological heterogeneity between agents using generalized simplicial representation. *In European Workshop on Multi-Agent Systems (EUMAS'05)*.

[Sayed *et al.*, 2008] Sayed, A., Hacid, H. et Zighed, A. (2008). Enhancing distances with context awareness. *Atelier Mesures de Similarité Sémantique, à EGC'08*, pages 39–49.

[Searle, 1969] Searle, J. (1969). *Speech Acts*. Cambridge University Press.

[Seco *et al.*, 2004] Seco, N., Veale, T. et Hayes, J. (2004). An Intrinsic Information Content Metric for Semantic Similarity in WordNet. *In Proc. ECAI'2004, the 16th European Conference on Artificial Intelligence*, pages 1089–1090.

[Shannon et Weaver, 1948] Shannon, C. E. et Weaver, W. (1948). The mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423.

[Shapiro, 1982] Shapiro, S. (1982). Generalized augmented transition network grammars for generation from semantic networks. *In Proceedings of the Seventeenth Meeting of the Association for Computational Linguistics*, pages 25–29.

[Shapiro, 1996] Shapiro, S. (1996). Formalizing English. *International Journal of Expert Systems*, 9(1):151–171.

[Shapiro, 2000] 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*, pages 175–195.

[Shapiro et Kandefer, 2005] Shapiro, S. et Kandefer, M. (2005). A SNePS approach to the wumpus world agent : or Cassie meets the wumpus. *In In Proceedings the Nonmonotonic Reasoning, Action and Change Workshop at IJCAI (NRAC-05)*, Edinburgh, Scottland.

[Smith *et al.*, 2004] Smith, M. K., Welty, C. et McGuinness, D. L. (2004). Owl web ontology language guide. http ://www.w3.org/TR/owl-guide/.

[Smith, 1998] Smith, R. W. (1998). An evaluation of strategies for selectively verifying utterance meanings in spoken natural language dialog. *International Journal on Human-Computer Studies*, 48(5):627–647.

[Smith *et al.*, 1995] Smith, R. W., Biermann, A. W. et Hipp, D. R. (1995). An architecture for voice dialog systems based on prolog-style theorem proving. *Computational Linguistics*, 21(3):281–320.

[Smith *et al.*, 1992] Smith, R. W., Hipp, D. R. et Biermann, A. W. (1992). A dialog control algorithm and its performance. *In Proceedings of the third conference on Applied natural language processing*, pages 9–16, Morristown, NJ, USA. Association for Computational Linguistics.

[Sowa, 1984] Sowa, J. (1984). *Conceptual Structures : Information Processing in Mind and Machines*. Addison-Wesley.

[Sowa et Borgida, 1991] Sowa, J. et Borgida, A. (1991). *Principles of Semantic Networks : Explorations in the Representation of Knowledge*. Morgan Kaufmann Publishers.

[Steichen *et al.*, 2006] Steichen, O., Bozec, C., Thieu, M., Zapletal, E. et Jaulent, M. (2006). Computation of semantic similarity within an ontology of breast pathology to assist inter-observer consensus. *Computers in Biology and Medicine*, 36(7-8):768–788.

[Strube et Ponzetto, 2006] Strube, M. et Ponzetto, S. (2006). WikiRelate ! Computing semantic relatedness using Wikipedia. *Proc. of AAAI*, 6:1419–1424.

[Stuckenschmidt et Timm, 2002] Stuckenschmidt, H. et Timm, I. (2002). Adapting communication vocabularies using shared ontologies. *Proceedings of the Second International Workshop on Ontologies in Agent Systems, Workshop at 1st International Conference on Autonomous Agents and Multi-Agent Systems*, pages 6–12.

[Sussna, 1993] Sussna, M. (1993). Word Sense Disambiguation for Free-text Indexing Using a Massive Semantic Network. *In* Bhargava, B. K., Finin, T. W. et Yesha, Y., éditeurs : *Proc. of the 2nd International Conference on Information and Knowledge Management (CIKM 93)*, pages 67–74, Washington, DC, USA. ACM.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

[Thieu *et al.*, 2004] Thieu, M., Steichen, O., Zapletal, E., Jaulent, M. et Bozec, C. L. (2004). Mesures de similarité pour l'aide au consensus en anatomie pathologique. *Ingénierie des Connaissances (IC)*.

[Touretzky, 1986] Touretzky, D. (1986). *The mathematics of Inheritance Systems*.

[Trojahn *et al.*, 2008] Trojahn, C., Quaresma, P. et Vieira, R. (2008). Conjunctive Queries for Ontology based Agent Communication in MAS. *In 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 829–836.

[Turney, 2006] Turney, P. (2006). Similarity of Semantic Relations. *Computational Linguistics*, 32(3):379–416.

[Valencia, 2000] Valencia, E. (2000). *Hétérogénéité Sémantique entre Agents modélisés en Logique de Descriptions*. Thèse de doctorat, Université de Paris XI.

[Valencia et Sansonnet, 2004] Valencia, E. et Sansonnet, J.-P. (2004). Building Semantic Channels between Heterogeneous Agents with Topological Tools. *In European Workshop on Multi-Agent Systems (EUMAS'04)*.

[Valtchev, 1999] Valtchev, P. (1999). *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse de doctorat, Université Grenoble 1, Grenoble.

[van Diggelen *et al.*, 2006a] van Diggelen, J., Beun, R., Dignum, F., van Eijk, R. et Meyer, J. (2006a). ANEMONE : an effective minimal ontology negotiation environment. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 899–906.

[van Diggelen *et al.*, 2006b] van Diggelen, J., Beun, R., Dignum, F., van Eijk, R. et Meyer, J. (2006b). Combining normal communication with ontology alignment. *LNCS*, 3859:181.

[van Diggelen *et al.*, 2006c] van Diggelen, J., de Jong, E. et Wiering, M. (2006c). Strategies for Ontology Negotiation : Finding the Right Level of Generality. *LNCS*, 3859:164.

[Ventresque *et al.*, 2008] Ventresque, A., Cazalens, S., Lamarre, P. et Valduriez, P. (2008). Enrichissement sémantique de requêtes utilisant un ordre sur les concepts. *In Atelier "Mesures de similarité sémantique", EGC'08*, pages 29–38.

[Viterbo *et al.*, 2008] Viterbo, J., Mazuel, L., Charif, Y., Endler, M., Sabouret, N., Breitman, K., El Fallah Seghrouchni, A. et Briot, J.-P. (2008). *Ambient intelligence : Management of distributed and heterogeneous context knowledge*. CRC Studies in Informatics Series. Chapman & Hall.

[Wahlster, 2003] Wahlster, W. (2003). SmartKom : Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. *Proc. of the Human Computer Interaction Status Conference*, pages 47–62.

[Watkins et Dayan, 1992] Watkins, C. et Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza - a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

[Winograd, 1972] Winograd, T. (1972). *Understanding Natural Language*. New York Academic Press.

[Wu et Palmer, 1994] Wu, Z. et Palmer, M. (1994). Verb semantics and lexical selection. *In 32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico.

[Yang et Powers, 2005] Yang, D. et Powers, D. M. W. (2005). Measuring semantic similarity in the taxonomy of wordnet. *In ACSC '05 : Proceedings of the Twenty-eighth Australasian conference on Computer Science*, pages 315–322, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Zhong *et al.*, 2002] Zhong, J., Zhu, H., Li, J. et Yu, Y. (2002). Conceptual graph matching for semantic search. *In ICCS '02 : Proceedings of the 10th International Conference on Conceptual Structures*, pages 92–196, London, UK. Springer-Verlag.

[Ziegler *et al.*, 2006] Ziegler, P., Kiefer, C., Sturm, C., Dittrich, K. et Bernstein, A. (2006). Detecting similarities in ontologies with the soqa-simpack toolkit. *In 10th International Conference on Extending Database Technology (EDBT 2006)*, volume 3896 de *Lecture Notes in Computer Science*, pages 59–76, Munich, Germany, March 26-31. Springer.

