Contribution à l'apprentissage dans le dialogue homme-machine

Luis Villasenor-Pineda

UNIVERSITE JOSEPH FOURIER - GRENOBLE 1
INFORMATIQUE ET MATHEMATIQUES APPLIQUEES

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Discipline : Informatique

Présentée et soutenue publiquement

par

Luis VILLASEÑOR-PINEDA

le 8 février 1999

Contribution à l'apprentissage dans le dialogue homme-machine

Composition du jury :

Rapporteurs :	Gérard SABAH
	Anne NICOLLE
Examinateurs :	Denis VERNANT
	Yves CHIARAMELLA
Directeur de thèse :	Jean CAELEN

Thèse préparée au sein du laboratoire de Communication Langagière
et Interaction Personne-Système - Fédération IMAG

A Esther por los pasos compartidos
a Damián por los primeros

Un grand merci à Jean Caelen pour m'avoir accueilli dans l'équipe GEOD, pour sa confiance et son aide. Je l'en remercie tout particulièrement.

Merci, bien sûr à ceux qui ont accepté de juger mon travail, Anne Nicolle, Gérard Sabah, Denis Vernant et Yves Chiaramella. Leurs conseils et remarques m'ont permis d'apporter à cette thèse de nombreuses améliorations.

Merci à tous les membres de l'équipe, à mes collègues de bureau successifs, Anne Spalanzani, Ludovic Imberdis et Zakaria Kurdi, et en particulier à Solange Hollard pour avoir effectué cette tâche ingrate de lire et relire ce manuscrit avec attention et minutie.

Enfin, je tiens à exprimer ma gratitude envers toutes les personnes qui ont contribué scientifiquement et humainement à la réalisation de ce travail de recherche, et tout spécialement à Esther Amador pour son aide et son appui pendant toutes ces années.

Table de Matières

Table de Figures

CHAPITRE I

Introduction

L'objectif primordial de la recherche sur les interfaces homme-machine est le développement de modes de communication entre l'homme et la machine qui soient tolérants aux erreurs et d'apprentissage facile [Ogden & Bernick 97].

Une idée poursuivie depuis longtemps est l'utilisation de la langue naturelle comme mode de communication. Ainsi, une interface en langue naturelle ne nécessitera qu'un temps minimum d'apprentissage puisque, à la différence des systèmes à base de langage de commande ou encore des interfaces graphiques, l'utilisateur maîtrise déjà ce mode de communication. Mais, après de nombreuses années de recherche, la construction d'un tel système s'est avérée très difficile –il y a même des chercheurs qui affirment qu'un ordinateur ne sera jamais capable de comprendre une langue naturelle [Winograd & Flores 86]–.

Cependant, grâce aux importants développements technologiques autour de l'ordinateur, et spécialement les progrès en reconnaissance de la parole, l'intérêt pour l'utilisation de la langue naturelle –en contextes finalisés correctement définis– a été relancé, particulièrement dans le cadre des

interfaces multimodales. Bien entendu, la machine n'a qu'une compréhension limitée de la langue naturelle, et cette limitation engendre des obstacles que l'utilisateur doit compenser : il doit mémoriser toutes les restrictions imposées par le concepteur au langage d'interaction, et il doit être préparé aux comportements de la machine en réponse à ses énoncés (car celle-ci n'aura certainement pas les mêmes comportements qu'un interlocuteur humain). En conséquence, la communication en langue naturelle risque d'être plus opaque ou encore d'alourdir le travail lié à la tâche et, si l'on n'y prend pas garde, on peut finir par perdre tous les avantages qu'apporte la langue naturelle.

Néanmoins, la langue naturelle a des avantages par rapport aux autres modes d'interaction entre l'homme et la machine. Des études sur l'interaction multimodale apportent des éléments intéressants [Krause 93, Ogden & Bernick 97, Bellalem & Romary 95, Zanello 97]. On sait par exemple que des situations sémantiquement riches ou des tâches à séquencement complexe font davantage appel aux ressources de la langue naturelle. Les objets non perceptibles, les actions différées, les actions conditionnelles, etc., sont aussi plus facilement exprimables en langue naturelle. Mais aussi, et surtout, avec la langue naturelle il est possible de former facilement de nouveaux concepts et de construire de nouveaux énoncés.

Notre hypothèse est que pour favoriser au maximum l'interaction entre l'homme et la machine, il faut donner à cette dernière des capacités d'apprentissage par le dialogue. Le dialogue étant alors vu comme un double processus de conduite interactive de la tâche et d'acquisition de connaissances.

1.1 Le dialogue homme-machine finalisé

Dans le cas du dialogue homme-machine, l'intérêt primordial n'est pas la réalisation d'un système capable de participer à une conversation quelconque. Nous nous intéressons aux dialogues *finalisés*, c'est-à-dire, orientés par la réalisation d'une tâche ou la résolution d'un problème. Il existe des approches très variées pour aborder ce problème, chacune centrant son intérêt sur un point particulier de la communication homme-machine. Nous

pouvons les regrouper dans trois grandes classes d'applications de systèmes de dialogue [Pierrel & Romary 97] :

- *le transfert ou la saisie d'information* : le dialogue est utilisé pour remplacer un langage formel de communication avec, principalement, une base de données.
- *la commande d'actions* : le dialogue est utilisé pour commander un système automatisé ; le dialogue sert à déterminer *ce qu'il faut faire*.
- *l'acquisition de connaissances* : le dialogue est utilisé pour la construction d'un modèle de référence commun entre les interlocuteurs ; le dialogue sert à déterminer *comment il faut faire*.

Dans notre cas, la modélisation du dialogue homme-machine nous permettra d'envisager l'ordinateur comme un partenaire dans la réalisation d'une tâche. Le dialogue ne serait alors qu'une forme d'interaction qui permet à l'utilisateur d'employer la machine comme un participant dans la réalisation de son plan. L'objectif est le perfectionnement de la machine comme outil adapté et efficace vis-à-vis de l'utilisateur et de la tâche.

En conséquence, notre modèle de dialogue homme-machine prend en compte : (i) les règles qui gouvernent les échanges (langagiers ou non-langagiers) entre les participants du dialogue ; (ii) la coordination des actions qui ont pour finalité la réalisation d'un plan ; (iii) la construction de connaissances communes.

1.1.1 Les échanges : le langage et l'action

La modélisation du dialogue homme-machine nous amène à un champ de recherche pluridisciplinaire : la linguistique, la pragmatique, l'intelligence artificielle sont les principales disciplines concernées. La motivation centrale est la recherche de mécanismes qui expliquent pourquoi une séquence de phrases est cohérente et comment le contexte influe sur l'usage et la compréhension d'une expression linguistique [Grosz et al. 89].

« Le langage se construit par l'action » : c'est un des résultats principaux de Piaget [Piaget 64]. Pour lui, l'enfant construit son langage comme résultat de

l'assimilation des actions sur le monde. Appeler cette personne « maman » ne fonctionne que si celle-ci accourt au cri de « maman ».

Réciproquement « Le langage construit l'action ». C'est la thèse principale de l'école anglo-saxonne en philosophie du langage [Austin 70, Searle 72]. Dans cette théorie, parler c'est agir, produire des actes ; communiquer, c'est agir sur l'interlocuteur. Cette théorie permet de considérer le langage comme une forme d'action et, par généralisation, de considérer le dialogue comme une séquence d'actions planifiées ayant pour objectif un *but visé sous-tendu par une intention*. Cette conception présume qu'il existe un *équilibre rationnel* entre les connaissances, actions et intentions du locuteur. Par conséquent, dans une situation de dialogue finalisé, on suppose que la série d'actions que le locuteur est en train de faire, coïncide avec la réalisation de ses intentions, et que le locuteur adopte seulement des intentions qui sont possibles à réaliser.

« Le dialogue est une interaction : il renvoie le langage à l'action et réciproquement ». Le dialogue est une suite coordonnée d'actions (langagières et non-langagières) devant conduire à un but. Le dialogue avance dans le temps et tend à réduire les divergences entre les interlocuteurs. Au cours de cette interaction ils modifient leurs connaissances, leurs croyances, acquièrent de nouvelles connaissances tant sur la situation, que sur leur interlocuteur ou sur la langue.

1.1.1 La coordination d'actions

Parmi les principales difficultés dans la modélisation du dialogue homme-machine, on trouve l'identification des intentions et des croyances des interlocuteurs. Dans une situation de dialogue, la reconnaissance des intentions et des croyances s'inscrit dans la reconnaissance de plans. Il s'agit en effet de reconnaître le plan de l'interlocuteur. Dans notre cas, le plan est probablement incomplet ou sous-spécifié, et c'est au fur et à mesure du déroulement du dialogue que l'utilisateur précise son plan. La reconnaissance du plan permet à un observateur –la machine– d'inférer les buts de l'utilisateur. Ainsi, l'interprétation des actions de l'utilisateur par rapport à un ensemble de plans possibles rend la machine mieux placée pour coopérer à la

résolution du problème. Elle peut assister l'utilisateur et/ou prendre la relève dans la réalisation de la tâche.

Par ailleurs, pour favoriser l'interaction, il faut donner à la machine des capacités d'action par la langue mais aussi laisser l'opportunité à l'utilisateur d'agir par manipulation directe. Nous n'allons pas distinguer une action élémentaire faite de manière gestuelle et une action complexe faite par la langue et qui pourrait se ramener à une série d'actions gestuelles. Le niveau de granularité d'une action gestuelle est souvent plus fin que celui d'un acte de langage. Par exemple pour dessiner une maison, il suffira peut-être d'ordonner "dessine une maison" alors qu'un dessin gestuel nécessiterait une série d'actions différentes pour tracer le corps de la maison puis le toit, puis les fenêtres, etc.

1.1.2 La construction de connaissances communes

Envisager l'ordinateur comme un partenaire dans la réalisation d'une tâche implique de *comprendre* l'utilisateur pour coopérer à la résolution de son problème. Ces capacités de compréhension de l'utilisateur et d'adaptation à la tache nécessitent de mettre en œuvre des processus d'apprentissage car il n'est pas envisageable de prévoir toutes les situations d'usage ni tous les types d'utilisateurs a priori. La machine doit donc doublement s'adapter :

- d'une part, elle doit acquérir les concepts manipulés à travers la langue et qui sont souvent "naturels" (donc implicites) pour l'utilisateur humain,
- d'autre part, elle doit apprendre des plans d'action dans le contexte d'usage de l'utilisateur et de manière suffisamment générique pour être réutilisables.

Notre idée est donc de fonder notre modèle de dialogue sur la notion d'apprentissage des savoirs et des savoir-faire. Cela conduit notamment à la recherche d'un modèle adéquat de représentation des connaissances apte à faciliter l'apprentissage incrémental, l'élaboration de mécanismes de raisonnement et la construction de plans.

1.2 Objectif

Notre travail propose un modèle de dialogue homme-machine finalisé pour un *cadre actionnel*, c'est-à-dire pour des situations où la communication homme-machine est centrée sur l'exécution d'une séquence d'actions en fonction des buts de l'utilisateur. Comme dans le dialogue humain, il est opportun de profiter du dialogue non seulement pour obtenir des informations et coordonner des actions, mais aussi pour apprendre. Notre idée est donc de fonder notre modèle de dialogue sur la notion d'apprentissage des savoirs et des savoir-faire.

L'idée envisagée est la définition d'un modèle de dialogue indépendant de la tâche, où le domaine de travail est défini par l'application mais où le système de dialogue ne connaît pas *a priori* la ou les tâches possibles à faire. La machine apprendra les tâches, et à partir de cette connaissance –acquise à travers le dialogue– elle pourra intervenir et même guider l'utilisateur dans des situations analogues ultérieures. L'interprétation des actions montrées par l'utilisateur par rapport aux tâches apprises permet à la machine de coopérer à la résolution du problème.

Cette interprétation entraîne la modélisation des *états mentaux* de l'utilisateur, exprimant ses croyances et ses intentions. A partir de celles-ci, la machine peut interpréter et produire les actes de dialogue. Cette modélisation ne tente pas de donner un comportement humain à la machine mais de trouver les raisons d'agir de l'utilisateur.

La modélisation des échanges dialogiques entre l'utilisateur et la machine est réalisée dans un cadre logique fondé sur l'action qui permet d'expliquer le déroulement d'un dialogue homme-machine [Caelen 95]. Elle contient des éléments d'une logique épistémique –pour la représentation des connaissances–, d'une logique de l'action –pour décrire les actions et les effets produits par ces actions–, et d'une logique dialogique –pour exprimer les engagements, les interruptions et les incompréhensions lors des échanges dialogiques, en fonction des buts du dialogue. De cette façon, notre modèle intègre la gestion des échanges langagiers et non-langagiers, la coordination des actions à partir de la reconnaissance de plans en fonction des actions

observées, et la construction de connaissances communes à travers l'acquisition de nouvelles tâches.

1.3 Plan

Les chapitres suivants développent les aspects que nous avons brièvement abordés dans ce premier chapitre. Le deuxième chapitre est un état de l'art dans le domaine des modèles de dialogue homme-machine. Celui-ci présente les travaux marquants sur le dialogue homme-machine et rejoint les travaux sur le dialogue humain issus de la linguistique et des sciences cognitives. Puis il introduit notre approche : ses fondements et ses caractéristiques par rapport aux autres modèles présentés. Le troisième chapitre décrit les deux processus sur lesquels l'attitude coopérative de la machine est fondée : l'acquisition des savoir-faire au travers d'exemples et, la reconnaissance d'une situation à partir des actions observées. La première partie de ce chapitre introduit la représentation du plan de la tâche et discute son apprentissage ; la deuxième partie présente la méthode utilisée pour la reconnaissance de tâches. Le quatrième chapitre présente notre logique pour le dialogue coopératif homme-machine, qui intègre des éléments épistémiques, dynamiques et dialogiques. Dans le cinquième chapitre, nous présentons la mise en œuvre informatique de notre modèle du dialogue. Cette dernière est une plate-forme qui nous a permis de valider notre modèle de dialogue et d'évaluer l'application d'un tel dialogue dans une situation de conception. Enfin, le dernier chapitre donne les conclusions et perspectives qui résultent de ce travail.

CHAPITRE II

Dialoguer : une activité conjointe

La question qui se pose dans le dialogue homme-machine peut être résumée comme la recherche des processus qui interviennent dans la compréhension et la production des échanges dialogiques dans une situation déterminée.

Bien sûr dans le cas du dialogue homme-machine l'intérêt primordial n'est pas la réalisation d'un système capable de participer à une conversation quelconque, mais celui des dialogues *finalisés*, c'est-à-dire, orientés par la réalisation d'une tâche ou la résolution d'un problème. D'ailleurs, notre intérêt particulier est centré sur des situations dites de *conception* où la machine sert comme outil pour la création. Dans ce cadre, les processus associés à la conception sont étroitement liés au déroulement du dialogue. La réalisation de la tâche, sa description, la répartition en sous-tâches –et même l'apprentissage et la reconnaissance– sont des mécanismes sur lesquels nous allons focaliser notre attention.

Ce chapitre présente les principales idées sur l'évolution des systèmes de dialogue homme-machine (DHM ci-après) avec leurs contributions et restrictions vis-à-vis d'un dialogue en situation de conception. Cette présentation, qui ne prétend pas être exhaustive, montre les travaux marquants sur le DHM et rejoint, en certains cas, les travaux sur le dialogue humain issus de la linguistique et des sciences cognitives. Dans la dernière section, nous discuterons notre approche : ses fondements, en rappelant nos objectifs, et ses caractéristiques par rapport aux autres modèles présentés.

2.1 Les premiers pas dans le dialogue homme-machine

Les premiers systèmes du DHM ont centré leur attention sur une simple paire question-réponse. Dans ces systèmes, il n'y pas de véritable dialogue. L'interaction a été réduite à l'élaboration d'une réponse à une question posée. Ainsi, l'interaction a été fortement contrainte et le fonctionnement correct du système dépendait de la pertinence des informations présentées par l'utilisateur au cours d'un tel dialogue.

Le premier système à aborder des questions intéressantes pour le dialogue a été le système SHRDLU [Winograd 71]. Le système était capable de mener un dialogue dans un domaine d'application réduit facilement maniable. Le système SHRDLU simulait un bras mécanique, dans un monde de blocs, que l'utilisateur pouvait commander pour réaliser des actions élémentaires (par exemple, saisir un bloc, l'empiler sur autre, etc.) et les résultats de ces actions étaient visualisés dans un écran.

U :	*Pick up a big red block*
S :	*ok*
U :	*Grasp the pyramid*
S :	*I don't understand which pyramid you mean*
…	

Figure 2.1. Exemple de dialogue avec SHRDLU

Dans ces circonstances –un domaine simple avec des actions définies *a priori*– il est possible de gérer un dialogue simple sans problèmes d'incompréhension. Le système SHRDLU a été un apport important, mais

l'évolution des compétences du système s'est avérée impossible, à cause de la rigidité de ses fondements et d'une grande dépendance au domaine.

Un autre système important dans les premiers pas du DHM était le système GUS [Bobrow et al. 77]. GUS a été un système de renseignements de vols aériens. Le dialogue dans ce cas était destiné a remplir un schéma sous-jacent, qui est à l'origine des interventions de GUS et est la base de l'interprétation de celles de l'utilisateur. Les figures 2.2 et 2.3 montrent un dialogue avec GUS et les schémas sous-jacents qui le guident.

GUS	Bonjour, je suis GUS, je peux vous aider à effectuer un voyage simple en avion. Où voulez-vous aller ?
Client	Je veux aller à San Diego, le 28 mai.
GUS	À quelle heure voulez-vous partir ?
Client	Je dois être à San Diego avant 10 heures.
GUS	Le vol 102 de P.S.A. qui arrive à 9h15 vous convient-il ?
Client	Quel est le vol suivant ?
GUS	Le vol 310 d'Air Californie, qui part à 8h30 et arrive à 10h
Client	Je prendrai le premier.
GUS	À quel nom est-ce que je fais la réservation ?
Client	Dan Bobrow
GUS	J'ai confirmé le vol suivant : vol 102 de P.S.A. mercredi 28 mai de San José à San Diego qui part à 7h30 et arrive à 9h15 Quel jour voulez-vous revenir ?
Client	vendredi soir
GUS	Le vol qui part à 19h45 vous conviendrait-il ?
Client	Parfait
GUS	J'ai confirmé le vol suivant : vol 307 de P.S.A. vendredi 30 mai de San Diego à San José qui part à 19h45 et arrive à 21h30 Au revoir

Figure 2.2. Exemple de dialogue avec GUS (d'après [Bobrow et al. 77])

Bien que GUS soit capable de réaliser un dialogue simple, l'utilisation de schémas impose un cadre rigide qui empêche de gérer des phénomènes propres du dialogue comme le malentendu ou l'incompréhension.

