Thèse

présentée pour l'obtention du titre de

Docteur de l'Université Paris XI

en informatique

par Guillaume Pitel

MICO : La notion de Construction Située pour un Modèle d'Interprétation et de Traitement de la Référence pour le Dialogue Finalisé

Soutenue le 17 septembre 2004.

Francis Corblin, Professeur à l'Université Paris IV-Sorbonne, *examinateur*
Gérard Ligozat, Professeur à l'Université Paris XI, *examinateur*
Laurent Romary, Directeur de Recherche au LORIA-INRIA, *rapporteur*
Paul Sabatier, Directeur de Recherche au LIF-CNRS, *rapporteur*
Jean-Paul Sansonnet, Directeur de Recherche au LIMSI-CNRS, *directeur de thèse*

Résumé

La question de la résolution de la référence est d'une importance majeure lorsque l'on souhaite s'intéresser à la généricité dans les systèmes de dialogue, ce qui est nécessaire pour traiter le sous-cas particulier des agents assistants d'interface. En effet, s'il est relativement aisé de résoudre une référence à un objet du contexte lorsque ce dernier est limité à quelques types d'objets connus, le problème est beaucoup moins trivial si l'on souhaite pouvoir étendre la gestion des références à des situations moins contraintes. Des ambiguïtés pragmatiques peuvent apparaître, des catégories différentes peuvent moins bien « coller » avec des mots indépendants, ou bien des caractères être communs à plusieurs objets, et ne plus être discriminants. C'est pour ces différentes raisons que, sans aller jusqu'à traiter le problème de la référence dans le cadre général du dialogue non contraint, nous avons cherché à donner un cadre de résolution de la référence dans le cas des agents assistants d'interface.

Nous avons constaté les nombreux parallèles entre les systèmes de production de Newell, dont on sait qu'ils permettent de modéliser une partie des processus cognitifs, et les constructions de Fillmore, qui permettent de décrire les relations entre formes syntaxiques et représentations sémantiques. Cependant, il manque à ces notions la capacité de traiter des structures ayant des caractéristiques topologiques complexes, comme les représentations servant à un raisonnement temporel, qui sont pourtant nécessaires dans de nombreux modèles de raisonnement.

Le modèle d'interprétation constructionnelle (MIC) que nous avons conçu est un modèle d'interprétation pour le dialogue finalisé, où les différentes entrées du système de dialogue, qu'elles soient textuelles, visuelles ou autre sont traitées de manière homogène au moyen d'une opération unique : les **s-constructions**, une forme évoluée des constructions issues de la grammaire de construction de Fillmore.

Nous avons introduit la notion de s-construction (aussi appelée construction située) afin de pouvoir prendre en compte le fait que l'information traitée par le système de dialogue (décrite par des **schémas** issus de la sémantique des frames) peut être organisée à un niveau supérieur, y compris dans des structures dotées d'une topologie. La topologie qui organise les informations que contiennent ces structures peut être modélisée par des logiques particulières, qui permettent de décrire un système de contraintes. Par exemple, la logique de Allen pour la modélisation temporelle des événements peut être utilisée pour structurer les schémas représentant les événements. Les relations ainsi décrites peuvent être utilisées ensuite pour résoudre certaines références temporelles : l'expression « avant cela » désignant le segment temporel précédant l'événement désigné par « cela ». Ainsi, dans les s-constructions, les contraintes sur l'organisation de l'information (i.e. sur les conteneurs) peuvent être décrites tout autant que les contraintes sur le contenu de l'information. Dans notre modèle, l'interface permettant de décrire la logique qui supporte les caractéristiques topologiques d'un certain conteneur est nommée **contexte**, le modèle logique proprement dit n'est pas décrit dans le modèle, mais laissé à la charge d'un système externe.

La mise en œuvre du modèle d'interprétation constructionnelle, MICO, est inspirée par les travaux de Bryant sur l'implémentation d'une grammaire de construction grâce à une méthode d'analyse par tronçon (*chunk parsing*) et des travaux de Engel sur l'analyse linguistique par systèmes de production utilisée dans le projet SmartKom. Pour cette implémentation, nous introduisons la notion **d'observateur**, qui est l'équivalent opérationnel des s-constructions. Un observateur est capable de reconnaître certaines formes dans les environnements (équivalent opérationnel des contextes) et peut produire une nouvelle information dans un environnement si cette forme est reconnue.

La faisabilité d'une interprétation basée sur ce modèle n'est pas évidente, car la grammaire décrite par les s-constructions est une grammaire contextuelle, et donc indécidable. Afin de rendre l'exécution des observateurs possible, nous adjoignons deux mécanismes

pour guider le parcours du graphe d'interprétation. Le premier mécanisme utilise une information de pondération sur la probabilité de déclenchement *a priori* permettant d'ordonnancer plus efficacement l'exécution des observateurs. Cette pondération peut être soit calculée manuellement, soit déterminée automatiquement grâce à un algorithme d'apprentissage. Le second mécanisme est une généralisation du principe de restriction de sélection, généralisation rendue possible du fait que les informations manipulées dans notre modèle sont structurées. L'idée est d'augmenter la probabilité de déclenchement des observateurs capables de produire l'information attendue par un certain observateur, qui ne dispose que d'une partie de son motif de déclenchement. Le fait que les informations soient structurées topologiquement et dans des contextes distincts permet de spécifier qu'une certaine information est attendue dans une certaine zone (une zone étant l'ensemble des lieux qui respectent certaines contraintes, relativement à un contexte donné), et donc de décider assez rapidement si un observateur particulier a des chances de produire l'information attendue à l'endroit prévu. Ce mécanisme permet aussi de décrire la coercition à la demande, en spécifiant qu'un observateur ne peut être déclenché que si sa production est attendue. Enfin, en couplant ce mécanisme avec une approche de satisfaction partielle de contraintes pour décrire les motifs de déclenchement des observateurs, on assure au système d'interprétation une certaine robustesse.

Afin de montrer que le modèle d'interprétation constructionnelle peut être utilisé pour traiter des phénomènes complexes, nous avons conçu un modèle de résolution extensionnelle de la référence capable de traiter des prédicats vagues, et qui utilise un version continue des domaines de référence (Corblin, Reboul). Dans ce cadre, nous soulevons plusieurs questions sur la pertinence de l'utilisation d'une représentation propositionnelle des entités pour l'analyse linguistique. Nous montrons que la prise en compte du contexte dans l'analyse des expressions référentielles induit l'existence de plusieurs interprétations possibles pour un même prédicat référentiel appliqué à un même type d'objet. Nous proposons qu'une représentation possible d'un des rôles de prédicat puisse être basée sur des fonctions agissant dans des domaines de références. Ayant constaté l'adéquation de ce modèle aux besoins des systèmes d'interprétation des agents conversationnels, nous montrons comment l'adapter au modèle d'interprétation unifié que nous avons conçu.

Abstract

We have designed the MICO model (Observer-based Constructional Interpretation Model) in order to provide an answer to certain needs of pratical dialogue systems, such as genericity and adequacy with cognitive theories, which have to be fulfilled in order to produce evolutive and expressive dialogue systems. The constructional interpretation model provides a homogeneous way to describe treatments of different inputs of the dialogue system, such as textual, visual or other kinds of inputs.

The system is based on a single kind of operation : s-constructions (situated constructions), an advanced version of Fillmores constructions. S-constructions offer the ability to describe the way information such as temporal of visual information can be processed. In MICO, the structures containing the information are called contexts. The implementation of the contexts is let to an external part, the model only helps to describe the interface of the structure, through relations and operations on the locations of the structure.

We introduced the notion of observer for the implementation of the model. Observers are the computational counterpart of s-constructions, and are able to recognize patterns in context, then to produce new information. In order to make the model computationally tractable, we also propose to combine several mechanisms for execution control.

As an illustration of MICO expressiveness, we conceived a concrete reference resolution model handling vague predicates and using a continuous version of domains of reference (Corblin, Reboul). We propose a representation for one of the possible roles of referential predicates, based on several functions acting on domains of reference, and we adapt it to the constructional interpretation model.

Sommaire

Table des figures

Liste des tableaux

Liste des Spécifications

Préambule

Quelques mots sur l'historique de la maturation de cette thèse sont présentés ici, car c'est un élément d'importance pour une bonne compréhension de la motivation qui a guidé nos recherches au long de ces premières années. Les travaux exposés ici prennent leurs racines dans les réflexions menées avec Jean-Paul Sansonnet sur des méthodes automatiques permettant de sélectionner un objet désigné par un utilisateur face à un agent conversationnel assistant d'interface. Le jeu consistait à se mettre dans la peau d'un utilisateur sans connaissance des mécanismes internes de l'application ou de la machine, et de poser des questions ou commander un assistant imaginaire, doué de compréhension humaine et d'une connaissance parfaite de ces mécanismes. Il s'agissait de déterminer une procédure permettant de traiter la variabilité linguistique et la désignation des objets (la référence) supportant divers attributs (*e.g.* la couleur, la taille).

Même en s'interdisant de traiter les problèmes qui nous semblaient trop complexes, comme les aspects temporels ou dynamiques, les métaphores ou les métonymies, nos premières réflexions nous ont rapidement amené à nous heurter aux problèmes des références mettant en jeu les caractères spatiaux, des références à des objets apparaissant à l'utilisateur sans avoir une contrepartie dans la représentation informatique, où encore des références apparemment triviales dont la signification changeait en fonction du contexte. Il semblait qu'on ne pût pas si facilement subdiviser en strates le phénomène de la référence, et que la complexité du problème ne dépendait pas seulement de la référence en elle-même, mais aussi du type d'objet sur laquelle elle portait, et surtout de la manière dont ces objets étaient représentés dans « l'esprit » de l'assistant.

Le problème de la représentation, si souvent discuté et source de grandes controverses, semblait dans notre situation aussi être le réel sujet à aborder. Cette première constatation faite, il nous a semblé impensable de continuer à travailler sur le modèle de représentations basique que nous avions pris comme support de réflexion, et il nous a donc fallu œuvrer à trouver un nouveau modèle dans lequel nous pourrions exprimer des solutions à ces problèmes. Le modèle initial, celui du projet InterViews, est présenté en annexe. Le modèle que avons élaboré et que nous adoptons maintenant comme support de réflexion est présenté dans cette thèse.

Le premier pas qui nous a mené à développer cette thèse fut de constater qu'un même objet pouvait avoir plusieurs représentations différentes en fonction du point de vue dont on se place. Un objet aussi simple qu'une grille de jeu de morpion peut être vu d'au moins trois manières différentes : une liste de lignes, une liste de colonnes, une liste de couples *(coordonnées, contenu)*. Il en est de même avec un simple cercle géométrique, qui peut au choix être représenté par un point et un rayon, un point et un diamètre, trois points quelconques, etc. Certes, il est très simple de passer d'une représentation à une autre, mais pour autant, aucune de ces représentations ne semble être *la* bonne représentation.

Ce phénomène est assez connu des programmeurs, qui doivent constamment trouver des structures adaptées à représenter tel ou tel élément de programme. Les représentations choisies par ceux-ci dépendent de deux facteurs : parcimonie et adaptation. Parcimonie car il faut limiter au maximum l'utilisation de la mémoire et le temps de calcul des fonctions associées aux objets ; Adaptation car il faut que la représentation contienne toutes les données nécessaires à l'usage qui va en être fait. C'est de cette double recherche et du fait qu'il n'existe pas de *bonne* représentation pour un phénomène ou un objet qu'apparaissent si souvent les problèmes lors de l'évolution d'un logiciel. Les changements de représentations impliquent en effet des modifications en cascade dans le code, et ils sont rendus nécessaires par le changement d'usage des données dans le programme.

Ces constatations nous ont amené à remettre en cause le bien fondé d'appuyer la résolution de la référence sur la notion de représentation figée. Pour remplacer ce modèle bien pratique et si répandu, il nous fallait partir du principe que nous ne disposions pas *a priori* de la représentation nécessaire pour mener à bien une opération d'extraction de la référence. Il nous était par conséquent indispensable de disposer d'une fonction permettant de *transformer* une représentation qui était disponible mais non adaptée en une représentation adaptée au type d'accès requis par la forme de l'expression référentielle. Ce n'était qu'à cette condition que la procédure d'extraction pourrait avoir un caractère systématique, primordial pour le traitement automatique de la langue.

Ces fonctions peuvent être vues comme des opérateurs de morphisme ou encore comme des opérateurs de traduction. Nous avons choisi de considérer que ces fonctions sont des opérateurs de changement de point de vue, associant à une représentation partielle d'un objet une autre représentation partielle, non déductible systématiquement de la première représentation. L'opérateur $\Omega : (t_\beta) \mapsto t_\alpha$ est déclenché lorsqu'un point de vue de type t_α sur un objet est requis, et qu'un autre point de vue de type t_β sur cet objet est disponible. Cette définition peut être étendue afin de prendre en compte plusieurs points de vue dans sa partie gauche, permettant de considérer un groupe d'objets comme une seule entité, par exemple. Parce qu'ils produisent un point de vue à partir de l'observation d'autres points de vue, nous avons appelé ces opérateurs des *observateurs.*

Notre idée initiale était de concevoir un module de résolution de la référence pour des systèmes de dialogue existant, en utilisant notre approche par observateurs. En étudiant plus précisément les interfaces de systèmes de dialogue, nous nous sommes retrouvés face au dilemme suivant : le traitement effectué par les modules de référence des systèmes de dialogue était largement trop en aval pour que notre approche puisse être d'une quelconque utilité. En réalité, c'était dans le module sémantique que les observateurs devaient agir, afin de remplacer les connaissances *ad hoc* introduites pour adapter le système de dialogue à une tâche précise. Le module de résolution quant à lui se contentant la plupart du temps d'utiliser des informations de saillance pour sélectionner les éléments les plus significatifs lorsqu'une ambiguïté apparaissait.

Les modules de traitement sémantique existants étant basés sur l'approche que nous avions déterminée comme inadéquate, il ne nous était pas possible non plus d'ajouter les observateurs à ces modules. A partir de là, nous avons donc entamé une recherche sur des modèles de traitement de la langue auxquels il aurait été possible d'intégrer la notion d'observateurs. C'est cette recherche qui nous a amené à nous intéresser aux systèmes de productions utilisés en psychologie et aux versions cognitives et incarnées des grammaires de construction.

Cette démarche a nécessité une approche transversale dans le vaste domaine du traitement de la langue, alliant psychologie, représentation des connaissances, ingénierie des systèmes de dialogue, modèles informatiques d'analyse. Si une telle entreprise est passionnante, elle est en revanche beaucoup moins efficace en termes d'apport scientifique direct, et beaucoup plus lente à parvenir à un résultat mature. Les lecteurs me pardonneront les manquements et les imprécisions qui sont le prix à payer pour avoir fait le choix d'une démarche transversale.

Introduction

La question de la résolution de la référence est d'une importance majeure lorsque l'on souhaite s'intéresser à la généricité dans les systèmes de dialogue, ce qui est nécessaire pour traiter le sous-cas particulier des agents assistants d'interface. En effet, s'il est relativement aisé de résoudre une référence à un objet du contexte lorsque ce dernier est limité à quelques types d'objets connus, le problème est beaucoup moins trivial si l'on souhaite pouvoir étendre la gestion des références à des situations moins contraintes. Des ambiguïtés pragmatiques peuvent apparaître, des catégories différentes peuvent moins bien « coller » avec des mots indépendants, ou bien des caractères être communs à plusieurs objets, et ne plus être discriminants. C'est pour ces différentes raisons que, sans aller jusqu'à traiter le problème de la référence dans le cadre général du dialogue non contraint, nous avons cherché à donner un cadre de résolution de la référence dans le cas des agents assistants d'interface.

Si on l'envisage dans son sens le plus large, la question de la référence ouvre un vaste champ d'étude, et ce dans de nombreuses disciplines comme la psychologie, la linguistique ou la philosophie. En informatique linguistique, même en mettant de côté les aspects anaphoriques, il reste encore un vaste choix de problématiques ouvertes à la réflexion. Les phénomènes tels que la métaphore et ses affiliés comme la métonymie ou la méronymie, la génération d'expressions référentielles ou la résolution de l'ancrage des expressions référentielles dans une représentation formelle (c'est-à-dire la résolution des entités codées informatiquement qui sont désignées dans un énoncé), n'ont jusqu'à présent que des solutions partielles ou *ad hoc*.

Nous avons travaillé dans le cadre des agents assistants d'interface, qui est un sous-cas du cadre des systèmes de dialogue homme-machine introduisant de nouvelles contraintes, principalement pour l'ancrage dans une situation et la nécessaire généricité de la description linguistique. Nous avons cherché à donner un cadre systématique au processus d'ancrage référentiel en situation de dialogue, en nous intéressant à l'environnement partagé par l'utilisateur et l'assistant, c'est-à-dire les connaissances et les perceptions que l'utilisateur considère comme accessible à son interlocuteur virtuel.

Les travaux que nous avons menés dans cette direction nous ont conduit à adopter certains principes « philosophiques » qui, sans révolutionner radicalement en surface les modèles utilisés pour traiter la langue, influencent profondément les choix que nous pouvons faire dans la conception et la mise en œuvre du processus d'interprétation. Partant de cette constatation, nous avons établi une architecture d'interprétation que nous avons voulue à la fois simple et complète, afin d'intégrer profondément les résultats que nous avons obtenu sur la référence avec un modèle d'interprétation global.

Nous exposons nos travaux en trois parties. Dans la première partie, nous présentons le Modèle

d'Interprétation Constructionnelle (MIC), après avoir argumenté en faveur d'une approche unifiée et fondée sur le principe de constructions. Dans la deuxième partie, nous présentons un modèle fonctionnelle pour la résolution extensionnelle de la référence dans le cadre d'une interaction avec un support visuel, et nous décrivons la mise en œuvre de ce modèle en utilisant le MIC. Dans la dernière partie, nous spécifions les caractéristiques du modèle d'interprétation constructionnelle par observateur (MICO), une réalisation possible du MIC structurée autour de la notion d'observateur, et nous présentons aussi le système InterViews, qui a été à l'origine de nos travaux.

Architecture d'interprétation pour les systèmes de dialogue finalisé

Dans un premier temps, nous définissons le cadre pratique dans lequel s'inscrivent nos travaux. Dans le chapitre 1, nous étudions les évolutions dans les systèmes de dialogue et nous exposons des cas typiques dans lesquels des approches modulaires et logiques se révèlent incapables de rendre compte de certains phénomènes linguistiques ou plus généralement, de certains phénomènes cognitifs. Nous définissons par la même occasion l'objet principal de notre étude, l'agent assistant d'interface, c'est-à-dire une entité logicielle capable de s'interfacer entre une application et un utilisateur en jouant un rôle de médiateur et d'interprète. Nous portons tout particulièrement notre attention sur la nécessaire **généricité** des agents assistants d'interface, et sur ce que cela implique quant à la posture à adopter vis-à-vis de l'intégration des ontologies dans le système d'interprétation.

Nous exposons ensuite dans les chapitres 2 et 3 les théories sur lesquelles nous avons fondé notre modèle, en présentant tout d'abord différentes grammaires, dont la grammaire ECG qui est la principale source d'inspiration de nos travaux. Cette grammaire est directement issue de la grammaire de construction de Fillmore, construite autour de la notion de construction, qui est la description du lien entre la *forme linguistique* et la *sémantique.*

Nous argumentons dans le chapitre 2 en faveur d'une architecture d'interprétation unifiée, en décrivant les caractéristiques comparées des systèmes unifiés et des systèmes modulaires hétérogènes. En particulier, nous nous sommes intéressé aux résultats des systèmes de production de Newell, qui permettent de modéliser différentes fonctions cognitives de façon convaincante. Nous sommes alors amenés à nous interroger sur les caractéristiques à ajouter au modèle de la grammaire ECG pour nous permettre de décrire ces fonctions cognitives et les règles linguistiques avec un principe unique.

Il nous a semblé utile de présenter alors différents modèles permettant de traiter des phénomènes cognitifs particuliers en utilisant des structures de représentation variées. Nous décrivons les espaces mentaux de Fauconnier, les systèmes logiques ou numériques permettant de raisonner sur le temps ou l'espace, et surtout les domaines de références, qui sont un outil de choix pour modéliser la résolution de la référence dans le cadre du dialogue finalisé. Ce développement nous amène à proposer de prendre en compte des structures évoluées dans le cadre fixé par la grammaire ECG.

Le chapitre 3 nous permet d'exposer ensuite quelques réflexions sur le statut des représentations dans les modèles linguistiques, et notamment le statut des catégories par rapport aux ontologies. Nous développons dans ce chapitre la théorie des Points de Vue, qui donne un

rôle beaucoup plus souple aux catégories que celui qui leur est traditionnellement attribué, en considérant que la catégorie d'un élément n'est pas définitoire, mais qu'elle dépend du contexte. Cette théorie est naturellement adaptable à un modèle fondé sur les constructions, mais rejette la hiérarchisation des types telle qu'elle est pratiquée dans la sémantique des frames de Fillmore, utilisée dans la grammaire de construction.

Nous terminons cette partie par le chapitre 4, dans lequel nous décrivons le modèle d'interprétation constructionnelle. Ce modèle est fondé sur la grammaire ECG, mais introduit notamment la notion de *contexte*, qui permet de représenter les structures complexes, ainsi que la notion de *s-construction*, qui est une version des construction utilisant les contextes. D'autre part, ce modèle, conformément à la théorie des Points de Vue, ne reprend pas le mécanisme d'héritage de la grammaire ECG.

Résolution Extensionnelle de la Référence

Afin de démontrer la possibilité de décrire des phénomènes non linguistiques et reposant sur des structures de donnée dotés d'une topologie, nous avons choisi dans la partie II de notre thèse, de concevoir un modèle de résolution extensionnelle de la référence, c'est-à-dire un modèle capable d'expliquer la liaison entre une expression référentielle et les entités « perçues » par le locuteur.

Pour cela nous avons tout d'abord exploré, dans le chapitre 5, les différents approches à la résolution de la référence qui sont couramment employées dans les systèmes de traitement automatique des langues, en distinguant notamment les approches coréférentielles des approches extensionnelles que nous adoptons. Cette étude succinte du problème de la référence, met aussi l'accent sur l'importance d'une approche non propositionnelle pour la référence dans le cas du dialogue finalisé, principalement lorsqu'il y a support visuel.

Nous nous engageons ensuite, chapitre 6, dans une étude du rôle des expressions référentielles dans les énoncés, ce qui nous permet de restreindre notre champ d'investigation à un sous-cas des expressions référentielles. Ceci nous permet d'introduire une approche fonctionnelle dans la modélisation des expressions référentielles.

Dans le chapitre 7, nous développons cette approche fonctionnelle en un modèle relativement complet permettant de décrire le processus de résolution d'une expression référentielle comportant des prédicats référentiels vagues. Ce modèle est présenté en deux étapes, une première où la base fonctionnelle est décrite et une seconde où elle est complétée par la possibilité de partitionner le résultat de la première étape, ce qui nous permet de rejoindre le modèle des domaines de référence. Le modèle qui résulte finalement de cette proposition est le modèle fonctionnel de résolution extensionnelle de la référence.

Enfin, le chapitre 8 développe l'implémentation du modèle ci-dessus dans le modèle d'interprétation constructionnelle. Nous donnons les spécifications des *schémas*, des *contextes* et des *s-constructions* qui permettent de décrire notre modèle de résolution de la référence en suivant une approche constructionnelle contextualisée, démontrant ainsi les possibilités descriptives du modèle MIC.

Implémentations et Réalisations

Ces différentes étapes ne seraient pas abouties sans la possibilité d'envisager une mise en œuvre informatique du modèle. Dans le chapitre 9, nous décrivons le système MICO, en introduisant la notion centrale **d'observateur**, qui est la version opérationnelle des s-constructions. Nous donnons les spécifications permettant de contrôler l'exécution des observateurs afin de rendre le système suffisamment efficace pour être utilisable. Nous introduisons aussi les spécifications attendues des implémentations des contextes, qui sont une partie très importante de notre modèle.

La dernière partie est consacrée, chapitre 10, au système InterViews. Le système InterViews est une architecture de système de dialogue fondé sur le postulat que les informations linguistiques doivent être intégrées à la représentation de l'application interfacée. Le système InterViews a été proposé par Jean-Paul Sansonnet, et les problèmes posés par son développement ont été à la base des réflexions qui nous ont finalement menés à concevoir le modèle MIC, le modèle fonctionnel de résolution extensionnelle de la référence et le système MICO. C'est pour ces raisons que nous avons jugé utile de le présenter ici.

Première partie

Architecture d'interprétation pour les systèmes de dialogue finalisé

Chapitre 1

Interaction Homme-Machine Naturelle

Ce chapitre concerne la problématique générale des systèmes d'interaction homme-machine utilisant les médias naturels (paroles, gestes). Le panorama des approches que nous dressons nous permettra de situer notre cadre de recherche. Nous nous appliquerons particulièrement à relever les apports et les lacunes des différents systèmes étudiés vis-à-vis des problèmes propres à une interaction homme-machine dans le cadre du dialogue finalisé, avec pour objectif final de poser la typologie des fonctions interactionnelles qui peuvent être prises en compte dans un système de dialogue.

En effet, même avec un médiateur humain expert dans l'utilisation d'un logiciel et qui obéirait aux ordres qu'on lui donne, toutes les possibilités ne sont pas envisageables, parce qu'elles font appel à des compétences cognitives indépendantes de l'outil (l'application informatique) avec laquelle l'utilisateur interagit. Par exemple, un tableur ne sera pas avant longtemps capable d'analyser intelligemment les données qui y sont enregistrées. Un navigateur web ne sera pas non plus capable de donner une opinion politique sur une page que vous êtes en train de visiter. D'autres logiciels peuvent certes le faire un jour, mais ce qui nous intéresse ici est de définir précisément ce qui entre ou pas dans la finalité d'un système d'interaction. Les agents conversationnels et leurs capacités interactionnelles étant le cadre de cette thèse, nous nous servirons de cette typologie pour définir précisément les éléments d'un logiciel qui doivent et peuvent être interfacés (médiés) dans un agent conversationnel.

1.1 Motivation pour les interfaces naturelles

Les outils de traitement de l'information évoluent très rapidement, maintenant à une vitesse proche des limites des capacités d'apprentissage des êtres humains, de nouveaux besoins se font sentir chez les utilisateurs non-initiés aux outils de traitements de l'information (Norman, 1998), et même chez les utilisateurs initiés, pour pouvoir rester à jour des innovations. Le désir de faire sauter les barrières qui limitent la disponibilité des fonctions les plus complexes de ces outils est souvent présent dans les doléances des utilisateurs, même si force est de constater que les efforts déployés dans le domaine de l'ergonomie par les développeurs de logiciel n'ont pas encore permis de lier puissance et simplicité d'emploi, nécessitant systématiquement un

travail non négligeable de la part de l'utilisateur avant que ce dernier ne puisse utiliser un logiciel à son plein potentiel.

Les interfaces évoluent certes régulièrement vers une plus grande simplicité d'utilisation (Nielsen, 1995), mais il existe toujours un moment ou leurs limites deviennent une lourde contrainte. De plus, pour citer Jakob Nielsen : « *Chaque nouvelle fonctionnalité est un élément de plus à apprendre pour l'utilisateur, mais aussi un élément de plus susceptible de ne pas marcher ou de rendre le reste du site plus difficile à comprendre.* »[1]. Afin de trouver des réponses aux souhaits des utilisateurs, de nombreuses études ont été, et sont encore menées, en ergonomie et en psychologie, bien sûr, mais aussi dans certaines branches de l'informatique avancée, comme le traitement de la langue et de la communication multimodale.

Il s'agit en réalité de trouver un moyen de rapprocher deux mondes issus de deux évolutions différentes : le monde humain et le monde des machines, produit du côté mathématique et analytique de l'esprit humain. Les difficultés qui apparaissent lorsque l'on tente de faire rencontrer ces deux mondes sont nombreuses (Hicks and Essinger, 1991), mais il n'en reste pas moins que l'attrait supposé des systèmes capables de telles prouesses est tel que de nombreuses tentatives continuent à être menées pour simplifier la communication entre un humain et une machine.

Lorsque l'on s'intéresse au grand public, on peut se poser légitimement la question de savoir quelle est la meilleure modalité d'interaction. Par delà les outils classiques, comme le clavier et la souris dont on peut considérer qu'il deviennent une extension « naturelle » pour l'utilisateur de l'outil informatique, il faut s'interroger sur la sémantique même de l'interaction, c'est-à-dire ce que l'on « veut dire » à l'interlocuteur, et ce que l'interlocuteur « veut dire » lorsqu'il interagit avec l'ordinateur. Ce problème se pose de manière cruciale lorsque l'utilisateur est désemparé face à une situation qu'il ne maîtrise plus, voire qu'il ne comprend pas ; l'expérience montre que le recourt à la langue est alors instinctif. On peut facilement mesurer ce que l'intégration d'agents assistants capables de dialoguer pourrait apporter comme amélioration dans l'utilisation des applications, simplement en comptabilisant les heures passées à installer, configurer, explorer les logiciels à la recherche de fonctions particulières.

1.2 Interfaces naturelles

Il s'agit ici de définir différentes catégories d'interfaces dites naturelles, c'est-à-dire qui n'imposent pas un cadre contraignant auquel l'utilisateur devra s'adapter, mais au contraire s'adaptent aux habitudes de communication des humains. Les interfaces naturelles sont plus ou moins naturelles, et chacune accepte différentes facettes de la communication humaine.

De manière générale, on peut construire la hiérarchie suivante pour les types de dialogue :

1. Le dialogue **général** : c'est le but ultime des recherches en intelligence artificielle. Un système doté de la compétence de dialogue général est caractérisé par Turing (1950) de la manière suivante : soit un humain connecté à ce système via un canal qui cache la nature du système en question. Si l'humain établit une discussion avec le système et qu'il est incapable de dire si son interlocuteur est une machine ou un autre humain, alors on peut considérer que le système est doué de la compétence de dialogue général. Ceci

[1] Traduction.

implique que le système doit **au moins** être capable de produire des réponses cohérentes avec les énoncés de l'humain, qu'il doit montrer dans ses interventions qu'il dispose de bon sens, d'une connaissance normale des sujets de la société dans laquelle vit l'humain, d'une capacité de raisonnement logique minimale et d'une capacité à apprendre à partir de ce que peut lui dire l'humain. Le dialogue homme-machine est parfois considéré comme un sous-cas du dialogue général, dans lequel l'utilisateur verrait qu'il parle avec une machine. Il a été montré que dans cette situation, le locuteur humain se comporte différemment que lors d'un dialogue homme-homme, notamment en limitant son expression corporelle, et en simplifiant son discours à la fois en utilisant des constructions simples et en choisissant son vocabulaire dans un lexique limité. Les expériences ayant montré ceci ont cependant un côté biaisé, puisque le sujet a généralement une connaissance propre des capacités des machines à comprendre (et à percevoir) les expressions gestuelles, discours complexes ou attitudes diverses. On ne peut donc pas réellement considérer ce sous-cas comme significatif.

2. Le dialogue **finalisé** : il est caractérisé par le fait que l'échange est causé par la nécessité d'accomplir une tâche. Le dialogue a donc un objectif, une fin, d'où l'attribut *finalisé*. Le trait particulier de ces dialogues est que a) le lexique utilisé est réduit par rapport au lexique général, en fonction du **domaine** du dialogue (par exemple un dialogue finalisé servant à établir un diagnostic médical va utiliser uniquement des termes médicaux, tandis qu'un dialogue portant sur le choix d'une musique dans un magasin va porter sur les styles musicaux, les artistes, les instruments de musique, etc.) et b) que les énoncés, qu'ils soient des ordres, des questions, des affirmations, sont guidés par la **tâche** à accomplir (le dialogue entre un chirurgien et ses assistants lors d'une intervention ne comportera pas les mêmes types d'énoncés que le dialogue entre un médecin et une infirmière à propos de l'évolution de l'état de santé d'un patient). Ces deux limitations permettent d'imaginer pouvoir traiter informatiquement le dialogue, puisque elles réduisent très nettement la complexité du problème, même si elles ne suppriment pas toutes les problématiques du traitement de la langue encore sans solution.
   Les systèmes de dialogue finalisé sont caractérisée par deux dimensions : la finalité du dialogue et le support du dialogue.

   (a) Finalité du dialogue

      i. Le dialogue de **commande**. Il consiste à traiter l'échange entre un donneur d'ordre et un exécutant. Un type de dialogue de commande pourrait être de commander le fonctionnement de sa maison (domotique) par la voix, avec des ordres de type « Ouvre les volets », « Active l'alarme » ou encore « Si la fenêtre est ouverte, coupe le chauffage ».

      ii. Le dialogue de **renseignement**. De nombreux services par téléphone existant actuellement consistent à renseigner un utilisateur sur certains domaines. L'illustration la plus courante est probablement le service de renseignements téléphonique, qui permet entre autres d'obtenir le numéro de téléphone d'une personne. Bien sûr, on peut aussi donner certains ordres (« Passez-le moi directement »), mais tant qu'ils restent très peu nombreux et normalisés, on peut considérer l'activité comme un dialogue de renseignement.

      iii. Le dialogue **mixte**. La plupart des activités demandent à la fois des capacités de commande et de renseignement, par exemple pour utiliser une application

de lecture de courrier électronique, il faut pouvoir commander : « Efface les courriers vieux de plus de 15 jours », mais aussi poser des questions : « Est-ce que j'ai reçu un courrier de Jean-Paul avec une pièce jointe il y a quinze jours ? ».

(b) Support du dialogue

i. Le dialogue **monomodal**. Le dialogue n'implique pas uniquement la parole, deux humains en discussion utilisent leur visage, leur mains, ainsi que différents expressions sonores que l'on peut qualifier de non-verbales (soupirs, rires). Le dialogue monomodal est un sous-ensemble du dialogue qui ne s'intéresse qu'à la modalité verbale, considérant qu'aucune information n'est transmise par un autre canal. On peut même réduire le dialogue à du texte, comme cela se passe sur les *chats* (chambres de discussions virtuelles sur internet).

ii. Le dialogue **multimodal**. Le dialogue multimodal possède de nombreux aspects différents. La multimodalité peut être en entrée (visage, gestes, émotions dans la voix, pointage avec la souris), en sortie (affichage d'informations, affichage d'un personnage auquel on associe l'incarnation du système, signaux sonores non verbaux). La plupart du temps, les modalités traitées par le système dépendent surtout du type de tâche pour lequel on le destine. Si les techniques de fusion des informations tendent à devenir plus génériques, la manière dont on extrait le sens exprimé par un acte de dialogue dans son ensemble reste très lié à la tâche visée.

A. Support **visuel**. C'est l'un des éléments les plus importants dans un dialogue, et il reste très problématique lorsqu'il est au centre de l'interaction. En effet, là où dans un dialogue monomodal, on sait que les références vont être construites à partir d'indices linguistiques relativement clairs, le fait que l'utilisateur dispose d'une visualisation des éléments de l'application fait qu'il va pouvoir utiliser beaucoup plus d'indices perceptuels moins évident à modéliser. C'est notamment pour cela que la plupart des systèmes de dialogue finalisés avec support visuel intègrent aussi des modalités gestuelles ou de pointage (souris, crayon optique, écran tactile) afin de simplifier la résolution des références.

B. Support des **gestes**. Afin de guider les systèmes de résolution de la référence, la prise en compte de modalités permettant le pointage des objets a très vite été l'objet de recherches. L'intérêt évident de cette approche est cependant tempérée par le fait que les utilisateurs disposant de ce nouveau moyen d'expression, ne l'utilisent pas toujours de la manière attendue. Ainsi, les gestes de pointage peuvent dessiner des formes complexes servant à entourer le ou les objets que l'utilisateur veut désigner. La fusion des informations est donc particulièrement délicate.

Cette première typologie nous permet de situer les agents conversationnels assistants d'interface par rapport aux différentes familles de systèmes de dialogue. Leur objectif est principalement de traiter quatre aspects :

– la commande/contrôle,
– le renseignement,

– le support visuel,
– le raisonnement et la planification.

Nous détaillons ces quatre aspects avons de présenter plus spécifiquement les caractéristiques attendues des agents conversationnels assistants d'interface.

1.2.1 Systèmes de commande et contrôle

Commençons par un court historique des interfaces pour les applications informatiques. Les premières interfaces ont été les interpréteurs de commande (en anglais : *shell* c'est-à-dire coquille). Leur fonctionnement est principalement orienté autour de deux facteurs : la navigation dans un système de fichiers (le contenu informationnel du logiciel) et l'exécution de commandes externes. Sur ce premier niveau de communication, un deuxième niveau d'interface, propre à chaque logiciel, est venu se greffer. Pour lire ses courriers, par exemple, l'utilisateur exécute la commande *mail* sous Unix, et se trouve alors dans un autre environnement, avec des commandes courtes permettant de visualiser, supprimer ou faire suivre des messages, ou bien d'en écrire de nouveaux.

Ces modèles à base de commandes et de raccourcis clavier ont vite montré leurs limites du point de vue de leur utilisabilité. Quel utilisateur non-informaticien supporterait d'apprendre la pléthore de commandes disponibles pour chacun des logiciels qu'il veut utiliser, ainsi que la syntaxe à utiliser pour chaque commande paramétrable ? Les interfaces proposant des menus déroulants ont vu le jour sur des terminaux capables de n'afficher que du texte afin d'augmenter le confort de l'utilisateur. Le principal atout de ces interfaces est de montrer à l'utilisateur les actions (ou catégories d'actions) qui sont à sa disposition. Ainsi, face à son application, l'utilisateur est capable de chercher rapidement parmi les choix qui lui sont proposés pour faire apparaître une autre liste de commandes ou de catégories de commandes (les sous-menus) disponibles.

Dans le même temps, les premières interfaces pseudo-graphiques ont donné la possibilité de ne plus se trouver face à une interaction séquentielle figée. En effet, les commandes nécessitant un grand nombre de paramètres ont d'abord été améliorées en demandant séquentiellement à l'utilisateur les valeurs pour chaque paramètre, afin d'éviter de devoir imposer une syntaxe trop rigide. Ces approches sont cependant extrêmement contraignantes, car il est bien souvent impossible de faire marche arrière. Pour éliminer ce problème, les boîtes de dialogue ont été introduites, regroupant sur un écran dans lequel l'utilisateur peut naviguer, l'intégralité des paramètres à fixer. L'utilisateur dispose ainsi d'une vision globale sur une question particulière, et peut modifier les paramètres rentrés à tout moment.

Le passage du mode texte au mode graphique a enfin permis de donner une apparence plus agréable aux interfaces, mais aussi d'introduire des icônes, qui permettent une perception rapide d'un plus grand nombre d'éléments à l'écran.

Au-delà des différentes interfaces particulières, le modèle actuel prévalant dans les IHM[2] reste fondé sur la notion d'événements déclenchant des actions. Il est donc normal que les premiers travaux sur l'interfaçage naturel se soient orientés sur l'analyse de commandes simples en langue humaine.

[2]Interfaces Homme-Machine.

Une illustration parfaite d'un système d'analyse de commande en langue humaine est le système commercial Dragon NaturallySpeaking (de la société ScanSoft). Ce logiciel permet principalement deux choses : transcrire la voix en texte numérisé et commander quelques applications (deux applications de traitement de texte répandues) au simple son de la voix. Le logiciel permet de donner des ordres de ce genre :

– « souligne ceci »
– « souligne les trois derniers mots »
– « plus gros » (sur texte sélectionné)

Ce type d'outil est très intéressant pour les personnes handicapées. Il peut aussi être utile à des personnes ayant à utiliser des logiciels dans des cas où leurs mains sont indisponibles (par exemple un pilote d'avion ou un chirurgien). En revanche, il ne répond pas aux attentes que nous avons défini ci-dessus, notamment parce que l'utilisateur doit connaître auparavant les termes acceptés par le système de compréhension, ainsi que les constructions autorisées. En effet, il n'est pas facile de savoir si on peut dire « recopie les deux premières lignes en italique à la fin du paragraphe » à moins de disposer de la grammaire et du lexique acceptés par le système. Le fait de risquer un échec de compréhension ne peut que rebuter l'utilisateur qui n'aurait pas un besoin critique d'utiliser la voie orale pour commander un logiciel.

Les principales lacunes de ce type de système concernent la capacité à être adaptés à d'autres outils. Les verbes d'action autorisés ont en effet une sémantique bien précise dans l'application et se traduisent donc directement, de même que les prédicats et les noms. Adapter un tel système à une autre tâche implique de modifier la quasi intégralité du lexique utilisé pour associer mots et commandes logicielles. De plus, seul un nombre limité de constructions verbales est susceptible d'être correctement interprété par ces systèmes, qui fonctionnent généralement avec quelques motifs de phrases prédéfinies.

1.2.2 Renseignement

Outre les systèmes permettant de commander les logiciels (ou les machines électroniques), les concepteurs de systèmes de dialogue se sont très tôt penchés sur la question de donner une interface naturelle aux systèmes d'interrogation de base de données (Androutsopoulos et al., 1995). En effet, ces systèmes donnent accès à une mine d'information extrêmement profitable à qui sait les utiliser, mais les utiliser à travers leur langage d'interrogation natif (dont le plus connu est SQL, Structured Query Language). Cela reste cependant une affaire de spécialistes ou au moins d'amateurs éclairés. Or ces systèmes ne servent bien entendu pas qu'aux informaticiens, et on a très tôt souhaité avoir la possibilité de poser à ces systèmes les mêmes questions qu'on pourrait poser à un assistant spécialiste chargé de rechercher une information dans une base de données.

Les résultats obtenus dans cette voie ont été très satisfaisants, principalement parce que le vocabulaire étant généralement vernaculaire, puisque spécifique à des professions ou à des thèmes assez restreints, les ambiguïtés lexicales y sont très rares. D'autre part, les constructions grammaticales qui interviennent dans des questions posées sur de grandes quantités de données restent relativement stables, les principales variations se situant au niveau des éléments du lexique.

En revanche, si l'on souhaitait concevoir une telle interface pour une base de données réellement imposante (Gazdar (1996) donne l'exemple d'une base qui recenserait toutes les infor-

mations sur la Finlande et ses habitants), alors on est confronté au problème que la bonne interprétation d'une question nécessite une prise en compte de l'importance du contexte dans la langue. Les approches symboliques échouent car elles ne prennent pas en compte l'interrelation existant entre niveaux syntaxiques, sémantique et pragmatiques, et les approches statistiques échouent car elles ne peuvent prendre en compte des fenêtres de mots suffisamment larges pour capturer les influences distantes, et sont au contraire trompées par la présence de certains mots qui nécessiteraient des connaissances symboliques pour pouvoir être ignorés.

Les solutions aux problèmes d'interprétation de la langue n'ont donc que peu avancé avec ces systèmes d'accès aux bases de données, qui ont par contre nécessité des avancées importantes sur les problèmes de construction automatique d'ontologies, puisque celles-ci étaient nécessaires pour construire des motifs de questions simples couvrant une vaste gamme de questions possibles.

1.2.3 Interaction avec support visuel

L'utilisation d'une interaction naturelle par l'usage de la langue a nécessairement évolué des systèmes d'interrogation de base de données à des systèmes d'interaction avec une composante visuelle, afin de prendre en compte à la fois des applications robotiques ayant une action dans le monde physique et les applications logicielles ayant une interface graphique. Ce type d'interaction a posé de nouveaux problèmes, d'une part parce qu'elle a introduit des notions spatiales, et d'autre part car ces différentes applications ne sont intéressantes que si elles intègrent un aspect dynamique (déplacement des objets par exemple).

Le célèbre système SHRDLU (Winograd, 1973) a marqué l'histoire de la linguistique informatique comme un système de compréhension très impressionnant, capable d'apprendre des mots, de comprendre des expressions référentielles complexes, et d'avoir une mémoire des actions passées permettant un traitement de qualité pour les anaphores. Passé la première impression positive, ses défauts ont cependant terni son succès, et son auteur lui-même a reconnu ses limites, notamment du fait de sa conception très opportuniste et *ad hoc*, qui ne laissait presque aucune possibilité de le faire évoluer vers d'autres applications que ce pour quoi il était destiné.

Ce qui nous intéresse particulièrement dans ce système, c'est qu'il présentait à l'utilisateur un affichage où l'on pouvait trouver les éléments géométriques que le système manipule, ainsi que la « main » virtuelle d'un robot imaginaire à qui l'utilisateur adresse ses ordres.

Dans un cadre spatial limité à quelques lignes et colonnes d'objets dont les caractéristiques de taille et de couleur sont fixées et non ambiguës, retrouver le ou les éléments désignés par une expression référentielle ne pose pas de réel problème, mais à tout le moins, le système ne se limitait pas à faire désigner les objets par des étiquettes uniques, et permettait de démontrer que dans un cas restreint, le lien entre perceptions humaine et objets informatiques pouvait être établi. SHRDLU permettait aussi, à travers un mécanisme assez simple, de faire apprendre au système de nouveaux mots, à condition que le sens du nouveau mot soit construit à partir de sens existants, et que la construction du nouveau sens soit elle aussi, relativement simple.

Voici un exemple de construction établie par SHRDLU en réponse à l'énoncé suivant : « Y-a-t'il un bloc sous un triangle ? »[3]

[3]C'est un dialogue imaginé, puisque le dialogue de démonstration de SHRDLU était en anglais. Ce dialogue

– (exists (and (is ?X1 Block) (is ?X2 Triangle) (under ?X1 ?X2)))

A partir de la représentation du « monde » de SHRDLU, l'évaluation de cette formule logique permet de trouver les X1 et X2 qui satisfont les trois conditions, et permet donc de résoudre la description indéfinie « un bloc sous un triangle ».

L'apprentissage de nouveaux termes se déroule de la manière suivante[4] :

U : Y-a-t'il une maison ?
S : Désolé, je ne connais pas le mot « maison ».
U : Une « maison » est un triangle sur un bloc.
S : Je comprends.
U : Y-a-t'il une maison maintenant ?
S : Non.
U : Construis-en une.
S : OK.

La capacité d'apprentissage de SHRDLU, pour impressionnante qu'elle soit, est cependant limitée par plusieurs facteurs. D'une part, le système ne peut pas apprendre à partir d'une situation, c'est-à-dire que si l'utilisateur construit une forme particulière à l'aide des blocs (par exemple un château), il ne pourra pas apprendre à SHRDLU que cette forme se désigne par le terme « château ». Plus ennuyeux encore, il ne peut y avoir de définition imprécise, alors qu'une maison peut être définie soit de la manière vue dans le dialogue précédent, soit de manière plus évoluée par la figure 1.1. SHRDLU n'est de plus pas capable de gérer deux définitions d'un même mot, et encore moins de créer une généralisation à partir de plusieurs définitions. Enfin, la gestion de l'apprentissage est strictement limitée aux noms, rendant impossible tout définition d'action composée.

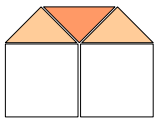


FIG. 1.1 – Un modèle de « maison » plus compliqué pour SHRDLU.

Ces limitations sont principalement dues à l'utilisation de connaissances *ad hoc* saupoudrées dans les différentes procédures d'analyse de SHRDLU. Ce problème reste encore très présent dans les systèmes de dialogue récents, et on le désigne souvent comme le problème de la généricité. C'est une des raisons pour laquelle l'approche modulaire a eu du succès, car elle permet de concentrer dans des modules particuliers les éléments variables qui dépendent de la tâche et du domaine traités, et par conséquent d'isoler les éléments invariants qui peuvent être réutilisés.

Beaucoup d'autres systèmes de dialogue intègrent le support visuel, notamment les systèmes basés sur des agents conversationnels animés, qui utilisent des figures humanoïdes ou anthropomorphiques comme représentants du système de dialogue.

peut être consulté sur la page web http ://hci.stanford.edu/~winograd/shrdlu/

[4]Il ne s'agit pas ici d'un dialogue fonctionnant réellement avec SHRDLU, mais d'une libre adaptation française d'un sous-ensemble des dialogues présentés par Winograd pour faire la démonstration de son système.

1.2.4 Planification et raisonnement

TRIPS Le projet TRIPS (Ferguson and Allen, 1998), acronyme qui signifie The Rochester Interactive Planning System, définit une architecture de système de dialogue qui permet d'assister un utilisateur dans des tâches de planification. TRIPS permet ainsi d'aborder le problème particulier de la planification dans le cadre plus général des systèmes de dialogue. C'est un moyen d'étudier des phénomènes complexes propres à un cas particulier (et donc de vérifier qu'un modèle de dialogue est adaptable) sans viser le dialogue libre, qui reste, au vu des connaissances actuelles, hors de portée. L'architecture de TRIPS (figure 1.2) a été proposée afin d'isoler au maximum les composants dépendants de la tâche et du domaine du dialogue des composants systématiquement réutilisables. L'intérêt évident de ce découpage est qu'il permet à des personnes différentes de travailler sur des problématiques précises qui interviennent dans le dialogue sans interférer avec les autres modules.

TRIPS est un système avec support visuel, mais ceci a plus ou moins d'impact selon la tâche à laquelle il est appliqué. Par exemple, dans la tâche Pacifica, l'utilisateur visualise une carte de chemin de fers avec les différentes interconnexions, cependant cette visualisation n'a aucun impact sur la manière dont les différents points sont désignés, puisqu'ils possèdent tous une étiquette avec leur nom, et que c'est par ce moyen unique que l'utilisateur y réfère. En revanche, pour la tâche TRIPS911 (centre d'intervention d'urgence), une carte plus complexe est affichée, et l'utilisateur réfère parfois à certains points ou routes à l'aide de relations spatiales, de croisements, etc.

L'une des applications majeures de TRIPS est un système permettant d'assister la supervision d'un centre d'intervention d'urgence, où un coordinateur doit envoyer différents moyens de secours en différents lieux géographiques, en respectant certaines contraintes imposées par les types d'accidents, la vitesse des véhicules, la nature du terrain, etc.

Du point de vue des modèles théoriques phares de l'architecture, TRIPS est fondé sur un analyseur ascendant (*bottom-up*) dit « chart parser » (Allen, 1995). Les constructions syntaxiques produites par l'analyseur sont ensuite transformées en expressions logiques en se basant sur le lexique de *frames* du projet FRAMENET (Baker et al., 1998; Fillmore, 1982). Pour adapter le système à un domaine dont l'ontologie n'est pas couverte par FRAMENET, TRIPS utilise un mécanisme de traduction entre une ontologie spécialisée et l'ontologie générale (Dzikovska et al., 2003), afin de pouvoir utiliser les compétences de déduction génériques pour un domaine particulier. La résolution des références est traitée par une réécriture des expressions logiques prenant en compte les caractéristiques propres de l'expression référentielle et les éléments accessibles dans la représentation du monde (Byron and Allen, 2002).

Concernant ses capacités de planification, outre un gestionnaire spécialisé dans la prise en compte du dialogue, TRIPS dispose d'un module permettant de représenter les plans d'exécution d'action, afin notamment de supporter les tâches comme celle qui est présentée plus haut. Le planificateur utilisé dans TRIPS est hérité d'un projet antérieur, TRAINS, et possède la particularité de permettre une planification à initiative partagée (Ferguson et al., 1996). Quand l'utilisateur demande une action au système, celui-ci calcule un plan permettant d'atteindre le but fixé par l'utilisateur, avec une technique dérivée de l'approche classique en planification (Sacerdoti, 1977). Ensuite, le système est capable de présenter le plan qu'il a conçu si celui-ci semble poser un problème (par exemple si l'utilisateur veut aller d'une ville A à une ville C, que la ville B se trouve dans le plan conçu par le système, mais que le passage

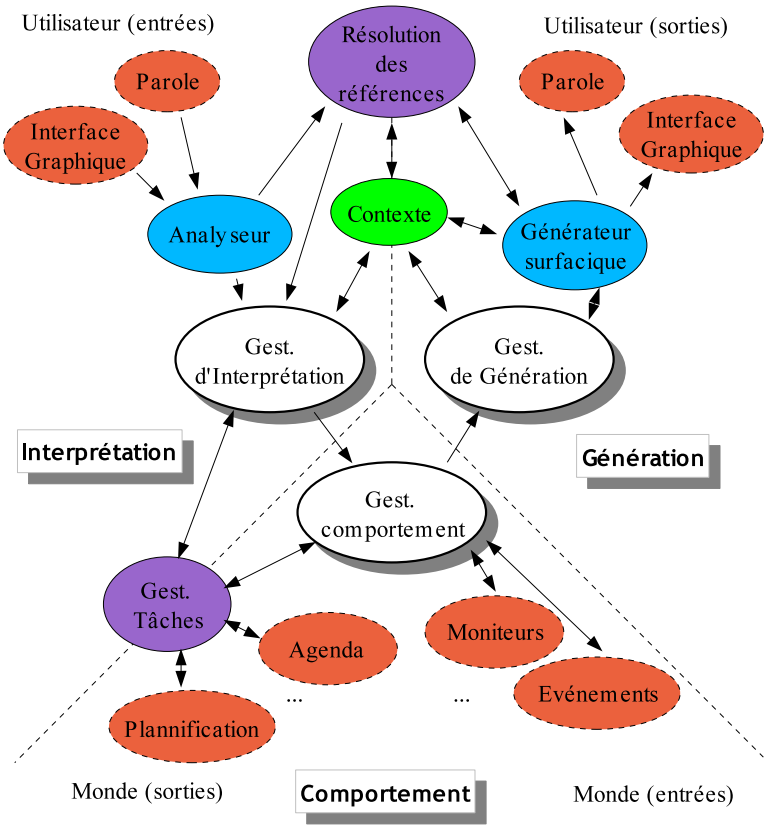


FIG. 1.2 – Architecture de TRIPS

par cette ville B est retardé pour une raison ou une autre). L'utilisateur peut alors décider s'il garde son choix initial ou bien s'il préfère modifier son parcours global pour éviter ce chemin.

SmartKom Le projet SmartKom (Wahlster et al., 2001) est davantage axé sur l'aspect multimodal du dialogue, comme on peut le constater sur son schéma général de fonctionnement (figure 1.3). De notre point de vue, un des points particulièrement intéressant de SmartKom est qu'il utilise SPIN (Engel, 2002), un analyseur fondé sur un système de production (Newell, 1973). Or le modèle d'interprétation constructionnelle que nous proposons dans cette thèse est largement inspiré du fonctionnement des systèmes de production, et le fait qu'un tel système puisse être utilisé dans une application réelle est selon nous un indice très probant de la faisabilité d'une telle approche.

Un aspect particulièrement de SmartKom est son objectif de multimodalité, notamment représenté par le module de fusion des modalités, que l'on peut voir dans le schéma de fonctionnement. Ce genre de module se retrouve dans d'autres systèmes (Martin and Béroule, 1999) et se distingue par la capacité de combiner des informations diverses (principalement paroles et gestes) communiquées par l'utilisateur pour désigner des objets. Cette approche, même si elle est particulièrement dirigée vers un objectif pratique assez simple dans la plu-

part des cas, repose sur des fondements psychologiques reconnus (Hatwell, 1994; Hartline, 1985) qui montrent l'importance de prendre en compte de manière coordonnée les différentes informations et les différents points de vues qui peuvent être accessibles à un moment donné. Cette idée est largement reprise dans le modèle d'interprétation que nous proposons dans cette thèse, même si nous n'avons pas orienté spécifiquement notre discours vers la fusion de différentes modalités, mais plus généralement vers la prise en considération de manière coordonnée de différentes sources d'information.

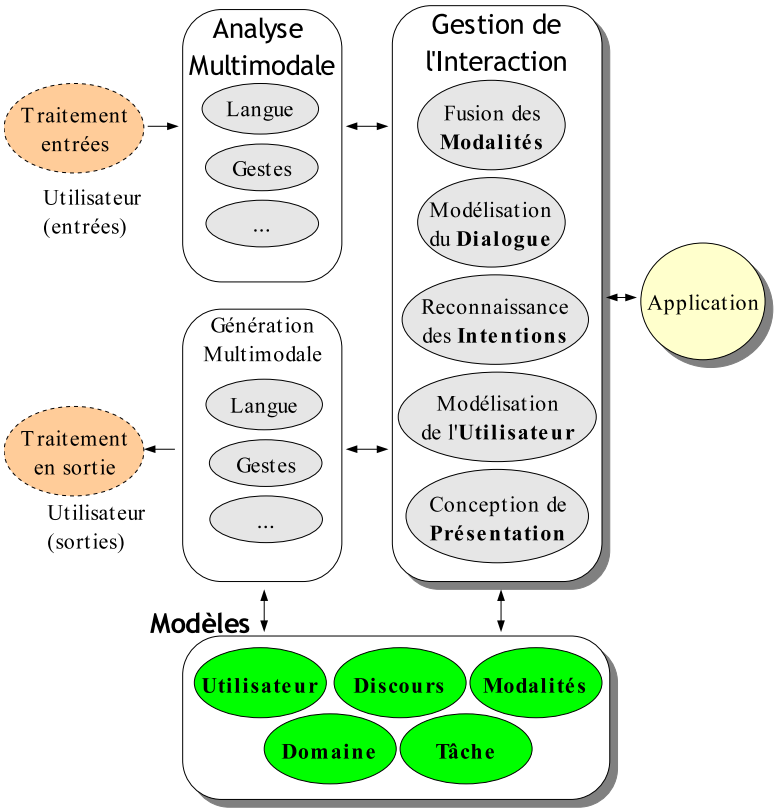


FIG. 1.3 – Architecture générale de SmartKom

InterViews Le projet InterViews (Sansonnet et al., 2003), initié en juin 1999 par Jean-Paul Sansonnet, tenta d'apporter un début de réponse aux problèmes posés ci-dessus, grâce à une approche utilisant un modèle de description de fonctionnalités par des agents, baptisés *agents dialogiques*.

Le choix de cette approche est issu d'un constat sur l'évolution des techniques de programmation au cours de l'histoire de l'informatique, qu'on a vu passer des descriptions de fonctionnement purement séquentielles, puis procédurales, modulaire, pour arriver à notre époque à une approche orientée objet très répandue, où l'on choisit de regrouper ensemble données et opérations sur les données. Les techniques les plus récentes, généralement affiliées à la notion d'agent, semblent, quant à elles, chercher à regrouper encore plus de caractéristiques (but,

croyances, etc.) dans des blocs descriptifs indivisibles.

On constate donc une tendance croissante à regrouper toutes les informations qui concernent une même unité de programmation, ou peut-être, une même unité de sens. Ceci a amené assez naturellement à penser qu'il était peut-être possible de mettre en parallèle ces unités de sens crées par des programmeurs, avec les unités sémantiques présentes dans le lexique.

Bien entendu s'est posé à nous un problème ardu, dû à la différence entre les moyens utilisés par les programmeurs dans leurs programmes et par un humain pour représenter l'environnement extérieur. Cette différence engendre un phénomène d'incompréhension, les concepts créés par un humain provenant de ses perceptions et de son héritage génétique et culturel où entrent en jeu les émotions, entre autres, et ceux des programmes, qui sont plus orientés vers les formalismes logiques des techniques informatiques.

Les *agents dialogiques* sont des représentations d'information regroupant des capacités de représentation symbolique, de raisonnement réflexif statique et dynamique, ainsi que de contrôle et d'assistance en langue naturelle.

Un tel agent offre la possibilité au système dont il fait partie d'être exécuté, comme tout programme, mais aussi et surtout d'être interrogé et commandé par l'intermédiaire de la langue naturelle.

L'entité de base du projet InterViews est la *vue*. Une vue est définie comme un composant actif qui intègre, dès sa conception, des représentations sémantiques qui permettront de le produire dans une page Web et des représentations sémantiques qui permettront d'interagir avec lui via un assistant dialogique.

Une vue peut-être comparée à un « petit monde opératoire ». Dans le domaine de l'intelligence artificielle, ce terme réfère classiquement à un modèle de représentation et de traitement des connaissances *localiste* et *dynamique*. Dans un modèle *localiste*, les connaissances portent sur un domaine sémantique bien délimité ; contrairement aux modèles d'intelligence artificielle classique qui visent une approche globale des connaissances (dans ce cas, local $\Leftrightarrow$ partiel), l'approche localiste affirme qu'il n'y a pas de connaissance globale possible mais des « points de vues » qu'il faut composer, voire confronter ; cette approche est défendue par la communauté des Systèmes Multi-Agents. Elle est particulièrement bien adaptée à la structure des informations disponibles sur internet.

La notion de modèle de représentation dynamique est interprété de manières assez différentes selon les communautés scientifiques qui y font appel (Ishida, 1989). Par modèle de représentation dynamique, nous entendons ici un système qui possède un état (représenté par des variables d'état) et des processus internes capables d'agir sur cet état. Dans les systèmes classiques de représentation et de traitement des connaissances, le monde est vu comme une structure statique : il ne possède qu'un seul état. Les représentations déclaratives et le raisonnement logique sont bien adaptés aux besoins de ces systèmes. Au contraire, l'approche dynamique met l'accent sur les changements des états du monde qui est vu comme un système en cours d'évolution. La notion d'état dynamique impose une représentation en extension des objets du monde (les choses) et le traitement, ou encore le raisonnement, porte alors sur un « état des choses ».

En résumé, une vue constitue :

1. un domaine sémantique bien délimité qui constitue un « point de vue »

2. des processus qui font évoluer un « état des choses ».

Remarque terminologique : La notion de vue, telle qu'elle est définie ci-dessus, a une signification très différente de la notion de vue communément admise en Interaction Homme-Machine (IHM). En IHM, pour un objet donné, on définit des vues multiples, associées par exemple à des modes d'affichage ou à des utilisations spécifiques.

En pratique, une vue correspond à une classe de composants actifs qui peuvent être instanciés, composés puis intégrés dans des pages web. On distinguera deux types de composants selon leur mode d'activité :

1. les composants locaux dont l'exécution (le *runtime*) est effectuée chez l'utilisateur ; ce mode correspond à peu près à celui des applets et il est bien adapté aux applications dans le domaine culturel : enseignement, jeux etc.
2. les composants d'interface dont l'exécution se partage entre un « *runtime* » effectué chez l'utilisateur et l'appel à des fonctions de service à distance ; ce mode client/serveur sert à interfacer des applications lourdes déjà existantes et le composant sert alors de médiateur (au sens des bases de données hétérogènes).

Afin de pouvoir répondre à des questions sur le fonctionnement du système d'agents, ce dernier doit pouvoir accéder à la représentation du fonctionnement des vues qu'il contient. Cette représentation n'est pour l'instant pas envisagée comme étant décrite en langue naturelle, mais au contraire, comme un formalisme logique.

Le raisonnement sur le fonctionnement a pour but d'interpoler l'interprétation d'une requête pour l'exécuter, même si cette requête demande la réalisation d'actions qui ne sont pas prévues par les agents eux-mêmes. Ainsi, si nous décrivons un compteur comme étant un agent capable de deux actions : se mettre à zéro et incrémenter sa valeur de 1 (c'est une « vue » minimaliste, mais complète), alors si l'utilisateur demande à ce compteur de se mettre à une valeur donnée, un module de raisonnement sur le fonctionnement devrait être capable de déduire que pour réaliser cette action, il est nécessaire de commencer par réinitialiser le compteur, avant de l'incrémenter autant de fois qu'il le faut pour arriver à la valeur voulue.

D'autre part, il est indispensable de pouvoir raisonner sur la structure statique des agents, c'est-à-dire, sur l'organisation des agents entre eux, du point de vue des dépendances d'appartenance et de hiérarchie. Par exemple, en langage naturel, on réfère à une partie d'un concept en utilisant une construction de la forme : *partie* **de** *concept*. Le raisonnement nécessaire pour résoudre une telle construction va consister à atteindre, dans la base de connaissances construite à partir des agents, instanciés ou non, celui qui correspond à la sous-partie demandée de l'agent représenté par *concept*.

1.3 Agents assistants d'interface

1.3.1 Identification des besoins émergeants

L'adéquation entre l'utilisateur et ses outils, après un temps d'adaptation variable, ressemble à une relation de communion, l'outil devenant un prolongement du corps. Il est d'ailleurs bien souvent nécessaire que cette cohésion soit atteinte pour que les choses acquièrent le statut

d'outil, et ils ne sont avant cela que des objets dont on souffre de se servir. Il en est de même pour les interfaces logicielles : pour que l'usage de l'outil qu'elles médient soit optimalisé, les interfaces doivent être proches de la perception profonde qu'a l'utilisateur de son outil. Aussi proche soit-elle de l'utilisateur, cependant, un temps d'adaptation est toujours nécessaire à sa totale appropriation. L'écrivain doit pratiquer une machine à écrire ou un traitement de texte avant de l'utiliser pour sa création ; un infographiste doit avoir joué des heures avec les outils et les filtres de son dernier outil avant de pouvoir se laisser aller à écouter son inspiration pour produire ses œuvres ; un programmeur doit avoir exploré l'intégralité des menus de son interface de déboguage avant de pouvoir l'utiliser en ne pensant plus qu'à la correction des erreurs.

Or les applications évoluent sans cesse, améliorant leur capacités en acquérant de nouvelles fonctionnalités, gérant des types d'objets toujours plus complexes et aux caractéristiques plus paramétrables. Pour une simple tâche de rédaction, nous sommes passés de la technologie des machines à traitement de texte autorisant une demi-douzaine de polices de caractères, aux outils de mise en page permettant de produire en quelques heures des documents de qualité professionnelle. Et pourtant, les concepteurs de logiciels ont vu leur boulimie de fonctionnalités réfrénée par les utilisateurs, incapables d'appréhender des applications trop paramétrables, trop riches en possibilités, aux manuels interminables. Le vrai problème des concepteurs de logiciels n'est plus d'inventer de nouvelles fonctions mais trouver comment supprimer toutes les fonctions trop rarement utilisées et qui peuvent être avantageusement remplacées par la combinaison d'opérations élémentaires, légèrement plus lente mais plus accessible aux utilisateurs finaux.

Si les concepteurs d'application ont été contraints de limiter l'accès aux fonctionnalités évoluées dans les interfaces, l'utilisateur est tout de même submergé par la quantité d'outils nouveaux qui sont mis chaque jour à sa disposition pour faire ceci ou cela. En effet, les logiciels ne pouvant proposer d'emblée une pléthore de fonctionnalités, ils deviennent modulaires, permettent une intégration poussée avec d'autres logiciels, ou bien peuvent se faire adjoindre de petits modules (des *plugins*). Ainsi, le butineur *Mozilla*, simple d'usage en standard, peut recevoir plus d'une centaine de modules, permettant d'envoyer des messages électroniques cryptés (*enigmail*), de bloquer tous les *pop-ups* agaçants et les publicités diverses qui foisonnent dans certains sites (*adblock*), d'accéder à de multiples moteurs de recherches (*mycroft*), de rechercher rapidement une chaîne de caractères dans les pages visitées ou d'accéder à des informations sur la recherche (*googlebar*), etc.