Annexes

Annexe A

La fonction $refine$

Cette annexe décrit l'algorithme de construction d'évènements en VDL. Son contenu est extrêmement dépendant de la sémantique opérationnelle du langage VDL. Ainsi, la lecture du tutoriel VDL semble un prerequis indispensable à la bonne compréhension de cette partie.[1]

A.1 Hypothèses

L'algorithme consiste à enrichir les évènements obtenus à partir des préconditions de subsomptions en utilisant les préconditions de structures. Les préconditions de contexte serviront à supprimer les évènements incohérents avec l'état courant. Nous donnons ici deux hypothèses importantes pour l'algorithme.

A.1.1 Hypothèse sur l'ensemble des nœuds témoins

Notation :

> Soit n un nœud booléen d'une précondition de structure et evt un évènement à compléter. On pose $L = evalNode(n, evt)$ alors l'*ensemble des nœuds témoins* T_L est défini tel que $T_L = \cup_{t \in L / t = \{\surd, f, g\}} \{t\}$. Moins formellement (et dans la mesure où l'algorithme n'est pas encore donné à ce niveau), l'*ensemble des nœuds témoins* est le sous-ensemble des fils d'un nœud booléen d'une précondition de structure tel que chaque nœud ne possède dans son interprétation complète aucune référence à un évènement externe.

Nous posons maintenant une hypothèse sur le contenu des nœuds booléen (type *equals, is-a,* etc.) :

Hypothèse :

> $\forall L, T_L \neq \emptyset$. L'ensemble des nœuds témoins de chaque nœud booléen d'une précondition de structure n'est jamais vide.

[1] Vous pouvez trouver ce manuel à l'adresse `http://www-poleia.lip6.fr:8180/~sabouret/demos/html/documentation.html`

Ces nœuds donnent une valeur de référence pour les autres nœuds dépendant de l'évènement. C'est cette valeur qui permet, dans la plupart des cas, d'instancier correctement l'évènement. En cas de non respect de cette hypothèse, l'algorithme ne peut pas conclure au sujet de cette précondition de structure, il génère alors une liste d'évènement vide. Illustrons l'importance de cette hypothèse par un exemple :

Exemple :

```
<equals>
  <event-get>
    <take/>
  </event-get>
  <value>
    5
  </value>
</equals>
```

Dans ce style de situation, `<value>5</value>` est un nœud témoin. Ce nœud donne donc la valeur attendue pour `<event-get><take/></event-get>`. Sans ce nœud, il ne serait pas possible de conclure sur la structure de l'évènement.

A.1.2 Liens entre les préconditions

Pour construire un évènement, nous nous servons des préconditions de structure et de subsomption. Nous sommes amenés a poser l'hypothèse suivante :

Hypothèse :

> Soit P_{st} une précondition de structure. Soit P_{sb} une précondition de subsomption et soit $P_{st_{event-get}}$ le nœud référence vers l'évènement. Alors :
>
> $$\forall p_{sb} \in children(P_{sb}),\ calcRef(children(P_{st_{event-get}}), p_{sb}) \neq \emptyset$$

Ceci signifie en pratique, qu'il existe un lien entre les préconditions « de structure » et les préconditions « de subsomption ». Sans cette hypothèse, il est possible de construire une infinité d'évènement amenant à la même action (i.e. l'utilisateur peut entrer une infinité d'évènement correct).

L'heuristique que nous proposons (actuellement considérée dans notre algorithme) dans le cas de non-respect de l'hypothèse, l'évènement choisi est celui ayant la plus petite profondeur (dans l'exemple suivant l'évènement 1)[2].

Exemple :

Prenons le cas où l'hypothèse n'est pas respectée. Nous allons construire deux évènements corrects, amenant au même résultat, ce qui est dû à l'incertitude sur la structure exacte de

[2] Ce choix n'a pas de réel importance, puisque tous les messages sont équivalents. Il semble néanmoins logique de penser que l'évènement le plus petit est le plus clair à comprendre.

l'évènement. La génération peut se poursuivre a l'infinie.

Préconditions P_{sb} de subsomption :

```
<event>
  <take/>
</event>
```

Préconditions P_{st} de structure :

```
<guard>
  <equals>
    <event-get>
      <object/>
    </event-get>
    <circle/>
  </equals>
</guard>
```

Dans ce cas, il n'y a aucun lien entre les deux. Nous pouvons construire (par exemple) les évènements corrects suivants :

Évènement 1 :

```
<take>
  <object>
    <circle/>
  </object>
</take>
```

Évènement 2 :

```
<take>
  <noeud1>
    <object>
      <circle/>
    </object>
  </noeud1>
</take>
```

A.2 Algorithme

A.2.1 Fonctions utiles

Ce paragraphe définit un certain nombre de fonctions utiles pour la définition des algorithmes de génération d'évènements.

A.2.1.1 La fonction *copyStructure*

Soit $E, F \in \Upsilon$, la fonction $copyStructure : \Upsilon \times \Upsilon \longrightarrow \Upsilon$ permet de créer un nœud flottant possédant la structure de E mais possédant aussi les attributs correspondant a F. La relation $E \succeq F$ doit donc être vérifiée. La fonction est définie comme suit :

$copyStructure(E, F) =$

$$\begin{cases} \surd & \text{Si } E \not\succeq F \text{ ou } E = \surd \text{ ou } F = \surd \\ node & \text{Sinon} \end{cases}$$

Tel que la définition de $node \in \Upsilon$ soit :

$$\begin{cases} tag_{node} = tag_E \\ attributes(node) = attributes(F) \\ content(node) = content(F) \\ children(node) = \cup_{e' \in children(E)} copyStructure(e', f'),\ f' \in children(F) \text{ et } tag_{e'} = tag_{f'} \end{cases}$$

En clair, le résultat est exactement la structure de E mais chaque nœud possède les attributs des nœuds correspondant dans F.

Exemple :

```
A = <a></b></a> et B = <a name=''toto''><b name=''tutu''/><c/></a>
copyStructure(A,B) = <a name=''toto''><b name=''tutu''></a>
```

Propriété

$E \succeq_{attr} copyStructure(E, F) \succeq F$

L'opérateur $\succeq_{attr}$ est défini dans la section A.2.2.1.

A.2.1.2 La fonction *compareNode*

Soit $E, F \in \Upsilon$ et $tag \in A^*$, la fonction $compareNode : \Upsilon \times \Upsilon \times A^* \longrightarrow \mathbb{B}$ permet de vérifier si E et F suivent la relation définie par tag. Elle est définie ainsi :

$compareNode(E, F, tag) =$

$$\begin{cases} true & \text{Si } tag = equals \text{ et } E = F \\ true & \text{Si } tag = is - a \text{ et } tag_E = tag_F \\ false & \text{Sinon} \end{cases}$$

Cette fonction permet d'écrire facilement des tests sur deux nœuds et de généraliser une partie de l'algorithme. En effet, l'interprétation des nœuds *equals* et *is-a* est équivalente à l'exception de la comparaison nœud par nœud. L'algorithme énoncé dans la section A.2.3.2 est donc commun, au paramètre tag de cette fonction près.

A.2.1.3 La fonction *searchEvent*

Soit $E \in \mathcal{P}(\Upsilon \times \Upsilon \times \mathcal{P}(\Upsilon))$ (ensemble définie dans l'entête de la section A.2.3), la fonction $searchEvent : \mathcal{P}(\Upsilon \times \Upsilon \times \mathcal{P}(\Upsilon)) \longrightarrow \Upsilon$ recherche parmi le premier élément e du triplet t dans la liste L, un nœud tel que :

$searchEvent(L) =$

$$\begin{cases} e & \text{Si } \exists\{e,f,g\} \in L/e \neq \surd \\ \surd & \text{Sinon} \end{cases}$$

Cette fonction, pendant l'interprétation d'un nœud n permet de récupérer le contenu d'un *event-get* (s'il existe) parmi l'interprétation des enfants de n, afin de le transmettre l'or de l'interprétation de n.

Remarque :

1. Cette fonction n'existe pas dans l'implémentation, les calculs correspondant n'ayant pas de raison d'être séparés explicitement. Elle est introduite ici à titre de simplification pour l'écriture de l'algorithme.
2. $tag_e = event - get$, par construction de l'algorithme (section A.2.3.1).
3. Normalement, il ne devrait exister qu'un seul nœud e au maximum répondant au critère donné. Autrement dit, il n'y devrait y avoir qu'un seul nœud *event-get* dans une pré-condition.