Schéma DIALOGUE

Attribut	***Type***	***Procédures attachées***
Client	Personne	créer schéma
Date actuelle	Date	obtenir date
Sujet	Voyage	créer schéma

Schéma VOYAGE

Attribut	***Type***	***Procédures attachées***
Aéroport local	Ville	défaut : lieu présent
Aéroport externe	Ville	
Aller	Trajet	créer schéma
Retour	Trajet	créer schéma
…		

Schéma TRAJET

Attribut	***Type***	***Procédures attachées***
Ville départ	Ville	lien : aéroport local
Ville arrivée	Ville	demander au client
Date trajet	Date	demander au client
Horaire départ	Heure	demander au client
Horaire arrivée	Heure	calculer
…		

Figure 2.3. Exemple de schémas utilisés par GUS (d'après [Bobrow et al. 77])

Ces premiers systèmes ont mis en évidence la nécessité de déterminer le rôle de l'information extra linguistique –la connaissance associée au domaine, l'intention du locuteur, les différences entre les croyances du locuteur et celles de l'auditeur– dans le traitement du dialogue. Pour aborder ces questions, nous retrouvons un ensemble de travaux qui cherchent dans les théories du dialogue humain des clés pour établir la structure du dialogue, l'enchaînement des actes et la place de l'intention du locuteur. La section suivante introduit brièvement les principes de ces différentes théories. Les sections 2.3, 2.4 et 2.5 présentent différents modèles du dialogue qui les utilisent.

2.2 Considérations issues de la philosophie du langage

2.2.1 La théorie des actes de langage

La théorie la plus importante et la plus influente dans les travaux de DHM est la théorie des *actes de langage* fondée par [Austin 70] et développée par [Searle 72, Vanderveken 88]. Cette théorie rompt avec la conception saussurienne de la linguistique selon laquelle, la langue –l'objet de la linguistique– doit être différenciée de son usage –la parole–. Austin montre avec les énoncés dits *performatifs* l'impossibilité de dissocier les sens de l'énonciation. Par exemple, en disant « Je déclare la séance ouverte » le locuteur réalise en même temps l'acte d'ouvrir la séance. Autrement dit, cette énonciation ne décrit pas un état du monde mais accomplit une action, un changement du monde. C'est le point de départ de l'idée selon laquelle *dire, c'est faire.*

Cette fonction actionnelle est généralisée à tous les types d'énoncés. Ainsi, pour Austin, l'unité minimale de la communication humaine n'est ni la phrase ni une autre expression. C'est l'accomplissement de certains types d'actes. En prononçant une phrase, un locuteur accomplit un (parfois plusieurs) actes. Austin propose une analyse des *actes de langage* selon trois aspects :

a) la fonction *locutoire*, qui est le contenu propositionnel de l'énoncé (et comprend la production d'une suite de signes selon des règles de la langue).

b) la fonction *illocutoire*, qui est la valeur performative de l'énoncé ou encore l'effet de l'énoncé sur la situation discursive ;

c) la fonction *perlocutoire*, qui est l'effet de l'acte sur l'allocutaire.

De cette manière Austin fait la distinction et établit le lien entre l'acte propositionnel et l'acte illocutionnaire, ainsi dans les énoncés suivants :

- S'il vous plaît, reprenez de la soupe !
- Vous allez reprendre de la soupe.
- Reprendrez-vous de la soupe ?

La même proposition est exprimée dans l'accomplissement de trois actes illocutionnaires différents : une requête, une prédiction et une question. Si la notion centrale en ce qui concerne les contenus propositionnels est celle de la *vérité*, la notion correspondante en ce qui concerne les actes illocutionnaires est celle de *satisfaction*.

2.2.2 La taxonomie des actes de langage de Searle

A la suite d'Austin, Searle a développé la théorie des actes de langage. Il a notamment insisté sur la fonction illocutoire comme unité fondamentale de signification, et proposé une taxinomie des fonctions illocutoires. Cette taxonomie est réalisée à travers trois critères principaux : les différences quant à la *finalité de l'acte* –le but illocutoire–, *la direction d'ajustement* entre les mots et le monde, et les différences touchant les *états psychologiques* exprimés.

A partir de ces critères, Searle arrive à une classification des actes illocutoires en cinq types :

1. Les *assertifs*. Ce type d'actes engagent le locuteur à la vérité de la proposition exprimée. Les mots s'ajustent au monde. Par exemple, « conclure » ou « déduire ».
2. Les *directifs*. Avec ce type d'actes le locuteur cherche à obtenir que l'auditeur fasse quelque chose. Le monde s'ajuste aux mots par l'intermédiaire d'autrui. Par exemple « demander » ou « questionner ».
3. Les *commissifs*. Sont les actes illocutionnaires qui engagent le locuteur à l'accomplissement d'une action future. Le monde s'ajuste aux mots, grâce à l'obligation de la part du locuteur. Par exemple « promettre ».
4. Les *expressifs*. Ces actes expriment l'état psychologique du locuteur. Il n'y a pas de trace d'ajustement entre le monde et les mots. Par exemple « remercier » ou « déplorer ».
5. Les *déclaratifs*. C'est le cas des actes performatifs d'Austin : au moment de l'énonciation de l'acte s'instaure la correspondance souhaitée entre le contenu propositionnel et la réalité. Le monde et les

mots s'ajustent mutuellement, par exemple « déclarer la séance levée ».

2.2.3 Considérations sur la logique illocutoire

Les travaux de Searle et Vanderveken [Searle & Vanderveken 85, Vanderveken 88, 90] formalisent la théorie des actes de langage en proposant une *logique illocutoire* capable de définir tous les types d'actes de langage à partir de la taxonomie précédente.

Le développement de la logique illocutoire a permis la distinction entre deux valeurs fondamentales d'un acte : sa réussite et sa satisfaction. Un acte est *réussi* tout simplement s'il peut être accompli. Cela veut dire que le locuteur exprime son but, en respectant les conditions syntaxico-sémantiques, de façon sincère, dans un contexte où les présupposés du locuteur sont vérifiés, selon le degré d'engagement approprié, etc. Par conséquent, la réussite d'un acte dépend uniquement des conditions en relation à sa force illocutoire (dans la logique illocutoire, la force illocutoire n'est plus l'unité élémentaire des actes de langage, elle est décomposée en six constituants interdépendants). Par contre, la satisfaction d'un acte fait intervenir sa réponse, c'est-à-dire sa composante perlocutoire. Un acte est *satisfait* si son contenu propositionnel est vrai ou devient vrai suite à son énonciation.

2.2.4 Une extension dialogique de la logique illocutoire

Les travaux de Trognon et Brassac [Trognon & Brassac 92] proposent une *interprétation dialogique* de la logique illocutoire, dans cette conception, la force illocutoire d'un acte n'est effective qu'après la réponse de l'interlocuteur car elle est fonction de cette réponse.

Les notions de *réussite* et de *satisfaction* d'un acte de langage, reprises de travaux de Vanderveken, sont fondamentales dans ce modèle. C'est à travers ces deux notions que les auteurs étendent la logique illocutoire pour considérer un dialogue comme une suite d'actes de langage qui s'enchaînent les uns aux autres. Ainsi, pour Trognon et Brassac, la satisfaction d'un acte dépend de sa réponse, alors que sa réussite dépend des conditions dans

lesquelles il a été accompli. Un acte satisfait est nécessairement réussi, mais l'inverse n'est pas vrai.

Par exemple, un acte directif, dont la direction d'ajustement va « du monde aux mots » (dans la mesure où le locuteur tente de rendre le monde conforme à ses mots, cf. § 2.2.2) est satisfait si le comportement de l'allocutaire rend le contenu propositionnel vrai et si ce comportement est motivé par l'acte lui-même. Ainsi, un ordre tel que « je vous demande de sortir » est satisfait si l'interlocuteur sort et, s'il sort à cause de cet ordre, et non pas parce qu'il avait justement prévu de sortir.

Le fondement de ce modèle réside donc dans le jeu croisé des conditions de satisfaction d'un acte de langage et la réussite d'un autre acte de langage avec lequel il est en relation. Par ailleurs, ce modèle s'appuie aussi sur la notion de paire adjacente et sur le modèle structurel de la conversation (cf. § 2.3) pour expliquer comment sont liés les actes de langage dans un dialogue [Ghiglione & Trognon 93]. La notion de paire adjacente est utilisée pour rendre compte de la structure linéaire des actes de langage quand ils appartiennent au même plan de discours ; et le modèle structurel pour rendre compte de la structure hiérarchique du dialogue due aux relations de dépendance entre les actes de langage.

Parmi les modèles de dialogue qui sont fondés sur la théorie des actes de langage, on trouve deux lignes principales. Un premier groupe de travaux, liés principalement à l'étude du discours dans le domaine de la linguistique, tente d'analyser le dialogue à travers sa structure. Un deuxième ensemble de travaux, issus notamment du domaine de l'informatique, présente le dialogue comme une activité planifiée. Les prochains paragraphes présentent les principales notions de ces différentes approches.

2.3 Les approches fondées sur la structuration

Ces modèles ont été conçus pour étudier et comprendre les mécanismes du dialogue à travers sa structure, c'est-à-dire qu'ils décrivent le dialogue en définissant ses constituants et la structure qui spécifie les liens de composition, de succession ou de subordination entre eux. Le modèle

genevois [Roulet et al. 85] est le plus représentatif parmi les travaux de cette approche. Le modèle permet d'identifier *a posteriori* les liens hiérarchiques et fonctionnels entre les constituants du discours. L'analyse hiérarchique du modèle genevois repose sur plusieurs niveaux de *constituants :* les *incursions* sont les plus grandes unités dans un dialogue, les *échanges* sont les plus petites unités dialogiques, les *interventions* sont les constituants uniques des échanges, et les *actes de langage* les plus petites unités monologiques.

- *L'acte de langage* est l'unité minimale de l'énonciation (c'est l'acte de langage d'Austin [Austin 70]). Un acte de langage correspond à un but communicatif et il peut être soit un acte *directeur* qui explicite le but du locuteur, soit un acte *subordonné* à cet acte directeur.

- *L'échange* est l'unité minimale de l'interaction. Un échange contient au moins deux actes de langage de locuteurs différents.

- *L'intervention* est l'unique constituant de l'échange. Une intervention peut être soit *monologale* quand elle est formée d'un ou plusieurs actes de langage produits par le même locuteur, soit *dialogale* si elle est formée d'une *intervention principale* et d'un *échange subordonné.*

- *L'incursion* correspond au niveau de description le plus général et est délimitée par la rencontre et la séparation des deux interlocuteurs. Une incursion « conforme aux règles » comprend un *échange d'ouverture* du dialogue, une suite d'échanges définissant chacun une *transaction* particulière, puis un *échange de clôture* du dialogue.

En ce qui concerne l'analyse fonctionnelle, le modèle genevois identifie principalement deux types de liens fonctionnels entre les constituants de la structure hiérarchique : les fonctions *illocutoires* et les fonctions *interactives.* Les constituants d'un discours monologique sont reliés par des fonctions interactives, et les constituants d'un discours dialogique sont reliés par des fonctions illocutoires. Ainsi les interventions d'un échange sont liées par des fonctions illocutoires : soit la fonction illocutoire *initiative* pour les interventions qui donnent des droits ou imposent des contraintes à l'interlocuteur (par exemple assertion, demande d'information) ; soit la fonction illocutoire *réactive* qui renvoie à une intervention antérieure (par exemple acceptation, confirmation, refus). Par ailleurs, les fonctions

interactives permettent d'expliciter le rôle d'un constituant subordonné à l'égard de l'acte directeur (par exemple explication, justification).

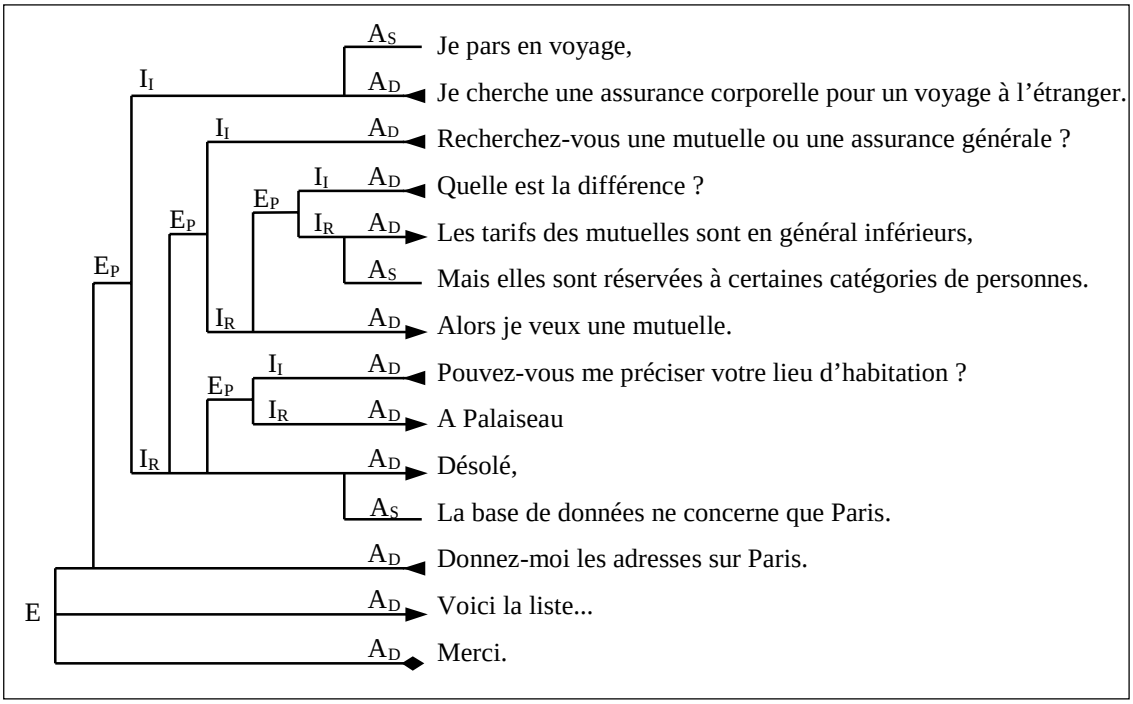


Figure 2.4. Structure d'un discours d'après le modèle Genevois (d'après [Lehuen 97]).

À travers ces deux analyses, il est possible de rendre compte de la structure d'une conversation. La figure ci-dessus montre un exemple de ces analyses. Un échange E est constitué par deux interventions I : l'une avec une fonction initiative I_I et l'autre avec une fonction réactive I_R. Une intervention est constituée d'au moins un acte de langage directeur A_D et, éventuellement, d'autres actes subordonnés A_S. Un échange peut être subordonné à une intervention soit comme échange préliminaire E_P soit comme échange complémentaire E_C.

2.3.1 Les travaux de Grau, Sabah et Vilnat

Les travaux développés aux LIMSI par [Grau et al. 94, Vilnat 97] s'inscrivent dans cette approche structurelle. Le contrôle du dialogue est réalisé en trois pas. D'abord, une analyse informationnelle permet de reconnaître le thème d'une phrase et d'établir un lien entre celle-ci et les phrases précédentes. Les phrases sont interprétées en utilisant des

connaissances pragmatiques rencontrées dans des situations concrètes. A partir de cette connaissance, on construit un arbre qui représente l'évolution des thèmes, il précise comment les thèmes ont été introduits et développés. Ce modèle regroupe tout ce qui a été dit lors du dialogue sur les différents sujets abordés indépendamment de leur ordre d'apparition.

Ensuite, une analyse de l'intention communicative est réalisée pour reconnaître l'acte de langage et sa fonction illocutoire. Pour les auteurs un acte de langage est formé d'un prédicat généralisé (à chaque concept de l'application correspond un prédicat généralisé) et d'une fonction illocutoire (pour rendre compte de l'intention communicative). Les fonctions illocutoires sont inspirées du modèle genevois. Elles sont classées de la manière décrite dans la figure 2.5, afin de les étiqueter comme initiatives ou réactives, et de les associer aux interventions de l'utilisateur.

fonctions illocutoires initiatives	fonctions illocutoires réactives	
offre-requête	acceptation	refus
demande d'information	réponse positive	réponse négative
demande de confirmation	confirmation	infirmation
assertion	évaluation positive	évaluation négative

Figure 2.5. L'association des fonctions illocutoires initiatives et réactives (d'après [Vilnat 97]).

La fonction illocutoire dépend du contexte du dialogue, et en particulier de l'état de la structure du dialogue. Cette structure conserve l'information concernant le déroulement du dialogue. Les auteurs utilisent une *grammaire* du dialogue –basé dans les liens hiérarchiques proposés par le modèle genevois– pour la construction de cette structure. Ainsi le résultat est un arbre où les *actes de langage* sont les feuilles qui sont donc réunies en *échanges*, et ceux-ci en *interventions*.

Finalement, le dernier pas dans l'interprétation d'une intervention de l'utilisateur est la gestion de l'interaction. Le but est de fournir des informations au gestionnaire du dialogue pour détecter des cas de malentendu

et de choisir un comportement adéquat afin de garder un dialogue cohérent (pour traiter ces situations les auteurs adaptent les idées proposées par [Luzzati 92]).

2.3.2 Problèmes liés aux modèles fondés sur la structuration

Le principal problème de l'approche structurale est qu'elle fait l'hypothèse qu'il existe une structure qui décrit le dialogue à partir des régularités dans les échanges, et que cette structure est déterminée *a priori.* Ainsi tous les dialogues, au moins théoriquement, peuvent être analysés à partir de cette structure. Bien que la fonction première des modèles structuraux soit de décrire le fonctionnement d'un dialogue et pas de doter un système informatique d'une compétence dialogique, différents systèmes de DHM ont été obtenus (par exemple, SUNDIAL [Bilange 92], STUDIA [Chevallier 92], Diabolo [Vilnat 97]). Ces systèmes tentent de mettre en œuvre une *grammaire* pour le dialogue, qui sert à identifier l'intervention de l'utilisateur et à produire l'intervention pertinente de la part de la machine. Les modèles structuraux adoptent une démarche descriptive et non explicative du dialogue, en effet, ils minimisent la dépendance du dialogue envers le contexte qui l'entoure et le caractère opportuniste du dialogue.

2.4 Les approches fondées sur la planification

Les modèles de dialogue fondés sur la planification partent de l'hypothèse que les personnes impliquées dans un dialogue ont un *comportement rationnel* qui les conduit à élaborer et à exécuter des plans pour atteindre leurs buts. De façon simplifiée, la planification est définie comme le calcul *a priori* des étapes nécessaires pour parvenir à un but à partir d'une situation initiale ; et un plan est un ensemble organisé d'actions dont la réalisation doit permettre d'atteindre le but. Ainsi, l'idée de base d'un tel système de dialogue est de déterminer le plan de l'utilisateur qui est la raison et la motivation de son comportement –et, par conséquent, à l'origine de ses interventions dialogiques–, pour ainsi fournir les réponses pertinentes vis-à-vis du but visé.