La situation est encore pire pour les utilisateurs de distributions émanant du logiciel libre (RedHat-Linux, FreeBSD, Debian-Hurd), où l'utilisateur est confronté à une quantité pléthorique d'applications disponibles. En général, la première étape de l'utilisation d'une telle distribution est consacrée à l'installation des logiciels, les utilisateurs passent chaque application en revue, en évaluant son utilité potentielle d'après le commentaire qui lui est associé. Souvent on a tendance à installer plus d'applications que nécessaire, par peur de rater l'outil indispensable. Mais de toute façon, une fois l'installation effectuée, il est impossible de revenir sur chaque élément installé pour apprendre à l'utiliser, l'évaluer, et finalement choisir ou pas de l'adopter. Au final, c'est un énorme gâchis de potentiel car rares sont les utilisateurs capables de profiter pleinement de la palette d'outils mis à leur disposition dans ces distributions.

En conséquence, après avoir été sauvé de la noyade dans les logiciels, c'est finalement dans la

liste interminable des applications disponibles que l'utilisateur arrive à se perdre. Ne sachant pas quels outils utiliser dans quels cas, il se restreint à quelques applications classiques, perdant le bénéfice que pourrait lui procurer tous ces logiciels dont il ne connaît parfois même pas l'existence. Lorsque cela lui devient absolument nécessaire, il lui faut chercher, installer, configurer et finalement apprendre à utiliser ces nouveaux logiciels. Une perte de temps considérable, mais pour un gain de temps et de qualité suffisant pour la rendre acceptable.

Là où cette perte de temps devient inacceptable, c'est quand elle doit être reproduite quelques mois plus tard parce qu'une nouvelle version est sortie ou qu'un logiciel concurrent est apparu sur le marché. On ne peut que craindre l'intensification de cette situation, forçant l'utilisateur à consacrer toujours plus de temps à apprendre de nouvelles fonctionnalités pour rester au meilleur niveau.

En conclusion, le *comment faire* prend petit à petit le pas sur le *quoi faire* dans l'esprit de l'utilisateur, réduisant le potentiel cognitif qu'il peut consacrer à exprimer sa créativité et ses connaissances propres, la réelle valeur ajoutée de son travail. Il est donc nécessaire de penser à des systèmes de dialogues finalisés destinés à assister l'utilisateur dans l'utilisation de son ordinateur. Dans la suite, nous distinguons plusieurs aspects différents pour cette assistance.

Aide sur le contenu

L'agent assistant doit pouvoir renseigner l'utilisateur sur les informations contenues dans une application ou gérées par elles. Par exemple, sur l'arborescence des fichiers et répertoires (« de quelle taille est le répertoire "/usr/share" ? »), les messages électroniques (« de qui est le dernier message que j'ai reçu contenant une image ? »), les parties d'un document texte (« combien est-ce qu'il y a de chapitres ; qui sont les auteurs ? »).

Cet aspect recoupe les diverses formes de questionnement que l'on peut trouver dans les systèmes d'interrogation de bases de données. La principale difficulté pour un tel système est que les applications n'exposent pas nécessairement leur contenu sous une forme aussi facilement accessible qu'une base de données.

Assistance sur les fonctionnalités

Un agent assistant d'application doit pouvoir renseigner l'utilisateur sur ses capacités (et donc indirectement sur les capacités de l'application que l'agent assistant médie). Ceci implique que la connaissance de l'agent assistant doit englober les compétences de l'application médiée, mais aussi ce que l'application ne sait pas faire mais qu'un utilisateur pourrait attendre d'elle.

Autrement dit, l'agent assistant doit être capable de reconnaître des fonctionnalités qui ne sont pas traitées par l'application pour laquelle il sert d'interface. Exemples :

– « Est-ce que je peux transformer un fichier PostScript en PDF ? »,
– « Peux-tu envoyer ce message en crypté ? »
– « Peux-tu me donner la météo de demain ? »

Assistance sur le fonctionnement

L'agent doit être capable de comprendre les références faites par l'utilisateur sur la dynamique de l'exécution. Par exemple,

– « Montre-moi les courriels que tu as classés comme indésirables ? », ou bien
– « Où as-tu classé les courriers de Jean-Paul ? »,
– « Pourquoi as-tu mis ce fichier à la poubelle ? »

Cette compétence nécessite non seulement que l'agent conserve une mémoire de ses actions et des actions de l'application, mais aussi qu'il ait une capacité de raisonnement sur ces actions (Sabouret and Sansonnet, 2001).

Assistance d'exécution

L'agent assistant doit être capable de réaliser seul des tâches impliquant des compétences de bas niveau dans les applications utilisés par l'utilisateur. Exemples :

– « Envoies à Jean-Paul une version zippée de ma thèse avec juste les images, le fichier BIB et le fichier LYX »,
– « Mets toutes les images dans un nouveau répertoire et fait une archive »,
– « Inclus les images dans ces fichiers à la fin de mon manuscript ».

Cette compétence nécessite des capacités de raisonnement et de planification évoluées. Ce point a été particulièrement traité dans (Sabouret and Sansonnet, 2001).

Dialogue

Dans le cas où certaines informations manquent à l'agent, ou bien si son degré de certitude sur ces informations n'est pas suffisamment élevé, il doit prendre l'initiative du dialogue pour obtenir de l'utilisateur les données qui lui font défaut. Exemples :

– Util : « Mets toutes les images dans un nouveau répertoire et archive-les »
– Syst : « Comment dois-je appeler le nouveau répertoire ? »
– Util : « Images »

1.3.2 Généricité

Dans leur version idéale, les agents assistants d'interface doivent pouvoir être utilisés pour n'importe quelle application. Cela ne signifie pas qu'il doive exister un agent général capable de comprendre toutes les applications, mais que chaque application doit pouvoir être dotée d'une interface de médiation dialogique, intégrable naturellement dans l'interface générale, de même que les interfaces graphiques s'intègrent dans des fenêtres et communiquent avec le système de fenêtrage pour gérer l'icônification ou le redimensionnement. La même préoccupation est exprimée dans le projet OZONE (Gaiffe et al., 2004). Ce projet vise à permettre offrir un mode d'interfaçage standard en langue naturelle pour des applications d'usage général, mais aussi de permettre que plusieurs applications puissent utiliser la même interface, sans passer par une application centrale. La solution explorée dans OZONE consiste à utiliser une architecture multi-agents afin d'assurer la traduction entre les caractéristiques propres à une application et le coeur de la représentation du système.

La plupart des autres systèmes de dialogue, comme par exemple TRIPS (Ferguson and Allen, 1998), ne permettent pas d'intégrer plusieurs applications dans une même interface. En revanche, ils permettent de passer relativement facilement d'une application à une autre grâce

à des modules de traduction d'ontologies (Dzikovska et al., 2003). Dans leur approche, un lexique général est défini une fois pour toutes, et les différents modules du système se basent sur ce lexique. Ensuite, pour adapter le système de dialogue à une nouvelle application, une ontologie spécifique est créée pour pouvoir décrire les différents éléments de celle-ci, et un module de traduction est adjoint au système pour faire l'adaptation entre les concepts spécifiques à l'application et les concepts génériques manipulés par le système central. Par exemple, si dans une application, les oranges jouent le rôle de cargaison, alors une règle de traduction permettra de passer du concept « orange » au concept « cargaison », qui est connu du système de base.

1.3.3 Rôle de médiateur

Un agent assistant d'interface se distingue d'une simple interface en langue naturelle par le fait qu'il dispose de compétences de communication qui sont indépendantes de l'application à laquelle il sert d'interface. En effet, une application classique n'est pas censée se « souvenir » des actions passées de l'utilisateur, et même si la plupart offrent des possibilités de défaire des actions ou bien d'avoir un historique des documents consultés, quasiment aucune ne donne accès à la liste des commandes passées par l'utilisateur. Bien évidemment, rien n'empêche de le faire, mais l'intérêt de disposer de cette information dans une interface classique est très limité. En revanche, dans le cadre d'une interface naturelle, où un unique énoncé peut désigner une séquence complète d'actions atomiques, il devient très intéressant (et même indispensable si l'on veut pouvoir interpréter les anaphores) de conserver les ordres passés.

De plus, une interface naturelle doit être capable de supporter les requêtes comportant des éléments inconnus et répondre à l'utilisateur en lui révelant son incapacité à effectuer l'action demandée. C'est un problème qui ne peut pas survenir lorsque l'interface est définie statiquement, comme par exemple avec une interface graphique, puisque l'interface contraint les actions accessibles à l'utilisateur.

Enfin, alors qu'une interface graphique classique ne présente que les actions atomiques accessibles à un instant donné, l'intérêt d'un agent assistant est de donner accès à des commandes complexes. Cette interface doit donc avoir une connaissance propre des commandes complexes afin de construire les séquences d'actions atomiques qui réalisent l'action demandée. L'agent assistant doit donc disposer de méta-connaissances sur l'application.

Finalement, l'agent assistant d'interface se comporte comme un intervenant humain coopérant avec l'utilisateur, connaissant à la fois les possibilités d'une application et les besoins de l'utilisateur qui veut utiliser cette application sans en connaître les mécanismes internes. Pour cette raison, nous considérons que l'agent assistant d'interface se comporte comme un médiateur dans un schéma de communication à trois pôles (ce schéma est donné dans la figure 10.1 au chapitre 10).

L'approche par médiateur consiste à considérer que l'agent conversationnel doit être modélisé comme un assistant humain, c'est-à-dire principalement une entité douée de perception et d'une capacité motrice (bien qu'aucun extrémisme ne soit nécessaire ni pour l'un, ni pour l'autre : les perceptions comme les actions des médiateurs peuvent être encodées par des schémas de haut niveau, tant que ces schémas[5] conviennent à la finalité de l'interface). En

[5]Dans cette partie préliminaire à toute formalisation, la notion de schéma ne correspond pas à une conception bien définie, elle est utilisée ici de manière volontairement vague.

plus de ces capacités, qui permettent au médiateur :

1. de recevoir les énoncés provenant de l'utilisateur,
2. de percevoir un modèle de l'application médiée,
3. de produire une action sur l'application,
4. de produire un énoncé en réponse à l'utilisateur,

le médiateur dispose de compétences sur l'application, qui lui permettent d'élaborer des schémas de commande plus ou moins complexes, et dispose de compétences linguistiques en rapport avec l'application, qui lui permettent de relier certains schémas avec des actions à effectuer sur l'application. Il est important de noter que le médiateur doit, de par sa capacité à percevoir l'application, disposer d'une description d'interface qui combine à la fois :

1. les caractères de l'interface utilisateur graphique (communément dénommée *GUI*),
2. les caractères plus profonds de l'application, qui ne sont pas nécessairement visibles, mais sont nécessaires pour disposer d'une connaissance totale du logiciel, et sont de toutes façons présupposés par l'utilisateur.

Ces différentes compétences nécessitent, dans le médiateur une structuration particulièrement travaillée de la représentation de ce que l'utilisateur perçoit du logiciel. Une « simple » représentation parallèle au modèle informatique ne saurait suffire à cette fin, contrairement à ce que nous avions initialement posé comme hypothèse dans le système InterViews.

Conclusion

L'idée d'une interface naturelle capable de servir de médiateur entre un utilisateur humain et un logiciel est née des objectifs initiaux de l'Intelligence Artificielle, mais a beaucoup évolué, suite à différentes déconvenues dans la réalisation de ces objectifs. Le problème de la langue et de la compréhension de celle-ci a longtemps été au coeur des différents systèmes qui ont été construits dans ce but, mais les limites de ces systèmes se sont vite révélées, en ce qui concerne leur manque de généricité.

Les systèmes multimodaux ont alors émergé pour, dans un premier temps tout du moins, simplifier le travail de désambiguïsation que pouvait nécessiter le traitement de la langue. Assez rapidement, ces systèmes ont permis de mettre en avant l'importance de la prise en compte du contexte, et notamment, de la prise en compte des référents accessibles pour le traitement des expressions référentielles.

Nous avons donné un aperçu rapide des différentes familles de systèmes visant à un interfaçage naturel entre l'utilisateur et une application logicielle. Nous avons décrit les motivations premières de ces différents systèmes, en mettant en avant le rôle qu'ils devraient jouer pour répondre réellement aux besoins des utilisateurs finaux. Le point qui nous a parut extrêmement important, et qui est aujourd'hui assez partagé (Allen et al., 2000; Gaiffe et al., 2004), concerne la généricité des systèmes de dialogue, c'est-à-dire leur capacité à être adaptés et intégrés rapidement à des situations nouvelles (typiquement, permettre de communiquer avec

un nouveau logiciel dans le même « cadre », si possible en donnant accès simultanément à plusieurs logiciels). Cette nouvelle génération de systèmes de dialogue, pas encore idéaux, mais nettement plus prometteurs, nous le désignons par le terme « *agents assistants d'interface* ».

A la question « quelles sont les solutions technologiques à mettre en place pour créer cette nouvelle génération ? », nous argumentons dans le chapitre suivant en faveur de trois choix théoriques majeurs :

1. Les grammaires de construction, héritières des grammaires cognitives,
2. Une architecture unifiée, fondée sur un unique principe de traitement,
3. Une prise en compte au premier ordre des structures d'informations dotées de topologies.

Chapitre 2

Modèles d'interprétation

Pourquoi choisir le terme de « modèles d'interprétation » ? Dans la problématique du dialogue homme-machine, plusieurs termes sont utilisés pour nommer les modèles permettant de passer d'un énoncé à une action quelconque. On parle ainsi de modèles de traitement de la langue, de modèles d'analyse, de modèles de compréhension ou encore de modèles grammaticaux. Notre choix s'est porté sur le terme « interprétation », et une explication de notre choix permettra de se faire une premier idée de notre attitude vis-à-vis de la langue, comparée aux autres sources d'information qu'un agent assistant d'interface peut obtenir. En effet, pour nous, les énoncés ne sont qu'une source parmi d'autres, et à ce titre, son traitement doit se faire dans un cadre identique aux traitements des autres sources.

La plupart des théories linguistiques présentées dans les sections suivantes, et notamment les plus récentes, portent dans leur dénomination le terme « grammaire ». Dans son acception classique cependant, une grammaire est un code normatif qui décrit les règles permettant de déterminer qu'un énoncé dans une certaine langue est formé de manière correcte ou non. La définition de « grammaire » dans l'encyclopédie publiée par l'ATILF (déf. 1) précise qu'en tant que science, la grammaire concerne l'étude des éléments constitutifs d'une langue, de la phonétique à la syntaxe en passant par l'orthographe et la morphologie, mais ne s'intéresse normalement pas aux conditions d'utilisation de la langue, et donc à la sémantique et à la pragmatique (quelles que soient les sens donnés à ces deux termes). Pourtant, il existe une nette tendance dans les théories linguistiques à étendre la portée de la grammaire aux notions sémantiques et même pragmatiques (par exemple pour HPSG ou la grammaire cognitive).

Le terme « grammaire » est donc trop ambigu et de toute façon trop restreint pour exprimer la problématique qui nous intéresse, et qui concerne le traitement des énoncés dans le cadre d'un dialogue finalisé avec support extralinguistique (visuel ou autre).

Choisir le terme « compréhension » aurait impliqué au contraire que l'on cherche à reproduire une caractéristique de l'être humain que l'on ne sait pas encore bien définir, et qui nécessite probablement une modélisation non seulement du monde extérieur, déjà difficile à envisager, mais aussi une modélisation du monde intérieur, avec tout ce que cela implique sur les questions des émotions et de l'incarnation. En effet, on peut comprendre la douleur de quelqu'un, comprendre ses intentions, au même titre qu'on peut comprendre ce qu'il nous dit. Or notre propos ici n'est ni de modéliser un humanoïde doué de sentiments, ni de résoudre le problème des émotions. Le terme « compréhension » n'est donc pas convenable pour désigner ce qui devrait être traité par notre modèle.

GRAMMAIRE ***n. f.*** **XIIe siècle. Dérivé du latin grammatica, emprunté du grec grammatikê, « grammaire, culture ».**

1. **Ensemble des règles qui forment le système d'une langue et que l'on doit suivre pour parler ou écrire conformément à l'usage. Étudier la grammaire du français. Savoir sa grammaire. Discuter un point de grammaire. Une faute de grammaire, contre la grammaire.** ***Anciennt.*** **Classes de grammaire, les classes de sixième, cinquième et quatrième de l'enseignement secondaire, qui précédaient les classes de lettres.**
2. **Science qui a pour objet l'étude systématique des éléments constitutifs d'une langue, comprenant notamment la phonétique, l'orthographe, la morphologie, la syntaxe. Grammaire historique. Grammaire descriptive, qui se borne à analyser et décrire les faits de langue, par opposition à Grammaire normative, qui énonce des règles, présente des modèles. Grammaire comparée, qui étudie les rapports entre diverses langues dérivant le plus souvent d'une source commune. Grammaire comparée du grec et du latin. Grammaire générale, qui cherche à déterminer des lois valables pour toutes les langues. Grammaire structurale, fondée sur les principes d'analyse formelle de la linguistique structurale. Grammaire générative, voir Génératif.** ***Spécialt.*** **Agrégation de grammaire, qui se distingue de l'agrégation de lettres par son programme étendu de philologie. Titre célèbre : Grammaire générale et raisonnée, dite Grammaire de Port-Royal (1660).**
3. **Ouvrage où sont exposées les règles qui régissent l'usage d'une langue. Composer une grammaire grecque, une grammaire latine. Consulter une grammaire.**
4. **Par anal. Ensemble des principes, des règles qui permettent de s'initier à un art, à une technique.**

Définition 1: *Grammaire* (ATILF)

Le terme « analyse » est plus proche de ce qui nous intéresse, mais d'une part il ne transmet pas la notion d'action en réponse à un énoncé et d'autre part, est un peu trop vague et générique pour désigner à lui seul l'objectif de notre modèle. Une analyse peut en effet être faite à différents niveaux : morphologique, syntaxique, pragmatique, etc. Or notre modèle a pour objectif de couvrir ces différents niveaux et de produire finalement une action adaptée à ce qui est exprimé dans un énoncé.

Le terme « traitement » quant à lui se focalise sur l'aspect opérationnel de la transformation d'un énoncé, et pose le même problème de généralité que le terme « analyse ».

Finalement, c'est le terme « interprétation » qui nous a semblé le mieux adapté pour décrire notre problématique et notre objectif. Tout d'abord, il n'implique pas de réduire le problème aux questions linguistiques, et donc permet de prendre en compte les aspects sémantiques et pragmatique de la communication, mais aussi les aspects purement extra-linguistiques. D'autre part, le terme « interprétation » désigne selon nous un acte plus restreint que celui désigné par « compréhension », en ne requérant pas que le sujet dispose des mêmes caractéristiques que le locuteur du message dont il est question, et donc en autorisant qu'un ordinateur puisse en être l'auteur.

2.1 Théories linguistiques

2.1.1 Grammaires d'unification

Les grammaires d'unification sont fondées sur une représentation des éléments du lexique sous la forme de structures de traits. Une structure de traits est un ensemble de couples attri-

but/valeur où la valeur est soit une valeur atomique, soit une valeur partagée (avec un autre trait de la structure globale), soit une structure de traits. Cela fait des structures de traits des structures équivalentes à des graphes acycliques dirigés (DAG). La grammaire GPSG (Gazdar et al., 1985) (*Generalized Phrase Structure Grammar,* Grammaire Syntagmatique Généralisée) et la grammaire HPSG (Pollard and Sag, 1994) (*Head-driven Phrase Structure Grammar,* Grammaire Syntagmatique Guidée par les Têtes) sont deux grammaires d'unification parmi de nombreuses autres telles que les CUG (*Categorial Unification Grammar*), FUG (*Functional Unification Grammar*) et LFG (*Lexical-Functional Grammar*).

Le gros avantage des grammaires d'unification est qu'elles permettent d'exprimer des dépendances et les contraintes d'agrément (en genre, nombre, cas, etc.) en utilisant les informations d'une seule représentation, là où il fallait plusieurs niveaux dans la grammaire générative de Chomsky (1982). De plus, les structures de traits étant une modalité de représentation très souple, ces grammaires ont été pensées pour intégrer des traits sémantiques.

Ces différentes grammaires sont aussi fondées sur une notion de contrainte, mais ne les utilisent que pour vérifier les structures possibles construites à partir du lexique. Ces grammaires n'utilisent pas réellement la satisfaction de contraintes comme mécanisme opérationnel (Blache, 2000), mais simplement pour filtrer les bonnes structures d'arbres. C'est un reproche qui est juste, mais on peut le contrer en considérant que ces différentes grammaires ont été proposées aussi pour rendre compte de la langue d'une manière psychologiquement plausible, contrairement aux premières grammaires génératives (voir chapitre 5). Or une approche purement fondée sur la satisfaction de contraintes ne semble pas répondre à ce critère.

Un aspect particulièrement intéressant de HPSGHPSG est la possibilité de définir des règles générales sur les entrées lexicales, permettant de produire de nouvelles entrées à partir des définitions de base. De plus, les entrées lexicales sont intégrées à une hiérarchie de types, fournissant un mécanisme d'héritage facteur d'importantes économies dans l'écriture de la grammaire.

2.1.2 Grammaire Cognitive

La grammaire cognitive est une théorie de la langue établie par Langacker (1987) avec pour objectif de rendre compte de la langue en tant qu'élément indissociable de la cognition, ce qui était déjà proposé dans (Talmy, 1978). Notamment, les relations entre les schémas perceptuels et la langue (Talmy, 1983), largement absents des grammaires génératives sont, selon les tenants d'une linguistique cognitive, requises pour mener à bien une interprétation correcte.

La grammaire cognitive se définit largement en opposition à l'approche suivie dans la tradition initiée par Chomsky, qui veut que la langue soit d'abord un langage formel descriptible grâce aux outils mathématiques. L'approche cognitive considère au contraire que la langue est d'abord un moyen d'évoquer des schémas cognitifs socialement partagés, et qu'elle s'est naturellement structurée en tendant vers un squelette formalisable.

La grammaire cognitive telle qu'elle était présentée dans l'ouvrage de Langacker (1987) a été largement critiquée, à juste titre, pour son manque de formalisation et le fait qu'aucune piste sérieuse n'était proposée pour mettre en œuvre un système de compréhension de la langue basé sur cette approche. La grammaire cognitive propose cependant l'idée que le lexique « cognitif » décrirait les mots directement reliés aux schémas cognitifs qu'ils peuvent évoquer, idée qui a été reprise dans la grammaire de construction.

L'intérêt majeur de l'ouvrage de Langacker a été de mettre en avant la récurrence de certains schémas cognitifs (notamment le schéma « *trajector - landmark* » que je traduis par « trajecteur - repère »). Sa faiblesse majeure est qu'il ne propose pas de pistes pour utiliser ces idées de manière opérationnelle.

2.1.3 Grammaire de Construction

La grammaire de construction (GC) (Fillmore, 1988; Fillmore and Kay, 1995) est un formalisme grammatical qui rompt radicalement avec les approches générativistes et transformationnelles. L'idée phare de la GC est que non seulement les mots, mais aussi toute forme linguistique, comme les motifs syntaxiques ou les motifs de dialogue, est associée à une contrepartie sémantique. Cette conception de la langue et de la compositionnalité, c'est-à-dire la capacité à produire de nouveaux sens en combinant des symboles existants, est significativement différente de celle développée depuis Frege, qui affirme que le sens est directement déductible de la structure syntaxique à partir d'un lexique associant mots et sémantique.

La GC permet notamment de rendre compte de manière très élégante du fait que certaines constructions linguistiques sont porteuses d'un sens qui ne peut pas être construit à partir de leurs composantes, par exemple la construction ditransitive en anglais :

(E1) *Barbara baked Emma a cake* (Barbara a cuisiné un gâteau à Emma)

La construction anglaise ditransitive « *X verbe Y GN* » implicite la notion de don (le gâteau une fois cuisiné par Barbara est donné à Emma) dans de nombreux cas. De nombreux verbes peuvent remplacer le verbe *cook* (*buy*, *bring* par exemple), sans nécessairement que la notion de don ne soit explicitement ou implicitement associée à eux. De même, ce n'est pas dans le GN qu'on peut trouver le sens du don. C'est donc bien la construction « *X verbe Y GN* » qui est associée à une notion de don.

Goldberg (1995) montre la nécessité d'associer les constructions de surface à des aspects sémantiques en relevant le fait que, dans une approche strictement compositionnelle, rendre compte d'un phénomène comme la construction ditransitive nécessiterait d'enrichir le lexique avec des versions ditransitives de tous les verbes pouvant entrer dans ce cadre. Par exemple il faudrait un sens de « *to bake* » qui puisse prendre deux arguments et qui serait alors lié à une notion « faire quelque chose avec l'intention de le donner à quelqu'un ». Cela conduirait bien entendu à une inflation du lexique et à un recul par rapport à l'objectif premier de l'approche compositionnelle, qui est de rendre compte de la productivité de la langue, c'est-à-dire de la capacité à créer des sens non-appris à partir de sens appris.

Une caractéristique majeure de la GC est d'associer **directement** forme et sens, sans passer obligatoirement par une étape syntaxique intermédiaire, où seules les catégories syntaxiques et les dépendances grammaticales sont exprimées. En cela, la GC hérite de la grammaire cognitive (Langacker, 1987), qui considérait déjà que la grammaire ne devait pas marquer une séparation entre syntaxe et sémantique, mais plutôt mêler les deux. Sans nier l'importance de la syntaxe, la GC ne la considère pas comme une étape préliminaire obligatoire, mais comme une composante qui intervient de manière transversale dans l'élaboration d'une représentation sémantique.

La GC met aussi l'accent sur l'importance des schémas cognitifs dans l'interprétation sémantique. Contrairement aux grammaires comme HPSG, qui ne s'intéresse à la sémantique que du point de vue des rôles joués par les arguments des verbes, la GC considère que les schémas cognitifs peuvent être évoqués par différents éléments, et donc ne se limitent pas à des rôles sémantiques préféfinis de type *agent, objet, patient, etc* (rôles dont l'établissement d'une liste est d'ailleurs très polémique). Cette approche provient des travaux en linguistique cognitive, qui ont permis de mettre en avant le fait que certains schémas mentaux interviennent dans de nombreuses configurations, et peuvent être combinés entre eux. Par exemple la notion de **conteneur** implique certaines associations (l'intérieur, l'extérieur, la frontière) qui peuvent être évoquées à la fois lorsque l'on parle d'une bouteille que lorsque l'on parle d'une chambre.

2.1.4 Grammaire ECG

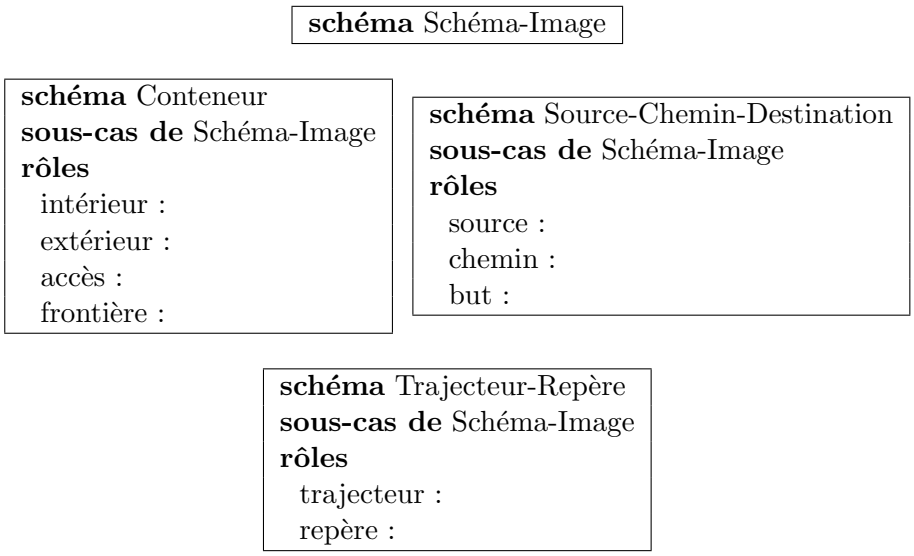
La grammaire ECG (*Embodied Construction Grammar*, Grammaire de Construction Incarnée) proposée par Chang et al. (2002) et Bergen and Chang (2002) est dérivée de la grammaire de construction. Elle a été conçue afin de répondre à deux des problèmes de la GC.

Le premier problème est que la définition proposée par Fillmore and Kay (1995) n'est pas suffisamment formalisée pour en déduire une implémentation informatique. L'expressivité de la grammaire de construction pose d'ailleurs d'importantes questions sur la faisabilité d'une telle implémentation, et aucune mise en œuvre de la GC n'avait pu être menée à bien jusqu'à maintenant.

Le second problème est que la faculté de simulation mentale n'est pas prise en compte dans la GC. En effet, l'interprétation d'un énoncé ne consiste pas simplement à construire une représentation sémantique, l'allocutaire essaye aussi de s'imaginer (construire une simulation mentale de...) la situation décrite ou implicitée par l'énoncé. Les inférences faites à partir de cette simulation font partie intégrante du « sens » attribué à l'énoncé. Or si la GC a pour objectif d'utiliser en parallèle les informations syntaxiques et sémantiques, elle n'inclut en revanche pas le niveau de traitement pragmatique, et c'est pour cette raison que l'ECG est conçue pour pouvoir être relayée par un système de simulation basé sur les schémas d'exécution (Narayanan, 1997; Bailey et al., 1997), modèle fonctionnant sur un principe différent des GC.

Quatre types d'entités permettent de décrire une grammaire ECG : les **schémas**, les **constructions**, les **appariements** et les **espaces mentaux**.

Les schémas sont des structures abstraites qui décrivent formellement une représentation conceptuelle comme un ensemble de **rôles** et de **contraintes** sur ces rôles. Les rôles peuvent être définis par un simple nom ou bien être spécifiés par une restriction qui est soit un schéma soit un type dans une ontologie externe.
Les schémas permettent de représenter différentes structures d'information, dont les structures de trait de HPSG (Pollard and Sag, 1994), les schémas visuels de (Lakoff and Johnson, 1980), illustrés dans la spécification 1.
Un mécanisme de sous-catégorisation permet de définir la relation entre un schéma A et un schéma B plus générique, le schéma A *hérite* alors automatiquement des rôles et contraintes définies dans le schéma B. Un schéma peut aussi **évoquer** des schémas auxquels il est fortement corrélé.
Ce mécanisme d'évocation permet une plus grande souplesse dans la conception, selon les



Spécification 1: Schémas cognitifs primitifs

auteurs. Un exemple d'utilisation est donné à la spécification 2. On peut voir dans cette figure le schéma utilisé pour représenter la notion de déplacement dans un conteneur, présent par exemple dans « Mets le courrier *dans* le dossier "urgent" ». Ce schéma est hérité du schéma trajecteur-repère (*Trajector-Landmark* en version anglaise), qui décrit une relation d'un élément avec un autre, où un élément (le trajecteur) est considéré comme défini *par rapport* à l'autre (le repère). Mais la notion *déplacement-dans* implique aussi un déplacement d'un point A à un point B, qui est instanciée par un schéma source-chemin-destination (*Source-Path-Goal* en version anglaise), représentant comme son nom l'indique un déplacement dont le point de départ est la *source*, le point d'arrivée est la *destination* et le trajet suivi est le *chemin*.

Les constructions sont, dans les modèles hérités des Grammaires Cognitives, les éléments de base de la connaissance linguistique. Ce sont des paires qui définissent le lien entre une *forme* et sa *signification*. Une construction est définie principalement par ces deux blocs, qui définissent les rôles et les contraintes de ces deux pôles, d'une manière similaire aux schémas. Une construction peut aussi spécifier d'autres instances de constructions sur lesquelles elle s'appuie[1].

Les appariements (*map*) servent à exprimer des liens de similarité entre deux entités, afin de décrire les transitions inter-domaines, principalement utilisées pour les métaphores, les inférences métaphoriques ou les métonymies. Un appariement permet de décrire les liens entre les sous-rôles des types de schéma impliqués[2].

[1] Ces instances sont appelées les constituants, par exemple dans la phrase « à droite de la chaise », le lien forme-sens de « à droite de » est définie par une construction **préposition-spatiale** et « la chaise » par une construction **expression-référentielle**. La construction qui fait le lien entre « à droite de la chaise » et sa signification aura ces deux constructions comme constituants, et pourra donc spécifier des contraintes sur leurs rôles.

[2] Par exemple, la métaphore « l'amour est un voyage » qui apparaît dans des expressions comme « notre relation est dans une impasse », « je me suis embarqué dans une aventure », « elle a continué son chemin seule ». Pour cette métaphore, l'appariement doit établir les relations entre les descriptions des deux concepts :

Les espaces mentaux (*mental spaces*) permettent de structurer les schémas établis par les constructions afin de distinguer les différents points de vue. Cette notion est bien entendu directement inspirée des travaux de Fauconnier (1984). La manière précise dont les espaces mentaux sont utilisés dans la grammaire ECG reste malheureusement très allusive, à l'image de la définition même des espaces mentaux.

La grammaire ECG est l'un des plus aboutis des modèles de linguistique cognitive du point de vue de la formalisation. Bryant (2003) a réalisé pour l'ECG un analyseur basé sur le *chunk parsing* de Abney (1991) et le *chart parsing* (Earley, 1986). Cette réalisation, cependant, ne couvre pas l'intégralité de la grammaire ECG.

schéma Déplacement-Dans
sous-cas de Trajecteur-Repère
évoque scd : Source-Chemin-Destination
rôles
  repère : Conteneur
contraintes
  scd.chemin ↔ repère.accès
  scd.source ↔ repère.extérieur
  scd.but ↔ repère.extérieur

Spécification 2: Le schéma déplacement-dans, correspondant à un déplacement dont la cible est un conteneur, par exemple dans : « *Mets le message dans la corbeille* »

Il est enfin intéressant de noter que la grammaire ECG est développée dans le cadre du projet NTL (Neural Theory of Language) qui travaille notamment sur l'hybridation entre systèmes connexionnistes et systèmes symboliques ainsi que sur les fonctionnement cognitifs portant sur les relations spatiales et la mémoire, entre autres. Ceci nous semble être un encouragement assez fort dans l'idée que les différentes théories linguistiques devraient rejoindre les théories cognitives plus générales, ce dans le sens duquel nous allons.

2.2 Architectures Modulaires *versus* Unifiées

Dans ce qui suit, nous décrivons deux grands courants architecturaux pour les systèmes de traitement cognitifs et plus précisément pour les systèmes de traitement de la langue. D'un côté, l'approche modulaire, qui consiste à considérer que l'on peut isoler des sous-ensembles du traitement. De l'autre côté, l'approche unifiée, qui consiste à dire que les différentes fonctions (cognitives ou linguistiques) fonctionnent selon les mêmes principes et qu'elles ne sont pas isolables *a priori*.

Cette distinction trouve son principal fondement dans les débats psychologiques portant sur la nature de la cognition, mais elle a trouvé un écho très naturel dans le domaine linguistique. Contrairement à la psychologie, où les approches unifiées sont relativement bien établies, les architectures de traitement de la langue restent en grande majorité orientées vers une approche

l'amour (l'amoureux, le but, la relation, les difficultés) et le voyage (le voyageur, la destination, le véhicule, les obstacles).

modulaires, principalement par nécessité. Du point de vue des théories linguistiques, il est en effet difficile de trouver un socle commun permettant de toutes les représenter. Pour l'aspect pratique, d'autre part, modulariser permet de mieux se figurer le schéma de fonctionnement global (on verra que cet argument ne permet en fait pas de favoriser les approches modulaires).

En conclusion, nous montrons que, conformément à l'intuition, une architecture unifiée est souhaitable pour de nombreuses raisons, à la fois d'ordre théoriques, mais aussi pratiques.

2.2.1 Architectures Modulaires

Architectures cognitives modulaires

Dans le domaine psychologique, et plus précisément pour la théorie des systèmes cognitifs, adopter une approche modulaire consiste à supposer qu'il existe des systèmes de traitement autonomes (correspondant à des fonctions cognitives identifiables, c'est-à-dire nommables) qui fonctionnent selon leur principes propres (Fodor, 1983). L'autonomie se traduirait par exemple par le fait que le fonctionnement du module visuel ne serait **absolument** pas influencé par le fonctionnement du module auditif. Fodor a ainsi identifié le système visuel comme le système modulaire par excellence (voir aussi Pylyshyn (1999)). Le fait que l'écoute d'un certain bruit puisse nous faire « voir » différemment est donc selon lui impossible. S'il y a influence, c'est à un autre module que reviendrait la responsabilité de la prendre en compte.

Par analogie, on peut supposer que le niveau de traitement syntaxique de la langue pourrait lui aussi être autonome, hypothèse largement soutenue par l'approche générativiste en linguistique. Pourtant plusieurs expériences (Kim et al., 2003; van Herten et al., 2003) semblent montrer que le traitement syntaxique ne précède pas nécessairement le traitement sémantique, et donc que les deux fonctions (si toutefois il est juste de considérer que ce sont des fonctions séparées) s'influencent mutuellement. D'autre part, même parmi les tenants de l'hypothèse modulaire, la possibilité que les niveaux de cognition supérieurs impliquent eux aussi de tels modules est mise en question, l'argument principal étant que les niveaux de cognition supérieurs nécessitent de rassembler des connaissances de tant de sources différentes qu'il est impossible de les isoler dans des composants séparés.

Un des arguments pour mettre en doute la modularité des fonctions hautes est l'impossibilité de conserver isolément les buts et les connaissances à long terme ou encore les points de vue sur un problème donné qui seraient encapsulés. Il faudrait donc pour faire intervenir des interfaces d'échanges où certaines informations pourraient être co-manipulées par différents modules. Dans le système ACT-R de Anderson (1993), que celui-ci considère comme modulaire, la solution adoptée consiste à utiliser des buffers de partage d'information entre les modules. Bien que cela résolve en apparence le problème de la modularité, cela ne fait en fait que confirmer les critiques de cette approche, car toute utilisation de ces espaces d'échange crée en réalité un second acteur dans l'architecture de la cognition, rompant avec la stricte autonomie des fonctions cognitives, et donc avec la vision modulaire pure. En effet, si deux modules doivent échanger des informations via un espace « tampon », c'est qu'il y a intersection des deux modules. Cela signifie que leurs « principes de fonctionnement » doivent être suffisamment compatibles pour supporter un échange. Or dans l'approche modulaire, les modules sont censés pouvoir suivre des principes de fonctionnement totalement différents.

Si le système de production d'ACT-R possède une certaine modularité, et pallie le problème de l'interaction entre les modules qui possèdent de nombreuses interfaces en utilisant de petites fenêtres (les buffers), c'est qu'en réalité, il s'agit d'un modèle basé sur une approche unifiée, et que les modules ne sont là que pour l'aspect pratique. En effet, il est tout à fait compréhensible de regrouper les « briques » du processus de traitement qui portent sur les mêmes fonctions, ce qui expliquerait que certaines fonctions cognitives soient facilement associables à des zones cérébrales. En revanche, le fait que les composants du processus de traitement soient proches ou regroupés n'est pas un indice d'une hétérogénéité dans les principes qui sous-tendent les fonctions.

Une autre source de doute (Kosslyn, 2001; Uttal, 2001) quant à la modularité des systèmes cognitifs vient des limites dans la localisation des fonctions cognitives dans la recherche sur l'imagerie cérébrale. Des questions restent en suspens à la fois concernant les statistiques qui permettent d'identifier les régions intéressantes et sur la consistance des données entre différentes études. Il est alors envisageable de considérer que certaines fonctions cognitives ne soient pas localisées dans des régions spécifiques du cerveau, et par conséquent remettre en question l'existence de modules fonctionnels.

D'un autre côté, cet argument n'est pas réellement recevable, puisque la localisation des fonctions cérébrales n'est pas un prérequis à l'existence de modules, qui sont davantage des unités « conceptuelles » de traitement cognitif, et peuvent donc être distribuées dans l'espace. Cet argument annule, en contrepartie, l'utilisation de l'argument « imagerie cérébrale » en faveur d'une approche modulaire.

Architectures Linguistiques Modulaires

Une écrasante majorité des architectures pour le traitement des langues suit une approche modulaire. Cette situation provient principalement de deux raisons :

1. L'approche compositionnelle à la Frege est largement répandue dans la recherche linguistique informatique
2. La mise en œuvre d'une architecture d'interprétation de la langue nécessite le plus souvent d'intégrer des composants issus de théories diverses, et suivant donc les principes différents. La seule approche viable pour l'intégration d'éléments hétérogènes est de travailler sur les frontières, en créant des « modules interfaces » qui ont eux-mêmes leurs propres principes pour permettre aux autres modules de fonctionner de concert.

Parmi les nombreuses architectures de traitement automatique des langues, on distingue aussi deux grandes familles.

– La première regroupe les architectures séquentielles, qui adoptent non seulement l'approche modulaire, mais considèrent en plus que le traitement de la langue se fait en chaîne (phonétique → phonologique → morphologique → syntaxique → sémantique → pragmatique). Cette approche, très utilisée dans les grammaires formelles (notamment celles des langages informatiques) est quasiment abandonnée aujourd'hui, car elle supporte très mal les aspects ambigus et implicites de la langue humaine. L'architecture du projet GATE (Cunningham et al., 1995) suit néanmoins cette approche, mais s'intéresse principalement à fournir une plate-forme permettant de concevoir rapidement une chaîne de traitement pour le texte.

– La seconde approche est dite distribuée, et considère que chaque module peut éventuellement communiquer avec tous les autres modules, sans qu'un ordre particulier ne soit imposé. Cette distribution n'équivaut certes pas au parallélisme qui permettrait d'exécuter simultanément les processus décrits par les modules ; il est néanmoins plus logique de chercher à faire correspondre les architectures distribuées avec des modèles d'exécution parallèles.

Parmi les approches distribuées, on trouve encore différentes architectures. Des architectures en étoile comme Galaxy COMMUNICATOR (Seneff et al., 1998) dont tous les composants communiquent via un *hub* selon un formalisme de communication bien établi, ont l'avantage d'une grande simplicité architecturale. Ces architectures posent des problèmes importants pour l'intégration de nouveaux composants, puisque ceux-ci doivent, pour remplacer un module précédent, s'adapter à son format de communication (par exemple pour proposer un meilleur module de décodage vocal). Ajouter un composant totalement nouveau qui ne se contente pas d'utiliser tous les composants existants, mais devrait s'intégrer à eux, nécessiterait de modifier tous les formats de communication et donc tous les composants. Bien sûr, cette architecture a été avant tout conçue pour permettre une certaine flexibilité à l'intérieur d'un cadre théorique plutôt strict, et il faut prendre cette critique plutôt comme une critique sur l'objectif de l'architecture que sur ses limitations à l'intérieur de cet objectif.

Des approches moins centralisées, comme CARAMEL-1 (Sabah, 1990) ou TRIPS (Ferguson and Allen, 1998) fonctionnent elles aussi sur le modèle de modules hétérogènes, mais les liaisons entre les différents modules sont mieux définies, et permettent d'éviter le goulet d'étranglement du *hub*. La critique quant à l'évolutivité des modules reste cependant valable, puisque toute innovation dans l'interface d'un module se répercutera nécessairement sur les modules avec lesquels il a des accointances. Dans une optique un peu différente, le projet TALISMAN (Stefanini and Demazeau, 1995) permet l'interaction d'agents disposant de compétences hétérogènes pour mener à bien une analyse linguistique, sa portée reste cependant limitée à une analyse restreinte.

2.2.2 Architectures Unifiées

Nous présentons ici deux familles d'architectures unifiées, d'une part les modèles opérationnels de la cognition conçus autour d'un principe unique, et d'autre part les systèmes de traitement de la langue qui s'appuient sur des démarches unifiées pour traiter certains aspects de la langue. Bien entendu, ces deux familles ont de nombreux points commun, la langue étant considérée comme un aspect de la cognition. Cependant, si les modèles cognitifs unifiés ont déjà été utilisés comme support pour le traitement de la langue, ces tentatives n'ont pas été poussées très loin, restant souvent cantonnées aux aspects syntaxiques du traitement. Inversement, les modèles unifiés de traitement de la langue n'ont à notre connaissance jamais été l'objet de tentatives pour les étendre à la simulation des fonctions cognitives.

Modèles Cognitifs Unifiés

Systèmes de production La discipline psychologique a réellement ouvert la voie à une recherche en profondeur sur l'architecture du système cognitif avec les travaux portant sur les systèmes de production. Les systèmes de production ont été introduits par Newell and Simon (1972) et Newell (1973) dans une étude sur les stratégies adoptées par les humains

pour la résolution de problèmes. Ils ont été utilisés pour modéliser le comportement des sujets pendant la résolution d'un certain problème, et pour vérifier l'adéquation entre les hypothèses formulées par les psychologues sur le fonctionnement interne de la cognition et les résultats des expériences.

Grossièrement, on peut voir les systèmes de production comme des langages de programmation, le langage étant extrêmement réduit puisque limité à des règles Perception/Action. Vu comme des outils de programmation, les systèmes de production sont assez peu performants, car ils ne correspondent pas du tout à l'intuition des programmeurs habitués aux langages procéduraux, qui réfléchissent plutôt en termes de registres mémoire et d'opérations. Cependant, ils se rapprochent de certains systèmes récents fonctionnant sur le mode dit « *dataflow*[3] », dans lequel c'est le flux d'information qui est au coeur de la réflexion, et qui est construit autour d'unités de transformation de l'information. Le mode « *dataflow* » permet notamment de décrire un algorithme pour qu'il puisse s'exécuter, sans problème, de manière distribuée.

En revanche, un système fondé sur de telles règles de réaction corrobore de nombreuses hypothèses sur le fonctionnement du système cognitif. Le système cognitif est en effet réalisé par un grand nombre de neurones, qui sont grossièrement des unités recevant des informations de plusieurs sources et produisant une information unique en fonction des informations reçues.

Ce que l'on appelle production (ou règle de production) est une règle de type SI-ALORS qui permet de modéliser un comportement local. La partie SI représente la condition nécessaire et suffisante (la perception) à l'exécution de l'action contenue dans la partie ALORS. Par exemple, pour faire un modèle partiel de comportement d'un automobiliste face à un feu rouge, on pourrait établir les productions du tableau 2.1.

Condition (SI)	Action (ALORS)
(Feu rouge ou Panneau STOP) et Roulant	Arrêter
Feu vert et Arrêté	Démarrer
Panneau STOP et Arrêté	Vérifier voie
Panneau STOP et Arrêté et Voie libre	Démarrer

Tab. 2.1 – Exemple partiel de règles de production pour modéliser le comportement d'un automobiliste.

L'intérêt de ce type de règles est grand pour représenter les connaissances nécessaires à traiter un problème donné, et c'est le même genre de règles qui est utilisé dans les systèmes experts qui permettent d'intégrer dans un programme informatique les compétences humaines sur un sujet formellement identifié et représenté. Cependant, leur intérêt ne s'arrête pas là, et l'idée d'utiliser les productions comme élément minimal d'un système mimant la cognition humaine est apparu presque simultanément. En effet, dans certaine de leurs formes, ces règles permettent de modéliser correctement le schéma comportemental perception/action (ou stimulus/réponse). Les productions sont les entités minimales de représentation des connaissances procédurales constitutives des systèmes de productions. Elles peuvent aussi facilement être rapprochées des modèles formels utilisés dans les théories du fonctionnement des systèmes neuronaux naturels, même si elles sont définitivement leur pendant symbolique.

[3]Qu'on peut traduire par « flux d'informations »

Le plaidoyer de Newell (1990) pour la recherche d'un modèle architectural unifié de la cognition a donné le signal de départ à une grande variété de propositions, mises en œuvre informatiquement de manière effective, et permettant de comparer les modèles théoriques aux résultats expérimentaux. Cette approche s'est développée pour sortir, en psychologie, de la tendance naturelle qu'a toute discipline à s'orienter vers la spécialisation des modèles en segmentant le problème global en sous-problèmes plus facilement abordables. Face au développement de différentes théories disparates cherchant à modéliser les fonctions cognitives supposées isolées, l'objectif d'une théorie unifiée est venu à Newell en raison de la maturité grandissante des différents modèles, et aussi parce qu'il pensait que son modèle de système de production (Newell, 1973) permettait de mettre en œuvre les différentes théories des fonctions cognitives. Anderson (1993) cite entre autres les recherches sur les aspects du traitement visuel, de la différence entre connaissances déclaratives et procédurales (Squire, 1987), et sur la langue bien sûr (Fodor, 1987), pour illustrer la spécialisation en psychologie.

L'intérêt de réunir les différents modèles en une seule théorie est évident pour toute science, où la spécialisation reste un recourt temporaire face à une complexité trop grande pour être d'emblée modélisée par un modèle général. Les récentes avancées en psychologie, grâce aux différentes techniques d'observations du système cognitif qui ont pu se développer récemment ont donc offert aux sciences cognitives la possibilité de se réconcilier avec l'idée de proposer des théories unifiées de toute la cognition.

Il est intéressant de constater que même dans les théories générales de la cognition inspirées des systèmes de production, deux approches semblent s'opposer. D'un côté on trouve les modèles radicaux qui tentent de modéliser la cognition par l'agencement de différents modules pouvant communiquer entre eux afin de donner naissance au phénomène cognitif proprement dit. De l'autre côté, les modèles qui tentent de proposer à un niveau plus fin les composants minimums de la cognition dans lesquels celle-ci pourrait toute entière être réalisée. Dans la première famille, celle des modèles théoriques modulaires intégrés, on trouve ACT-R (Anderson, 1993) et EPIC (Meyer and Kieras, 1997). Dans la seconde famille, celle des modèles sans module, on trouve SOAR (Laird et al., 1991) et CAPS (Just et al., 1999). En réalité, la situation est plus complexe que cela, car si ces systèmes sont effectivement construits suivant différents niveaux de cloisonnement, ils ont tous en commun le fait d'être principalement des systèmes de production (hybrides ou non, c'est-à-dire purement symboliques ou mêlant symboles et caractères continus), et mélangeant mémoires procédurales représentées par les productions et mémoires déclaratives dans ce qui est appelé soit tampons (buffers) soit plus simplement mémoire(s) de travail (WM pour working memory).

Les différents systèmes implémentés présentent des caractéristiques différentes, selon les contraintes qu'ils imposent sur l'exécution des règles de production (en série avec une règle à la fois, en semi-série avec plusieurs exécutions mais un seul chemin cognitif majeur, ou totalement parallèles avec plusieurs chemins cognitifs simultanément).

Globalement, on retiendra surtout que ces différents systèmes convergent fortement dans leurs conclusions sur les paramètres de la cognition, et qu'ils ont récolté de nombreux succès en reproduisant fidèlement des comportements humains sur certaines tâches. En tant qu'approche psycho-informatique, les systèmes de production sont donc clairement à prendre en compte pour un modèle d'interprétation tel que nous le souhaitons.

Systèmes connexionnistes Alors que les systèmes de production restent principalement des systèmes symboliques, c'est-à-dire fonctionnant grâce à des informations discrètes, les systèmes connexionnistes sont axés sur une vision continue du fonctionnement de la cognition. L'unité de base dans ces systèmes est un neurone formel, c'est-à-dire une boîte ayant des entrées et une sortie. Chacune des entrées est pondérée par une certaine valeur et est connectée soit à une perception directe, soit à la sortie d'un autre neurone. Dans tous les cas, les entrées ou les sorties sont des valeurs continues normalisées d'une certaine manière. En construisant de grands réseaux de tels neurones et en précisant des contraintes particulières sur les fonctions de calcul de la sortie des neurones et les règles de mise à jour des poids sur chaque entrée, il est possible de produire des systèmes capables d'apprendre à catégoriser correctement certaines perceptions (par exemple trouver quelle lettre de l'alphabet est représentée par une matrice de pixels).

Cette capacité n'est cependant pas suffisante pour modéliser l'ensemble de la cognition. De plus, les systèmes connexionnistes purs, s'ils reprennent bien l'idée de neurones, n'ont toujours pas réussi à prendre en compte l'aspect dynamique du système neuronal humain, dans lequel les temps de transmission des informations ainsi que les fréquences des pulsations jouent d'après certains un rôle primordial. Enfin, le comportement des systèmes connexionnistes est difficile à maîtriser car ils fonctionnent comme des boîtes noires. En effet, la plupart des réseaux de neurones sont composés de plusieurs couches de neurones qui ne sont connectés ni à une entrée directe, ni à une sortie. On appelle ces couches des couches cachées. Ces couches jouent un rôle primordial dans l'apprentissage, mais les poids affectés à leurs connexions n'ont aucune signification claire. Par voie de conséquence, il est très difficile de modifier le comportement d'un réseau de neurone dans un sens précis. Il est donc impossible d'utiliser les systèmes connexionnistes pour modéliser un fonctionnement précis, et donc, même si ces modèles sont très proches du système biologique qu'ils tentent de modéliser, leur intérêt est limité pour représenter un système cognitif entier, car il sera très difficile de le faire fonctionner pour atteindre un but précis.

Pour pallier ces différents inconvénients, plusieurs travaux (notamment ceux de Shastri (1999)) ont proposé des systèmes hybrides entre connexionnisme pur et fonctionnement symbolique, afin de permettre une certaine programmabilité tout en conservant la souplesse de l'approche continue. Les résultats obtenus semblent extrêmement encourageants, et montrent qu'il est possible de construire des systèmes qui soient en partie contraints par des connaissances symboliques *a priori* tout en offrant la puissance des systèmes connexionnistes.

Modèles Linguistiques Unifiés

Le Word Expert Parsing Le Word Expert Parsing (Small, 1980; Adriaens and Devos, 1994) est une théorie de la compréhension du langage naturel fondée sur l'hypothèse que les objets du langage (les mots et leur partie) sont individuellement autant porteurs d'information sur leur propre analyse que d'information sémantique. Un système de compréhension du langage naturel devrait donc, selon cette idée, diriger son analyse en fonction de la connaissance (pragmatique, sémantique, lexicale, morphologique) apportée par chaque mot. Plus généralement, cette approche est inspirée par la sémantique procédurale (Woods, 1981; Johnson-Laird, 1977), qui postule que le « sens » des mots **est** leur rôle dans le processus d'interprétation (et non pas comme il est parfois interprété, que le sens des mots est un morceau de code

exécutable qui est combiné par une vision compositionnelle à la Frege). Autrement dit, la sémantique procédurale explique la compositionnalité comme le produit de l'interaction de processus

Pour utiliser cette connaissance dans le traitement effectif de phrases isolées, Small propose de coordonner l'analyse par une interaction explicite entre Word Experts. Cette interaction permet à l'ensemble des experts activés lorsqu'une phrase est traitée par le système de communiquer entre eux pour se mettre d'accord sur le sens qu'ils doivent prendre dans le contexte global de la phrase. Une source de connaissances interagissant dans ce modèle est donc, pour Small, représentée par un agrégat de questions et d'actions. Les experts peuvent ainsi construire de manière conditionnelle des structures conceptuelles représentant le sens qu'ils s'attribuent, agir sur l'état global du système, ou encore interroger des connaissances externes (relatives au discours ou à des données non représentées sous forme de Word Expert, par exemple).

Cette représentation décentralisée du processus d'analyse conduit le système à se reposer sur les actes de communication entre experts pour l'échange d'informations et les choix à faire concernant la signification globale vers laquelle le système doit se diriger. Pour contrôler cette population d'experts, chacun d'entre eux doit être décrit de manière à se « mettre en pause » lorsqu'il a besoin d'information provenant d'un autre expert, et à reprendre son activité lorsque cette information arrive.

Cette théorie n'a cependant rencontré qu'un succès d'estime, notamment parce qu'il se révèle très vite impossible de décrire une « grammaire » de Word Experts dès l'instant que l'on veut couvrir un pan raisonnable du lexique. En effet, le gros inconvénient de cette approche est que chaque Word Expert doit prévoir une quantité de situations qui dépend indirectement du nombre d'éléments lexicaux qui doivent être gérés par la grammaire.

Hactar Le modèle de traitement HACTAR (Lebarbé, 2004) est fondé sur une approche multi-agent de l'analyse linguistique. La théorie linguistique réalisée dans HACTAR consiste en une analyse par fragments (*chunck parsing*, (Abney, 1991; Vergne, 1999)). La mise en relation entre analyse et agents est effectuée par une instanciation des différentes unités (mots, chunks, propositions ou phrases) sous la forme d'un agent. Comme tous les agents ont la même structure, l'analyse est conduite par un principe unique, basé sur l'observation des environnements internes aux agents et de l'environnement externe, suivie d'une action sur cet environnement. La « brique » de base de l'analyse est donc bien unique, et le modèle HACTAR est de type unifié.

La principale limitation de ce système est qu'il n'existe qu'un unique environnement externe, limitant les possibilités du système à une analyse textuelle, et ne permettant pas d'intégrer des images issues de l'interprétation du texte. D'autre part, le modèle restant à un niveau syntaxique, il est relativement facile d'épuiser les informations que les agents devront être capables d'extraire du texte, mais l'adapter pour prendre en compte des dépendances sémantiques demanderait probablement un travail très important pour chaque agent.

ILLICO Le système ILLICO (Pasero and Sabatier, 1995, 1998) suit lui aussi une approche unifiée, dans le sens où toutes ses composantes sont décrites sous une forme propositionnelle, inspirée de la grammaire de métamorphose Colmerauer (1978). Cela illustre à la fois :

– le fait que le choix d'une approche unifiée ne préjuge en rien de considérations cognitives dans la théorie d'interprétation employée, puisque ILLICO se fonde sur une conception vériconditionnelle du sens,
– le fait qu'une approche unifiée n'empêche pas un découpage du système en modules, puisque ILLICO est structuré en niveaux de bonne formation (syntaxique, sémantique, conceptuel, contextuel), ce qui permet de structurer les règles en fonction de leur rôle dans tel ou tel niveau.

ILLICO est conçu comme un ensemble de règles associant des catégories (syntaxiques pour le niveau syntaxique, par exemple) à d'autres catégories, ou bien à des *contraintes*. C'est un système fondé sur une grammaire d'unification, et fonctionnant à l'aide d'un moteur d'inférence à la PROLOG. Ces règles ont l'avantage certain de pouvoir décrire aussi bien les niveaux syntaxique que conceptuels, ou bien une théorie linguistique, qui dans son cas est la Grammaire Noyau du Français, ou encore des connaissances générales comme les relations ontologiques.

Il est principalement utilisé dans des applications d'interrogation de bases de données, pour lesquelles une description propositionnelle est généralement parfaitement adéquate (par exemple pour obtenir des information géopolitiques ou économiques sur des pays), mais il sert aussi à des systèmes d'aide à l'apprentissage de seconde langue, ou bien comme système de génération.

Comme nous le verrons par la suite, excepté son orientation vériconditionnelle, les choix effectués dans ILLICO sont assez proches des nôtres, notamment du fait que les règles utilisées se rapprochent sensiblement de la notion de s-constructions que nous allons introduire ensuite.

2.2.3 Avantages et inconvénients d'une approche unifiée

Une approche unifiée semble intuitivement être mieux adaptée qu'une approche modulaire pour représenter le fonctionnement des processus cognitifs. Cette approche n'est cependant pas exempte de défauts. Voyons d'abord les points qui posent problème dans l'approche unifiée :

Obligation de se conformer à l'approche, et donc de représenter tous les processus cognitifs avec les mêmes outils. Les approches modulaires permettent d'adapter la mise en œuvre aux principes théoriques dont on pense qu'ils dirigent une certaine fonction, alors que ce n'est pas le cas dans une approche unifiée. Or même si l'on dispose d'un formalisme déclaratif permettant de modéliser toutes les fonctions souhaitables, ce n'est pas forcément dans tous les cas le formalisme le plus abordable pour les concepteurs. Prenons le cas de la résolution extensionnelle des expressions référentielles, qui est étudiée à la fin de cette thèse. Pour mettre en place le processus que nous proposons, il est nécessaire de passer par une approche assez contre-intuitive pour un programmeur. Il aurait en effet probablement préféré une heuristique avec un fonctionnement séquentiel, plutôt qu'une modélisation par des règles conditions/action comme l'impose le modèle d'interprétation constructionnelle présenté plus loin.

Efficacité computationnelle. De manière assez similaire, imposer un modèle déclaratif pour représenter l'intégralité des processus cognitifs implique que de nombreuses optimisations ne pourront pas être aisément implémentées, alors même qu'une approche modulaire permettrait de traiter de la manière la plus adéquate certains problèmes.

Ces défauts sont cependant plus d'ordre pratique que théorique et ils sont largement contrebalancés par les avantages d'une approche unifiée :

Modularisabilité. Avec une approche unifiée, il reste tout à fait possible de regrouper en paquets les éléments qui interviennent dans une même fonction, permettant de segmenter l'ensemble du modèle cognitif en sous-parties qui peuvent alors être réparties entre les développeurs de la base connaissances. Cet aspect de la modularité, nécessaire du point de vue du génie logiciel, est donc gardé intact avec une approche unifiée.

Phénomènes transversaux. Des phénomènes qui ne sont pas spécifiques à certaines fonctions, comme par exemple l'apprentissage, l'anticipation ou la capacité à conduire plusieurs analyses en parallèle, peuvent être élaborés sans se soucier des particularités d'implémentation de chaque fonction cognitive. Ce qui permet de simplifier la vérification de théories s'appliquant à de tels phénomènes, et de proposer des modèles plus simples.

Interopérabilité. Il est plus facile pour un développeur travaillant sur une telle représentation de chercher à atteindre des informations traitées par une autre fonction que celle dont il s'occupe principalement, puisqu'il connaît déjà le fonctionnement du système. Les différents « paquets » peuvent donc être mis en collaboration de manière plus efficace et ce, plus rapidement.

Souplesse. Dans une approche unifiée, il est beaucoup plus facile de tester simultanément deux approches concurrentes pour la mise en œuvre d'une fonction cognitive (ou sous-fonction) à des fins de comparaison. Une telle approche donne en effet naturellement un découpage très fin de la représentation, or un tel découpage permet non seulement d'isoler des fonctions aussi petites que l'on souhaite, mais aussi d'intégrer des « chemins parallèles » qui vont donner une interprétation alternative dont le résultat sera comparable avec les « chemins » normaux de l'interprétation.

Optimisations globales. Alors qu'il semble *a priori* plus difficile de mettre en œuvre des optimisations particulières avec un mode de programmation imposé, on sait qu'en fait des optimisations globales peuvent être plus facilement intégrées. C'est un aspect bien connu des langages fonctionnels ou par contraintes, qui s'avèrent capables de produire des solutions bien plus optimales que ce que pourrait faire un programmeur « procédural », et en demandant finalement beaucoup moins d'efforts au programmeur.

2.3 Topologies et Espaces pour la langue et la cognition

Nous mettons en exergue, dans cette section, le fait que de nombreuses théories à la fois en linguistique, en intelligence artificielle, et en sciences cognitives (Eschenbach et al., 1994) font appel à des ensembles structurés, dotés ou non d'une mesure de distance entre les éléments qu'ils contiennent.

Notre objectif est de montrer que, pour un modèle d'interprétation comme celui que nous voulons définir, il serait intéressant d'intégrer dans le coeur du principe la notion d'**espace structuré** par **une ou plusieurs dimensions** et doté d'une **topologie**.

2.3.1 Intervalles et raisonnement

Le raisonnement sur les relations spatiales et temporelles est un problème d'intelligence artificielle particulièrement saillant pour les systèmes de dialogue de type agents assistants d'interface. L'aspect dynamique de l'interaction et l'implication de la dimension visuelle font qu'il est difficilement pensable de concevoir un tel système de dialogue sans prendre en compte les aspects temporels ou spatiaux.

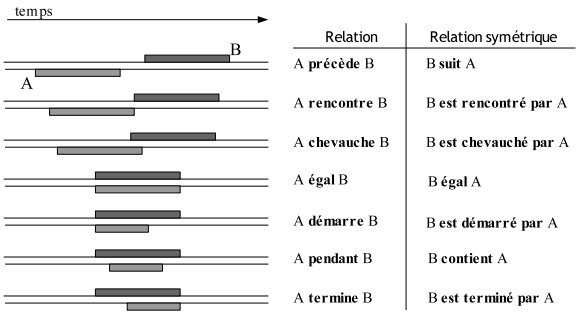


FIG. 2.1 – Relations temporelles sur des intervalles définis par deux instants (Allen 1983)

Ainsi, le modèle de raisonnement temporel proposé par Allen (1983) s'appuie sur 13 relations (figure 2.1) portant sur des intervalles de temps définis par deux instants. Ce modèle n'est pas le seul permettant de traiter des intervalles temporels d'autres ont été proposés pour prendre en compte des relations plus complexes : dans l'optique de généraliser le modèle de raisonnement de Allen à des représentations temporelles plus évoluées, Ligozat (1998) décrit un modèle de raisonnement basé sur des intervalles généralisés, qui consistent en des ensembles ordonnés d'instants, permettant de représenter et de raisonner sur des événements complexes qui ne se décrivent pas simplement avec des paires d'instants (par exemple une hospitalisation pour intervention chirurgicale, qui sera représentée par l'instant d'entrée à l'hôpital, l'instant de l'intervention, et l'instant de la sortie de l'hôpital).

En ce qui concerne le raisonnement spatial, de nombreux modèles existent (Cohn and Hazarika, 2001), soit purement qualitatifs, soit prenant en compte des informations quantitatives. Pour les modèles directement liés aux problèmes de traitement de la langue, on trouve les cadres de Schang (1997), qui permettent de représenter des structures de boîtes imbriquées, notamment pour traiter les références vis-à-vis d'une interface comportant des fenêtres. Pour les questions de référence aux objet, il existe aussi des modèles prenant en compte les aspects quantitatifs dans le plan (Schirra, 1993) ou dans l'espace tridimensionnel (Gapp, 1994).

Ces différents modèles de raisonnement décrivent des sémantiques des relations entre des intervalles plus ou moins complexes ou entre des zones spatiales de différente nature. Il est clair qu'aucun modèle logique ne permet de traiter tous les types de raisonnement, et donc qu'il est nécessaire d'intégrer d'une manière ou d'une autre ces différents modèles hétérogènes pour construire un système autorisant le raisonnement dans un maximum de situations.

2.3.2 Espaces Mentaux

Les **espaces mentaux** (Fauconnier, 1984) sont définis comme des « *ensembles structurés et modifiables, construits dans chaque discours en accord avec les indications fournies par les expressions linguistiques* ». La théorie des espaces mentaux est avant tout une théorie de la référence, qui s'appuie d'ailleurs en grande partie sur les travaux sur les *fonctions référentielles* (Nunberg, 1979). L'idée maîtresse de la théorie des espaces mentaux est que les entités qui sont cibles de références dans le discours sont distribuées dans des espaces imbriqués. Le discours lui-même impose la construction de ces espaces mentaux ainsi que la création de connexions entre les entités, qui associent celles-ci par des *fonctions référentielles*. Par exemple, dans l'énoncé « *Dans ce film, Clint Eastwood est un traître* » (Fauconnier, 1984, p. 34), deux points de vue sont introduits (voir figure 2.2), et chacun peut être la cible d'une anaphore. Si cet énoncé est suivi de « *Il vend des secrets d'état à la mafia* », c'est le personnage du film qui est l'antécédent, tandis qu'avec « *Il a reçu un oscar pour ce rôle* », il s'agit l'acteur Clint Eastwood. La notion d'espace mental permet d'expliciter ce phénomène en localisant les points de vue possibles sur une même entité dans des ensembles distincts, tout en spécifiant les liens existants entre ces points de vue.