A.2.2 Calcul des références

Le calcul des références énoncées dans VDL permet de déterminer un ensemble de nœud de la vue à partir d'une référence VDL (le principe d'une référence VDL est expliqué dans le Tutoriel VDL). Néanmoins, il ne permet pas de mémoriser le chemin suivi pour trouver cet ensemble. On appelle *référence instanciée*, une référence possédant la structure XML d'une autre référence, mais dont chaque nœud peut contenir plus d'attributs. Ces deux références sont unies par une relation de subsomption, mais avec des contraintes de structures supplémentaires, ce qui en fait une relation de subsomption réduite. Nous introduisons donc la notion de *subsomption faible.*

A.2.2.1 Subsomption faible

On note cette relation $\preceq_{attr}$(par complémentarité a la notation $\preceq$ de la subsomption). Elle est définie de la manière suivante :

Soit $X = children(X)$ et $Y = children(Y), \forall(x,y) \in \Upsilon^2, x \succeq_{attr} y$ et si seulement si :

$$\begin{cases} tag(x) = tag(y) \\ attributes(x) \subseteq attributes(y) \\ content(x) = content(y) \\ |X| = |Y| \\ \forall x_i \in X/i \in [1,...,|X|], x_i \succeq_{attr} y_i \end{cases}$$

En d'autre termes, $x \succeq_{attr} y$ si x et y sont identiques aux attributs près, x en possédant autant ou moins que y.

Exemples :

<b/> $\preceq_{attr}$ <a><b/>
<b att=''two''/> $\preceq_{attr}$ <a><b/>

Propriété :

« Si $x \succeq_{attr} y$, alors $x \succeq y$ ». La réciproque est fausse.

Démonstration de la propriété :

1. Nous faisons une démonstration par récurrence. Supposons l'hypothèse suivante vraie au rang n : P_n :"Si n est la profondeur maximum de deux arbres VDL t_1 et t_2, et que $t_1 \succeq_{attr} t_2$, alors $t_1 \succeq t_2$"

   (a) Démontrons d'abord le rang initial P_1 :
   t_1 et t_2 n'ont pas de fils (profondeur 1), alors comme $t_1 \succeq_{attr} t_2$, on a :

   $$\begin{cases} tag(t_1) = tag(t_2) \\ attributes(t_1) \subseteq attributes(t_2) \\ content(t_1) = content(t_2) \end{cases}$$

   Donc $t_1 \succeq t_2$.

   (b) Supposons maintenant P_n vraie au rang n, démontrons alors P_{n+1} :
   Soit t_1 et t_2 de profondeur maximum $n+1, T_1 = children(t_1)$ et $T_2 = children(t_2)$, comme $t_1 \succeq_{attr} t_2$, on a :

   $$\begin{cases} tag(t_1) = tag(t_2) \\ attributes(t_1) \subseteq attributes(t_2) \\ content(t_1) = content(t_2) \\ |T_1| = |T_2| \\ \forall t_1^i \in T_1 / i \in [1, ..., |T_1|], t_1^i \succeq_{attr} t_2^i \end{cases}$$

   Or, par hypothèse P_n, les nœuds fils de t_1 et t_2 sont de profondeur n, donc la dernière condition devient :

   $$\forall t_1^i \in T_1 / i \in [1, ..., |T_1|], t_1^i \succeq t_2^i$$

   Donc, $\forall t_1^i \in T_1 / i \in [1, ..., |T_1|], \exists t_2^j \in T_2 / j \in [1, ..., |T_2|], t_1^i \succeq t_2^j$. Ce qui correspond à la dernière hypothèse de la subsomption, donc $t_1 \succeq t_2$.
   Donc si P_n est vraie, alors P_{n+1} est vraie.

   La relation de récurrence est vérifiée, donc si $x \succeq_{attr} y$, alors $x \succeq y$.

2. Posons comme hypothèse la propriété "Si $x \succeq y$ alors $x \succeq_{attr} y$" vraie, comme :

   <a/> $\succeq$ <a><b/>

   alors

<a/> $\succeq_{attr}$ <a><b/>

Ce qui est faux, donc l'hypothèse est fausse.

A.2.2.2 Calcul des références instanciées, la fonction $findRef$

Soit $r = \{e_1, ..., e_n\} \in \mathcal{P}(\Upsilon)$ une référence et $T \in \Upsilon$un élément. La fonction

$$\begin{array}{rlll} findRef: & P(\Upsilon) \times \Upsilon \rightarrow & P(\Upsilon \times P(\Upsilon)) \\ & (r, T) & Res \end{array}$$

calcule le résultat de la référence r dans l'élément T et la référence instanciée selon le chemin suivi pour un résultat dans l'ensemble Res des couples de type $(N, \{d_1, ..., d_n\})$ tel que :

1. $calcRef(\{d_1, ..., d_n\}, T) = N$
2. $\forall i \in \{1, ..., n\}, d_i \preceq_{attr} e_i$

La fonction $findRef(r, T)$ est définie récursivement de la façon suivante :

$findRef(\{e_1, ..., e_n\}, T) =$

$$\begin{cases} \emptyset & \text{Si } n = 0 \text{ ou } tag(T) \in I_{calcRef} \\ \{T, \emptyset\} & \text{Si } n = 1 \text{ et } e_1 \succeq T \\ \cup_{t' \in children(T)} \{A, \{copyStructure(e_1, T)\} \cup B\} & \text{Si } n > 1 \text{ et } e_1 \succeq T \\ \quad / findRef(\{e_2, ..., e_n\}, t') = \{A, B\} & \\ \cup_{t' \in children(T)} findRef(\{e_1, ..., e_n\}, t') & \text{Sinon} \end{cases}$$

Remarques

1. Comme *calcRef* la référence peut être constitué de nœuds flottants ou de nœuds intégrés à la vue. Par contre, les résultats sont toujours des nœuds de la vue (donc non-flottants) ce qui garantit le nombre fini de résultats. Néanmoins, le résultat retourné contient uniquement des nœuds flottants, copie de la solution incluant les nœuds de la vue, l'objectif principal étant de les insérer dans des évènements en cours de construction (donc disjoints de la vue).
2. $findRef$ suit le même parcours que *calcRef* et calcule le même résultat mais mémorise en plus au fur et à mesure le chemin suivi. Il est plus efficace de fusionner la recherche du résultat *calcRef* (aussi utile ici pour le résultat final) et la recherche des références instanciées.

A.2.3 La nouvelle interprétation des nœuds

Nous introduisons ici l'algorithme permettant d'interpréter une précondition VDL (i.e. un arbre VDL) en tenant compte du nouvel objectif (la génération d'évènement) et donc de la nouvelle méthode de calcul des références (section précédente). La nouvelle interprétation conserve le résultat de la méthode ζ, mais calcule en plus le nœud complémentaire à ce résultat pour valider la précondition. Par exemple, en reprenant l'exemple de la section A.1.1, le nœud

equals ne sert plus à vérifier l'égalité mais à affecter la valeur `<value>5</value>` au nœud *event-get*.

Nous définissons donc la fonction $evalNode : \Upsilon \times \Upsilon \longrightarrow \mathcal{P}(\Upsilon \times \Upsilon \times \mathcal{P}(\Upsilon))$ capable d'interpréter un nœud e par rapport à un évènement evt. Nous distinguons les différentes interprétations en fonction de tag_e. Un triplet du résultat sera constitué de la manière suivante :

1. La référence *event-get* : Cette référence correspond au nœud *event-get* rencontré pendant le parcours, mais dont ses fils sont interprétés (*i.e.* il représente une référence sur l'évènement) qui sera rencontré pendant le parcours. Ce nœud est un nœud VDL vide ($\surd$) tant qu'aucun *event-get* n'a été rencontré.
2. Le résultat normal : Ce résultat est celui de la méthode ζ pour les mêmes paramètres.
3. Le chemin suivi : Ce chemin correspond aux chemins instanciés trouvés grâce à la méthode $findRef$.