Ce type d'approche est fondé sur le principe des actes de langage de [Austin 70] selon lequel *communiquer, c'est agir*. Dans cette conception, la poursuite d'un but donne lieu à la planification d'une séquence d'actions dont certaines sont langagières et d'autres non. De cette manière, une intervention langagière est alors jugée comme une action dans un plan pour atteindre un but. Les travaux de Cohen et Perrault [Cohen & Perrault 79] sont les premiers à formaliser la théorie des actes de langage en termes de planification.

2.4.1 Les travaux de Allen et Perrault

Le premier modèle de ce type est celui de [Allen & Perrault 80]. L'idée centrale consiste à reconnaître le plan de l'interlocuteur, d'après un ensemble de plans possibles préétablis, en utilisant des techniques de planification classiques en intelligence artificielle. Ainsi, pour avoir une connaissance plus précise de ce dont a besoin l'interlocuteur lorsqu'il engage un dialogue, le modèle utilise le plan reconnu pour déterminer les obstacles potentiels qu'il peut rencontrer lors de son exécution et tenter de lui fournir les éléments nécessaires à leur résolution. Par exemple, devant la requête du prix d'un billet de train, le système pourra inférer que le but de l'utilisateur est de prendre le train, la connaissance du prix du billet étant une des étapes du plan permettant la réalisation de ce plan. Le système fournira alors non seulement le prix du billet, mais pourrait également proposer de lui vendre le billet (ce qui constitue la prochaine étape pour réaliser le but). De plus, le système pourra détecter les obstacles qui existent à la réalisation du but en évaluant les connaissances de l'utilisateur. Il détectera par exemple que l'utilisateur ne connaît pas le quai de départ du train. La réponse du système pourra donc inclure également cet élément.

Il s'agit donc non pas réellement de construire un plan pour résoudre un problème, mais plutôt d'être en mesure de reconnaître un plan, à partir de ce que dit l'utilisateur. Le fait que l'utilisateur pose une question est toujours reconnu comme un acte de langage faisant partie du plan construit par lui. Le formalisme choisi pour représenter les actes de langage est très proche de celui utilisé classiquement pour la planification en intelligence artificielle. Les actes de langage sont représentés en utilisant des règles à la STRIPS

[Fikes & Nilsson 71]. Ainsi ils sont définis avec des *préconditions* qui doivent être vérifiées pour que l'action puisse avoir lieu ; un *corps* qui décrit de façon détaillée l'action, éventuellement en la décomposant en actions plus simples ; et des *effets* qui sont des faits à ajouter ou supprimer après l'exécution de l'action. Par exemple, la figure ci-dessous montre la règle pour l'acte de langage INFORMER.

INFORMER (Locuteur, Auditeur, Proposition)

Préconditions:	VEUT (Locuteur, INFORMER (Locuteur, Auditeur, Proposition)) SAIT (Locuteur, Proposition)
Effets:	SAIT (Auditeur, Proposition)
Corps:	ENONCER (Locuteur, Auditeur, Proposition)

Figure 2.6. Exemple de schéma d'action pour un acte de langage.

La reconnaissance du plan

Les connaissances utilisées par le système pour reconnaître un plan ne sont ni celles de l'interlocuteur puisque c'est le système lui-même qui construit le plan, ni celles du système puisque le plan construit est supposé être celui de l'interlocuteur. Ces connaissances sont celles que le système prête à son interlocuteur : ce que le système croit que l'interlocuteur croit. Par conséquent, les règles utilisées par la machine M pour calculer les buts de l'interlocuteur A (pour Agent) sont généralement du type « M croit que A veut P ». Ces règles ont recours à des opérateurs de logique modale, pour désigner la croyance B (pour *Believe*) et la volonté W (pour *Want*), ainsi « M croit que A veut P » sera noté simplement comme : MBAW P.

Le modèle d'Allen et Perrault considère trois ensembles de règles pour que la machine M infère le plan du locuteur A :

Règles d'inférence concernant les actions :

Ces règles tentent de retrouver le plan du locuteur A à partir des actions observées.

- *Si le locuteur A a pour but P, et si P est une précondition de ACT, alors il est possible que A veuille réaliser l'action ACT.*

  MBAW (P) $\Rightarrow$ MBAW (ACT) *précondition-action*

- *Si A veut exécuter l'action B, et si B appartient au corps de ACT, alors il est possible que A veuille réaliser l'action ACT.*

  MBAW (B) ⇒ MBAW (ACT) *body-action*

- *Si E est l'effet de l'exécution de ACT, et si A veut réaliser l'action ACT, alors A souhaite E.*

  MBAW (ACT) ⇒ MBAW (E) *action-effect*

- *Si A veut qu'un agent N veuille l'action ACT, alors il est possible que A souhaite l'action ACT.*

  MBAW(N WANT ACT) ⇒ MBAW (ACT) *want-action*

Règles d'inférence concernant les connaissances :

Ces règles font le lien entre les buts pour acquérir une certaine connaissance et les buts ou actions qui dépendent de cette connaissance.

- *Si A veut savoir si la proposition P est vraie, alors il est possible que A souhaite que P soit vraie.*

  MBAW (A KNOW-IF (P)) ⇒ MBAW (P) *know-positive*

- *Si A veut savoir si la proposition P est vraie, alors il est possible que A souhaite que P soit fausse.*

  MBAW (A KNOW-IF (P)) ⇒ MBAW (¬P) *know-negative*

- *Si A veut savoir si la proposition P(a) est vraie, alors il est possible que A cherche à établir la valeur d'un terme X de P.*

  MBAW (A KNOW-IF (P(a))) ⇒ MBAW(A KNOW-REF (X : P(X))) *know-value*

- *Si A veut connaître la valeur d'un terme X pour la proposition D(X), alors il est possible que A veuille le but ou l'action P qui comporte la proposition D(X).*

  MBAW (A KNOW-REF (X : D(X))) ⇒ MBAW(P(X : D(X))) *know-term*

Règles pour la reconstruction du plan :

Le processus de reconstruction d'un plan peut être décrit de la même manière que le processus d'inférence du plan. Les règles suivantes sont des règles inverses de celles qui concernent les actions.

- *Si l'agent X veut réaliser l'action ACT et si P est une précondition de ACT, alors il doit s'assurer que P soit vraie.*

  X VEUT (ACT) ⇒ X VEUT (P) *action-précondition*

- *Si l'agent X veut réaliser l'action ACT et si B appartient au corps de ACT, alors il doit réaliser B.*

  X VEUT (ACT) ⇒ X VEUT (B) *action-body*

- *Si l'agent X veut obtenir l'effet E et si E est produit par l'action ACT, alors il veut réaliser ACT.*

  X VEUT (E) ⇒ X VEUT (ACT) *effect-action*

- *Si l'agent X veut rendre vraie la proposition P, alors il doit connaître sa valeur de vérité.*

  X VEUT (P) ⇒ X VEUT (X KNOW-IF (P)) *know*

Ces ensembles des règles sont utilisés dans la construction de *plans partiels*, composés de deux parties. Une partie nommée *l'alternative* qui est construite en effectuant toutes les inférences possibles à partir des actions observées ; et une deuxième partie, nommé *l'expectative* qui est reconstruite par planification à partir du but supposé du locuteur. Chaque plan partiel est ensuite évalué à l'aide de diverses heuristiques visant à prédire dans quelle mesure il peut être le plan « correct ». Chaque heuristique permet d'appliquer, à chaque plan partiel, un facteur multiplicatif qui augmente ou diminue ses chances de se retrouver sélectionné.

2.4.2 Les travaux de Grosz

Une autre ligne dans l'approche fondée sur la planification sont les travaux de Grosz et Sidner [Grosz & Sidner 90]. Leurs travaux sont fondés sur leur propre théorie de structuration du discours. Cette théorie décrit la structure d'un discours à partir de trois composants : une structure linguistique, une structure intentionnelle et un état attentionel. La *structure linguistique* consiste en segments de discours et les relations qui existent parmi eux. La *structure intentionnelle* consiste en objectifs poursuivis par un segment de discours. Enfin, *l'état attentionel* est une abstraction du centre d'intérêt des interlocuteurs. Pour la modélisation de la structure intentionnelle ces auteurs proposent un cadre fondé sur la planification : le *plan partagé* [Grosz & Kraus 96, Lochbaum 94]. Leur modèle tente de tenir compte de l'état de connaissances et de croyances des interlocuteurs au cours du dialogue. C'est un modèle basé sur les états mentaux des interlocuteurs pour la représentation de plans collaboratifs.

FIP(G, α, T_p, T_α, R_α, C_α)

Un agent G a un plan individuel complet au temps T_p, pour réaliser une action α au temps T_α utilisant le schéma R_α dans le contexte C_α.

1. *G* a un schéma pour α
   $R_\alpha = \{\beta_i, \rho_j\} \wedge BEL(G, R_\alpha \in Schémas(\alpha), T_p)$

2. Pour chaque sous-action βi du schéma,
   G a l'intention de faire lui-même chaque sous-action β_i
   $Int.To(G, \beta_i, T_p, T_{\beta i}, R_{\beta i}, C_{\beta i/\alpha})$

   Il existe un schéma $R_{\beta i}$ pour β_i tel que

   i. *G* croit que lui peut réaliser β_i selon
   $(\exists R_{\beta i})\ [\ BEL(\ G, CBA(\beta_i, R_{\beta i}, T_{\beta i}, T_p, contraints(C_\alpha) \cup \{\rho_j\}\ )) \wedge$

   ii. *G* a un plan individuel complet pour β_i
   $FIP(G, \beta_i, T_p, T_{\beta i}, R_{\beta i}, C_{\beta i/\alpha})\ ]$

Notations :
FIP : Plan Individuel Complet *(Full Individual Plan)*
BEL : opérateur de croyance *(Believe)*
Int.To : opérateur indiquant qu'un agent a l'intention de faire une action
CBA : un agent a la capacité de réaliser une action *(Can Bring About)*

Figure 2.7. Définition d'un Plan Individuel Complet (d'après [Lochbaum 93])

Il distingue ce qui concerne l'état mental de celui qui élabore un plan, de ce qui est lié à l'établissement du schéma permettant de réaliser une action. Ainsi un agent G a un *plan individuel* pour un acte α si : (i) G sait comment réaliser α (G connaît un schéma pour α) ; (ii) G croit qu'il peut réaliser les sous-actions spécifiées dans le schéma de α ; (iii) G a l'intention de réaliser les sous-actions ; et (iv) G a un plan individuel pour chaque sous-action. La figure 2.7 montre la définition d'un plan individuel complet. Grosz et Kraus donnent également la forme de plans individuels et de plans partagés complets et partiels. Dans les plans partagés, plusieurs agents collaborent à l'exécution d'une tâche. Un plan partagé comporte des *croyances mutuelles* par rapport aux intentions et capacités des agents. En outre, puisqu'un plan partagé est un plan multi-agent, les sous-plans qui en découlent sont à réalisés soit par un plan partagé soit par un plan individuel d'un des agents.

2.4.3 Les travaux de Sabah et Vilnat

Un autre travail dans cette ligne est la proposition de [Sabah 97, Vilnat 97]. Le modèle de dialogue proposé est composé de deux phases. La

première (basée sur une approche structurale cf. 2.3.1) interprète l'intervention de l'utilisateur et la seconde tente de reconnaître le plan de l'utilisateur d'après les résultats partiels obtenus dans la première phase. Cette deuxième phase est compose d'un pré-processeur pragmatique et d'un module de calcul des intentions.

Le pré-processeur pragmatique est chargé d'examiner la structure linguistique de l'intervention dans la recherche d'indices qui permettent de déterminer la nature intentionnelle des actions de l'utilisateur. D'après ces indices, une analyse simple permet d'avoir des informations sur l'état mental de l'utilisateur, qui aidera à inférer l'intention de l'utilisateur. Une fois que l'intention de l'utilisateur est établie, et par conséquence son but, la machine détermine son propre plan pour atteindre ce but. C'est lors du déroulement de ce plan que la machine rencontre de possibles difficultés. A ce moment là, un sous-plan est activé. Par exemple, si la machine est incapable d'identifier un paramètre pour la réalisation d'une action du plan alors, on active un sous-plan qui consiste à demander l'information manquante. Ce résultat est renvoyé au générateur d'actes de langage et les actes générés provoquent la mise à jour de la structure du dialogue.

L'originalité de ce travail réside essentiellement dans le pré-processeur pragmatique. L'idée est de tirer le maximum d'informations au niveau linguistique des interventions pour déduire l'état mental de l'utilisateur et ainsi optimiser la reconnaissance de ses intentions.

2.4.4 Les travaux de Litman et Allen

Dans les modèles précédents, la planification est le processus chargé de la gestion de l'ensemble du dialogue. Pour aborder ce problème, Litman [Litman & Allen 87, 90] propose un système de reconnaissance de plans qui étend les travaux d'Allen [Allen & Perrault 80]. Le système proposé est fondé sur un ensemble de schémas de plans spécifiques du domaine –*les plans du domaine*–, un ensemble de schémas de métaplans indépendants du *domaine* – *les métaplans*[1] *du discours*– et un algorithme de reconnaissance incrémentale.

[1] le principe de la *métaplanification* est introduit par Wilensky [Wilensky 83]

Ainsi, le modèle de Litman prend en compte les relations qu'entretiennent les énoncés et les plans du domaine à travers les métaplans du discours et, de cette façon, il est possible de gérer des échanges ne concernant pas directement la tâche, tels que les sous-dialogues de clarification. Litman identifie six métaplans du discours qui sont répartis en trois classes : les métaplans pour suivre le cours normal du dialogue, les métaplans pour la clarification et la correction d'un échange, et les métaplans pour introduire un plan ou changer le thème du dialogue.

Les métaplans du discours sont exprimés de la même façon que les plans de domaine (en utilisant le formalisme STRIPS), mais en termes de prédicats relatifs à la structure ou à l'état d'un autre plan. Par exemple, dans la figure 2.8, le métaplan INTRODUIRE-PLAN introduit un nouveau sujet de discussion qui n'a pas de rapport avec le sujet de conversation précédent. Le *corps* est la demande de l'action du locuteur vers l'auditeur ; les *effets* sont l'adoption par l'auditeur du plan introduit et l'action immédiate à exécuter dans le plan ; et les *contraintes* sont que l'action demandée doit être un élément du plan introduit et que l'agent de l'action demandée doit être l'auditeur.

INTRODUIRE-PLAN (Locuteur, Auditeur, Action, Plan)

Corps:	DEMANDE (Locuteur, Auditeur, Action)
Effets:	VEUT (Auditeur, Plan) PROCHAIN (Action, Plan)
Contraintes:	ELEMENT (Action, Plan) AGENT (Auditeur, Action)

Figure 2.8. Exemple de métaplan du discours

Enfin, l'algorithme de reconnaissance vise à intégrer chaque énoncé de l'utilisateur à la représentation du dialogue déjà élaborée. Pour cela, il comporte une pile de plans qui représente l'état courant du dialogue. Initialement, la pile est vide. Dès le premier énoncé, la pile est constituée de deux plans : un plan du domaine (situé à la base) et un métaplan du discours qui fait le lien entre l'énoncé de l'utilisateur et le plan du domaine. S'il faut entrer dans un sous-dialogue de clarification, par exemple, le système empile

un second métaplan, ayant pour objet, non plus le plan du domaine, mais le premier plan du discours.

2.4.5 Problèmes liés aux modèles fondés sur la planification

Dans les premiers modèles de ce type, on a vu que la planification est le seul processus chargé de la gestion du dialogue. Bien que les travaux de Litman tentent d'aborder le problème en incluant des métaplans pour le discours, on ne voit pas clairement comment ces systèmes peuvent aborder des phénomènes rencontrés fréquemment dans le dialogue, comme le respect des règles conversationnelles (par exemple, à une question correspond une réponse). Mais surtout, cette approche suppose qu'il existe un nombre fini de plans pour chaque action possible, que les actions possibles sont aussi en nombre fini et, en plus, que tout le monde les connaît.

Un autre problème, particulièrement dans le modèle proposé par Grosz, est posé par la lourdeur du formalisme. La représentation d'un plan pour aboutir à une action, même simple, devient très complexe. La logique sur laquelle repose le modèle est très complexe, ce qui rend les processus de raisonnement difficiles à mettre en œuvre. Bien que ce type de logiques fassent l'objet de nombreuses recherches en intelligence artificielle, la manipulation de ces connaissances n'est pas toujours bien définie.

2.5 Les approches fondées sur l'interaction rationnelle

Une autre ligne de travail dans le dialogue homme-machine est la description plus précise de la *rationalité*. Cette approche étend les modèles fondés sur la planification pour tenter d'expliciter le dialogue comme une *interaction rationnelle*. Elle fait l'hypothèse que le dialogue n'est pas une activité primaire, mais une conséquence d'un comportement intelligent, c'est-à-dire qu'un système de dialogue doit être un système intelligent, et le dialogue un résultat de son comportement.

REQUEST (x, y, acte)

Préconditions :	*BMB (y, x,* *SINCERE (x, GOAL (x,* *P-GOAL (y, DONE (y, acte))))) ∧* les agents ***x*** et ***y*** croient mutuellement que ***x*** est sincère par rapport à son objectif : que ***y*** adopte le but de réaliser l'***acte***.
	BMB (y, x, *ALWAYS (COMPETENT (y, DONE (y, acte)))) ∧* ***x*** et ***y*** croient mutuellement que ***y*** a la compétence pour accomplir l'***acte***.
	BMB (y, x, ¬ ALWAYS (¬DONE (y, acte))) ***x*** et ***y*** croient mutuellement que ***y*** n'a pas fait l'***acte.***
Corps :	*BMB (y, x,* *GOAL (x, BEL(y,* *GOAL (x, P-GOAL (y, DONE (y, acte))))))* une demande de ***x*** a pour objectif de modifier les croyances de ***y*** en transmettant le but de ***x,*** que ***y*** adopte comme but propre la réalisation de l'***acte***.
Effets :	*BMB (y, x, GOAL (x, ◊ DONE (y, acte)))* après une demande ***x*** et ***y*** croient mutuellement que le but de ***x*** est que ***y*** exécute vraisemblablement l'***acte***.