Pour Fauconnier, « *Communiquer, c'est parvenir à partir d'indices linguistiques et pragmatiques semblables à opérer les mêmes constructions d'espaces (ou tout au moins des constructions voisines)* ». D'après lui, l'interprétation d'un énoncé passe donc prioritairement par la création d'espaces mentaux. Il évoque clairement le fait que les problèmes de polysémie et de métaphore passe de même par l'utilisation de ces connecteurs qui permettent de relier des entités de deux espaces mentaux distincts : « [...] *la manière classique d'essayer de rendre compte de la multiplicité de sens d'une forme linguistique donnée consiste à lui associer systématiquement, à un niveau plus abstrait, un ensemble de représentations (formes logiques ou structures sémantiques, etc.) ; le format des espaces fait apparaître une tout autre organisation : les structures de phrase sont limitées à leurs propriétés grammaticales ; ces structures, plus simples, donnent des indications sur la construction d'espaces correspondants dans tel ou tel contexte.* ».

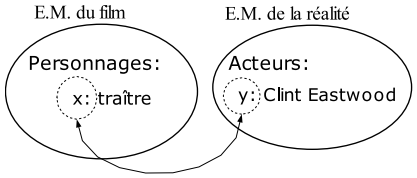


FIG. 2.2 – Exemple d'utilisation des espaces mentaux. « Dans ce film, Clint Eastwood est un traître. »

On peut adresser deux critiques majeures à la théorie des espaces mentaux telle qu'elle est décrite par Fauconnier. Premièrement, elle est délicate à mettre en œuvre, notamment parce qu'elle ne s'appuie pas sur des théories linguistiques bien formalisées. La théorie décrit ainsi différents constructeurs d'espaces (*space-builders*) qui sont des signes linguistiques particuliers introduisant automatiquement de nouveaux espaces mentaux. Ces signes semblent pouvoir être indifféremment d'origine grammaticale (par exemple certains temps) que sémantique (certains verbes) voire pragmatiques. Deuxièmement, rien dans la théorie ne permet de décider en quoi certaines fonctions référantes, qui relient deux entités, doivent être préférées à

d'autres lorsque plusieurs sont possibles. Ainsi Moeshler and Reboul (1994) posent un problème similaire à celui-ci : dans « Georges Sand est sur l'étagère du haut », la fonction référante utilisée est celle qui associe **un auteur à ses œuvres** ; en revanche, dans « Je ne peux pas partir, le voisin est garé en double-file. », il s'agit de la fonction qui associe **le propriétaire à l'objet qu'il possède** (ici, le voisin et sa voiture, puisque c'est la voiture qui est garée). Si la deuxième fonction est équivalente à la première, pourquoi alors ne peut-on par choisir la fonction *propriétaire-objet possédé* pour « George Sand est sur l'étagère du haut », ce qui amènerait à en conclure que les livres que possède George Sand sont sur l'étagère du haut. La théorie des espaces mentaux ne donne aucun indice pour choisir entre deux fonctions lorsque les deux sont possibles.

2.3.3 Domaines de Référence

La notion de *domaine de référence* (parfois désigné par le terme *domaine d'interprétation* ou de *sélection*) a été introduite par Corblin (1987) pour rendre compte des phénomènes de reprise référentielle dans le cas des expressions référentielles démonstratives et pour le calcul référentiel des expressions définies modifiées (c'est-à-dire un groupe nominal défini comportant une spécification supplémentaire sur le nom, exemple : « le livre rouge »). Pour les expressions définies modifiées, Corblin analyse leur interprétation de la façon suivante :

> « Peut-être alors doit-on poser que pour les définis modifiés, la reprise est considérablement atypique en raison du calcul référentiel propre au défini. Celui-ci désigne un individu de la classe N, recruté comme seul des N à posséder la propriété mentionnée par le modifieur. Dans *le livre rouge*, *rouge* est **le critère utilisé pour sélectionner un livre particulier**. L'interprétation renvoie donc à un domaine de livres, dont un des individus est sélectionné parce qu'il est rouge. C'est pourquoi l'interprétation renverra naturellement à un groupe de livres où le critère trouve à s'appliquer, **plutôt qu'à une mention antérieure**, fût-elle mention d'un livre. Comme on l'a indiqué plus haut, le calcul propre au défini consiste à utiliser le contenu préfixé pour recruter un particulier, ce qui suppose la détermination de deux termes : **un domaine de sélection, un critère de sélection**. En tout état de cause, le critère de sélection est unique, et appartient au contenu du groupe nominal. » (Corblin, 1987, p.194), nous avons mis certains passages en gras.

Il introduit deux notions fondamentales : le **domaine de sélection** (qui deviendra domaine de référence) qui est un ensemble de référents construit antérieurement, et un **critère de sélection** qui est l'information permettant de distinguer le référent ciblé par l'expression de l'ensemble des référents possibles, et dont on devine qu'il doit être opéré par une fonction disant pour tout élément du domaine de sélection si oui ou non l'élément répond au critère.

Prise isolément de toute mention antérieure, l'expression référentielle démonstrative ne permet pas de trouver le référent (elle n'est pas saturée) par rapport à l'ensemble des éléments accessibles, et il faut donc s'appuyer sur un sous-ensemble qui a été évoqué précédemment pour appliquer le critère de différenciation. Pour pouvoir expliquer la fonction référentielle du démonstratif, l'hypothèse de Corblin est donc qu'il faut nécessairement introduire une

représentation de la mention antérieure liée à la désignation démonstrative (cette mention pouvant être d'origine linguistique ou bien être dérivée du contexte). Cette représentation doit être un ensemble de référents dont un des éléments est mis en exergue par le critère de différenciation contenu dans l'expression référentielle démonstrative.

Les domaines de références ont été repris dans la théorie des représentations mentales (Reboul et al., 1997), afin de rendre compte du fait qu'une représentation mentale ne peut être introduite qu'à condition de disposer d'un critère de différenciation pour isoler un individu particulier (le critère de différenciation permettant de distinguer l'individu de tous les autres). Comme ce critère de différenciation seul ne permet pas de distinguer une entité précise, il faut introduire un « fond » sur lequel il pourra être appliqué. Reboul et al. utilisent la notion de *domaine de référence*, qui permet de représenter le « fond » contenant les entités potentiellement accessibles et sur lequel le critère de différenciation serait appliqué pour construire la représentation mentale correspondant à une expression référentielle.

Le concept de domaine de référence a été utilisé avec succès dans au moins deux optiques différentes. D'une part pour rendre compte de la reprise référentielle (Salmon-Alt, 2001), et d'autre part pour la question de la référence dans le cadre du dialogue multimodal (Landragin, 2003), notamment en prenant en compte des aspects de pertinence de la théorie de Sperber et Wilson (Sperber and Wilson, 1995).

Dans (Salmon-Alt, 2001) les domaines de références sont définis comme des « *ensembles contextuels locaux structurés de façon à prédire la distribution des différents marqueurs référentiels* ». Ce modèle permet notamment de traiter des phénomènes concernant la référence anaphorique. Le fait de représenter les différentes étapes de sélection par des ensembles structurés permet de rendre compte de certains phénomènes de préférence dans la reprise nominale. Les domaines de référence permettent d'organiser des entités en fonction de leurs relations entre elles selon un critère de différenciation (par exemple bleus/non-bleus), ce qui de fait le partitionne entre entités accordées au prédicat, et entités non accordées.

Landragin (2003) utilise quant à lui la notion de pertinence de Sperber and Wilson (1995) pour ordonner les différentes entités d'un domaine de référence. De cette manière, il montre que le critère de différenciation des domaines de référence peut non seulement être un critère prédicatif comme dans les travaux de Salmon-Alt et de Reboul *et al.*, mais plusieurs critères à caractère continu peuvent ainsi être combinés afin de définir les candidats les plus « saillants » parmi un ensemble d'entités. L'utilisation de domaines de référence permettant de segmenter la résolution en différentes étapes, et ainsi de pouvoir s'appuyer sur un domaine intermédiaire comme point de départ pour une future résolution.

Conclusion

La réalisation d'un système d'interprétation pour les agents assistants d'interface tel que nous l'avons défini, doit donc selon nous reposer sur trois critères :

- Une théorie linguistique permettant la prise en compte de schémas cognitifs. La grammaire ECG nous a semblé le meilleur point de départ pour réaliser cet objectif.
- Une architecture unifiée, permettant à la fois une bonne cohésion entre les différentes composantes de l'interprétation, une plus grande facilité de développement et supportant la possibilité de décrire des phénomènes transversaux aux différentes composantes.

– Un modèle permettant de prendre en compte les différentes structures disposant éventuellement d'une topologie, et cela d'une manière intégrée à l'architecture unifiée.

Il reste un aspect important de l'interprétation que nous n'avons pas abordé, même s'il est largement couvert par la sémantique des *frames* de Fillmore utilisée dans la grammaire ECG. Il s'agit du problème de la représentation des informations ainsi que d'un problème corollaire concernant les relations ontologiques entre les représentations. Nous avons choisi de dédier le chapitre suivant à cette question particulièrement importante de notre point de vue.

Chapitre 3

Considérations théoriques sur le rôle des représentations dans l'interprétation

3.1 Représentations, Connaissances et Concepts

La représentation de la connaissance est un problème qui s'exprime à la fois dans la communauté de l'intelligence artificielle et dans la communauté linguistique :

– En intelligence artificielle, la représentation a donné lieu à de nombreuses propositions, telles que différentes familles de logiques, les *frames* de Minsky (1981), les graphes conceptuels de Sowa (1984), les systèmes de règles (Newell, 1973) pour la connaissance procédurale, ou bien selon une approche non symbolique pure comme les approches connexionnistes hybrides (Shastri, 2000). Dans ces travaux, on trouve une même conception de la représentation des concepts, avec l'idée que les concepts ont un lien direct avec les mots, et qu'ils « remplacent » les éléments de la réalité dans la pensée et permettent de structurer celle-ci. C'est la représentation telle qu'elle est conçue par Kant (1787), qui, pour ce qui nous intéresse, considère que les mots représentent des concepts, qui sont à leur tour des représentants des objets du monde. Par exemple, le mot « *table* » représente le concept TABLE qui lui-même « représente » tout objet appartenant à la catégorie des tables. Il faut noter que pour cette explication, la catégorie des tables, et donc l'ensemble des objets du monde qui sont des tables doit pouvoir être construite sans rencontrer de problème d'ambiguïté. Ce qui pose problème pour les nouvelles formes de tables (une table qui lévite ?) ou les formes dégradées de tables (une table coupée en deux, à qui on a enlevé un pied, etc.). Le problème de représentation est intrinsèquement lié au problème de la référence, puisqu'il conditionne la manière dont l'allocutaire d'un énoncé contenant une expression référentielle va reconnaître ou non l'objet désigné[1].
– En linguistique, différentes théories descriptives ont aussi un lien avec la représentation. La sémantique interprétative de Rastier (1987) cherche à représenter le sens des mots en leur attribuant des traits sémantiques. La sémantique du prototype de Kleiber (1990) s'intéresse

[1] Si l'allocutaire a une représentation du concept « table » correspondant à un plaque avec quatre barres fixées du même côté de la plaque et sur ses coins, alors lui dire « Met le vase sur la table » dans une situation où une table est soutenue par seulement trois piedsdevrait lui poser un sérieux problème.

davantage à attacher le sens des catégories aux éléments prototypiques auxquels ils se rattachent, c'est-à-dire aux éléments qui correspondent une norme commune qui est la base de la communication.

Nous nous intéresserons ici uniquement aux représentations issues des recherches en intelligence artificielle. Dans ces nombreux travaux, les recherches ont porté sur le *comment* représenter, mais assez peu sur le *pourquoi* et le *quoi.* La place de la représentation et la justification de son rôle dans les systèmes d'inférence et plus généralement dans l'intelligence artificielle a été assez relativement peu remise en cause.

En intelligence artificielle, une position relativement standard vis-à-vis de la représentation est exposée par (Davis, 1993) dans sa définition des cinq rôles de la représentation de la connaissance. Le premier rôle donné à la représentation de la connaissance est celui de substitut (*surrogate*), en directe application de la position Kantienne sur le rôle du concept comme composant **constitutif** des contenus représentationnels de la pensée (Boros, 1999). Dans cette hypothèse, c'est à partir de ce substitut que les humains et les programmes peuvent manipuler, c'est-à-dire utiliser à des fins intellectuelles, les choses et objets du monde auxquels il n'est pas possible d'accéder de manière directe. On trouve cette même idée dans (Charniak and McDermott, 1985) ou la représentation est considérée comme une « version » du monde, comme si le monde était en réalité une superposition de versions plus ou moins détaillées et que la représentation consistait à saisir une de ces versions :

> "*Representation is a stylized version of the world. Depending on how it is to be used, a representation can be quite simple, or quite complex*"

Pour introduire la notion de **point de vue**, qui est construite sur une autre hypothèse que la tradition philosophique de Kant, nous mettons en avant un autre aspect de ce que pourrait être une *représentation.* Pour cela, nous allons d'abord travailler sur une ambiguïté dans l'acception courante du terme « représentation », qui selon nous est à la source d'une certaine confusion dans les discours sur la représentation, mais aussi sur l'idée qu'on peut se faire du rôle d'une représentation dans le domaine de l'intelligence artificielle.

3.1.1 Représentation

Représentation n.f. Action de représenter, de présenter de nouveau : *exiger la représentation d'un passeport.* || Le fait de jouer une pièce de théâtre. || Image graphique, picturale, etc. de qqch. || Image mentale d'un objet donné. || *Litt.* (au pl.) Remontrances faites avec mesure. || Corps des représentants d'une nation : *la représentation nationale.* || Action de traiter les affaires pour le compte d'une maison de commerce. || *Dr.* Procédé juridique par lequel une personne, le représentant, agit au nom et pour le compte d'une autre, le représenté ; action de recueillir une succession à la place d'un ascendant prédécédé. •*En représentation* (Chorég.), v. INVITÉ, E. || *Système de représentation*, correspondance ponctuelle, biunivoque, entre les points de l'ellipsoïde terrestre, définis par leur longitude λ et leur latitude φ, et les points du plans, définis par leurs coordonnées cartésiennes $x, y : x = f(\lambda, \varphi),\ y = g(\lambda, \varphi)$.

Définition 2: *Représentation*, Petit Larousse, 1992

D'après les définitions 2 et 3, nous pouvons distinguer que le terme « représentation » peut être interprété de deux manières différentes, très proches sous certains aspects, mais dont la

Du latin *repraesentatio,* « action de mettre sous les yeux », d'où « image »

1. En psychologie, tout acte par lequel l'esprit se rend présent quelque chose (perception, idée, image).
2. En politique, fonction des personnes qui représentent le peuple (qui se prononcent en son nom) dans l'exercice du pouvoir (exemple : « une démocratie représentative »).

Pour **Shopenhauer**, le monde que nous avons sous les yeux n'est pas le vrai monde ; c'est une "représentation" qui n'existe que « dans son rapport avec un être percevant, qui est l'homme lui-même ».

Définition 3: *Représentation*

différence pose un réel problème pour la question du *sens* et de sa représentation. Dans une première interprétation, une représentation est une vue de quelque chose, une reproduction, un « autre » original. Une photographie, un croquis, une maquette miniature, un ensemble de données techniques ou encore une description textuelle d'un objet sont des *représentations* de celui-ci. Cette interprétation semble porter une notion qui pourrait s'exprimer par « une parmi d'autres », par exemple si l'on parle de la représentation de la Tour Eiffel par un certain dessinateur, on sous-entend que c'est une représentation particulière parmi d'autres qui ont été ou seront potentiellement faites de cet ouvrage. De la même manière, lorsque l'on assiste à une représentation de *L'école des femmes* de Molière, il est sous-entendu qu'il ne s'agit que d'une instance parmi d'autre de la pièce. Pour éliminer tout risque de confusion, nous parlerons de *point de vue* pour désigner cette interprétation.

La seconde interprétation trouve sa meilleure illustration dans le terme « représentation nationale », qui désigne l'assemblée des parlementaires. Dans cette acception, la représentation donne l'idée de « agir à la place de », « parler au nom de ». Elle porte l'idée qu'une représentation est *unique* et qu'elle *remplace* un autre élément unique pour une circonstance donnée. Ainsi, l'émissaire représente son roi le temps de sa mission, le parlement représente le peuple le temps de son mandat, le porteur d'une procuration représente un autre électeur pour l'acte du vote, le chargé de communication d'une entreprise représente cette entreprise pour les relations avec le public. Cette interprétation sera quant à elle désignée par *délégation.*

Les deux interprétations se rejoignent sur deux points, c'est la cause de l'utilisation d'un même terme pour les désigner.

– D'une part, il existe un lien fort entre l'original et sa représentation, avec une notion de fidélité de la représentation envers l'original. Dans le cas des *points de vue,* on parlera d'un portrait fidèle, d'une maquette qui respecte bien les proportions ; dans le cas de la *délégation*, on verra des critiques sur des élus qui ne transmettent pas la volonté des électeurs.
– D'autre part, dans les deux cas, la représentation est partielle : pour les *points de vue*, une peinture n'expose qu'un angle donné, une maquette omet des détails, un schéma ne possède pas la matérialité de l'original ; pour la *délégation*, l'émissaire ne possède pas les pouvoirs du roi ou encore, le parlement n'a pas délégation du pouvoir exécutif du peuple.

3.1.2 Concepts

La définition de *concept* est intimement liée aux questions de la langue, et pas seulement aux questions de la pensée. Dans la tradition platonicienne, le concept est attaché à une chose en

tant qu'il définit sa place dans la structure méthodiquement construite des relations des choses les unes par rapport aux autres. Descartes reprend en grande partie cette école de pensée en associant à chaque objet une essence naturelle précédant toute perception. Les choses et les concepts sont liés par nature, sans que l'oeil de l'homme n'y change rien. Kant critique cette position en postulant le rôle actif du sujet de l'expérience dans la qualification des choses. Il donne une base très constructive pour comprendre le *concept* à partir des définitions qui selon lui établissent une graduation des interprétations du terme *idée*.

> « Le terme générique est celui de représentation en général (*repraesentatio*), dont la représentation accompagnée de conscience (*perceptio*) est une espèce. Une perception qui se rapporte uniquement au sujet, comme modification de son état, est sensation (*sensatio*), une perception objective est connaissance (*cognitio*). Cette dernière est ou intuition ou concept (*intuitus vel conceptus*). L'intuition se rapporte immédiatement à l'objet et est singulière ; le concept s'y rapporte médiatement, au moyen d'un signe qui peut être commun à plusieurs choses. Le concept est ou empirique ou pur, et le concept pur, en tant qu'il a uniquement son origine dans l'entendement (et non dans une image pure de la sensibilité), s'appelle notion. Un concept tiré de notions et qui dépasse la possibilité de l'expérience est l'idée ou concept rationnel. » (Kant, 1787)

De son point de vue, le concept se rapporte à un objet par l'intermédiaire d'un signe. Il existe donc bien pour Kant une chaîne relationnelle entre la chose, son symbole et son concept, concept qui a son tour est constituant de la pensée, se **substituant** à la chose dans l'esprit du sujet pensant. Le lien entre concept et langue est ici représenté par l'usage du signe qui médie l'objet pour la pensée. Le signe est généralement un mot, et les concepts combinés entre eux forment des jugements, soit analytiques soit synthétiques. Pour Kant, l'idée est alors un « concept tiré de notions et qui dépasse la possibilité de l'expérience ». Il est assez difficile de juger des limites de l'expérience, aussi la distinction concept empirique/concept pur peut se révéler délicate. Une alternative à cette distinction pourrait être concept empirique/concept inféré.

De manière générale, c'est ce point de vue qui domine largement en intelligence artificielle, avec en avant poste les travaux sur la représentation de la connaissance, qui cherchent à représenter les choses et les prédications (ce qu'on peut dire des choses). Wittgenstein quant à lui a introduit une approche différente de celle de Kant, en cherchant lui-même à faire la critique de la tradition platonicienne. Dans un premier temps Wittgenstein (1921) a proposé une thèse proche de celle de Kant dans laquelle il considère que les propositions de la langue sont les miroirs de différents aspects du monde. La fonction de la langue est donc pour le premier Wittgenstein de reproduire la réalité, et donc ses objets. On en déduit alors naturellement que les concepts sont dans la pensée les constituants symétriques des choses dans le monde.

Wittgenstein a cependant, dans ses derniers écrits (Wittgenstein, 1963), réfuté cette première thèse, en s'intéressant à la *signification* des mots. Il nie en effet la possibilité qu'on puisse trouver dans le monde ou dans la pensée une matérialité à la signification. Il faut, pour Wittgenstein, considérer que la signification d'un mot est simplement le rôle joué par ce mot dans le processus d'interprétation de l'énoncé qu'il compose. Par conséquent, le *concept* en tant que composante de la pensée, doit nécessairement être la compétence permettant d'interpréter un mot dans un énoncé. Le *concept* correspondant au mot « table » serait donc

l'ensemble des compétences nécessaires pour interpréter la signification de ce mot dans toute occurence de phrase que l'on peut être amené à entendre.

Les thèses défendues par Wittgenstein ont partiellement inspiré la réflexion de Johnson-Laird, qui a proposé dans (Johnson-Laird and Miller, 1976) un modèle profondément axé sur le rôle dynamique des mots dans la construction de la signification des énoncés, modèle qu'il a repris plus précisément dans (Johnson-Laird, 1977). Johnson-Laird propose en effet que la signification d'un énoncé soit la procédure qui conduit l'interprétant à produire une réponse ou une action. Cela conduit à dire que les mots ne sont pas simplement substitués par des contenus sémantiques statiques avant que ceux-ci ne soient combinés entre eux pour former un autre contenu sémantique statique, mais que les mots sont transformés en une entité dynamique qui, en étant exécutée, donne sa sémantique à la phrase. Malheureusement Johnson-Laird a principalement conçu sa sémantique comme une vérification de la valeur de vérité d'une phrase, ou plus précisément en considérant qu'une proposition est une fonction d'un monde possible dans une valeur de vérité. Cette approche a donné lieu à une première critique, montrant que la sémantique d'une phrase n'est pas réduite à sa valeur de vérité (problème des jumeaux sur des terres différentes)[2]. Le second problème avec la thèse de Johnson-Laird est que la traduction d'un énoncé en une fonction, si elle est possible, ne serait finalement pas très différent de la traduction de l'énoncé en une structure statique, structure qui serait ensuite utilisée pour paramétrer l'exécution du programme qui ferait l'interprétation finale. Les programmes informatiques ne sont que des informations statiques interprétées par un autre programme, ou bien directement exécutées par un processeur, c'est-à-dire un assemblage de matière statique, elle même « interprétée » par les lois physiques. Pour résumer, la différence entre codage statique et codage procédural n'est qu'une question de point de vue, et l'argumentation de Johnson-Laird en faveur de la sémantique procédurale est infondée. Nous voilà alors devant une alternative : soit la sémantique procédurale de Johnson-Laird est une interprétation erronée de la thèse de Wittgenstein, soit la thèse de Wittgenstein n'est finalement qu'une reformulation de la thèse plus classique.

L'idée forte de Wittgenstein est que « *la signification est dans l'usage* » (*Meaning is use*). Mais quelle définition de « signification » utilise-t-il pour pouvoir la contester ? Il s'agit semble-t-il de la signification vue comme le rôle des mots dans la compréhension, ou encore la manière dont il doivent être utilisés pour parvenir à la compréhension. Notre interprétation de ceci est que la *signification* des mots n'est pas leur projection dans le monde ou ce à quoi ils correspondent dans le monde, mais *l'usage* qui est fait de ces mots dans la communication, leur *rôle* dans la construction du sens de la phrase dans laquelle ils apparaissent. Autrement dit, il ne faut pas voir les mots comme les composants d'un code, mais comme des indices permettant à l'allocutaire de se créer une « image ».

[2] L'argumentation de Fodor sur le paradigme des terres jumelles repose sur la situation suivante : « *Consider a twin earth paradigm : On earth-1 person-1 believs that Ronald Reagan is a nice guy and in fact he really is. On earth-2 person-2 (person-1's twin) also believes that RR is a nice guy, but on earth-2 RR is a meanny who just pretends to be a nice guy. The internal representations of RR for person-1 and person-2 are identical yet the truth value of their beliefs and hence the meaning of their respective statements are different.* » (Fodor, 1978).

3.2 Théorie des Points de Vue

D'après ce que nous avons exposé des différentes conceptions sur les représentations et les concepts, nous adoptons les hypothèses suivantes :

1. les informations manipulées par le système cognitif ne sont pas des représentations au sens de **substitut**, mais des représentations au sens de **point de vue**.
2. par voie de conséquence, nous considérons que la notion de concept telle qu'elle est définie par Kant est trop forte pour être utilisée comme composant de la cognition. A plus forte raison, une ontologie des concepts ne devrait pas être utilisée pour la modélisation d'un phénomène cognitif.
3. afin de rendre compte des relations entre objets et des capacités de catégorisation du système cognitif, nous proposons de faire reposer sur une action de catégorisation, autrement dit, une fonction de reconnaissance.

L'idée principale qui a émergé de nos travaux sur la référence a été de considérer que lors du processus d'interprétation d'un énoncé, ce ne sont pas toujours les mêmes points de vue sur les éléments (désignés par les mots de l'énoncé) qui sont utilisés. La sélection de tel ou tel point de vue pour un mot dépendant du contexte dans lequel le mot se trouve. Pour reprendre le terme de Fillmore ou les idées de Langacker sur l'appariement forme-sens, c'est donc la *construction* dans laquelle un élément apparaît qui détermine le point de vue qui sera adopté lors de l'interprétation pour cet élément.

Nous proposons de nous appuyer sur une idée simple pour fonder notre système d'interprétation. Nous appelons cette idée la Théorie des Points de Vue (TPV). Elle peut se résumer à l'hypothèse suivante :

> Le système cognitif et tout système qui a pour objectif de mimer celui-ci, est équivalent à un ensemble de fonctions telles que chacune de ces fonctions permet de produire un nouveau point de vue particulier à partir d'un ensemble d'autres points de vue préexistants. Ces points de vue sont des informations, la forme de ces informations (données continues ou discrètes) n'est pas déterminée *a priori,* elle peut être librement choisie en fonction des besoins. Les points de vue peuvent véhiculer aussi bien des perceptions que des actions.

La *théorie des Points de Vue* propose que ces fonctions soient des instance d'une opération de **transition de point de vue**, qui permet, en simplifiant, de remplacer la relation ontologique « est-un » par l'opération « Peut Être Vu comme ». L'aspect modal de l'opération est important, puisqu'il permet de contraindre la relation par certaines conditions (voir ci-dessous pour une illustration). L'opération de transition de point de vue permet d'expliquer le caractère compositionnel de la langue puisqu'elle permet de donner un point de vue nouveau à partir de plusieurs autres points de vue.

Ceci a notamment pour conséquence que dans les systèmes d'interprétation de la langue dérivés de cette approche, certaines caractéristiques doivent être respectées. Ces caractéristiques sont présentées ci-dessous.

3.2.1 Statut des Mots et des autres Perceptions

Dans la TPV, les mots et les énoncés n'ont pas un statut spécial par rapport aux autres perceptions, autrement dit aux informations venants d'autres modalités, notamment la modalité visuelle. En particulier, les mots ne sont pas reliés à des concepts qui se **substituent** à eux (pas de lexique dans le sens classique du terme). La notion de concept (concept de « table », concept de « prendre ») n'est donc pas utilisée dans la TPV. Concernant la signification et le sens ou la compréhension, leur définition donnant lieu à des approches trop divergentes, nous prenons le parti de ne pas les utiliser, et de nous concentrer sur la notion de *processus d'interprétation* qui nous semble plus simple à définir de manière consensuelle. Le *processus d'interprétation* est le processus réalisé par un système cognitif en réponse à la perception de son environnement, et qui le conduit à réaliser soit une action, soit une auto-modification (mémorisation).

Dans la théorie des points de vue on considère que l'interprétation consiste à dire *comment des perceptions peuvent être vues, dans le contexte des autres perceptions accessibles au moment de l'interprétation.*

3.2.2 Principe de non-catégorisation

Aucune présupposition n'est faite sur l'existence d'une ou plusieurs structures définissant des relations arbitraires entre les mots. En particulier, aucune notion de catégorie, lien de subsomption ou autre n'est présupposé entre les perceptions. La langue humaine et son utilisation permettant justement de construire ces notions, de les remodeler et de les supprimer, il semble nécessaire que leur existence ne soit pas un prérequis à un modèle d'interprétation.

Dans la théorie des points de vue, on considère qu'une perception donnée peut ou non être vue d'une autre manière, plutôt que de dire telle perception appartient ou pas à telle catégorie.

3.2.3 Principe d'enrichissement compositionnel

Comme un ensemble de points de vue peut à son tour être vu d'une nouvelle manière, l'opération de transition de point de vue peut fonctionner comme une règle dans un formalisme grammatical. En effet, étant des perceptions comme les autres, les mots servent de matière première en tant qu'information pour construire par fusion des informations plus riches, mais qui ne se substituent pas aux autres. Cela signifie que le processus d'interprétation ne construit pas des *représentations* de plus en plus abstraites d'un énoncé, comme autant de strates successifs, mais qu'il **enrichit** la perception brute par l'action d'opérations de fusion d'information jusqu'à ce que les informations puissent déclencher une action, soit interne en modifiant les opérations de fusion soit externe par une action agissant sur un élément moteur (qui a un impact sur l'environnement), y compris une production verbale.

Dans la théorie des points de vue, on considère que l'interprétation d'un énoncé dans un contexte donné, c'est l'ensemble des points de vue qui ont pu être construits à partir de l'énoncé et du contexte.

3.2.4 Principe généralisé de coercition

Le principe de coercition (Croft, 1991; Pustejovsky, 1991; Sag and Pollard, 1991; Pustejovsky, 1995; Goldberg, 1995; Michaelis, 2004) permet de décrire comment dans certains cas, une représentation peut être forcée dans une autre représentation afin de rentrer dans un schéma connu. En informatique, on connaît aussi ce mécanisme sous le nom de transtypage. Ce principe peut être considéré comme un dérivé linguistique du principe d'accommodation de Talmy (1977), qui tente par ce moyen d'expliquer les phénomènes de l'ordre de la métaphore.

Dans les grammaires de construction notamment, la coercition permet de rendre compte de certaines « souplesses » de la langue dans des cas exceptionnels.

Prenons le cas de certaines constructions qui évoquent un schéma de mouvement et sont constituées d'un verbe (qui isolément, n'évoque pas le mouvement) et d'un groupe prépositionnel de direction (*idem*) : « tasser » peut être utilisé dans « Il faut tasser le sable », où il n'évoque pas de mouvement, ou bien dans « J'ai tassé le sable dans le seau. », qui évoque un déplacement du matériau **sable** vers le contenant **seau**. De manière équivalente, « dans le seau » n'évoque pas le mouvement dans « Il y a du gravier dans le seau. »[3].

La construction qui permet de rendre compte de ce phénomène est donc composée d'un verbe et d'un groupe prépositionnel de direction, or certains groupes prépositionnels qui ne sont pas habituellement ambigus (« à l'intérieur de la maison » ne semble pas pouvoir être vu comme une direction) peuvent en fait être vues comme des directions, tandis que d'autres ne peuvent jamais l'être avec certains verbes (« sur la table » pour le verbe « tasser »).

3.2.5 Illustration de la théorie des points de vue

L'intérêt de la théorie des points de vue est particulièrement clair lorsqu'on la compare au paradigme de programmation objet et notamment à certaines difficultés que cette dernière ne permet pas de résoudre de manière satisfaisante. Par exemple, le cas très connu de l'héritage *figure* → *rectangle* → *carré*, exemple souvent cité pour illustrer la violation du principe de substitution de Liskov (Liskov, 1988), aussi connu sous comme le problème de l'héritage par restriction. On le trouve parfois aussi sous la forme *figure* → *ellipse* → *cercle.*

Ce problème s'énonce de la manière suivante : pour décrire les types de figures géométriques, un développeur de logiciel utilisant le paradigme objet décrit une hiérarchie dont le sommet est le type *figure.* Il sous-classe ensuite ce type avec le type *rectangle*, puisque un *rectangle* **est une** *figure*, et sous-classe le type *rectangle* avec le type *carré,* puisqu'un *carré* **est un** *rectangle.* Cette conception est représentée par la figure 3.1. Le problème soulevé par cette configuration est qu'elle viole le principe de substitution de Liskov, qui s'énonce ainsi :

> *What is wanted here is something like the following substitution property : If for each object o1 of type S there is an object o2 of type T such that for all programs*

[3]Ces exemples sont des tentatives pour traduire les constructions présentées par Goldberg (1995, pp.157–158) :
- « Sam squeezed the rubber ball inside the jar »
- « Sam sneezed the napkin off the table »

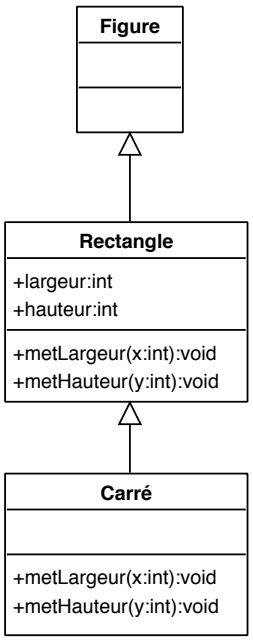


FIG. 3.1 – Héritage par restriction

> *P defined in terms of T, the behavior of P is unchanged when o1 is substituted for o2 then S is a subtype of T.* Liskov *(1988)*
>
> « Ce qui est recherché peut être exprimé par la propriété de substitution suivante : **si** pour chaque objet *o1* de type S il y a un objet *o2* de type T tel que pour tout programme P définit par rapport au type T, le comportement de P **n'est pas modifié** quand *o1* est substitué à *o2,* **alors** S est un sous-type de T. »

En effet, une hiérarchie comme celle définie par notre concepteur implique une redéfinition dans *carré* de certaines méthodes de *rectangle,* car un carré restreint certaines propriétés du rectangle. Typiquement, la méthode permettant de changer la largeur d'un rectangle « assure » qu'elle ne modifie que la largeur, or comme le carré restreint sa hauteur à être égale à sa largeur, la méthode définie pour le carré ne peut pas assurer cette propriété. Un objet de type *carré* ne peut donc pas être substitué systématiquement à un objet de type *rectangle,* et donc, selon la définition de Liskov, le type *carré* n'est pas un sous-type de *rectangle.*

Cette conclusion est contre-intuitive, et implique notamment que, pour faire une bonne modélisation, *rectangle* et *carré* ne soient pas père et fils, mais frères, c'est à dire qu'ils héritent tous les deux directement du type *figure.* Dans cette configuration, il n'y a donc pas plus de rapport entre un carré et un rectangle qu'entre un rectangle et un cercle.

Cette approche, si elle est tout à fait concevable dans le cadre de la programmation (il n'est nul besoin pour faire un bon programme d'être en adéquation avec l'intuition), pose un problème non négligeable pour une représentation « conceptuelle » utilisable dans le cadre de l'interprétation de la langue. En effet, pour interpréter correctement un énoncé, il est

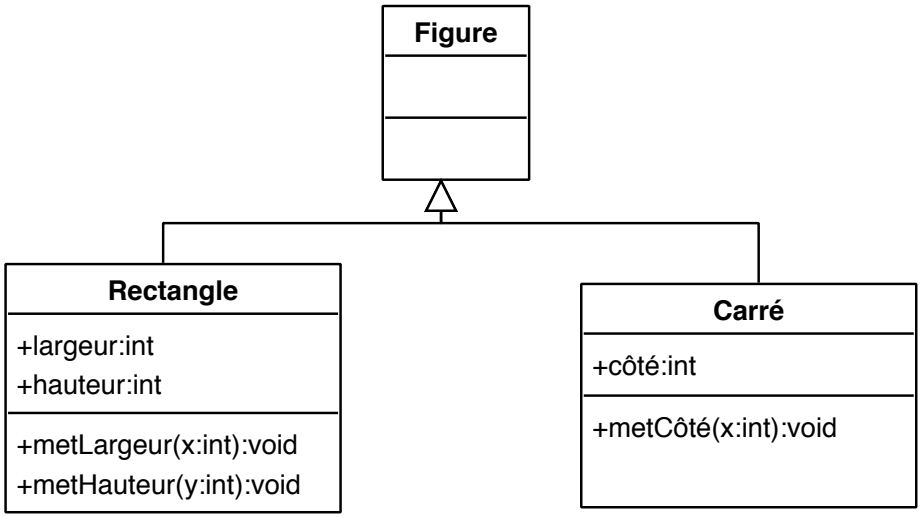


FIG. 3.2 – Héritage respectant le principe de substitution de Liskov

nécessaire de représenter le fait qu'intuitivement, un carré puisse être vu comme un rectangle.

Ce problème vient principalement du fait que le paradigme objet pose le *type* des éléments comme prévalant sur tout le reste. Un élément n'existe donc que parce qu'il est une instance d'un certain type, et ne peut pas changer de statut typologique. De plus, le paradigme objet tire son intérêt du fait qu'il associe les types aux opérations (méthodes) qui leur sont liées, or selon nous, intuitivement, les actions ne sont pas liées à des types d'objet, elles semblent avoir une existence particulière.

Pour illustrer ce problème, imaginez simplement que vous soyez devant un logiciel de dessin vectoriel. Vous dessinez un carré grâce à l'outil approprié, puis vous vous ravisez et décidez de l'élargir. Pour vous ça ne devrait pas poser de problème, le carré va simplement devenir un rectangle. Pour le logiciel en revanche, il n'est pas compréhensible de vouloir changer la largeur sans changer la hauteur *simultanément*. Vous allez donc devoir supprimer votre carré et créer un nouvel objet de type rectangle. Bien entendu, les bons logiciels de dessin du commerce permettent de faire ce genre de choses, mais pour cela, ils doivent passer par des typologies assez éloignées de l'intuition, ce qui n'est bien sûr pas notre objectif.

Dans un modèle respectant la théorie des points de vue, les relations entre *figure*, *rectangle* et *carré* seraient définies ainsi :

- un *rectangle* **peut être vu (P.E.V.)** comme une *figure*
- un *carré* **P.E.V.** comme une *figure*
- un *rectangle* **P.E.V.** comme un *carré* (si sa largeur est égale à sa hauteur)
- un *carré* **P.E.V.** comme un *rectangle*
- l'opération *changerLargeur* ne peut être appliquée à un *carré,* mais peut être appliquée au point de vue *rectangle* qui lui est attaché. L'objet résultant de cette transformation ne pourra plus être vu comme un *carré* à moins d'être remis à une taille où sa largeur égale sa hauteur.

Bien entendu, la théorie des points de vue semble assez difficile à adapter aux besoins de la programmation de logiciels. Elle poserait aussi des problèmes insolubles pour prouver la validité logique d'un modèle qui serait fondé sur elle. En revanche, si l'on se donne comme objectif de concevoir un système non déterministe et qu'on accepte qu'il ne soit pas décidable, alors cette théorie est un outil puissant pour la représentation de la compétence d'interprétation, puisqu'elle considère que la connaissance ontologique s'exprime par le même principe ou opération que la propriété compositionnelle de la langue.

Conclusion

Nous avons argumenté, dans ce chapitre, en faveur d'une approche différente de la représentation de l'information et des relations ontologiques entre les représentations. Cette approche consiste à considérer que les représentations jouent le rôle de *points de vue* sur les « choses », que ce soient des « concepts » ou des entités perçues, plutôt que de considérer les représentations comme des substituts à ces choses. La vision de la représentation par Points de Vue permet d'exprimer aussi bien :

– les relations ontologiques en contexte, en décrivant que certaines perceptions *peuvent être vues* d'après un autre aspect dans certaines situations,
– les relations compositionnelles entre un motif de symboles (par exemple linguistique) et des schémas sémantiques évoqués par ce motif.

Cette théorie de la représentation que nous avons appelée théorie des Points de Vue, nous permet de proposer une solution adaptable aux contraintes que nous avons posées au chapitre 2 pour notre modèle d'interprétation, tout en évitant l'écueil d'une représentation des relations ontologiques ne prenant pas en compte le contexte et qui soit centrée sur les catégories, ce qui est particulièrement handicapant pour la généricité du système.

Chapitre 4

Modèle d'interprétation constructionnelle

Ce chapitre expose le modèle que nous proposons pour l'**interprétation** dans le cadre d'un système de dialogue de type agent assistant d'interface. Nous avons, dans les chapitres précédents, décrit les phénomènes qui doivent être pris en compte par un tel système, à savoir les phénomènes dus aux caractéristiques visuelles (et donc spatiales), ceux dus aux caractéristiques temporelles, les références vagues, les métaphores, les métonymies, l'attribution directe des référents (résolution extensionnelle), le rôle du dialogue, la capacité d'apprentissage, ainsi que l'extensibilité (à rapprocher avec le polymorphisme dans les approches de la programmation orientées objet) dans une optique de généricité.

Dans cette thèse, nous n'avons pas couvert l'intégralité de ces points en profondeur. Notamment le rôle du dialogue ne fait l'objet d'aucun développement, de même que les aspects temporels. Nous avons cependant essayé de garder à l'esprit l'ensemble de ces points durant toute la conception du modèle.

La philosophie générale du modèle d'interprétation constructionnelle est inspirée par les différents modèles présentés au chapitre 2. L'idée directrice étant que les différents principes théoriques utilisées dans les modèles exposés doivent pouvoir être représentés grâce à un principe unique, généralisant les autres. Pour cela, nous avons adopté des points de vue légèrement décalés sur chacune des trois théories majeures qui ont été les guides de notre élaboration.

1. Comme il a été montré par Bryant (2003), les constructions de la Grammaire de Construction peuvent être traduites directement en informatique sans passer, comme cela est habituellement pratiqué, par une grammaire d'unification comme HPSG qui, elle est ensuite traduite en règles de grammaire hors-contexte. On sait que cette approche, qui a pour objectif de transformer une grammaire théoriquement non décidable en temps linéaire en une grammaire décidable, ne fait que déplacer le problème puisqu'elle génère une explosion du nombre de règles. L'approche de Bryant inspirée du *chunk parsing (Abney, 1991)* semble démontrer que la traduction directe n'est pas pire que l'approche hors-contexte, tout en étant plus proche de la théorie, et en offrant beaucoup plus de possibilités au niveau du développement de la grammaire (puisqu'il n'est pas nécessaire de remonter depuis la règle hors-contexte jusqu'à la construction pour comprendre un problème). Nous partons donc du principe qu'il n'est pas impossible

de traduire les constructions directement en unités de traitement. Nous exposons au chapitre 9 les principes qui vont permettre l'implémentation effective et efficace d'une telle approche, notamment en nous fondant sur la notion *d'attente*.

2. Pour prendre en compte le fait que de nombreuses logiques et modèles de raisonnement cherchent à modéliser des espaces ayant des caractères topologiques, nous introduisons la notion de **contexte**. Un contexte est un **conteneur** d'informations. Les informations en question sont des instances des **schémas** de la grammaire de construction. Un contexte est défini par :

   (a) des caractéristiques propres tels que rôles et contraintes sur les rôles (en cela il hérite des schémas).

   (b) une topologie décrite par les **lieux**, les **relations** entre les lieux et des **opérations** entre les lieux.

   (c) une liste d'instances de schémas associés à des lieux.

3. Comme les instances de schémas sont **situées** (dans les contextes) dans le Modèle d'Interprétation Constructionnelle, nous adaptons la notion de construction en proposant la notion de **construction située**. Une construction située, ou **s-construction**, décrit les relations entre des instances de schémas situées dans des contextes. On appelle ces instances de schémas les **constituants** (à la fois dans les constructions et dans les s-constructions).
   Mais contrairement aux **constructions** de la GC, les s-constructions ne décrivent pas simplement l'appariement entre des constituants dans les deux pôles *forme* et *signification*. Or cet aspect des constructions permet de déduire simplement quels sont les éléments existants qui doivent être observés (ceux du pôle forme) et ceux qui doivent être produits (ceux du pôle signification). Par conséquent, les s-constructions doivent *explicitement préciser* si un contexte est du côté des **observations** ou du côté des **productions**. Comme il est de plus envisageable qu'un contexte soit à la fois l'objet d'une observation et d'une production, il est en fait nécessaire de spécifier ceci pour *chaque constituant*.

4.1 Génèse du modèle d'interprétation constructionnelle

Nous avons donc considéré les **constructions** des grammaires de construction, non pas simplement comme une partie d'un formalisme décrivant de manière déclarative l'association entre une forme linguistique et une représentation sémantique, mais davantage comme des **règles de production** dont les **conditions** seraient *les contraintes sur la forme*, et où l'**action** serait *la formation de la structure sémantico-pragmatique*. L'idée est de pouvoir mettre en œuvre les constructions directement dans un système opérationnel, sans passer par une phase où les constructions sont exprimées dans une grammaire d'unification, avant d'être finalement implémentées comme des règles hors-contexte. Nous considérons donc que les constructions sont des unités de traitement, et donc qu'elles ont un aspect dynamique important.

D'un autre côté, nous avons considéré les **règles de production** non plus seulement comme des opérateurs pouvant modifier librement l'ensemble des stocks de connaissances exprimés

en logique du premier ordre, mais comme des opérateurs s'appuyant sur la sémantique des *frames* définie par Fillmore (1982). Les informations décrites dans la partie « condition » des règles de production deviennent les contraintes des constructions, exprimées de manière déclaratives. Les contextes et les informations à partir desquels ils définissent leurs conditions ou dans lesquels ils produisent de nouvelles informations sont donc décrits dans un formalisme suivant l'approche de Fillmore.

Enfin, si ces deux démarches permettent d'unifier grammaire et règles de production, il reste à prendre en compte les domaines de références, les cadres spatiaux, les classes de comparaisons et les intervalles temporels généralisés. Pour cela, il nous faut considérer les conteneurs des informations manipulées par les constructions ou les règles de production non pas comme des conteneurs non structurés, mais comme des espaces disposant de dimensions et d'une topologie permettant d'effectuer des mesures de distance entre les éléments contenus.

Au final, le modèle d'interprétation constructionnel repose sur trois concepts, permettant de modéliser une grammaire de construction, un système de production, et les différentes approches basées sur des espaces. Ces concepts sont :

1. les **schémas** qui permettent de décrire l'information,
2. les **contextes** qui permettent de décrire les structures qui vont contenir les instances de schémas dans une topologie,
3. les **s-constructions** qui permettent de décrire les relations entre les instances de schémas observées et celles qui sont produites.

4.2 Formalisme

Le modèle d'interprétation constructionnelle est principalement fondé sur la grammaire de construction. A ce titre, nous avons fondé le formalisme permettant de décrire les différents concepts du modèle d'interprétation constructionnel sur une forme proche du formalisme des grammaires de constructions. Nous avons choisi plus précisément de nous inspirer du formalisme utilisé pour la grammaire ECG (Bergen and Chang, 2002).

4.2.1 Schémas

Les schémas permettent de représenter le contenu informationnel manipulé par les constructions situées. Leur définition est principalement issue de la sémantique des frames (Fillmore, 1982). Un schéma S est défini par plusieurs groupes de caractéristiques :

hérite de Ce champ désigne le ou les schémas desquels S va hériter les rôles et les contraintes. Le schéma S a ainsi accès à tous les éléments définis dans ses schémas parents (son schéma parent direct, le schéma parent de celui-ci, et ainsi de suite).
Il faut noter une nette différence entre ce type d'héritage et l'héritage défini dans la sémantique des frames de Fillmore (et aussi l'héritage du paradigme de programmation orienté objet). En effet, alors que l'héritage dans la sémantique des frames décrit une liaison ontologique impliquant une prise en compte de la généralisation, l'héritage dans

schéma <*Schéma*>
hérite de <*Schéma*>
évoque
  <*instance*> : <*Schéma*>
rôles
  <*rôle*>
  <*rôle*> : <*Type*> // *Plusieurs types sont prédéfinis (Entiers, Réels, Chaîne Phonétique, ...)*
  <*rôle*> : <*Schéma*>
contraintes
  <*rôle*> ← <*valeur*> // *valeur assignée*
  <*rôle*> ↔ <*rôle-local* | *instance.rôle*> // *les valeurs des deux côtés sont identiques*
  <*prédicat*(<*rôle*>, <*rôle*>)>

Spécification 3: Description d'un Schéma dans le modèle d'interprétation constructionnelle.

le modèle d'interprétation constructionnelle n'est qu'un héritage des propriétés descriptives.
Illustration : soit un schéma X ayant un rôle *a* contraint à être de type A et un schéma *b* de type B qui hérite du type A ; dans la grammaire de construction, *a* peut être « rempli » par *b* directement ; dans le modèle d'interprétation constructionnelle, il faut qu'une construction soit définie explicitement, qui permette de définir la relation entre un schéma de type A et un schéma de type B, ce qui fera que *b* sera transformé en *b'* de type A, *b'* pouvant alors être utilisé pour remplir *a*.
Enfin, il faut noter que contrairement aux langages de programmation, aucun mécanisme ne permet de limiter l'accessibilité des éléments (déclarations privées, protégées ou publiques dans les langages orientés objet).

évoque Désigne les schémas qui ont un rapport avec le schéma principal S. Par exemple, un schéma décrivant un déplacement évoque un schéma de type schéma-mouvement-causé (*caused-motion-schema*), même s'il ne donne pas les informations sur tous les rôles de ce schéma. On peut comparer les schémas *évoqués* à ce qui, dans le paradigme objet, est désigné par le nom d'*association*, par opposition aux *compositions*, qui elles, s'approchent davantage des *rôles* définis ci-dessous et décrivent les éléments qui font partie intégrante de l'information définie par le schéma.
Le mécanisme d'*évocation* a été introduit dans la grammaire ECG afin de donner une expressivité plus grande que le simple héritage. L'idée est qu'un schéma évoqué existe en-dehors du schéma S, ce qui permet par la suite de « fusionner » deux schémas partiellement décrits provenant de deux sources différentes.
Un schéma *évoqué* est libellé par une étiquette qui permettra de le référencer dans les contraintes du schéma ou dans les contraintes des constructions situées qui y réfèrent.

rôles Ce champ décrit les attributs et liaisons qui définissent le schéma. C'est donc lui qui permet de donner le cadre du contenu informationnel des instances du schéma et leur composition. Chaque attribut est défini par un type, qui peut être un schéma ou un type atomique (types caractère, entier, réel, etc.). Un attribut représente une valeur intrinsèque au schéma, qui ne peut pas être partagée par plusieurs schémas, contrairement aux éléments *évoqués*.
Les *rôles* peuvent être vus comme des réceptacles, leur valeur n'étant pas spécifiée ex-

plicitement, mais définie par les contraintes qui s'appliquent dessus.
Pour désigner un rôle, on utilise la notation par chaîne d'attributs, par exemple : *mouvement.cause* ou bien encore *figure.couleurRGB.composanteRouge* .

contraintes Les contraintes définissent les relations qui doivent être vérifiées entre les différents champs *rôles* (y compris les rôles hérités et les rôles des schémas évoqués). Plusieurs types de contraintes sont définies, dont :

L'assignation, exprimée par l'opérateur ←. L'assignation force un rôle à être rempli par une certaine valeur. La partie gauche de l'assignation est forcément un rôle, sa partie droite est forcément une valeur constante. Exemple : *patient.genre* ← MASCULIN.

L'identification, exprimée par l'opérateur ↔. L'identification contraint ses deux parties à avoir le même contenu. Les deux parties de l'expression sont soit des rôles simples, soit des expressions mettant en jeu un seul rôle simple. Exemple : *self.arrivée* ↔ *scd.destination.* (*scd* étant une instance d'un schéma *source-chemin-destination*)

Prédicats. Outre les deux relations définies ci-dessus, le formalisme de définition des contraintes du modèle d'interprétation constructionnelle, comme celui de l'ECG, permet d'utiliser des contraintes prédicatives dont la sémantique est décrite de manière extérieure au modèle. On peut notamment définir ainsi des contraintes arithmétiques. Exemple : *plus-petit-que(x,y)*

4.2.2 Contexte

La notion de contexte recouvre, entre autres, les caractéristiques des espaces mentaux et des domaines de référence en introduisant une méthode générale de définition des relations topologiques entre objets.

- La fonction principale d'un contexte est celle de **conteneur**, c'est-à-dire qu'un contexte va d'abord permettre de regrouper ensemble plusieurs instances de schémas.
- La seconde caractéristique majeure des contextes est qu'ils permettent d'affecter une **position** à chaque instance de schéma contenue par le contexte. A partir de cette position et de la topologie définie par le contexte, il est ainsi possible de définir des relations extrinsèques entre les positions de différentes instances de schémas.

Par rapport à la définition formelle des espaces mentaux proposée dans (Chang et al., 2002), nous introduisons cette notion de structure topologique via la définition de concepts permettant de décrire des contraintes utilisables ensuite dans les constructions. Le formalisme de description des contextes représenté figure 4 permet de définir à la fois les informations sur le contexte (des rôles et des contraintes sur les rôles), et la structure du contexte. La structure est définie par des **lieux** (des descriptions de position abstraites liées aux entités et définissant leur position dans la structure), des **relations** (sur lesquelles on pourra poser des contraintes, applicables entre les entités) et des **opérations** permettant de combiner des positions pour construire des contraintes complexes.

La définition des contextes reprend les caractéristiques d'un schéma, à savoir les champs **hérite de**, **évoque**, **rôles** et **contraintes**. Un contexte est donc au moins équivalent à un

contexte <*Contexte*>
hérite de <*Contexte*>
rôles
  <*rôle*>
  <*rôle*> **:**<*Type*>
  <*rôle*> **:**<*Schéma*>
contraintes
  <*rôle*> ←<*valeur*>
  <*rôle*> ↔ <*rôle*>
  <prédicat(<*rôle*>, <*rôle*>)>
lieux
  <*lieu*> // *par exemple :* segment
relations
  <*relation*(<*lieu*>, <*lieu*>,...)> ↦ <*Type*>
  // *par exemple :* précède(segment, segment) ↦ booléen
opérations
  <*opération*(<*lieu*>, <*lieu*>,...)> ↦ <*lieu*>
  // *par exemple :* union(segment, segment) ↦ segment

Spécification 4: Définition des contextes dans la notation ECG Bergen and Chang (2002).

schéma en tant que représentation d'information. A ces caractéristiques se rajoutent trois champs permettant de décrire la topologie du contexte :

lieux Ils permettent de définir les informations sur la « localisation » des entités contenues dans les contextes. Une information de lieu peut être rattachée directement à une entité, ou bien construite à partir d'autres lieux pour pouvoir imposer des contraintes dessus. Par exemple, si l'on veut représenter le contexte verbal écrit, ses lieux seront PLACE-SYMBOLE (pour les lettres et les signes), et SEGMENT-SYMBOLE (pour les morphèmes, entre autre). Pour une représentation d'un domaine de référence, qui est une chaîne d'entités séparées par une distance, il faut définir les lieux NOEUD, et CHAINE. Un même noeud peut être lié à plusieurs entités.
:e formalisme ne décrit pas le contenu informationnel des lieux[1], qui est laissé à la responsabilité de l'implémentation, afin de pouvoir opérer les optimisations nécessaires, notamment si l'on envisage des contextes décrivant des espaces à deux dimensions ou plus.

relations Elles définissent les rapports structurels entre les lieux, et permettent donc de décrire dans les constructions les contraintes que l'on veut voir respecter sur la localisation des entités dans les contextes. Une relation doit spécifier les types de lieux sur lesquelles elle est applicable, ainsi que le domaine de sa valeur.

Par exemple, pour le contexte verbal, qui est un espace à une dimension, on aura la relation VECTEUR(PLACE-SYMBOLE, PLACE-SYMBOLE) → $\mathbb{Z}$ qui donne le vecteur à

[1]Par exemple qu'une distance entre deux lettres est représentée par un **réel** et qu'une distance entre deux objets sur un plan est représentée par un **complexe**.

une dimension entre deux symboles (VECTEUR(X,Y)=0 si X=Y ; -1 si Y suit immédiatement X ; 1 si X suit immédiatement Y). Pour les domaines de référence, qui sont des chaînes d'éléments, on aura RAPPORT_SIMILARITÉ(NOEUD, NOEUD) $\rightarrow \mathbb{R}^+$, avec RAPPORT_SIMILARITÉ(X,Y)=1 si X et Y sont au même niveau sur l'échelle du prédicat (par exemple X et Y aussi grands) ; 0 (resp. ∞) si Y (resp. X) est hors de l'échelle mais pas X (resp. Y) ; >1 (resp. <1) si X (resp. Y) est au-dessus de Y (resp. X) sur l'échelle du prédicat[2].

opérations Elles permettent de construire des lieux à partir d'autres lieux. Elles permettent notamment de travailler sur des lieux construits de manière abstraite, par exemple pour des groupements perceptifs (plusieurs objets proches ayant une ou plusieurs caractéristiques communes, et qui peuvent être perçus comme un seul élément).

Par exemple, pour construire une chaîne dans les domaines de référence à partir de deux noeuds, on aura : CHAÎNE(NOEUD, NOEUD) $\rightarrow$ CHAINE ; pour extraire le premier noeud d'une chaîne : PREMIER(CHAINE) $\rightarrow$ NOEUD.

contexte Linéaire
rôles
longueur : réel
nb-élément : entier
longueur-moyenne-élément : réel
contraintes
<rôle> $\leftarrow$<valeur>
<rôle> $\leftrightarrow$ <rôle>
lieux
segment
relations
égal(segment, segment)$\mapsto$booléen
précède(segment, segment)$\mapsto$booléen
rencontre(segment, segment)$\mapsto$booléen
// par exemple : précède(segment, segment) $\mapsto$ booléen
opérations
union(segment, segment) $\mapsto$ segment
intersection(segment, segment) $\mapsto$ segment

Spécification 5: Exemple de définition d'un contexte à une dimension.

Il faut noter que le formalisme de description d'un schéma ne porte absolument aucune information sur la sémantique opérationnelle des relations et des opérations, ni même sur le contenu des lieux en terme d'information (exemple : un segment dans un contexte à une dimension tel que celui défini figure 5 est décrit par une liste de couples de réels, chaque couple contenant les coordonnées de début et de fin d'un sous-segment). Cette sémantique opérationnelle des contextes est la réalisation effective de la logique qui permet de décrire

[2]La relation RAPPORT_SIMILARITÉ n'est pas symétrique, car elle indique un rapport de similarité par rapport à une certaine dimension ou composante, par exemple le rapport entre deux éléments du point de vue de leur rapprochement avec la couleur « bleue ».

une certaine topologie, et par conséquent, cette sémantique doit être implémentée par une approche particulière, que notre formalisme ne cherche pas à détailler, notamment parce que cela impliquerait de décrire des algorithmes procéduraux, et donc proposer tout un formalisme pour cela.

Un contexte est donc simplement la déclaration de l'interface permettant aux s-constructions d'utiliser les différentes topologies définies dans le modèle d'interprétation de manière uniforme. L'implémentation exacte de la topologie étant laissée à la charge d'un langage de programmation classique. Bien entendu, cela signifie que si l'implémentation n'est pas parfaitement valide, l'utilisation d'une certaine topologie sera sujette à introduire de l'inconsistance dans l'interprétation.

4.2.3 Constructions situées

s-construction <*S-Construction*>
évoque
  <*cons*> **:**<*S-Construction*>
contextes
  <*contexte*> **:**<*Contexte*>
  // *exemple :* txt **:**Lex-Env
constituants
  <*instance*> **:**<*Schéma*>**@**<*contexte*> / *E|S*
  // ***E*** *pour entrée (observation),* ***S*** *pour sortie (production)*
  // *par exemple :* a **:**Adjectif**@**txt **/** E
contraintes structure
  <*contexte.relation*(<*instance*>, <*instance*>, <...>)> ← <*valeur*>
  <*contexte.relation*(<*contexte.opération*(<*instance*>, <...>)>, <...>)> ← <*valeur*>
  <*contexte.relation*(<...>)> ↔ <*contexte.relation*(<...>)>
  // *par exemple :* txt.précède(sujet, verbe) ← vrai
  // *les domaines des deux côtés des contraintes doivent être identiques*
contraintes contenu
  <*rôle*> ← <*valeur*>
  <*rôle*> ↔ <*rôle*>
  <*prédicat*(<*rôle*>,<*rôle*>)>

Spécification 6: Définition des constructions situées

Les **constructions situées** (s-construction dans notre formalisme) dont la structure est décrite sur la spécification 6, définissent les relations liant des informations (représentées par des instances de schémas) contenues dans deux ensembles de contextes d'interprétation : l'ensemble **observation** et l'ensemble **production**. Ces deux ensembles de contextes peuvent être vus comme des généralisations des pôles **forme** et **signification** des constructions dans la grammaire de construction, rendues insuffisantes par le fait qu'on ne veut plus limiter les constructions aux analyses morphologiques, syntaxiques et sémantiques, mais prendre en compte des niveaux d'analyse nécessitant des structures plus complexes indispensables pour prendre en compte, par exemple, le support visuel.

Une s-construction spécifie donc les relations entre d'une part des instances de schéma **observables** (par exemple, qui vont permettre de représenter l'énoncé prononcé par l'utilisateur) et les instances de schéma **produites** qui vont représenter un point de vue sur les observables de manière à participer à la construction de la représentation finale qui permettra d'effectuer l'action répondant à l'énoncé. De ce fait, elle définit une *transition de points de vue*.

Contrairement aux **schémas** et aux **contextes**, une **s-construction** n'a pas pour finalité de contenir une information, elle ne contient donc pas de rôle. Dans le même ordre d'idée, nous n'avons pas autorisé l'héritage entre s-constructions, contrairement à la grammaire ECG qui permet de faire hériter une construction d'une autre. Une s-construction est définie par les champs suivants :

évoque Comme les schémas et les contextes, une s-construction peut en évoquer une autre, ce qui permet par exemple de « forcer » une transition de point de vue à être réalisée même quand elle ne l'aurait pas été d'après sa propre définition. Ainsi, une s-construction peut déclencher par évocation deux transitions de point de vue simultanément.

contextes Ce champ définit la liste des contextes dans lesquels la s-construction va observer et produire des instances de schémas. Les contextes sont définis par une étiquette (qui sera utilisée dans le reste de la déclaration de la s-construction pour y référer) et un type de contexte (défini précédemment). Une s-construction établissant un lien entre le mot « mets » entré par l'utilisateur et un point de vue lemmatisé définira deux contextes :

- *txt* de type **Linéaire-Mots**
- *lem* de type **Linéaire-Lemmes**

constituants Les constituants sont les instances de schémas entre lesquelles la s-construction va spécifier les relations qui doivent être vérifiées. Le champ contient une liste d'instances de schémas, définis par une étiquette (qui sera utilisée dans les contraintes pour référer aux différents constituants), un type de schéma (qui contraint l'instance à être une instance d'un certain schéma), le contexte dans lequel l'instance doit être localisée (qui doit être une étiquette de contexte définie dans le champ **contextes**) et enfin un symbole spécifiant si le constituant fait partie du pôle des observations (symbole E pour Entrée) ou du pôle des productions (symbole S pour Sortie). Pour continuer notre exemple, les constituants seraient :

- m **:**Mot**@**txt/E (étiquette *m*, de schéma *Mot*, dans le contexte *txt*, pôle *observation*)
- l **:**Mettre**@**lem/S (étiquette *l*, de schéma *Mettre*, dans le contexte *lem*, pôle *production*)

contraintes structure Les contraintes portant sur la structure définissent les relations structurelles entre les instances de schémas contenues dans le même contexte, ou bien entre les positions d'instances de schémas contenues dans des contextes différents. Ici, on aurait simplement :

- $lem.\text{position}(l) \leftrightarrow txt.\text{position}(m)$

contraintes contenu Les contraintes définies dans ce champ portent sur les contenu des instances de schéma. Ce sont les mêmes types de contraintes que celles définies dans les schémas et les contextes. Dans notre exemple, on aurait :

- $lem.\text{orig} \leftrightarrow txt$

– m.forme ← « mets »
– l.origine ↔ m
– l.pers ← 2SING
– l.mode ← IMPÉRATIF

L'exemple développé ici pour illustrer la composition des s-constructions est regroupé dans la spécification 7. Il faut noter que c'est un exemple extrêmement simple de ce que peut représenter une s-construction. Il s'agit en fait de l'équivalent constructionnel d'une partie d'une entrée de lexique dans une grammaire lexicale.

s-construction Mets-Imp
contextes
txt **:***Linéaire-Mots*
lem **:***Linéaire-Lemmes*
constituants
m **:***Mot***@***txt/E*
l **:***Mettre***@***lem/S*
contraintes structure
lem.position(l) ↔ *txt*.position(m)
contraintes contenu
lem.orig ↔ *txt*
m.forme ← « mets »
l.origine ↔ m
l.pers ← 2SING
l.mode ← IMPÉRATIF

Spécification 7: S-Construction pour « mets » à l'impératif

4.3 Processus d'interprétation

Le modèle d'interprétation constructionnelle décrit le processus d'interprétation comme une succession de transitions de points de vue, tels que nous les avons présentés précédemment (chapitre 3), ce qui correspond à une succession d'observations suivies de productions. Les **constructions situées** décrivent la relation entre les deux parties de la transition, l'observation et la production, mais elles ne décrivent pas comment la transition va opérer. Pour décrire ceci, il nous a fallu concevoir le pendant opératoire des constructions situées, nous l'avons désigné par le terme **observateur**.

Un observateur ressemble par plusieurs aspect aux sources de connaissances des *blackboards* (Hayes-Roth, 1985), mais il en diffère par plusieurs aspects :

– les « étages » ou niveaux dans un *blackboard*, sont organisés de manière linéaire, comme un empilement, alors que les contextes sont librement structurés,
– les sources de connaissances sont attachées à un niveau du *blackboard*, alors que les observateurs peuvent requérir l'instanciation d'un nouveau contexte, ainsi que déterminer les contextes dans lesquels ils choisissent leurs constituants ,

– le contrôle des sources de connaissances est assuré par une heuristique spécialisée dans un *blackboard,* alors que les **observateurs** s'organisent selon plusieurs principes directeurs, qui sont détaillés dans le chapitre 9.