A.2.3.1 Les nœuds à référence

– Si $tag_e = get$,
Soit $en = \cup_{i=1}^{n} evalNode(e_i, evt) = \cup_{i=1}^{|en|}\{en^i_{evt}, en^i_{node}, en^i_{ref}\}$, soit $f = \cup_{i=1}^{|en|} en^i_{node}$, alors on pose $s = findRef(f, view(e)) = \cup_{i=1}^{|s|}\{s^i_{node}, s^i_{ref}\}$. Alors,

$$evalNode(e, evt) =$$

$$\cup_{i=1}^{|s|}(\cup_{c \in children(s^i_{node})}\{searchEvent(en), c, s^i_{ref} \cup (tag_{s^i_{node}}\{attributes(s^i_{node})\})\})$$

Le principe est de transférer la racine nœud du résultat de $findRef$ en fin de liste de la référence. Nous obtenons ainsi le résultat habituel par ζ d'un *get* (puisqu'il contient alors les fils du nœud résultat) et nous gardons le chemin pour arriver à ce résultat. Nous ne perdons pas le nœud racine du nœud résultat, perdu avec l'interprétation normal du get.

Exemple :

Nœud dans le code	Requête *get*	Résultat *calcRef*

`<content name=''test''>`
`  <a>`
`  <b/>`
`  <c/>`
`  </a>`
`</content>`
	`<get>`
	`  <content/>`
	`  <a/>`
	`</get>`
		{
		`<a>`
		`  <b/>`
		`  <c/>`
		`</a>`
		}

Résultat ζ	Résultat *evalNode*	
{ <b/>, <c/> }	{ {<content name="test"/>, <a/>}, {<content name="test"/>, <a/>} }	{ <b/>, <c/> }

Comme on le voit dans cet exemple, le résultat de la fonction ζ est inclus dans le résultat de *evalNode*. Nous gardons en plus le chemin pour atteindre ces nœuds ce qui implique dans notre cas les nœuds `<content/>` et `<a/>`.

Remarque sur l'exemple :

L'apparition d'un doublet dans le tableau au lieu d'un triplet permet de simplifier l'écriture (la référence à l'évènement $searchEvent(en)$ n'y figure pas).

- Si $tag_e = size - ref$,
  Soit $en = \cup_{i=1}^{n} evalNode(e_i, evt) = \cup_{i=1}^{|en|}\{en^i_{evt}, en^i_{node}, en^i_{ref}\}$, soit $f = \cup_{i=1}^{|en|} en^i_{node}$, alors on pose $s = findRef(f, view(e)) = \cup_{i=1}^{s_{num}}\{s^i_{node}, s^i_{ref}\}$ avec $s_{num} = |s|$. On pose ensuite le nœud VDL $s_{vdl} =< value > s_{num} < /value >$. Soit $t = \cup_{i=1}^{s_{num}}(s^i_{ref} \cup \{s^i_{node}\})$. Alors,

  $evalNode(e, evt) =$

  $$\cup_{i=1}^{|t|}\{searchEvent(en), s_{vdl}, t_i\}$$

  L'interprétation normale permet de rendre le nombre d'éléments existant par rapport à la référence. Dans notre cas, nous générerons tous les résultats possibles à partir de la référence incomplète. La valeur s_{num} n'est donc pas représentative du résultat obtenu normalement par ζ. Elle donne juste un indicateur pour savoir si il existe ou non des nœuds correspondant aux critères (i.e. Si $s_{num} = 0$, alors aucun nœud ne correspond aux critères, pas d'évènement correct au final). C'est l'algorithme suivant (partie *is-a* et *equals*).
- Si $tag_e = event - get$,
  Le fonctionnement du *event-get* est équivalent au fonctionnement normal. Néanmoins, le nœud *event-get* avec son contenu interprété est sauvegardé afin de faciliter la recherche de l'emplacement d'insertion des instances une fois l'interprétation terminée.
  Soit $en = \cup_{i=1}^{n} evalNode(e_i, evt) = \cup_{i=1}^{|en|}\{en^i_{evt}, en^i_{node}, en^i_{ref}\}$, soit $f = tag_e[\cup_{i=1}^{|en|} en^i_{node}]$, soit $R = \cup_{r \in calcRef(f,e)} children(r)$. Alors,

  $evalNode(e, evt) =$

  $$\begin{cases} \{\emptyset, evt, \emptyset\} & \text{Si } children(f) = \emptyset \\ \{f, \surd, \emptyset\} & \text{Si } R = \emptyset \\ \cup_{r \in R}\{f, r, \emptyset\} & \text{Sinon} \end{cases}$$

A.2.3.2 Les nœuds à retour booléen

Dans un premier temps, nous supposerons que les nœuds à retour booléen sont les fils directs du nœud *guard* ou les fils directs des nœuds *and, or* et *not*[3]. A ce titre, le retour de la méthode *evalNode* appliquée à ce type de nœud est $P(\Upsilon)$ (la liste des évènements générés avec cette précondition booléenne).

- Si $tag_e = equals$ ou $tag_e = is - a$:
  Ces deux tags sont regroupés car le traitement est similaire, à la primitive de comparaison de nœud près (voir la définition de la méthode *compareNode*, section A.2.1.2). Comme nous le disions, **le retour de *evalNode* dans ce cas est une liste d'évènements $\mathcal{P}(\Upsilon)$.**
  Soit $en = \cup_{i=1}^{n} evalNode(e_i, evt)$. Soit l'ensemble des nœuds témoins (voir hypothèse 1.2.1) $T_{en} \in en$. On choisi arbitrairement $t = \{\surd, t_{node}, t_{ref}\} \in T_{en}$. Soit de plus l'égalité $e_{ref} = children(searchEvent(en))$.

  Dans les cas suivants, $evalNode(e, evt) = \emptyset$:

  $$\begin{cases} \text{Si } T_{en} = \emptyset \\ \text{Si } \exists f \in T_{en} / compareNode(f_{node}, t_{node}, tag_e) = false \\ \text{Si } \exists f \in en / tag_{f_{node}} = value \text{ et } content(f_{node}) = \text{"0"} \end{cases}$$

  Dans ces situations en effet, le résultat est inconsistant avec l'interprétation des nœuds. Le cas $T_{en} = \emptyset$ correspond au non-respect de l'hypothèse 1.2.1 sur les nœuds témoins.

  Dans les autres cas, on note la liste des valeurs à insérer $E \in \mathcal{P}(\Upsilon)$=

  $$\cup_{f=\{f_{evt}, f_{node}, f_{ref}\} \in en/t} \begin{cases} t_{node} & \text{Si } f_{node} = \surd \\ f_{ref} & \text{Si } tag_{f_{node}} = value \text{ et } content(f_{node}) \neq \text{"0"} \\ f_{ref} & \text{Si } compareNode(f_{node}, t_{node}, tag_e) = true \\ \surd & \text{Sinon} \end{cases}$$

  Alors, $evalNode(e, evt) = createEvent(evt, e_{ref}, E)$(définition de *createEvent* dans la section A.2.4.1). Le principe est de constituer une liste E d'éléments à insérer dans l'évènement.

- Si $tag_e = and$:
  On considère dans ce cas que les préconditions sont successives. On les traite donc séparément, à la manière définie pour le traitement de plusieurs *guard* (voir section A.3).
  Exemple d'équivalence sans *and :*

```
<action>                    <action>
  <guard>                     <guard><X1/><guard/>
    <and>                     <guard><X2/><guard/>
      <X1/>          <=>      <Y1/>
```

[3] Nous laissons en perspectives la possibilité d'une recherche de généralisation en fonction de l'efficacité de l'algorithme à l'usage.

```
      <X2/>                <action/>
    </and>
  </guard>
  <Y1/>
</action>
```

Définissons l'ensemble récursif R_n tel que :

$$R_n = \cup_{r \in R_{n-1}} evalNode(e_n, r)$$

avec $\forall n \in [1, .., |children(e)|], e_n \in children(e)$ et $R_0 = \{evt\}$. On a alors :

$$evalNode(e, evt) = R_{|children(e)|}$$

Le principe ici est de dire que les évènements doivent être créés par analyse cumulée des préconditions avec comme objectif de spécialiser toujours plus les évènements. Les évènements en sortie de l'analyse d'une précondition sont donc l'entrée de l'analyse de la précondition suivante.