Notations :
BEL (x, p) : *x* croit *p*
BMB (x, y, p) : *x* et *y* croient mutuellement *p*
SINCERE (x, p) : *x* est sincère par rapport à *p*
GOAL (x, e) : *e* est une action poursuivie par *x*
P-GOAL (x, e) : *e* est un but « persistant » de *x*
DONE (y, a) : l'action *a* vient d'être faite par *y*
ALWAYS (p) : *p* est toujours vraie
COMPETENT (x, a) : *x* a le savoir-faire nécessaire pour réaliser *a*

Figure 2.9. Définition d'une demande entre deux agents (d'après [Cohen & Levesque 84])

2.5.1 Les travaux de Cohen et Levesque

Les premiers travaux sur cette approche sont ceux de Cohen et Levesque [Cohen & Levesque 90a, 90b]. L'idée est d'établir une théorie de la communication dans une théorie de l'interaction rationnelle, laquelle est

soutenue par une théorie de l'action qui, finalement, est définie par des états mentaux des agents. Leur première proposition[2] [Cohen & Levesque 84] a été définie en fonction de quatre opérateurs modaux (*BEL* croyance, *BMB* croyance mutuelle, *GOAL* action poursuivie, *AFTER* action réalisée). Avec la combinaison de ces opérateurs il est possible de décrire comment les objectifs des agents et leurs plans déterminent leurs actions. Egalement, il est possible de caractériser comment les croyances sur les plans et croyances des autres agents influent sur les plans et croyances d'un agent. Ainsi, les auteurs caractérisent l'interaction rationnelle à partir de propriétés distinctives d'un agent coopératif –sincérité, savoir-faire, obligation– et des effets dus aux échanges langagiers. Par exemple, la figure 2.9 montre les préconditions nécessaires et les effets d'une demande.

2.5.2 Les travaux de Sadek

Selon Sadek [Sadek 96] le dialogue est défini comme une activité motivée et non primitive, c'est-à-dire qu'il émerge des capacités de raisonnement, de perception et d'apprentissage. Un agent participe à un dialogue car il cherche un *équilibre rationnel* (entre ses croyances, incertitudes, intentions et plans). De cette façon, à partir d'un cadre de comportement rationnel –utilisant un modèle mental fondé sur les attitudes primitives de croyance, incertitude et intention– et un modèle de l'action il est possible de construire un *agent rationnel dialoguant*. Ainsi l'auteur fait deux hypothèses : (i) communiquer peut être considéré comme un comportement rationnel reposant sur des actions ; (ii) l'établissement et la poursuite du dialogue peuvent être entièrement justifiés par les principes de comportement rationnel.

Les concepts d'attitudes mentales et d'action sont formalisés dans le cadre d'une logique modale de premier ordre. Elle utilise trois opérateurs modaux : *K* la croyance, *U* l'incertitude et *C* le choix. Ainsi la formule *K(i, p)* peut être lue comme « l'agent *i* croit que *p* est vraie », la formule *U(i, p)* comme « l'agent *i* est incertain de la vérité de *p* », et la formule *C(i, p)* comme « l'agent *i* désire que *p* soit vraie ». L'intention n'est pas une attitude mentale

[2] Celle-ci est par la suite modifiée dans [Cohen & Levesque 90a].

primitive, elle repose sur la notion de *but* et *persistance* et elle est définie à partir des opérateurs *C* et *K*. Enfin, pour permettre le raisonnement sur l'action, les opérateurs *Faisable*, *Fait* et *Agent* sont définis. La formule *Faisable(a, p)* peut être lue comme « l'action *a* peut avoir lieu, après quoi *p* sera vraie », la formule *Fait(a, p)* comme « l'action *a* vient juste d'avoir lieu, avant quoi *p* était vraie » et la formule *Agent(i, a)* comme « l'agent *i* est l'unique agent des événements apparaissant dans *a* ». Dans ce cadre, la théorie proposée rend compte des phénomènes d'observation et de planification d'actions, ainsi que du problème de la révision et de la persistance des croyances à la suite de la production ou de l'observation d'une action. A partir de ces opérateurs le formalisme décrit les principes de rationalité d'un agent vers ses plans et actions, comme le comportement coopératif de l'agent.

Pour l'auteur, un acte communicatif est caractérisé par des *préconditions de faisabilité* et par *des effets perlocutoires*. Les premières sont les conditions qui doivent être satisfaites pour que l'acte soit planifié et les deuxièmes caractérisent les raisons pour lesquelles un acte est réalisé. Par exemple, la figure ci-dessous montre la définition de l'acte communicatif INFORMER.

<i, INFORMER (j, ϕ) >

Préconditions de Faisabilité :

$K(i,\phi) \wedge \neg K(i, K(j,\phi)) \wedge \neg K(i, K(j,\neg K(i,\phi))) \wedge \neg K(i, K(j, K(i, K(j,\phi))))$

Pour que *i* soit capable d'informer *j* que ϕ est vraie, il faut que *i* croie lui-même *p* et que *i* ne croie pas que déjà *p*. De plus, *i* ne doit pas penser que *j* le croit non capable de l'informer au sujet de *p*, et *i* ne pense pas que *j* le croit non pertinent à son égard.

Effets Perlocutoires :

$K(j,\phi)$

L'acte d'informer à pour effet la détermination de la valeur de vérité de ϕ dans l'état mental de *j*.

Figure 2.10. Définition de l'acte INFORMER (d'après [Sadek 90])

2.5.3 Problèmes liés aux modèles fondés sur l'interaction rationnelle

Le travail de Sadek est fondé sur l'hypothèse que l'établissement et la poursuite d'un dialogue coopératif sont entièrement justifiés par les principes de comportement rationnel et, par conséquent, il n'est pas nécessaire de

disposer d'un modèle structurel du dialogue pour participer à un dialogue. Les travaux de Cohen et Levesque se trouvent dans la même situation. Bien qu'ils arrivent à aborder des propriétés spécifiques de la communication (par exemple si une phrase est interrogative ou déclarative [Cohen & Levesque 90b]), ils évitent les problèmes de cohérence du dialogue. Ainsi, dans ces modèles, le discours n'existe pas en tant que tel, l'interaction n'a pas d'existence propre. Ces modèles ne considèrent pas les contraintes qu'impose le processus dialogique lui-même [Vernant 92].

Ce type de modèles expliquent le dialogue comme une manifestation de l'aptitude coopérative d'un agent, et celle-ci est définie à travers des principes primitifs qui se trouvent à la base de l'adoption de l'intention, de la sincérité ou de la pertinence, par exemple. On ne voit pas clairement à quel niveau il faut arriver dans la description de ces principes primitifs, pour arriver à un comportement coopératif quand l'agent en question est une machine.

Par ailleurs, il faudra faire un compromis opératoire dû aux difficultés dans la construction pratique d'un système informatique. Il ne faut pas seulement aborder le problème de la reconnaissance de plans *per se* –un processus complexe et coûteux– il faut aussi mettre en œuvre un démonstrateur capable de réaliser des inférences dans une logique de ce type.

2.6 L'approche adoptée

Cette section explique et présente notre approche en fonction de nos objectifs et examine les différences et recouvrements avec les approches préalablement exposées. Les premiers paragraphes présentent les idées et considérations sur lesquelles s'appuie notre approche, pour finir avec un regard comparatif sur les approches précédentes.

2.6.1 Considérations sur l'approche de Vernant : *l'interaction communicationnelle*

Parmi les principaux apports des travaux de Vernant [Vernant 92, 97] figure la redéfinition d'une théorie des actes de discours qui tente d'élargir le

caractère monologique de la théorie des actes de langage, pour considérer le caractère dialogique du procès de communication. Sa théorie relève d'une théorie générale de l'action. Les actes de discours sont d'abord des actes – dans le même sens qu'Austin– et sont produits, comme n'importe quelle autre action, par un agent qui possède des croyances, intentions et émotions. Cependant, le plus important est que ces actions sont des *actes de discours*, c'est-à-dire qu'ils sont provoqués par une intention de communication. Ainsi, leur objectif principal est celui de faire connaître, de partager des croyances, des intentions, etc. De cette façon, un acte de discours est une action sur autrui qui passe d'abord par la modification de ses états mentaux, dont la fonction première est communicationnelle, « de faire connaître à l'allocutaire une conviction du locuteur à propos d'un fait du monde et d'obtenir en retour une réaction cognitive ».

Pour analyser les actes de discours l'auteur considère deux possibilités : une analyse *interactionnelle* et une autre *transactionnelle*. L'analyse interactionnelle rend compte de la fonction communicationnelle des actes de discours et met en relation le locuteur et l'allocutaire comme coauteurs d'un procès dialogique. Les actes de discours sont dirigés par des contraintes d'interaction coopérative –formulées initialement par Grice sous la forme de maximes– qui conditionnent les fonctions dialogiques des actes de discours. L'analyse transactionnelle rend compte du contexte et de sa relation avec l'acte de discours. La coopération langagière est orientée vers une transformation du monde par ou avec autrui.

A partir de ces deux analyses, Vernant arrive à une nouvelle classification des actes de discours, basée sur ces quatre critères :

1. Il n'existe pas de médiation discursive pour obtenir la transformation du monde. On retrouve dans ce critère les performatifs d'Austin. L'action dite est réalisée par son énonciation, l'interaction produit directement une transaction.

2. La relation entre interaction (les mots) et transaction (le monde). On retrouve dans ce critère la direction d'ajustement de Searle. Mais pour Vernant il existe seulement quatre possibilités : (i) la double direction : les *déclarations*, l'interaction a en même temps valeur de transaction ; (ii) la direction des mots au monde : les *assertifs* expriment l'obligation du

locuteur vers le monde ; (iii) la direction du monde aux mots : les engageants, l'interaction consiste en un engagement sur une transaction future ; et (iv) la direction des mots aux mots : les *métadiscursifs*, quand l'acte de discours prend pour objet transactionnel cet acte même.

Il faut noter que la direction d'ajustement vide, pour le cas des expressifs de Searle, est considérée par Vernant comme non pertinente, car il est impossible de concevoir une interaction qui ne vise pas une transaction. « Si mots et monde n'ont pas de relation, l'acte de discours perd tout sens, toute finalité ».

3. La relation entre l'agent du dire de l'acte langagier lui-même et le sujet du dit de l'action non langagière décrite.

4. La distinction du contenu de la transaction quand il porte sur une action ou un état. Ainsi, on fait une distinction entre un acte qui impose une action à autrui et celui qui provoque une modification de son état mental.

A partir des ces analyses interactionnelle et transactionnelle, Vernant exprime sa perspective pragmatique de l'acte de discours. Néanmoins pour rendre compte d'un modèle de dialogue, il faut considérer les contraintes que le processus dialogique lui-même impose.

2.6.2 Le dialogue comme une *action conjointe*

D'après Vernant, les actes de discours dans une interaction effective se manifestent en étroite interdépendance. C'est-à-dire qu'un type d'acte de discours ne peut pas être déterminé par la seule forme linguistique de son énonciation. La classification de l'acte dépend de sa *fonction dialogique* (par exemple, sa place et son rôle au cours du dialogue). Dans les dialogues suivants, le même énoncé prend des valeurs différentes par le contexte dialogique dans lequel il se trouve.

Question de A :	Donnez-moi un exemple d'acte indirect
Réponse de B :	« Pouvez-vous me donner l'heure ? »

Question de A :	Pouvez-vous me donner l'heure ?
Réponse de B :	Bien sûr, il est 15 heures

Figure 2.11. Le rôle de l'acte en fonction du dialogue (d'après [Vernant 97]).

Ainsi le dialogue est construit pas à pas par échanges entre le locuteur et l'allocutaire. Bien sûr, ces échanges sont gouvernés par des règles syntaxiques et sémantiques, mais le dialogue reste une activité ouverte imprévisible. Ainsi, une interaction langagière est une *activité conjointe* car à travers la fonction dialogique d'un acte de discours, chaque interlocuteur participe à la création du dialogue.

Néanmoins, un tel processus interactionnel n'est pas aléatoire, car il respecte toujours des règles qui restent implicites et servent à guider le dialogue au fur et à mesure de son déroulement (par exemple, les principes de rationalité et de coopération).

L'auteur propose un *modèle projectif du dialogue* pour décrire la production des échanges dialogiques. Vernant présente son modèle [Vernant 92] dans le cadre d'un dialogue informatif élémentaire. En ce cas, la situation est simple : il y un unique médium (les paroles) et deux acteurs. Dans ce type de dialogue les rôles des interlocuteurs sont bien différenciés : le locuteur (le demandeur) demande une information que l'allocutaire (le répondant) peut fournir. Ainsi le dialogue est vu comme un processus de convergence à l'issue duquel la demande d'information est satisfaite.

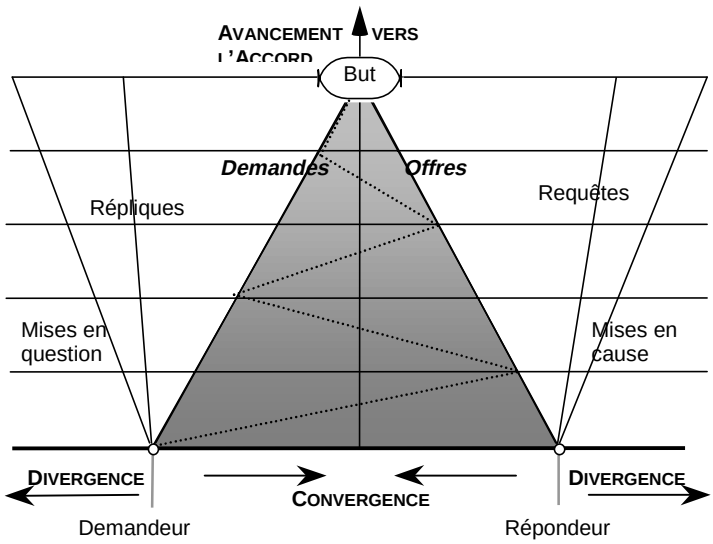


Figure 2.12. Modèle *projectif du dialogue informatif* (d'après [Vernant 92])

Ce processus de convergence, lorsqu'il est idéal, se déroule comme une avancée progressive en réduisant la différence informationnelle initiale

jusqu'à parvenir au partage de l'information. L'avancement du dialogue se réalise à travers une alternance strictement respectée entre demandes et offres. Pourtant, il peut également présenter des interventions qui s'éloignent du but (manifester une incompréhension, le refus d'une requête, etc.), celles-ci font sortir l'intervention de l'espace de convergence défini par les lignes d'*offres* et de *demandes*. La figure 2.12 montre graphiquement ce processus de convergence.

2.6.3 Considérations sur le travail d'Ozkan

Les travaux d'Ozkan [Ozkan 94] partent de cette approche communicationelle et l'étendent pour rendre compte des mécanismes dans des dialogues humains orientés par la tâche où : les rôles des interlocuteurs sont contraints, et la réalisation de la tâche implique la construction d'un univers commun.

Parmi les diverses contributions du travail d'Ozkan, il y deux aspects sur lesquels s'appuie notre travail : la définition d'acte de discours en termes de modalités et l'adaptation du modèle de dialogue de Vernant.

2.6.3.1 Les modalités d'un acte comme fonctions dialogiques

Pour Ozkan un acte est « l'unité manifeste minimale ayant une visée » et elle est contrainte par son domaine d'application. Ainsi un acte peut être de type *actionnel* si la visée porte sur le monde, sur la tâche même ; ou de type *communicationnel* si la visée porte sur les connaissances communes. Un autre critère pour différencier les actes est l'intervention ou la non-intervention de l'allocutaire par rapport à la visée de l'acte : un acte est *direct* si sa visée n'implique pas la participation de l'allocutaire, par contre, un acte est *indirect* si l'allocutaire est impliqué. A partir de ces deux critères, l'auteur définit quatre modalités :

- *faire* (f) : un acte dont la visée est actionnelle directe, un acte destiné à produire un changement du monde sans l'intervention de l'allocutaire.

- *faire-faire* (ff) : un acte dont la visée est actionnelle indirecte, un acte destiné à produire un changement du monde par l'intermédiaire de l'allocutaire.

- *faire-savoir* (fs) : un acte dont la visée est communicationnelle directe, un acte destiné au changement des connaissances communes.

- *faire-faire-savoir* (ffs) : un acte dont la visée est communicationnelle indirecte, un acte destiné au changement des connaissances communes par intermédiaire de l'allocutaire.

A travers ces modalités, l'interaction est vue comme un phénomène dynamique, une succession de changement d'états. Ainsi, pour analyser un dialogue, il faut considérer trois types d'états : les *états du monde*, la description des objets et ses relations ; les *états épistémiques*, les connaissances et croyances communes résultantes de l'interaction ; et les *états déontiques*, les choix, obligations et interdits qui lient les interlocuteurs. Le travail d'Ozkan centre son intérêt sur les deux derniers types d'états et les décrit à partir des modalités suivantes :

modalités épistémiques : *Savoir* (S) et *Croire* (C). La différence entre Savoir et Croire dépend du degré de certitude de la connaissance. Ainsi une connaissance incertaine fait l'objet de Croire, et une connaissance certaine l'objet de Savoir. Une connaissance certaine est consensuelle, elle fait l'objet d'un accord entre les interlocuteurs, à la différence de la connaissance incertaine.

modalités déontiques : *Devoir* (D) et *Pouvoir* (P). Elles dénotent un état d'obligation ou un état de motivation. Un Devoir est une obligation et implique un choix forcé. Un Pouvoir est un choix possible. C'est précisément à travers ces modalités que la relation entre les interlocuteurs est décrite. Dans la situation étudiée par Ozkan, les deux interlocuteurs ont des rôles différents : un interlocuteur est considéré comme l'instructeur et le deuxième comme le manipulateur. Cette relation de supériorité est saisie par ces modalités déontiques.

A partir de la description d'un acte par modalités, l'auteur arrive à expliquer la *cohérence* d'enchaînement d'actes dans un dialogue.

2.6.3.2 L'adaptation du modèle de Vernant

Le travail d'Ozkan adapte le modèle de dialogue de Vernant pour saisir la structure en sous-buts propre d'un dialogue orienté par la réalisation d'une tâche. Le dialogue se présente à la fois comme une convergence globale et comme une série de convergences locales au niveau de chaque sous-but. De plus, le processus de convergence est un processus discontinu, c'est-à-dire, la participation d'un acte au processus de convergence n'est évaluée qu'une fois l'acte terminé.

La figure 2.13 montre les processus de convergence locales de deux sous-buts. Au début du premier sous-but l'écartement est maximal, puis le dialogue procède sans problème vers l'accord. Dans le cas du deuxième sous-but, il y une rupture, et donc une divergence, qui est réparée par l'instructeur pour, finalement, arriver à l'accord.

Dans ce processus de convergence, il est approprié de faire la distinction entre but *atteint* et but *satisfait*. Un but est considéré comme *satisfait* quand il a été l'objet d'un accord –cet accord pouvant être explicite ou implicite–. Si cet accord n'a pas été observé, le but continue à être pertinent pour l'interaction subséquente, même si le but est déjà *atteint*. Un but est considéré comme *atteint* lorsqu'il ne reste aucun acte pour contribuer à sa réalisation.