Pour illustrer la manière dont nous envisageons l'exécution du processus d'interprétation, prenons l'exemple d'un utilisateur dialoguant avec le système qui permet de manipuler des formes géométriques simples dotées d'une couleur. L'utilisateur veut faire exécuter une certaine action, à savoir supprimer des objets qu'il voit à l'écran, et qui lui semblent être des carrés de couleur bleue.

– Il prononce (ou tape au clavier) un énoncé comme « *supprime les deux grands carrés bleus* », qui devient perceptible au système.
– Les éléments de l'énoncé vont déclencher des observateurs, par exemple le sous-énoncé « *les deux grands carrés bleus* » va être vu comme un motif « ARTICLE-DÉFINI NUMÉRAL ADJECTIF NOM ADJECTIF » qui lui-même va pouvoir évoquer un information de type EXPRESSION-RÉFÉRENTIELLE. Si l'énoncé dans son ensemble est construit d'une manière qui soit satisfaite par ce point de vue, l'exécution de l'observateur va être encouragée.
– D'autre part, un point de vue plus sémantique sur le sous-énoncé va être produit quasi simultanément, en considérant les signes qui composent « *les deux grands carrés bleus* » comme cinq extracteurs référentiels, ce qui va permettre de compléter le point de vue EXPRESSION-RÉFÉRENTIELLE en lui associant les cinq extracteurs.
– Enfin, pour ce qui concerne cette expression référentiel du moins, il faut encore un observateur dont le motif serait : EXPRESSION-RÉFÉRENTIELLE@ÉNONCÉ (c'est-à-dire une instance du schéma EXPRESSION-RÉFÉRENTIELLE dans un contexte de type ÉNONCÉ), et le contexte DOMAINE-RÉFÉRENCE. Ceci permettrait de faire le lien entre les objets accessibles sur le support visuel, et l'expression référentielle.
– Parallèlement, le système « perçoit » en permanence les objets de l'application. Le contexte requis de type DOMAINE-RÉFÉRENCE va être initialement rempli avec les objets venant du contexte qui contient les éléments visibles.

Cet observateur va alors, en déclenchant d'autres observateurs intermédiaires, construire un contexte contenant la perception qui est la cible de la référence « *les deux grands carrés bleus* ». Cette référence va être rattachée à l'expression référentielle, et va pouvoir être utilisée pour produire une représentation de l'action à mener, en réponse à l'énoncé.

4.4 Observateurs et ontologie

Dans le modèle d'interprétation constructionnelle, les observateurs, les schémas et les environnements permettent de décrire le processus d'interprétation. Il existe, on l'a vu, un mécanisme d'**héritage** entre les différentes définitions de la même famille. Ainsi, un schéma A peut être défini comme un sous-cas d'un schéma B. Dans ce cas, le schéma A sera composé des rôles et contraintes décrites explicitement dans sa définition, ainsi que des rôles et contraintes décrites dans la définition de B. Par exemple, dans la spécification 8 est défini un schéma *Cercle* qui hérite d'un schéma *Figure*. Le contenu réel du schéma *Cercle* est représenté dans le schéma de droite.

Un avantage évident de ce mécanisme de sous-catégorisation est d'éviter de redéfinir des rôles lorsqu'ils sont partagés par plusieurs schémas. Bien entendu, l'héritage n'a d'intérêt réel que

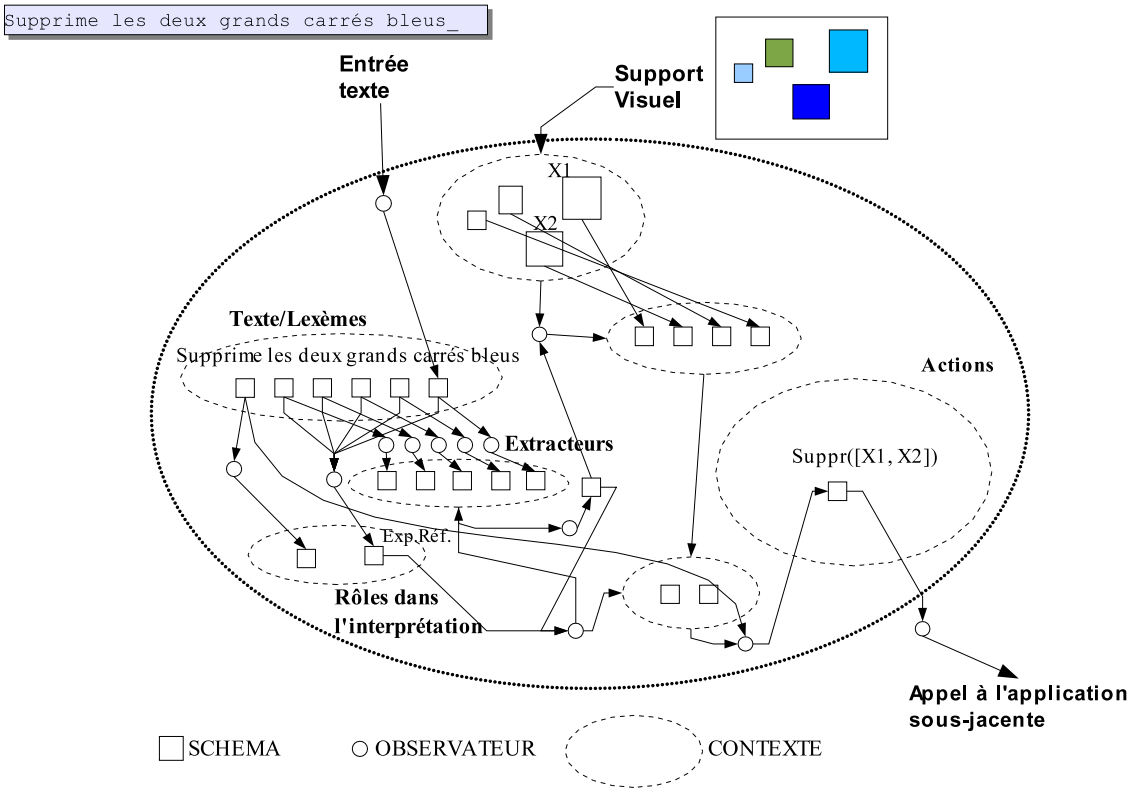


FIG. 4.1 – Schéma général du processus d'interprétation du modèle d'interprétation constructionnelle

schéma Figure
rôles
couleur : Couleur
centreGravité : Point

schéma Cercle
hérite de Figure
rôles
rayon : réel
centre : Point
contraintes
centre ↔ *centreGravité*

⇔

schéma Cercle
rôles
couleur : Couleur
centreGravité : Point
rayon : réel
centre : Point
contraintes
centre ↔ *centreGravité*

Spécification 8: Sous-catégorisation (héritage) dans les schémas

s'il s'applique aussi aux opérations définies sur les schémas, et non pas seulement comme « raccourci » pour la déclaration.

Dans le cas de la programmation objet, l'héritage permet de rendre une méthode qui s'applique à un objet de classe A, applicable à un objet de classe B si la classe B hérite de la classe A. Par exemple si on définit *ChangeCouleur(Figure f)*, alors on doit pouvoir appeler *ChangeCouleur(c)* si *c* est un *Cercle* et que *Cercle* hérite de *Figure*.

Dans le cas de la grammaire de construction, et par conséquent du modèle d'interprétation constructionnelle, ce sont les constructions qui doivent supporter l'héritage. Cela signifie que si un des composants d'une construction X doit être une instance de *Figure*, alors une instance de *Cercle* doit être acceptée pour remplir ce composant.

On connaît relativement bien les limites de cette approche dans les langages de programmation. Des problèmes se posent notamment lorsque l'on souhaite autoriser l'héritage multiple. En cas d'héritage multiple, il est en effet impossible de choisir entre les différents parents possibles lors de l'accès à une variable ou l'appel d'une méthode si celles-ci sont définies pour les deux classes. Par exemple si on a deux classes *Avion* et *Bateau* et qu'on veut décrire un hydravion, on fera une classe *Hydravion* héritant à la fois de *Avion* et de *Bateau*, mais si les deux ont une variable « moteur » ou une méthode « démarrage », soit on doit explicitement dire à quelle variable ou méthode on veut accéder, soit on doit passer par une super-classe commune aux deux parents (ici : *Véhicule*) qui définit, seule, les variables et méthodes communes.

Cette solution n'est cependant pas très convaincante, car il faut alors définir une pléthore de classes très générales, et souvent incompatibles. Si l'on veut définir un pédalo comme une classe héritant à la fois du vélo et du bateau, alors il faut définir la classe véhicule sans moteur, ainsi qu'une classe « bateau à rame ». L'intérêt de l'héritage multiple pour la clarté et la simplicité du code est donc difficile à arguer.

Par conséquent, il est tout aussi difficile de considérer que la sous-catégorisation de la grammaire de construction soit une solution idéale au problème de la généricité des constructions (c'est-à-dire qu'une construction définie pour un type A puisse être utilisée avec tous les sous-types de A). En plus de ces problèmes purement pratiques, on peut s'interroger sur la plausibilité des mécanismes d'héritage par rapport à la cognition. Considérer que ce mécanisme est au coeur des capacités de généralisation des fonctions cognitives reviendrait à postuler que les concepts sont explicitement représentés dans le système cognitif, et que leurs liens ontologiques sont explicitement définis, or c'est une vision peu plausible.

La solution à ce problème réside selon nous dans un changement radical d'approche sur les catégories et les concepts, suivant la Théorie des Points de Vue. Comme nous l'avons expliqué au chapitre 3, il s'agit de s'appuyer pour décrire la généricité sur la seule notion de « changement de point de vue ». Si l'on définit une construction X comme reconnaissant un schéma A et produisant un schéma « Vue de A comme B » (p. ex. A=Cercle, B=Figure), alors on définit implicitement un lien ontologique « est-un » entre A et B. Toute construction définie pour reconnaître B fonctionnera avec A à travers cette « Vue de A comme B ». Ce qui permet très facilement de décrire le fait qu'un Hydravion soit à la fois un Avion et un Bateau, car il suffit de deux constructions, l'une produisant une « Vue de Hydravion comme Avion » et l'autre « Vue de Hydravion comme Bateau ». Le lien de généralisation suit le même chemin que dans l'approche classique, comme le montre la figure 4.2. La différence est cependant subsantielle, à savoir que dans le cas de l'héritage classique, l'hydravion est simultanément un avion **et** un bateau, alors que dans l'approche par points de vue, une instance donnée peut être vue, à un moment donné de l'interprétation, **soit** comme un avion, **soit** comme un avion, en fonction du point de vue nécessaire à la bonne interprétation.

En plus de permettre un héritage puissant offrant une grande généricité, ce mécanisme permet de traiter des représentations différentes d'un même objet. En effet, un cercle, par exemple, peut être défini de différentes manières : soit un point et un rayon, soit un point et un diamètre, soit un segment (qui détermine les deux extrémités d'un diamètre), soit encore par

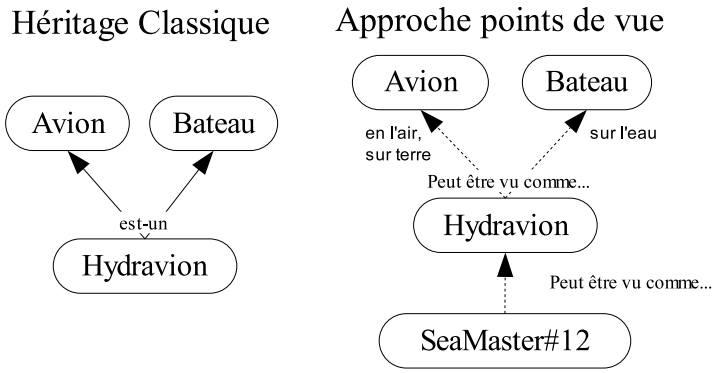


FIG. 4.2 – Héritage classique à gauche. Approche par points de vue à droite.

son triangle inscrit, etc. Avec un modèle d'héritage classique, illustré figure 4.3, il faut passer par un mécanisme de coercition explicite, et il est impossible de distinguer la relation entre deux représentations différentes du même objet et deux objets de type différents.

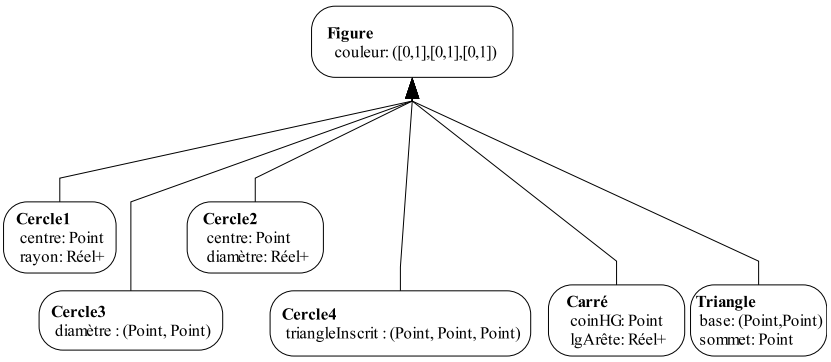


FIG. 4.3 – Hiérarchie décrivant différentes représentations d'un cercle et d'autres figures dans un mécanisme d'héritage classique

Dans le modèle des points de vues, en revanche, le passage d'une représentation à une représentation équivalente se fait par une transformation directe (figure 4.4), tandis que le passage d'un type à un type plus générique se fait à travers un point de vue, qui encapsule la représentation du type parent et l'instance d'origine à partir de laquelle le point de vue a été généré.

Conclusion

Nous avons présenté les différentes composantes du modèle d'interprétation constructionnelle, inspiré de la grammaire de construction et de la grammaire ECG. Nous avons notamment

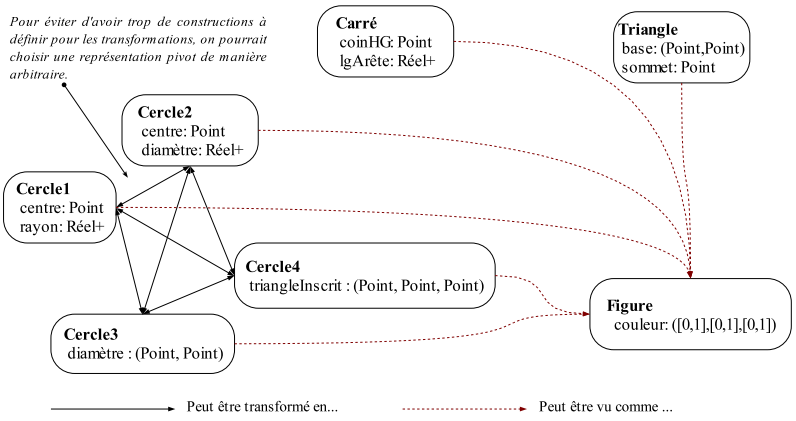


FIG. 4.4 – Configurations des transitions de point de vue pour quatre concepts représentant un cercle et d'autres figures ainsi que leur type parent.

introduit la notion de *contexte* et la notion de *s-construction*, qui est la version située des constructions, prenant en compte les *contextes*.

Alors que les constructions initialement proposées dans les grammaires de constructions ne permettait de décrire que des contraintes sur le **contenu** des instances de schéma ou sur l'**ordre des mots**, nous avons introduit une notion plus générale permettant de décrire des contraintes structurelles sur n'importe quel conteneur structuré qui puisse être implémenté informatiquement et interfacé via le formalisme des *contextes,* avec trois composantes : les *lieux*, les *relations* entre les lieux et les *opérations* sur les lieux.

Nous avons ainsi atteint notre objectif de réunir dans un formalisme unifié la puissance descriptive nécessaire pour supporter aussi bien une théorie linguistique comme la grammaire de construction, qu'un modèle reposant sur des structures dotées d'une topologie, comme les domaines de référence, les espaces mentaux, ou les modèles de raisonnement sur l'espace ou le temps.

Parallèlement, par rapport au modèle de la grammaire ECG, nous avons éliminé les caractéristiques ontologiques dans la description des schémas, qui allaient à l'encontre de nos conclusions sur la représentation des catégories, présentées au chapitre 3. Pour supporter les mécanismes de généralisation et de coercition, nous avons proposé de nous reposer simplement sur les *s-constructions*, qui sont alors considérées comme des opérateurs de transition de Points de Vue.

Ayant dérivé le modèle d'interprétation constructionnelle à partir de la grammaire ECG, nous ne jugeons pas nécessaire de montrer qu'il permet de décrire l'aspect linguistique de l'interprétation. Il nous faut en revanche montrer qu'il peut effectivement être utilisé afin de décrire un modèle utilisant effectivement d'autres structures que celle permettant de traiter l'ordre des mots. C'est ce que nous faisons dans la partie II de notre thèse.

Deuxième partie

Résolution Extensionnelle de la Référence dans le Modèle d'Interprétation Constructionnelle

Chapitre 5

Le phénomène référentiel

5.1 Philosophie de la référence

La question de la référence s'est d'abord posée en termes philosophiques, et de manière étroitement liée à la notion de représentation que nous avons abordé au chapitre 3. Les réflexions sur la référence prennent leurs racines modernes dans les travaux de Frege, qui ont notamment donné naissance, avec les travaux de Russel (1905) et Church (1941), à la notion de condition de vérité comme représentation d'un énoncé, et donc aux approches véricondi tionnalistes de la compréhension.

Dans *Sinn und Bedeutung*, (*Bedeutung* pour lequel nous préférons la traduction anglaise de Church, *denotation*, aux traductions *meaning* de Evans (1982) ou *reference* couramment utilisé), Frege développe une approche du phénomène référentiel et du sens qui l'amène à l'idée qu'un groupe nominal dénote un objet, qu'un groupe verbal dénote un concept, et finalement qu'une phrase dénote une valeur de vérité. Il rejette ainsi toute interprétation extensionnelle de la dénotation, qui permettrait de dire que deux expressions référentielles ayant la même extension (dénotant le même objet) peuvent être substituées indifféremment dans une phrase sans en changer le sens.

Cette approche permet notamment d'expliquer que des expressions référentielles différentes qui désignent la même entité n'aient pas le même rôle dans la compréhension. L'exemple le plus célèbre étant celui de Vénus, qui était désignée aussi comme « l'étoile du matin » et « l'étoile du soir » car apparaissant en premier le soir et disparaissant en dernier le matin. Les énoncés « Vénus est Vénus », « Vénus est l'étoile du matin », « L'étoile du matin est Vénus », « L'étoile du matin est l'étoile du soir » n'ont en effet pas la même interprétation, alors même que chacune des trois expressions référentielles « Vénus », « l'étoile du matin » et « l'étoile du soir » désigne le même objet, ce qui prouve que la signification d'une expression ne se réduit pas à ce qu'elle désigne.

Montague (1970) a développé le premier un modèle de sémantique intensionnelle en s'appuyant sur la logique du lambda-calcul (Church, 1941), et sur l'approche véricondi tionnaliste, en donnant les outils mathématiques permettant de construire la dénotation d'une phrase sous la forme d'une expression de vérité, en cohérence avec les travaux de Frege. Pavel Tichý, qui en proposa une formulation plus élégante quasiment simultanément (Materna, 1994), développa la logique intensionnelle transparente (Transparent Intensional Logic), dans laquelle

la signification est représentée par des constructions, c'est-à-dire des fonctions d'ordre supérieur permettant de construire une fonction finale de (w,t)[1] dans une valeur de vérité (Tichý, 1988).

Si Frege a ouvert la voie à la formalisation du *sens,* il a aussi énormément contribué à lier linguistique et logique, ce qui n'était pas nécessairement son objectif premier. Notamment, on peut se poser la question de savoir si un énoncé dénote effectivement un *sens*, dans l'optique de Frege, comme l'ont jugé Montague et les tenants de la thèse vériconditionnaliste. En admettant cette hypothèse si facilement, ils ont probablement réduit le champ d'investigation de la linguistique aux expressions logiques, là où il aurait peut-être fallu prendre du recul pour comprendre le rôle de la langue vis-à-vis des autres activités cognitives de l'être humain. C'est ce reproche qu'ont adressé au mouvement « logicien » Johnson-Laird and Miller (1976) ou Langacker (1987), en essayant de fonder l'étude de la langue davantage sur les perceptions et les aspects cognitifs de la communication que sur l'encodage et le décodage d'un message supposé représenter une formule logique qui serait dans les pensées du locuteur.

En contrepartie de cette prise de distance d'avec la logique, la linguistique cognitive n'a pas su prendre dans la formalisation logique les outils nécessaires pour donner à une théorie une intelligibilité et une validité suffisante pour être acceptée.

Pour conclure ce bref aperçu des considérations philosophiques, nous proposons ces définitions :

Sens d'un énoncé : actions correspondant à l'intention du locuteur ou de l'auteur, dépendant du contexte d'élocution.

Sens d'un mot : usage du mot dans une situation donnée (par exemple on peut parler du sens de « *bedeutung* » dans les écrits de Frege).

Signification d'une phrase : information qui, appliquée au contexte, produit le sens de l'énoncé dont on a induit la phrase.

Signification d'un mot/d'une expression : information qui permet de construire la signification de la phrase qui contient le mot/l'expression.

Dénotation : contenu *intensionnel* de ce qui est désigné par une phrase/une expression référentielle.

Référence en extension : ensemble des entités calculé par la dénotation appliquée à un contexte donné.

5.2 Référence linguistique

La définition du problème de la référence, tel qu'il est abordé par une grande partie des travaux en linguistique informatique, peut être résumée par cette citation :

> « The problem of establishing referential coherence in discourse can be rephrased as the problem of determining the proper antecedent of a given anaphoric expression in the current or the preceding utterance(s) and the rendering of both

[1] w pour le monde, t pour le temps.

> as referentially identical (coreferential). This task can be approached in a very principled way by stating general constraints on the grammatical compatibility of the expressions involved (e.g., Haddock 1987 ; Alshawi 1992). Linguists have devoted a lot of effort to identifying conclusive syntactic and semantic criteria to reach this goal, e.g., for intrasentential anaphora within the binding theory part of the theory of Government and Binding (Chomsky 1981), or for intersentential anaphora within the context of the Discourse Representation Theory (Kamp and Reyle 1993). » (Strube and Hahn, 1999)

Cette approche, que l'on appellera approche linguistique (car elle n'explore les problèmes que du point de vue de la langue, en occultant le rôle du contexte d'élocution), s'enracine dans une vision profondément informationnelle de la langue, où celle-ci est vue comme un médium de codage d'une information. Le problème de la compréhension du message se traduit alors par la reconstruction de la structure informationnelle initiale à partir de la forme aplatie. On rapprochera bien sûr cette approche du schéma de communication décrit par Jakobson and Pomorska (1983) qui modélise l'échange linguistique, qu'il soit verbal ou écrit, par six éléments : l'émetteur, le récepteur, le message, le code, le contexte et le contact. Le codage consiste alors à aplatir une représentation structurée complexe de façon à ce qu'elle puisse transiter par le canal vocal (on passe d'une représentation à N dimensions vers une représentation à une dimension). Le contexte joue alors le rôle d'une clef, nécessaire au décodage du message, et qui n'est pas contenu dans le message pour des raisons d'économie, puisqu'il est supposé connu de l'allocutaire, le récepteur du message.

Dans l'approche vériconditionnelle, le terme « référence » est traditionnellement relié, dans le domaine linguistique, à la résolution d'anaphores, le plus souvent pronominales ou plus généralement à la notion de coréférence dans le texte. Notre démarche orientée vers les agents assistants d'interface nécessite pourtant de considérer plutôt la référence comme acte consistant à relier une expression à une perception. En effet, dans les systèmes de dialogue, contrairement aux applications linguistiques orientées texte, le rôle de l'interface et donc du contexte extra-linguistique est prédominant. Si cette aspect de la référence n'a été que peu exploré jusqu'ici, c'est que :

1. il est principalement computationnel, c'est-à-dire que les heuristiques à mettre en œuvre pour sa résolution ne sont nécessaires que si l'on dispose de représentations que l'on doit effectivement manipuler, par exemple un objet auquel il faut appliquer une méthode,
2. c'est un problème trivial dans les cas restreints, notamment quand les types d'objets, leurs attributs et leur disposition est réductible à une représentation propositionnelle, ou lorsqu'ils peuvent être directement associés à des entrées lexicales.

Si l'on veut à terme pouvoir proposer un système qui puisse supporter un grand nombre de référents potentiels, comme c'est le cas pour un agent assistant d'interface, il est indispensable de disposer d'un modèle d'appariement efficace entre expressions référentielles et objets de l'application.

Nous avons cependant plusieurs arguments qui tendent à démontrer que l'approche extensionnelle de la référence, si elle est étudiée dans un modèle de représentation non propositionnel, peut être une fondation plus prometteuse pour la question de la compréhension de la langue y

compris pour les problèmes de résolution des pronoms anaphoriques. Malgré la différence majeure entre notre problématique et les questions traitées par l'approche coréférentielle, nous donnons un aperçu de cette approche avant d'exposer l'approche extensionnelle.

5.3 Approche coréférentielle

L'approche coréférentielle défini la résolution de la référence comme la recherche dans le discours passé de l'expression qui a introduit en premier un certain référent. Soit le texte suivant :

> « **Le président** a demandé au premier ministre de faire ses propositions sur la réforme de l'assurance maladie avant un mois. **Il** a aussi demandé à l'ensemble des ministres de mener une réflexion sur leurs méthodes de dialogue avec les différents partenaires sociaux. ».

L'expression « le président » n'est pas considérée comme une référence à une entité particulière dans la représentation du monde que pourrait avoir le lecteur. Le « il » de la deuxième phrase est en revanche considéré comme une référence à l'expression « le président ».

L'approche coréférentielle ne fait donc pas le lien entre la linguistique et le reste du monde. Cette démarche est principalement motivée par un objectif de recherche d'information, où les pronoms anaphoriques ne portent pas d'information et sont donc inutiles s'ils ne sont pas enrichis du contenu de leur antécédent.

5.3.1 Théorie du liage

La théorie du liage a été introduite dans le cadre de la grammaire générative nommée Government and Binding (GB) de Chomsky (1982), pour rendre compte des règles régissant la bonne formation des liaisons anaphoriques. Dans le cadre générale de GB, le liage se situe quasiment en bout de chaîne, juste après la structure de surface et avant la forme phonétique, montrant que dans l'approche génératiste de Chomsky, les pronoms et anaphores ne sont finalement considérés que comme des abréviations d'expressions référentielles complètes, ce qui peut être illustré par une transformation (Jean$_1$ plaint **Jean**$_1$ $\mapsto$ Jean$_1$ **se**$_1$ plaint).

Caractéristiques de l'approche générativiste chomskienne

Pour évaluer correctement la contribution de la théorie du liage dans la problématique de la référence, il nous faut tout d'abord situer la grammaire générative par rapport aux autres courants linguistiques. Les hypothèses de base de GB sont les suivantes :

- autonomie et primauté de la syntaxe
- approche lexicaliste
- modularisation et symétrie de l'analyse

Autonomie/Primauté de la syntaxe La question de l'autonomie de la syntaxe a été discutée au chapitre 2, notamment sur son aspect psychologique. Cette hypothèse impose de considérer le traitement syntaxique (dérivation de l'arbre syntaxique à partir de la phrase) comme autonome des autres niveaux d'interprétation (classiquement, la sémantique et la pragmatique), et donc de poser une séparation imperméable entre les informations à caractère syntaxique et les autres.

La conséquence de cette posture épistémologique est que la grammaire générative se donne comme objectif initial de vérifier la grammaticalité des énoncés (la notion de phrase bien formée et la primauté de la syntaxe) et non pas de chercher à construire une représentation de tout énoncé compréhensible. La justification du choix de cette approche pour aborder l'étude de la compétence de compréhension est que la reconnaissance de la grammaticalité d'un énoncé fait partie de la compétence linguistique, et donc que son étude doit mener à approcher la définition de la compétence linguistique, et aboutir à décrire la compréhension.

Approche lexicaliste La syntaxe étant considérée comme primale, et comme elle s'appuie sur les catégories syntaxiques des mots, il faut définir un dictionnaire associant les mots à leurs catégories grammaticales. Dans le cadre de la grammaire générative, ce sera le *lexique base*. Ce lexique est effectivement à la base de la séquence de modules qui composent la théorie du gouvernement et du liage (dernière émanation de la grammaire générative avant le programme minimaliste), puisque le premier étage de l'édifice, la théorie $\bar{X}$, décrit les règles de réécriture de formation des syntagmes à partir de leur catégorie grammaticale (par exemple SN →Det N).

Cette démarche impose de lier toute l'information sémantique aux symboles terminaux de la langue (les mots ou leurs sous-composants), et donc de considérer finalement qu'un système linguistique se réduit à des lexèmes et à des règles de bonne formation. Autrement dit, il n'est pas possible dans ce cadre d'affecter une information sémantique à une caractéristique structurelle de la langue (par exemple une répétition du même mot).

Modularité La dernière caractéristique majeure de la grammaire générative GB est son organisation modulaire et séquentielle. La construction de la forme logique et de la forme phonétique découle d'une suite d'applications de traitements indépendants les uns des autres. Cette démarche a pour conséquence de donner aux différents aspects de la théorie des structures très dissemblables, et par conséquent de construire un modèle de la langue hétérogène et dont les parties sont difficilement modifiables de manière indépendante, puisque les différents modules dépendent des frontières du module suivant et précédent dans la séquence de traitement.

Le liage

Pour en venir précisément à la question du liage, c'est-à-dire à l'établissement du lien entre une anaphore et son antécédent, la théorie édicte les contraintes qui doivent être respectées par les liaisons entre pronominaux et anaphores d'une part, et expressions référentielles (noms propres, phrases nominales « le chien qui court dans le jardin »). Ces contraintes s'appuient tout naturellement sur l'arbre syntaxique construit par les modules précédent de la séquence

générative. Particulièrement, deux notions construites à partir de cet arbre sont utilisées : la notion de c-commande, et la notion de liberté. La notion de liberté d'un élément signifie que cet élément ne peut être lié vers un autre, et donc qu'il est référent originel. La notion de c-commande correspond à un certain nombre de contraintes structurelles que nous ne détaillerons pas.

La théorie du liage donne les règles déterminant la validité d'une phrase contenant une anaphore. Par exemple, la théorie du liage permet de déduire que « *Martine*$_i$ *se*$_i$ *lève* » est valide, que « *Martine*$_i$ *arrive, elle*$_i$ *est en retard* » est valide, mais que « **Martine*$_i$ *arrive, se*$_i$ *est en retard* » ne l'est pas.

5.3.2 Théorie du liage dans HPSG

La grammaire HPSG (Head-Driven Phrase Structure Grammar) a marqué une étape très importante dans l'histoire de la linguistique car elle a permis d'exprimer de manière beaucoup plus compacte et unifiée les traitements définis dans les modules hétérogènes de la théorie GB. Elle se caractérise principalement par sa capacité à synthétiser diverses solutions émises dans des courants linguistiques variés dans un modèle élégant, et qui permet de prendre en compte un vaste champ de phénomènes linguistiques.

C'est tout d'abord une grammaire générative et lexicalisée, héritant de certaines caractéristiques de la théorie GB. HPSG se donne pour objectif de déterminer ce qui est ou pas une phrase bien formée et d'être capable de générer l'ensemble des phrases bien formées. Elle est lexicalisée, comme GB, chaque élément du lexique étant représenté par une structure de traits (appelée aussi matrice attribut-valeur) héritée principalement de la grammaire d'unification fonctionnelle (Functional Unification Grammar) donnant au lexique une expressivité beaucoup plus grande que le lexique-base de GB. C'est aussi avant tout une grammaire guidée par les principes et donc sans règle de transformation, la construction des structures s'opérant par unification de traits.

Du point de vue référentiel, HPSG s'inspire quasi exclusivement de la théorie du liage, mais en la modifiant de manière substantielle, avec principalement une adaptation des principes centraux du liage aux notions d'unification qui sont au centre de la théorie.

On ne peut donc pas considérer que la prise en compte de la référence se soit nettement améliorée au sein des grammaires de la famille générative depuis la théorie du liage introduite dans GB.

5.3.3 Théorie de la représentation du discours

La Théorie de la Représentation du Discours, plus connue sous l'acronyme DRT, a été proposée dans (Kamp and Reyle, 1993) comme une solution pour résoudre deux problèmes fortement liés que la sémantique de Montague (1970) ne permettait pas de traiter. D'autres théories, comme la sémantique des situations (Barwise and Perry, 1983), la théorie des changements de fichiers (Heim, 1982), ont été proposées dans la même optique.

Les problèmes non résolus par la sémantique de Montague sont : d'une part le traitement des anaphores inter-phrastiques, c'est-à-dire principalement les pronoms dont l'antécédent se trouve dans une phrase précédente. Et d'autre part, la prise en compte de la portée des

quantificateurs dans la relation anaphorique coréférentielle, permettant d'exprimer des liaisons autorisées lorsque les antécédents sont inclus dans une quantification. La DRT est donc une théorie de la représentation sémantique de la langue, inspirée de la logique des prédicats du premier ordre proposée par Russel, et qui permet d'intégrer la théorie de la quantification généralisée (Barwise and Cooper, 1981) tout en traitant les particularités et les différences entre résolution anaphorique inter-phrastique et intra-phrastique.

Dans la théorie vériconditionnaliste de la représentation sémantique, principalement formalisée par Montague d'après les travaux de Frege (1892), Russell (1952) et Carnap (1937), il est postulé qu'il existe une transformation directe de la structure syntaxique vers une formule de vérité. Les pronoms indéfinis se traduisent par l'application du quantificateur existentiel $\exists$ sur la variable représentant l'élément qui suit le pronom (les pronoms définis se traduisent par le même quantificateur existentiel mais avec une condition d'unicité supplémentaire). Or cette approche pose un problème dès l'instant qu'on produit une anaphore sur cet élément en dehors du bloc dans lequel l'élément a été introduit (soit une phrase précédente, soit dans une construction conditionnelle de type « Si ..., alors ... »).

(E2) *Si Jean possède un âne, il est content*

(F2) $\exists x/(ane(x) \wedge possede(jean, x)) \Rightarrow content(jean)$

(E3) *Si Jean possède un âne, il le bichonne*[2]

Comme (E2) et (E3) sont syntaxiquement identiques sauf pour le dernier groupe, la formule de vérité pour (E3) devrait être identique à la formule de vérité (F2) calculée à partir de (E2), sauf la sous-formule représentant le dernier groupe, soit la formule (F3).

(F3) $\exists x/(ane(x) \wedge possede(jean, x)) \Rightarrow bichonne(jean, x)$

(E4) *Pour tout âne que Jean possède, il le bichonne*

Or d'après Hess (1991), cette approche est problématique, car la formule (F3) n'est pas valide d'un point de vue logique, la variable x n'étant pas liée dans le contexte de l'implication. Deux solutions sont alors possibles : soit la portée du quantificateur existentiel est étendue, mais cela rompt avec le principe de compositionnalité, soit on ne traduit plus le pronom indéfini par un quantificateur *existentiel*, mais par un quantificateur *universel* ; cette seconde solution amène à traduire les expressions plus simples que les *donkey-sentences* par des expressions logiques assez éloignées de la sémantique initiale.

Pour résoudre ce problème, la DRT introduit une représentation intermédiaire entre la représentation syntaxique et la formule de vérité, avec la notion de structure de représentation du discours (DRS). Une DRS permet de représenter les variables évoquées par le discours ainsi que les contraintes qui portent dessus, et elle est enrichie au fur et à mesure de la « lecture » du discours. Un discours complet sera donc finalement représenté par une unique DRS contenant

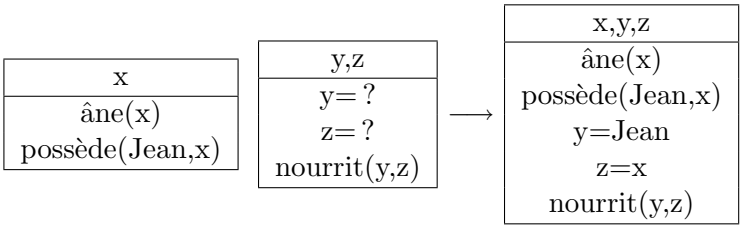
[2]Le lecteur coutumier des donkey-sentences aura remarqué que nous n'avons pas repris la thématique un peu trop violente et matérialiste qui caractérise les classiques exemples du domaine.

tous les éléments introduits durant le discours, et toutes les contraintes sur ces éléments. La sémantique du discours sera la formule de vérité dérivée de la DRS. Il faut noter que du fait qu'elle introduit un niveau intermédiaire entre la syntaxe et la sémantique, la DRT n'est pas une théorie compositionnelle (Amsili and Bras, 1998). De plus, en ayant une composante fortement dynamique (les phrases sont introduites dans l'ordre dans la DRS principale), la DRT rompt avec l'approche consistant à considérer que les phrases ont une sémantique statique. En cela elle a été inspirée par certaines approches cognitives (Fodor, 1990) qui considèrent que l'énoncé est une instruction adressée par le locuteur qui permet à celui qui la reçoit de se construire la représentation que le locuteur veut transmettre .

Dans une DRS, une anaphore est représentée par une égalité dont un des membres n'est pas lié. Par exemple :

(E5) *Jean possède un âne. Il le nourrit.*

(F5) $\exists x/(ane(x) \wedge possede(x))$



Spécification 9: « Jean possède un âne. Il le nourrit. »

L'avantage de passer par des DRS est que celles-ci permettent de décrire un intermédiaire entre la représentation syntaxique et la représentation en logique du premier ordre, en conservant uniquement les notions d'introduction d'un nouveau référent dans le discours et de contraintes sur les variables. Une fois la structure globale construite, on peut en dériver la représentation en logique du premier ordre. La valeur de vérité en DRT n'est donc plus restreinte au niveau de la phrase, mais s'étend à tout un discours.

Cette théorie, si elle permet de passer outre les limitations les plus évidentes de l'approche vériconditionnelle, bute cependant sur plusieurs points. Notamment, elle est incapable de traiter les métonymies, c'est-à-dire les cas où l'antécédent n'est pas explicitement introduit, mais peut être retrouvée à partir d'une représentation complète des éléments introduits. Par exemple dans (E6), la référence « les plus calmes » se base sur le fait qu'un laboratoire peut être considéré du point de vue des gens qui y travaillent, et donc que le fait de parler du laboratoire évoque parallèlement ces gens, qui peuvent devenir antécédents d'une anaphore future.

(E6) *Après l'annonce de la ministre, le laboratoire est en effervescence. Même* ***les plus calmes*** *sortent dans les couloirs pour discuter des actions à entreprendre.*

Il en est de même pour les références méronymiques, ou l'élément qui est introduit explicitement dans le discours évoque implicitement un élément qui englobe le premier (exemple (E7)).

(E7) *J'ai perdu une boucle d'oreille. J'aurais du penser à* ***les*** *enlever avant d'aller me baigner.*

D'autre part, les contraintes qui sont posées dans la DRT pour déterminer le bon antécédent parmi les variables introduites précédemment sont principalement des contraintes sur les portées et sur la syntaxe. Restant attaché à une modélisation propositionnelle des éléments du discours, il est difficile de traiter dans la DRT des caractéristiques continues comme la taille, le poids ou la couleur d'objets plus « réels ».

De manière plus générale, si la DRT prend en compte le contexte de l'énonciation d'une phrase, il ne s'agit que du contexte textuel statique. L'aspect dynamique de la DRT ne permet pas de traiter l'évolution des référents, mais simplement une construction séquentielle de la représentation sémantique d'un discours.

Utiliser la DRT pour représenter l'environnement de l'énonciation est réellement problématique si l'on veut traiter la référence dans un cadre non purement linguistique, car il faudrait une structure initiale contenant tous les éléments du contexte non textuel ainsi que toutes les contraintes qui s'y appliquent. Il est donc nécessaire de penser comment cette théorie peut être adaptée à un cadre de traitement plus général, notamment capable de gérer un support visuel, dans lequel, non allons le montrer, une représentation par prédicats logiques des relations entre différents éléments n'est pas suffisante pour rendre compte du comportement de certains prédicats référentiels.

5.4 Approche extensionnelle

Par opposition à l'approche coréférentielle, l'approche extensionnelle donne un rôle très important aux représentations extra-linguistiques. Du coup, si l'on reprend le même texte qu'en 5.3 :

> « **Le président** a demandé au premier ministre de faire ses propositions sur la réforme de l'assurance maladie avant un mois. **Il** a aussi demandé à l'ensemble des ministres de mener une réflexion sur leurs méthodes de dialogue avec les différents partenaires sociaux. »

La résolution extensionnelle s'attachera d'abord à relier l'expression « le président » à un élément de la représentation du monde. La résolution du pronom « il » prendra donc en compte le fait que l'élément à relier doit être cohérent avec la suite de la phrase (comme « il » est sujet de « demandé », il doit être relié à une personne ; si « le président » avait été relié à un camembert, il n'aurait pas pu coréférer avec « il »).

Pour le traitement des ellipses, il faut considérer que la résolution de la référence s'intéresse aussi bien à la représentation du monde qu'à la représentation de l'analyse linguistique. Dans ce cas, si une certaine structure syntaxique est créée par l'analyse d'une première phrase, les sous-parties reprises par l'ellipse sont considérées dans la même perspective que les représentations du monde, et donc peuvent être utilisées de la même manière que pour résoudre une anaphore ou une expression référentielle.

5.4.1 Théorie des représentations mentales

La théorie des Représentations Mentales (Reboul et al., 1997) s'est ancrée dans un projet de recherche portant sur la résolution de la référence dans le cadre de dialogues finalisés, thème se rapportant très nettement au sujet de cette thèse. La genèse de cette théorie rejoint en partie celle des approches dynamiques de la résolution de la référence comme la DRT, notamment pour l'héritage qu'ils partagent sur les référents du discours (Karttunen, 1976). La théorie des Représentations Mentales (RM) s'appuie aussi en partie sur une théorie dynamique différente de la DRT, la théorie des changements de fichiers (Heim, 1982) dont l'objectif était de rendre compte du problème des anaphore inter-phrastiques.

Dans la théorie de Karttunen, dont l'objectif avoué est de traiter la coréférence, il est proposé de mettre en place une sorte de mémoire des référents discursifs évoqués dans un texte et d'enrichir cette mémoire en fonction des informations nouvelles glanées au fil du texte. Cette mémoire sert ensuite pour traiter les anaphores, en considérant que les référents discursifs qui y sont stockés peuvent faire l'objet d'une reprise référentielle dans la suite du discours. Karttunen émet implicitement l'hypothèse que les expressions référentielles ne sont pas des ponts vers d'autres éléments du discours, mais vers les entités *imaginées* à partir de ces éléments. Cela n'empêche en rien de vouloir traiter de cette manière des coréférences dans des textes purs. De manière plus précise, Karttunen définit une théorie simple pour l'introduction des référents du discours dans la mémoire, dite théorie de la familiarité : une description indéfinie introduit un nouveau référent et une description définie va chercher un référent existant dans la mémoire.

Les travaux de Heim vont un peu plus loin dans la spécification de la mémoire des référents du discours en proposant de les structurer en fichiers, un nouveau fichier étant créé à chaque nouvelle phrase. Les fichiers contiennent les individus (sous forme de *cartes*) et les informations qui les concernent (des propositions sur les individus), mais aussi les relations entre les différentes cartes. De même que dans la DRT avec les structures de représentation du discours, l'évaluation du sens ne se fait plus en considérant la valeur de vérité d'une phrase, mais en considérant la valeur de vérité d'un fichier.

L'apport principal de Heim est de préciser l'idée de Karttunen en introduisant une sémantique claire de changement de fichier, chaque nouveau fichier étant créé à partir des précédents, en associant chaque forme logique avec un « potentiel de changement de fichier ».

La principale originalité des représentations mentales est de créer une représentation des référents qui contiennent non seulement les informations purement issues du discours, mais aussi celles provenant de connaissances extérieures. Cette facette est héritée en partie des travaux de Recanati (1993) sur les fichiers mentaux, qui portent sur la représentation des pensées singulières (pensées qui réfèrent à des objets uniques, par opposition aux pensées portant sur des généralités). Pour Recanati, la représentation des entités du monde est structurée selon différents points de vues, par exemple une description encyclopédique ou bien les caractéristiques perceptibles d'une entité. Ses fichiers mentaux sont une représentation des entités qui est identifiable au mode de présentation (c'est-à-dire à une manière de référer à une entité).

Les représentations mentales sont, dans cette lignée, des structures d'information qui déterminent un mode d'accès à une certaine entité. Elles sont composées de plusieurs entrées :

1. une **étiquette**,

2. une entrée **logique**, qui décrit les relations logiques et les relations d'appartenance entre les RM,
3. une entrée **encyclopédique**, qui contient toutes les informations d'ordre général sur l'entité, notamment les informations dues à son statut ontologique, ainsi que les informations qui lui sont propres mais qui ne sont pas décrites dans les autres entrées[3],
4. une entrée **visuelle**, à rapprocher des informations perceptives dans les fichiers mentaux de Recanati, qui contient les informations nécessaires à la reconnaissance de l'apparence de l'entité,
5. une entrée **spatiale**, qui contient les aspects spatiaux intrinsèques de l'entité (par exemple pour une voiture les orientations haut/bas, avant, dessus/dessous) ainsi que les relations spatiales que l'entité peut avoir avec d'autres RM,
6. une entrée **lexicale**, qui contient par exemple les expressions référentielles utilisables pour désigner l'entité (ce qui est plutôt en désaccord avec les idées de Recanati sur le fait qu'un fichier mental est lié à un mode de présentation et donc à une unique expression référentielle),
7. une entrée **d'identification**, entrée *ad hoc* pour un système de dialogue où la RM serait un pont vers un objet informatique connu de l'application commandée.

Cette approche consistant à regrouper les informations liées à une certaine entité permet de disposer d'une grande précision lors d'une résolution d'expression référentielle, y compris si celle-ci contient des informations spatiales ou si elle nécessite des connaissances sur les hyponymes ou hyperonymes. D'autre part, il est possible de créer une RM qui représente un groupe d'entités, permettant de traiter les références aux groupes en accédant à une seule RM.

La théorie des représentation mentales est une approche dynamique, les RM sont donc créées et modifiées tout au long du processus de « compréhension ». Les opérations définies sur les RM sont :

création une RM est créée chaque fois qu'une nouvelle entité fait son apparition, soit dans la modalité visuelle, soit dans la modalité verbale, soit encore lorsque son existence peut être inférée d'un événement intervenu dans une de ces deux modalités.

modification si de nouvelles informations portant sur une entité déjà identifiée apparaissent dans une des modalités, elles sont ajoutées dans les entrées correspondant à leur statut.

fusion deux RM sont fusionnées lorsqu'il est avéré qu'elle réfèrent à la même entité[4].

duplication c'est un cas très particulier, utilisé notamment pour modéliser la duplication d'entités (pour une photocopie, par exemple). La duplication n'est pas une simple recopie, puisqu'elle nécessite au moins de modifier les entrées étiquette et identification.

[3] Le statut un peu « fourre-tout » de cette entrée n'est pas très satisfaisant, le fait qu'elle soit définie négativement par rapport aux contenus des autres entrées fait qu'il est difficile de se faire une idée précise de son rôle dans la représentation des entités.

[4] Cette notion de fusion est très discutable, car les deux représentations correspondant à la même entité ne devraient pas disparaître. Comme le note Recanati, si je connais deux personnes, et que je me rend compte que ces deux personnes n'en sont qu'une, je peux toujours par la suite parler de chacun des deux « modes de présentations » que je possède (possédais) de cette personne.

groupement un groupement est créé lorsque plusieurs RM sont rapprochées dans une des modalités. Dans la modalité verbale par une coordination, par exemple. Dans la modalité visuelle, par proximité liée à des facteurs gestaltistes.

extraction permet de créer une ou plusieurs RM à partir d'une unique RM. Notamment utilisé pour traiter les méronymies.

La théorie des représentations mentales a été d'abord imaginée afin de permettre un traitement de la résolution extensionnelle de la référence (que nous présentons dans la section suivante). Sa première application réalisée par Popescu-Belis (1999) a pourtant été dédiée au traitement des anaphores, principalement parce qu'elle permet une représentation beaucoup plus riche que la DRT, et donc qu'elle permet de traiter certaines références dont l'antécédent n'est pas directement présent dans le discours, mais peut être construit à condition de disposer d'informations sémantiques et pragmatiques suffisantes.

L'autre intérêt de la théorie des représentations mentales est qu'elle reprend la notion de domaine de référence proposé par Corblin (1987), afin de rendre compte que certaines anaphores « recherchent » leur référent dans un ensemble forcément focalisé, c'est-à-dire dont un des éléments a été mis en exergue par une construction linguistique ou par tout autre moyen (notamment un geste).

En revanche, la théorie des représentations mentales ne se définit pas aisément par rapport à une théorie linguistique, contrairement à la DRT qui s'appuie sur l'approche de Montague. Cette situation implique selon nous qu'elle ne peut pas être intégrée sur le long terme dans une démarche où elle pourrait être mise à l'épreuve de différents phénomènes.

5.4.2 Domaines de référence

Nous définissons la résolution extensionnelle de la référence comme le processus consistant à sélectionner les entités désignées par une expression référentielle dans une situation donnée (typiquement déterminée par des objets présents dans l'environnement de l'élocution), mettant donc en jeu des aspects pragmatique du traitement de la langue. Les approches logiques telles que la DRT (Kamp and Reyle, 1993) se révélant inadaptées ou inapplicables dans certains cas (prédicats vagues, métaphores, groupements perceptuels, glissements typologiques, relations spatiales), d'autres voies ont été explorées, utilisant des classes de comparaison (Kyburg and Morreau, 2000) pour les prédicats vagues, les cadres (Schang, 1995) pour les relations spatiales, les représentations mentales (Reboul, 1999) pour les métonymies, les domaines de référence (Salmon-Alt, 2000) pour la reprise nominale, et les extracteurs référentiels différentiels pour les prédicats vagues et relationnels (Pitel and Sansonnet, 2003a).

Les domaines de références (Corblin, 1987; Reboul et al., 1997; Salmon-Alt, 2001) sont définis comme des « *ensembles contextuels locaux structurés de façon à prédire la distribution des différents marqueurs référentiels* » (Salmon-Alt, 2001). Ce modèle permet de traiter des phénomènes concernant la référence anaphorique. Le fait de représenter les différentes étapes de sélection par des ensembles structurés permet de rendre compte de certains phénomènes de préférence dans la reprise nominale.

Les domaines de référence permettent d'organiser des entités en fonction de leurs relations entre elles selon un critère de différenciation (par exemple bleus/non-bleus), ce qui, de fait, le partitionne entre entités accordées au prédicat, et entités accordées à sa négation. Ceci permet

de distinguer les entités potentiellement candidates à l'expression référentielle des autres. Dans (Pitel and Sansonnet, 2003a) le critère de différenciation est remplacé par une fonction de calcul de similarité, ceci afin de pouvoir traiter les prédicats relationnels comme *« grand »*, *« lourd »*, etc. en permettant de trier les entités en fonction de leur relations selon le prédicat référentiel. Cette fonction ne permet que de créer une partition entre candidats possibles et totalement impossibles (à exclure des étapes suivantes), mais pour gérer les candidats « préférés » afin de traiter les références plurielles (« les gros cubes »), une fonction de partition supplémentaire est nécessaire, qui doit prendre le contexte en compte. La représentation d'un prédicat référentiel nécessite alors trois fonctions. Ce point est abordé en détail dans le chapitre 7.

5.4.3 Référence spatiale

La référence spatiale est un domaine particulièrement difficile à rapprocher de l'analyse linguistique, car elle mélange allègrement perception, pertinence et constructions linguistiques (Zock and Briffault, 1995), aussi est-elle généralement traitée par des heuristiques indépendantes (Schang, 1997; Briffault, 1992).

Un des problèmes posé par les prépositions spatiales est qu'elles ne peuvent être traitées par la logique du premier ordre. En effet, être « *à droite* » d'un objet, par exemple, n'est pas une propriété purement binaire, vraie ou fausse. Comme l'illustre la figure 5.1, pour résoudre l'expression référentielle « le rond à droite du carré », il faut prendre en compte la distance entre les objets candidats et le référentiel (le carré X), la distance par rapport à l'axe horizontal, etc. Or si on considère le prédicat « être à droite de » uniquement sous son aspect logique, le rond en haut dans la figure 5.1 pourrait être un aussi bon, voire meilleur, candidat que le rond entouré d'un trait en pointillés, ce qui n'est pas le cas dans les expérimentations.

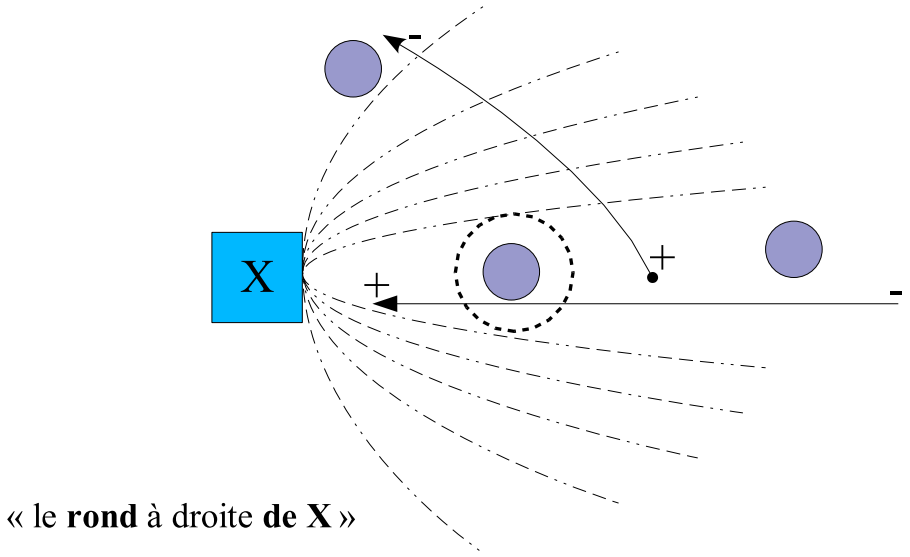


FIG. 5.1 – Caractéristiques quantitatives en deux dimensions du prédicat référentiel spatial « *à droite de* ».

Une autre caractéristique importante des prédicats spatiaux est leur comportement inhabituel vis-à-vis de la compositionnalité. En effet, alors que pour un prédicat comme « grand » ou « à droite », les modifieurs comme « plus » ou « moins » agissent dans le même sens que la sélection, c'est l'inverse qui se passe pour les prédicats comme « à droite **de** ». Ceci est illustré sur les figures 5.2 et 5.3, qui montrent la répartition des candidats pour les différents prédicats modifiés ou non. on voit que pour « à droite », le modifieur « plus » ne change pas le sens de la répartition, alors que pour « à droite de » qui sélectionne les objets proches de la position de référence (ici la position de X), le modifieur « plus » implique que l'on revient au sens de « à droite » (sans position de référence). Ceci semble être un indice assez fort que le sens des prédicats référentiels ne peut pas toujours être calculé avec une approche compositionnelle à la Frege, mais qu'une approche constructionnelle est nécessaire.

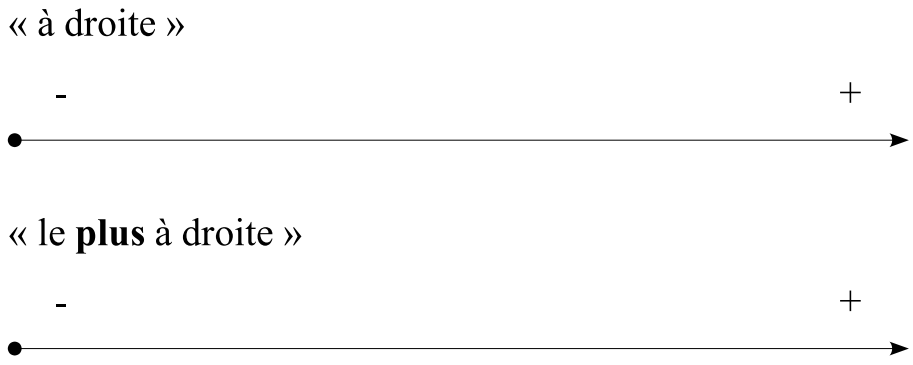


FIG. 5.2 – Comportement du prédicat « **à droite** »

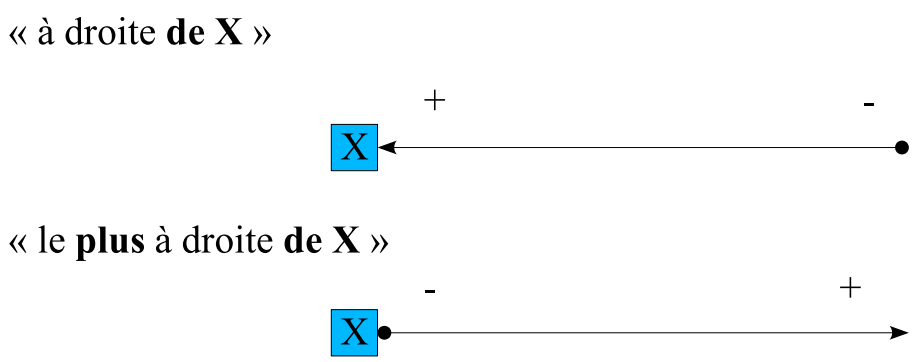


FIG. 5.3 – Comportement du prédicat « **à droite de** »

La figure 5.4, illustre différentes situations où la prise en compte des informations pragmatiques portées par le support visuel est primordial pour l'analyse syntaxique, autrement dit de la désambiguïsation par la pragmatique. En effet, le même énoncé va être interprété différemment, à la fois du point de vue syntaxique et du point de vue référentiel, en fonction du

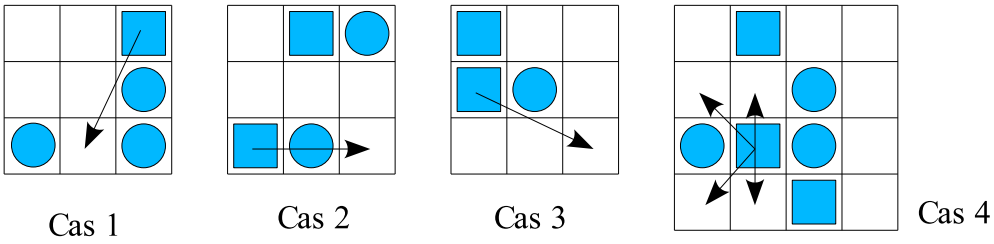


FIG. 5.4 – Situations différentes amenant des interprétations différentes de l'énoncé « Déplace le carré à gauche du rond en bas à droite »

contexte dans lequel il est interprété.

Pour réussir à analyser correctement l'énoncé « *Déplace le carré à gauche du rond en bas à droite* » dans les différents cas de figure présentés figure 5.4, il faut d'une part choisir entre ces différentes segmentation syntaxiques [5] :

1. « Déplace | le carré | à gauche du rond en bas à droite » (cas 1),
2. « Déplace | le carré à gauche du rond | en bas à droite » (cas 2),
3. « Déplace | le carré à gauche du rond en bas | à droite » (cas 3),
4. « Déplace | le carré à gauche du rond en bas à droite » (cas 4).

D'autre part, il faut que le référent repère, c'est-à-dire le point fixe par rapport auquel le déplacement sera effectué, soit résolvable s'il est disponible. Enfin, il faut que l'élément à déplacer puisse être résolu, et que le mouvement à effectuer soit réalisable, c'est-à-dire qu'il existe un passage libre entre le point de départ et d'arrivée. Toutes ces informations peuvent (et selon nous, doivent) être prises en compte relativement tôt dans l'interprétation, sans nécessairement chercher d'abord à trouver toutes les analyses syntaxiques possibles.

En effet, si l'on découpe d'abord l'expression en blocs indivisibles (le, carré, à gauche, du rond, en bas, à droite), on peut très facilement à partir de la situation vérifier que certains blocs ne peuvent pas être résolus seul et doivent donc être combinés. Ainsi, « du rond » ne peut être résolut isolément que s'il n'y a qu'un rond dans la situation. Une fois cette première analyse effectuée, on peut savoir quels segments sont compatibles avec la situation, et donc guider rapidement l'analyse vers la solution qui s'adapte à la situation.

5.4.4 Résolution par la théorie de la pertinence

La théorie de la pertinence de Sperber and Wilson (1995) permet de décrire les caractéristiques supérieures de la communication, en stipulant certaines conditions pour qu'un énoncé soit pertinent vis-à-vis d'une situation donnée. Landragin (2003) a proposé de définir le processus

[5] Cet exemple est bien entendu construit de toutes pièces, et ne pose probablement pas de problème dans une situation de dialogue humain-humain normale, la prosodie ayant de forte chances d'aider grandement la segmentation correcte. Cependant, si l'on présente cet énoncé dans sa version textuelle à des sujets humains mis face aux différentes situations, ceux-ci trouvent sans problème l'interprétation correcte rapidement.

de résolution référentiel dans le cadre d'un dialogue avec support visuel et gestes de pointage par rapport à cette notion de pertinence, en décrivant comment la mise en relation de ce qu'expose l'utilisateur et les hypothèses formées par la théorie de la pertinence permettent de sélectionner plus efficacement les bons objets.

Cette approche met particulièrement en avant l'importance des indices perceptifs non linguistiques dans le phénomène référentiel dans les situations où l'environnement du dialogue ne peut être ignoré, et par conséquent, l'importance de la prise en compte de l'environnement dans l'interprétation des énoncés, principalement vis-à-vis des critères spatiaux et en termes de proximité entre les critères qui sont affectés par un humain aux objets perçus.

Conclusion

Dans ce chapitre, nous avons abordé l'approche coréférentielle de la référence, en présentant notamment la théorie du liage de Chomsky et la théorie de la représentation du discours, qui nous semblent bien représentatifs de cette approche. Nous avons ensuite étudié les différentes voies qui cherchent davantage à faire le lien entre les expressions référentielles et les objets dans une représentation du monde. Cette approche, que nous caractérisons par le terme d'approche extensionnelle, englobe aussi bien la résolution d'expressions référentielles spatiales dans un contexte avec support visuel, que la résolution de références temporelles dans un contexte de dialogue avec évolution dynamique.

Cette étude partielle des deux grandes différentes approches de la résolution de la référence, qui sont l'approche coréférentielle et l'approche extensionnelle, nous a permis de constater que l'approche coréférentielle n'était pas la mieux adaptée à la problématique des agents assistants d'interface. Nous allons donc, dans la suite, nous concentrer sur l'aspect extensionnel de la résolution de la référence, en nous inspirant notamment des travaux sur la théorie des représentations mentales et du modèle des domaines de référence appliqué au dialogue finalisé.

Pour préciser davantage notre champ d'étude, nous nous intéressons dans le chapitre suivant à caractériser les différents types d'expressions référentielles, et à determiner le rôle qu'elles jouent dans le processus d'interprétation des énoncés. Nous pouvons ainsi préciser la catégorie des expressions référentielles que nous pouvons envisager de traiter dans le cadre d'une approche extensionnelle.

Chapitre 6

Etude des expressions référentielles

6.1 Rôles des expressions référentielles

Nous nous plaçons dans un cadre d'interprétation en contexte (c'est-à-dire que l'on considère par défaut que le contexte dans lequel un énoncé est prononcé est accessible au système d'interprétation) et dans une perspective où le « *sens* » que l'on veut calculer à partir d'un énoncé est l'*action* que le locuteur à l'*intention de faire exécuter* par l'allocutaire. Nous considérons alors la manière dont peut être utilisée une expression référentielle (ER) en situation, en admettant qu'un énoncé « appelle » une certaine action de la part de celui qui le reçoit, et que cette action implique d'utiliser certains aspects (et seulement certains) des éléments qui composent l'énoncé. Selon les énoncés, la même expression référentielle peut ainsi se traduire de manières différentes dans l'utilisation qui en est faite pour construire l'action appelée. Par exemple, les phrases suivantes contiennent toutes la même ER : « un mouton ». Nous allons présenter des analyses des différents rôles qui, selon nous, sont joués par cette même ER en fonction de l'action à réaliser en réponse à la réception de chaque énoncé.

(E8) *J'ai acheté un mouton*

L'énoncé (E8) transmet directement une information. Dans une situation neutre, dans laquelle ni le locuteur, ni le récepteur n'ont de lien particulier avec les moutons, c'est un énoncé qui n'amène aucune réponse particulière. Le rôle de l'ER est réduit à sa plus simple expression, quasiment celui d'un signe quelconque. A peu de choses près, on pourrait remplacer « mouton » par « espalan », cela n'empêcherait pas nécessairement que le récepteur puisse rapporter que « un-tel a acheté un espalan », même si un espalan n'existe pas (à ma connaissance). La curiosité naturelle des gens l'emportant généralement, on peut supposer qu'utiliser un terme inconnu amène tout de même l'interrogation « Qu'est-ce qu'un espalan ? », car le fait de ne pas pouvoir se faire une image des choses dont ont parle est plutôt gênant. Cependant, aucune action autre que la mémorisation n'étant appelée directement par l'énoncé, la connaissance de l'image associée au mouton n'est pas *nécessaire* à l'utilisation ultérieure de cette information.

(E9) *Ceci est un mouton.*

L'énoncé (E9) établit un lien entre le signe « mouton » et l'élément désigné par « ceci ». L'action appelée par l'énoncé consiste là encore en une mémorisation, mais cette fois-ci deux cas peuvent se présenter. Soit l'allocutaire a déjà associé le signe à une forme ou à une définition (qui lui permet de créer une image d'un mouton), auquel cas il n'acceptera cet énoncé que si ce qu'il perçoit est bien compatible avec l'image que lui évoque le signe « mouton » (autrement dit s'il reconnaît dans son observation un élément qu'il peut désigner par ce signe), soit il n'a associé aucune information à ce signe, auquel cas il doit créer une nouvelle association entre le signe et ce qu'il perçoit. Dans le premier cas, il peut ne pas être d'accord, et ce pour deux raisons : soit ce qu'il observe évoque quelque chose d'autre chez lui, soit ce qu'il connaît des moutons n'est pas compatible avec ce qu'il perçoit, autrement dit, il n'est pas capable de reconnaître dans ce qu'il perçoit une chose qu'il peut désigner avec le signe « mouton ». Dans ce dernier cas, l'ER « un mouton » est utilisée comme **opérateur de reconnaissance**.

(E10) *Dessine-moi un mouton*

Dans l'énoncé (E10), l'action appelée consiste à produire une figuration de l'image que se fait le locuteur du signe « mouton ». Pour ce faire, l'allocutaire doit utiliser sa compétence à produire une image (mentale ?) à partir du signe, et à représenter figurativement l'image qu'il a produite. En effet, la seule mise en œuvre de la capacité à reconnaître un élément et à lui associer un signe ne permet pas de la produire, ni figurativement, ni même linguistiquement. On trouve un parallèle à cette analyse dans le fait qu'être capable de comprendre un énoncé ou de reconnaître un élément ou un état n'est pas équivalent à être capable de produire ce même énoncé ou à figurer ou décrire ce même état. On l'observe notamment dans l'asymétrie entre compréhension et production lors de l'apprentissage d'une langue : les enfants comprennent avant de savoir parler, et on sait toujours mieux lire qu'écrire dans une langue étrangère. Le rôle de l'ER « un mouton » est donc ici d'être un **opérateur de production**.