- Si $tag_e = or$:
  Le principe est de considérer qu'il est possible de construire des évènements différents pour chaque *guards*.
  Exemple d'équivalence sans *or :*

```
<action>                    <action>
  <guard>                     <guard><X1/><guard/>
    <or>                      <Y1/>
      <X1/>          <=>    <action/>
      <x2/>                 <action>
    </or>                     <guard><X2/></guard>
  </guard>                    </Y1>
  <Y1/>                     </action>
</action>
```

  On calcule :

$$evalNode(e, evt) = \cup_{e' \in children(e)} evalNode(e', evt)$$

- Si $tag_e = not$:
  Le modèle actuel ne tient pas compte des tags *not*. Il semble difficile de récupérer des informations concrètes facilement. En effet, la génération serait alors par négation d'instances, l'ensemble final doit donc être fini et calculé à l'avance pour que l'information soit utilisable.
  Dans certains cas, le *not* pourra permettre d'éliminer des nombres impossibles, mais le modèle actuel ne tient pas compte des évènements à valeur numériques (voir les perspectives).
- Si $tag_e = greater - than$:

Le modèle actuel ne tient pas compte des évènements à valeur numérique. Ce modificateur sera donc étudié au moment de l'introduction du typage des nombres (voir les perspectives, section A.3).

A.2.3.3 Les autres nœuds

- Si $tag_e = size$,
  Soit $en = \cup_{i=1}^{n} evalNode(e_i, evt) = \cup_{i=1}^{|en|}\{en^i_{evt}, en^i_{node}, en^i_{ref}\}$, soit $f = \cup_{i=1}^{|en|} en^i_{node}$, alors on pose $s = evalNode(f, evt) = \cup_{i=1}^{s_{num}}\{s^i_{node}, s^i_{ref}\}$ avec $s_{num} = |s|$. On pose ensuite le nœud VDL $s_{vdl} =< value > s_{num} < /value >$. Soit $t = \cup_{i=1}^{s_{num}}(s^i_{ref} \cup \{s^i_{node}\})$.
  Alors, $evalNode(e, evt) = \cup_{i=1}^{|t|}\{s_{vdl}, t_i\}$.
  Le traitement est équivalent a celui du $size-ref$. La seule différence est que le contenu du nœud e n'est pas une référence.
- Si tag_eest un modificateur connu,
  Soit $L = \zeta(e, evt, \surd)$, alors $evalNode(e, evt) = \cup_{i=1}^{|L|}\{\nabla, L_i, \emptyset\}$.
  Ce cas ne devrais pas se produire, les autres modificateurs (add, put, random, etc.) n'apparaissent pas (a priori) dans les préconditions.
- Sinon (i.e. l'interprétation de e n'est pas définissable comme une opération de nœud), soit $E = \cup_{c \in children(e)} evalNode(c, evt)$, alors,

$$evalNode(e, evt) = tag_e[\cup_{e' \in E}\{searchEvent(E), e'_{node}, e'_{ref}\}]$$

Dans le cas où $E = \emptyset$ ou si $children(e) = \emptyset$, soit f une copie de e (avec le même contenu texte si il existe), alors

$$evalNode(e, evt) = \{\nabla, f, \emptyset\}$$

A.2.4 La création des évènements

A.2.4.1 La fusion évènement - valeurs

Soit $e \in \Upsilon$ un évènement à compléter, $e_{ref} \in \mathcal{P}(\Upsilon)$ une référence sur un emplacement dans cet évènement et $T \in \mathcal{P}(\mathcal{P}(\Upsilon))$ une liste de valeurs à insérer dans l'évènement. Alors la fonction $createEvent(e, e_{ref}, T) \longrightarrow \mathcal{P}(\Upsilon)$ crée une liste d'évènement à partir de chaque valeur de T en complétant l'évènement e a l'emplacement e_{ref}. On définie alors la fonction $createEvent$ tel que :

$createEvent(e, e_{ref}, T) =$

$$\cup_{t \in T} \begin{cases} e & \text{Si } t = \emptyset \\ e/children(calcRef(e_{ref}, e)) = t & \text{Si } f_{child} = \emptyset \text{ et } t \neq \emptyset \\ e/children(calcRef(e_{ref}, e)) = \cup_{t' \in t / \exists c \in f_{child} / tag_c = tag_{t'}} t' & \text{Sinon} \end{cases}$$

avec $f_{child} = children(calcRef(e_{ref}, e))$ en début d'algorithme.

Remarques :

1. Si au début de l'algorithme $calcRef(e_{ref}, e) = \emptyset$, alors l'hypothèse A.1.2 n'est pas respectée. L'heuristique alors mise en place consiste à insérer le nœud t sur la première branche sans fils en prenant récursivement le premier fils.
2. De manière moins formelle, l'algorithme remplace les nœuds à l'emplacement désigné par la référence e_{ref}de l'évènement e par les nœuds de $t \in T$, si ces nœuds étaient référencés à cet emplacement (si l'emplacement ne contient aucun nœud, alors t est recopié en entier).
3. Il existe donc au plus $|T|$ évènements générables à partir d'un appel à la méthode *createEvent*.

A.2.4.2 Le parcours du code VDL

Soit $A \in \mathcal{P}(\Upsilon)$ l'ensemble des actions n'ayant pas d'ascendant *action*. On définit les quatre ensembles suivants pour une instance $a \in A$ (ces quatre ensembles sont définis de façon informelle dans la section 2.2.1) :

1. $P_{sub} \in children(a) / \forall e \in P_{sub}, tag_e = event$
2. $P_{str} \in children(a) / \forall e \in P_{str}, tag_e = guard$ et $\exists n \in \mathbb{N} / \exists e_1, .., e_n$tel que :
   (a) $\forall i \in [1, n-1], e_{i+1} \in children(e_i)$
   (b) $e_1 = e$
   (c) $tag(e_n) = event - get$
3. $P_{ctx} \in children(a) / \forall e \in P_{ctx}, tag_e = guard$ et $e \notin P_{str}$
4. $A_{sub} \in children(a) / \forall e \in A_{sub}, tag_e = action$

Nous définissons ici la fonction récursive $processAction(a, e) : \Upsilon \times \Upsilon \longrightarrow \mathcal{P}(\Upsilon)$ où $a \in A$ est une action (ou une sous-action) et $e \in \Upsilon$ est l'évènement en cours de traitement, dans le cas a est une sous-action est que l'on donne donc l'évènement construit a partir de l'action parente. Le résultat de l'analyse de tout le code de l'agent VDL est donc $\cup_{a \in A} processAction(a, \surd)$. L'algorithme de parcours se décompose en plusieurs étapes :

- La vérification contextuelle : une action ne peut se déclencher que si le contexte de l'agent le permet (avec ou sans évènement externe). Cette condition est validée uniquement si les préconditions de contexte sont valides. Donc,

$$\begin{cases} \text{Retour } \emptyset & \text{Si } \exists v \in (\cup_{e \in P_{ctx}} \zeta(e, \surd, \surd)) / content(v) = false \\ \text{Continuer} & \text{Sinon} \end{cases}$$

- Génération de la liste des évènements à compléter : Les évènements à compléter peuvent venir de deux sources :
  1. De la liste P_{sub} si $P_{sub} \neq \emptyset$
  2. Du paramètre e si $e \neq \surd$[4].

[4] $e = \surd$si $a \in A$ (i.e. l'action en cours de traitement n'a pas d'ascendant *action*) ou si les traitements récursifs précédents n'ont pas généré d'évènements (assez peu courant en pratique).

Exemple :

```
<action> // action 1
   <event><take/></event>
   ... // precondition(s) guard de l'action 1 : P1
   <action> // action 1.1
      <event><take><object/></take></event>
      ... // precondition(s) guard de l'action 2 : P2
      ... // modification(s)
   </action>
   ... // modification(s)
</action>
```

Dans notre exemple, en entrant dans l'action 1.1, la liste P_{sub} contient le nœud VDL `<take><object/></take>`, le paramètre e contient l'un des évènements générés grâce à `<take>` (nœud *event* de l'action 1) et l'analyse des préconditions P1. Pour l'analyse des préconditions P2, il faut donc tenir compte de ces deux sources, susceptibles de fournir de l'information. Pour un élément $p \in P_{sub}$, trois situations peuvent apparaître par rapport à e :

1. e est plus spécifique que p (i.e. $p \succeq e$). Dans ce cas, l'analyse de P1 a permis de produire des évènements cohérents avec l'action 1.1. Il est donc préférable de continuer l'analyse de P2 avec l'évènement e.
2. p est plus spécifique que e (i.e. $e \succeq p$). Dans ce cas, l'évènement donné dans la clause *event* de l'action 1.1 donne plus d'informations que l'analyse de P1 n'a pu en donner. Il est donc préférable de continuer l'analyse de P2 avec l'évènement p.
3. e et p n'ont pas de liens particuliers (i.e. $e \not\succeq p$ et $p \not\succeq e$). Alors, il y a incohérence, l'évènement e généré par P1 ne peut pas valider la sous-action 1.1. Ce cas se produit forcement dans le cas de plusieurs sous-actions se répartissant les évènements (exemple, action *drop* de l'agent *jojo*). Il n'y a donc rien à analyser.