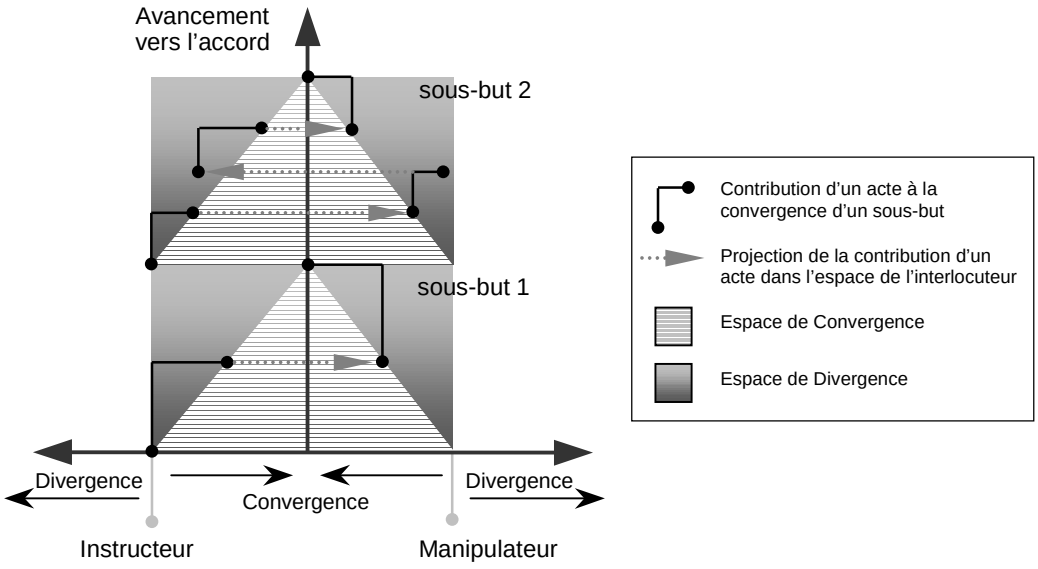


Figure 2.13. Modèle dynamique d'interaction (d'après [Ozkan 94])

2.6.4 La construction d'un objet de référence commun

Il existe d'importantes notions de la psychologie cognitive sur lesquelles le DHM s'appuie pour aborder le problème de la coopération homme-machine. Parmi elles, il existe une notion intéressante en relation avec notre travail : la construction de la *référence* au cours du dialogue. Les travaux de [Clark & Wilkes-Gibbs 86, Heeman & Hirst 95] montrent l'aspect coopératif impliqué dans la construction d'un objet de référence commun. Le modèle proposé par Clark et Wilkes-Gibbs prend en compte la différence entre le dialogue et l'écrit : les interlocuteurs ont un temps limité pour s'exprimer ; l'allocutaire doit attendre pour entendre et tenter de comprendre une phrase pendant qu'elle est articulée ; et les interlocuteurs échangent des phrases de façon alternée.

Le processus de construction d'une description d'un objet, afin de l'identifier parmi d'autres objets, est considéré comme la tentative d'identification du référent parmi les connaissances mutuelles des interlocuteurs. Il y a deux éléments principaux dans ce processus : la *présentation* –à travers une description de l'objet de référence par le locuteur– et *l'acceptation* –les deux interlocuteurs doivent accepter *mutuellement* que l'allocutaire ait compris la référence du locuteur avant la continuation du dialogue–. Les auteurs distinguent deux types d'acceptation possibles : l'acceptation *présupposée* quand l'allocutaire laisse passer le locuteur à la contribution suivante, et l'acceptation *confirmée* quand l'allocutaire répond avec des *continuateurs* (par exemple : *d'accord*, *oui*). Avec ces deux types d'acceptation les interlocuteurs reconnaissent mutuellement la dernière présentation comme identifiée.

Dans le cas où la contribution est rejetée, la description initiale est reformulée par l'un des deux interlocuteurs. Cette reformulation peut prendre la forme d'une *réparation* de la phrase, au moment de l'articulation, due à la détection d'un problème ; d'une *extension* de la description initiale ; ou encore, d'une *substitution* d'une partie de la description par d'autres qualificatifs.

Une autre expérience, tirée de la psychologie cognitive, qui a abordé le problème de la référentiation, est le projet *Compèrobot* [Andrès 95, Nicolle 96, Nicolle & Vivier 97, Vivier & Nicolle 97]. Dans cette expérience un

partenaire machine est utilisé pour inciter des enfants, dans une situation de dialogue finalisé, à produire des consignes adéquates même si cela suppose correction et reformulation jusqu'à obtenir la précision que requiert leur exécution. Les auteurs proposent un modèle qui tente de conserver à la fois un maximum de contrôle dans la construction du dialogue en train de se faire, et la construction de la référence commune.

Parmi les principales observations de cette expérience, les auteurs montrent que la principale difficulté d'un enfant est la description du référent. L'enfant –bien qu'il soit capable de déplacer les figures de carton pour construire le modèle demandé– doit construire un système de référence du monde dont il n'avait pas besoin auparavant. De plus, les difficultés de compréhension de son interlocuteur montrent à l'enfant la faiblesse de son système en cours de construction, et par là il prend conscience de l'existence du système de référence d'autrui et de la nécessité de négocier une co-référenciation. Dans une telle situation les auteurs font clairement référence à la nécessité d'adaptation et d'évolution des compétences de la machine, caractéristiques souhaitables, sinon indispensables, dans un système de dialogue homme-machine. A partir de ces besoins [Lehuen 97] a proposé un modèle de dialogue, qui, grâce à l'interaction, est capable d'acquérir des compétences linguistiques.

2.6.4.1 L'acquisition des compétences linguistiques

Les travaux de [Lehuen 97] tente de trouver une approche pour faire évoluer les compétences linguistiques de la machine. L'idée de base est l'élimination des limitations des interfaces de dialogue en langue naturelle qui proviennent de représentations trop normatives. C'est-à-dire des représentation qui forcent l'utilisateur à se focaliser sur le fonctionnement de la machine, au détriment de l'utilisation qu'il voudrait en faire. Ainsi, l'auteur tente de pas donner une compétence linguistique à l'avance, sinon de mettre en place des capacités pour acquérir cette compétence. La solution proposée est composée de trois modèles : un modèle de dialogue dynamique orienté vers la gestion des non-attendus ; un modèle d'interprétation des énoncés fondé sur l'usage et non sur une norme ; et un modèle d'apprentissage symbolique.

L'interprétation d'énoncés est réalisée d'abord, par un procédé dérivé du *pattern-matching* –en évitant toute norme linguistique–, puis un système de règles (qui prennent compte le contexte et le lexique) produit l'interprétation de l'énoncé. Dans le cas d'un échec de la communication, des nouvelles règles sont créées à partir de l'interprétation de l'énoncé. Ainsi, le système de dialogue s'adapte aux structures linguistiques employées par ses interlocuteurs à travers l'interaction.

2.6.5 Un regard comparatif

Avant de présenter notre approche, il faut rappeler certaines notions liées à notre problématique. Comme nous l'avons vu, notre objectif est la construction d'un système de dialogue coopératif finalisé homme-machine en situation de *conception*. Dans cette situation le système de dialogue joue le rôle d'intermédiaire auprès de l'utilisateur de manière à commander la machine. Dans ce cas le dialogue sert à *déterminer ce qu'il faut faire*. En outre, en situation de conception il est impossible de prévoir toutes les tâches à l'avance, puisque par définition la conception est une activité créatrice. Cependant, la conception passe souvent par une réutilisation de tâches, et, dans ces circonstances il y a un intérêt à apprendre la tâche. Dans ce cas le dialogue sert aussi à *déterminer comment il faut le faire*.

C'est par l'acquisition de nouvelles tâches que la machine sera en position d'assister l'utilisateur, de manière coopérative, dans une session ultérieure. La reconnaissance du but poursuivi a pour objectif la prise en charge de la réalisation de la tâche, avec bien sûr, l'accord de l'utilisateur. Ainsi le système devient-il l'acteur qui agit pour transformer le contexte.

Par ailleurs, ce dialogue qui sert à établir *quoi* faire et *comment* le faire, s'inscrit dans une interaction multimodale. Cette circonstance nous oblige à prendre en compte, de façon particulière, les effets d'un acte gestuel dans le dialogue.

D'après ces notions, les points importants à considérer sont :

- Notre modèle s'appuie sur la définition des actes de langage à partir des

modalités proposées par Ozkan et prend en compte les notions du modèle projectif de dialogue de Vernant. Comme dans toutes les approches basées sur les notions d'actes de langage –proposées initialement par Austin–, notre modèle s'inscrit dans une théorie générale de l'action. Cette conception permet de considérer l'acte de langage comme une forme d'action et de voir la recherche d'un but, à travers un dialogue, comme une succession d'actions dont certaines sont langagières et d'autres non.

- Un acte est, d'après la définition d'Ozkan et emprunté par nous, « l'unité minimale du discours ayant une visée ». Néanmoins, la fonction première d'un acte est communicationnelle. Ainsi un acte devient un *acte de dialogue* : une action sur autrui qui passe d'abord par la modification de ses états mentaux, de façon à faire connaître une conviction du locuteur à propos d'un fait du monde et(ou) à obtenir une réaction de l'allocutaire.

- Cet aspect communicationnel met en relation le locuteur et l'allocutaire comme coauteurs du dialogue. Ainsi les actes de dialogue modifient leurs états mentaux et s'enchaînent en respectant des présupposés de rationalité et de coopérativité. Cependant, dans le dialogue homme-machine, nous ne considérons pas la machine comme un coauteur du dialogue. Elle est simplement une assistante capable d'agir en adoptant les intentions de l'utilisateur qui sont transmises à travers le dialogue. Cette dissymétrie entre l'homme et la machine est capturée en différenciant les effets d'un acte de dialogue selon qu'ils proviennent de l'utilisateur ou du système informatique. Les actes de dialogue produits par l'utilisateur sont toujours porteurs d'intention à la différence des actes produits par la machine.

  C'est à travers les modalités des actes de dialogue et la distinction entre les deux possibles interlocuteurs d'un dialogue que notre approche aborde cet aspect communicationel.

- Par ailleurs, notre modèle de dialogue est basé sur le modèle projectif de Vernant : le dialogue est vu comme un processus de convergence vers l'accord. Un acte de dialogue de l'utilisateur fait connaître à la

machine son intention : le *but* poursuivi. C'est autour du but que les actes de dialogue s'enchaînent et le dialogue devient un processus de convergence qui cherche la *satisfaction* du but souhaité. Chaque acte de dialogue contribue à l'avancement vers cette satisfaction. Notre modèle reprend l'adaptation d'Ozkan pour structurer le dialogue en convergences locales correspondants à la résolution de sous-buts. Pour modéliser le dialogue à partir de ce processus de convergence notre formalisme distingue entre but *posé*, *atteint* et *satisfait* –ces notions sont adaptées du concept de satisfaction proposé par Vanderveken, étendu par Trognon et Brassac et repris et adapté par Ozkan–. Un but est *posé* comme effet d'un acte de dialogue qui exprime l'intention de l'utilisateur, c'est le premier pas dans le processus de convergence. Un but est *atteint* si l'action ou la séquence d'actions pertinente vis-à-vis de l'intention exprimée est accomplie. Le but est considéré comme *satisfait* quand l'utilisateur exprime un avis favorable sur l'adéquation de la réaction obtenue. Cet accord est l'état final du processus de convergence.

Cette vision du dialogue est plus proche des modèles structuraux du dialogue que ceux qui sont fondés sur la planification –ou de ceux basés sur l'interaction rationnelle, dans lesquels le traitement du dialogue *per se* est complètement absent, puisqu'ils assument le dialogue comme un résultat d'un comportement rationnel–. Néanmoins, l'approche de Vernant nous permet d'expliquer le dialogue en fonction d'un but qui n'est pas complètement spécifié, comme c'est le cas dans une situation de conception. Cette perspective nous permet aussi de lier un acte de dialogue à ses conséquences sur le monde en considérant, en même temps, leurs effets dialogiques.

- Ainsi, le formalisme développé explique-t-il le dialogue comme un enchaînement d'actes vers l'accomplissement d'un but. Le dialogue est une succession de changements d'états causés par chaque acte de dialogue. Ces états sont considérés à travers trois types de modalités : *dynamiques* (pour décrire les actions et les effets produits par ces actions) ; *épistémiques* (pour la représentation de connaissances) et *dialogiques* (pour exprimer les contributions de l'acte au processus de

convergence du dialogue en fonction du but recherché). Ce modèle ne vise pas à donner un comportement de type humain à la machine mais seulement à lui fournir des éléments pour réagir dans un dialogue finalisé. On ne cherche pas à donner des états mentaux à la machine mais à modéliser les raisons d'action de l'interlocuteur [Caelen 95]. Ainsi nous nous écartons de la ligne proposée par Sadek, qui cherche la construction d'un agent rationnel, pour ainsi arriver au dialogue coopératif. Même les modalités déontiques proposées par Ozkan sortent de notre cadre de dialogue homme-machine.

- A la différence des approches antérieures –à l'exception de travaux de Grosz–, notre approche est orientée vers la commande de la machine pour la réalisation d'une tâche et pas uniquement pour le transfert d'informations. Dans ce contexte, la machine est conçue comme une assistante qui apporte son savoir-faire à la réalisation du but exprimé par l'utilisateur. Pour obtenir cette réponse coopérative de la machine, un processus de reconnaissance de la tâche est intégré au modèle. Cependant, il faut aussi intégrer un processus d'acquisition des tâches puisqu'il n'est pas envisageable de prévoir d'avance toutes les tâches. Bien sûr, il faut incorporer ces deux processus à notre modèle du dialogue. Cependant, la conséquence principale de la reconnaissance d'un plan ne comprend pas seulement un effet dialogique –comme dans le cadre de travaux de Allen ou Litman–, elle comprend surtout un effet actionnel. La machine prendra l'initiative, si l'utilisateur le souhaite, et sera la responsable de l'achèvement de la tâche.

- La différence principale entre notre modèle et les autres est la prise en compte de la dimension de la mémorisation, en vue de l'apprentissage à partir de la répétition. Notre idée d'acquérir un savoir-faire à travers le dialogue pour faire progresser les compétences de la machine n'est pas discutée dans les autres modèles du dialogue : il semble que le dialogue prenne appui chaque fois sur une situation nouvelle. Aucun modèle de dialogue –à l'exception du travail de Lehuen [Lehuen 97] pour le cas de l'acquisition de compétences linguistiques– ne mentionne clairement l'apprentissage.

CHAPITRE III

Dialogue et Coopération

Ce chapitre présente des considérations relatives à la coopération dans un modèle de dialogue. L'idée générale est de favoriser l'interaction entre l'homme et la machine à travers le dialogue, mais aussi de profiter du dialogue pour adapter la machine à l'homme. L'apprentissage est essentiel pour améliorer la communication ; si la machine apprend à faire une tâche, à acquérir de nouveaux concepts et à agir de manière opportune, alors l'utilisateur pourra lui confier des travaux de production assistée. La machine sera capable de coopérer avec lui. En retour, l'apprentissage permet de guider le dialogue en utilisant une stratégie basée sur des intentions et qui est le fondement de toute stratégie coopérative. Pour nous, la coopération est la capacité de la machine d'adapter son comportement face à une situation déjà rencontrée, pour assister l'utilisateur. Ainsi, cette attitude coopérative de la machine est fondée sur deux processus principaux : l'acquisition des savoir-faire à travers des exemples, et la reconnaissance d'une situation à partir des actions observées.

Il existe plusieurs approches qui tentent de développer, de manière très diverse, cet aspect coopératif de l'interaction homme-machine. Par exemple, du point de vue de l'intelligence artificielle, il y a différentes propositions pour acquérir et traiter la connaissance résultant de l'expérience. Traditionnellement le problème a été abordé selon trois aspects –bien que cette division ne soit pas aussi nette–, (i) l'acquisition de connaissances à travers des exemples [Michalski 83, Bratko et al. 91, Quinlan 93, Carbonell 86] ; (ii) la reconnaissance d'une situation à travers des observations [Kautz 87, Song & Cohen 96, Goodman & Litman 92] ; et, (iii) la construction d'une solution adaptée à une situation analogue aux circonstances déjà rencontrées [Leake 96, Mille & Napoli 97, Veloso & Carbonell 93]. Dans tous ces cas, le résultat attendu est l'adaptation de la machine à un nouvel environnement ou à un nouveau problème. Différentes tentatives prennent l'interaction comme base de l'acquisition de connaissances pour atteindre cette adaptation [Laird et al. 86, Rosenbloom & Laird 86, Sabah 97, Lehuen 97].

Par ailleurs, du point de vue des interfaces adaptatives homme-machine, où l'idée est de rendre l'ordinateur plus simple à utiliser, il y a des propositions qui tentent la construction d'un modèle de l'utilisateur –à partir de données recueillies au cours de l'utilisation du système– pour reconnaître, prédire et assister la résolution du problème en cours [Maybury 93]. Dans ce cas-là, l'apprentissage se fait de manière *incrémentale*, c'est-à-dire qu'à chaque fois qu'il existe une interaction avec l'interface, la base de connaissances est mise à jour. Cela se fait à partir d'un petit nombre d'exemples, puisqu'une interface adaptative doit apprendre rapidement. Il existe deux grandes classes d'interfaces adaptatives [Langley 97] : les interfaces *informatives* et les interfaces *génératives*. Les premières sont des interfaces qui tentent de filtrer ou sélectionner l'information présentée à l'utilisateur (par exemple, [Lang 95, Pazzani et al. 96]). Par contre, les interfaces génératives interviennent dans la construction d'une structure de connaissance, par exemple dans des logiciels de création de documents ou de dessin, tableurs électroniques, etc. (par exemple, [Dent et al. 92, Motoda & Yoshida 97, Cypher 91, Karsenty & Pachet 95]).

Notre travail utilise différentes notions issues de l'intelligence artificielle. Le système que nous visons peut être classé parmi les interfaces adaptatives génératives. Les sections suivantes présentent les aspects fondamentaux pour la prise en compte de la coopération dans notre modèle de dialogue. Une première partie introduit la représentation du *plan* de la tâche et discute son apprentissage ; une deuxième partie examine la méthode adoptée pour la reconnaissance de tâches.

3.1 Quelques considérations à propos du dialogue coopératif

Comme nous l'avons vu auparavant, nous sommes intéressés par la modélisation du dialogue dans les situations de conception, pour lesquelles il n'est pas possible de prévoir toutes les tâches à l'avance, puisque par définition la conception est une activité créatrice. Cependant on s'aperçoit – chez les architectes par exemple– que l'acte de conception passe souvent par une réutilisation de plans architecturaux anciens, simplement réorganisés ou réagencés [Deshayes 98, Lebahar 98]. Dans ces conditions on a intérêt à apprendre un *plan* (pour faire une tâche ou résoudre un problème) en le construisant pendant le déroulement du dialogue à partir des buts poursuivis par l'architecte. L'apprentissage du plan permettra ultérieurement de l'assister de manière coopérative ou même d'anticiper de manière constructive.