(E11) *Montre-moi un mouton (dans ce pré)*

L'énoncé (E11), qui a pourtant exactement la même structure syntaxique que le (E10), appelle un autre usage du GN, car c'est clairement le rôle de reconnaissance qui est mis à contribution pour réaliser l'action demandée. En effet, l'allocutaire doit utiliser l'information « un mouton » pour trouver dans son champ de perception un élément qu'il peut reconnaître comme associable au signe « mouton ».

(E12) *Est-ce que c'est un mouton ?*

Dans l'énoncé (E12), c'est le rôle d'opérateur de reconnaissance (de mouton) qui est mis en action par la réalisation de l'action appelée par l'énoncé. L'allocutaire doit observer un élément précis (désigné ou mis en avant par effet de saillance) de son champ de perception et vérifier qu'il peut bien être vu comme « un mouton ».

(E13) *Qu'est-ce qu'un mouton ?*

L'énoncé (E13) est une demande de définition, c'est-à-dire que l'allocutaire doit produire une suite d'informations qui permette au locuteur de (E13) de se faire une « idée » ou une image de ce qui peut être associé à un mouton. C'est donc comme dans (E10) le rôle d'opérateur de production qui est appelé par cet énoncé, mais cette fois-ci, une production verbale peut être acceptée alors qu'une production figurative était obligatoire dans (E10).

On obtient donc finalement trois types de rôles pour les ER, le rôle de signe, le rôle d'opérateur de reconnaissance, et le rôle d'opérateur de production. Dans le tableau 6.1, nous avons récapitulé les rôles de l'expression référentielle pour chaque énoncé. Nous y avons adjoint un rôle secondaire, dont nous estimons qu'il peut être amené à apparaître en tant que produit dérivé de l'action (par exemple pour (E8), il peut être nécessaire pour l'allocutaire de produire une image du mouton pour inférer certaines informations à partir de l'énoncé brut), mais qui sont selon nous inutiles à la réalisation immédiate de l'action attendue par l'illocuteur.

Enoncé	Rôle primaire	Rôle(s) secondaire(s)
(E8) : « J'ai acheté un mouton »	signe	op. production
(E9) : « Ceci est un mouton »	signe	op. reconnaissance
(E10) : « Dessine-moi un mouton »	op. production	op. reconnaissance
(E11) : « Montre-moi un mouton »	op. reconnaissance	-
(E12) : « Est-ce que c'est un mouton ? »	op. reconnaissance	-
(E13) : « Qu'est-ce qu'un mouton ? »	op. production	-

TAB. 6.1 – Rôles de l'expression référentielle dans la construction d'une action répondant à l'énoncé.

Cette catégorisation des expressions référentielles nous permet de préciser ce que nous attendons comme contenu dans le produit de l'interprétation des expressions référentielles, et donc la représentation *in fine* de l'expression référentielle.

Cette méthode nous permet d'ores et déjà de restreindre notre champ d'investigation à une catégorie précise, à savoir les caractéristiques des expressions référentielles du point de vue de leur fonction d'opérateur de reconnaissance. C'est dans cette optique que nous allons étudier dans la section suivante les différents éléments qui composent les expressions référentielles.

6.2 Eléments des expressions référentielles

Les expressions référentielles sont des assemblages de mots (qui sont souvent appelés prédicats, car ils expriment une vérité sur l'élément désigné) qui indiquent comment trouver le ou les éléments référencés ou encore comment produire une image (au sens large, et donc pouvant inclure des modalités autres que la vision) qui peut être cible de la référence. Nous nous intéresserons uniquement dans cette analyse à l'aspect « opérateur de reconnaissance », et considérerons ces mots qui composent les expressions référentielles comme des opérateurs permettant de reconnaître certains éléments. Nous étendons cette notion de reconnaissance à la notion de sélection d'un ensemble d'éléments candidats, et adoptons à la place du terme *prédicat* le terme *extracteur référentiel*. Ce terme est à mettre en relation avec l'opération que nous supposons faite par le processus de résolution extensionnelle d'une référence, qui

consiste à extraire le ou les bons éléments à partir d'une représentation comportant plusieurs candidats potentiels.

Une expressions référentielle couramment rencontrée est composée d'un déterminant, d'un ou plusieurs adjectifs, et d'un nom. S'y rajoutent assez souvent des syntagmes prépositionnels, adjectivaux ou adverbiaux, ainsi que des groupes subordonnés. L'établissement d'une typologie des prédicats est motivée par deux objectifs principaux : tout d'abord tenter d'être précis dans la couverture des expressions référentielles, c'est-à-dire savoir ce que l'on peut gérer et ce que l'on ne peut pas, ensuite hiérarchiser les expressions référentielles afin de les ordonner éventuellement dans le processus de résolution. Il s'agit donc soit d'extraire les prédicats des expressions référentielles et de les caractériser en fonction d'une méthodologie formellement définie afin de les organiser en classes, soit de se doter d'un ensemble de catégories choisies et répartir les différents prédicats dans ces catégories.

Peu d'études ont été menées sur ce sujet précis, l'une des plus notable est d'ailleurs issue de travaux portant sur des problématiques de génération d'expressions référentielles (Dale and Reiter, 1995), et non pas sur leur résolution. La raison de ce faible intérêt vient selon nous du fait que dans une optique d'interprétation des expressions référentielles, il n'est pas très problématique de considérer simplement que la sélection des éléments désignés par l'expression référentielle se fait par filtrage propositionnel des variables (qui représentent les éléments accessibles) en utilisant les prédicats logiques (déterminés à partir des adjectifs et des noms qui la composent). L'importance relative des prédicats les uns par rapports aux autres n'a alors pas d'influence sur le résultat du filtrage, puisque la seule opération consiste à éliminer les variables qui ne correspondent pas aux prédicats. Ce que nous appelons filtrage propositionnel immédiat, se décrit de la manière suivante : si les entités candidates sont représentées par trois objets (ou variables) x, y, z ayant pour prédicats $rouge(x)$, $vert(y)$, $rouge(z)$, $carre(x)$, $triangle(y)$, $cercle(z)$, alors résoudre l'expression référentielle « le cercle rouge » consistera simplement à trouver k tel que $rouge(k) \wedge cercle(k)$, qui est strictement équivalent à $cercle(k) \wedge rouge(k)$.

Dans le cas de la génération d'expressions référentielles, il est nécessaire de vérifier l'adéquation cognitive entre l'expression produite et ce qui est attendu par un auditeur humain, or appliquer l'approche propositionnelle à l'envers conduit à des résultats très mauvais, qui accumulent les informations inutiles. Ceci implique que les approches utilisées en génération doivent sélectionner les prédicats à utiliser, et pour cela ne pas utiliser tous les prédicats de manière indifférente. Notamment, Dale et Reiter ont établi une distinction entre prédicats typologiques, prédicats intrinsèques, et prédicats relationnels. Ils ont aussi établi un ordre préférentiel entre les types de prédicats qui leur permet de calculer les prédicats les plus discriminants d'un point de vue cognitivement plausible.

D'autres problèmes se posent quant à la normalisation du sens des prédicats. En effet, parmi les types de prédicats dont il est connu que le sens dépend de l'élément sur lequel ils s'appliquent, les adjectifs de taille (grand garçon, petit mammouth) sont particulièrement étudiés quant à leur caractère « vague », terme auquel nous préférons le qualificatif de « dépendant du contexte » (suivant en cela Bosch (1983)). La plupart des travaux sur l'imprécision n'ont pas un caractère applicatif, et la plupart ont comme principal objectif de déterminer, pour une phrase possédant un attribut vague, sa contribution au contenu propositionnel en termes de sémantique vériconditionnelle. Pourtant la prise en compte des prédicats vagues permet de poser certaines contraintes sur la sémantique de nos *extracteurs référentiels*, les rendant

ainsi suffisamment génériques pour pouvoir supporter aussi les caractères vagues.

6.2.1 Notion d'extracteur référentiel

Nous appellerons **extracteurs référentiels** les fonctions qui permettent de représenter le rôle des termes d'une expression référentielle qui permettent la construction de l'extension de l'expression. Le terme d'extracteur est choisi pour mettre en exergue le fait que ces fonctions vont sélectionner certains éléments dans l'ensemble des éléments disponibles (un domaine de référence). Le terme d'extracteur référentiel regroupe fonctions attribuables aux noms, adjectifs ou prépositions utilisés dans les expressions référentielles pour sélectionner tel ou tel élément de la représentation du monde. Le rôle d'extracteur n'est pas le seul qui puisse être attribués à ces terme, on peut l'illustrer à partir de l'étude des trois cas suivants :

1. « Efface le carré **bleu**. »
2. « Crée un carré **bleu**. »
3. « Est-ce que le carré est **bleu** ? »

Dans le premier cas, qui est le cas qui nous intéresse ici, « *bleu* » est utilisé pour extraire les entités du monde pouvant être vues comme étant bleues. Dans le second cas, « *bleu* » est utilisé pour « produire » une couleur bleue. Dans le troisième cas, « *bleu* » est utilisé pour vérifier qu'un objet peut être vu comme étant bleu.

Pour le moment, nous nous plaçons donc dans le cadre illustré par le premier cas, et nous nous intéresserons à pouvoir représenter le pouvoir de sélection des extracteurs.

6.2.2 Catégories syntaxiques des extracteurs

Dans un premier temps, nous allons aborder les différents types d'extracteurs d'après leur catégorie syntaxique. Cette courte étude ne prétend pas mener une étude exhaustive sur les différentes composants des expressions référentielles. Une telle étude demanderait de se lancer dans une démarche de linguiste que nous ne sommes pas en mesure de mener à bien. Notre objectif ici est juste de montrer comment nous comptons caractériser les différents types d'éléments en fonction d'une approche ensembliste de la résolution de la référence.

Articles Les articles, définis ou indéfinis, n'effectuent aucun ordonnancement, mais simplement une partition. Par exemple pour « le gros carré » on considère que l'ensemble des éléments a d'abord été trié et partitionné par l'extracteur associé à « carré », puis par « gros ». L'opération effectuée par l'extracteur associé à « le » est donc limitée à partitionner le domaine créé par les deux précédents extracteurs, en isolant l'élément le plus haut classé. Si cette isolation échoue (par exemple si plusieurs éléments sont à égalité), alors l'extraction échoue.

Adjectifs Qualificatifs Les adjectifs qualificatifs permettent de créer une distinction en fonction de certaines qualités des éléments. Leurs rôles en tant qu'extracteurs référentiels revient à sélectionner les éléments qui peuvent être qualifiés de ces qualités ou non. Il faut

cependant faire attention au fait que ces qualités ne sont pas nécessairement liées aux éléments de façon absolue et définitive. Notamment, les adjectifs vagues portent sur des qualités qui doivent être considérées en contexte, voire en fonction du reste de l'expression référentielle dont ils font partie (c'est d'ailleurs l'un des points déterminants de notre approche de la résolution extensionnelle de la référence).

Cardinaux Les adjectifs cardinaux peuvent soit exprimer une quantité (« Donnez-m'en deux kilos »), soit un dénombrement (« Donnez-moi deux bonbons »). Dans les cas qui nous intéressent, où l'expression référentielle joue un rôle d'opération de reconnaissance (par exemple dans « Supprime les trois messages récents »), l'adjectif cardinal combiné à l'article permet selon nous de sélectionner les n éléments les plus hauts classés par la sous expression référentielle immédiatement supérieure (dans ce cas « les messages récents »).

Ordinaux Les adjectifs ordinaux fonctionnent de manière moins homogène. « second », « deuxième », etc. ne permettent que de sélectionner un élément dans une liste ordonnée (ou plusieurs éléments à égalité dans le tri, ce qui permet de dire éventuellement « les troisièmes »). Par contre, « premier » et « dernier » imposent un ré-ordonnancement (selon un critère implicité par la sous-expression référentielle). Ce qui permet de dire « Supprime les trois derniers messages », le critère implicité pour ordonner les messages étant un ordre d'arrivée dans le temps, et « dernier » inversant le tri par rapport à « premier ».

Noms Fondamentalement, le rôle des noms est très proche du rôle des adjectifs qualificatifs, à ceci près qu'il est nécessaire de considérer la catégorie des objets comme une de leur qualité, ce qui n'est pas couramment fait. De plus, cette qualité ne doit pas être fixée de manière unique, car un même objet peut être catégorisé de différentes manière en fonction de la situation. Par conséquent, nous choisissons de considérer les noms de la même manière que les adjectifs, car ainsi ils peuvent être exprimés d'une façon unifiée. De plus, il existe une relation forte entre cette approche et notre théorie des Points de vue, puisque l'adéquation d'un élément à un prédicat de type nom correspond au fait que l'élément *puisse être vu* comme un élément du type véhiculé par le nom.

6.2.3 Modes d'accès des extracteurs

Il semble d'après ces observations que s'appuyer sur la catégorie syntaxique des composants d'une expression référentielle ne soit pas la manière la plus adéquate de les étudier. En effet, certains extracteurs fonctionnent différemment à l'intérieur de la même catégorie (les cardinaux), alors que d'autres semble pouvoir fonctionner d'après un même principe (les adjectifs et les noms).

Ceci nous amène à nous intéresser davantage aux informations auxquelles accèdent les extracteurs, car dans les travaux récents sur la résolution de la référence dans le dialogue finalisé, le problème de la représentation des éléments de l'application médiée et de leur lien avec les expressions référentielles qui les désignent est généralement considéré comme négligeable par rapport au traitement unifié des divers phénomènes de référence anaphoriques. Ainsi Salmon-Alt (2001), s'intéressant aux différences de comportement entre plusieurs types de références,

donne la représentation des entités de l'application (et par rapprochement, du processus de sélection des entités) en faisant le présupposé que : « *Pour une propriété intrinsèque par exemple, les éléments du domaine de référence doivent posséder cette propriété.* » (Salmon-Alt, 2001, p. 150). Cette supposition amène à une représentation *propositionnelle* des entités, et à une sélection des entités par prédicats logiques du premier ordre. Cette approche est largement répandue dans les travaux proposant des formalismes pour le traitement de la référence et plus généralement dans les travaux traitant de la représentation des connaissances. Ainsi Byron and Allen (2002) expriment les champs de restriction par des propositions de la forme (color x RED), et Dale and Reiter (1995) définissent les objets du monde pour la génération de la référence selon le formalisme présenté tableau 6.2. Ces formalismes sont fortement reliés aux notions de *frames* et leur utilisation dans la modélisation de la langue, provenant des travaux de Minsky (1975) et Charniak (1978).

Object1 : ⟨size, large⟩, ⟨colour, red⟩, ⟨material, plastic⟩
Object2 : ⟨size, small⟩, ⟨colour, red⟩, ⟨material, plastic⟩
Object3 : ⟨size, small⟩, ⟨colour, red⟩, ⟨material, paper⟩
Object4 : ⟨size, medium⟩, ⟨colour, red⟩, ⟨material, paper⟩

TAB. 6.2 – Formalisme de représentation des objets dans (Dale et Reiter, 1995)

Dale and Reiter (1995) distinguent trois types de composantes pour caractériser un élément :

– Le type des objets (p. ex. triangle, carré),
– Les propriétés intrinsèques (p. ex. taille, couleur),
– Les propriétés relationnelles (p. ex. position spatiale).

Salmon-Alt explique le lien entre la description et l'entité par l'opération suivante (ici dans le cas d'une expression indéfinie) : « *A l'intérieur de la nouvelle partition*[1]*, l'indéfini recrute son référent par l'extraction d'un élément quelconque qui sera distingué des autres éléments par un critère de différenciation reposant sur des informations prédicatives.* » (Salmon-Alt, 2001, p. 149). Un exemple de cette approche est la résolution de l'expression référentielle « *le grand carré bleu* », sélectionnant les entités de l'application de type carré et répondant aux prédicats *grand(x)* et *bleu(x)*.

Mais qu'en est-il si l'on a une représentation telle que celle-ci, qui décrit bien un carré bleu, mais en décrivant ses quatre côtés comme des segments quelconques, et sa couleur d'après les composantes cyan, jaune et magenta ?

Polygone
couleur : {cyan :0.8, jaune : 0.2, magenta : 0.1}
points : {(0.,0.),(1.0,0.0),(1.0,1.0),(0.0,1.0)}

Cet exemple montre la limite des approches fondées sur l'utilisation de prédicats, à la fois pour les caractéristiques typologiques (un carré est aussi bien un polygone, alors faut-il absolument lui attribuer un type « naturel » ?) et pour ses qualités (la couleur, la taille, peuvent être

[1]Une partition sépare la liste des candidats et non candidats à la référence en fonction d'un critère de différenciation, par exemple la couleur bleue.

exprimées de différentes façons, y en a-t-il une qui soit meilleure que les autres ? doit-on obligatoirement choisir, pour une application de dialogue donnée, un mode de représentation unique qui soit nécessairement adapté à une description propositionnelle ?).

6.2.4 Extracteurs intrinsèques

Les extracteurs intrinsèques s'appuient sur les caractéristiques intrinsèques, qui sont des caractères stables (ils sont toujours présents) et indépendants du contexte. Typiquement, les caractères intrinsèques seront représentés algorithmiquement par les attributs des objets. Nous allons nous attacher dans cette section à démontrer que même si ces caractères sont effectivement indépendants du contexte, il n'en reste pas moins que l'opération d'extraction, effectuée dans les expressions qui les utilisent, est dépendante du contexte.

Dans la suite de cet article, nous ferons référence au contexte opératoire pour parler de l'état de l'application avec laquelle l'utilisateur dialogue, au contexte ontologique pour parler de l'état des relations qui relient les différentes catégories d'entités entre elles, et au contexte dialogique lorsqu'il s'agira de faire référence à l'état de l'interaction entre l'utilisateur et l'application.

La grandeur

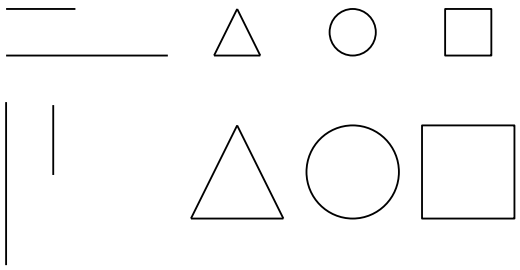


FIG. 6.1 – Palette des formes disponibles, corpus Ozkan

Salmon-Alt étudie les variantes de la référence à partir d'un corpus de dialogues relatant les échanges entre un utilisateur et un opérateur destinés à faire réaliser un dessin complexe à l'opérateur à partir d'un ensemble d'objets graphiques décrits dans la figure 6.1. Ces dialogues sont tirés d'un corpus construit à partir d'une expérimentation décrite dans (Ozkan, 1993).

Dans cette situation, l'approche prédicative est tout à fait efficace, puisque toute entité X de l'application pourrait être représentée sous la forme :

EntitéType X = (taille, couleur) où :

- **EntitéType** ∈ {barre, triangle, rond, carré}
- **taille** ∈ {grand, petit},
- **couleur** ∈ {blanc, noir, rouge, vert, bleu}.

Pourtant, il suffit d'ajouter une taille possible (moyen) pour que cette approche soit mise en défaut pour résoudre l'expression « le grand carré » dans le contexte présenté figure 6.2, puisque le carré situé à gauche est bien un grand carré par rapport à l'ensemble des carrés de la figure, mais son attribut taille ayant la valeur moyen, il ne peut pas être recruté comme candidat à l'expression référentielle « grand carré ».

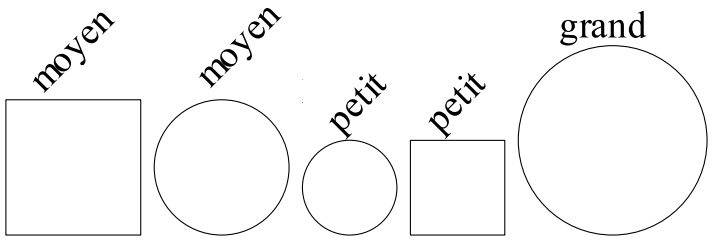


FIG. 6.2 – Problème avec trois tailles (petit, moyen, grand)

La simplification faite dans ce cas, d'ailleurs relevée dans (Dale and Reiter, 1995) sans être approfondie, est que la taille en tant qu'attribut intrinsèque ne peut avoir pour valeurs possibles l'ensemble {petit, moyen, grand} car ces valeurs ne sont pas intrinsèques sauf lorsqu'elles sont normalisées dans un contexte précis (par exemple les tailles de vêtements) et perdent dans ce cas leur caractère relationnel pour être restreints à un rôle de signe (ou d'étiquette). La taille en tant qu'attribut intrinsèque devrait donc simplement contenir les dimensions physiques et objectives des objets.

Ceci montre que la signification dénotationnelle des adjectifs n'est donc pas, même dans un cas aussi simple que grand ou petit, réductible à une approche par prédicats.

La couleur

Nous venons de voir que la grandeur, contrairement à la taille (ces termes sont-ils bien choisis ?), est une caractéristique dépendante du contexte opératoire et du contexte ontologique. Le fait qu'il soit si aisé de se tromper sur ce point montre la difficulté qu'il peut y avoir à déterminer exactement la catégorie (typologique, intrinsèque ou relationnel) des caractères définissant les objets. Nous allons maintenant montrer que cette difficulté est due au fait que même un caractère intrinsèque ne peut être traité de manière uniquement interne à l'objet dans le calcul de la référence, c'est-à-dire que l'opération d'extraction du référent par un adjectif portant sur une caractéristique intrinsèque ne peut être effectuée par une action hors de tout contexte. Alors que l'approche prédicative semblerait poser moins de problèmes que dans le cas précédemment étudié de la grandeur, elle est tout autant limitée à une tâche spécifique, où l'on sait que seul un nombre fini de couleurs clairement distinguables vont apparaître.

Nous illustrons figure 6.3[2] la difficulté qu'il y a à définir la sémantique des adjectifs de couleur dans le cadre du calcul référentiel. En effet, si dans le cas 1, « la feuille bleue » désigne

[2]La figure étant en couleur, et celles-ci étant diversement rendues selon la modalité d'affichage ou d'impression, les affirmations qui y réfèrent peuvent être diversement appréciées des lecteurs. Les variations interpersonnelles dans la perception des couleurs peuvent aussi être un facteur de désaccord.

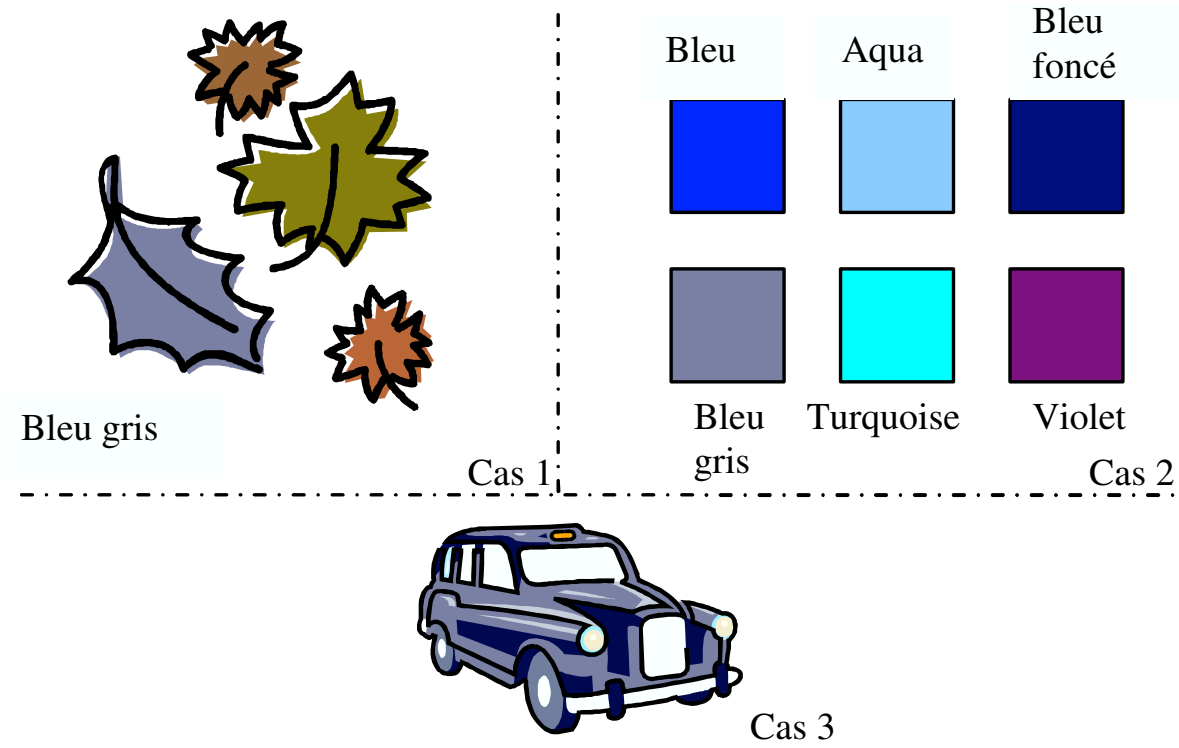


FIG. 6.3 – Catégorisation des couleurs en fonction de l'objet et du contexte.

clairement la feuille en bas à gauche, si l'on demande à un humain de désigner dans le cas 2 les carrés qui ne sont pas de couleur bleu, il choisira les trois carrés du bas. Enfin, si l'on demande la couleur de la voiture dans le cas 3, la plupart des utilisateurs répondent « noir », même s'ils répondent aussi pour la plupart par l'affirmative si on leur demande, dans une autre expérimentation, si la voiture est bleue. Pourtant, la feuille, le carré en bas à gauche et la voiture ont tous la même couleur.

Une feuille, qui est typiquement dans les couleurs vertes, jaunes, rouges ou marrons, peut donc être catégorisée comme bleue dès l'instant où sa couleur s'éloigne de ces couleurs typiques pour aller vers le bleu, tandis qu'une figure géométrique de la même couleur, qui en tant que figure n'a pas de couleur typique, ne pourra pas être catégorisée comme bleue. C'est ce que nous appelons dépendance au contexte ontologique[3]. De plus, si un objet est perçu entouré par d'autres objets, la couleur des ces objets déterminera en partie la manière dont sa propre couleur sera catégorisée par un observateur humain. C'est ce que nous appelons le contexte opératoire.

Importance du contexte pour certains attributs intrinsèques

D'après ces observations, le rôle du contexte ne peut pas être négligé si l'on s'intéresse au calcul référentiel des adjectifs de couleur. Force est donc de constater que même pour des

[3]Voir à ce propos les travaux de Dubois and Grinevald (1999) sur la catégorisation des couleurs d'un point de vue ethno-sociologique et ergonomique.

attributs intrinsèques, le contexte détermine en partie la manière dont les mots qui font référence à ces attributs doivent être pris en compte pour le calcul de l'objet qu'ils servent à désigner. La distinction caractères intrinsèques vs. relationnels, qui semblait suffisante pour dire que le calcul sur les attributs intrinsèques pouvait se réduire à du calcul indépendant du contexte est selon nous invalidée par ces observations. Cette conclusion rejoint celle de Pateras et al. (1995) portant sur des situations de dialogue où les caractéristiques utilisées dans la référence ne peuvent pas être mises en relation directe avec un schéma de base de données, ou plus généralement, avec une représentation prédicative des connaissances. Dans ces travaux, un utilisateur dialogue avec un robot et lui désigne des objets dans son environnement. La solution proposée pour choisir le référent correctement consiste à utiliser des fonctions floues d'appartenance à un ensemble. Cependant, ces fonctions sont définies par rapport à la tâche, (*cf.* figure 6.4) et les auteurs ne proposent pas de moyen pour prendre en compte le contexte. Il en est de même dans Lammens and Shapiro (1993) où les auteurs construisent par apprentissage des fonctions de catégorisations de la couleur, fonctions qui sont au final indépendantes du contexte. Ces recherches montrent cependant clairement qu'une représentation sémantique des adjectifs nécessite un formalisme permettant d'exprimer des fonctions complexes, et permettant d'en définir précisément l'usage[4].

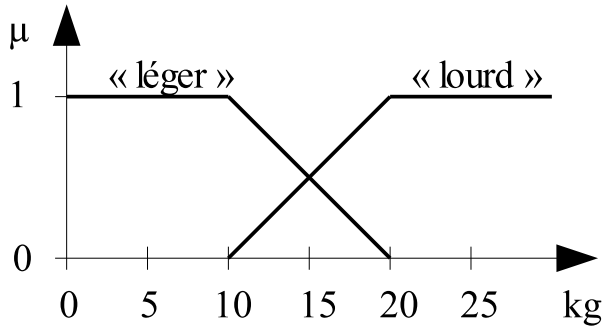


FIG. 6.4 – Catégorisation du poids par une fonction floue d'après Pateras et al. (1995)

Discussion

La question de la représentation sémantique des extracteurs référentiels intrinsèques pour la dénotation est donc posée. Nous estimons que cette représentation devrait supporter les cas d'utilisation suivants (ici avec l'exemple de la couleur bleue) :

- Choisir le bon référent en fonction du contexte. (« Supprime la feuille bleue » dans les Cas 1 et 2, figure 6.3)
- Pouvoir dire si, dans un contexte donné, des objets peuvent être considérés comme bleus ou non (« Montre-moi les objets qui ne sont pas bleus » dans le Cas 2, figure 6.3).
- Pouvoir être utilisé pour déterminer si X est plus/moins bleu que Y (et par conséquent permettre aussi de traiter les comparatifs et superlatifs)

[4]Noter aussi que « léger » et « lourd » seront utilisés différemment par des personnes différentes. Par exemple une fillette pourra trouver un cartable lourd, alors même qu'un haltérophile le trouvera léger. Il faudrait donc dans l'absolu prendre en compte un contexte « personnel », lié au locuteur, dans le processus d'interprétation.

D'autre part, le fait que les extracteurs portant sur des caractéristiques intrinsèques aux objets s'avèrent dépendant du contexte implique que la représentation de ces extracteurs devra se faire par une fonction prenant en compte l'ensemble du contexte. Cette fonction pourra naturellement être décomposée si nécessaire en plusieurs fonctions ayant un champ plus limité, mais qui puissent être itérées ou combinées de manière à être équivalente à une fonction prenant tout le contexte en compte.

6.2.5 Extracteurs relationnels

Les extracteurs relationnels (ou extracteurs extrinsèques) s'appuient sur des relations provenant du contexte. Comme nous venons de le voir, la distinction entre extracteur intrinsèque et extracteur relationnel est quelque peu difficile à expliciter dès l'instant où l'on a constaté que même les extracteurs intrinsèques dépendent du contexte. Nous allons cependant présenter ici l'exemple d'une famille d'extracteurs relationnels qui a donné lieu à de nombreuses études : les prépositions spatiales. La référence spatiale est un sujet qui a été largement abordé dans la littérature scientifique. D'un point de vue de linguistique théorique on peut citer les travaux de Talmy (1983); Langacker (1987); Vandeloise (1986) avec notamment l'étude de la sémantique des prépositions spatiales, et d'un point de vue plus algorithmique par l'étude des solutions de mise en œuvre de la résolution d'expressions locatives. Plusieurs travaux proposent des modèles de compréhension ou de génération d'expressions référentielles dans des environnements bidimensionnels (Wazinski, 1992; Schang, 1995; Herzog and Wazinski, 1994) ou tridimensionnels (Gapp, 1994), hiérarchisés (Schang, 1995), ou hybrides (Dzikovska et al., 2000). Ces différentes approches ne sont pas nécessairement compatibles entre elles, ainsi la notion de cadre de Schang (1995) permet de gérer des contextes imbriqués hiérarchiquement dans un espace bidimensionnel pour la représentation d'objets d'une interface graphique, tandis que l'approche de Gapp (1994) permet de définir des zones d'applicabilité des prépositions par rapport à des bâtiments en trois dimensions.

Nous montrons ci-dessous des exemples de la relation spatiale « *à droite de* » tirés d'un corpus artificiel. La situation sur laquelle repose ce corpus consiste en une tâche dans laquelle l'utilisateur doit déplacer des éléments de formes, tailles et couleurs différentes afin de les positionner d'une certaine manière. On suppose de plus que certaines pièces peuvent être cachées, obligeant l'utilisateur à poser des questions afin de découvrir les caractéristiques des pièces en question :

– « Est-ce que le carré bleu est à droite du triangle » (1.a)
– « Mets le carré bleu à droite du triangle » (1.b)
– « Efface le carré bleu à droite du triangle » (1.c)

D'après nous, dans ces trois cas, les expressions contenant des relations spatiales ont des rôles différents :

1. Dans le cas (1.a), l'expression « à droite du triangle » peut être considéré comme un segment générant une fonction qui doit **vérifier** que la position d'un objet particulier appartient à un ensemble de positions valides (ici l'ensemble des positions situées à droite d'un certain point).
2. Dans le cas (1.b), « à droite du triangle » est appelée à générer une position unique, n'existant pas auparavant en tant que position d'un objet. C'est pourquoi nous considérons que ce segment doit être traité comme un **générateur** de position.

3. Dans le cas (1.c), l'expression est destinée à **sélectionner** un objet parmi d'autres. Son rôle est donc équivalent à une fonction prenant en argument un sélecteur produisant tous les possibles contextes référant à « le carré bleu ».

Relativité de la signification des expressions référentielles

De ces travaux n'émerge aucun consensus sur le sujet, excepté concernant le fait que les approches purement logiques ou géométriques sont inadéquates pour la représentation des relations spatiales. En dehors de cela, on remarque une certaine hétérogénéité entre les travaux portant sur la génération d'expressions référentielles spatiales, et ceux sur l'interprétation de ces expressions. De plus, en fonction du point de vue spécifique sur la représentation spatiale (2D, 3D, géométrique, symbolique) chaque auteur propose des heuristiques spécialisées pour les différentes catégories de relations spatiales.

La diversité des heuristiques permettant de décrire le sens des relations spatiales dans tel ou tel contexte s'explique par deux facteurs. Premièrement, la typicalité intrinsèque des entités sur lesquelles portent ces relations : selon qu'une expression référentielle est appliquée à un objet orienté (une voiture, une personne) ou non (un cube, un mot dans un texte), sa signification va être différente. Deuxièmement, la typicalité contextuelle de la situation dans laquelle la relation est exprimée : selon qu'un bâtiment est considéré dans une vue aérienne ou par une personne présente devant, la prise en compte de l'orientation (l'entrée principale est typiquement considérée comme le devant) est différente.

Problèmes de représentation

En dehors de la variation de la modalité spatiale sur laquelle ces études portent, il existe une variation qui est selon nous intéressante à prendre en compte dans la représentation des informations spatiales, à mettre en parallèle avec la représentation des caractéristiques intrinsèques. En effet, il existe plusieurs solutions algorithmiques à la représentation de la position des éléments.

1. Représentation **intrinsèque** : l'élément possède un attribut position contenant ses coordonnées dans un système de visualisation. C'est typiquement le cas pour les représentations de dessins vectoriels.
2. Représentation **extrinsèque non spatiale** : l'élément ne connaît pas sa position, mais une structure de données externe aux éléments contient les coordonnées de tous les objets. Cette représentation est équivalente à la représentation 1.
3. Représentation **extrinsèque spatiale** : une structure de données imitant la topologie de l'espace contient les éléments. Leur position est déterminée par l'emplacement de stockage. C'est typiquement le cas pour un jeu d'échec par exemple.
4. Représentation **hiérarchique** : la position des objets est déterminée par un emboîtement de conteneurs orientés, soit horizontalement, soit verticalement, voire dans une grille. C'est le cas pour la représentation des interfaces utilisateurs communes.

Ces différentes solutions d'implémentation montrent que le point de vue adopté par les programmeurs sur une caractéristique comme la position spatiale est variable, et que cette va-

riabilité doit être prise en compte pour concevoir un système de résolution de la référence générique du point de vue de l'application cible.

Discussion

A notre avis, il est très probable qu'il n'existe pas **une** bonne représentation des relations spatiales, mais au contraire plusieurs manières de les envisager en fonction des différents environnements dans lequel elles ont un sens. Ainsi, selon que l'on se place dans le cadre du trafic routier, ou dans un traitement de texte, le sens de « sur la droite » est très différent. La signification des expressions relationnelles dépend donc à la fois des entités sur lesquelles elles sont appliquées, mais aussi du contexte dans lequel elles sont exprimées. Cette constatation nous amène au postulat que si l'on veut proposer un formalisme un tant soit peu unifié de la résolution de la référence, il faudrait pouvoir prendre en compte différentes représentations de l'espace (différents points de vues), et donc différentes heuristiques permettant de traiter la référence dans ces différentes représentations, non pas en agrégeant simplement les propositions existantes dans des modules distincts, mais en imaginant un cadre formel dans lequel ces différentes propositions pourraient être implémentées au fur et à mesure des besoins.

6.2.6 Extracteurs typologiques

Les extracteurs typologiques sont incarnés par les noms, ils représentent l'accès aux éléments par leur attribut le plus stable. En général, ce mode d'accès semble assez simple à résoudre dans le cadre du traitement extensionnel de la référence, tout simplement parce que les éléments accessibles sont typés explicitement dans leur définition. En effet, la représentation algorithmique, issue des travaux sur la représentation par cadres ou frames (Minsky, 1975; Fillmore, 1982) ont toutes deux en commun de typer les objets. Or, de même qu'il existe plusieurs points de vue pour représenter la position spatiale, il existe plusieurs points de vue pour se représenter les éléments du monde. Prenons l'exemple de la représentation d'un carré. Voici trois manières de représenter un carré (il en existe probablement beaucoup plus, peut-être une infinité si on considère la possibilité d'une représentation picturale) :

1. **Type** : carré
2. **Type** : figure géométrique, **Forme** : carré
3. **Type** : polygone régulier, **Nombre de côtés** : 4, **Angle** : 90 degrés

Le choix d'une représentation plutôt qu'une autre est assez difficile à soutenir, et pourtant l'attitude la plus courante vis-à-vis du problème de la multiplicité de points de vue est généralement réduite à l'utilisation d'une ontologie spécifiant les relations de synonymie, d'hyperonymie, d'hyponymie, etc. entre les objets. Les phénomènes de métaphore ou de métonymie sont ainsi souvent étudiés en conservant la vision d'une catégorie « vraie » des éléments, à laquelle viendrait s'ajouter une opération d'adaptation basée sur une ontologie.

Plusieurs travaux portant sur la résolution de la référence exposent ce problème de glissement de catégorie dans la référence, c'est-à-dire de l'utilisation d'un type A pour référer à un élément de type B.

(E14) *Salmon-Alt (2001, p.107) expose un exemple tiré du corpus Ozkan, où l'interlocuteur, souhaitant dessiner une petite fille tenant un ballon à partir des formes de la figure 6.1, parle à un moment d'un rond, avant de parler d'une tête pour désigner le même objet. Ce qui était un rond au milieu d'autres figures géométriques est donc devenu la tête de la petite fille. Le même phénomène est présenté pour le triangle, qui devient une pyramide dans le dessin final.*

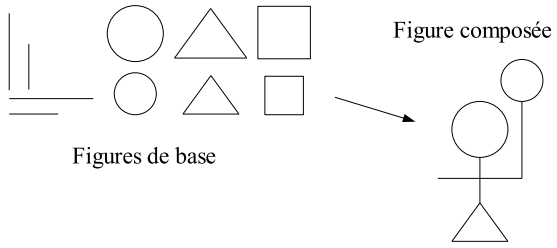


FIG. 6.5 – Un cas d'utilisation dans le corpus de Ozkan (1993).

La solution proposée par Salmon-Alt (2001, p.129) est d'attribuer au même objet les différents liens lexicaux qui sont susceptibles de servir dans l'expression référentielle. Dans un cas précis, où des triangles sont utilisés pour figurer des pyramides, les deux termes « triangle » et « pyramide » sont ainsi reliés au type d'objet TRIANGLE. Cette solution, spécifique à l'application à laquelle est destiné, et même à la situation prévue d'utilisation, ne prend pas en compte le fait qu'un triangle ne sera jamais désigné par le terme « pyramide » s'il n'est pas au préalable mis dans un contexte bien particulier. De plus, elle ne fonctionne pas dans le cas de la petite fille, puisque à la fois le ballon et la tête de la petite fille sont représentés par des cercles (voir la figure 6.5 qui illustre l'exemple (E14)). On ne peut pas à la fois établir de lien lexical entre cercle et tête et entre cercle et ballon.

(E15) *(a)« La cambrioleuse a blessé grièvement le propriétaire du pavillon à la tête. » ;*
(b)« La victime est restée plusieurs heures inconsciente avant de pouvoir donner l'alerte. »

Dans l'exemple (E15), l'expression « la victime » réfère au propriétaire du pavillon. Ce lien ne peut être établi qu'à la condition d'avoir une représentation de l'action de blesser intentionnellement qui puisse permettre de déduire qu'une personne objet de cette action puisse ensuite être désignée par le terme « victime ». Cependant il est faux d'attribuer un lien entre « le propriétaire du pavillon » et « victime » avant l'expression de l'énoncé (*a*). Ces deux exemples démontrent que l'attribution des caractéristiques typologiques doit être considérée comme un **acte dynamique**. Ce point met particulièrement en évidence le fait qu'une ontologie statique n'est pas une solution suffisante pour la représentation des connaissances dans le cadre du dialogue homme machine, notamment dans le cas des agents assistants d'interface, où l'évolution de la situation est inévitable.

Dzikovska et al. (2003) expose le problème intéressant de la dépendance contextuelle du type, et propose une solution pour spécialiser le lexique pour chaque domaine de dialogue différent par l'intermédiaire de règles de traduction. En effet, une ontologie est sensée définir de manière absolue les relations entre différentes catégories, or les relations ne sont pas les mêmes selon qu'on se place dans un contexte ou dans un autre.

(E16) *Dans (Rastier, 1987), l'auteur présente une variation typologique due à la polysémie : les « boyaux » et les « tripes » n'ont pas les mêmes liens ontologiques selon qu'on se place dans un contexte alimentaire ou dans un contexte de course cycliste.*

C'est une illustration supplémentaire de l'importance du contexte dans la manière dont les éléments manipulés pour l'interprétation d'un énoncé doivent être considérés (et représentés).

(E17) *Dans (Dzikovska et al., 2003), les auteurs comparent les différences entre les oranges en tant que cargaison, dans le cadre d'une application de gestion de livraisons, et les oranges en tant que nourriture, dans le cadre d'une application d'assistance médicale. C'est un cas typique où la représentation utile des entités varie selon le contexte : dans le cas du transport, la représentation des oranges peut se limiter à un lien ontologique du type {orange est-une cargaison} ; dans le cadre médical, la représentation devrait inclure les taux de vitamines, sucres, ainsi que les éventuels indicateurs de contre-indication thérapeutiques à la consommation.*

Ici, ce n'est même plus l'aspect ontologique qui est modifié par le contexte (des bananes restent des bananes), mais l'*usage* qui est fait de l'image évoquée par le signe « banane » durant l'interprétation. Cela illustre le fait que le *point de vue* utile à adopter sur les éléments dépend en grande partie de la situation du dialogue et du rôle dans les énoncés des termes qui les désignent. Ces exemples, portant sur les extracteurs typologiques, vont dans le sens de la théorie des Points de Vue, exposée au chapitre 3.

Discussion

Au delà de ces exemples montrant l'impossibilité de lier un type aux objets de l'environnement, et donc d'utiliser la théorie des Points de Vue pour traiter le problème des extracteurs typologiques, il nous faut répondre à la question suivante : peut-on rapprocher les extracteurs typologiques des extracteurs relationnels et intrinsèques ?

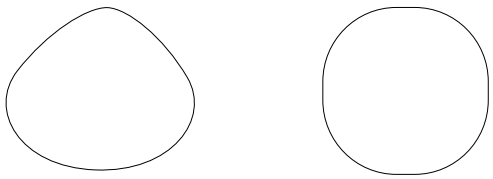


FIG. 6.6 – Formes à mi-chemin entre des formes normées (cercle et triangle à gauche, carré et cercle à droite)

De notre point de vue, la réponse est oui. En effet, si l'appartenance à une catégorie semble intuitivement plus binaire que la position relative dans l'espace de deux objets, c'est en réalité une vérité qui ne s'applique qu'à un monde idéal, ou chaque objet correspondrait précisément au canon de sa catégorie. Or dans la réalité, les choses sont très différentes. Non seulement

la perception par un humain de son environnement est loin d'être parfaite, mais surtout, la catégorisation des objets se fait exclusivement à travers la perception, et les objets n'ont absolument pas de catégorie propre. Ainsi on pourra difficilement attribuer une catégorie précise correspondant à un nom aux formes de la figure 6.6. A gauche, on a un triangle arrondi (ou un cercle affiné en haut), à droite un cercle carré ou un carré très arrondi. Dans ces deux cas, on peut facilement interchanger les deux noms qui correspondent aux catégories les plus acceptables pour chaque objet. De la même manière, on peut dire qu'un objet est plus « carré » qu'un autre, ce qui montre bien que les extracteurs typologiques fonctionnent de manière très proche des autres extracteurs, qu'ils soient intrinsèques ou relationnels.

Conclusion

Dans ce chapitre, nous avons étudié les éléments d'une expression référentielle sous plusieurs point de vues. Nous avons tout d'abord cherché à distinguer les différents rôles ou les différentes fonctions qu'une expression référentielle pouvait jouer dans un énoncé, ceci afin de définir notre objet d'étude. La fonction que nous avons choisi d'étudier est la fonction d'opérateur de reconnaissance, c'est-à-dire la capacité à extraire (reconnaître) les bons candidats dans un ensemble d'éléments.

Nous avons alors posé la notion d'extracteur référentiel, qui est l'aspect fonctionnel d'un élément d'une expression référentielle qui serait vue comme un opérateur de reconnaissance. A partir de cette notion, nous avons cherché à distinguer les différents types d'extracteurs, d'abord en les comparant en fonction de leur catégorie syntaxique (adjectifs, articles, noms), puis en les comparant du point de vue des informations auxquelles les extracteurs accèdent pour remplir leur rôle (informations typologiques, intrinsèques, extrinsèques).

Cette étude nous a permis de déterminer que la catégorie syntaxique ne semblait pas jouer un rôle majeur sur le fonctionnement des extracteurs, ou plus exactement qu'on ne pouvait pas se fonder sur elle pour déterminer les familles d'extracteurs. En nous intéressant davantage aux modes d'accès des extracteurs, c'est-à-dire aux informations auxquelles les extracteurs doivent accéder dans les éléments pour les choisir, nous avons pu nous orienter vers une approche commune à différents extracteurs. Dans le chapitre suivant, nous proposons une modélisation pour ces extracteurs.

Chapitre 7

Modélisation des extracteurs pour la résolution extensionnelle de la référence

Dans ce chapitre, nous cherchons à développer un modèle permettant de calculer le ou les éléments désignés par une expression référentielle, et prenant en compte les prédicats vagues tels que les couleurs et les prédicats relatifs, qu'ils soient implicites tel que ceux portants sur des caractères continus (taille, poids, etc.) ou explicites tels que ceux portants sur la position spatiale.

Nous avons vu au chapitre précédent que les trois types d'extracteurs que nous avons choisi d'étudier fonctionnent d'une manière très proche, et qu'ils ont en commun de pouvoir exprimer des comparaisons entre deux objets. A partir de cette comparaison, nous faisons l'hypothèse que les différentes utilisations des extracteurs que nous voulons supporter peuvent être reproduites.

Nous proposons de représenter les extracteurs référentiels, c'est-à-dire les termes de la langue qui servent à désigner un objet parmi d'autres, par des structures de données contenant des fonctions. Cette approche permet de donner une explication des phénomènes que nous avons décrits à la section 6.2.

Nous nous positionnons dans le cadre d'une application de dialogue où l'utilisateur peut parler d'objets comme ceux visibles à la figure 7.2. Nous considérons pour l'instant que les éléments de l'application sousjacente sont des objets (au sens algorithmique du terme) définis par leur type (ici tous sont de type carré) et par deux attributs : la couleur (un triplet de valeurs comprises dans [0,1]) et la taille (une valeur comprise dans]0,+∞[).

7.1 Proposition préliminaire

Nous proposons figure 7.1 une définition des extracteurs référentiels pour « bleu » et « grand » comme base préliminaire de réflexion. Cette définition se résume à une fonction de comparaison entre deux éléments typés, et qui renvoie une valeur entre -1 et 1.

Nous discutons ensuite cette première proposition pour aboutir à une seconde version, qui permette de traiter un cas supplémentaire de la référence. Dans cette section, nous présentons des fonctions de calcul de proximité sur les caractéristiques intrinsèques des objets. Pour construire effectivement ces fonctions et pour leur donner la validité cognitive nécessaire à leur exploitation effective, des expérimentations psychologiques sur la verbalisation de la perception des couleurs dans différents contextes devraient être mises en oeœuvre. Nous allons cependant faire ici l'hypothèse qu'il est possible de construire ces fonctions.

Extracteur - bleu
$fProx$(**entité** a, **entité** b) $\rightarrow$ $[0, +\infty] \cup \perp$

Extracteur - grand
$fProx$(**entité** a, **entité** b) $\rightarrow$ $[0, +\infty]$

FIG. 7.1 – Définitions préliminaires des extracteurs bleu et grand

La fonction fProx[1] doit permettre de trier un ensemble d'éléments en fonction d'une caractéristique (ici, *bleu* ou *grand*). Pour l'extracteur *bleu*, elle donne le rapport entre les couleurs de deux objets projetées selon la dimension bleue, (à ne pas confondre avec la composante bleue dans un codage RVB[2], qui peut être très élevée sans que la couleur finale ait un quelconque rapport avec le bleu).

- Si a est **autant** *bleu* que b, la valeur retournée par la fonction est **1**,
- Si a est **plus** *bleu* que b, la valeur retournée est comprise dans $]\mathbf{0}, \mathbf{1}[$,
- Si a est **moins** *bleu* que b, la valeur retournée est comprise dans $]\mathbf{1}, +\infty[$,
- Si a (resp. b) ne peut pas être estimé par rapport à l'extracteur *bleu,* et que b (resp. a) peut l'être, alors la valeur retournée est $+\infty$[3] (resp. **0**),
- Si a et b ne peuvent être estimés par rapport à l'extracteur *bleu,* la valeur retournée est $\perp$.

A partir de ces fonctions, il est alors possible de trier les entités appartenant à un ensemble de candidats potentiels (par exemple dans un domaine de référence), et de séparer les entités à exclure des entités à conserver.

A partir de la situation expérimentale exposée figure 7.2, nous illustrons l'utilisation des fonctions de différenciation dans le cadre de la résolution de l'expression référentielle suivante :

« *le grand carré bleu* »

1. Nous prenons comme point de départ pour le domaine de référence l'ensemble des carrés de la scène (ici tous les objets). Nous ne détaillons pas la phase de tri pour l'extracteur « *carré* »

[1] Bien qu'inhabituel en mathématiques, l'intervalle fermé sur l'infini est ici nécessaire.

[2] Codage RVB : représentation d'une couleur par un triplet de valeurs entre 0 et 1 décrivant l'intensité des composantes rouges, vertes et bleues.

[3] Ce qui nécessite d'avoir un système arithmétique qui puisse calculer avec un symbole représentant l'infini.

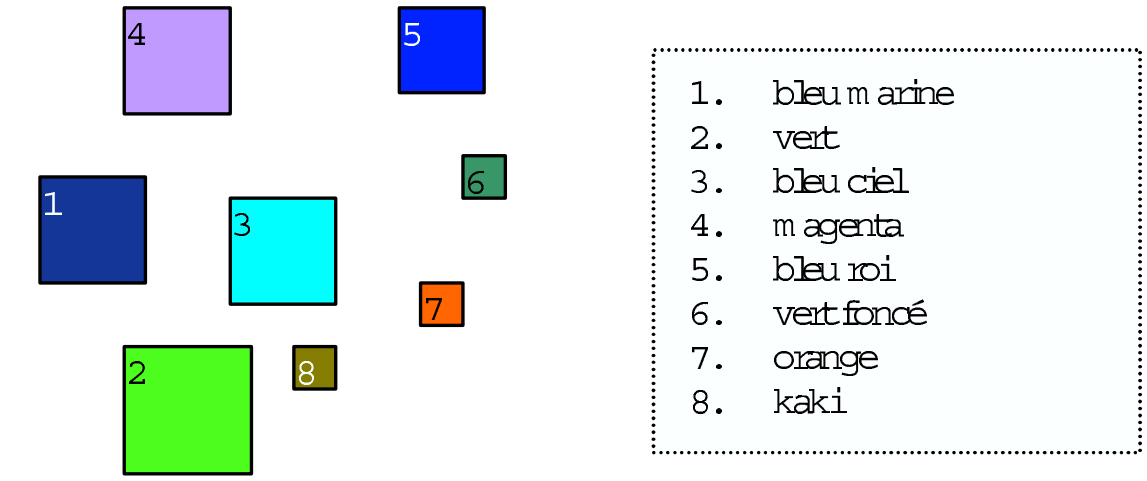


FIG. 7.2 – Situation de test pour la représentation fonctionnelle des couleurs et des tailles.

2. Nous appliquons la fonction de différenciation de l'extracteur référentiel « *grand* ». On suppose ici que la grandeur est une relation transitive, telle que[4] :

$$prox_{Grand}(x,z) = prox_{Grand}(x,y) \times prox_{Grand}(y,z)$$

On peut alors produire une liste triée d'entités, et annoter le lien entre deux entités par la distance qu'il y a entre elles. Le résultat de cette première étape est illustré par la figure 7.3. Dans cette figure ainsi que dans celles qui suivent, les cercles numérotés représentent les carrés de même numéro, et l'arc valué entre deux cercles représente le résultat de la fonction $prox_{Grand}$(x,y) avec x le cercle de gauche et y le cercle de droite.

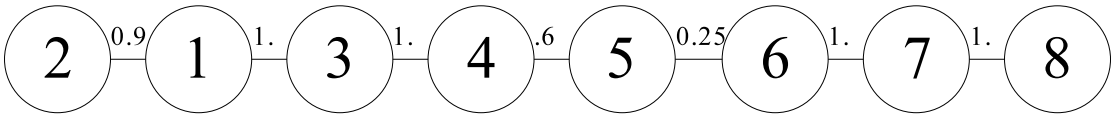


FIG. 7.3 – Etape 1 de la séquence de tris, avec un tri par la fonction de l'extracteur « *grand* »

3. Nous appliquons la fonction de différenciation de l'extracteur référentiel « *bleu* ». En théorie, si la couleur est représentée par un triplet de valeurs dans [0,1], la fonction de différenciation appliquée à n entités doit produire une matrice de $n \times n$ triplets. En pratique nous allons accepter ici que l'opérateur de comparaison entre deux éléments colorés, du point de vue du *bleu*, est transitif, et qu'il peut être exprimé avec une seule valeur. Il y a au moins un moyen pour faire ceci, en considérant les vecteurs entre un

[4] Attention à ne pas confondre la fonction *prox* avec une fonction de distance. A titre de rappel, deux éléments identiques selon le point de vue de l'extracteur (dans le cas de l'extracteur de grandeur, deux éléments de même taille), auront un rapport de proximité de 1, alors que leur « distance » serait de 0.

point que l'on considérera être le « bleu parfait »[5] et chacun des deux points à estimer, et en appliquant à ces vecteurs une fonction qui donnera en quelque sorte la « ligne de niveau » sur laquelle se situe le point estimé. La ligne de niveau 0 étant la ligne de séparation entre les couleurs qui peuvent encore être considérées comme bleues, et celles qui ne le peuvent pas du tout. Dans ce cas, le rapport entre deux couleurs sur la dimension bleue est simplement le rapport entre les lignes de niveau. On peut alors, comme dans le cas de la grandeur, utiliser un algorithme de tri pour produire une liste triée d'entités. Le résultat de ce tri est montré dans la figure 7.4 :

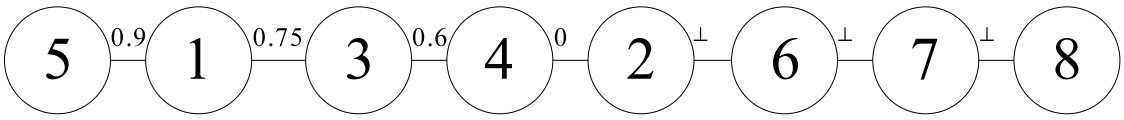


FIG. 7.4 – Etape 2 de la séquence de tris, avec un tri par la fonction de l'extracteur « *bleu* »

Nous posons arbitrairement la fonction de combinaison de ces deux tris comme étant la suivante[6] :

$$\text{prox(x,y)} = \alpha.\text{prox}_{Bleue}\text{(x,y)} \times \beta.\text{prox}_{Grand}\text{(x,y)}$$

où α et β sont des coefficients en rapport avec l'importance relative des caractères couleur et taille qui devraient être déduits à partir d'une expérimentation. Pour le moment, on prend $\alpha = \beta = 1$, et on obtient le résultat de la figure 7.5 :

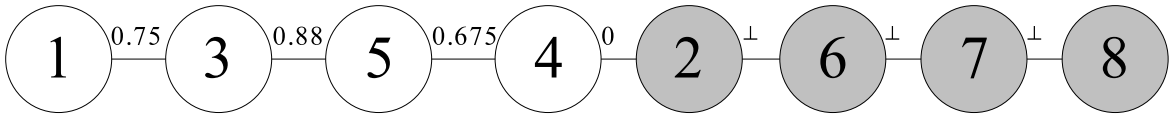


FIG. 7.5 – Résultat de la résolution de l'expression référentielle « grand carré bleu ».

Le résultat du calcul « *le grand carré bleu* » appliqué à la situation de la figure 7.2 donne le carré 1 comme référent préféré comme on peut le constater sur la figure 7.5 Les éléments grisés sont ceux pour lesquels la fonction de comparaison renvoie $\perp$. Ils devraient être exclus si l'expression référentielle était au pluriel comme dans l'expression référentielle : « *les grands carrés bleus* ». La même méthode appliquée à l'expression « *le petit carré bleu* » produirait comme référent préféré le carré numéro 5. Dans les deux cas, cette méthode apporte un résultat cohérent avec ce à quoi on s'attend (ces cas on été soumis à quelques sujets pour

[5] Qu'on suppose défini par défaut hors contexte, avant d'éventuellement le modifier si le contexte peut être considéré comme une influence non négligeable.

[6] Tout autre fonction de combinaison pourrait être valide. En l'absence de données expérimentales, nous avons choisi la plus simple.

avoir une validation expérimentale minimale). On remarquera que la fonction de combinaison (ici le produit des rapports deux à deux) est commutative, et donc que l'ordre de prise en compte des extracteurs est de ce fait indifférent.

7.2 Discussion sur la proposition préliminaire

La réponse apportée par ce calcul à la résolution de l'expression référentielle « le petit carré bleu » est sujette à discussion sur trois points.

1. La construction des fonctions de calcul de similarité pose problème, puisqu'il s'agit pour nous de s'approcher le plus possible des phénomènes de classification humaine. Idéalement, il serait selon nous indispensable de procéder à une étude psychologique permettant de déterminer les fonctions donnant des résultats proches des expérimentations. De plus, la simplification consistant à considérer que la distance entre deux couleurs puisse être projetée sur une échelle à une dimension n'est peut-être pas justifiée.
2. D'autre part, il est tout à fait possible que l'on considère que la réponse attendue à une telle référence soit « Il n'y a pas de petit carré bleu », ce que la méthode ne permet pas en l'état, puisque la fonction de grandeur ne permet pas l'exclusion de candidats. (La validation expérimentale sur l'énoncé « le grand carré orange » dans la situation décrite figure 7.2 donne en effet une réponse négative, mais donnerait l'ensemble **{7}** avec la méthode de la proposition préliminaire).
3. Enfin, que se passerait-il si l'on voulait maintenant résoudre l'expression référentielle « *les carrés bleus* » ? La méthode de calcul préliminaire ne permet pas d'expliquer le phénomène observé autour du cas 2 de la figure 6.3 (dépendance contextuelle opératoire), puisque les couleurs présentes dans la situation ne sont pas prises en compte dans le calcul de similarité effectué dans la proposition préliminaire.
   La question est donc de savoir si le référent de l'expression « *les carrés bleus* » dans la situation de la figure 7.2 est l'ensemble des carrés **{1,3,4,5}** ou **{1,3,4}** ou **{1,3}** ou encore un autre ?

   L'expérimentation[7] montre que les sujets donnent comme réponse l'ensemble **{1,3,5}**, le carré 4, situé à la même distance du bleu que le carré 3 est écarté. La validation expérimentale sur « *les petits carrés* » produit la réponse **{6,7,8}**, alors que « *le petit carré bleu* » produit bien la réponse **{5}**, qui n'est pas dans l'ensemble **{6,7,8}** bien que dans les deux cas, l'adjectif « *petit* » a été utilisé.

Comme nous le verrons dans ce qui suit, les réponses que nous proposons pour résoudre les deux derniers problèmes, à savoir

– la réponse négative et
– le traitement des références plurielles,

[7] Les tests effectués sur ces figures sont trop peu nombreux pour être présentés comme une vraie expérimentation psychologique. Nous nous autorisons tout de même à prendre en compte leurs résultats car ils semblent peu sujet à variation, et qu'il ne s'agit pas là de notre objectif principal. Une phase de validation plus sérieuse est cependant à prévoir dans le développement de nos travaux.

consistent d'une part à effectuer un partitionnement du domaine de référence, et d'autre part à ordonner l'application des fonctions des extracteurs de façon à nous rapprocher au mieux de la manière humaine « moyenne » d'interpréter les expressions référentielles.

7.3 Proposition avec partitionnement des domaines de référence

L'approche fortement relationnelle de la représentation des extracteurs par fonctions de similarité pose le problème de la prise en compte du caractère semi-relatif de certains usages des extracteurs référentiels (dans les cas où la norme de comparaison est très solidement établie). Nous entendons par caractère semi-relatif le fait qu'on puisse considérer qu'un objet est caractérisé par un certain adjectif dans un contexte donné. Par exemple, on peut dire qu'une personne est grande et qu'un autre ne l'est pas, donc la grandeur n'est pas seulement un caractère relationnel. Ce fait nous amène à conclure qu'une simple fonction de comparaison n'est pas suffisante pour représenter le pouvoir dénotationnel d'un extracteur référentiel, et qu'il est nécessaire d'y adjoindre un mécanisme de partitionnement.

7.3.1 Prise en compte du contexte dans la fonction de similarité

Comment, avec la fonction de proximité de l'extracteur de grandeur proposée en proposition préliminaire 7.1, est-il possible de résoudre l'expression référentielle « les grands carrés bleus » ? Nous voyons en effet que cette fonction permet simplement de trier les entités en fonction de leur taille, ce qui est normal puisque la fonction ne peut pas *a priori* exclure les petits objets, eux-mêmes pouvant être grands relativement à d'autres. Mais de ce fait, la fonction $f\text{Prox}$ ne permet pas de séparer les grands des non-grands. Il faut donc trouver un mécanisme permettant d'exclure certaines entités. D'autre part, l'expérimentation montre que les sujets ne catégorisent pas de la même manière selon le contexte. Ainsi dans le cas 2 de la figure 7.6, les sujets considèrent que seuls les carrés du haut sont des carrés bleus, alors que si les carrés du dessous sont vus seuls, le carré de gauche et celui du milieu sont considérés comme bleus. Il faut donc aussi un mécanisme permettant de marquer les éléments typiques en fonction du contexte.

1. Une solution serait de faire passer l'opération d'extraction sur la couleur en premier, et d'appliquer ensuite la fonction de similarité pour la grandeur sur les entités ayant un rapport compris dans [0,1] après extraction sur la couleur. Cette solution présente cependant plusieurs inconvénients :
   - Elle ne permet pas de répondre « il n'y a pas de grand carré orange » si le locuteur utilise l'expression référentielle « le grand carré orange » dans le cas de la figure 7.2.
   - Elle nécessite de trouver une justification au fait de faire passer la couleur avant la grandeur.
2. Une autre solution consisterait à effectuer l'étape de sélection des référents candidats en se basant non pas sur une partition binaire, mais sur une partition ternaire découpée en exclus, possibles et typiques. Avec cette solution, les entités typiques sont utilisées pour résoudre les expressions définies plurielles, les entités exclues sont utilisées pour déterminer la « non-existence », et les possibles sont utilisées pour résoudre la référence

définie au singulier lorsqu'il n'y a pas de candidats typique. Cette solution pose aussi un problème, car elle ne permet pas de résoudre la référence définie plurielle lorsqu'il n'y a pas de candidat typique.

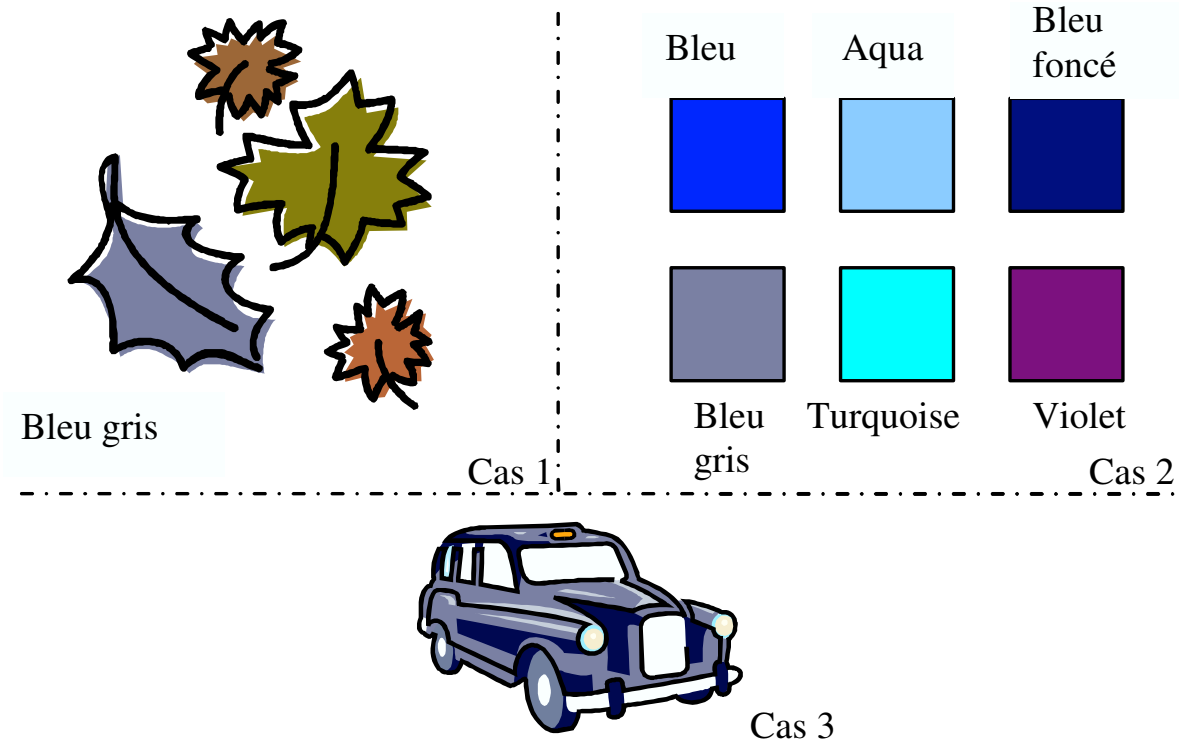


FIG. 7.6 – Catégorisation des couleurs en fonction de l'objet et du contexte (identique à la figure 6.3).