La liste des évènements à compléter est donc définie par :

$$E_{com} = \begin{cases} P_{sub} & \text{Si } e = \surd \\ \{e\} & \text{Si } P_{sub} = \emptyset \\ \cup_{p \in P_{sub}} \begin{cases} e & \text{Si } e \succeq p \\ p & \text{Si } p \succeq e \\ \surd & \text{Sinon} \end{cases} & \text{Sinon} \end{cases}$$

- L'analyse de E_{str} : Définissons l'ensemble récursif $R_n = \cup_{r \in R_{n-1}} evalNode(e_n, r)$ avec $\forall n \in [1, .., |E_{str}|], e_n \in E_{str}$ et $R_0 = E_{com}$. On calcule $E_{event-str} = R_{|E_{str}|}$. Le principe ici est de dire que les évènements doivent être créés par analyse cumulées des préconditions avec comme objectif de spécialiser toujours plus les évènements. Les évènements en sortie de l'analyse d'une précondition sont donc l'entrée de l'analyse de la précondition suivante.
- L'analyse de A_{sub} : On calcule $E_{event-sub} = \cup_{a \in A_{sub}} (\cup_{e \in E_{event-str}} processAction(a, e))$.

A la différence de l'analyse des préconditions de structure, les sous-actions sont disjointes, et chacune crée son propre ensemble d'évènement.

- Calcul de l'ensemble final : Les modifications que produisent une action peuvent être situées à plusieurs niveaux. En effet, Dans le cas d'une action constituée de plusieurs sous-actions, chaque sous-action peut être capable de produire des modifications indépendamment des autres. Dans certains cas, il est possible d'avoir donc un évènement déclenchant toutes les sous-actions possibles et un évènement ne déclenchant que la première action (quand un évènement est bloqué à un niveau, il ne peut pas descendre dans les niveaux suivants). Il faut en tenir compte pour savoir si les évènements trouvés sans tenir compte du calcul sur les sous-actions doivent être ajoutés en plus des sous-actions :

  Retour final =

$$\begin{cases} E_{event-str} \cup E_{event-sub} & \text{Si } \exists e \in children(a)/BasicAction(e) = true \\ E_{event-sub} & \text{Sinon} \end{cases}$$

A.3 Perspectives d'améliorations

L'algorithme actuel ne fonctionne que si la liste des évènements possibles à l'instant courant est finie. Ainsi, si l'évènement a besoin d'un paramètre « chaîne de caractères », mais sans contraintes particulière, il n'est pas possible de modéliser ce résultat. De même, une contrainte liée aux nombres (*e.g.* une contrainte dont la sémantique serait « $2 \geq x \geq 0$ ») ne sera pas gérée.

Ces modifications impliquaient au préalable une gestion des types au sens de VDL qui n'existait pas. La gestion des types vient d'être intégrée au modèle avec les types suivants :

- *string* : C'est le type par défaut. Les nœuds avec le type *string* (ou sans déclaration de type) peuvent contenir tout type de contenu CDATA.
- *number* : doit contenir un nombre.
  Remarque : Si une action tente d'écrire une valeur de type *string*, le type du nœud sera automatiquement converti en *string*. Cette remarque s'applique aux deux prochains types.
- *integer* : doit contenir un nombre entier (positif ou négatif).
- *boolean* : doit contenir un nœud booléen (true ou false).

L'utilisation des types dans VDL donne des nœuds de la forme :

```
<my-variable name="toto" type="integer">3</my-variable>
<value type="boolean">true</value>
```

La modification des algorithmes doit prendre en compte le fait qu'un évènement puisse être décliné à l'infini et proposer des schémas d'évènements avec des règles d'instanciation claires et précises. Ce travail reste à faire et à concevoir.

Annexe B

Code VDL de l'agent « Jojo »

```
<view>
  <!-- Display -->
  <display>
    <outer-box>
      <cell><object>
        <shape>square</shape>
        <color>white</color>
        <blockSize>medium</blockSize>
        <image>examples/jojo/fig1.gif</image>
      </object></cell>
      <cell><object>
        <shape>circle</shape>
        <color>white</color>
        <blockSize>small</blockSize>
        <image>examples/jojo/fig2.gif</image>
      </object></cell>
      <cell><object>
        <shape>triangle</shape>
        <color>red</color>
        <blockSize>big</blockSize>
        <image>examples/jojo/fig3.gif</image>
      </object></cell>
      <cell><object>
        <shape>square</shape>
        <color>red</color>
        <blockSize>small</blockSize>
        <image>examples/jojo/fig4.gif</image>
      </object></cell>
      <cell><object>
        <shape>circle</shape>
        <color>blue</color>
```

```
      <blockSize>medium</blockSize>
      <image>examples/jojo/fig5.gif</image>
    </object></cell>
    <cell><object>
      <shape>square</shape>
      <color>green</color>
      <blockSize>medium</blockSize>
      <image>examples/jojo/fig6.gif</image>
    </object></cell>
    <cell><object>
      <shape>triangle</shape>
      <color>green</color>
      <blockSize>tiny</blockSize>
      <image>examples/jojo/fig7.gif</image>
    </object></cell>
  </outer-box>
  <newline/>
  <text>Hand :</text>
  <in-hand>
    <nothing><text>nothing</text></nothing>
  </in-hand>
  <newline/>
  <grid>
    <line name="upper">
      <row name="left">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="center">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="right">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
    </line>
    <newline/>
    <line name="middle">
      <row name="left">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="center">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="right">
        <cell><image>examples/jojo/cell.gif</image></cell>
```

```
      </row>
    </line>
    <newline/>
    <line name="lower">
      <row name="left">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="center">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
      <row name="right">
        <cell><image>examples/jojo/cell.gif</image></cell>
      </row>
    </line>
    <newline/>
  </grid>
</display>
<!-- Actions -->
<action>
  <name>drop object to a cell</name>
  <event> <!-- event structure -->
    <drop><position><line/><row/></position></drop>
    <drop><position><outer-box/></position></drop>
  </event>
  <guard> <!-- hand must not be empty -->
    <not> <is-a> <get><in-hand/></get> <nothing/> </is-a> </not>
  </guard>
  <action>
    <name>Drop to outer-box</name>
    <guard>
      <equals><event-get><position/></event-get><outer-box/></equals>
    </guard>
    <!-- move the object to outer-box (creates the cell) -->
    <add>
      <path><outer-box/></path>
      <cell><get><in-hand/></get></cell>
    </add>
    <put>
      <path><in-hand/></path>
      <nothing><text>nothing</text></nothing>
    </put>
  </action>
  <action>
    <name>Drop to a cell</name>
```

```
      <guard> <!-- must drop to a cell -->
        <is-a>
          <get><event-get><position/></event-get></get>
          <cell/>
        </is-a>
      </guard>
      <guard><not>
        <equals>
          <event-get><position/></event-get>
          <outer-box/>
        </equals>
      </not></guard>
      <!-- cell must be empty (which is not the case for outer-box) -->
      <guard> <equals>
          <size><get><event-get><position/></event-get><object/></get></size>
          <value>0</value>
      </equals> </guard>
      <!-- cell must be unique -->
      <guard>
        <unique-ref><event-get><position/></event-get><cell/></unique-ref>
      </guard>
      <!-- move the object to the cell -->
      <put>
        <path><event-get><position/></event-get><cell/></path>
        <get><in-hand/></get>
      </put>
      <put>
        <path><in-hand/></path>
        <nothing><text>nothing</text></nothing>
      </put>
    </action>
  </action>
  <action>
    <name>Take an object from a cell</name>
    <event> <!-- event structure -->
      <take><object/></take>
    </event>     <!-- hand must be empty -->
    <guard> <is-a> <get><in-hand/></get> <nothing/> </is-a> </guard>
    <!-- object must exist and be unique -->
    <guard><unique-ref>
      <event-get><take/></event-get>
    </unique-ref></guard>
    <!-- move object from cell to hand -->
    <put>
```

```
      <path><in-hand/></path>
      <object><get><event-get><take/></event-get></get></object>
    </put>
    <!-- if object was in grid, display the empty cell image -->
    <action>
      <name>Remove object from grid</name>
      <guard>
        <unique-ref><grid/><event-get><take/></event-get></unique-ref>
      </guard>
      <put>
        <path><cell><event-get><take/></event-get></cell></path>
        <image>examples/jojo/cell.gif</image>
      </put>
    </action>
    <!-- if object was in outer-box, remove the cell -->
    <action>
      <name>Remove object from outer-box</name>
      <guard><unique-ref>
        <outer-box/><event-get><take/></event-get>
      </unique-ref></guard>
      <del> <path><cell><event-get><take/></event-get></cell></path> </del>
    </action>
  </action>
</view>
```