Les problèmes soulevés à propos de l'apprentissage de la tâche sont :

- l'acquisition d'un nouveau plan d'action, pertinent pour la tâche en question,
- sa caractérisation à partir d'exemples,
- l'assignation d'un but dans le contexte du dialogue.

Puis, dans un deuxième temps, pour l'assistance à l'utilisateur, les problèmes sont :

- la reconnaissance du but poursuivi à partir des actions observées, pour inférer la logique d'action générale de l'utilisateur et identifier son plan d'action,
- l'activation pertinente des stratégies d'assistance à l'utilisateur dans le cours du dialogue.

L'acquisition du plan –de la même façon que l'assistance à l'utilisateur– est réalisée en utilisant des sous-dialogues adaptés [Caelen 95]. Par exemple, dans le cadre d'une application de dessin, le sous-dialogue suivant est utilisée pour l'acquisition d'une tâche :

U : Dessine une maison
M : Qu'est-ce qu'une maison ? Montre-moi… en la dessinant…
U : < *série d'actions pour faire le dessin* >
U : Voilà une maison
M : d'accord
etc.

Dans cet exemple, les actions sont toutes référencées au même but « dessiner(maison) ». La confirmation permet de s'assurer que le but est bien satisfait.

Au départ, on suppose que l'on dispose d'actions élémentaires bien définies. Puis, après que l'utilisateur a posé un but, il s'agit pour la machine d'observer, d'ordonner et d'associer cette séquence d'actions au but posé. Nous sommes donc ici dans un contexte *maître-apprenant* où le maître est l'utilisateur et l'apprenant est la machine :

- pendant la phase d'observation la machine enregistre la série d'actions montrées par l'utilisateur ;
- à partir de l'enregistrement, la machine tente de distinguer des relations entre les actions, par exemple, la succession dans le temps ou la causalité ; pour établir un ordre, probablement partiel, entre les actions ;
- dans la phase d'association, on attache le plan obtenu au but poursuivi par l'utilisateur.

A la fin de cette étape, la machine est capable de recommencer la tâche à partir d'une demande de l'utilisateur. Mais l'apprentissage ne s'arrête pas là. La nouvelle tâche apprise doit être *généralisée* et mise en rapport avec les autres tâches connues. Cette généralisation permet la formation d'une *abstraction* liée au but posé par l'utilisateur. De cette façon, la connaissance acquise évolue pour se rapprocher du concept réel détenu par l'utilisateur.

3.2 La représentation du plan

Une difficulté majeure est la manière de représenter les connaissances pour permettre cet apprentissage, car on fait l'hypothèse que les connaissances acquises par la machine sont incomplètes et dynamiques et qu'elles doivent rendre explicites les différents rapports entre les actions. La modélisation d'un monde qu'introduit le concept d'action exige la prise en compte de connaissances temporelles. Diverses approches ont été proposées pour modeler des notions temporelles comme l'antériorité, le recouvrement, la simultanéité entre actions, leur causalité, etc. Différents travaux en *raisonnement temporel* ont été proposés, parmi lesquels : le calcul situationel de [McCarthy & Hayes 69], la logique temporelle de [McDermott 82], le calcul d'intervalles de [Allen 84] et la logique d'intervalles de [Shoham 87].

Les idées ci-après sont basées sur le travail de Shoham qui définit d'une manière générale une proposition temporelle $\langle i, p \rangle$ laquelle associe une assertion logique p à un intervalle de temps i. La syntaxe et la sémantique de cette logique sont définies dans [Shoham 87] et [Dean & Wellman 90].

Pour parler de changements du monde, on utilise la notion d'*événement* pour indiquer la transition d'une proposition d'un état à un autre. Considérons deux propositions temporelles $\langle (t_1, t_2), p \rangle$ et $\langle (t_4, t_5), \neg p \rangle$ telles que t_2 précède t_4 ($t_2 \leq t_4$) ; la transition de p à $\neg p$ qui se produit dans un instant t_3 compris entre t_2 et t_4 est déterminée par un *événement*. En général, une proposition temporelle $\langle i, p \rangle$ s'écrit ***HOLDS(i, p)*** où l'intervalle i est le couple $t_a \backslash t_b$, où $t_a \leq t_b$. Pour souligner le rôle d'événement d'une expression temporelle nous l'écrirons ***OCCURS(i, p)***. L'intervalle i où l'événement se produit est l'instant $t_a \backslash t_b$, où $t_a = t_b$[3]. Par exemple :

$$\text{OCCURS}(t_1, \text{dessiner}(\text{triangle}, x_1, y_1)) \wedge \text{OCCURS}(t_2, \text{déplacer}(\text{triangle}, x_2)) \wedge \text{OCCURS}(t_3, \text{dessiner}(\text{carré}, x_3, y_2)) \wedge \text{OCCURS}(t_4, \text{déplacer}(\text{carré}, x_4))$$

décrit la séquence d'actions que l'utilisateur a effectuée pour dessiner une maison et dans laquelle ***dessiner(triangle, x, y)*** représente l'action de dessiner un triangle à la position x (où x est un vecteur) et de taille y. Chaque

[3] [Shoham 87] présente une catégorisation des propositions temporelles. Par rapport à celle-ci, une proposition ***HOLDS(i, p)*** est équivalente à une expression du type *liquide* et ***OCCURS(i, p)*** à une expression du type *gestalt*.

événement correspond à une action de base. Chaque action de base est définie par une règle *à la STRIPS* qui permet de récupérer l'ensemble des propositions temporelles affectées par l'exécution de l'action. Par exemple, la règle simplifiée suivante établit les conditions et les effets pour l'action **déplacer** :

$$\text{HOLDS}(\ t_0\backslash t_1,\ \text{position}(\ \text{carré},\ x_1\ ))\ \wedge$$
$$\text{OCCURS}(\ t_1,\ \text{déplacer}(\ \text{carré},\ x_2\ ))\ \supset\ \text{HOLDS}(\ t_1\backslash t_2,\ \text{position}(\ \text{carré},\ x_2\ ))$$

A partir de ces règles, on introduit une relation de précédence temporelle entre les propositions. Par exemple, après la séquence d'actions pour dessiner la maison, on récupère les propositions suivantes :

$$\text{HOLDS}(\ t_0\backslash t_1,\ \neg\text{dessiné}(\ \text{triangle}\ ))\ \wedge$$
$$\text{HOLDS}(\ t_1\backslash t_a,\ \text{dessiné}(\ \text{triangle}\ ))\ \wedge$$
$$\text{HOLDS}(\ t_1\backslash t_2,\ \text{position}(\ \text{triangle},\ x_1\ ))\ \wedge$$
$$\text{HOLDS}(\ t_2\backslash t_b,\ \text{position}(\ \text{triangle},\ x_2\ ))\ \wedge$$

$$\text{HOLDS}(\ t_0\backslash t_3,\ \neg\text{dessiné}(\ \text{carré}\ ))\ \wedge$$
$$\text{HOLDS}(\ t_3\backslash t_c,\ \text{dessiné}(\ \text{carré}\ ))\ \wedge$$
$$\text{HOLDS}(\ t_3\backslash t_4,\ \text{position}(\ \text{carré},\ x_3\ ))\ \wedge$$
$$\text{HOLDS}(\ t_4\backslash t_d,\ \text{position}(\ \text{carré},\ x_4\ ))$$

La figure 3.1, une adaptation d'un *time map* [Dean & McDermott 87], montre les propositions et leur précédence temporelle. Les lignes noires droites représentent les intervalles de temps et les lignes pointillées courbes la précédence temporelle. Cette relation entre les propositions est déduite d'une part, par les règles associées à chaque action et d'autre part, par les durées de réalisation de la tâche. Le premier cas est une relation forte qui oblige une précédence temporelle *nécessaire*. On n'est pas capable de changer l'ordre d'exécution (avant de déplacer l'objet, il doit exister). Le deuxième cas entraîne une relation temporelle *fortuite* i. e. $t_2 \leq t_3$ (le dessin du triangle avant le carré ne change rien au résultat final). Ces deux types de relation permettent de mieux appréhender la combinaison temporelle des actions, et donc de mieux faire ressortir les parallélismes possibles et les séquencements obligatoires. De cette façon, on peut donc obtenir un plan, non pas comme un ensemble de séquences d'actions, mais comme un ensemble d'actions partiellement contraintes dans le temps les unes par rapport aux autres et par rapport à certains faits.

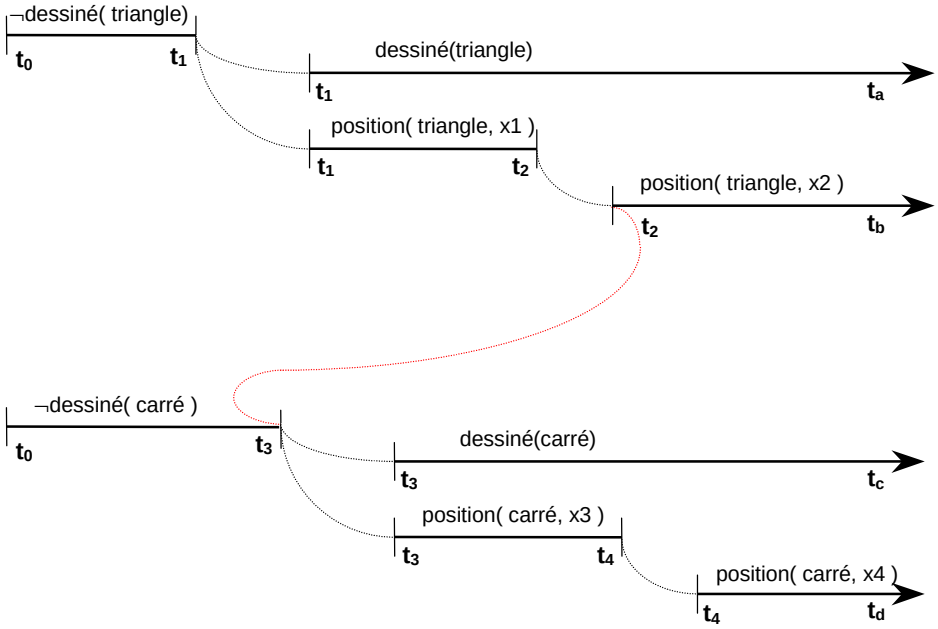


Figure 3.1. Les propositions et leur précédence temporelle.

La distinction entre ces deux types de relation permet de relier une série d'actions qui sont dépendantes entre elles. Une action dépend d'une autre si ses effets sont des conditions pour l'exécution de l'autre, c'est-à-dire, s'il existe une relation temporelle nécessaire entre elles. On considère cette série d'actions comme une seule action. Cette action *composée* est liée aux autres actions, probablement composées aussi, à travers une relation temporelle fortuite. Après avoir distingué les actions composées, on peut appliquer des *opérations de réduction* dans chacune pour éliminer les propositions redondantes et simplifier le plan.

Dans l'exemple antérieur, on peut réduire l'action composée pour dessiner le carré au schéma suivant :

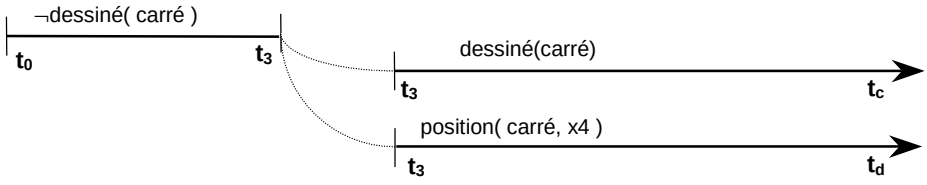


Figure 3.2. Action composée après sa réduction.

cette réduction entraîne l'élimination de l'action ***déplacer*** et la modification de la création du carré directement à la position x_4.

Une autre opération importante, que l'on peut tenter après la séparation des actions composées est la recherche de relations spatiales entre les objets sur lesquels elles sont centrées. L'établissement des relations spatiales peut être réalisé à travers les déductions réalisées par la machine (cf. § 3.4.3) et/ou à travers l'aide directe de l'utilisateur [Li 94].

3.3 L'objet comme résultat des actions

À travers ce formalisme pour la représentation des connaissances, on est donc capable de décrire la séquence d'actions pour faire la tâche, mais aussi de produire des objets comme résultats de ces actions. Le but de l'utilisateur est vu comme l'ensemble d'effets produits par la réalisation des actions. But et objet sont donc deux notions complètement liées. Pour décrire les objets, il faut focaliser notre attention sur la séquence d'actions observées et sur le résultat de ces actions, dans le contexte du but à atteindre. De cette manière, les objets « table » et « chaise » sont différents non par leurs formes mais par les actions réalisées.

Le processus d'abstraction ainsi que la reconnaissance des intentions prennent aussi l'action comme élément de base. L'utilisateur peut montrer deux séquences différentes pour faire la même tâche, mais le processus d'assimilation nous permet, à partir des actions réalisées, de former une abstraction plus générale. Ainsi, la connaissance de la machine évolue pour se rapprocher du concept exprimé par l'utilisateur. De la même façon, le processus de reconnaissance des intentions prend l'action comme élément de base. C'est à partir des actions observées que la machine tente d'inférer le but de l'utilisateur.

3.4 L'abstraction de la tâche

Après avoir acquis le plan pour faire une tâche, une autre considération importante est la généralisation de celle-ci par rapport aux autres exemples de la même tâche. Le résultat est une *abstraction* du plan pour atteindre le but poursuivi par l'utilisateur. Au cours de plusieurs sessions de travail, la

connaissance de la machine évolue pour se rapprocher du *concept* exemplifié par l'utilisateur. Cette étape est le perfectionnement des connaissances, l'amélioration des performances au cours de l'expérience. On part de l'hypothèse que les connaissances incorporées par le système sont incomplètes et qu'elles évoluent à chaque fois que l'utilisateur montre un nouvel exemple d'un même concept.

Il y a différents paradigmes pour l'apprentissage automatique [Carbonell 89]. Les quatre paradigmes d'apprentissage principaux sont : l'inductif, l'analytique, le génétique et le connexionniste. Bien que ces paradigmes aient le même objectif, ils apprennent de façon très différente. Toutefois, tous les quatre voient le processus d'apprentissage comme une modification de la connaissance courante de l'apprenant par interaction avec une source d'information extérieure. Dans notre cas, où l'utilisateur tente de montrer son savoir-faire à la machine, et où la machine et l'utilisateur doivent partager des ressources et faire des raisonnements ensemble, les connaissances doivent être codifiées de manière compréhensible par tous les deux [Carbonell et al. 83]. Les paradigmes « génétique » et « connexionniste » ne représentent pas leurs connaissances de façon simple à interpréter par l'utilisateur. Il semble donc qu'il soit plus difficile de les utiliser ici.

Pour la formation d'un concept, notre travail s'appuie sur le paradigme inductif en utilisant des stratégies « d'apprentissage de concepts par l'exemple ». L'idée principale est l'adaptation de la description d'une tâche chaque fois que l'utilisateur donne un exemple de cette tâche. Dès que l'on apprend une nouvelle tache, le concept créé est la première approximation de la description de celle-ci. Lorsque l'utilisateur présente un nouvel exemple de la tâche, un processus de *généralisation* adapte l'approximation actuelle du concept pour inclure les caractéristiques propres de l'exemple montré. Le résultat de la généralisation, entre l'approximation courante et l'exemple actuel, est une nouvelle approximation plus complète. Cependant, dès la première approximation, bien qu'inexacte et/ou incomplète, on peut l'utiliser pour reproduire la tâche.

Dans notre cas, l'apprentissage est caractérisé par :

a) La formation d'un concept, il est (i) empirique –au départ, on ne connaît rien sur la tâche–, (ii) il procède de façon incrémentale –la description d'une tâche est ajustée à chaque occurrence d'un nouvel exemple de cette tâche, et (iii) il se réalise uniquement à partir d'exemples positifs.

b) Les tâches sont classifiées avec l'aide et sous le contrôle de l'utilisateur. L'utilisateur guide la formation de nouveaux concepts, ainsi que l'affinage de concepts déjà appris.

c) La description d'une tâche doit permettre sa reconnaissance ; dans une situation ultérieure, on tente d'inférer la tâche à partir des actions observées.

3.4.1 La formation des concepts

Il y a différentes méthodes pour réaliser des généralisations. Des opérations comme *la restriction de conjonction* ou *le remplacement de constantes par des variables* sont utilisées pour dériver des généralisations [Michalski 83]. Les processus de généralisation ont été amplement explorés, Plotkin ayant été le premier à analyser rigoureusement cette notion. La *généralisation la plus spécifique* (basée sur la *θ-subsumption*) est la première solution proposée par Plotkin [Bergadano & Gunetti 96]. L'idée est de trouver entre deux termes la généralisation minimale possible, c'est-à-dire une expression qui conserve le maximum de caractéristiques communes aux expressions de départ.

La généralisation la plus spécifique de deux termes t_1 et t_2, dénotée $g(t_1, t_2)$, est définie de la manière suivante :

- $g(t_1, t_2)$ est une généralisation de t_1 et t_2 s'il existe deux substitutions σ_1 et σ_2 telles que :

  $$g(t_1, t_2).\ \sigma_1 = t_1 \text{ et } g(t_1, t_2).\ \sigma_2 = t_2$$

- ***g(t₁, t₂)*** est plus spécifique que toute autre généralisation selon l'ordre partiel sur les termes :

  t ≥ t', ***t*** est plus spécifique que ***t'***, si et seulement s'il existe une substitution σ telle que ***t'.σ = t***

Pour faire le calcul de la généralisation la plus spécifique de deux termes t_1 et t_2, on a :

- si $t_1 = t_2$, alors $g(t_1, t_2) = t_1$

- si t_1 est $f(s_1,s_2 \ldots s_n)$ et t_2 est $f(e_1,e_2 \ldots e_n)$
  alors
  $g(t_1, t_2) = f(g(s_1, e_1), g(s_2, e_2) \ldots g(s_n, e_n))$

- si t_1 et t_2 sont différents (c'est-à-dire des fonctions ou des variables différentes, ou bien l'un est une fonction et l'autre une variable) alors $g(t_1, t_2)$ est une variable ***v***, où la même variable est partout utilisée comme la généralisation de ces deux termes.

Par exemple, la généralisation la plus spécifique de $t_1 = dessiner(obj_1, x_1, y_1)$ et $t_2 = dessiner(obj_2, x_2, y_1)$ est $g(t1, t2) = dessiner(X, Y, y_1)$.