La solution que nous proposons est un intermédiaire entre ces deux ci-dessus. La partition ternaire proposée en (2) est nécessaire pour pouvoir distinguer entre entités typiques, entités exclues et entités possibles. Pour la grandeur cependant, cette partition ne peut être appliquée qu'après que l'extracteur a été appliqué sur la couleur, comme il est proposé en (1), il est donc nécessaire de comprendre pourquoi les opérations d'extraction devraient être ordonnées.

7.3.2 Ordonnancement des opérations d'extraction

Pour justifier un ordre dans la mise en œuvre des opérations d'extraction, nous nous appuyons dans un premier temps sur les observations émises dans (Dale and Reiter, 1995) à propos de l'importance relative à attribuer aux caractères des objets pour la génération d'expressions référentielles. Ces observations ont une forte corrélation avec les maximes de Grice (1975), et suivent globalement le principe d'économie maximum. Les auteurs proposent de prendre en compte les caractères dans l'ordre suivant : type > propriété intrinsèque > propriété relationnelle.

De cette observation issue de la génération d'expressions référentielles, nous proposons de généraliser en considérant que le coût d'utilisation d'un certain caractère est relatif à la quantité

d'information contextuelle à prendre en compte pour sa mise en œuvre dans le processus d'extraction référentielle. Nous exprimons alors la règle d'économie par la proposition suivante :

Plus le contexte opératoire est significatif dans la sémantique d'un extracteur référentiel, plus tard cet extracteur référentiel doit être mis en œuvre dans la chaîne de résolution.

La perception des couleurs étant assez peu modifiée[8] par le contexte opératoire, contrairement à la perception de la grandeur, l'ordre d'interprétation est donc bien couleur puis grandeur. Cette approche permet aussi d'expliquer pourquoi le type (le nom) est toujours pris en compte avant les adjectifs dans la résolution de l'expression référentielle. L'expression « *le grand carré bleu* » s'exprime donc de manière non ambiguë par : le grand, parmi les bleus, parmi les carrés. Cette explication nous permet du même coup de nous poser des questions sur la sélection par le type (la catégorie) des éléments. Nous avons en effet vu dans la section 6.2 que les caractères typologiques ne sont pas si simples à traiter qu'il y paraît de prime abord, et que le contexte était un élément important pour traiter les extracteurs typologiques. Cependant, nous nous sommes pour l'instant contentés de considérer que la sélection des objets de type carré était suffisamment simple pour ne pas avoir à s'en préoccuper. Comme nous venons de voir que l'ordre de prise en compte des extracteurs référentiels dépendait du poids du contexte dans l'opération d'extraction, cela pourrait aussi signifier que les différents types d'extracteurs n'ont pas autant de différences dans leur réalisation qu'on aurait pu croire à priori.

7.4 Représentation finale des extracteurs référentiels

D'après les propositions que nous avons émises précédemment, un extracteur référentiel est donc représenté par un agrégat de trois fonctions. Figure 7.7, nous voyons un extracteur défini par :

- Une fonction fProx, permettant de trier des entités selon un critère donné en donnant un rapport de similarité selon ce critère entre deux entités. En plus des deux entités dont elle calcule le rapport, cette fonction peut prendre en argument un point de référence, optionnel ou non selon l'extracteur. Ce point de référence est déterminé par le contexte ontologique, c'est-à-dire par la catégorie des entités du domaine de référence sur lequel la différenciation est appliquée. Dans le cas de l'extracteur bleu, ce point serait le bleu typique pour une catégorie d'entités donnée.
- Une fonction fPossibles, permettant de marquer le point d'exclusion dans le domaine de référence produit par le tri effectué avec la fonction fProx. Pour l'extracteur grand, le point d'exclusion sera le dernier rapport de similarité entre entités strictement inférieur à

[8] Pour comprendre le calcul du coût d'interprétation, il est nécessaire de voir le processus d'extraction comme un algorithme paresseux, ne prenant en compte les informations que lorsque c'est nécessaire. Du coup, la prise en compte du contexte opératoire pour l'opération d'extraction sur la couleur n'est nécessaire que si le rapport entre les éléments d'un domaine de référence est inférieur à la variation maximum apportée par la prise en compte du contexte. Cette explication permet de comprendre que, même si la couleur est tout autant dépendante du contexte que la grandeur, le poids du contexte est bien moins important sur la première que sur la seconde. Le calcul effectif de ce poids est une question non négligeable, qui reste pour l'instant hors du champ de nos travaux.

1 (c'est-à-dire les objets les plus petits). Pour l'extracteur couleur, ce point correspond au premier rapport de similarité égal à $\perp$.
– Une fonction *f*Préférés, permettant de marquer le point à partir duquel les entités ne sont plus typiques pour un extracteur donné. Nous n'avons pu déterminer de règle générale pour cette fonction, mais pour la taille par exemple, elle pourrait marquer le point où le rapport de similarité entre une entité et la première entité de la liste devient inférieur à un certain seuil, ou bien encore le point ou le rapport de similarité d'entités consécutives tombe en dessous d'un certain seuil.

Extracteur :
$fProx$(**entité** a, **entité** b, ref) $\rightarrow [0, +\infty] \cup \perp$
$fPossibles$(**DomaineRef** d) $\rightarrow$ *DomaineRef*
$fPréférés$(**DomaineRef** d) $\rightarrow$ *DomaineRef*

FIG. 7.7 – Représentation d'un extracteur référentiel avec prise en compte des marques d'exclusions et de typicalité.

7.4.1 Algorithme de traitement des opérateurs de sélection référentielle.

L'algorithme de construction d'un domaine de référence intermédiaire s'exprime comme ceci :

1. Choisir le domaine de référence courant en fonction du type de l'expression référentielle (définie, indéfinie, etc.) tel que proposé dans (Salmon-Alt, 2001)
2. Parmi tous les extracteurs, faire dans l'ordre croissant de dépendance au contexte
   (a) Trier les entités non exclues du domaine de référence courant avec la fonction de similarité de l'extracteur référentiel.
   (b) Marquer la borne d'exclusion dans la liste triée.
3. Si la totalité du domaine de référence est exclu, fin et échec de la résolution
4. Si le type de l'expression référentielle nécessite de sélectionner les référents typiques, alors aller en 5 sinon aller en 6
5. Parmi tous les extracteurs, faire dans le même ordre que 2 : Marquer la borne typique dans la liste triée des candidats référents. (Le calcul de la borne typique est effectué en fonction des entités présentes dans le domaine non exclu)
6. A partir de la liste triée et marquée, extraire le ou les candidats référents vainqueurs en fonction du type de l'expression référentielle.
7. Mettre à jour le domaine de référence courant.

Algorithme 1: Algorithme de traitement des opérateurs de sélection référentielle.

7.4.2 Exemple de résolution avec partitionnement

La figure 7.8 illustre les trois étapes de la résolution d'un extracteur (« carré ») dans une situation où les figures présentes dans l'environnement peuvent être soit des carrés, soit des

carrés aux coins arrondis, soit des cercles, soit des rectangles. La première étape consiste à effectuer le tri des éléments disponibles suivant la fonction de calcul de la « proximité » des éléments entre eux par rapport au prédicat. La seconde étape consiste à couper la chaîne à l'endroit où les éléments deviennent totalement inacceptables pour le prédicat. La dernière étape consiste à couper la chaîne à l'endroit qui sépare les éléments préférés des éléments qui peuvent encore être acceptables, mais sont nettement différenciés par rapport aux éléments préférés.

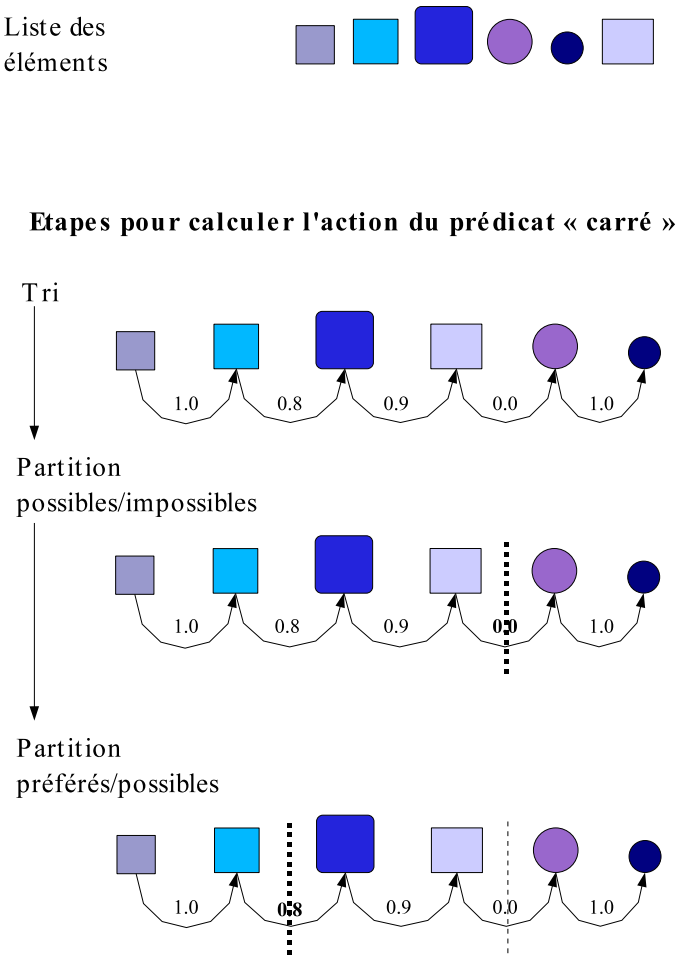


FIG. 7.8 – Les trois étapes de la résolution pour un extracteur.

7.5 Application du modèle fonctionnel aux références relationnelles

La différence entre extracteurs intrinsèques et extracteurs relationnels tient principalement en ce que ces derniers ne peuvent être résolus sans prendre en compte le contexte opératoire de l'énoncé. En dehors de cette plus forte dépendance au contexte, le principe de représentation des extracteurs reste le même. Quelques remarques particulières doivent cependant être formulées ici, car il existe une forte imprécision quant à la sémantique dénotationnelle de certains extracteurs relationnels. Dans le contexte de la figure 7.2, nous avons demandé à plusieurs sujets de pointer l'élément qui, selon eux, était désigné par les énoncés suivants :

(1) « Quel est le carré le plus à gauche de 8 ? » ; la réponse est généralement **2**, éventuellement **1**.

(2) « Quel est le carré le plus à droite de 1 ? » ; la réponse est soit **3**, soit **6**.

On constate en revanche qu'aux questions suivantes, la réponse est toujours la même :

(3) « Quel est le carré à droite de 1 ? » ; la réponse est **3**.

(4) « Quel est le carré de gauche ? » ; la réponse est **1**.

(5) « Quel est le carré le plus à droite ? » ; la réponse est **6**.

Les observations (1) et (2) nous ont conduit à émettre la supposition qu'il existe peut-être des extractions implicites sur des caractères comme la proximité spatiale, et que ces extractions implicites sont relativement instables dans le cas général. L'observation (5) pose le problème de déterminer le point de référence par défaut de la relation *à-droite-de*, puisque le comportement sans référence explicite est stable, alors que le comportement avec référence explicite est instable.

7.6 Application du modèle fonctionnel aux noms

Dans un premier temps, nous considérons les noms comme des prédicats typologiques, c'est-à-dire les prédicats qui permettent de désigner des objets en fonction du type (de la catégorie) qui peut être attribuée à ces objets.

Nous avons vu comment une représentation fonctionnelle pouvait permettre de traiter les expressions référentielles portant sur les caractères intrinsèques. Nous avons cependant jusqu'ici fait l'impasse sur la sélection des éléments par leur type, en prenant comme hypothèse de départ, comme c'est couramment le cas, que l'extraction des référents d'un certain type se faisait naturellement par une opération prédicative.

Ainsi, l'expression référentielle « *les carrés* » sera simplement traitée en recherchant tous les éléments dont le type est carré dans le contexte de l'application. Les phénomènes de glissement typologique que nous avons montré à la section 6.2 ne sont en général pas traités, sauf en étendant les termes référents aux types par l'utilisation de synonymes, hyperonymes ou hyponymes (plus généralement, les relations ontologiques). Or nous avons vu que les glissements typologiques pouvaient n'être autorisés que dans certains contextes (l'exemple (E15) du cambrioleur et de la victime). L'utilisation des différents termes pour référer à un type n'est

de plus pas équiprobable selon la situation, ce que ne reflètent pas les réseaux de relations ontologiques, qui ont pour caractéristique d'être statiques et généralement ne spécifient pas de pondération sur leurs liens.

Si l'on veut unifier la résolution de la référence afin que les opérations d'extraction sur les caractères intrinsèques, relationnels et typologiques puissent être représentées par un même formalisme, il est nécessaire de pouvoir mesurer une « distance sémantique » entre les types des éléments de l'application et ceux des termes utilisés pour y référer. Pour ce faire, nous considérons qu'il est nécessaire de voir les relations ontologiques non plus comme des relations statiques définitoires du monde, mais comme des transitions de points de vue possibles à un instant donné.

La fonction *f*Prox présentée figure 7.7 a pour arguments deux objets *a* et *b* de type *entité*. Nous n'avons volontairement pas détaillé le processus d'accès aux attributs de taille ou de couleur de ces entités dans la fonction, et nous avons supposé qu'il existe un moyen d'y parvenir. Une approche assez classique (Charniak, 1978; Minsky, 1981; Schank, 1981) de la représentation nous conduirait à considérer que les entités sont définies de la manière suivante :

- **Type** Carré *hérite de* Figure
  - **Couleur** : ([0,1], [0,1], [0,1])
  - **Taille** : $\Re_*^+$

Dans cette approche, l'accès à la valeur se fait par une opération générique d'accès à un attribut (notée *a.Couleur* pour accéder à l'attribut *Couleur* de l'élément *a*). Choisir ce mode de représentation signifie que pour tout élément identifiable de l'application auquel on donne le type *Carré*, on considère que cet élément doit posséder un attribut *Couleur*. Or si nous suivons notre objectif de généricité, il n'est pas envisageable de considérer que le programmeur de l'application ait conçu une hiérarchie de types susceptible de correspondre à celle utilisée dans les définitions des extracteurs référentiels. Le problème est d'autant plus prégnant si l'on considère les cas où les extracteurs référentiels ne concernent pas directement des éléments de l'application, mais des constructions issues de l'analyse de la référence, comme par exemple des groupes d'objets. On doit bien se rendre compte que parler de la couleur d'un groupe d'objet est une opération d'extraction référentielle en elle-même, aussi devons nous envisager une solution pour traiter de manière générique l'accès aux caractéristiques des éléments.

Conclusion

Nous nous sommes intéressés à la modélisation des extracteurs pour la résolution extensionnelle de la référence. Pour cela, nous avons tout d'abord fait une proposition préliminaire fondée sur l'idée que le rôle des extracteurs était modélisable comme une fonction de comparaison. Cette première proposition s'est cependant heurtée à deux difficultés majeures. Tout d'abord cette modélisation ne permet pas de retourner une réponse négative lorsqu'il n'y a aucun candidat satisfaisant à l'expression référentielle, et ensuite, elle ne permet pas de répondre correctement aux expressions référentielles plurielles.

Ceci nous a conduit a proposer un modèle plus complet, capable de générer une partition ternaire de l'ensemble des éléments, en séparant les candidats préférés, les candidats possibles et les candidats rejetés. Cette modélisation correspond à une extension du modèle des domaines de référence dont nous nous sommes largement inspirés.

Nous avons ensuite détaillé l'usage de cette modélisation pour les trois modes d'accès des extracteurs : typologique, intrinsèque et relationnel (ou extrinsèque). Les modes d'accès intrinsèques et relationnels ne posent pas de problème, tandis que l'accès aux caractères typologiques nécessite d'après nous de s'appuyer sur l'approche que nous avons proposée au chapitre 3, décrite par la théorie des Points de Vue, pour éviter l'écueil de considérer le type comme un attribut définitoire des éléments.

Chapitre 8

Modélisation du processus de résolution fonctionnelle dans le Modèle d'Interprétation Constructionnelle

Dans ce chapitre, nous allons développer l'implémentation du modèle de résolution proposé précedemment, en nous intéressant particulièrement aux problèmes soulevés par le fait d'exprimer les fonctions de comparaison en utilisant les s-constructions. Le modèle fonctionnel de résolution extensionnelle de la référence (Pitel and Sansonnet, 2003a) présenté dans le chapitre 7 utilise une représentation des prédicats référentiels à trois fonctions :

– une fonction de tri, retournant le rapport entre deux entités selon le point de vue d'un certain prédicat,
– une fonction de partitionnement possibles/impossibles,
– une fonction de partitionnement préférés/possibles.

Pour un prédicat référentiel, il faut donc représenter trois transitions : d'abord entre le domaine de référence d'origine (dO) et le domaine de référence trié en fonction du prédicat (dT), puis entre le domaine de référence trié et le domaine bi-partitionné (dP_1), enfin entre le domaine de référence bi-partitionné et le domaine de référence à trois partitions (dP_2). La figure 8.1 schématise ce processus global pour une expression référentielle.

Dans ce schéma, le contexte dO est étiqueté *contexte objets,* les contextes dT sont étiquetés *contexte trié/<extracteur>,* les contextes dP_1 sont étiquetés *contexte partitionné_1/<extracteur>* et les contextes dP_2 sont étiquetés *contexte partitionné_2/<extracteur>*. Les flèches indiquent que l'élément qui se situe à l'origine de la flèche est un constituant d'une s-construction qui produit l'élément pointé par la flèche.

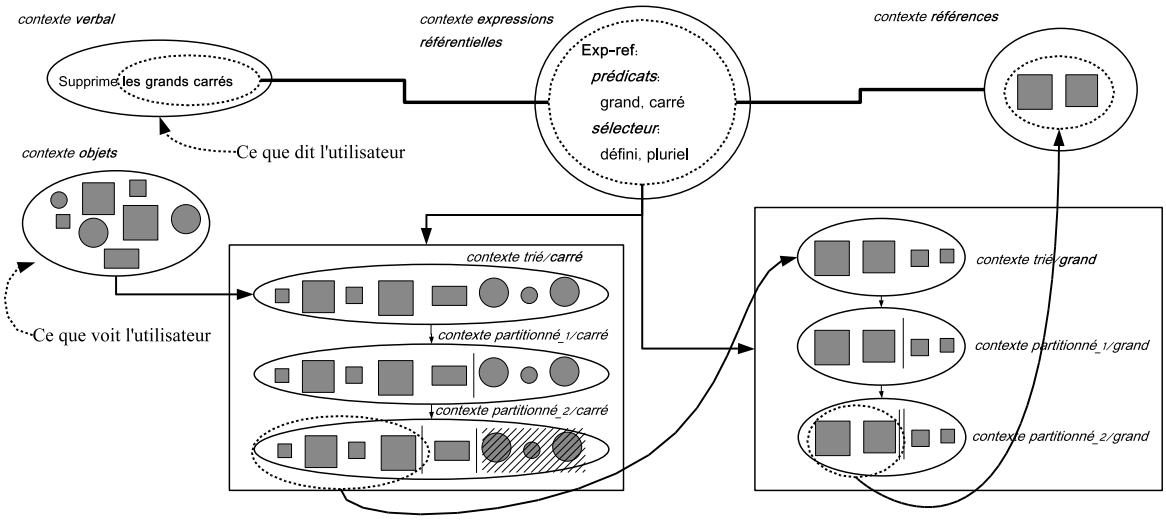


FIG. 8.1 – Schéma de résolution de l'expression référentielle « **les grands carrés** ».

8.1 Discussion sur l'implémentation

8.1.1 Tri

Le problème principal qui se pose ici est de traiter le tri avec le modèle d'interprétation constructionnelle. En effet, si les problèmes de tri sont très classiques en algorithmique (procédurale, fonctionnelle ou distribuée), nous avons avec le modèle d'interprétation constructionnelle une variante légèrement différente, notamment parce que les contextes du modèle peuvent disposer de certaines « compétences » de tri à cause de leur implémentation d'une certaine topologie, sans pour autant que cela soit trivial de les utiliser depuis les s-constructions. Nous avons envisagé deux solutions à ce problème :

1. Une solution inspirée de la technique de tri dite « tri-fusion », qui est une application du paradigme « diviser pour régner ». Le principe, illustré à la figure 8.2, repose sur le fait que la fusion de deux listes triée se fait avec un complexité en $O(n)$, autrement dit en temps linéaire. En appliquant la fusion de manière récursive, cela revient à morceler la liste initiale en morceaux de plus en plus petits (jusqu'à des morceaux de cardinalité 2), ce qui se fait en complexité $O(log(n))$, c'est à dire en temps logarithmique. Au final, la complexité du processus est donc de $O(n.log(n))$.

   Pour appliquer cet algorithme à notre principe, il faudrait donc créer des contextes contenant de moins en moins d'éléments, puis se reposer sur une s-construction qui trierait les deux éléments du contexte. Ensuite, il faut trouver un moyen de fusionner les listes triées. C'est justement ce qui pose problème, car si la description fonctionnelle ou procédurale de cette action est simple, seule la définition fonctionnelle est ici utilisable, et elle impliquerait la création d'un grand nombre de contextes (un contexte à chaque appel récursif). Donc, si cette solution semble réalisable, elle paraît aussi trop lourde.

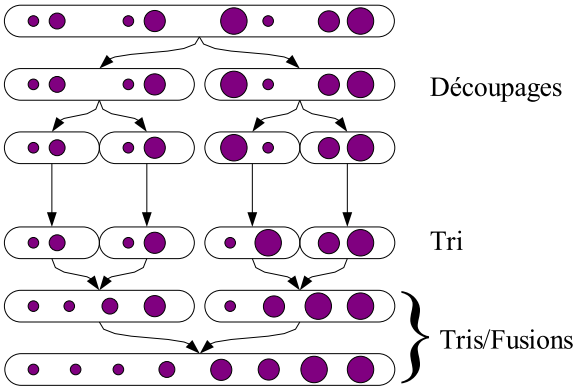


FIG. 8.2 – Processus de tri pour l'extracteur « petit » utilisant la technique du tri-fusion.

2. L'autre possibilité consiste à faire reposer le tri davantage sur le *contexte*, en utilisant une *s-construction* qui calcule dans le contexte destination la position relative d'un élément provenant du contexte d'origine par rapport à un élément déjà présent dans le contexte destination. Les figures 8.3 et 8.4 illustrent le processus sur deux étapes différentes. La première consiste en un ajout simple de l'élément dans le contexte destination, qui est possible puisque le lien créé est directement celui spécifié par la contrainte déterminée par la s-construction. La seconde montre comment un élément est ajouté lorsqu'il doit s'insérer entre deux éléments déjà présents dans le contexte destination. C'est l'implémentation du contexte qui est chargé de cette étape.
   Si cette solution semble plus simple que la solution utilisant le tri-fusion, elle présente plusieurs inconvénients qui n'apparaissent pas dans l'autre solution. Premièrement, il faut « amorcer » le modèle en plaçant un des éléments (n'importe lequel) du contexte origine dans le contexte destination. Deuxièmement, il faut définir les *s-constructions* de tri en double, car si l'on veut pouvoir comparer un cercle et un carré, par exemple, il faudra une *s-construction* qui observe un cercle dans dO et un carré dans dT mais aussi, une *s-construction* qui observe un carré dans dO et un cercle dans dT. Ces deux inconvénients ne sont pas rédhibitoires, mais un troisième pourrait l'être : si on se trouve dans la situation ou un élément X de dT a un taux de similarité de 0 (c'est-à-dire que ce sera ensuite un élément exclus) avec son voisin de gauche, et qu'une *s-construction* « choisi » cet élément X comme élément référence du calcul de similarité avec un élément Y de dO, la s-construction posera simplement comme contrainte que *rapportSimilarité(Y,X) = 0*. Le contexte sera alors incapable de placer le nouvel élément Y' dans dT par rapport aux autres éléments de dT. Une solution possible, même si elle est peut satisfaisante car sans fondement autre que pratique, consiste simplement à choisir toujours comme élément de référence dans dT l'élément le plus à gauche.

Nous avons choisi de présenter la spécification de la seconde solution, car elle nous paraît opérationnelement plus envisageable. Il sera toutefois intéressant de comparer les deux solutions à la fois selon l'aspect de leur rapidité d'exécution, de leur simplicité de spécification et de leur capacités d'extension.

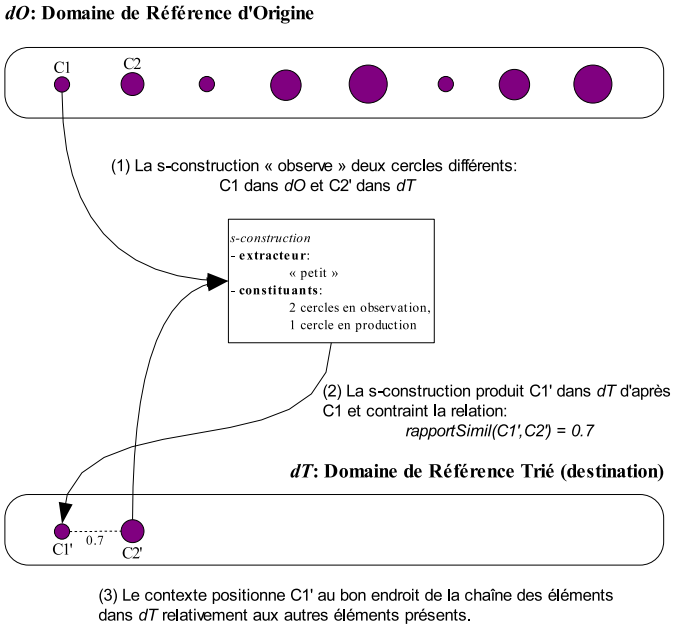


FIG. 8.3 – Schématisation du processus de tri dans le modèle d'interprétation constructionnelle, utilisant une technique d'ajout simple. Etape ne nécessitant pas d'insertion.

8.1.2 Ordonnancement des extracteurs

Bien qu'il précède l'action des extracteurs, nous n'abordons l'ordonnancement des extracteurs qu'après avoir discuté le tri, car il dépend de la technique choisie auparavant pour le tri. En effet, l'ordonnancement des extracteurs, c'est-à-dire le choix de l'ordre dans lequel sont traités les différents éléments d'une expression référentielle, est principalement un problème de tri.

Nous n'avons pas abordé en détail ce problème dans le chapitre précédent. Notamment, si nous avons évoqué la possibilité d'ordonner les extracteurs en fonction de leur dépendance au contexte, nous n'avons pas développé de procédure permettant de mesurer cette dépendance. Nous allons donc simplement postuler qu'il existe des fonctions de comparaisons entre extracteurs, fonctions qui peuvent dépendre du contexte, et qui seront implémentées dans le modèle d'interprétation constructionnelle avec une approche similaire à celle utilisée pour implémenter la fonction de tri du modèle fonctionnel de résolution de la référence.

Dans ce cadre, l'ordonnancement des extracteurs va donc consister simplement à implémenter un tri d'après la méthode exposée précédemment. Ce choix n'est évidemment pas une réponse totale au problème. La véritable problématique qui consiste à savoir comment choisir entre deux extracteurs est finalement déléguée à la mise en place de *s-constructions* effectuant la comparaison.

Pour ces raisons, nous n'illustrerons pas l'opération d'ordonnancement des extracteurs ici, nous nous contenterons de décrire les *contextes* et *s-constructions* nécessaires à la phase de tri des éléments, qui est dans son principe identique au tri des extracteurs.

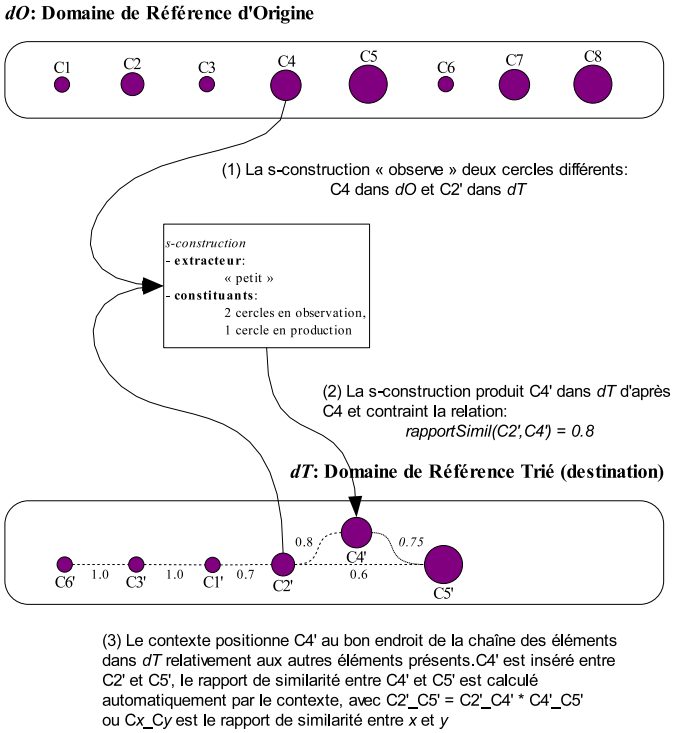


FIG. 8.4 – Schématisation du processus de tri dans le modèle d'interprétation constructionnelle, utilisant une technique d'ajout simple. Etape nécessitant une insertion entre deux éléments.

8.1.3 Partitionnement

Pour la question du partitionnement, deux questions se posent. La première consiste la manière dont seront traitées les différentes partitions du domaine de référence à chaque étape de la résolution. Il est en effet possible soit :

– de créer un contexte pour chaque partition, ou bien
– de créer un contexte pour chaque domaine de référence, et spécifier l'appartenance d'un élément à une certaine partition à l'aide d'un prédicat.

Comme le nombre de partitions est constant, nous avons choisi la seconde solution qui nous semble nettement moins lourde que la première, la manipulation de plusieurs contextes étant toujours plus délicate que la manipulation d'un seul contexte.

La seconde question qui se pose concerne la détermination du point de partitionnement. En effet, si l'on veut conserver un fonctionnement générique, le point de partitionnement ne peut être déterminé par le contexte lui-même, même si celui-ci peut fournir des informations qui peuvent être utiles au partitionnement. Dans l'exemple que nous donnons, le partitionnement est assez simple puisse qu'il repose justement sur un point déterminé directement par le contexte, il faudrait cependant garder à l'esprit que le calcul du point peut s'avérer plus

complexe et nécessiter l'écriture de contraintes évoluées. Dans ce cadre, il faut s'interroger sur l'éventualité de rajouter une étape dans le processus, consistant en une *s-construction* qui établirait dans le *contexte* le point de partitionnement en fonction de l'extracteur en cours de traitement.

Nous n'avons pas déterminé cette étape intermédiaire, qui restera par conséquent dans les perspectives de développement du modèle.

8.2 Schémas

Pour décrire ce modèle, il est nécessaire de spécifier un ensemble de points de vue avec des schémas :

- *rect_APP* : le type de schéma qui permet de représenter les rectangles de l'application médiée
- *cercl_APP* : idem pour les cercles
- *v_carré* : point de vue *carré* sur une entité
- *v_cercle* : point de vue *cercle* sur une entité
- *v_rectangle* : point de vue *rectangle* sur une entité
- *v_élém_coloré* : point de vue *élément coloré* sur une entité
- *v_élém_surface* : point de vue *surface* sur une entité
- *v_élém_pl_gd_long* : point de vue de la *plus grande longueur*
- *v_exp_ref* : point de vue *expression référentielle*

D'autres schémas seraient nécessaires pour décrire l'ensemble du processus incluant l'interprétation de l'énoncé. Nous nous limitons ici à spécifier les schémas qui nous semble illustrer de manière relativement exhaustive une pré-implémentation du modèle de résolution en utilisant le modèle d'interprétation constructionnelle.

Il est de toute façon probable, comme pour la conception d'un programme, que les choix effectués ici doivent être au moins partiellement remis en cause lors d'une future mise en œuvre réelle du modèle, il est particulièrement difficile de vérifier la validité d'un tel système sans pouvoir le tester. Notre objectif ici est plus de donner un aperçu des techniques à utiliser pour passer d'un modèle à son implémentation en version constructionnelle. Ces schémas sont exposés dans les spécifications 10, 11 et 12.

Les schémas de la spécification 12 illustrent des vues génériques permettant de traiter des extracteurs de couleur ou de taille. Pour la taille deux vues sont proposées, soit en fonction de la surface d'un élément, soit en fonction de sa plus grande longueur. L'utilisation de l'un ou l'autre en fonction du contexte reste à déterminer en fonction de facteurs psychologiques, que nous n'avons pas étudié. Ces schémas servent donc d'exemples de ce qui pourrait être fait pour la mise en œuvre du modèle fonctionnel de résolution de la référence.

8.3 Contextes

Les contextes à mettre en œuvre pour décrire le modèle fonctionnel de résolution de la référence sont au nombre de quatre :

schéma rect_APP
rôles
x,y : Réel
largeur,hauteur : Réel
couleur : couleur_APP *// couleur_APP est un triplet (R,G,B)*
contraintes
supérieur(largeur,0)
supérieur(hauteur,0)

schéma cercle_APP
rôles
x,y : Réel
rayon : Réel
couleur : couleur_APP *// couleur_APP est un triplet (R,G,B)*
contraintes
supérieur(rayon,0)

Spécification 10: Schémas nécessaires pour décrire les différents objets du point de vue de l'application

schéma vue
rôles
ref : Référence
refFin : Référence

schéma v_carré
hérite de vue
rôles
long_arête : Réel
contraintes
supérieur(arête,0)

schéma v_cercle
hérite de vue
rôles
rayon : Réel
contraintes
supérieur(rayon,0)

schéma v_rectangle
hérite de vue
rôles
largeur,hauteur : Réel
contraintes
supérieur(largeur,0)
supérieur(hauteur,0)

Spécification 11: Schémas nécessaires pour décrire différentes vues sur des figures géométriques

– *DomaineOrigine* : sans structure particulière, contient les éléments issus du focus précédent dans la résolution (par exemple les éléments possibles choisis par l'action de l'extracteur précédent). Voir spécification 13.
– *DomaineTrié* : doté d'une structure de type liste chaînée simple valuée, c'est-à-dire une liste d'éléments reliés entre eux par un arc auquel on attribue une valeur entre 0 et 1, avec au maximum deux liaisons par élément, sauf pour le premier et le dernier qui n'en ont qu'une. Ce contexte contient tous les éléments provenant de *DomaineOrigine*, triés en fonction de l'extracteur en action. *DomaineTrié* doit en plus disposer de rôles qui spécifient les éléments « intéressants » qui seront utilisés pour le partitionnement. Il peut s'agir d'un élément adjacent au rapport de similarité minimum. Plus de développements sont cepen-

schéma v_élém_coloré
hérite de vue
rôles
comp_rouge : Réel
comp_vert : Réel
comp_bleu : Réel

schéma v_élém_surface
hérite de vue
rôles
surface : Réel

schéma v_élém_pl_gd_long
hérite de vue
rôles
long_max : Réel

Spécification 12: Autres schémas utiles pour le modèle de résolution de la référence

dant nécessaires pour ce point, cependant. Certaines fonctions de partitionnement peuvent en effet s'avérer trop complexes pour se fonder uniquement sur ce type de données. Voir spécification 14.

– *DomainePartPossibles* : doté de la même structure que DomaineTrié, mais avec en plus deux prédicats unaires *Possible()* et *Impossible()*. Les éléments qui seront déterminés comme candidats possibles pour l'extracteur en cours de traitement auront la relation *Possible()*, les autres la relation *Impossible()*. Voir spécification 15.
– *DomainePartPréférés* : idem que *DomainePartPossibles,* mais avec le prédicat *préféré()*.Voir spécification 15.

contexte *DomaineOrigine*
rôles
cardinal **:** entier
dest : DomaineTrié
extracteurRef : Extracteur
lieux
point

Spécification 13: Définition du CONTEXTE *Domaine* qui sert de point de départ au processus de résolution.

8.4 S-Constructions

8.4.1 Pour le tri

Amorçage

Si une instance de contexte *DomaineTrié* est créé, qui est la destination d'une instance de contexte *DomaineOrigine,* alors un élément du domaine d'origine est « copié » dans le domaine

contexte *DomaineTrié*
rôles
cardinal **:** entier
vide : booléen
plein : booléen
dernierMinimum : noeud
extracteurRef : Extracteur
dest : DomainePartPossibles
lieux
noeud
chaîne
relations
rapportSimilarité(elem, elem) $\rightarrow \mathbb{R}^+$
$\forall$x :elem|*rapportSimilarité(noeud(x),suivant(noeud(x)))*≥ 1
opérations
premier(chaîne) $\rightarrow$ *noeud*
dernier(chaîne) $\rightarrow$ *noeud*
cardinal(chaîne) $\rightarrow$ *entier*
suivant(noeud) $\rightarrow$ *noeud*
noeud(elem) $\rightarrow$ *noeud*
chaîne(noeud,noeud) $\rightarrow$ *chaîne*

Spécification 14: Définition du contexte *DomaineTrié*.

contexte *DomainePartPossibles*
hérite de *DomaineTrié*
rôles
premierMinimum : noeud
dest : DomainePartPréférés
relations
possible(elem) $\rightarrow$ *{vrai, faux}*
impossible(elem) $\rightarrow$ *{vrai, faux}*

Spécification 15: Définition du contexte *DomainePartPossibles* contenant la première étape du partitionnement.

contexte *DomainePartPréférés*
hérite de *DomainePartPossibles*
relations
préféré(elem) $\rightarrow$ *{vrai, faux}*

Spécification 16: Définition du contexte *DomainePartPréférés* contenant la dernière étape du partitionnement.

s-construction amorçage-tri-extracteur
contextes
dO **:***DomaineOrigine*
dT **:***DomaineTrié*
constituants
x **:***Ref***@***dO/E*
x' **:***Ref***@***dT/S*
contraintes contenu
dO.dest $\leftrightarrow$ *dT*
dT.vide $\leftrightarrow$ *vrai*
x'.ref $\leftrightarrow$ *x.ref*

Spécification 17: S-construction d'amorçage.

trié[1]. Une fois que le domaine trié contient un élément, les *s-constructions* qui implémentent la fonction de tri de notre modèle de résolution de la référence vont pouvoir être « déclenchées ». C'est la s-construction *amorçage-tri-extracteur* qui effectue cette opération.

S-constructions permettant l'ajout d'un élément du domaine d'origine dans le domaine trié

La spécification 18 donne un exemple de *s-construction* qui effectue le tri en décrivant « où » une entité doit être placée dans le contexte représentant le domaine de référence trié, si il y a deux éléments vus comme des carrés, un dans le domaine trié et un dans le domaine d'origine. Une nouvelle entité est « créée » dans le domaine trié. Comme cette *s-construction* s'applique entre deux carrés, le rapport de similarité entre les deux est de 1.0.

La spécification 19 montre un cas plus intéressant, puisqu'il s'agit de calculer le rapport de similarité entre un élément vu comme un carré et un élément vu comme un rectangle, du point de vue de l'extracteur *carré*. La fonction pour calculer le rapport de similarité a une forme proche de celle de la figure 8.5, où l'abscisse correspond au rapport longueur sur largeur. Il s'agit bien entendu d'une estimation grossière de la fonction « réelle » qui pourrait être déterminée par des expérimentations.

D'autres s-constructions de tri sont nécessaires qui ne sont pas présentées ici. En fait, il faut deux s-constructions pour chaque combinaison possible de comparaisons entre types différents (carré/rectangle, rectangle/carré, carré/cercle,cercle/carré, rectangle/cercle, cercle/rectangle), et une s-construction de tri pour chaque type (carré/carré, rectangle/rectangle, cercle/cercle).

[1]Il s'agit en réalité d'un abus de langage important que de parler de *création*, ou de *copie*. En effet, le modèle d'interprétation constructionnelle est un modèle de **description** de l'interprétation. Les s-constructions ne font que spécifier les contraintes entre des éléments, sans composante dynamique. Cependant, comme la finalité du modèle d'interprétation constructionnelle est d'être réalisé par des opérations du même type que des règles de production, nous nous autorisons ce genre d'abus de langage dans ce chapitre. Le lecteur doit cependant garder à l'esprit que les s-constructions ne décrivent pas une opération, mais des contraintes sur une relation entre plusieurs éléments.

s-construction extracteur-carré-tri-entre-carrés
contextes
dO **:***DomaineOrigine*
dT **:***DomaineTrié*
constituants
xcar **:***v_carré***@***dO/E*
ycar **:***v_carré***@***dT/E*
xcar' **:***v_carré***@***dT/S*
contraintes structure
$dT.rapportSimil(xcar', ycar) \leftarrow 1.0$
contraintes contenu
$dT.premier(ycar)$
$dO.\text{dest} \leftrightarrow dT$
$dO.\text{cardinal} > dT.\text{cardinal}$
$schemaDe(dO.\text{extracteurRef}) \leftrightarrow v\_extracteur\_carré$
$xcar'.ref \leftrightarrow xcar.ref$
$ycar.ref \nleftrightarrow xcar.ref$

Spécification 18: Définition de la s-construction permettant de trier des éléments en fonction de l'extracteur carré, ici entre deux carrés.

s-construction extracteur-carré-tri-entre-carré-rectangle
contextes
dO **:***DomaineOrigine*
dT **:***DomaineTrié*
constituants
xcar **:***v_carré***@***dO/E*
ycar **:***v_rectangle***@***dT/E*
xcar' **:***v_carré***@***dT/S*
contraintes structure
$dT.rapportSimil(xcar', ycar) \leftarrow \frac{min(ycar.largeur, ycar.hauteur)}{max(ycar.largeur, ycar.hauteur)}^{k}$
contraintes contenu
$dT.premier(ycar)$
$dO.\text{dest} \leftrightarrow dT$
$dO.\text{cardinal} > dT.\text{cardinal}$
$schemaDe(dO.\text{extracteurRef}) \leftrightarrow v\_extracteur\_carré$
$xcar'.ref \leftrightarrow xcar.ref$
$ycar.ref \nleftrightarrow xcar.ref$

Spécification 19: Définition de la s-construction permettant de trier des éléments en fonction de l'extracteur carré, entre un carré et un rectangle.

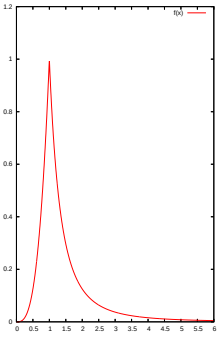


FIG. 8.5 – Forme de la fonction de calcul (avec k=3) du rapport de similarité pour un rectangle dont le rapport longueur sur largeur est figuré en abscisse.

8.4.2 Pour le partitionnement

Concernant les deux phases de partitionnement, la mise en œuvre s'appuie, comme nous l'avons évoqué en début de chapitre, sur les informations calculées directement par les contextes, et plus précisément sur le dernier minimum et le premier minimum (dont des exemples de calcul sont illustrés à la figure 8.6).

Le dernier minimum est le premier noeud du contexte en partant de la droite ayant le rapport de similarité par rapport à son voisin de gauche inférieur aux taux de similarité de tous ses voisins de droite.

Le premier minimum est le premier noeud du contexte en partant de la gauche ayant le rapport de similarité par rapport à son voisin de droite inférieur aux taux de similarité de tous ses voisins de gauche.

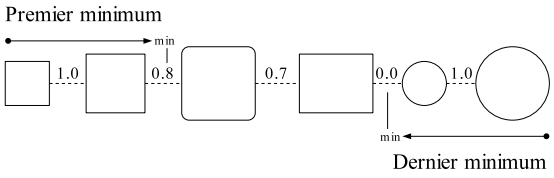


FIG. 8.6 – Détermination des premier et dernier minima.

Amorçage

Comme pour le tri, il faut tout d'abord une s-construction permettant un amorçage. En fait, comme il faut choisir si l'élément d'amorce est dans la partition possible ou dans la partition impossible, il faut par conséquent définir deux s-constructions pour l'amorçage de la première étape de partitionnement (spécification 20), et une pour la seconde étape (spécification 24).

s-construction amorçage-partition-possible-extracteur
contextes
dT **:***DomaineTrié*
dP1 **:***DomainePartPossibles*
constituants
x **:***Ref***@***dT/E*
x' **:***Ref***@***dP1/S*
contraintes structure
dT.rapportSimil(x, dT.dernierMinimum) **< 1**
dP1.possible(x')
contraintes contenu
dT.dest ↔ *dP1*
dP1.vide ↔ *vrai*
dT.plein ← *vrai*
x'.ref ↔ *x.ref*

s-construction amorçage-partition-impossible-extracteur
contextes
dT **:***DomaineTrié*
dP1 **:***DomainePartPossibles*
constituants
x **:***Ref***@***dT/E*
x' **:***Ref***@***dP1/S*
contraintes structure
dT.rapportSimil(x, dT.dernierMinimum) **> 1**
dP1.impossible(x')
contraintes contenu
dT.dest ↔ *dP1*
dT.vide ↔ *vrai*
x'.ref ↔ *x.ref*

Spécification 20: Définition des s-constructions d'amorçage pour les partitions possible et impossible.

S-constructions pour le partitionnement

Pour le reste de l'opération, les s-constructions fonctionnent un peu sur le même principe que pour le tri, à ceci près qu'il faut définir une s-construction par partition de destination. En revanche, le partitionnement fonctionne indépendamment du schéma de l'élément traité (ici, qu'il soit vu comme un carré ou comme un cercle n'importe pas pour les s-constructions). Les spécifications 22, 23 permettent d'établir le partitionnement possible/impossible, tandis que la spécification 24 permet d'établir le partitionnement préférés/possibles.

s-construction amorçage-partition-préféré-extracteur
contextes
dP1 **:***DomainePartPossibles*
dP2 **:***DomainePartPréférés*
constituants
x **:***Ref***@***dP1/E*
x' **:***Ref***@***dP2/S*
contraintes structure
dP1.rapportSimil(x, dP2.premierMinimum) $<$ **1**
dP1.possible(x)$\leftrightarrow$*dP2.possible(x')*
dP1.impossible(x)$\leftrightarrow$*dP2.impossible(x')*
dP2.préféré(x')
contraintes contenu
dP1.dest $\leftrightarrow$ *dP2*
dP1.plein $\leftarrow$ *vrai*
*dP*2.vide $\leftarrow$ *vrai*
x'.ref $\leftrightarrow$ *x.ref*

Spécification 21: Définition des s-constructions d'amorçage pour le partitionnement des éléments préférés.

s-construction extracteur-carré-partition-possibles
contextes
dT **:***DomaineTrié*
dP1 **:***DomainePartPossibles*
constituants
xcar **:***Ref***@***dT/E*
ycar **:***Ref***@***dP1/E*
xcar' **:***Ref***@***dP1/S*
contraintes structure
dP1.rapportSimil(xcar',ycar) $\leftrightarrow$ *dT.rapportSimil(xcar,ycar.ref)*
dT.rapportSimil(xcar, dT.dernierMinimum) $<$ **1**
dP1.possible(xcar')
contraintes contenu
dP1.premier(ycar)
dT.dest $\leftrightarrow$ *dP1*
dT.cardinal $>$ *dP1*.cardinal
schemaDe(dT.extracteurRef*)* $\leftrightarrow$ *v_extracteur_carré*
xcar'.ref $\leftrightarrow$ *xcar.ref*
ycar.ref $\nleftrightarrow$ *xcar.ref*

Spécification 22: Définition de la s-construction permettant de partitionner les éléments possibles pour l'extracteur « carré ».

Conclusion

Dans ce chapitre a été développé partiellement l'implémentation du modèle fonctionnel de résolution de la référence (décrit au chapitre 7) dans le modèle d'interprétation constructionnel

s-construction extracteur-carré-partition-impossibles
contextes
dT **:***DomaineTrié*
dP1 **:***DomainePartPossibles*
constituants
xcar **:***Ref***@***dT/E*
ycar **:***Ref***@***dP1/E*
xcar' **:***Ref***@***dP1/S*
contraintes structure
dP1.rapportSimil(xcar',ycar) $\leftrightarrow$ *dT.rapportSimil(xcar,ycar.ref)*
dT.rapportSimil(xcar, dT.dernierMinimum) $>$ **1**
dP1.impossible(xcar')
contraintes contenu
dP1.premier(ycar)
dT.dest $\leftrightarrow$ *dP1*
dT.cardinal $>$ *dP1*.cardinal
schemaDe(dT.extracteurRef*)* $\leftrightarrow$ *v_extracteur_carré*
xcar'.ref $\leftrightarrow$ *xcar.ref*
ycar.ref $\nleftrightarrow$ *xcar.ref*

Spécification 23: Définition de la s-construction permettant de partitionner les éléments impossibles pour l'extracteur « carré ».

(décrit au chapitre 4). Nous n'avons étudié uniquement les éléments formels nécessaires à la description de l'action d'un extracteur, la combinaison de plusieurs extracteurs n'a pas été développé ici.

Le développement de cette implémentation permet de révéler plusieurs points positifs sur le modèle d'interprétation constructionnelle, mais aussi certains points négatifs. Les points positifs sont les suivants :

– la modélisation d'un processus complexe reste à un niveau de difficulté acceptable,
– la version constructionnelle du processus de résolution fonctionnelle de la référence est parallélisable,
– la modélisation constructionnelle permet de bien structurer la programmation tout en étant générique et extensible. Ainsi si l'on veut par exemple rajouter un type de schéma à prendre en compte pour le tri, il suffit de rajouter les s-constructions nécessaires, sans avoir à toucher à celles existantes.

Les points négatifs de ce développement sont les suivants :

– Dans la configuration actuelle, si l'on veut pouvoir trier des éléments de N types (dans l'exemple traité dans le chapitre, il y avait trois types : carré, rectangle et cercle), il faut $2 \times N!$ s-constructions, ce qui est extrêmement problématique en théorie, même s'il est peu problable d'avoir effectivement une très grande diversité de types (la plupart du temps on pourra regrouper plusieurs cas sous le même type). Une solution possible pour diminuer un peu ce nombre est de passer par un contexte intermédiaire qui contiendra les deux éléments à comparer, ce qui évite de doubler les s-constructions, mais cela ne suffit pas à résoudre

s-construction extracteur-carré-partition-préférés
contextes
dP1 **:***DomainePartPossibles*
dP2 **:***DomainePartPréférés*
constituants
xcar **:***Ref***@***dT/E*
ycar **:***Ref***@***dP1/E*
xcar' **:***Ref***@***dP1/S*
contraintes structure
dP2.rapportSimil(xcar',ycar) $\leftrightarrow$ *dP1.rapportSimil(xcar,ycar.ref)*
dP1.rapportSimil(xcar, dP1.premierMinimum) $<$ **1**
dP2.préféré(xcar')
dP1.possible(xcar) $\leftrightarrow$ *dP2.possible(xcar')*
dP1.impossible(xcar) $\leftrightarrow$ *dP2.impossible(xcar')*
contraintes contenu
dP2.premier(ycar)
*dP1.*dest $\leftrightarrow$ *dP2*
*dP1.*cardinal $>$ *dP2.*cardinal
*schemaDe(dP1.*extracteurRef*)* $\leftrightarrow$ *v_extracteur_carré*
xcar'.ref $\leftrightarrow$ *xcar.ref*
ycar.ref $\nleftrightarrow$ *xcar.ref*

Spécification 24: Définition de la s-construction permettant de partitionner les éléments impossibles pour l'extracteur « carré ».

le problème.

– La vérification de l'extracteur en cours de traitement se fait, dans nos exemples, dans les contraintes sur le contenu, ce qui signifie que toutes les s-constructions de tri définies devront être « vérifiées », et par conséquent qu'un grand nombre de vérifications inutiles seront effectuées. Il serait plus judicieux de trouver un moyen pour que cette vérification se fasse au niveau des constituants des s-constructions.

L'objectif de ce chapitre était de montrer qu'il était possible, avec un modèle de description de l'interprétation inspiré de la grammaire de construction et incluant la prise en compte de contextes structurant les informations, de décrire un processus complexe de résolution de la référence. C'est donc chose faite. La question est maintenant de savoir comment un tel modèle peut ensuite être implémenté dans un système opérationnel. C'est ce que nous allons voir dans le prochain chapitre, avec la présentation du le modèle d'interprétation constructionnelle par observateurs (MICO), qui est la spécification logicielle permettant de mettre en œuvre le modèle d'interprétation constructionnelle.

Troisième partie

Implémentations et Réalisations

Chapitre 9

Implémentation du Système d'Observateurs : MICO

Le modèle d'interprétation constructionnelle décrit au chapitre 3 est un modèle **descriptif**. Sa fonction est donc de décrire formellement et de manière statique les contraintes sur les entités manipulées lors de l'interprétation d'un énoncé dans une situation donnée. Le processus d'analyse en lui-même n'est pas directement dérivable de cette description, puisqu'elle est équivalente à une grammaire contextuelle, dont on sait qu'elle ne peut être implémentée informatiquement de manière efficace, dans l'absolu. Pour autant, les travaux effectués sur l'implémentation de la grammaire ECG (Bryant, 2003), ceux concernant l'utilisation de systèmes de production pour l'analyse de la langue (Engel, 2002), ainsi que le modèle des observateurs (Pitel and Sansonnet, 2003b) lui même inspiré des travaux de Small (1980) déterminent les grandes lignes de ce que pourrait être une telle implémentation.

Les linguistes utilisent certes la notion de construction depuis quelque temps afin de modéliser de nombreux phénomènes, mais jusqu'à maintenant, ces constructions sont d'abord traduites dans un formalisme comme HPSG, avant d'être finalement transformée en règles hors contexte afin d'être implémentées informatiquement. L'inconvénient de cette approche est qu'elle produit une quantité de règles qui croît exponentiellement. L'avantage d'utiliser une grammaire hors contexte, qui peut être traitée en temps polynomial relativement au nombre de règles, a donc quasiment disparu. Il ne reste que l'aspect décidable de ces grammaires qui soit vraiment intéressant, mais c'est un aspect qui est relativement éloigné de la nature même de la langue.

Il nous faut donc proposer un système d'interprétation qui soit capable de traiter les **s-constructions** de manière efficace, en s'éloignant catégoriquement des approches fondées sur des grammaires hors contexte, utilisées dans les approches inspirées du génerativisme de Chomsky. Il va donc être nécessaire de s'intéresser aux approches orientées vers la robustesse, même si ce n'est pas une préoccupation première de notre système, car ces approches ont comme caractéristique de ne pas chercher systématiquement une solution d'analyse complète et absolument sûre. A ce titre, l'analyse robuste par tronçons (*chunk parsing*) proposée par Abney (1991) est particulièrement intéressante, car elle part du principe que les relations de dépendance entre clauses d'un énoncé ne doivent pas être identifiées lors d'une première étape d'analyse. Il arrive en effet très souvent que certaines relations d'attachement syntaxique nécessitent, pour pouvoir être déterminées, de disposer d'une analyse sémantique des composants, comme par exemple pour le rattachement des syntagmes prépositionnels. Par

conséquent, dans l'approche d'Abney, l'analyse se contente initialement de reconnaître certains syntagmes et de leur associer une analyse de surface, sans construire d'arbre syntaxique complexe, et donc en laissant les tronçons détachés les uns des autres. Afin de conduire une analyse complète, l'analyse par tronçons est organisée de manière stratifiée, une strate pour les tronçons, une pour les clauses, et une dernière pour les énoncés. L'attachement entre les tronçons est effectué dans la deuxième strate, en utilisant à la fois les informations syntaxiques et les informations sémantiques.

L'implémentation de la grammaire ECG (Bergen and Chang, 2002) proposée par Bryant (2003) est construite principalement autour de cette notion, en considérant les constructions comme des « *chunk recognizers* » de Abney. Nous adoptons une stratégie similaire, mais en approfondissant les questions de stratification de l'analyse, puisque nous avons postulé que l'interprétation doit s'appuyer sur des conteneurs d'information que nous avons appelé **contextes**, et qui peuvent bien entendu représenter les strates de l'analyseur par tronçons. Notre approche s'appuie de plus sur la notion de système de production, issue des travaux de Newell (1973) et exposée au chapitre 2, et sur l'utilisation qui en a été faite dans SPIN (Engel, 2002) pour le traitement des énoncés. Ces travaux montrent en effet la nécessité pour une approche comme la nôtre de prendre en compte l'ordonnancement des règles de production afin de mener une interprétation qui ne soit pas trop coûteuse en temps.

De plus, l'originalité majeure de notre modèle impose d'autre contraintes. Comme les contextes contenant les instances de schémas sont structurés par une topologie, il nous faut un modèle de traitement des contraintes qui soit efficace tout en étant générique. A notre avis, les bénéfices à tirer de cette plus grande expressivité compensent selon nous le surcoût qu'elle impose. Les arguments que nous avançons dans le sens de l'intérêt de cette méthode sont les suivants :

– l'intégration des données contextuelles dans le processus d'analyse devrait aider à guider celui-ci en éliminant plus rapidement les candidats inadéquats. Si l'on peut prendre en compte les éléments visibles à l'écran, certaines ambiguïtés syntaxiques ou sémantiques pourront être levées plus tôt.
– la répartition des éléments à analyser en ensembles structurés devrait permettre de guider localement l'interprétation. En effet, là où un système non structuré ne peut que chercher globalement une certaine information pour pouvoir compléter le motif attendu par une règle, un modèle ou chaque composante de l'interprétation est structurée peut choisir de rechercher cette information localement (dans une sous-partie précise d'un contexte), et donc déclencher moins de règles inutiles.

Le modèle que nous proposons a été baptisé M.I.C.O. pour **Modèle d'Interprétation Constructionnelle basé sur les Observateurs**. C'est un modèle qui n'est pas totalement achevé, notamment parce que la gestion des contraintes ne pourra être validée avec une certitude raisonnable qu'après avoir mis en place un certain nombre de contextes supportant des topologies variées.

M.I.C.O. ne définit que l'architecture d'un système d'interprétation constructionnelle, mais ne présente aucun choix particulier sur la manière dont l'interprétation doit avoir lieu, c'est-à-dire notamment quels sont les schémas utilisés et les constructions qui devront être implémentées.

Nous présentons dans ce chapitre le **moteur de traitement des observateurs**, ainsi que le **modèle objet** que nous avons défini pour le système.

Comme nous l'avons déjà évoqué au chapitre 4, les **observateurs** sont les composants infor-

matiques permettant d'implémenter les s-constructions de MIC. Nous allons établir tout au long de ce développement les caractéristiques qu'ils devront respecter.

9.1 Moteur de traitement

9.1.1 Discussion

Comme les *s-constructions* décrivent les contraintes qui doivent être vérifiées par leurs constituants, on pourrait penser dans un premier temps qu'une approche inspirée de la programmation par contraintes pourrait permettre de mettre en œuvre une résolution efficace. Une telle approche a d'ailleurs été envisagée par Blache (2000) afin de mettre en œuvre une analyse des grammaires syntagmatiques d'unification comme HPSG.

Pourtant, une modélisation suivant les techniques de programmation par contraintes n'est pas adaptée dans notre cas, car on cherche non pas à résoudre un ensemble de contraintes, mais plutôt à trouver quel sous-ensemble des contraintes est le plus respecté par une situation donnée. En effet, chaque *s-construction* définit un sous-ensemble de contraintes et si ces contraintes sont respectées par les valeurs présentes dans une base d'informations, alors la *s-construction* va introduire de nouvelles valeurs dans cette base.

L'approche de programmation par contraintes n'est donc pas appropriée, il faut plutôt envisager une approche dynamique proche des systèmes de production de Newell, mais mêlant règles de réécriture et processus d'unification des valeurs. De plus, la notion de *contexte* peut être rapprochée des niveaux dans les blackboards (Hayes-Roth, 1985), même si l'aspect séquentiel de ceux-ci disparaît dans le modèle d'interprétation constructionnelle. Les constructions peuvent dans ce cadre être comparées à des sources de connaissance, capables de produire certaines informations (des instances de schéma situées dans des contextes) en réponse à l'observation d'autres informations (idem).

La solution que nous proposons est de mettre en œuvre les *s-constructions* comme des unités de traitement agissant comme des règles de réécriture, ayant une activation progressive et auto-propagatrice, à la fois réactive et proactive. Ces unités seront appelées *observateurs.*

Activation Réactive/Proactive L'activation réactive/proactive est dérivée des approches ascendant/descendant (en anglais *top-down*/*bottom-up*). Le principe est qu'une règle est prise en compte **à la fois** lorsque sa partie gauche est remplie et lorsque sa partie droite est **attendue**, et non plus seulement lorsque sa partie gauche est remplie (approche réactive seule), ou seulement lorsque sa partie droite est attendue (approche proactive seule).

Généralement cette méthode d'analyse est utilisée avant tout afin de limiter l'espace de recherche des solutions. Pour notre système, cette approche est cependant réellement indispensable. En effet, d'après notre *théorie des points de vue*, les relations ontologiques sont mises en œuvre de la même manière que les « règles de grammaire », c'est-à-dire sous la forme de *s-constructions*. Or il n'est pas imaginable de produire tous les *points de vue* possibles pour une information donnée, car non seulement cela entraînerait une explosion combinatoire au niveau de la quantité d'information et des règles déclenchées, mais surtout cela introduirait des ambiguïtés là où il n'y en avait pas. Donc le fait qu'une certaine information A *puisse*

être vue comme une information B ne doit pas déclencher une règle dès que A est présent. En revanche la règle doit être déclenchée si B est attendu et que A est présent. Donc il s'agit bien d'utiliser le mécanisme d'activation « à la demande » qui soit à la fois réactif dans certains cas et proactif dans d'autres cas.

Dans notre approche, cela se traduit par l'introduction d'**un mécanisme d'attente**, permettant à une règle de signifier qu'elle attend une certaine information à un certain endroit dans un certain contexte. Le choix de l'observateur qui doit être exécuté à un instant donné va donc être déterminé à la fois par les éléments présents qui sont attendus dans son côté *observation*, mais également par les éléments attendus par d'autres observateurs, et que l'observateur est capable de *produire*.

Le mécanisme d'attente est supporté par des informations particulières de la forme EXPECT(SCHÉMA) qui sont stockés comme les instances de schéma, dans des instances de contexte. Cette information est utilisée pour favoriser les observateurs qui produisent des instances de schéma du bon type en considérant implicitement que tous les observateurs ont une contrainte mineure[1] attendant la présence d'un tel signal d'attente. Les observateurs produisant ce type de schéma auront donc un score de satisfaction plus élevé que les autres.

Activation progressive L'application du principe réactif/proactif implique qu'une partie du problème soit de savoir quels sont les éléments attendus par les autres observateurs. Dans les grammaires classiques qui s'arrêtent au niveau syntaxique, la séquence analysée est généralement la phrase, on peut donc partir de cet objectif unique pour savoir quels sont les éléments à trouver par la suite. Dès l'instant où l'on ne considère plus seulement l'aspect syntaxique et où l'on peut en plus utiliser des connaissances de plus haut niveau pour contraindre d'avantage le type d'énoncé attendu (par exemple, dans un dialogue, on peut anticiper le type d'une réponse à partir des énoncés passés), il faut envisager une approche différente.

Toute règle de réécriture peut être considérée comme composée de deux parties : la partie observation et la partie production. Chaque partie est composée soit de zéro élément (la règle est soit toujours activable, par exemple pour une valeur par défaut, soit ne produit rien, ce dernier cas étant assez peu probable, sauf à des fins de conception), soit de un ou plusieurs éléments. Si la partie observation est composée de plus d'un élément, la présence de chaque élément va renforcer la probabilité que la règle soit finalement déclenchée. L'**activation progressive** des constructions permet de pré-activer une construction lorsqu'une partie des éléments qu'elle s'attend à observer est présente. Ce principe se traduit par l'attribution à chaque observateur en cours d'exécution d'un potentiel d'activation, calculé à partir de son taux de satisfaction de contraintes et de son potentiel informatif.

Prise isolément, cette caractéristique n'a aucun intérêt, si ce n'est éventuellement à des fins de conception, pour connaître à un moment donné de l'analyse quelles sont les constructions les plus à même d'être déclenchées dans les prochains cycles. En revanche, combinée avec la possibilité de propager le « potentiel d'activation » des observateurs, cela permet de sélectionner rapidement les voies qui ont la probabilité la plus forte de produire une analyse juste.

Activation propagatrice La technique d'activation proactive/réactive nécessite l'utilisation d'un mécanisme d'attente, qui se traduit par le fait qu'un observateur ayant un potentiel

[1] C'est-à-dire une contrainte qui peut ne pas être satisfaite.

d'activation non nul doit pouvoir produire un message (par l'intermédiaire d'un contexte) indiquant qu'il attend d'autres éléments pour pouvoir être effectivement exécuté. Il serait intéressant de pouvoir contrôler ce principe en rendant possible dans certains cas une propagation du message d'attente aux degrés 2 et plus. Par exemple, soit :

– un observateur X observant un motif composé des types A, B et C et produisant D,
– un observateur Y observant E et produisant A, et
– un observateur Z observant F et produisant E.

La situation illustrée figure 9.1 s'explique ainsi :

– si *X* observe B et C, alors il attend une instance de schéma de type A, qui peut être produit par *Y*.
– *Y* ne trouvant pas d'instance de type E, il produit un message d'attente sur E.
– *Z* va alors lui aussi être activé, produisant E, ce qui permet à *Y* de produire A, et donc à X de produire D

Le principe de la propagation d'activation est le suivant : tout d'abord pour les activation en observation, soit un observateur observant les éléments A, B et C et produisant l'élément D ; si A est présent, alors l'observateur produit EXPECT(B) et EXPECT(C), les observateurs produisant B ou C vont alors voir leur potentiel d'activation augmenter. Si le potentiel d'activation d'un de ces observateurs atteint un certain seuil, il va à son tour produire un message EXPECT(), et ainsi de suite. Inversement, pour les activations en production, le même observateur va, s'il a un potentiel d'activation non nul, proposer sa production aux autres observateurs, en générant un message PROPOSE(D), message qui à son tour pourra partiellement activer un autre observateur.