Annexe C

Traduction en langue naturelle des préconditions VDL

Cette annexe décrit la fonction $nlgen_{VDL}(n, b)$ qui n'était pas formalisée dans la section 4.2.1. Elle traite en particulier du cas des préconditions VDL, dans lesquelles sont utilisées divers mots-clefs du langage VDL. Ces mots-clefs ont une génération en langue naturelle différente de la fonction principale $nlgen_{VDL}(n)$, afin de mieux coller à leur sémantique.

C.1 Rappel : la notation utilisée

Soit St l'ensemble des chaînes de caractères possibles et Υ l'ensemble des arbres VDL possibles. De plus, soit $\mathcal{P}(E)$ l'ensemble des sous-ensembles possible de E. Pour un nœud de l'arbre $node \in \Upsilon$ donné, nous noterons :

- $tag(node) : \Upsilon \rightarrow St$, la fonction renvoyant l'étiquette XML de l'arbre $node$.
- $att(node) : \Upsilon \rightarrow \mathcal{P}(St \times St)$, la fonction renvoyant la liste des attributs du nœud. Pour une entrée $a \in att(node)$, nous noterons $a.key$ l'accès à la clef et $a.val$ l'accès à la valeur associée à cette clef.
- $content(node) : \Upsilon \rightarrow St$, la fonction renvoyant le contenu texte du nœud $node$. Si ce contenu n'existe pas, alors la fonction renvoie la chaîne vide $\surd$.
- $children(node) : \Upsilon \rightarrow \mathcal{P}(\Upsilon)$, la fonction renvoyant les fils de l'arbre $node$.

Nous noterons $\mathbb{B} = \{\top, \bot\}$ tel que $\top$ est la sémantique « vraie » et $\bot$ la sémantique « faux ». De plus, nous utiliserons comme opérateur unaire logique de négation le symbole $\neg$, tel que $\neg\top \Leftrightarrow \bot$ et $\neg\bot \Leftrightarrow \top$. Soit $e \in \mathcal{P}(\Upsilon)$ un ensemble de nœud VDL, alors nous noterons $e = \{e_1, e_2, ..., e_n\}$ la liste éléments de l'ensemble E.

C.2 Génération des mots clefs VDL

La fonction $nlgen_{VDL}(n, b) : \Upsilon \times \mathbb{B} \rightarrow St$ est définie par cas en fonction des mots clefs rencontrés (les mots-clefs étant toujours contenu dans le tag du nœud XML n) tel que :

- Si $tag(n) =$ "is-a"
  Ce mot-clef teste si le tag du premier nœud fils est le même que le tag des autres nœuds fils. L'équivalence des tags signifiant que les nœuds peuvent avoir le même « type ».

Soit $e_1 \in children(n)$ le premier fils du nœud n :
$nlgen_{VDL}(n, b) =$

$$nlgen_{VDL}(e_1, b) \oplus \begin{cases} \text{"is"} & \text{Si } \top \\ \text{"is not"} & \text{Si } \bot \end{cases} \oplus \left\{ \oplus_{c \in children(n) \setminus e_1} nlgen_{VDL}(c, b) \right\}$$

- Si $tag(n) =$ "equals"
  Ce mot-clef teste l'égalité stricte entre tous les fils du nœud n. Soit $e_1 \in children(n)$ le premier fils du nœud n :
  $nlgen_{VDL}(n, b) =$

$$nlgen_{VDL}(e_1, b) \oplus \begin{cases} \text{"is equals"} & \text{Si } \top \\ \text{"is not equals"} & \text{Si } \bot \end{cases} \oplus \left\{ \oplus_{c \in children(n) \setminus e_1} nlgen_{VDL}(c, b) \right\}$$

- Si $tag(n) =$ "get"
  Le mot-clef « get » correspond à une sorte de « pointeur » dans les langages de programmations plus traditionnelles. Lors de son interprétation, le nœud est remplacé par l'arbre XML du code de l'agent qui est référencé par ses fils :
  $nlgen_{VDL}(n, b) =$

$$\text{"the content of"} \oplus \left\{ \oplus_{c \in children(n)} nlgen_{VDL}(c, b) \right\}$$

- Si $tag(n) =$ "size"
  Ce mot-clef correspond tout simplement au nombre de fils du nœud :
  $nlgen_{VDL}(n, b) =$

$$\text{"the size of"} \oplus \left\{ \oplus_{c \in children(n)} nlgen_{VDL}(c, b) \right\}$$

- Si $tag(n) =$ "sizeref"
  Ce mot-clef est une sorte de fusion de « get » et « size », car il compte le nombre de nœuds correspondant à la référence citée :
  $nlgen_{VDL}(n, b) =$

$$\text{"the size ot the content of"} \oplus \left\{ \oplus_{c \in children(n)} nlgen_{VDL}(c, b) \right\}$$

- Si $tag(n) =$ "exists"
  Ce mot-clef vérifie l'existence d'un nœud correspondant à la référence citée :
  $nlgen_{VDL}(n, b) =$

$$\left\{ \oplus_{c \in children(n)} nlgen_{VDL}(c, b) \right\} \oplus \text{"must exist"}$$

- Si $tag(n) =$ "uniqueref"
  Ce mot-clef vérifie qu'il n'existe qu'un seul nœud dans le code de l'agent correspondant à la référence citée :
  $nlgen_{VDL}(n, b) =$

$$\left\{\oplus_{c \in children(n)} nlgen_{VDL}(c, b)\right\} \oplus \text{"must be unique"}$$

- Si $tag(n) =$ "event-get"
  Ce mot-clef est le point le plus complexe de cette fonction. Sémantiquement, il correspond à un « get », mais dont la référence n'est pas interprétée dans le code mais dans une capacité. Soit e une capacité du code de l'agent, et $calcRef(c, n)$ la fonction renvoyant un sous-arbre de e correspondant à la référence incluse dans le nœud « event-get » de la précondition n.[1] Alors, la génération en langue naturelle de la précondition n sachant que la capacité réfèrente est e est :
  $nlgen_{VDL}(n, b) =$

  $$\oplus_{c \in children(calcRef(e,n)))} nlgen_{VDL}(c, b)$$

  Ce mot-clef est important pour contextualiser la traduction d'une précondition. Par exemple, supposons une précondition simple du type « l'objet décrit dans le contenu de la requête doit exister dans l'environnement de l'agent ». Supposons la requête « prend le taille-crayon » et supposons qu'elle ne soit pas possible. C'est uniquement en confrontant la requête et la précondition qu'il est possible de générer une explication du type « l'objet taille-crayon n'existe pas dans l'environnement de l'agent ».
- Si $tag(n) =$ "value"
  Les nœuds « value » en VDL correspondent simplement à l'encapsulation d'un type primitif (*i.e.* nombre, chaîne de caractères, etc.). Seul leur unique fils est donc nécessaire :
  $nlgen_{VDL}(n, b) =$

  $$e_1$$

- Si $tag(n) =$ "not"
  Ce mot-clef inverse simplement le sens logique de la précondition :
  $nlgen_{VDL}(n, b) =$

  $$nlgen_{VDL}(n, \neg b)$$

- Sinon
  Si le nœud rencontré n'est pas un mot-clef, alors il suffit de propager l'information aux sous-nœuds. :
  $nlgen_{VDL}(n, b) =$

  $$tag(n) \oplus \left\{\oplus_{e \in children(n)} nlgen_{VDL}(e, b)\right\}$$

[1] La fonction *calcRef* est une fonction décrite dans la sémantique opérationnelle du langage VDL lui-même. La documentation VDL est accessible à l'adresse `http://www-poleia.lip6.fr:8180/~sabouret/demos/index.html`.