Dans le cas le plus simple, la généralisation la plus spécifique est exactement l'opération duale de l'unification. Le résultat du processus de généralisation est une hiérarchie qui exprime la relation de généralisation/spécialisation. Pour l'exemple ci-dessus, on a la hiérarchie suivante :

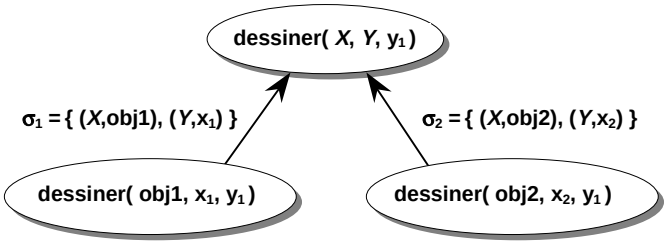


Figure 3.3. Relation de généralisation/spécialisation.

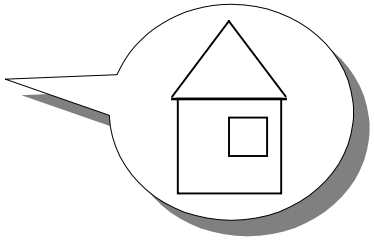
Les substitutions σ_1, σ_2 nous permettent de revenir aux exemples du départ. Cette spécialisation du concept général est nécessaire pour reproduire la tâche.

A partir de ces idées, on obtient une approximation du concept comme un ensemble d'actions qui décrivent la tâche. Cependant, il reste à caractériser le concept, c'est-à-dire, à trouver des associations parmi les éléments descriptifs du concept. La caractérisation d'un concept entraîne donc la recherche des relations inhérentes entre ses éléments, à partir des valeurs observées dans les exemples [Bratko et al. 91, Muggleton 95].

Par exemple, l'utilisateur dessine une maison, ce premier dessin devient la première approximation du concept « dessiner-maison » :

dessiner-maison$_1$ (avec une fenêtre)

OCCURS(t_1, dessiner(obj1, triangle, x_1, y_1)) ∧
OCCURS(t_2, dessiner(obj2, rectangle, x_2, y_2)) ∧
OCCURS(t_3, déplacer(obj2, x_3)) ∧
OCCURS(t_4, dessiner(obj3, carré, x_4, y_3)) ∧
OCCURS(t_5, changer_taille(obj3, y_4)) ∧
OCCURS(t_6, déplacer(obj3, x_5))



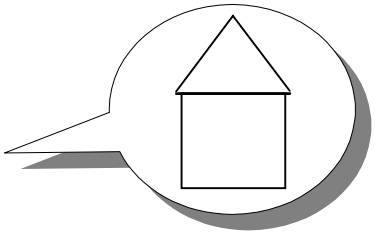
après la phase de réduction on obtient :

OCCURS(t_1, dessiner(obj1, triangle, x_1, y_1)) ∧
OCCURS(t_2, dessiner(obj2, rectangle, x_3, y_2)) ∧
OCCURS(t_4, dessiner(obj3, carré, x_5, y_4))

Supposons que l'utilisateur montre un deuxième dessin de « maison » à travers la séquence suivante :

dessiner-maison$_2$ (sans fenêtre)

OCCURS(t_1, dessiner(obj1, triangle, x_4, y_1)) ∧
OCCURS(t_2, dessiner(obj2, rectangle, x_3, y_2))



Le résultat de la généralisation de ce deuxième exemple par rapport à la première approximation est le suivant :

dessiner-maison - généralisation {dessiner-maison1, dessiner-maison2}

OCCURS(t_1, dessiner(obj1, triangle, Xt, Yt)) ∧
OCCURS(t_2, dessiner(obj2, rectangle, Xr, Yr)) ∧
HOLDS($t_2 \backslash t_b$, au-dessus(obj1, obj2))

La dernière proposition ***HOLDS(t_2\t_b, au-dessous(obj2, obj1))*** capture comme caractéristique inhérente au concept *maison,* qui est satisfaite par les deux exemples, le fait que le triangle est au-dessus du carré.

Ainsi la formation d'un concept entraîne deux pas : (i) la découverte des actions *caractéristiques,* c'est-à-dire les actions les plus représentatives de la tâche ; (ii) la recherche des relations *descriptives* qui sont respectées par tous les exemples de la tâche. Ces relations sont définies en utilisant des connaissances d'arrière-plan. Par exemple, des relations spatiales comme « au dessous de » ou « à droite de » sont utilisées pour décrire le concept appris. En conséquence, il faut étendre l'idée de la généralisation plus spécifique pour la prise en compte de ce type des connaissances. Une possibilité est la *subsumption généralisée* [Buntime 88] ou les idées proposées par la programmation logique inductive. Les paragraphes suivants présentent ces idées et leur application à notre problème.

3.4.2 La programmation logique inductive

La programmation logique inductive (PLI) est l'intersection entre l'apprentissage inductif de concepts et la programmation logique [Bergadano & Gunetti 96]. L'idée est d'appliquer les notions du paradigme d'apprentissage de concepts dans un cadre de programmation logique. Ainsi, dans la PLI le terme d'*apprentissage de prédicats* est alors plus adéquat.

Les systèmes d'apprentissage à partir des arbres de décision comme ID3 [Quinlan 86] ou C4.5 [Quinlan 93] sont des exemples de systèmes d'apprentissage en logique propositionnelle. Il existe deux limitations importantes dans tel type de systèmes : une expressivité restreinte par le formalisme de représentation de données –la logique propositionnelle– et l'impossibilité de prendre en considération des connaissances d'arrière-plan. L'utilisation de la programmation logique nous permet l'utilisation d'une logique de premier ordre, pour augmenter l'expressivité, et l'emploi des connaissances d'arrière-plan pour améliorer la qualité des résultats obtenus. Cependant, il y a, comme toujours, un compromis par rapport à la performance. Les premiers systèmes de la PLI avaient comme problème principal l'explosion combinatoire due surtout à la façon de faire la recherche

dans l'espace d'hypothèses. Depuis, d'autres types de systèmes ont été développés (par exemple, Clint [De Raedt 92] ou GOLEM [Muggleton 95]) qui mettent en place des méthodes de recherche plus pertinentes.

Une de ces techniques est basée dans les idées de Plotkin : la *généralisation relative la plus spécifique*. Cette méthode, basée sur l'idée de la généralisation plus spécifique discutée dans des paragraphes précédents, réalise une généralisation entre deux exemples positifs, en prenant en compte des connaissances d'arrière-plan. Néanmoins, ce type de généralisation peut seulement être appliqué si la connaissance d'arrière-plan est exprimée comme une conjonction de clauses atomiques sans variables libres.

Une autre méthode proposée par Muggleton [Muggleton 95] est l'*implication inverse*. Supposons que *B* est la connaissance d'arrière-plan, *E* un exemple positif et *H* une hypothèse. Alors, *H* doit satisfaire la condition

$$B \wedge H \models E$$

puisque *B* toute seule n'explique pas *E*. Cette condition peut être exprimée comme

$$B \wedge \neg E \models \neg H$$

laquelle est nommée implication inverse. D'après cette formule, Muggleton développe un algorithme très efficace pour calculer la meilleure hypothèse à partir d'un ensemble d'exemples positifs et des connaissances d'arrière-plan données. Cet algorithme est la partie centrale du système **Progol** [Muggleton 95]. Muggleton a démontré[4] qu'il est possible de calculer cette hypothèse, nommée *hypothèse plus spécifique* H_{PS}, si l'hypothèse et les exemples sont restreints à une seule clause. Puisque $\neg H$ doit être vraie en chaque modèle de $B \wedge \neg E$, elle doit contenir un sous-ensemble des littéraux atomiques de la H_{PS}. Alors,

$$B \wedge \neg E \models H_{PS} \models \neg H$$

[4] Voir aussi la discussion de [Furukawa et al. 97].

et pour toutes les hypothèses

$$H \models H_{PS}$$

Un ensemble de solutions pour *H* peut être trouvé en considérant les clauses que *θ-subsume* $H_{PS.}$

3.4.3 La caractérisation du concept

Comme nous l'avons vu, la formation d'un concept est réalisée en deux pas. La première phase est la découverte des actions *caractéristiques* du concept, qui sont le résultat d'une mise en relation entre les actions de deux exemples du même concept. Cette opération de sélection est un processus de généralisation (basé sur la méthode de la « généralisation la plus spécifique ») entre les différentes actions de deux exemples, guidé par une heuristique qui tente de trouver les généralisations *minimales*, c'est-à-dire les généralisations avec un minimum de perte d'information.

La deuxième phase tente d'établir toutes les relations possibles qui décrivent le concept, c'est-à-dire les relations existant entre les actions caractéristiques d'un même exemple, qui sont respectées par toutes les exemples du concept visé. Cet ensemble de relations *descriptives* est obtenu par le calcul de « l'hypothèse plus spécifique » avec comme connaissance d'arrière-plan les relations spatiales. Ainsi la description résultante est constituée de la généralisation des actions *caractéristiques* et d'un ensemble de relations *descriptives* entre ces actions.

La figure suivante montre le processus de formation d'un concept à partir des exemples précédents pour la tâche *dessiner-maison*,

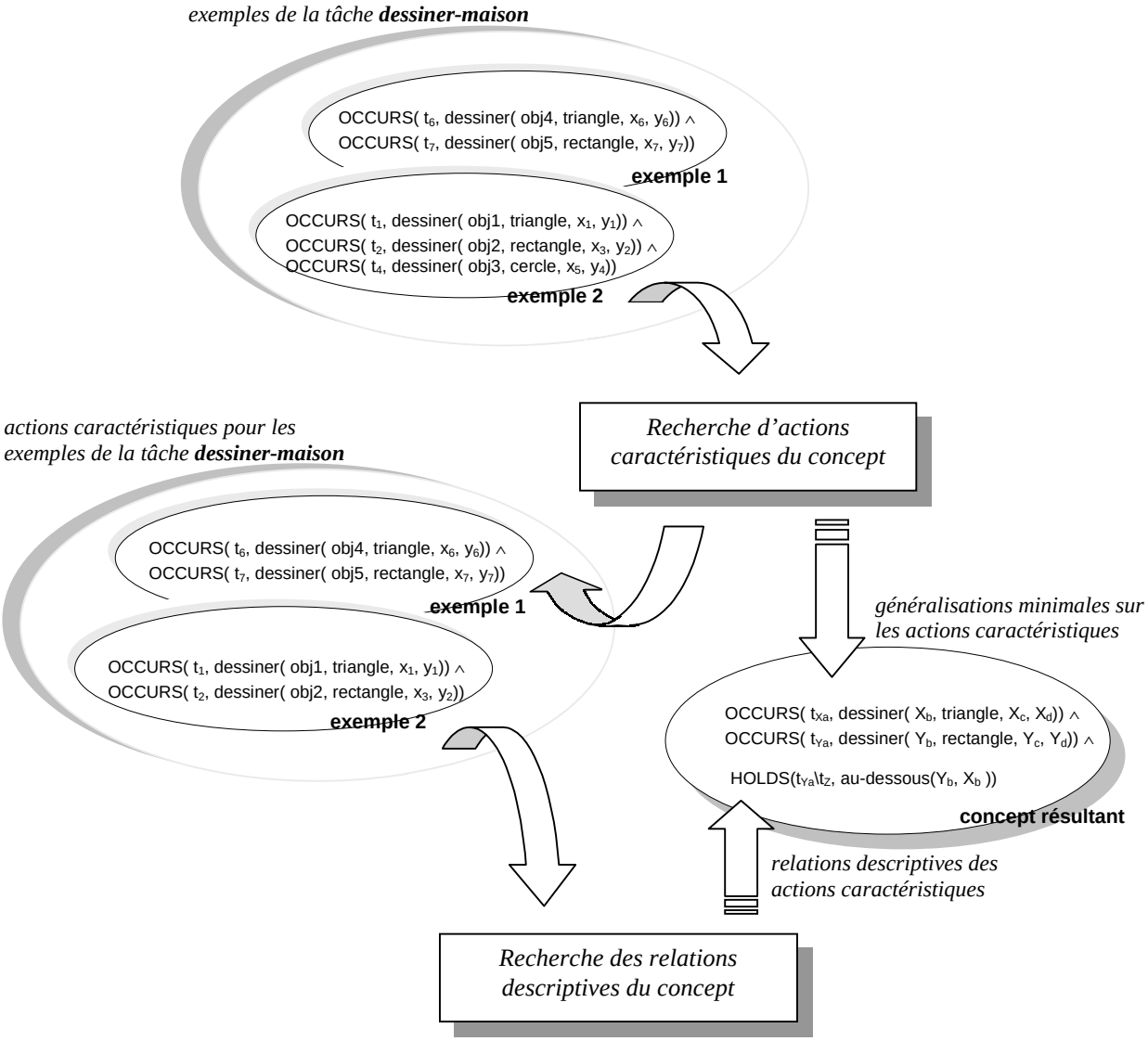


Figure 3.4. Processus de formation d'un concept.

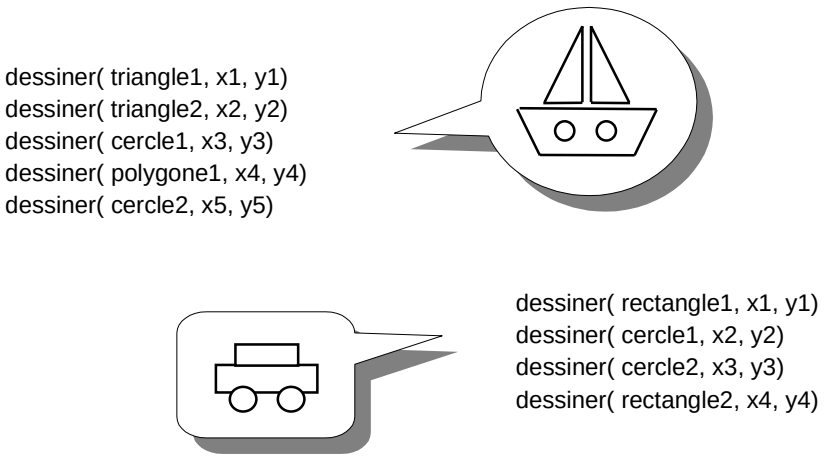
3.5 La reconnaissance de la tâche

La reconnaissance de la tâche peut être vue comme le problème inverse de l'apprentissage : pendant le déroulement du dialogue, les notions résultant de l'apprentissage dirigent la reconnaissance du but poursuivi par

l'utilisateur, c'est-à-dire qu'à partir des concepts appris, on tente de retrouver l'intention de l'utilisateur. Ainsi, la machine sera capable de coopérer lorsqu'une situation, analogue à une autre déjà vécue, est rencontrée. Dans ce cas-là, après la confirmation de l'utilisateur et éventuellement un sous-dialogue de particularisation de la tâche, le plan pourra être exécuté. De cette manière, en s'appuyant sur une stratégie dirigée par les intentions, la machine établit un dialogue qui converge vers la réalisation de la tâche.

La recherche de la tâche utilise un processus de raisonnement pour déduire le but de l'utilisateur à travers la succession d'actions observées. Comme nous l'avons évoqué, l'action est l'élément principal. Les dessins d'une table et d'une chaise sont différents par les actions réalisées, non par leurs formes perçues. Pour reconnaître le but de l'utilisateur il faut enregistrer et transformer les événements observés, et ensuite vérifier s'ils font partie d'un concept connu.

Le système de reconnaissance tente donc d'établir un lien entre une séquence répertoriée et la séquence d'actions observées. Cette opération est réalisée de manière incrémentale. Au fur et à mesure de l'occurrence des événements une description *récente* est construite. Lorsque cette description récente est appariée avec un sous-ensemble d'actions d'un concept connu, la machine peut donc continuer la tâche après la validation de l'utilisateur. Dans le cas d'ambiguïté, c'est-à-dire, quand le processus de reconnaissance obtient plus d'une hypothèse, il faut attendre une autre observation pour compléter la description récente jusqu'à tomber sur une seule hypothèse. Par exemple, on suppose deux concepts connus : *dessiner-bateau* et *dessiner-voiture*



Alors, si la description récente présente deux cercles, le processus de reconnaissance obtient deux hypothèses. On attend un nouvel événement pour faire disparaître l'ambiguïté, par exemple un rectangle. Au moment où l'utilisateur dessine un rectangle –et les relations descriptives sont respectées– la machine est dans les conditions de déclencher un sous-dialogue pour faire la tâche : elle peut donc anticiper sur le dessin d'une voiture.

Ainsi notre processus de reconnaissance est défini en deux pas : (i) la construction de la description *récente*, c'est-à-dire la description construite à partir des actions observées, en éliminant toute action intermédiaire redondante ; et, (ii) la reconnaissance proprement dite, qui cherche à établir le plan développé par l'utilisateur à partir de la description récente. Enfin, une fois qu'il existe une hypothèse valide, la machine sera en position d'assister l'utilisateur par un sous-dialogue de coopération. Les prochains paragraphes discutent brièvement le problème de la reconnaissance de plans et présentent la méthode adoptée ainsi que son application dans notre contexte.

3.5.1 La reconnaissance de plans

La reconnaissance d'un plan s'attache à découvrir le but recherché par l'utilisateur : partant d'une description incomplète des actions observées effectuées par l'utilisateur, un processus de reconnaissance retrouve le plan qui sous-tend et relie ces actions. Cette reconnaissance permet de prévoir les actions futures du sujet ou les éventuels obstacles, ou, comme dans notre cas, de donner à la machine les éléments suffisants pour compléter la tâche souhaitée.

Les premières méthodes pour la reconnaissance de plans ont été centrées sur l'explication de certains phénomènes dans le traitement du langage naturel –notamment dans des systèmes de recherche d'information–, comme le traitement d'actes de langage indirects [Allen & Perrault 80] ou la génération de réponses pertinentes [Litman & Allen 87]. Ensuite, la reconnaissance de plans a également été utilisée pour aider l'utilisateur inexpérimenté (par exemple [Wilensky et al. 88]) ou dans des systèmes d'enseignement. Néanmoins, c'est à partir du travail de Kautz [Kautz 87] qu'une méthode formelle a été développée –ensuite, d'autres approches ont été introduites

principalement pour tenter d'éliminer les limitations imposées par la méthode de Kautz [Goodman & Litman 92, Song & Cohen 96]–. Les sections suivantes présentent le formalisme proposé par Kautz et discutent son application dans notre travail.