Bien entendu, si ce mécanisme est implémenté tel quel, il y a un fort risque d'explosion combinatoire. Il faut donc mettre en place les gardes-fous nécessaires pour empêcher une surproduction de tels messages. Ceci est assuré par un calcul du potentiel d'activation et par un calcul du potentiel informatif des observateurs qui produisent les messages d'attente. Plus le potentiel d'activation et le potentiel informatif sont élevés, plus le message a de chance d'être propagé.

9.1.2 Schéma général

L'architecture du système permettant d'exécuter les observateurs est schématisée par ses principales structures de données dans la figure 9.2. Le moteur de traitement repose sur plusieurs familles de structures :

– les définitions portant sur les schémas, les contextes et les observateurs,
– l'espace de travail qui contient les instances de contexte et les instances de schémas contenues dans les contextes,
– des structures de pile contenant les instanciations d'observateurs (dans différents états)
– une structure de pile contenant les événements concernant l'espace de travail (c'est-à-dire les modifications apportées aux instances de contexte).

Principe d'exécution Le moteur de traitement suit un fonctionnement par **cycles**, un cycle correspondant grossièrement au trajet décrit par la boucle de flèches épaisses dans la

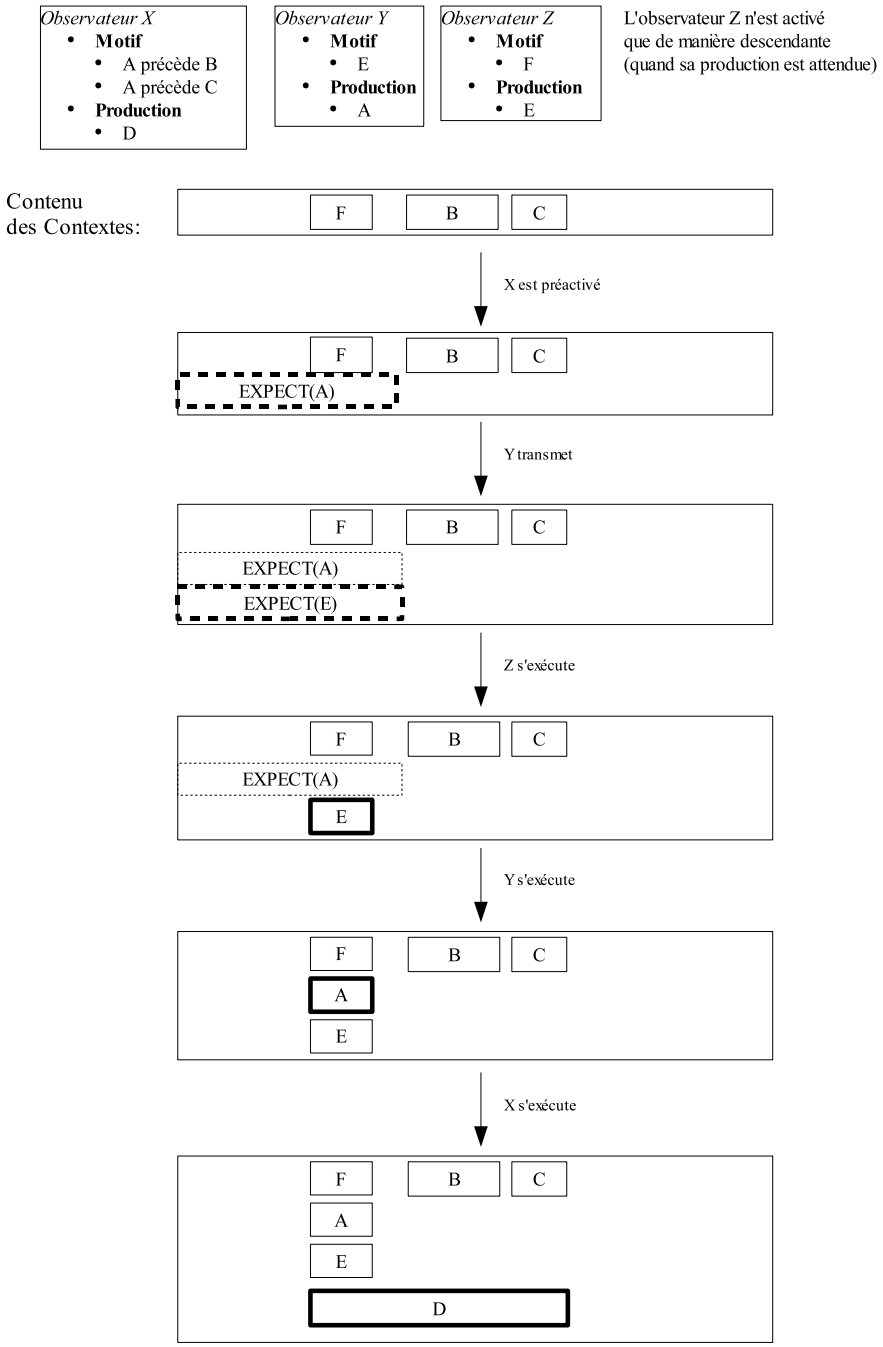


FIG. 9.1 – Activation propagatrice avec les contextes.

figure 9.2. A chaque cycle, des informations nouvelles sont entrées dans l'espace de travail, les observateurs qui peuvent produire un nouveau *point de vue* à partir de ces informations sont déclenchés, et produisent à leur tour des informations nouvelles dans l'espace de travail. Afin d'assurer l'interface avec le monde, certaines informations peuvent provenir de l'extérieur, et d'autres peuvent déclencher une action.

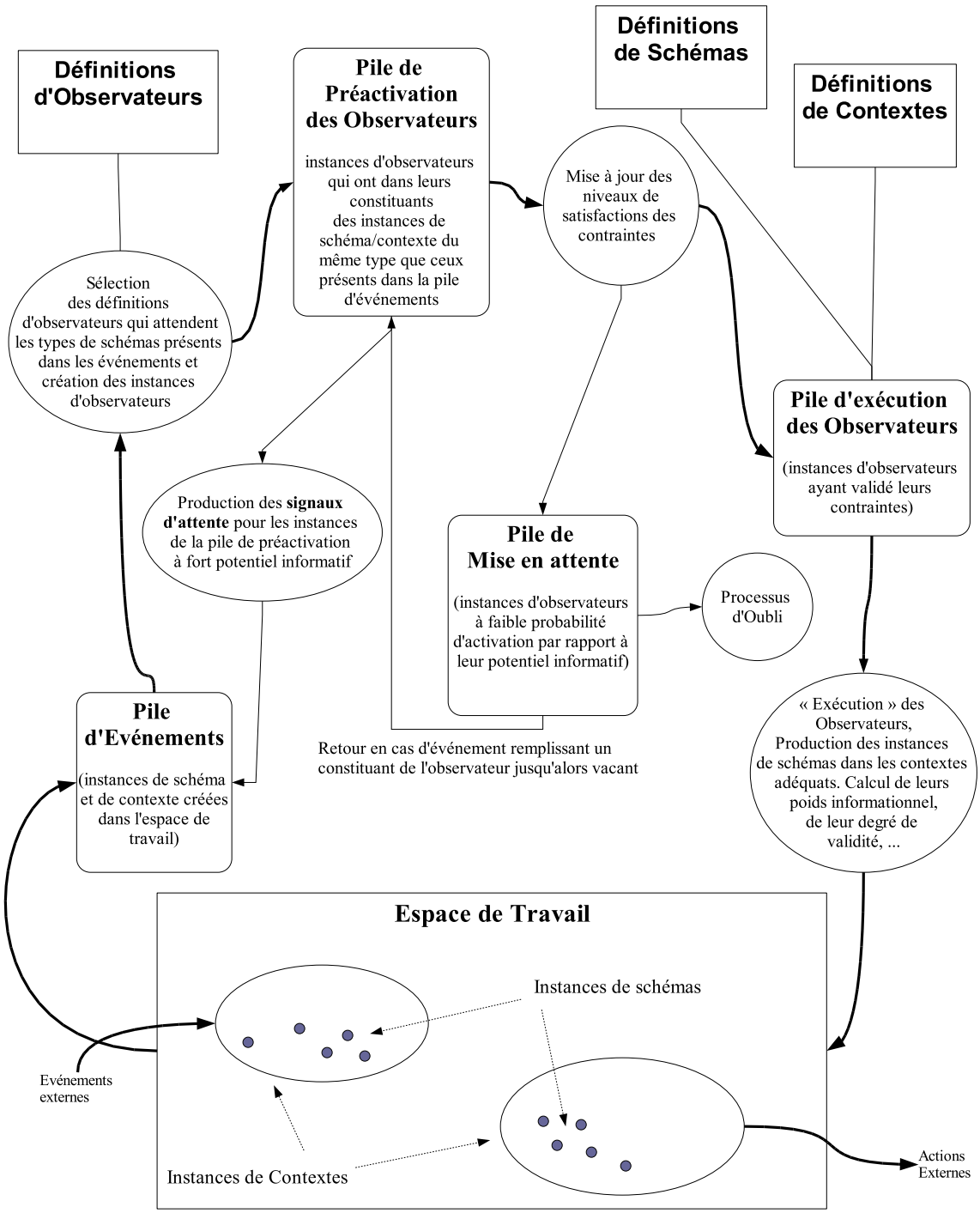


FIG. 9.2 – Principales structures de données du moteur de traitement du modèle d'interprétation constructionnelle par observateurs.

Définitions Les définitions des schémas, des contextes et des observateurs consistent en des représentations des différentes structures de données présentées au chapitre 3. Elles correspondent, dans le modèle UML du système (voir section 9.2), aux classes suffixées par « Def ». Une définition est comparable à une classe en langage objet, et elle permet de créer une instance, comme un moule permet de créer un objet d'une certaine forme. Pour les schémas, les contextes et les observateurs, il est possible d'avoir plusieurs instances à partir d'une même définition :

– pour les schémas, parce qu'un point de vue d'une certain type peut être produit à différentes occasions, à partir de différents points de vue.
– pour les contextes, parce que plusieurs structures de même topologie peuvent être présentes en même temps (par exemple si on a deux fenêtres d'interface graphique à l'écran, elles peuvent être vues comme des structures d'un certain type incluses dans une autre structure, du même type que les deux premières, et qui représenterait l'écran entier).
– pour les observateurs, parce qu'un même motif peut être trouvé à différents endroits de l'espace de travail, et donc entraîner le déclenchement du même type d'observateur plusieurs fois à un même moment.

L'espace de travail Les instances créées à partir des définitions de schémas et de contextes forment l'espace de travail. L'espace de travail peut être vu d'une part comme une structure arborescente, dont les noeuds non terminaux sont des instances de contextes, et dont les noeuds terminaux sont des instances de schémas. D'autre part, la possibilité de décrire des liens entre (a) les instances de schémas ou entre (b) les instances de contextes, ou encore entre (c) les instances de schémas et les instances de contexte, est supportée dans la représentation de l'espace de travail par une table d'association *identifiant/instance* qui associe un identifiant unique à une certaine instance de schéma ou de contexte, permettant de supporter le mécanisme de référence.

L'espace de travail contient enfin les « attentes » qui sont au même niveau que les instances de contextes, et qui sont donc des noeuds terminaux. Les *attentes* symbolisent le fait qu'un schéma d'un certain type est attendu par une instance d'observateur en état de préactivation afin de compléter son motif de constituants. Grâce aux contraintes structurelles du motif de l'observateur, une attente n'est pas nécessairement posée sur l'ensemble d'un contexte, mais peut être restreinte à une *zone* de taille très restreinte. Une zone peut être définie de deux manières :

– en extension, avec une liste de positions possibles,
– en semi-intension, grâce à une structure propre au contexte. Par exemple, pour un contexte avec une topologie à une dimension, un couple de deux ordonnées peut spécifier l'ensemble des points situés entre ces deux extrêmes.

L'espace de travail se traduit finalement par deux structures de données, une structure arborescente qui décrit les inclusions structurelles, et une table d'association qui décrit le graphe de relation des schémas entre eux.

Piles d'observateurs Les instances créées à partir des définitions d'observateurs sont contenues dans les piles de préactivation ou d'exécution. Une instance d'observateur est ajoutée à la pile de préactivation lors de la création dans l'espace de travail, d'une instance de

schéma dont le type correspond à un des constituants de l'observateur. Si un observateur qui est déjà dans la pile est « intéressé » par une des instances de schéma nouvellement arrivée, son statut est mis à jour à condition que cet ajout ne viole pas une contrainte majeure de l'observateur.

Pile d'événement La pile d'événements contient quant à elle des données structurées qui décrivent les modifications de l'espace de travail (provenant soit de modalités extérieures, soit de l'activité des observateurs). Les événements recensés dans cette pile concernent :

– les nouvelles instances de contextes,
– les nouvelles instances de schémas,
– les nouveaux messages d'attente d'un certain schéma,
– la suppression d'un message d'attente.

Le début de l'interprétation d'un énoncé est décrit par la figure 9.3. Comme le modèle d'interprétation constructionnelle par observateurs ne spécifie aucune contrainte sur le type d'analyse à mettre en œuvre, il ne s'agit que d'un exemple possible d'utilisation du modèle. On y distingue une première étape de *chunking,* ou analyse par tronçons, qui produit notamment une alternative sur le mot « le » qui peut être considéré, s'il est pris isolément, comme un pronom, tandis qu'il forme un tronçon NP s'il est combiné avec « carré ».

9.1.3 Principes de fonctionnement

Pondérations

Comme nous l'avons évoqué précédemment, le moteur de traitement de M.I.C.O. s'appuie sur deux types de pondérations : le **potentiel d'activation** et le **potentiel informatif**. Ces deux poids permettent de figurer, pour le premier, la probabilité estimée qu'une certaine instance d'observateur soit finalement exécutée, et pour le second, la prévision du gain pour le système si l'instance était exécutée.

Le **potentiel d'activation** est calculé en fonction des contraintes satisfaites dans une instance d'observateur en phase de préactivation. Pour calculer précisément ce potentiel d'activation, il faut que chaque contrainte soit affectée d'un poids relevant son importance relativement aux autres contraintes. Le potentiel d'activation correspond à la somme des poids des contraintes satisfaites divisé par la somme de toutes les contraintes d'un observateur.

Le **potentiel informatif** d'une instance d'observateur est calculé d'une part par une valeur propre inscrite dans la définition d'observateur correspondante, et d'autre part en évaluant la somme des **capacités informatives** des consistuants de l'instance. La capacité informative d'une instance de schéma est calculée lors de sa production en faisant la somme des capacités informatives des constituants de l'instance d'observateur qui la produit. Ceci implique que les instances de schéma introduites directement depuis l'extérieur ont une capacité informative fixée arbitrairement. Lorsque l'on veut évaluer la capacité informative d'un constituant non lié, il faut utiliser la capacité informative par défaut renseignée dans la définition du schéma correspondant au constituant.

Les messages d'attente transmettent aussi une capacité informative, mais dans ce cas, il s'agit de la part relative du constituant dans le potentiel informatif de l'instance d'observateur qui

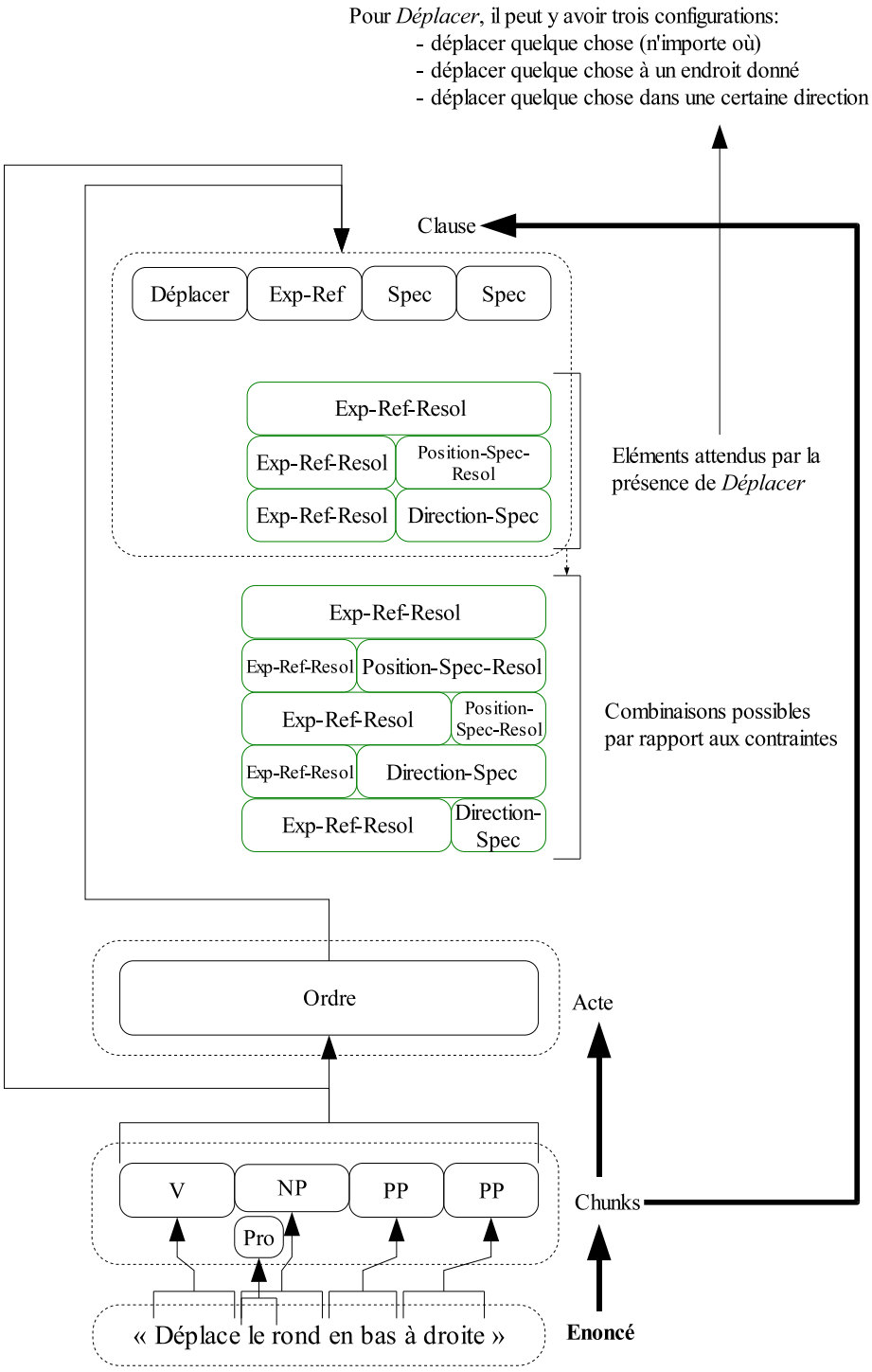


FIG. 9.3 – Premières étapes de l'interprétation d'un énoncé.

a produit le message d'attente. Par exemple si une instance d'observateur ayant un potentiel

informatif de 6 attend deux constituants A et B, et que A a une capacité informative de 2 et B une capacité informative de 1, alors le message d'attente de A aura une capacité informative de $\frac{6\times2}{2+1} = 4$ tandis que le message d'attente de B aura une capacité informative de $\frac{6\times1}{2+1} = 2$.

Mise en attente

Si une instance d'observateur préactivée reste pendant un certain temps sans qu'aucun événement n'ait modifié son état, elle est mise en attente, ce qui signifie que les messages d'attente qu'elle a produits sont supprimés. L'instance est cependant conservée en mémoire afin de garder les informations calculées sur ses contraintes.

Le délai de mise en attente d'une instance d'observateur dans la Pile de Préactivation est calculé à partir d'une valeur inscrite arbitrairement dans la définition d'observateur correspondante et du potentiel d'activation de l'instance d'observateur.

Gestion de l'oubli

Oubli des instances d'observateurs Lorsqu'une instance d'observateur mise en attente dépasse un certain « âge », c'est-à-dire qu'un certain nombre de cycles est passé depuis que l'instance a été mise dans la pile d'attente, alors cette instance est « oubliée ». Autrement dit, elle est supprimée totalement des données du moteur de traitement. Ceci ne pose pas réellement de problème de perte d'information, car même si un événement survient ensuite qui aurait pu l'activer, le processus repartira simplement de zéro, ce qui fera simplement perdre un peu de temps au système.

L'âge limite d'une instance d'observateur dépend d'une variable propre à la définition de l'observateur et de son potentiel d'activation.

Oubli des instances de contexte et de schéma Les instances de contexte et de schéma, tout comme les instances d'observateur, doivent pouvoir être oubliées. Les instances de contextes sont oubliées lorsqu'ils ne contiennent plus aucune instance de schéma et que leur âge limite, fixé arbitrairement dans la définition du contexte, est dépassé (l'âge correspond à la date courante en numéro de cycle moins la date de dernière modification). Les instances de schéma sont oubliées lorsque leur durée de vue inutile a dépassé leur âge limite, lui aussi fixé arbitrairement dans la définition du schéma (la durée de vie inutile correspond à la date courante moins la date de dernière utilisation).

9.1.4 Fonctionnement

Description du cycle

A chaque début de cycle, la pile d'événements est traitée événement par événement. Pour l'arrivée d'une nouvelle instance de schéma, la procédure suivante est respectée :

1. Une recherche est lancée dans une liste d'associations (schéma, observateur), stockée sous forme d'une table de hachage, afin de trouver les types d'observateurs qui ont un constituant dont le type correspond au type de schéma nouvellement arrivé. Le système crée une liste de traitements d'événement, avec une entrée par observateur concerné.
2. Parmi les instances d'observateur de la Pile de Préactivation, le système cherche celles à qui il manque un constituant du type du nouveau schéma. S'il en trouve alors elles sont complétées, le système ajoute à la liste de traitements d'événement une entrée par instance d'observateur concerné. Il calcule ensuite la satisfaction des contraintes affectées par cette nouvelle donnée.
   - Si une contrainte est violée par le nouveau constituant, l'instance d'observateur revient à son état d'origine et l'entrée correspondante de la liste de traitements d'événement est conservée.
   - Sinon, l'instance d'observateur est mise à jour et l'entrée de la liste de traitements est supprimée, le traitement est « consommé ».
3. La même opération de mise à jour est ensuite effectuée dans la Pile de Mise en Attente. Si des instances d'observateur sont ainsi mises à jour, elles remontent dans la Pile de Préactivation, et les messages d'attente qu'elles peuvent produire sont recréés.
4. Enfin, de nouvelles instances d'observateurs sont créées à partir de la liste des traitements d'événement finale et des définitions d'observateurs correspondantes.

Algorithme 2: Traitement de l'arrivée d'une nouvelle instance de schéma dans une instance de contexte.

Création de nouvelles instances d'observateurs

Lorsqu'une nouvelle instance d'observateur doit être créée en réponse à un événement, elle n'a qu'une seule instance de contexte et une seule instance de schéma *liées*. Celle-ci doit alors « trouver » un maximum de constituants acceptables dans les contextes existants. Pour cela, il faut tout d'abord *lier* le maximum d'instances de contexte qui satisfont les contraintes. Si des contraintes sur les instances de contexte sont décrites dans la définition de l'observateur, le système cherche toutes les instances de contexte qui les satisfont, sinon, toutes les instances de contexte ayant le bon type sont sélectionnées. Pour chaque combinaison d'instances de contexte satisfaisante, une instance d'observateur est créée à partir de l'instance initiale qui avait une seule instance de contexte et une seule instance de schéma liées.

La procédure est ensuite la même pour la liaison des instances de schéma, une nouvelle instance d'observateur est créée pour chaque combinaison de liaison satisfaisant les contraintes.

Ceci pose bien sûr un problème de combinatoire, mais il est de notre point de vue du ressort du concepteur de l'observateur de s'assurer que les contraintes posent suffisamment de restrictions pour éviter une surproduction d'instances d'observateur.

9.2 Modèle objet du système

Nous avons décrit une partie du modèle du système en UML, en prenant en compte le fait qu'une partie des objets doit permettre de définir la composition des schémas, contextes et

observateurs, et qu'une partie doit permettre leur utilisation dans le modèle de traitement. Le nom des classes de définition possède le suffixe « Def », celui des classes pour l'utilisation possède le suffixe « Instance ». Les figures 9.4, 9.5 et 9.6 exposent respectivement les représentations UML des schémas, des contextes et des observateurs.

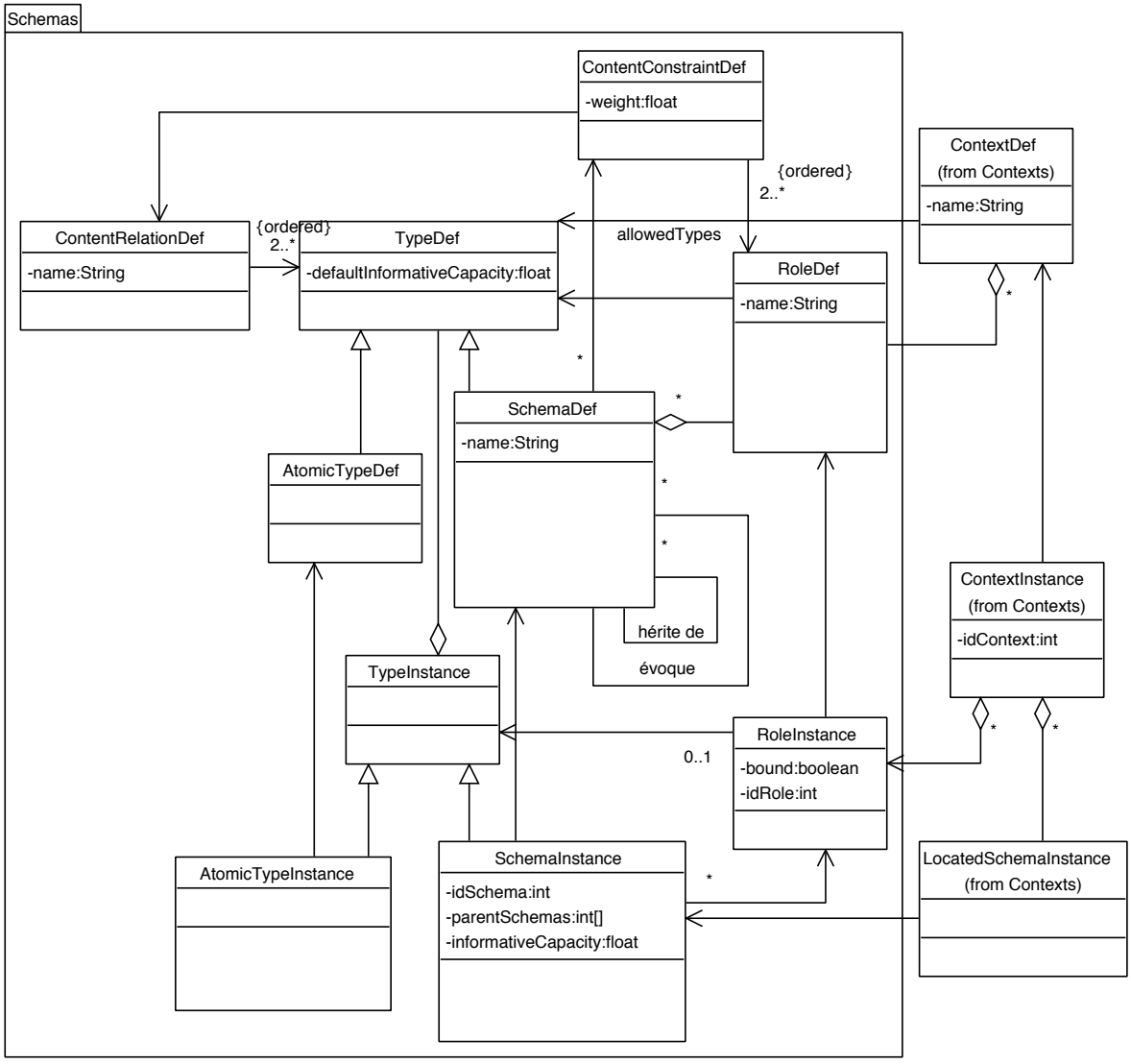


Fig. 9.4 – Définition UML des schémas

Ces représentations UML ne décrivent pas l'ensemble d'un système opérationnel, il manque notamment une description des contraintes, qui sont particulières à notre système car elles doivent non pas permettre de trouver une configuration acceptable de schémas, mais :

1. vérifier qu'un ensemble de constituants (des schémas localisés dans des contextes) respectent bien certaines contraintes,
2. si au moins un constituant n'est pas lié (c'est-à-dire que les contraintes ont commencé à être évaluées alors que l'ensemble des éléments nécessaires à la production ne sont pas

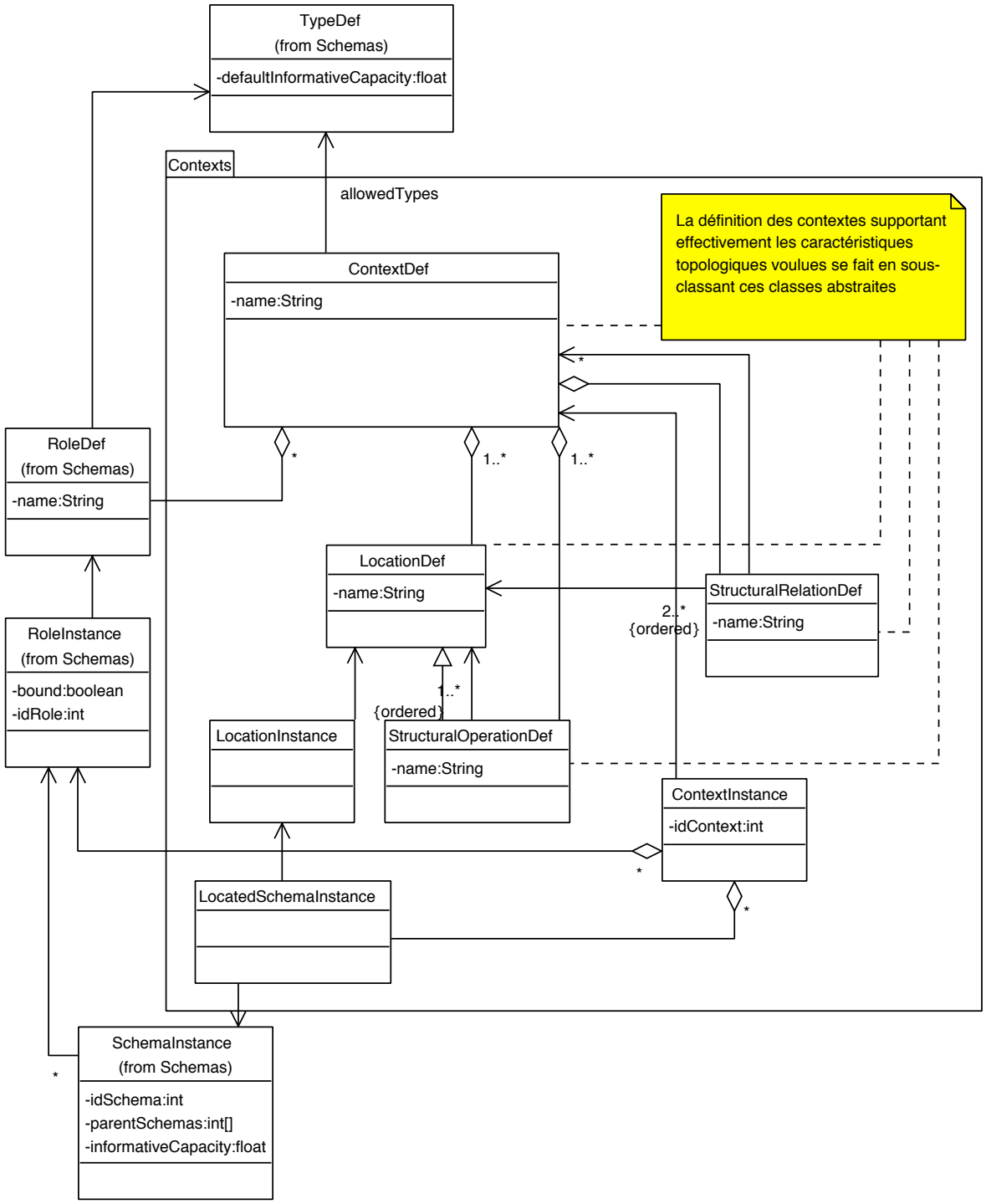


FIG. 9.5 – Définition UML des contextes

présentes), alors les contraintes doivent permettre de fournir un ensemble de lieux dans un contexte donné, où le constituant est « attendu ». Ceci permettra ensuite de poser un message d'attente dans la zone du contexte en question précisant le type du schéma

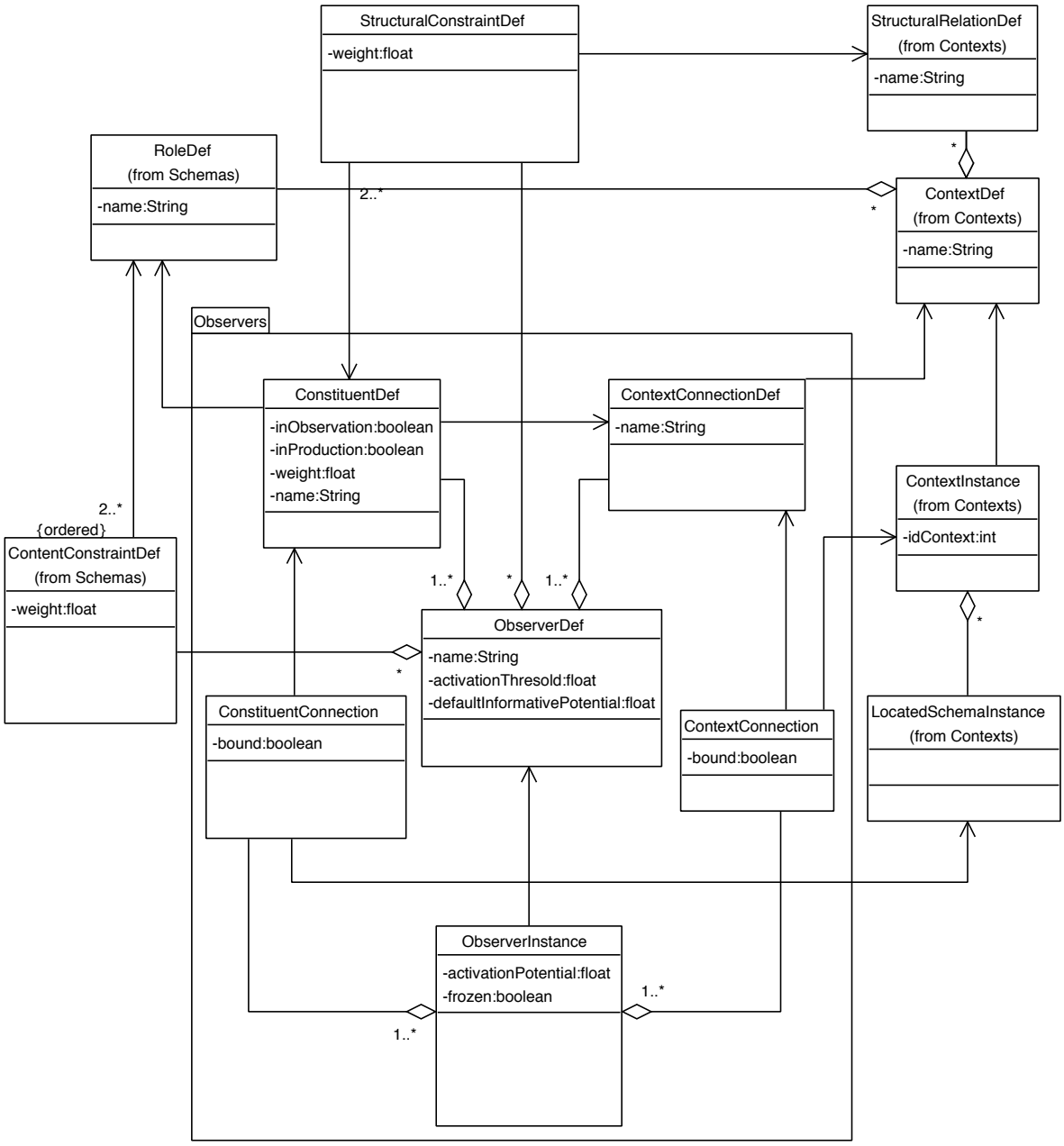


FIG. 9.6 – Définition UML des observateurs

attendu.

9.3 Contraintes

Le modèle d'interprétation constructionnelle, héritant en cela de la grammaire ECG, repose sur une description par contraintes. Les observateurs qui implémentent les s-constructions

vont donc, naturellement, eux aussi reposer sur les contraintes pour décrire les motifs qu'ils attendent dans les contextes, ainsi que la production de nouvelles instances de schéma.

La problématique de satisfaction de contraintes est beaucoup trop vaste pour que nous puissions faire dans cette thèse, un apport même minime à la question. Nous nous bornons ici à dresser un panorama des techniques qu'il faudra utiliser dans l'implémentation de M.I.C.O. afin de répondre aux besoins de l'approche d'interprétation constructionnelle telle que nous l'avons définie.

Les problèmes de satisfaction de contraintes (Montanari, 1974) permettent de modéliser de manière déclarative de nombreuses situations de la vie courante dans lesquelles il faut déterminer une configuration de choix qui respecte un ensemble de conditions. Un exemple connu consiste à modéliser un problème d'affectation de salle de cours à des professeurs et à des classes, sachant qu'au minimum, un même professeur ne peut donner deux cours à un même moment, que la même salle doit être occupée par une seule classe, et qu'une classe ne peut avoir qu'un cours simultanément. Ce type de problème est dénoté par le terme *CSP* pour *Constraints Satisfaction Problem.* Ces problèmes sont définis sur des domaines finis (le nombre de classes, de professeurs et de classes est limité) et, par conséquent, si on veut les appliquer à des questions arithmétiques, ils sont limités à des dimensions discrètes. Afin de modéliser d'autres problèmes comme par exemple pour la conception mécanique assistée par ordinateur, il faut introduire des contraintes sur des domaines continus (Sam-Haroud and Faltings, 1996), définissant des problèmes dénommés *CCSP,* pour *Continuous Constraints Solving Problems.* Ces problèmes nécessitent de modéliser les variables comme des intervalles, et posent notamment un problème quand au résultat attendu de la résolution. En effet, alors que les problèmes de satisfaction permettent de trouver des solutions, les problèmes de résolution (CCSP) permettent de trouver les relations entre plusieurs intervalles qui représentent les éléments intéressants.

Deux types de contraintes sont utilisées dans le système d'interprétation M.I.C.O. : les contraintes sur le **contenu** et les contraintes sur la **structure**. Pour les premières, on peut considérer que les travaux existants sur la résolution de contraintes sur des domaines continus permettent de répondre à nos besoins. Pour les secondes, qui concernent les *contextes* qui doivent être définis par les concepteurs finaux du système d'interprétation, elles ne sont pas directement de la responsabilité du système M.I.C.O., mais doivent être interfaçables avec le système.

9.3.1 Contenu

Les contraintes sur le contenu permettent de poser :

- des contraintes d'identité ou d'égalité entre des types atomiques ou structurés,
- des contraintes relationnelles ensemblistes entre des types atomiques et des types complexes (comme les listes ordonnées, les ensembles, ou les arbres),
- des contraintes relationnelles arithmétiques entre des types mathématiques (booléens, entiers, réels).

Le système M.I.C.O. nécessite de répondre à deux besoins :

- la satisfaction des contraintes (vérifier que les contraintes sont toutes satisfaites pour un ensemble donné de constituants)

– la solution des contraintes (une fois que les contraintes sur les constituants en entrée sont vérifiées, il faut trouver les valeurs qui permettent aux constituants en sortie d'être complètement définis afin de les produire)

Le problème de la satisfaction des contraintes est relativement simple, d'autant que dans le cas de notre système d'interprétation, il est peu probable d'avoir une grande quantité de contraintes à prendre en compte. En revanche, pour le problème de solutionnement des contraintes, on se heurte à plusieurs situations problématiques : pour les variables exprimées sur des domaines discrets, il peut exister des solutions multiples, auquel cas on peut en choisir une au hasard ; pour les variables exprimées sur des domaines continus, il peut exister une ou plusieurs plages valides, auquel cas on choisira pour l'instant de prendre une valeur « moyenne » sur l'ensemble des plages.

9.3.2 Structure

Les contraintes sur la structure, ou plus exactement les contraintes portant sur les relations définies dans les contextes (qui peuvent aussi bien représenter des relations ontologiques que des relations spatiales dans un espace à trois dimensions), permettent de contraindre les relations acceptables entre deux instances de schémas par rapport au contexte dans lequel elles sont contenues. Comme le système d'interprétation M.I.C.O. ne définit pas par lui-même les espaces dans lesquelles de telles contraintes peuvent être décrites, il doit simplement décrire l'interface qui permettra de communiquer avec les gestionnaires de contraintes des différentes configurations topologiques qui pourraient être implémentées.

Cette interface définit deux opérations :

– une opération de vérification de contrainte (si les termes de la contrainte sont fixés totalement), qui retourne un coefficient de satisfaction entre 0 (contrainte violée) et 1 (contrainte satisfaite).
– une opération de restriction (si un des termes de la contrainte n'est pas lié, c'est-à-dire qu'il peut prendre n'importe quelle valeur dans son domaine de définition) qui fixe la variable libre avec une valeur complexe [2] représentant l'ensemble des valeurs simples qu'elle peut prendre sans violer la contrainte. Cette opération permet à la fois de calculer la « position »[3] qui sera affectée à un message d'attente et la position finale qui sera affectée à l'instance de schéma.

La définition d'un nouveau contexte, et des relations qui l'accompagnent, doit donc spécifier pour chaque classe représentant les relations, une méthode *verify* et une méthode *restrict*.

9.3.3 Positions et Zones

Les contextes qui sont définis pour décrire des structures dotées de topologies quelconques doivent intégrer dans leur définition deux types particuliers : un (ou des) type(s) *position*, et un

[2] Cette valeur complexe est propre au contexte. Pour un contexte définissant un espace à une dimension, il peut s'agir d'une liste de couples *(x1,x2)* décrivant le début et la fin de chaque segment dans lesquels une variable peut prendre ses valeurs.

[3] Dans le cas des attentes, la position est en fait une zone qui couvre l'ensemble des positions acceptables pour l'instance de schéma attendue.

type *zone*. Le ou les type(s) *position* permettent, en association avec les instances de schéma, de situer les informations dans les contextes. Le type *zone* permet de décrire un ensemble d'éléments de type *position* qui peuvent être acceptables pour une *attente*. Cet ensemble de positions peut être soit défini de manière extensionnelle, soit de manière intensionnelle, mais ceci est laissé à la responsabilité du développeur du contexte.

Conclusion

Nous avons décrit l'architecture et le fonctionnement d'un système permettant de mettre en œuvre informatiquement le modèle d'interprétation constructionnelle. Cette architecture est fondée sur la notion d'observateur. Les observateurs peuvent être considérés comme des hybrides entre les constructions, les règles de production et les sources de connaissances dans les *blackboards*.

Nous avons décrit précisément les principes sur lesquels MICO est fondé, principes qui sont :

- Activation réactive/proactive,
- Activation progressive (en fonction de l'arrivée des nouvelles informations),
- Activation propagatrice (en rapport avec le principe d'activation réactive/proactive),
- Pondération des instances de schéma avec une capacité informative,
- Pondération des instances d'observateur avec le potentiel d'activation et le potentiel informatif.

A partir de ces principes, nous avons détaillé les structures de données utilisées dans MICO, et proposé une description partielle du système en utilisant la notation UML.

Dans une étape préliminaire de notre travail, nous avions conçu les observateurs comme des sous-routines, exécutant un code procédural non introspectable dans les étapes de reconnaissance et de production. Nous avions alors commencé à développer un système (figure 9.7) permettant de concevoir et d'exécuter cette première version des observateurs. Les inconvénients de ce système étaient les suivants :

- la programmation des parties procédurales des observateurs nécessite une expertise informatique qui nous a semblé peu propice à favoriser la participation d'experts en linguistique ou en psychologie.
- le fait de ne pas disposer d'informations déclaratives sur les contraintes portant sur la reconnaissance des motifs interdisait d'utiliser effectivement les contextes pour guider l'interprétation localement,
- l'utilisation des parties procédurales nécessitait d'utiliser et de programmer des gestionnaires d'événement quasiment pour chaque constituant de chaque observateur, ce qui s'est rapidement révélé trop lourd à gérer pour être réellement exploitable.

Comme nous avons basculé sur une approche utilisant les contraintes relativement tardivement, la réalisation d'un système MICO opérationnel n'a pas été possible dans les délais qui nous étaient impartis. La spécification de MICO est cependant, à notre avis, suffisamment solide maintenant pour permettre une mise en œuvre relativement rapide.

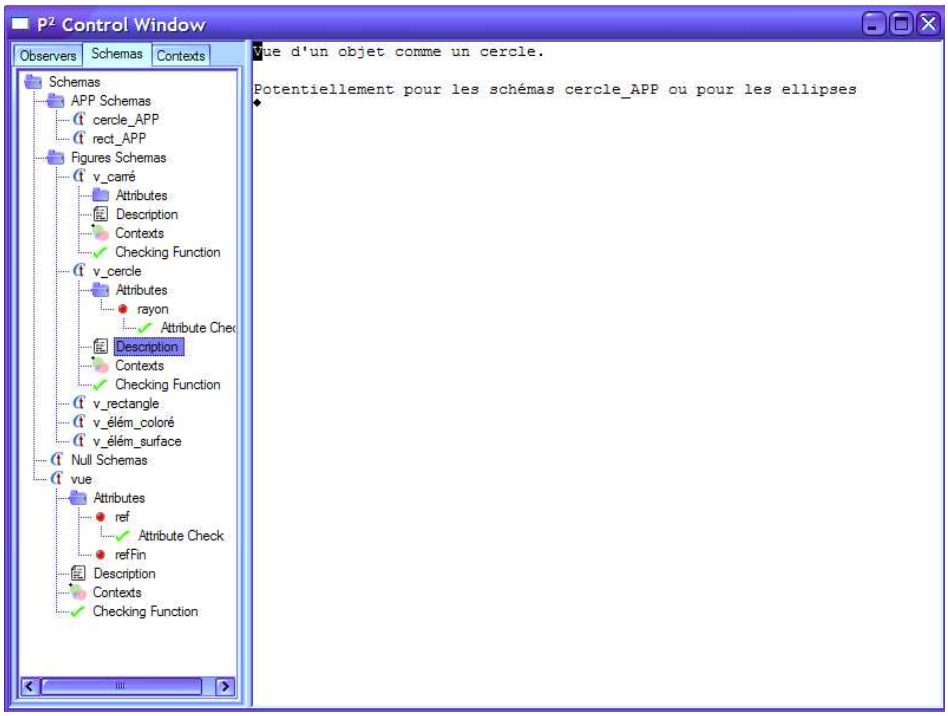


FIG. 9.7 – Une version préliminaire de l'outil de conception pour le système d'exécution des observateurs.

Chapitre 10

Architecture dialogique InterViews

InterViews est le projet qui a servi de support initial au développement des idées exposées dans cette thèse. L'objectif d'InterViews est de permettre de créer simplement l'interface naturelle d'un composant logiciel basique (par exemple une *applet* de sélection de fichier). Pour parvenir à ceci, on passe par une représentation du composant dans un langage particulier, qui doit permettre de générer à la fois l'interface naturelle et l'application exécutable elle-même.

Si les idées exposées dans le reste de cette thèse sont très éloignées des concepts qui ont servi de fondement au projet InterViews, c'est toutefois à partir de ce projet que sont nées les réflexions qui ont mené au modèle d'interprétation constructionnelle et au modèle fonctionnel de résolution extensionnelle de la référence. Nous avons en effet abandonné assez tôt l'idée de représenter les liens linguistiques qui permettent de référer à un objet **dans** la représentation de l'objet, en nous apercevant que la majorité des références portaient nécessairement sur une vision extérieure des éléments. Les notions introduites par InterViews nous paraissent cependant mériter d'être exposées, car elles peuvent apporter un éclaircissement pour comprendre les motivations qui ont présidé à l'élaboration de nos deux modèles principaux.

10.1 Architecture Générale

InterViews a été développé en suivant l'approche *médiateur*. Dans une approche médiateur du dialogue Homme/Machine, il y a trois « acteurs » séparés par deux interfaces, comme on le voit figure 10.1.

10.1.1 Le composant

Un composant est un système informatique *effectif* i.e. ayant une *activité*, une *exécution*, et comportant :

1. des programmes et des données,
2. un état interne qui évolue au cours de l'exécution du système.

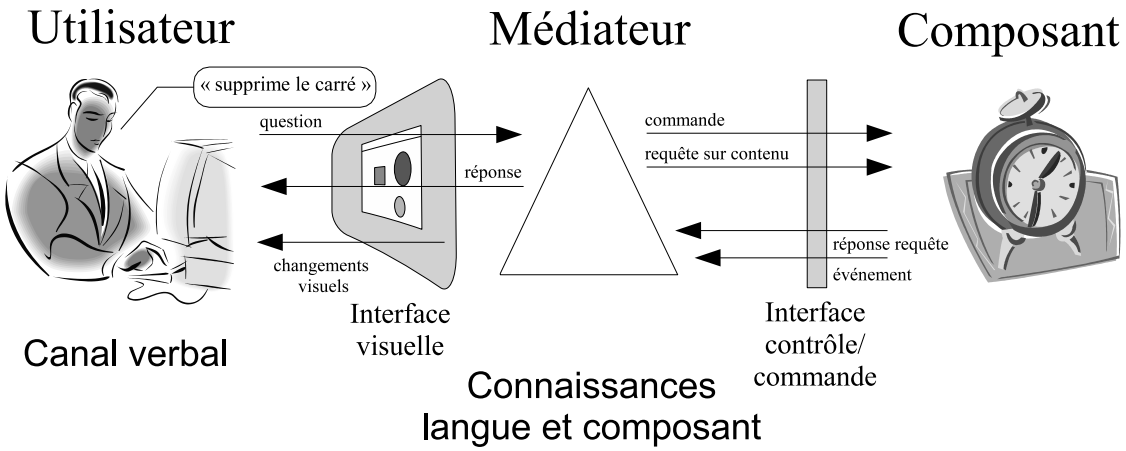


FIG. 10.1 – Architecture générale d'un système de dialogue suivant l'approche médiateur.

10.1.2 L'utilisateur

L'utilisateur est un humain qui interagit en langue naturelle avec le composant effectif. Nous dirons qu'une telle interaction est *dialogique*. Elle s'effectue selon les trois modes suivants :

1. CONTRÔLE : il s'agit essentiellement d'observer le composant :
   - Observation de la structure du composant.
   - :servation de l'état du fonctionnement du système.
2. COMMANDE : Il s'agit d'agir sur le composant en modifiant :
   - l'état de variables d'état du système[1].
   - le comportement courant.
3. ASSISTANCE : en cas de problèmes rencontrés lors du contrôle ou de la commande on demande au système de l'assistance au sujet de :
   - son passé : « Que s'est-il passé pour que j'en sois la ? » : Le médiateur doit alors faire du diagnostic.
   - son futur : « Que pourrais-je faire pour que j'arrive à ce que je désire ? » : Le médiateur doit alors faire de la planification.
   - (Interagir avec le présent du composant correspond au mode de Contrôle / Commande ci-dessus.)

10.1.3 Le Médiateur

Le Médiateur est une description, une représentation du composant qui est dédiée à l'interaction avec l'utilisateur. Cette représentation prend deux aspects :

[1] On supposera pour le moment qu'on ne peut pas modifier sa structure. Par exemple, on peut changer la valeur d'une variable mais on ne peut ni ajouter ni détruire de variables.

1. Aspect structurel statique :
   – Structure de données : ce sont essentiellement les variables d'état du composant,
   – Structure des actions : ce sont les fonctions activables dans le composant.
2. Aspect fonctionnement dynamique : il s'agit du programme proprement dit. Cet aspect est une originalité de notre modèle dans la mesure où les approches médiateur classiques ne représentent généralement que des structures (ce sont donc des représentations statiques). Cependant, on a constaté récemment l'avènement des *middleware* dans les systèmes d'information distribués. Ces logiciels établissent généralement des encapsulations ou bien des interfaces avec des sous-systèmes effectifs (i.e. qui ont une activité). InterViews se situe donc dans cette approche.

La représentation du composant par le médiateur est :

1. Contractée : le médiateur peut décrire moins finement que n'existent effectivement dans le composant :
   – les données et les actions,
   – les états physiques du système effectif.

   On appelle *granularité* de la représentation, ou du médiateur, le degré de finesse de sa description[2]. Pour le médiateur, le composant est composé de grains ou *atomes* qu'il *encapsule* en des touts sémantiques. L'utilisateur, interagissant avec le composant via le médiateur, il ne lui sera pas possible de « pénétrer » à l'intérieur d'un atome pour l'observer (contrôle), le modifier (commande) ou demander de l'aide (assistance) : nous dirons qu'un atome est *inintrospectable*.
2. Dilatée : Inversement, le médiateur peut disposer d'informations qui lui sont propres et dont le composant effectif n'a aucune connaissance :
   – des données propres : par exemple pour se souvenir des usages précédents du composant, des statistiques de fonctionnement etc.
   – des actions propres : par exemple des macro-fonctions *abstraites* qui s'implémentent en termes d'actions plus simples mais effectives au niveau du composant.

   Le médiateur agit alors vis à vis de l'utilisateur comme un macro-composant qui offre des macro-structures et des macro-fonctions.

10.1.4 L'Interface de Contrôle /Commande

Cette interface permet au médiateur d'interagir avec le composant en mode Contrôle/Commande[3]. Elle utilise généralement le langage de commande informatique du composant. Par cette interface, le médiateur peut :

1. Sonder le composant : il s'agit de lire, de mesurer des valeurs des états physiques que seul le composant peut produire (par exemple : mesurer la température effective à l'intérieur du rôti dans un four à micro ondes).

[2]Nous supposerons ici que ce degré de finesse est déterminé de manière *arbitraire* par le programmeur du médiateur qui décide souverainement de ce qu'il souhaite décrire et ne pas décrire du composant effectif. La question de l'évaluation qualitative (voire même quantitative) de la pertinence du degré de finesse de la représentation dépasse largement notre étude.

[3]Pas en mode assistance ! celui-ci reste le propre du médiateur.

2. AFFECTER le composant : il s'agit généralement de modifier le paramètres de contrôle des processus physiques qui s'exécutent dans le composant.

Entre le médiateur et le composant il y a un flux bidirectionnel de :

– commandes : médiateur ↦ composant, et de
– mesures : composant ↦ médiateur

que l'on peut voir comme un mode d'interaction de type CALL/RETURN.

10.1.5 L'Interface Perceptuelle

Cette interface permet à l'utilisateur d'interagir dialogiquement avec le médiateur et par delà, avec le composant. Pour notre propos, nous supposerons que cette interface est constituée d'un écran classique de station de travail muni des organes conventionnels (souris, clavier, ...) ; en particulier nous n'entrerons pas, ici[4], dans la problématique d'interfaces qui soient sous-dimensionnées perceptuellement (que ce soit par une limitation du matériel - ex. : portable - ou de l'utilisateur - ex. : handicap) ou tout au contraire sur-dimensionnées perceptuellement (Réalité augmentée, CAVES, ...).

L'interface perceptuelle « donne à voir » à l'utilisateur la description médiée du composant. L'utilisateur a donc :

– une perception statique : il s'agit d'entités fixes de l'interface : fenêtres, boutons, figures, textes, tables, ...
– une perception dynamique : il s'agit de comportements du composant qui transparaissent dans l'interface perceptuelle sous forme de *mouvements* d'entités ou de *changements* de valeur des variables d'état. Le comportement dynamique du composant peut être associé :
  – à des réactions aux requêtes de l'utilisateur : ce mode de fonctionnement des composants est dit *modal*,
  – à des processus internes au composant qui s'exécutent en l'absence d'actions de l'utilisateur : ce mode de fonctionnement des composants est dit *modless*.

L'humain construit un modèle cognitif (il se fait une certaine idée) du composant via cette interface perceptuelle. Norman (1999) a appelé *affordance* l'influence qu'a un artefact sur les humains qui s'en servent. Pour lui, un artefact bien conçu doit « parler » à son utilisateur, lui donner par son apparence et son comportement une intuition de son fonctionnement et de comment il faut s'en servir. Nous verrons plus loin le rôle essentiel que doit jouer le principe d'affordance dans le traitement des requêtes dialogiques.

A partir de ce modèle cognitif, l'utilisateur interagit principalement de manière dialogique avec le médiateur-composant c'est-à-dire en langue naturelle que nous appellerons (un peu abusivement) des *questions*. Entre l'utilisateur et le médiateur il y a un flux bidirectionnel de questions / réponses :

– QUESTIONS : utilisateur ↦ médiateur ; ce sont des phrases en langue naturelle,
– RÉPONSES : médiateur ↦ utilisateur ; ce sont des phrases en langue naturelle et/ou des actions (perceptibles dans l'interface).

[4]Pour cet exposé, car il est évident que c'est justement dans les cas d'interfaces non classiques que le dialogisme prend tout son intérêt et peut même devenir le seul moyen de communiquer).

10.2 Qu'est-ce qu'une question ?

10.2.1 Phases de traitement d'une question

Lors de la prise en compte d'une interaction dialogique entre un usager et un composant-médié, nous avons deux problèmes principaux à résoudre :

1. ANALYSER la question : c'est-à-dire *comprendre* la demande de l'utilisateur ce qui revient à être capable de la traduire dans un formalisme interne de requête formelle qui sera envoyé au médiateur
2. RÉSOUDRE la question : une fois la requête formelle construite, il reste à être capable d'y répondre :

– d'abord formellement : le système doit explorer le médiateur (structure et runtime) et construire une réponse formelle. Cela n'est pas toujours possible même si la requête est syntaxiquement et sémantiquement correcte :
  – problèmes de diagnostic,
  – problèmes de planification,
  – problèmes d'incapacité physique.
– ensuite « naturellement » : dans le cas où on a pu construire une requête formelle il faut la produire, c'est-à-dire construire une phrase en Langue Naturelle et/ou une action (perceptible dans l'interface) qui apparaisse pertinente pour l'utilisateur.

Dans un système de DHM classique, on peut considérer que le schéma général de traitement d'une question adressée à un composant est représenté figure 10.2.

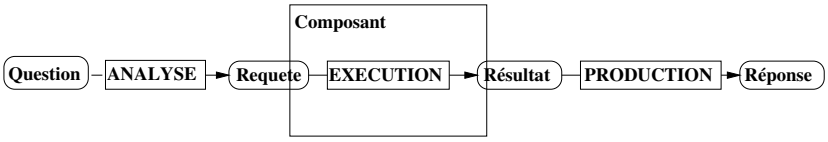


FIG. 10.2 – Séquence de traitement d'une question

Ce qui caractérise cette approche c'est qu'elle est *descendante*. On part de la phrase pour aller vers la requête et ceci se fait par des phases indépendantes les unes des autres et organisées de manière séquentielle :

1. Analyse lexicale,
2. Analyse syntaxique,
3. Analyse sémantique,
4. Analyse pragmatique (ou contextuelle ou située).

Dans le modèle d'analyse classique seule la dernière phase, l'analyse pragmatique, comme son nom l'indique, a affaire avec la *situation effective* dans (ou pour) laquelle la phrase est produite. Cette approche est tout a fait envisageable dans le cas de la compréhension de textes de nature narrative[5].

[5]et encore, les approches actuelles à base de « modèles de situation » (Kintsch (1988)) affirment la primauté du contexte sur la grammaire syntactico-sémantique pour la construction du sens.

Cependant, si on se restreint au domaine de la langue qui concerne le *dialogue* entre humains, cette approche est difficile à tenir : l'observation de divers corpus de dialogue (par exemple, les corpus de conversations téléphoniques) fait apparaître que la syntaxe des phrases est pauvre (mots isolés, inversés, simples juxtapositions, usage fort de l'implicite pour la référence (surtout la métonymie), etc.) et que la sémantique est fortement liée au contexte.

Maintenant, si nous considérons, à plus forte raison, le cas *tout particulier* du dialogue homme/machine, on constate immédiatement que le dialogue est quasiment toujours orienté-tâche i.e. c'est un dialogue où l'utilisateur humain est le *maître* et de plus ne dialogue avec la machine que dans le but d'effectuer (ou de faire effectuer) une certaine *tâche*. dans ce cas, tous les traits qui distinguent le dialogue humain de la narration se trouvent très accentués.

Phrase narrative c'est une phrase « de livre » à la syntaxe et à la sémantique très normées. C'est d'abord sur ce type de phrases qu'ont portées les recherches en TALN, où les aspects grammaticaux sont très forts.

Phrase de dialogue c'est une phrase émise entre humains de manière *orale*. Austin puis Searle ont bien montré la spécificité de la langue orale par rapport à la langue écrite.

Phrase dialogique c'est une phrase émise par un humain seul, dans le cadre strict de son interaction avec une machine, dans le but d'effectuer ou de faire effectuer une tâche.

Dans le projet InterViews nous ne traitons que des phrases dialogiques et c'est pourquoi nous adopterons une attitude inverse, *ascendante*, où le sens part de la situation (i.e. pour nous du runtime[6] du médiateur) pour analyser la phrase de l'utilisateur. Nous dirons que l'analyse de la phrase est dirigée par le médiateur.

10.2.2 Traitement de la phrase dialogique dirigé par le médiateur

Dans le modèle InterViews, le médiateur intervient à chaque phase de l'analyse de la phrase dialogique émise par l'utilisateur :

1. Analyse de la phrase : dirigée par le médiateur,
2. Exécution de la requête : dirigée par le médiateur,
3. Production de la réponse : dirigée par le médiateur.

L'analyse de la phrase dialogique est dirigée par le médiateur de deux manières :

1. On considère que les phrases de l'utilisateur sont *a priori* de nature dialogique, i.e. des demandes de contrôle / contrôle / assistance et non de nature narrative, poétique, etc. Cela revient à dire que les phrases seront vues *a priori* comme des *actes de langage* de la forme $V(R_i)$ où :
   - V est l'acte à effectuer : un *verbe*,
   - R_i est l'objet de l'acte : une *référence* à une entité *interne*[7] au médiateur.

[6] C'est-à-dire son aspect dynamique.

[7] Il est important de noter qu'un médiateur n'a pas d'autres connaissances du monde que de lui-même : il est "tout son monde". C'est pourquoi, toute référence ne peut être (du point de vue du médiateur) qu'à une entité qui lui est interne.

2. Dans notre modèle, nous considérons qu'à l'inverse d'une phrase "de livre" les phrases dialogiques sont fortement associées à une *intention d'agir* dans et sur une situation (i.e. le fonctionnement du composant). C'est pourquoi nous pensons qu'à l'opposé d'une phrase classique dont la sémantique est sensée être déterminée de manière *générique* (hors de tout contexte effectif), la sémantique d'une phrase dialogique doit être confrontée *dès le début de son analyse* à la situation (i.e. pour nous à la représentation de la situation statique-dynamique qu'est le médiateur). En conséquence, nous conserverons les quatre phases définies plus haut mais elles seront toutes filtrées / dirigés par le médiateur :
   - Analyse lexicale : le médiateur ne considère que les mots de son vocabulaire. Ce vocabulaire est composé de deux grandes catégories de mots : les mots prédéfinis qui servent à construire les représentations du médiateur et les mots spécifiques du composant qui sont fournis par le programmeur du médiateur.
   - Analyse syntaxique : les catégories syntaxiques ne sont pas les catégories classiques[8] (groupes verbaux, nominaux etc.) mais sont celles du médiateur (correspondant aux types structurels des entités internes au médiateur).
   - Analyse sémantique : elle est dirigée par les actes du médiateur (fonctions prédéfinies - génériques - ou bien fonctions définies spécifiquement par le programmeur).
   - Analyse pragmatique : elle est dirigée par l'état physique des variables du runtime, c'est-à-dire par la *chronologie* de ce qui s'est passé depuis que le composant médié a été mis en activité.

10.2.3 Dialogisme Générique

Dans les systèmes de dialogue Homme/Machine classiques, opérationnels, le traitement des requêtes en langue naturelle est généralement effectué de manière ad hoc : les questions, leur syntaxe sont :

1. décrits par le programmeur, in extenso,
2. de manière statique.

Dans InterViews, notre objectif est d'introduire un principe de généricité dans l'interaction avec des composants actifs. Cela veut dire que le programmeur doit décrire la structure et le fonctionnement de son composant dans une représentation, mais n'a pas à s'occuper de fournir ni le traitement des requêtes génériques sur la représentation, ni le traitement des questions (en langue naturelle) générique de la vue. C'est le système InterViews qui, prenant une vue VDL[9] engendre automatiquement :

- l'interface conceptuelle,
- l'interface de contrôle/commande,
- les requêtes formelles génériques qui permettent d'introspecter la structure et le fonctionnement de la vue,
- la lexicalisation de la vue i.e. les informations nécessaires à l'analyse des questions en langue naturelle.

[8] En fait, on retrouvera bien entendu des catégories très similaires.
[9] VDL est le langage de description des composants InterViews. Il est présenté plus loin.

Le programmeur n'a qu'à décrire la vue qui médie le composant en termes de structure et de fonctionnement.

10.3 VDL

Pour décrire les composants en suivant les principes évoqués précédemment, il a fallu concevoir un langage particulier. VDL (View Design Language) est le langage qui permet de décrire les composants (Vues) d'InterViews.

10.3.1 Notations

Dans ce qui suit, nous utiliserons deux notations (au risque d'alourdir le propos) ;

- une notation ou syntaxe formelle classique,
- une notation ou syntaxe que nous qualifierons de *concrète* et qui correspond à la manière dont on décrit effectivement le médiateur en VDL. Cette notation diffère légèrement de la notation formelle pour des raisons pratiques de programmation et aussi pour alléger l'écriture en VDL. Ici, elle permettra de donner des exemples.

Pour la notation formelle, nous utiliserons deux notions que nous définirons comme suit

Ensemble un ensemble sera une énumération d'éléments qui apparaissent une fois et une seule et qui ne sont pas ordonnés. Un ensemble d'éléments x_i sera noté : $\bigcup x_i$. En syntaxe concrète, un ensemble s'exprime par `{xi,...}`.

Séquence une séquence sera une énumération d'éléments qui peuvent apparaître plusieurs fois et qui sont ordonnés (dans l'ordre implicite de leur énumération). Une séquence d'éléments x_i sera notée : $\sum x_i$. En syntaxe concrète, une séquence s'exprime par `[xi,...]`.

Remarque : les notations $\bigcup$ et $\sum$ ci-dessus sont purement syntaxiques et n'ont pas de connotation logique pour $\bigcup$ ni de somme pour $\sum$.