Annexe D

Le fichier RDF : fonction map_{VDL} et définition des pondérations de l'ontologie

Un agent VDL est en pratique noté comme un arbre XML représentant ses actions, ses connaissances et sa représentation de l'environnement. Pour pouvoir communiquer, chaque agent possède sa propre ontologie. Cette ontologie doit être écrite en OWL. Comme nous l'évoquions en section 2.2.1, il existe une fonction injective $map_{VDL} : \mathcal{C}_{VDL} \rightarrow \mathcal{C}_{ONTO}$ permettant d'associer chaque concept présent dans le code de l'agent à l'un des concepts de l'ontologie. Cette fonction est modélisée par un fichier RDF d'équivalence. Ce fichier définit aussi les poids T_X associé aux différents types de relation de l'ontologie (section 3.2.1).

D.1 Syntaxe du fichier RDF

Plus précisément, un concept VDL est une chaîne de caractères présente dans le code XML de l'agent telle que :

1. Toutes les valeurs d'attributs[1] ainsi que le contenu texte des nœuds sont des concepts VDL.
2. Toutes les étiquettes de nœud XML de l'agent qui ne sont pas des mots-clefs du langage sont des concepts VDL.
3. Tous les mots-clefs du langage qui apparaissent dans un agent donné et qui sont associés explicitement à un concept de l'ontologie OWL sont des concepts VDL.

La création du fichier RDF est requise lorsque :

1. La chaîne de caractère présente dans le code VDL est différente de celle présente dans l'ontologie OWL de l'agent (*e.g.* dans l'agent Mike, $temp_{VDL} \equiv temperature_{OWL}$).
2. La chaîne de caractère du concept VDL est un mot-clef du langage. Dans certain cas, un programmeur peut décider d'utiliser une chaîne de caractère qui correspond à un

[1] Dans la version actuelle, les clefs d'attributs ne sont pas considérées comme des concepts VDL. En pratique, nous n'avons jamais constaté d'exemple dans les divers agents programmés tel qu'une clef d'attributs soit un concept utile pour l'interprétation sémantique.

mot-clef VDL, si celui-ci ne peut pas être mal interprété dans le contexte de l'agent. Par exemple, le concept VDL *object* de l'agent Jojo désigne aussi un mot-clef du langage, mais son utilisation dans l'agent ne pose pas de problème d'ambiguïté : $object_{VDL} \equiv object_{OWL}$.

3. La chaîne de caractère du concept VDL peut être ancrée à WordNet, mais sa lemmatisation pose des problèmes d'ambiguïté. Par exemple, le mot « drop » peut être un verbe en anglais ou un nom. Pour la sémantique « poser », il définit manuellement l'équivalence $drop_{VDL} \equiv drop_{WordNet:VERB}$.
4. Il est nécessaire de définir les poids T_X pour les relations de l'ontologie (section 3.2.1). Une relation dont aucun poids n'est défini prend la valeur par défaut de 0.0. Par exemple, le poids de la relation *projectionOf*, $projectionOf_{OWL} \rightarrow 0.8$.

L'écriture de ce fichier RDF suit des règles syntaxiques très strictes. Pour définir une équivalence, nous utilisons quatre namespaces :

- Le namespace de base : `http://vdl-validation-service/` ;
- Le namespace VDL : `http://vdl-validation-service/vdl-concepts/`,
- Le namespace OWL : `http://vdl-validation-service/owl-concepts/`
- Le namespace WordNet : `http://vdl-validation-service/wn-concepts/`.

Tous les triplets d'équivalence du fichier RDF doivent avoir la forme $\langle v, isEquivalentTo, o \rangle$ tel que v est un concept VDL, $isEquivalentTo$ est la relation de correspondance défini dans le namespace de base et o est un concept OWL ou WordNet. Il existe trois types de notations pour définir une équivalence WordNet :

1. La simple : il est juste nécessaire de définir le terme exact présent dans WordNet (*e.g.* `http://vdl-validation-service/wn-concepts/table`).
2. La version « catégorie syntaxique » : Vous pouvez définir la catégorie syntaxique du terme (*i.e. Part Of Speech*) dans l'ensemble $\{NOUN, VERB, ADVERB, ADJECTIVE\}$ à considérer pour le terme en paramètre, séparé par le caractère « : » (*e.g.* `http://vdl-validation-service/wn-concepts/take:VERB`).
3. La version « offset » : Un offset est un identifiant unique désignant précisément un concept donné, et non un groupe de concepts (*e.g.* `http://vdl-validation-service/wn-concepts/drop:VERB:13567`).

Pour définir le poids d'une relation, nous utilisons un namespace supplémentaire : `http://vdl-validation-service/owl-relations/`. Une définition de poids dans le fichier RDF suit la forme $\langle o, hasWeight, f \rangle$ telle que o est une relation OWL et f un réel.

D.2 L'exemple « salle à manger »

Pour illustrer la construction de ce fichier, nous présentons un exemple simple représentant une salle à manger. Il contient une table et un buffet. Sur la table sont posés un livre et un stylo. Sur le buffet sont posés une bouteille d'eau et une boite. L'agent est présent dans la pièce avec un verre à la main.

Dans cet exemple, supposons que le terme *contains* est un mot-clef du langage. L'agent possède une ontologie OWL qui contient tous les concepts utilisée pour référencer ces objets.

Une représentation possible de ce monde est la suivante :

```
<view>
  <table>
    <contains>
      <book/>
      <pencil/>
    </contains>
  </table>
  <bookshelf>
    <contains>
      <box/>
      <bottle>
        <contains> <water/> </contains>
      </bottle>
    </contains>
  </bookshelf>
  <agent>
    <holds> <glass/> </holds>
  </agent>
</view>
```

Un fichier RDF associé à ce modèle peut être :

```
<rdf :RDF
    xmlns :rdf="http ://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns :baseNS="http ://vdl-validation-service/">
  <!-- Nécessaire pour résoudre l'ambiguïté syntaxique -->
  <rdf :Description
   rdf :about="http ://vdl-validation-service/vdl-concepts/bookshelf">
    <baseNS :isEquivalentTo
        rdf :resource="http ://vdl-validation-service/owl-concepts/bookcase"/>
  </rdf :Description>
  <!-- Nécessaire pour résoudre l'ambiguïté de WordNet -->
  <rdf :Description
   rdf :about="http ://vdl-validation-service/vdl-concepts/bottle">
    <baseNS :isEquivalentTo
        rdf :resource="http ://vdl-validation-service/wn-concepts/bottle :NOUN"/>
  </rdf :Description>
  <!-- Nécessaire pour faire la différence entre concept et mots-clefs -->
  <rdf :Description
   rdf :about="http ://vdl-validation-service/vdl-concepts/contains">
    <baseNS :isEquivalentTo
        rdf :resource="http ://vdl-validation-service/owl-concepts/contains"/>
  </rdf :Description>
  <!-- Pour définir le poids des relations -->
  <rdf :Description
   rdf :about="http ://vdl-validation-service/owl-relations/canContain">
    <baseNS :hasWeight>0.8</baseNS :hasWeight>
  </rdf :Description>
</rdf :RDF>
```

Ce fichier définit les assertions suivantes :

1. $bookshelf_{VDL} \equiv bookcase_{OWL}$ (*bookshelf* n'est pas défini dans l'ontologie, mais il correspond au concept *bookcase*).
2. $bottle_{VDL} \equiv bottle_{WordNet:NOUN}$ (*bottle* est défini comme un nom dans WordNet et non un verbe).
3. $contains_{VDL} \equiv contains_{OWL}$ (*contains* est un mot-clef, mais aussi un concept dans ce cas particulier).
4. $canContain_{OWL} \rightarrow 0.8$ (*canContain* est une relation de l'ontologie avec le poids 0.8)

Index