10.3.2 Les quatre notions de base d'un médiateur

Notre modèle est organisé autour de quatre notions de base qui entretiennent des liens très étroits et que nous allons développer dans les sections suivantes :

Vue une vue est un *ensemble* de représentations qui forment une unité sémantique médiant un composant effectif.

Concept les représentations dans une vue sont appelées des concepts. Un concept s'exprime par les relations qu'il entretient avec d'autres concepts de la vue ou avec des expressions externes à la vue.

Relation une relation relie un concept soit à d'autres concepts internes à la vue soit à des expressions externes à la vue.

Expression une expression est une entité externe à la vue qui relie le concept qui la contient à une partie externe au médiateur (qui est soit dans l'implémentation soit dans le composant médié). Cette entité n'appartient pas au langage du médiateur. Elle est exprimée dans le langage de l'implémentation[10].

Définition d'une vue

Dans le projet InterViews, c'est au travers d'une interface perceptuelle associée au médiateur que l'utilisateur se construit un modèle cognitif du composant et interagit avec lui de manière dialogique. C'est pourquoi, dans le projet InterViews, l'association d'un médiateur et de son interface perceptive est appelée une *vue*.

Dans le projet InterViews, les vues sont décrites par le programmeur dans le langage VDL.

Une vue V d'identifieur v[11] est un ensemble de concepts C_i :

$$V \doteq v \rightarrow \cup C_i$$

Selon la manière dont on se situe vis à vis de l'interface perceptuelle d'une vue, on peut définir un intérieur et un extérieur :

- A l'intérieur d'une vue, c'est-à-dire pour le programmeur qui est en train de définir le médiateur, les concepts C_i se "connaissent" les uns les autres. Cela revient à dire qu'ils peuvent se référer les uns les autres par leur identifieur interne (ou ID), comme dans un langage de programmation classique.
- A l'extérieur d'une vue, c'est-à-dire pour l'utilisateur qui est en train de l'utiliser, les C_i ne sont pas donnés explicitement. On ne donne pas à l'utilisateur une liste des concepts avec leurs caractéristiques. En particulier, l'utilisateur n'a jamais accès aux ID des concepts qui demeurent internes au médiateur. Les concepts d'un médiateur se donnent à (perce-)voir via leur phénomène i.e. leur apparence dans l'interface perceptive : l'utilisateur les catégorise :
  - il se fait une idée *intuitive* (pas forcément exacte) des entités qu'il croit être celles du composant,
  - ensuite, il réfère les entités qu'il a catégorisé via leurs propriétés perceptuelles :
    - statiques : par exemple, la couleur, la forme, ... pour désigner des *choses* qu'il perçoit,
    - dynamiques : selon les mouvements ou les changements qu'il observe pour désigner les *behaviors* qu'il perçoit.

Définition d'un concept

Un concept C d'identifieur c est un ensemble de relations n-aires

$$C \doteq c \rightarrow \cup R_i$$

où une relation R_i relie le concept à $n \geq 0$ autres identifieurs de concept

[10] Le langage d'implémentation de VDL 0.4 étant Mathematica (Wolfram, 1988), les expressions seront des expressions Mathematica quelconques.

[11] Nous noterons les entités en majuscules et leurs identifieurs interne en minuscules. Cependant comme l'identifieur a pour rôle de dénoter l'entité nous ferons souvent l'abus de parler de la vue v au lieu de la vue V.

Définition d'une relation

Une relation R d'identifieur r est une relation n-aire qui est définie par

$$R \doteq r(c_0, \sum a_i)$$

– c_0 est une référence[12] à son concept-source,
– $\sum a_i$ est une séquence, éventuellement vide, d'arguments. Les arguments d'une relation sont
  – soit des références à des concepts. Ces concepts sont appelés : concepts-destination.
  – soit des expressions (externes).

$$a \doteq c \mid e$$

Remarque 1 : dans la définition d'un concept, pour alléger l'écriture, on factorise le concept-source c_0 et on écrira donc $c_0 \rightarrow \cup r_1 \sum a_i,\ r_2 \sum a_j,\ ....$ et non $c_0 \rightarrow \cup r_1(c_0,\ \sum a_i),\ r_2(c_0,\ \sum a_i), ...$

Remarque 2 : dans une définition de concept on ne peut pas voir apparaître plusieurs fois la même relation. Si nous voulons exprimer la relation binaire « a pour fils » (apf) au sujet du concept source $jean$ qui a trois fils $pierre$, $paul$ et $louis$ nous aurons

$$\bigcup\ apf \sum jean, pierre,\ apf \sum jean, paul,\ apf \sum jean, louis$$

qui devra se traduire en la définition de concept suivante (ou la relation apf est passée de binaire à n-aire apf^*)

$$jean \rightarrow \bigcup\ apf^* \sum pierre, paul, louis$$

Définition d'une expression

Dans VDL, les expressions n'apparaissent que comme arguments de relations. Les expressions sont des entités externes à VDL. Elle dénotent

1. soit une entité dans l'implémentation du médiateur
   Par exemple, les nombres (et les fonctions qui les manipulent) n'ont pas à être implémentés dans le médiateur. Il suffit qu'ils soient reliés à des concepts de la vue. Par exemple dans la définition de concept suivante

   $$\text{red} \rightarrow \bigcup\ \text{structure} \sum \text{cst},\ \text{content} \sum \texttt{RGBColor[1.,0.,0.]},\ \text{lex} \sum \texttt{``red''}$$

   Le concept d'identifieur `red` est lié par la relation *content* à l'expression implémentatoire Mathematica `RGBColor[1.,0.,1.]` qui permet de dessiner en rouge et via la relation `lex` à la chaîne de caractères Mathematica `''red''`.
   Un nombre bien connu est

[12] Une référence (au sens informatique) à un concept est constituée par le symbole identifieur interne du concept.

$$z \rightarrow \bigcup \text{structure} \sum \text{cst}, \text{content} \sum 0$$

De même on définira le concept booléen FALSE comme le concept f

$$f \rightarrow \bigcup \text{structure} \sum \text{cst}, \text{content} \sum \texttt{False}$$

Bien entendu, les expressions implémentatoires sont exprimées en termes de l'implémentation et lui sont spécifiques ; si on change le support implémentatoire il faut réécrire toutes ces expressions en termes du nouvel environnement. Par exemple, en Lisp le booléen FALSE s'écrira

$$f \rightarrow \bigcup \text{structure} \sum \text{cst}, \text{content} \sum \texttt{nil}$$

Les expressions (comme `RGBColor[1.,0.,1.]`, `False`, `nil`, ...) ne sont pas accessibles à l'introspection (c'est-à-dire en mode contrôle/commande/assistance) ; l'introspection s'arrête au niveau de l'identifieur du concept qui englobe l'expression[13].

2. soit une entité en relation avec une partie du composant effectif médié, via l'interface de contrôle/commande.
   Le programmeur du médiateur décompose le composant effectif en parties qu'il relie à des concepts de la vue comme le montre la figure suivante

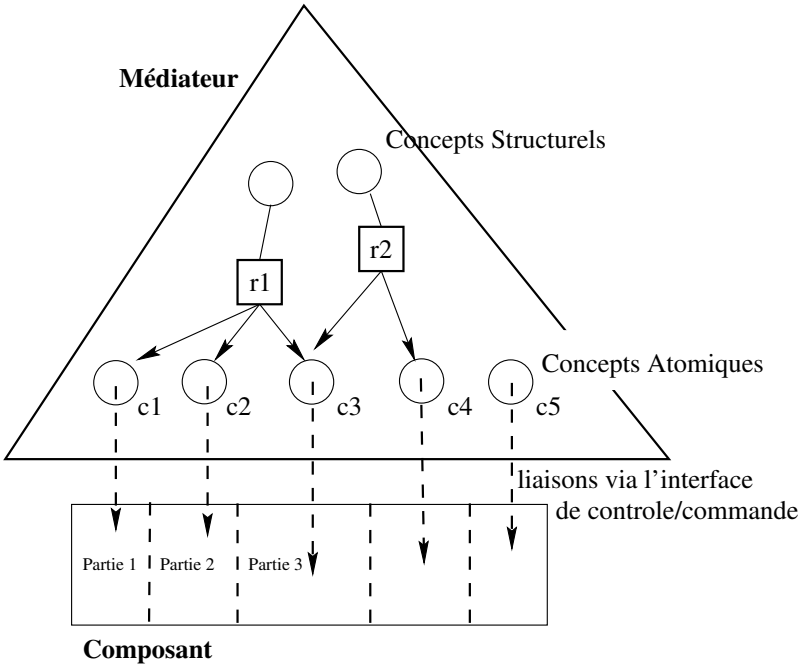


FIG. 10.3 – Liaisons entre le médiateur et les parties du composant effectif

La liaison entre un concept du médiateur et les parties du composant effectif se fait le plus souvent via une référence implémentatoire à la partie du composant. Par exemple, ce sera

– un nom de fichier : $c \rightarrow \bigcup$ `content` $\sum$ `''/jps/interviews/vdl0.4/intro'', lex` $\sum$ `''introduction''`
– une URL : `c` $\rightarrow \bigcup$ `content` $\sum$ `''http ://www.limsi.fr/jps/interview/online/coco'', lex` $\sum$ `''coco''`
– une image gif, un frame HTML, un thread Unix, une méthode d'un objet Java, ...

[13] En effet, à la question « what is the definition of red » la réponse `RGBColor[1.,0.,1.]` ne serait pas pertinente pour l'utilisateur qui ne doit pas avoir affaire avec l'implémentation.

10.3.3 Aspects structurels des vues

Représentation graphique d'une relation

Une relation VDL peut être représentée graphiquement par la figure suivante :

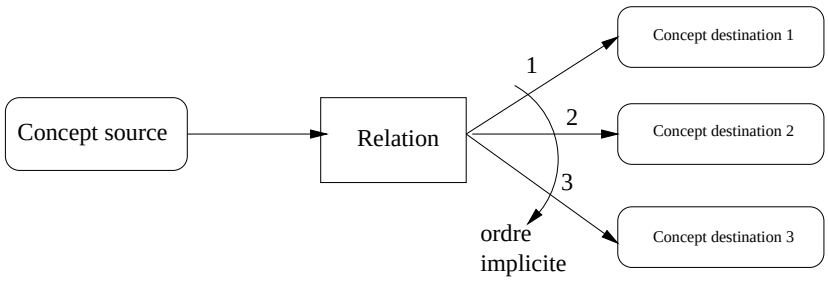


FIG. 10.4 – Représentation graphique d'une relation n-aire VDL.

Représentation graphique d'une vue

Une vue est un graphe de concepts reliés par des relations n-aires.

Soit la vue :

$$v \rightarrow \bigcup\ c_1 \rightarrow \bigcup r_1 \sum c_2, c_3,\ r_2 \sum c_3\ \ c_2 \rightarrow \bigcup r_1 \sum c_3, r_3,\ \ c_3$$

Elle correspond au graphe de concepts et de relations suivant :

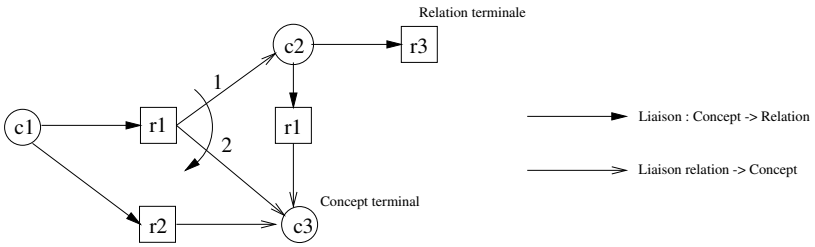


FIG. 10.5 – Un graphe conceptuel

La structure de représentation est très proche[14] de celles des *graphes conceptuels* définis par Sowa (1984). Il s'agit d'un graphe bipartite de noeuds concepts (les cercles) et de noeuds relations (les carrés). Le graphe d'une vue peut possèder des circularités.

– Un concept peut être *terminal* : par exemple `c3` n'est concept-source d'aucune relation.

[14]Il ne s'agit cependant pas du formalisme exact des graphes conceptuels. Seule la structure de graphe bipartite de concepts et de relations est reprise. En particulier, dans notre modèle il n'y a pas de variables au sens de la LPO.

– Une relation peut être *terminale*[15] : par exemple `r3` ne relie `c2` à aucun autre concept.

Eléments terminaux

Nous aurons les éléments terminaux suivants :

– Une expression est terminale.
– Une relation est dite terminale si :
  – sa séquence d'arguments est vide : par exemple, dans la figure ci-dessus, `r3` ne relie `c2` à rien,
  – ou bien si elle n'a que des expressions comme argument.
– Un concept est dit terminal si :
  – il n'est concept-source d'aucune relation : par exemple, dans la figure ci-dessus`,c3.`
  – ou bien si il est concept-source de relations, elles-même terminales.

Concepts structurels

Les concepts structurels sont les concepts non terminaux. Ils agrègent en une entité sémantiquement unie d'autres concepts de la vue

– soit des concepts terminaux,
– soit des concepts structurels.

Pour agréger dans un concept d'autres concepts on utilise les relations. Dans VDL, nous proposons de manière prédéfinie plusieurs relations (détaillées plus loin) dont les deux suivantes qui permettent de construire :

1. des agrégats statiques (ou records) : `rec` qui agrège selon la relation classique PARTOF :

   $$c \rightarrow \bigcup structure \sum rec,\ content \sum x, y, z$$

   – x,y et z sont des concepts, parties de c
   – c est composé des concepts parties : x, y et z.

2. des domaines statiques définis en extension[16] : `dom` qui agrège selon la relation classique ISA :

   $$c \rightarrow \bigcup structure \sum dom,\ content \sum x, y, z$$
   $$x \in c; y \in c; z \in c.$$

Sémantique d'une relation

La sémantique d'une relation se décompose en deux parties :

[15] Une relation terminale correspond en LPO à un prédicat unaire : dans l'exemple de la vue v, la définition de concept c2 → {r1[c3], r3}, peut se traduire par la formule prédicative au sujet de c2 suivante : r1(c2, c3)⋀r3(c2) où on voit bien que r3(c2) est unaire.

[16] C'est-à-dire en énumérant explicitement les éléments du domaine.

1. La sémantique *implicite* : cette sémantique est *implicite* dans la mesure où elle ne s'appuie que sur la syntaxe de la relation indépendamment de toute signification associée à l'identifieur de la relation[17]. Ecrire que $r(c_0, \sum a_i)$ veut dire que c_0 est en r-relation-ordonnée avec les $\sum a_i$ ni plus ni moins.
2. La sémantique *opérationnelle* : elle est déterminée par l'interprétation que des opérateurs externes à la vue peuvent faire des relations qu'ils rencontrent lorsqu'ils parcourent les concepts de la vue. Ceci sera développé plus loin dans la section consacrée aux observateurs.

Par sa forme, sa syntaxe, une relation n-aire VDL porte plusieurs informations qui peuvent être interprétées en une sémantique implicite :

1. Il existe un argument distingué : c'est le concept-source. On voit dans la figure ci-dessus qu'il y a *une et une* seule flèche qui relie le concept-source au noeud relation (le rectangle). L'interprétation sémantique que nous en ferons est que ce concept-source est l'entité au sujet de laquelle la relation apporte une information en la reliant à d'autres entités : les concepts-destination.
2. Les concepts-destination sont implicitement ordonnés. Cet ordre servira souvent de support aux interprétations.
3. Si la séquence des concepts-destination est vide alors la relation est terminale. Elle apporte une information *intrinsèque* (i.e. qui n'a pas besoin d'autres concepts pour s'exprimer) au sujet du concept-source.

10.3.4 Un exemple complet : coco

Coco est notre exemple paradigmatique. C'est un cas d'école où le médiateur se confond avec son composant effectif. Ce compteur compte tout seul (fonctionnement modless) grâce à un processus p qui incrémente sa variable i d'une quantité $s \geq 0$ à chaque top de l'horloge système si sa variable b vaut True. L'utilisateur peut agir sur le fonctionnement de ce compteur au moyen de quatre fonctions spécifiques, i.e. programmées explicitement dans le médiateur :

start arrête le compteur

stop redémarre le compteur

faster augmente la vitesse de comptage

slower diminue la vitesse de comptage (qui peut devenir nulle mais qui ne peut pas devenir négative)[18]

```
DEFVIEW[coco,
coco   = rec[i,s,b,ops]
           [vert,lex[``coco'']],
ops    = rec[start,stop,faster,slower]
```

[17] Une signification associée à une relation doit être fournie *explicitement* dans une table

[18] L'utilisateur peut s'en rendre compte expérimentalement. Cependant le test `If[get[s]>0,...` étant "enterré" dans une expression implémentatoire il n'est pas possible d'obtenir du médiateur qu'il puisse se représenter ce fait et à plus forte raison en rendre compte, suite à une question. Rappelons que cette problématique est traitée séparément par N. Sabouret dans le formalisme VDL-procédural(Sabouret and Sansonnet, 2001).

```
            [hor],
i       = var[1]
            [init[1],over[int],lex[‘‘counter’’]],
s       = var[1]
            [init[1],over[int],lex[‘‘speed’’]],
b       = var[True]
            [init[True],over[bool],lex[‘‘running’’]],
start   = fct[(put[b,True])&]
            [lex[‘‘start’’]],
stop    = fct[(put[b,False])&]
            [lex[‘‘stop’’]],
slower = fct[(If[get[s]>0,put[s,get[s]-1]])&]
            [lex[‘‘slower’’]],
faster = fct[(put[s,get[s]+1])&]
            [lex[‘‘faster’’]],
p       = proc[(If[get[b],put[b,get[b]+1]])&]
]
```

Remarque : les mots par lesquels l'utilisateur peut référer les quatre actions (voir leur relation `lex`) ont un *sens commun* qui décrit le but que l'utilisateur veut atteindre : par exemple `faster` ne se dit pas en termes de « comment le faire » (i.e. « increment your speed ») mais en termes du but à atteindre, but qui sera l'effet perçu de l'action dans l'interface perceptuelle (dans ce cas l'effet n'est pas un objet qui est catégorisé mais un comportement catégorisé, un behavior). Ceci est un exemple très simple qui met directement en pratique le principe d'affordance, c'est-à-dire le parallélisme qui devrait, le plus possible, exister entre la lexicalisation des concepts et les catégories perçues dans l'interface.

Outre l'exécution des quatre opérations *spécifiques*, l'utilisateur peut poser à Coco plusieurs autres types de questions qui correspondent à des requêtes *génériques* (i.e. que le programmeur n'a pas à spécifier dans le médiateur). Les requêtes génériques correspondent à des observateurs prédéfinis.

Par exemple, plaçons nous à l'instant d'exécution $\theta = 7$ où la valeur[19] de la variable `i` vaut 7, à la question « Please show me coco » le médiateur déclenchera l'observateur prédéfini $\phi_{show}(coco)$ qui répondra par l'affichage figure 10.6.

10.4 Architecture client-serveur pour l'interfaçage d'InterViews sur l'internet

Pour les besoins d'expérimentation du modèle développé pour InterViews, il nous a fallut concevoir une architecture client/serveur simple permettant de donner accès aux réalisations du projet sur le Web.

En effet, l'implémentation des différents modules d'InterViews a été réalisée dans le langage Mathematica de Wolfram Research (Wolfram, 1988), qui offre de très grandes facilités pour

[19] c'est-à-dire la relation `content` du concept `i`.

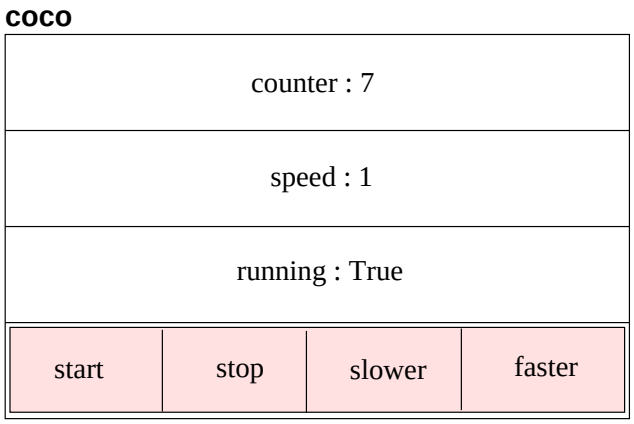


FIG. 10.6 – Affichage de l'état du *runtime* de Coco le compteur

les étapes de prototypage d'applications. Malheureusement, notre ambition d'exposer des exemples de dialogisme sur le Web était contrariée par le fait qu'il n'était pas possible de faire exécuter du code Mathematica dans un navigateur Web.

Pour pallier cette fonctionnalité absente de Mathematica, il nous a donc été nécessaire de concevoir une architecture la plus légère possible pour faire une passerelle entre un navigateur Web et nos programmes Mathematica.

10.4.1 Architecture

L'architecture globale est illustrée par la figure 10.7. On y distingue deux parties, la partie client, qui est celle de l'utilisateur, connecté sur internet, et la partie serveur, située au LIMSI, qui exécute les noyaux Mathematica.

Côté Client

L'objectif de cette architecture étant d'offrir un affichage continu des résultats produits par les vues VDL interprétées en Mathematica, il nous a parut évident qu'une solution purement HTML à notre problème ne serait pas viable. Pour cette raison, nous avons choisi d'utiliser le langage Java pour permettre à l'affichage d'être dynamique.

Du côté du client, le processus de connexion se déroule comme ceci :

1. récupération de la page HTML avec le code Java, contenant entre autre la classe WebLink qui gère tout le processus de connexion avec le serveur Mathematica,
2. exécution du code Java,
3. ouverture d'une connexion temporaire par un script CGI situé sur le même réseau local que le serveur Mathematica,
4. récupération des informations de connexion permanente avec le serveur Mathematica,
5. exécution du code Mathematica défini dans l'applet Java (initialisation, chargement des bibliothèques à distance), puis, tant que l'applet fonctionne :

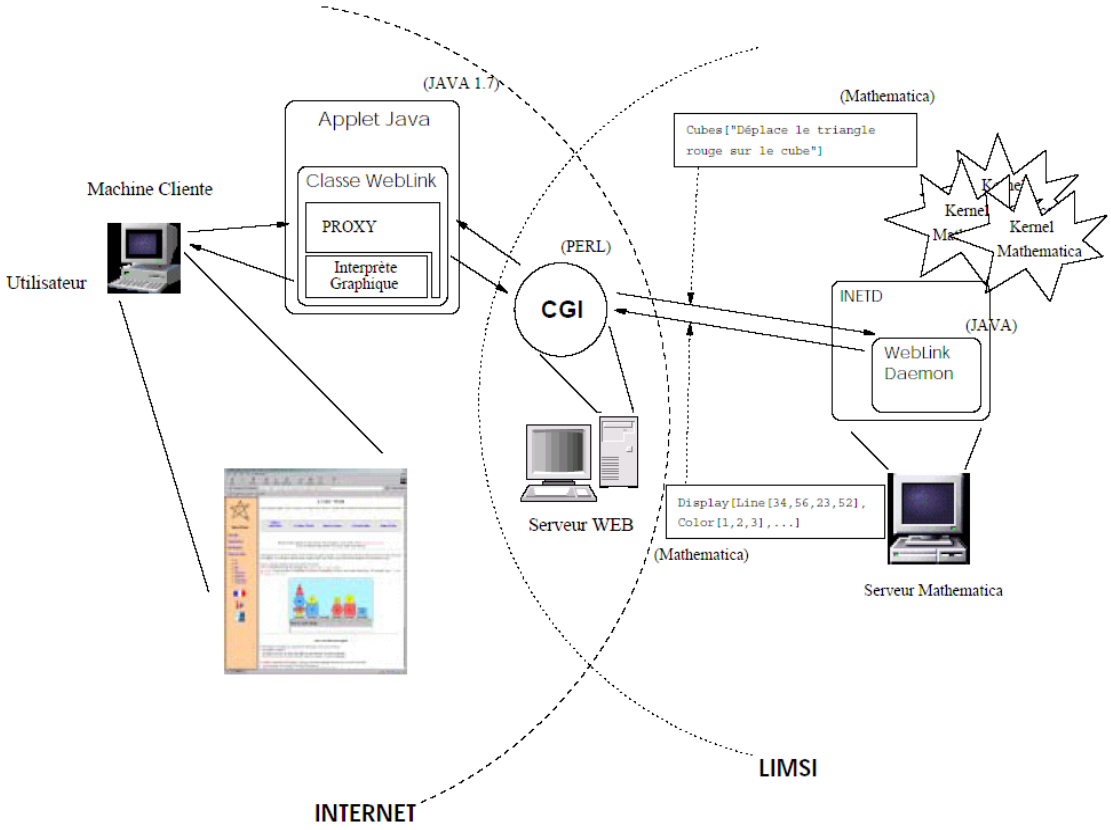


FIG. 10.7 – Architecture d'interfaçage internet/Mathematica pour InterViews.

- envoi des requêtes Mathematica,
- récupération de la réponse du serveur (sous forme d'une expression Mathematica),
- interprétation de l'expression reçue par l'applet (l'applet contient un parseur et un interprète que nous avons écrit),
- affichage du résultat de l'interprétation dans la fenêtre graphique de l'applet.

6. fermeture de la connexion.

Côté Serveur

Du côté du serveur sous Linux, un « daemon » nommé WebLinkDaemon écrit en *Java* est lancé par *inetd* lorsqu'une connexion arrive sur un port prédéfini. Cette connexion est créée par le script *CGI*, écrit en *Perl*, lorsque ce dernier est déclenché par une requête provenant de l'applet *Java* du client. Elle donne lieu à l'ouverture d'un nouveau port, dont l'identifiant est retourné au *CGI*, qui le transmet à son tour au client.

Le daemon *WebLinkDaemon* démarre alors un noyau *Mathematica*, avec lequel il possède une connexion directe, grâce à une classe fournie pour l'interopérabilité avec Java.

Ensuite, le client connaissant l'identifiant pour la connexion avec le *WebLinkDaemon*, il fait alors transiter ses requêtes par le *CGI*, qui doit être exécuté une fois pour chaque requête.

Cette architecture à quatre composants a été rendue indispensable pour trois raisons :

1. problème de sécurité sur le serveur Web,
2. Mathematica n'existe que pour Windows, Macintosh et Linux,
3. nécessité de conserver un connexion permanente pour ne pas redémarrer un noyau Mathematica à chaque requête (très coûteux en temps, évidemment).

10.4.2 Exemple

Plusieurs applications ont été créées autour du projet InterViews. Parmi elles, une imitation du célèbre SHRDLU (Winograd, 1973), nommée « Monde de Cubes ». La visualisation de cette application grâce à l'interface web est illustrée par la figure 10.8. Le principe du monde de cube est de pouvoir commander le déplacement de figures géométriques simples de différentes couleurs. En haut de la figure se trouvent les instructions à l'utilisateur. Au centre, l'interface graphique permettant d'afficher le monde de cubes. Cette interface dispose de plusieurs colonnes sur lesquelles des figures géométriques peuvent venir s'empiler. Juste en dessous de l'interface graphique, on voit la boîte de texte permettant au système d'envoyer les réponses aux requêtes de l'utilisateur, et encore en dessous, la boîte de texte permettant à l'utilisateur d'envoyer ses requêtes.

Conclusion

Le système InterViews, en révélant les problèmes posés par une approche dans laquelle représentation informatique et représentation linguistique était entremêlées, nous a conduit à chercher une voie de recherche différente. Notamment, en réfléchissant au moyen de résoudre les expressions référentielles dans ce modèle, nous avons abouti à des impasses qui nous ont conduites à imaginer la théorie des Points de Vue.

Le système InterViews a d'autre part eu le mérite indéniable de poser la question de la possibilité de concevoir l'interface linguistique naturelle d'une application dans le même formalisme que l'application. Notre conclusion a été que cela n'était pas possible, et qu'il fallait bel et bien concevoir un système extérieur à l'application pour jouer le rôle de médiateur avec l'utilisateur.

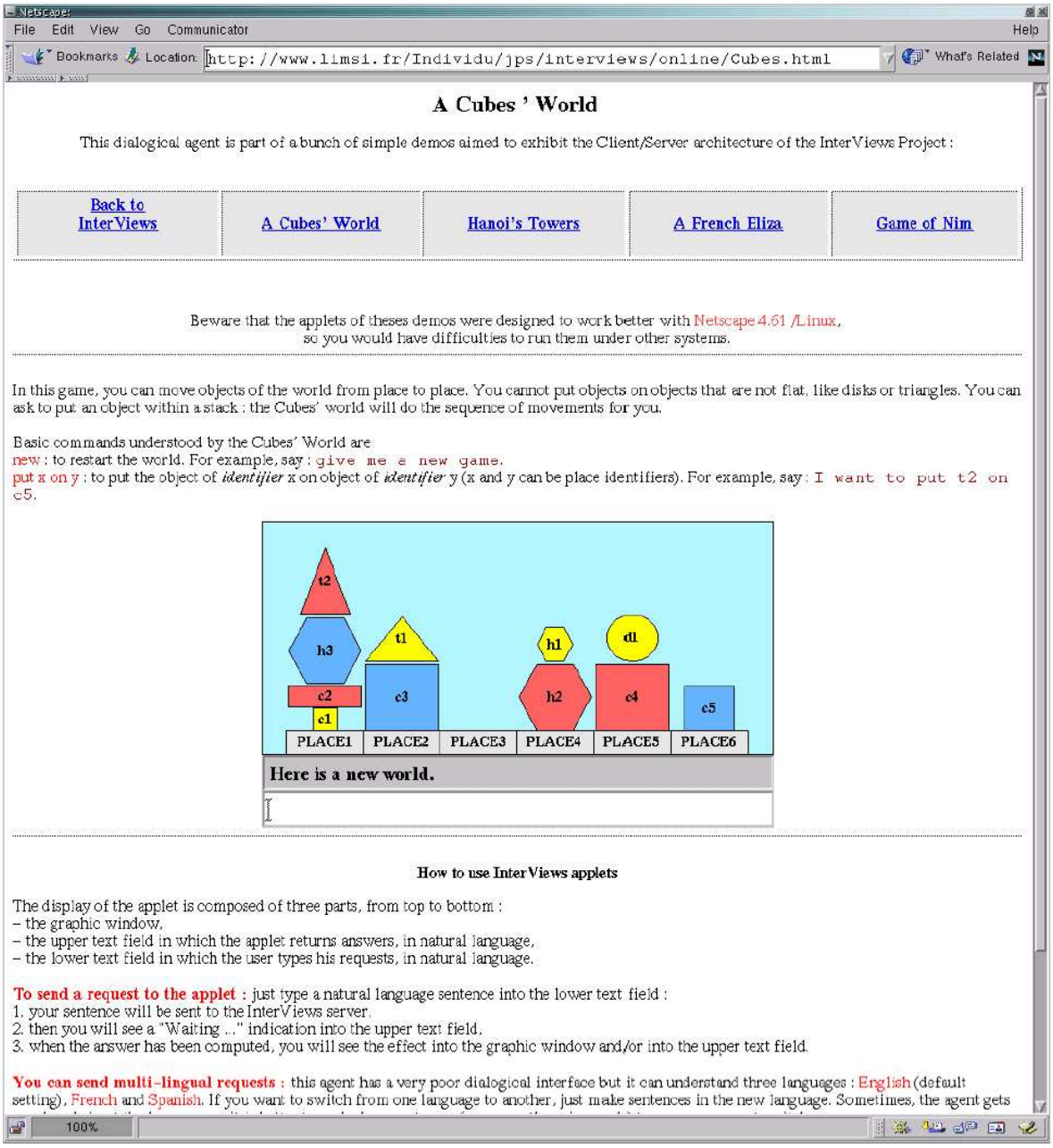


FIG. 10.8 – Page web du « Monde de Cube ».

Conclusion

La nécessité de pouvoir concevoir des agents assistants d'interface, c'est-à-dire des interfaces naturelles capable de servir de médiateur entre un utilisateur humain et un logiciel croît parallèlement au nombre de logiciels disponibles dans le commerce ou dans la communauté du logiciel libre. Le point qui nous est apparu particulièrement important pour développer les agents assistants d'interface concerne leur **généricité**, c'est-à-dire leur capacité à être adaptés et intégrés rapidement à des situations nouvelles (typiquement, permettre de communiquer avec un nouveau logiciel dans le même « cadre », si possible en donnant accès simultanément à plusieurs logiciels).

Modèle d'interprétation

La réalisation d'un système d'interprétation pour les agents assistants d'interface doit selon nous reposer sur trois critères :

- Une théorie linguistique permettant la prise en compte de schémas cognitifs. La grammaire ECG nous a semblée le meilleur point de départ pour réaliser cet objectif.
- Une architecture unifiée, permettant à la fois une bonne cohésion entre les différentes composantes de l'interprétation, une plus grande facilité de développement, et supportant la possibilité de décrire des phénomènes transversaux aux différentes composantes.
- Un modèle permettant de prendre en compte les différentes structures disposant éventuellement d'une topologie, et cela d'une manière intégrée à l'architecture unifiée.

Un problème majeur qui se pose avec les systèmes de dialogue concerne l'usage qu'ils font des ontologies, et la difficulté qu'il y a à adapter les ontologies à un usage générique. Pour cette raison, nous avons argumenté en faveur d'une approche consistant à considérer que les représentations jouent le rôle de points de vue sur les « choses », que ce soient des catégories ou des entités perçues, plutôt que des substituts à ces choses. La vision de la représentation par points de vue permet :

- d'exprimer les relations ontologiques en prenant en compte le contexte, c'est-à-dire de décrire que certaines perceptions *peuvent être vues* d'après un autre aspect dans certaines situations,
- d'exprimer les relations compositionnelles entre un motif de symboles (par exemple linguistique) et des schémas sémantiques évoqués par ce motif.

Nous avons alors introduit le modèle d'interprétation constructionnelle, inspiré de la grammaire de construction et plus particulièrement de la grammaire ECG. Par rapport à cette grammaire, nous avons proposé la notion de *contexte* et la notion de *s-construction*, qui est la

version située des constructions, c'est-à-dire la version des constructions prenant en compte les *contextes*.

Alors que les constructions initialement proposées dans les grammaires de construction ne permettait de décrire que des contraintes sur le **contenu** des instances de schéma ou sur l'**ordre des mots**, nous avons introduit une notion plus générale permettant de décrire des contraintes structurelles sur n'importe quel conteneur qui puisse être implémenté informatiquement et interfacé via le formalisme des *contextes,* avec trois composantes : les *lieux*, les *relations* entre les lieux et les *opérations* sur les lieux.

Nous avons atteint notre objectif de réunir dans un formalisme unifié la puissance descriptive nécessaire pour supporter aussi bien une théorie linguistique comme la grammaire de construction, qu'un modèle reposant sur des structures dotées d'une topologie, comme les domaines de référence, les espaces mentaux, ou les modèles de raisonnement sur l'espace ou le temps. Pour tenter de démontrer la possibilité effective de décrire des phénomènes nécessitant l'usage de telles structures, nous avons proposé d'implémenter un tel phénomène dans le MIC, et pour cela, nous avons conçu le modèle fonctionnel de résolution extensionnelle de la référence.

Modèle fonctionnel de résolution extensionnelle de la référence

Pour introduire ce modèle, nous avons tout d'abord dressé un aperçu rapide des deux grandes familles de modèle de résolution de la référence, à savoir les modèles coréférentiels et les modèles de résolution extensionnelle. Ceci nous a permis de restreindre précisément le cadre de notre recherche.

Nous avons étudié les éléments d'une expression référentielle sous plusieurs point de vues. Nous avons tout d'abord cherché à distinguer les différents rôles ou les différentes fonctions qu'une expression référentielle pouvait jouer dans un énoncé, ceci afin de définir notre objet d'étude. La fonction que nous avons choisi d'étudier est la fonction d'opérateur de reconnaissance, c'est-à-dire la capacité à extraire (reconnaître) les bons candidats à une expression référentielle parmi un ensemble d'éléments.

Nous avons alors posé la notion d'extracteur référentiel, qui est l'aspect fonctionnel d'un élément d'une expression référentielle qui serait vue comme un opérateur de reconnaissance. A partir de cette notion, nous avons cherché à distinguer les différents types d'extracteurs, d'abord en les comparant en fonction de leur catégorie syntaxique (adjectifs, articles, noms), puis en les comparant du point de vue des informations auxquelles les extracteurs accèdent pour remplir leur rôle (informations typologiques, intrinsèques, extrinsèques).

Cette étude nous a permis de déterminer que la catégorie syntaxique ne semblait pas jouer un rôle majeur sur le fonctionnement des extracteurs, ou plus exactement qu'on ne pouvait pas se fonder sur elle pour déterminer les familles d'extracteurs. En nous intéressant davantage aux modes d'accès des extracteurs, c'est-à-dire aux informations auxquelles les extracteurs doivent accéder dans les éléments pour les choisir, nous avons pu nous orienter vers une approche commune à différents extracteurs.

Nous avons alors proposé une modélisation des extracteurs pour la résolution extensionnelle de la référence. Pour cela, nous avons tout d'abord fait une proposition préliminaire fondée sur l'idée que le rôle des extracteurs était modélisable comme une fonction de comparaison.

Cette première proposition s'est cependant heurtée à deux difficultés majeures. Tout d'abord cette modélisation ne permet pas de retourner une réponse négative lorsqu'il n'y a aucun candidat satisfaisant à l'expression référentielle, et ensuite, elle ne permet pas de répondre correctement aux expressions référentielles plurielles.

Ceci nous a conduit a proposer un modèle plus complet, capable de générer une partition ternaire de l'ensemble des éléments, en séparant les candidats préférés, les candidats possibles et les candidats rejetés. Cette modélisation correspond à une extension du modèle des domaines de référence dont nous nous sommes largement inspirés.

Pour valider le fait que le modèle d'interprétation constructionnelle puisse supporter la description d'un tel traitement, qui s'appuie sur une structure dotée d'une topologie particulière, nous avons proposé une spécification du modèle fonctionnel de résolution extensionnelle de la référence à partir de contextes et de s-constructions. Ce développement nous a permis de constater les particularités du développement dans le modèle d'interprétation constructionnelle.

Enfin, nous avons décrit l'aboutissement de nos recherches, le système MICO, en spécificiant un système opérationnel permettant d'implémenter le modèle d'interprétation constructionnelle. Le système MICO est fondé sur la notion d'observateur, qui est l'équivalent opérationnel des s-constructions. La spécification des observateurs comporte des caractéristiques de pondération permettant de contrôler leur exécution. Ceci, couplée avec un principe d'exécution réactif/proactif, et le fait que ce principe soit guidé localement grâce aux contextes, nous permet d'envisager qu'un traitement réalisé sur MICO puisse effectivement être exécuté dans un temps acceptable.

Perspectives

Il est bien entendu indispensable de poursuivre ces recherches, à la fois pour valider le fait que le modèle d'interprétation constructionnelle puisse permettre de modéliser d'autres phénomènes, et pour valider la faisabilité du système MICO ainsi que l'efficacité de son modèle d'exécution des observateurs. C'est un des aspects qui manquent à nos travaux, et qui tient au fait que notre modèle pour MICO a évolué tardivement vers son état actuel, ne nous laissant pas le temps de réaliser son implémentation.

Parallèlement au système d'exécution proprement dit, il nous faudra développer un ensemble d'outils permettant de :

- concevoir les différents composants de MICO avec une interface adaptée,
- visualiser l'exécution d'une interprétation afin de pouvoir « déboguer » une implémentation d'un aspect particulier de l'interprétation,
- concevoir les observateurs de manière distribuée, car la généricité du système étant un de nos objectifs principaux, il faut s'assurer que la plus vaste communauté de chercheurs puisse contribuer au développement sur un base partagée, afin que chacun puisse tester ses implémentations dans un cadre le plus riche possible.

Concernant le développement du système d'interprétation proprement dit, c'est-à-dire le système décrit par les observateurs et les contextes, il devra se faire progressivement, et donnera probablement naissance à des versions parallèles, car dépendantes des choix théoriques fait par les concepteurs pour chaque phénomène. Il devrait être un outil de choix pour valider les théories linguistiques aussi bien que cognitives, au même titre que les systèmes comme SOAR ou ACT-R sont des outils pour valider les théories psychologiques. Pour les linguistes, notamment, il devrait permettre de décrire de manière assez intuitive de nombreux travaux qui jusqu'à présent doivent passer par une phase de traduction dans des grammaires d'unification pour être validés.

Bibliographie

Abney, Steven. Parsing by chunks. In Berwick, Robert, Abney, Steven, and Tenny, Carol, editors, *Principled-Based Parsing*. Kluwer Academic Publishers, Dordrecht, 1991. 33, 40, 61, 145

Adriaens, Geert and Devos, Mark. The Parallel Expert Parser. In Adriaens, Geert and Hahn, Udo, editors, *Natural Language Processing*, pages 350–375. Ablex Publishing Corporation, Norwood, New Jersey, 1994. 39

Allen, James. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26 :832–843, 1983. 43

Allen, James. *Natural Language Understanding*. Benjamin Cummings, 2ème edition, 1995. 15

Allen, James, Byron, Donna, Dzikovska, Myroslava, Ferguson, George, Galescu, Lucian, and Stent, Amanda. An architecture for a generic dialogue shell. *Journal of Natural Language Engineering, special Issue on Best Practices in Spoken Language Dialogue Systems Engineering*, (6(3)) :1–16, 2000. 24

Amsili, Pascal and Bras, Myriam. DRT et Compositionnalité. *Traitement Automatique des Langues*, 39(1) :131–160, 1998. 86

Anderson, John R. *Rules of the Mind*. Erlbaum, Hillsdale, NJ, 1993. 34, 38

Androutsopoulos, Ion, Ritchie, Graeme D., and Thanisch, Peter. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1(1) :29–81, 1995. 12

Bailey, David, Feldman, Jerome, Narayanan, Srini, and Lakoff, George. Modeling Embodied Lexical Development. In *Proceedings of CogSci97*, 1997. 31

Baker, Collin F., Fillmore, Charles J., and Lowe, John B. The Berkeley FrameNet project. In *Proceedings of the COLING-ACL*, Montreal, Canada, 1998. 15

Barwise, Jon and Cooper, Robin. Generalized quantifiers and natural language. *Linguistics and philosophy*, 4(2) :159–219, 1981. 85

Barwise, Jon and Perry, John. *Situations and Attitudes*. MIT-Bradford, Cambridge, 1983. 84

Bergen, Benjamin K. and Chang, Nancy C. Embodied Construction Grammar in simulation-based language understanding. Technical Report TR-02-004, International Computer Science Institute, 2002. xvii, 31, 63, 66, 146

Blache, Philippe. Le rôle des contraintes dans les théories linguistiques et leur intérêt pour l'analyse automatique : les grammaires de propriétés. In *Actes de TALN 2000*, Lausanne, Suisse, octobre 2000. 29, 147

Boros, Janos. Representationalism and Antirepresentationalism - Kant, Davidson and Rorty. *Deutsche Zeitschrift für Philosophy*, 47(4) :539–551, 1999. 50

Bosch, Peter. "Vagueness" is Context-Dependence. A solution to the Sorites Paradox. In Ballmer, Thomas T. and Pinkal, Manfred, editors, *Approaching Vagueness*, pages 189–210. North Holland, Amsterdam, 1983. 98

Briffault, Xavier. *Modélisation informatique de l'expression de la localisation*. Thèse de doctorat, Université Paris VI, mars 1992. 91

Bryant, John. Constructional analysis. Master's thesis, University of California at Berkeley, 2003. 33, 61, 145, 146

Byron, Donna K. and Allen, James F. What's a reference resolution module to do ? Redefining the Role of Reference in Language Understanding Systems. In *Proceedings of the DAARC2002*, 2002. 15, 101

Carnap, Rudolf. *The Logical Syntax of Language*. Philosophy and Scientific Method. International Library of Psychology, 1937. Translation by Amethe Smeaton, Countess von Zeppelin. 85

Chang, Nancy C., Feldman, Jerome, Porzel, Robert, and Sanders, Keith. Scaling cognitive linguistics : Formalisms for language understanding. In *Proceedings of the* 1^{st} *International Workshop on Scalable Natural Language Undestanding*, Heidelberg, Germany, 2002. 31, 65

Charniak, Eugene. On the use of framed knowledge in language comprehension. *Artificial Intelligence*, 11(3) :225–265, 1978. 101, 124

Charniak, Eugene and McDermott, Drew. *Introduction to Artificial Intelligence*. Addison-Wesley, 1985. 50

Chomsky, Noam. *Some Concepts and Consequences of the Theory of Government and Binding*. MIT Press, Cambridge, MA, 1982. 29, 82

Church, Alonzo. *The Calculi of Lambda Conversion*. Princeton University Press, Princeton, NJ, 1941. 79

Cohn, Anthony G. and Hazarika, Shymanta M. Qualitative spatial representation and reasoning : An overview. *Fundamenta Informaticae*, 46(1-2) :1–29, 2001. URL `citeseer.ist.psu.edu/cohn01qualitative.html`. 43

Colmerauer, Alain. Les grammaires de métamorphose. In Bolc, L., editor, *Natural Language Communication with Computers*, number 63 in Lecture Notes in Computer Science. Springer, 1978. 40

Corblin, Francis. *Indéfini, défini et démonstratif. Constructions linguistiques de la référence*. Droz, Genève, 1987. 45, 90

Croft, William. *Syntactic Categories and Grammatical Relations*. University of Chicago Press, Chicago, 1991. 56

Cunningham, Hamish, Gaizauskas, Robert G., and Wilks, Yorick. A General Architecture for Text Engineering (GATE) – A new approach to Language Engineering R&D. Technical Report CS – 95 – 21, Department of Computer Science, University of Sheffield, 1995. 35

Dale, Robert and Reiter, Ehud. Computational interpretation of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19 :233–263, 1995. 98, 101, 103, 119

Davis, Randall. What is knowledge representation ? *AI Magazine*, 14(1) :17–33, 1993. 50

Dubois, Danièle and Grinevald, C. Pratiques de la couleur et dénominations. In *Faits de langues*, pages 11–25. Ophrys, 1999. 104

Dzikovska, Myroslava, Allen, James, and Swift, Mary. When is a union really an intersection ? problems interpreting reference to locations in a dialogue system. In *Proceedings of GOTALOG'2000*, Gothenburg, Juin 2000. 106

Dzikovska, Myroslava, Swift, Mary, and Allen, James. Constructing custom semantic representations from a generic lexicon. In *Proceedings of the 5th International Workshop on Computational Linguistics*. 2003. 15, 23, 109, 110

Earley, Jay C. An efficient context-free parsing algorithm. In Grosz, B. J., Sparck Jones, K., and Webber, B. L., editors, *Natural Language Processing*, pages 25–33. Kaufmann, Los Altos, CA, 1986. 33

Engel, Ralf. SPIN : Language Understanding for Spoken Dialogue Systems Using a Production System Approach. In *Proceedings of ICSLP-2002*, Denver, Colorado, 2002. 16, 145, 146

Eschenbach, Carola, Habel, Christopher, and Smith, Barry, editors. *Topological Foundations of Cognitive Science*, volume 37 of *Reports of the Doctoral Series in Cognitive Science*, 1994. University of Hamburg. 42

Evans, Greg. *The Varieties of Reference*. Oxford University Press, Oxford, 1982. 79

Fauconnier, Gilles. *Espaces Mentaux*. Editions de Minuit, 1984. 33, 44

Ferguson, George and Allen, James. TRIPS : An Intelligent Integrated Problem-Solving Assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573, Madison, WI, July 1998. 15, 22, 36

Ferguson, George, Allen, James, and Miller, Brad. TRAINS-95 : Towards a Mixed-Initiative Planning Assistant. In *Proc. Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, Edinburgh, Scotland, May 1996. 15

Fillmore, Charles J. Frame Semantics. In of Korea, Linguistic Society, editor, *Linguistic in the Morning Calm*, pages 111–38. Hanshin, Seoul, 1982. 15, 63, 108

Fillmore, Charles J. The Mechanisms of Construction Grammar. *Berkeley Linguistics Society*, 14 :35–55, 1988. 30

Fillmore, Charles J. and Kay, Paul. Construction Grammar. *Language*, 64 :501–538, 1995. 30, 31

Fodor, Jerry. Tom swift and his procedural grandmother. *Cognition*, 6(3) :229–247, 1978. 53

Fodor, Jerry A. *The Modularity of the Mind*. MIT/Bradford Books, Cambridge, MA, 1983. 34

Fodor, Jerry A. *Psychosemantics : The Problem of Meaning in the Philosophy of Mind*. MIT Press, Cambridge, MA, 1987. 38

Fodor, Jerry A. *A Theory of Content and Other Essays*. MIT Press, Cambridge, 1990. 86

Frege, Gottlob. On concept and object. In Beaney, M., editor, *The Frege Reader*, pages 181–193. Blackwell Publishers, Malden, MA, 1892. 85

Gaiffe, Bertrand, Landragin, Frédéric, and Guignard, Matthieu. Le dialogue naturel comme un service dans un contexte mutli-applicatif : arguments pour une architecture multi-agents. In Paroubeck and Sansonnet (2004), pages 57–66. 22, 24

Gapp, Klaus-Peter. Basic meanings of spatial relations : Computation and evaluation in 3d space. In *Proceedings of AAAI94*, Seattle, WA, 1994. 43, 106

Gazdar, Gerald. Paradigm merger in natural language processing. In Milner, R. and Wand, I., editors, *Computing Tomorrow : Future Research Directions in Computer Science*, pages 88–109. Cambridge University Press, 1996. 12

Gazdar, Gerald, Klein, Ewan, Pullum, G.K., and Sag, Ivan. *Generalized Phrase Structure Grammar*. Blackwell, Oxford, UK, 1985. 29

Goldberg, Adele E. *Constructions : A Construction Grammar Approach to Argument Structure*. University of Chicago Press, 1995. 30, 56

Grice, Herbert Paul. Logic and conversation. In *Speech Acts*, pages 41–58. 1975. 119

Hartline, P. H. Multimodal integration in the brain. Combining dissimilar views of the world. In Cohen, M. J., editor, *Comparative neurobiology : modes of communication in the nervous system*, chapter 18, pages 309–334. John Wiley and Sons, 1985. 17

Hatwell, Yvette. Transferts intermodaux et intégration intermodale. In Richelle, M., Requin, J., and Robert, M., editors, *Traité de psychologie expérimentale*. Presses Universitaires de France, Paris, 1994. 17

Hayes-Roth, Barbara. A blackboard architecture for control. *Artificial Intelligence*, 26(3) : 251–321, 1985. 70, 147

Heim, Irene. *The semantics of definite and indefinite noun phrases*. Graduate Linguistic Student Association, Amherst, Mass., 1982. 84, 88

Herzog, Gerd and Wazinski, Peter. VIsual TRAnslator : Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, 8(2-3) :175–187, 1994. URL `citeseer.ist.psu.edu/herzog94visual.html`. 106

Hess, Michael. Recent developments in discourse representation theory. In King, M., editor, *Communication with Men and Machines*. University of Geneva, 1991. 85

Hicks, Richard and Essinger, James. *Making computers more human : Designing for human-computer interaction*. Elsevier Science Publishing Ltd., Oxford, England, 1991. 8

Ishida, Yoshiteru. A framework for dynamic representation of knowledge : A minimum principle in organizing knowledge representation. In Brachman, Ronald J., Levesque, Hector J., and Reiter, Raymond, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 170–179, Toronto, Canada, May 1989. Morgan Kaufmann. 18

Jakobson, Roman and Pomorska, Krystyna. *Dialogues*. The MIT Press, Cambdrige, MA, 1983. 81

Johnson-Laird, Philip N. Procedural Semantics. *Cognition*, 5 :189–214, 1977. 39, 53

Johnson-Laird, Philip N. and Miller, George. *Language and Perception*. Cambridge University Press, 1976. 53, 80

Just, Marcel A., Carpenter, Patricia N., and Varma, Shashank. Computational modeling of high-level cognition and brain function. *Human Brain Mapping*, 8 :128–136, 1999. 38

Kamp, Hans and Reyle, Uwe. *From Discourse to Logic*. Kluwer, Dordrecht, 1993. 84, 90

Kant, Immanuel. *Critique de la raison pure*. Hartnoch, 1787. 49, 52

Karttunen, Lauri. Discourse referents. In McCawley, J.D., editor, *Syntax and Semantics 7 : Notes from the Linguistic Underground*, pages 363–385. Academic Press, New York, 1976. 88

Kim, Albert, Chen, Janice, Rippey, Caitlin, and Osterhout, Lee. Combinatory semantic processing can occur independantly of syntactic support. In *Proceedings of the 9th Annual Conference on Architectures and Mechanisms for Language Processing*, Glasgow, Scotland, Août 2003. 34

Kintsch, Walter. The role of knowledge in discourse comprehension : A constructio-integration model. *Psychological Review*, 95(2) :163–182, 1988. 169

Kleiber, Georges. *Sémantique du prototype*. Presses Universitaires de France, 1990. 49

Kosslyn, Stephen M. "The Strategic Eye" : another look. *Minds and Machines*, 11 :287–291, 2001. 35

Kyburg, Alice and Morreau, Michael. Fitting words : Vague language in context. *Linguistics and Philosophy*, 23 :577–597, 2000. 90

Laird, John E., Newell, Allen, and Rosenbloom, Paul S. SOAR : An Architecture for General Intelligence. *Artificial Intelligence*, 47 :289–325, 1991. 38

Lakoff, George and Johnson, Mark. *Metaphors we live by*. University of Chicago Press, 1980. 31

Lammens, Johan M. and Shapiro, Stuart C. Learning symbolic names for perceived colors. In AAAI, editor, *Machine Learning in Computer Vision : What, Why and How ?*, volume FSS93-04. 1993. 105

Landragin, Frédéric. *Modélisation de la communication multimodale. Vers une formalisation de la pertinence*. Thèse de doctorat, Université Henri Poincaré, Nancy, 2003. 46, 93

Langacker, Ronald W. *Foundations of Cognitive Grammar*, volume 1. Stanford University Press, 1987. 29, 30, 80, 106

Lebarbé, Thomas. HACTAR : coopération entre unités, fonctions et domaines linguistiques par une plateforme SMA. In Paroubeck and Sansonnet (2004), pages 7–13. 40

Ligozat, Gérard. Generalized intervals : A guided tour. In *Proceedings of the ECAI-98 Workshop on Spatial and Temporal Reasoning*, Brighton, UK, 1998. 43

Liskov, Barbara. Data abstraction and hierarchy. *SIGPLAN Notices*, 23(5), 1988. 56, 57

Martin, Jean-Claude and Béroule, Dominique. TYCOON : six primitive types of cooperation for observating, evaluating and specifying cooperations. In *Proceedings of the AAAI Fall 1999 Symposium on Psychological Models of Communication in Collaborative Systems*, 1999. 16

Materna, Pavel. A radical challenge. *From the Logical Point of View*, 2(1-2), 1994. 79

Meyer, David E. and Kieras, David E. A computational theory of executive cognitive processes and multiple-task performance. Part 1. Basic mechanisms. *Psychological Review*, 104 :2–65, 1997. 38

Michaelis, Laura A. Type Shifting in Construction Grammar : An Integrated Approach to Aspectual Coercion. *Cognitive Linguistics*, 15 :1–67, 2004. 56

Minsky, Marvin. A framework for representing knowledge. In Winston, P., editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975. 101, 108

Minsky, Marvin. *Mind Design*, chapter A Framework for Representing Knowledge. MIT Press, 1981. 49, 124

Moeshler, Jacques and Reboul, Anne. *Dictionnaire encyclopédique de Pragmatique*. Le Seuil, Paris, 1994. 45

Montague, Richard. Universal Grammar. *Theoria*, 36 :373–398, 1970. 79, 84

Montanari, Ugo. Networks of Constraints : Fundamental Properties and Applications to Picture Processing. *Information Science*, 7(2) :95–132, 1974. 160

Narayanan, Srini. *KARMA : Knowledge-based Action Representations for Metaphor and Aspect*. Phd thesis, University of California at Berkeley, 1997. 31

Newell, Allan and Simon, Herbert A. *Human Problem Solving*. Prentice Hall, EngleWood Cliffs, NJ, 1972. 36

Newell, Allen. Production Systems : Models of control structures. In Chase, W. G., editor, *Visual Information Processing*. Academic Press Inc., New York, 1973. 16, 36, 38, 49, 146

Newell, Allen. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990. 38

Nielsen, Jakob. *Advances in Human-Computer Interaction*. Ablex Publishing Corporation, New Jersey, 1995. 8

Norman, Donald A. *The Invisible Computer : why good products can fail, the personal computer is so complex, and information appliances are the solution*. The MIT Press, Cambridge, Massachusetts, 1998. 7

Norman, Donald A. Affordances and Design. http ://www.jnd.org/dn.mss/affordances-design.html, 1999. 168

Nunberg, Geoffrey. The non-uniqueness of semantic solutions : Polysemy. *Linguistics and Philosophy*, 3(2) :143–184, 1979. 44

Ozkan, Nadine. *Vers un modèle dynamique du dialogue : analyse de dialogues finalisés dans une perspective communicationnelle*. Thèse de doctorat, INP, Grenoble, 1993. xii, 102, 109

Paroubeck, Patrick and Sansonnet, Jean-Paul, editors. *Actes de la Journée d'Etude ATALA Agental « Agents et Langue »*, Paris, France, Mars 2004. ATALA, ATALA. 194, 196

Pasero, Robert and Sabatier, Paul. ILLICO for natural language interfaces. In *Proceedings of the First Language Engineering Convention (LEC)*, Paris, 1995. 40

Pasero, Robert and Sabatier, Paul. ILLICO : un système générique pour la compréhension d'un sous-ensemble du français. Rapport de recherche, Laboratoire d'Informatique de Marseille, 1998. 40

Pateras, Claudia, Dudek, Gregory, and DeMori, Renato. Understanding referring expressions in a person-machine spoken dialogue. In *Proceesings of ICASSP'95*, Detroit, MI, 1995. xi, 105

Pitel, Guillaume and Sansonnet, Jean-Paul. A functional approach for resolution of extensional reference in practical dialogue. In *Proceedings of the International Symposium of Reference Resolution and its Application to question Answering and Summarization*, Venice, Italia, 2003a. 90, 91, 127

Pitel, Guillaume and Sansonnet, Jean-Paul. Toward a uniform architecture for processing application-oriented dialogue. In *Proceedings of AMLaP'03*, Glasgow, UK, 2003b. 145

Pollard, Carl and Sag, Ivan. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994. 29, 31

Popescu-Belis, Andrei. *Modélisation multi-agent des échanges langagiers : application au problème de la référence et à son évaluation.* Thèse de doctorat, Université Paris XI, 1999. 90

Pustejovsky, James. The generative lexicon. *Computational Linguistics*, 17(4) :409–441, 1991. 56

Pustejovsky, James. Linguistic constraints on type coercion. In Saint-Dizier, P. and Viegas, E., editors, *Computational Lexical Semantics*, pages 71–97. 1995. 56

Pylyshyn, Zenon. Is vision continuous with cognition ? The case of impenetrability of visual perception. *Behavorial and Brain Sciences*, 22 :341–423, 1999. 34

Rastier, François. *Sémantique Interprétative*. Presses Universitaires de France, Paris, 1987. 49, 110

Reboul, Anne. Reference, agreement, evolving reference and the theory of mental representation. In Coene, M., Mulder, W. De, Dendale, P., and D'Hulst, Y., editors, *Studia Linguisticae in honorem Lilianae Tasmowski*, pages 601–616. Unipress, Padova, 1999. 90

Reboul, Anne, Balkanski, Cécile, Briffault, Xavier, Gaiffe, Bertrand, Popescu-Bellis, Andrei, Robba, Isabelle, Romary, Laurent, and Sabah, Gérard. Le projet CERVICAL : Représentations mentales, référence aux objets et aux événements. Technical report, LORIA, Nancy, 1997. 46, 88, 90

Recanati, François. *Direct reference : from language to thought*. Basil Blackwell, Oxford, 1993. 88

Russel, Bertrand. On denoting. In Marsh, R., editor, *Bertrand Russell : Logic and Knowledge*. Routledge, 1905. 2001. 79

Russell, Bertrand. Descriptions. In Linsky, L., editor, *Semantics and the philosophy of language*, pages 95–108. University of Illinois Press, Urbana, 1952. 85

Sabah, Gérard. CARAMEL : A computational model of natural language understanding using a parallel implementation. In *Proceedings of the Ninth ECAI*, pages 563–565, Stockholm, 1990. 36

Sabouret, Nicolas and Sansonnet, Jean-Paul. Automated answers to questions about a running process. In *Proc. CommonSense 2001*, pages 217–227, 2001. 22, 178

Sacerdoti, Earl D. *A Structure for Plans and Behaviour*. Elsevier, New York, 1977. 15

Sag, Ivan A. and Pollard, Carl. An integrated theory of complement control. *Language*, 67 (1) :63–113, 1991. 56

Salmon-Alt, Susanne. Reference resolution within the framework of cognitive grammar. In *International Colloquium on Cognitive Science*, San Sebastian, Spain, 2000. 90

Salmon-Alt, Susanne. *Référence et Dialogue Finalisé : de la linguistique à un modèle opérationnel*. Thèse de doctorat, Université H. Poincaré – Nancy 1, Nancy, France, 2001. 46, 90, 100, 101, 109, 121

Sam-Haroud, Djamila and Faltings, Boi V. Consistency techniques for continuous constraints. *Constraints*, 1((1,2)) :85–118, 1996. 160

Sansonnet, Jean-Paul, Pitel, Guillaume, and Sabouret, Nicolas. Un langage de description d'agents dédié à l'interaction dialogique. In *Actes de Modèles Formels de l'Interaction*, pages 295–299, 2003. 17

Schang, Daniel. Application de la notion de cadre aux énoncés de positionnement et de référence. Technical Report 2529, Unité de recherche INRIA Lorraine, 1995. 90, 106

Schang, Daniel. *Représentation et interprétation de connaissances spatiales dans un système de dialogue homme-machine*. Thèse de doctorat, Université Henri Poincaré - Nancy I, 1997. 43, 91

Schank, Roger C. *Inside Computer Understanding*. Erlbaum, Hillsdale, NJ., 1981. 124

Schirra, Jörg R. J. A Contribution to Reference Semantics of Spatial Prepositions : The Visualization Problem and its Solution in VITRA. In Zelinsky-Wibbelt, Cornelia, editor, *The Semantics of Prepositions – From Mental Processing to Natural Language Processing*, pages 471–515. Mouton de Gruyter, 1993. 43

Seneff, Stephanie, Hurley, Ed, Lau, Raymond, Pao, Christine, Schmid, Philipp, and Zue, Victor. Galaxy-II : A reference architecture for conversational system development. In *Proc. ICSLP*, pages 931–934, Sydney, Australia, 1998. 36

Shastri, Lokendra. Advances in Shruti — A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11 :79–108, 1999. 39

Shastri, Lokendra. Types and Quantifiers in Shruti — a connectionist model of rapid reasoning and relational processing. In Wermter, S. and Sun, R., editors, *Hybrid Neural Symbolic Integration*, Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000. 49

Small, Steven L. *Word Expert Parsing : a Theory of Distributed Word-based Natural Language Understanding*. Phd thesis, University of Maryland, 1980. 39, 145

Sowa, John F. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984. 49, 176

Sperber, Dan and Wilson, Deirdre. *Relevance. Communication and Cognition*. Blackwell, Oxford UK and Cambridge USA, 2nd edition edition, 1995. 46, 93

Squire, Larry R. *Memory and Brain*. Oxford University Press, New York, 1987. 38

Stefanini, Marie-Hélène and Demazeau, Yves. TALISMAN : A multi-agent system for natural language processing. In Wainer, J. and Carvalho, A., editors, *Lecture Notes in Artificial Intelligence 991*, Springer-Verlag, 1995. 36

Strube, Michael and Hahn, Udo. Functional centering : Grounding referential coherence in information structure. *Computational Linguistics*, 25(3) :309–344, 1999. 81

Talmy, Leonard. Rubber-sheet cognition in language. In *Actes du 13th Annual Regional Meeting of the Chicago Linguistic Society*, University of Chicago, 1977. 56

Talmy, Leonard. The relation of grammar to cognition. In Waltz, D., editor, *Proceedings of TINLAP-2 (Theoretical Issues in Natural Language Processing)*, New York, 1978. Association for Computing Machinery. 29

Talmy, Leonard. How language structures space. In Pick, H. L. and Acredolo, L. P., editors, *Spatial Orientation*, pages 225–282. Plenum Press, New York, 1983. 29, 106

Tichý, Pavel. *The Foundations of Frege's Logic*. Walter de Gruyter, Berlin-New York, 1988. 80

Turing, Alan M. Computing machinery and intelligence. *Mind*, LIX(236) :433–460, 1950. 8

Uttal, William R. *The new phrenology : The limits of localizing cognitive processes in the brain*. MIT Press, Cambridge, MA, 2001. 35

van Herten, Marieke, Kolk, Herman, and Chwilla, Dorothee. How semantic analysis can overrule syntactic analysis : an ERP study. In *Proceedings of the 9th Annual Conference on Architectures and Mechanisms for Language Processing*, Glasgow, Scotland, Août 2003. 34

Vandeloise, Claude. *L'espace en français : sémantique des prépositions spatiales*. Les Editions du Seuil, Paris, 1986. 106

Vergne, Jacques. Etude et modélisation de la syntaxe des langues à l'aide de l'ordinateur, analyse syntaxique automatique non combinatoire. Habilitation à diriger les recherches, 1999. Université de Caen. 40

Wahlster, Wolfgang, Reithinger, Norbert, and Blocher, Anselm. Smartkom : Multimodal communication with a life-like character. In *Proceedings of Eurospeech2001*, Aalborg, Danemark, 2001. DFKI. 16

Wazinski, Peter. Generating spatial descriptions for cross-modal references. In *Actes de The 3rd Conf. on Applied Natural Language Processing*, pages 56–63, 1992. 106

Winograd, Terry. A procedural model of language understanding. In Schank, R. and Colby, K., editors, *Computer Models of Thought and Language*, pages 152–186. W. H. Freeman, 1973. 13, 182

Wittgenstein, Ludwig. *Tractatus logico-philosophicus*. Annalen des Naturphilosophie. International Library of Philosophy and Scientific Method, Routledge, 1921. Ed. angl. 52

Wittgenstein, Ludwig. *Philosophical Investigations*. Blackwell, Oxford, second edition, 1963. tr. by G. E. M. Anscombe. 52

Wolfram, Stephen. *Mathematica*. Addison-Wesley, Redwood City, CA, 1988. 173, 179

Woods, William A. Procedural semantics as a theory of meaning. In Joshi, A., Webber, B., and Sag, I., editors, *Elements of Discourse Understanding*. MIT Press, Cambridge, MA, 1981. 39

Zock, Michael and Briffault, Xavier. How to learn about space by building and exploring a microworld. In *Fourth International Symposium of Social Communication*, Santiago di Cuba, Cuba, 1995. 91

Table des matières