3.5.2 Le travail de Kautz

Kautz [Kautz 87] présente une méthode pour la reconnaissance de plans et fournit un fondement théorique pour l'inférence de plans indépendant du domaine d'application considérée. L'auteur introduit trois limites à la portée de son travail : (i) la méthode ne peut reconnaître que les plans qui sont répertoriés dans une bibliothèque de plans ; (ii) elle prend en compte des connaissances communes à l'observateur et l'agent et non les intentions et croyances de chacun de façon individuelle ; (iii) pour la planification, la méthode s'appuie sur le concept d'*événement*. Le travail repose sur deux suppositions importantes. En premier lieu, la bibliothèque de plans est considérée comme complète : toutes les réalisations d'un plan sont spécifiées dans la bibliothèque. En second lieu, toutes les actions observées ont une utilité puisqu'elles peuvent être expliquées par leur appartenance à un plan.

Cette théorie est intéressante pour deux raisons : d'une part elle fournit un fondement théorique à l'inférence de plan, de manière générale et sans être liée à un domaine particulier, d'autre part, la reconnaissance s'étend sur une suite d'observations, dont chacune permet d'affiner la conclusion tirée à partir des précédentes.

Cette théorie s'applique à une base de connaissances (spécifiques au domaine considéré) qui décrit les plans, les actions et les relations logiques entre les actions. A partir de cette base de connaissances, il est possible de déduire d'une action observée la disjonction des plans dont fait partie cette action. Cependant, lorsque plusieurs actions sont observées, la combinaison des déductions obtenues pour chaque action tend à produire un nombre démesuré de conclusions. Pour traiter ce problème, la méthode de reconnaissance de Kautz est basée sur un principe de *minimisation.* Elle détermine la classe des *plus petits ensembles de plans* parmi ceux qui contiennent les actions observées. Ce principe se justifie par la supposition que l'utilisateur cherche à

réaliser correctement son plan, et donc que les actions observées ont une certaine cohérence.

Nous présentons succinctement les bases de cette théorie. Nous décrivons la notion d'événement, centrale pour la théorie, et la structuration de la connaissance pour la reconnaissance ; puis nous donnons la définition de deux opérateurs d'inférence non-monotone : la c-implication, qui tire les conclusions d'une observation unique, et la mc-implication, qui tire les conclusions d'un groupe d'observations.

3.5.3 La notion d'événement

Il n'existe pas de distinction entre actions et plans, tous les deux sont représentés de la même façon, comme des *types d'événements*. Ils sont organisés en hiérarchies à l'aide de deux relations : la relation de *composition* et la relation d'*abstraction*. Un *plan* est constitué d'un ensemble (éventuellement ordonné) d'actions. Par exemple, le plan *DessinerMaison* est composé des actions *DessinerMurs, DessinerFenêtre* et *DessinerToit.* Un plan (ou une action) peut être plus ou moins général, et donc posséder des spécialisations. Par exemple, le plan *DessinerMaison* peut abstraire les plans *DessinerMaisonRustique et DessinerMaisonAlsacienne.* Une hiérarchie comprend des types d'événements, ainsi que des relations de composition et des relations d'abstraction sur ces types. Au sommet de la hiérarchie se trouve le type *Evénement Final* (E_F). C'est lui qui abstrait les types d'événements qui ne sont constituants d'aucun autre événement.

Une *observation* est une *description d'événements.* Elle inclut le type de l'événement et ses paramètres (tels que l'agent, l'instant auquel il s'est produit, etc.). Le résultat du processus de reconnaissance est une description d'un ou plusieurs événements de type E_F.

Le langage de représentation d'événements est le langage de prédicats du premier ordre avec égalité. Un modèle d'une formule est une interprétation du langage, qui associe à un terme, un individu ; à une fonction, une relation d'un n-uplet d'individus à un individu ; et à un prédicat, un ensemble de n-

uplets d'individus. Le domaine d'un modèle comprend des *intervalles temporels,* des *événements,* et les *paramètres des événements*.

Les *événements* sont des individus, et les *types* d'événements sont des prédicats. Diverses fonctions sur les événements, les *rôles,* lient les paramètres des événements. Par exemple, la formule :

$$\text{DessinerMaison(A)} \wedge (\text{agent(A)} = \text{Utilisateur}) \wedge (\text{temps(A)} = \text{T2})$$

décrit l'événement *A,* de type *DessinerMaison,* qui est réalisé par *l'utilisateur* à l'instant *T2*.

La connaissance de l'observateur est représentée à travers d'une hiérarchie d'événements. Une hiérarchie d'événements H est un quintuplet $\{H_E, H_A, H_{EB}, H_D, H_G\}$.

- H_E est l'ensemble des prédicats unaires de types d'événements. Il inclut les prédicats spéciaux *événement quelconque* (E_Q) et *événement final* (E_F).

- H_A est l'ensemble des axiomes d'abstraction, de la forme :

  $$\forall x, E_1(x) \supset E_2(x)$$

  pour $E_1, E_2 \in H_E$. Dans ce cas, E_2 *abstrait directement* E_1. La fermeture transitive de l'abstraction directe est l'abstraction ; et le fait que E_2 est le même ou abstrait E_1 est écrit E_2 *abstrait** E_1. L'événement spécial E_Q *abstrait** tous les types d'événements.

- H_{EB} est l'ensemble des prédicats de type d'événement *de base,* c'est-à-dire les membres de H_E qui n'abstraient aucun autre type d'événement.

- H_D est l'ensemble des axiomes de décomposition, de la forme

  $$\forall x, E_0(x) \supset E_1(f_1(x)) \wedge E_2(f_2(x)) \wedge \ldots \wedge E_n(f_n(x)) \wedge \kappa$$

  où $E_0, \ldots, E_n \in H_E$ et $f_1, \ldots, f_n$ sont des fonctions *rôles*, et κ est une sous-formule ne contenant aucun membre de H_E. La formule κ décrit

les *contraintes* sur E_0. Les E_1, ..., E_n sont les *composants directs* de E_0. Ni le type spécial E_F, ni aucun des types qu'il abstrait, n'apparaissent comme composants directs d'un autre type.

- H_G est l'ensemble des axiomes généraux, ceux qui ne contiennent aucun des membres de H_E. H_G inclut les axiomes concernant les relations temporelles entre les intervalles, ainsi que tous les autres faits ne concernant pas les événements.

3.5.4 La c-implication et la mc-implication

Dans la théorie de Kautz, la *reconnaissance* consiste à trouver l'événement (ou l'ensemble d'événements) de type E_F qui recouvre la séquence observée. Un tel événement est appelé *l'explication* de la séquence d'observations.

A partir des hiérarchies de composition et d'abstraction, il est possible de représenter des plans d'action structurés hiérarchiquement. Néanmoins, une telle hiérarchie ne justifie pas en soi l'inférence d'événements de type E_F à partir d'observations. Elle justifie bien la déduction des composants d'un événement à partir de celui-ci mais elle n'écarte pas la possibilité que ces composants surviennent sans lien avec les événements de type E_F.

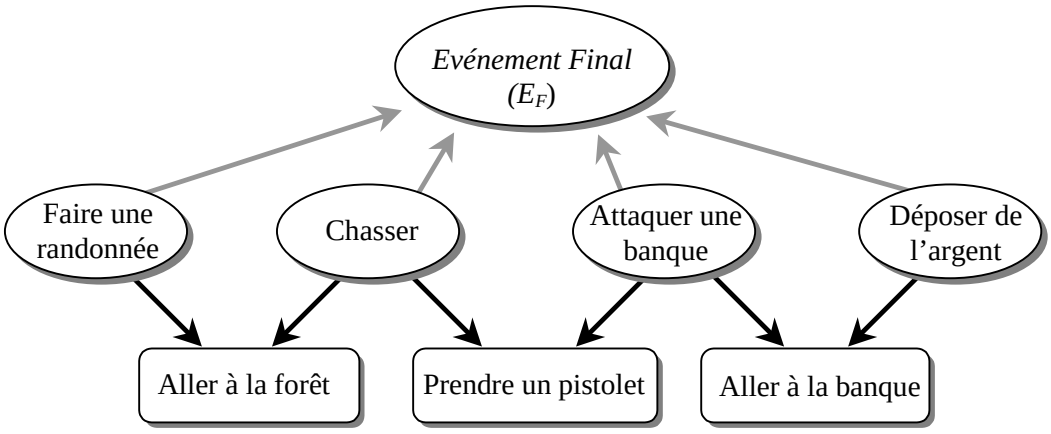


Figure 3.5. Hiérarchies de composition et abstraction (d'après [Kautz 90])

Considérons l'exemple, emprunté à [Kautz 90], dans la figure ci-dessus (les flèches en gris représentent l'abstraction « est-un », et les flèches en noir la composition « est-partie-de »). Supposons que *PrendrePistolet(C)* est observé. L'union de cette formule et de la hiérarchie H n'implique pas $\exists x$, *Chasser(x)*, ni $\exists x$, *Chasser(x)* $\vee$ *AttaquerUneBanque(x)*, ni même $\exists x$, $E_F(x)$. Il existe des modèles de *{PrendrePistolet(C)}* $\cup$ *H* dans lesquels aucune des ces affirmations n'est vraie. Par exemple (en décrivant un modèle par l'ensemble de ses atomes valides), aucune n'est vraie dans *{PrendrePistolet(C)}*, et seule la dernière est vraie dans *{PrendrePistolet(C), DeposerArgent(D), AllerBanque(f_1(D)), E_F(D)}*.

Pourtant, il semble raisonnable de conclure, d'après cette hiérarchie, que quelqu'un qui prend un pistolet prépare une partie de chasse ou un vol. Une telle conclusion peut se justifier en supposant que la hiérarchie est *complète*, c'est-à-dire, que tout événement observé, s'il n'est pas de type E_F, est un composant d'un autre événement, et que la relation de l'événement au composant apparaît dans la hiérarchie. Cette hypothèse de complétude peut s'exprimer par la définition d'une sous-classe particulière des modèles de *H*, nommés *modèles couvrants*. Dans notre hiérarchie, *{PrendrePistolet(C), C = f_1(D), Chasser(D), AllerForêt(E), E = f_2(D), E_F(D) }* et *{PrendrePistolet(C), C = f_1(D), AttaquerBanque(D), AllerBanque(E), E = f_2(D), E_F(D) }* sont des modèles couvrants de *H*, à la différence des autres modèles décrits plus haut.

Dans un modèle couvrant, tout événement est soit de type *événement final* (E_F), soit composant d'un événement de type E_F. Dans un modèle couvrant pour une observation donnée, les événements de type E_F sont les *explications* de l'observation. Pour obtenir les modèles couvrants, l'auteur s'appuie sur le concept de *modèle minimal* [McCarthy 80, 84] :

Soit M_1 un membre d'une classe de modèles μ et π un ensemble de prédicats. M_1 est *minimal en* π *parmi* μ si et seulement s'il n'existe aucun autre modèle M_2 tel que

- M_2 est un membre de μ
- M_1 et M_2 ont le même domaine

- M_1 et M_2 possèdent la même interprétation de tous les prédicats, constantes et fonctions en dehors de π

L'extension de tout membre de π dans M_2 est un sous-ensemble de l'extension de ce prédicat dans M_1

Alors, la construction de modèles couvrants se fait en deux pas. Primo, la hiérarchie d'abstraction est complétée (on obtient un modèle *A-clos*). Secundo, la hiérarchie de décomposition est complétée. La définition d'un modèle couvrant est la suivante :

Soit H une hiérarchie, et M un modèle de H_A.

- M est *clos sous spécialisation* si M est minimal en H_E - H_{EB} parmi les modèles de H_A
- M est *clos sous abstraction* si M est minimal en H_E - $\{E_Q\}$ parmi les modèles de H_A qui sont clos sous spécialisation.
- M est un *modèle A-clos* de H si M est un modèle de H et M est aussi un modèle de H_A qui est clos sous abstraction.
- enfin, un modèle M est un modèle couvrant de H si M est minimal en H_E - $\{E_F\}$ parmi les modèles *A-clos* de H.

La notion de modèle couvrant sert à définir une relation sémantique, la c-implication (notée $\models_c$). Plutôt que d'adopter arbitrairement un modèle couvrant particulier, il est raisonnable de retenir les propositions valides dans tous les modèles couvrants. Ces propositions sont dites *c-impliquées* par l'observation. Sur l'exemple, « *Chasser* ou *AttaquerBanque* » se produit dans tous les modèles couvrants dans lesquels un individu prend un pistolet :

$$\text{PrendrePistolet(C)} \ {}_H\models_c \exists x, \text{Chasser(x)} \vee \text{AttaquerBanque(x)}$$

Ainsi une formule Γ *c-implique* Ω, notée $\Gamma \ {}_H\models_c \Omega$, si Ω est vrai dans tous les modèles couvrants de H où Γ est vrai.

La c-implication se rattache de façon simple à l'implication ordinaire, par l'intermédiaire d'une fonction de fermeture, notée *cl*. Cette fonction s'applique à une hiérarchie H pour produire les axiomes de fermeture *cl(H)* de cette hiérarchie. Elle fait correspondre, à la définition sémantique de la c-implication, la déduction usuelle en logique du premier ordre. La fonction *cl*

génère les axiomes correspondant à trois hypothèses (d'exhaustivité, de disjonction et de décomposition) sur la hiérarchie. La hiérarchie d'abstraction est renforcée en posant l'hypothèse qu'il n'existe pas de types d'événement en dehors de H_E (hypothèse d'exhaustivité), et que deux types d'événements sont disjoints sauf si l'un est abstrait par l'autre ou qu'ils sont abstraits par un type commun (hypothèse de disjonction). La hiérarchie de composition est renforcée en posant l'hypothèse que les événements (à l'exception du type E_F) se produisent seulement en tant que composants d'autres événements (hypothèses de décomposition). Ainsi, les axiomes générés pour notre exemple seront de la forme :

pour l'hypothèse d'exhaustivité :

$$\forall x, E_F(x) \supset (FaireRandonné(x) \vee Chasser(x) \vee AttaquerBanque(x) \vee DéposerArgent(x))$$

pour l'hypothèse de disjonction :

$$\forall x, \neg FaireRandonné(x) \vee \neg DéposerArgent(x)$$

pour l'hypothèse de décomposition :

$$\forall x, PrendrePistolet(x) \supset ((\exists y, Chasser(y) \wedge x = f_1(y)) \vee (\exists y, AttaquerBanque(y) \wedge x = f_1(y)))$$

La c-implication ne combine pas l'information provenant de plusieurs observations. Nous voudrions réaliser l'intersection des explications possibles pour chaque événement. Cette opération est réalisée par la sélection de modèles qui minimisent le *nombre* d'événements de type E_F. Pour cela, il faut d'abord introduire la notion de *cardinalité minimale* :

Soit M_1 un membre d'une classe de modèles μ, et π un prédicat. M_1 est de *cardinalité minimale en π parmi μ* si et seulement s'il n'existe aucun autre modèle M_2 tel que :

- M_2 est un membre de μ
- le cardinal de l'extension de π dans M_2 est inférieur au cardinal de l'extension de π dans M_1.

Alors, soit Γ un ensemble de formules et H une hiérarchie, M est un *modèle couvrant minimal* de Γ (relatif à H) si

- *M* est un modèle de Γ
- *M* est un modèle couvrant de *H*
- *M* est de cardinalité minimale en E_F parmi les modèles couvrants de *H*

La notion de modèles couvrants minimaux permet d'introduire une nouvelle relation sémantique, la mc-implication : la formule Γ *mc-implique* Ω, noté $\Gamma_H \models_{mc} \Omega$, si Ω est vrai dans tous les modèles couvrants minimaux de *H* où Γ est vrai.

La mc-implication se rattache à l'implication ordinaire, par l'intermédiaire de règles par défaut de cardinalité minimale.

Soit la suite de propositions

MA_0 : $\forall x, \neg E_F(x)$

MA_1 : $\forall x, \forall y, E_F(x) \wedge E_F(y) \supset x = y$

MA_2 : $\forall x, \forall y, \forall z, E_F(x) \wedge E_F(y) \wedge E_F(z) \supset (x = y) \vee (y = z)$ i $(x = z)$

…

La première affirme qu'il n'existe pas d'événement de type E_F ; la seconde, qu'il n'existe pas plus d'un événement de type E_F ; la troisième, pas plus de deux, etc. Supposons qu'il existe un modèle couvrant minimal dans lequel l'extension du prédicat E_F est finie. Alors

$$\Gamma_H \models_{mc} \Omega$$

si et seulement si

$$\Gamma \cup cl(H) \cup MA_i \vdash \Omega$$

où *i* est le plus petit entier tel que la partie gauche de la relation soit consistante.

3.5.5 L'explication d'une observation

Comme nous l'avons vu dans les paragraphes précédents, notre processus de reconnaissance est défini en deux pas : (i) la construction de la description *récente*, et (ii) la reconnaissance proprement dite, qui cherche à établir le plan développé par l'utilisateur à partir de la description récente.

Le processus de formation de la description récente comporte une phase de réduction semblable à celle utilisée pour l'apprentissage d'une tâche, mais elle est plus complexe puisqu'elle doit prendre en compte sa nature incrémentale. A chaque fois que la machine observe un événement, celui-ci est intégré à la description récente. On fait les réductions appropriées pour obtenir une description plus simple, dépouillée des actions intermédiaires redondantes. Cependant, l'intégration d'une nouvelle action observée peut entraîner l'élimination des conditions qui invalident des déductions préalables.

Dans notre exemple avec les concepts « dessiner-bateau et dessiner-voiture », après que l'utilisateur a dessiné deux cercles, le processus de reconnaissance a établi deux hypothèses. Puis, si l'utilisateur change la position d'un cercle de façon à que les cercles restent alignés verticalement –ce qui implique une réduction de l'action « déplacer »–, la déduction doit être annulée puisqu'elle n'est plus valide. En même temps, ces nouvelles conditions créées renvoient à un autre ensemble d'hypothèses. Ainsi le processus de construction de la description récente et le processus de reconnaissance sont imbriqués. Si l'intégration d'une nouvelle observation n'affecte pas les observations précédentes, alors le processus de reconnaissance continue la recherche du plan à partir de cette nouvelle observation et de l'ensemble d'hypothèses calculées auparavant. Par contre, si la nouvelle observation transforme une des observations précédentes –à cause de l'application d'une opération de réduction– il faut vérifier si les hypothèses calculées par le processus de reconnaissance sont encore valables. Si ce n'est pas le cas, le processus de reconnaissance doit recommencer à partir de la nouvelle description récente.

Par ailleurs, le deuxième pas du processus de reconnaissance utilise les idées développées par Kautz. Il faut noter que cette méthode fait la supposition que la hiérarchie d'abstraction et de composition, sur laquelle se fonde la reconnaissance, est complète. En conséquence, à chaque présentation d'une nouvelle tâche, un processus d'intégration de cette nouvelle connaissance reconstruit à nouveau la hiérarchie.

3.6 Conclusions

Nous avons présenté dans ce chapitre les principaux aspects de la notion de coopération utilisés dans notre travail : l'apprentissage et la reconnaissance de tâches. Les principaux problèmes abordés à propos de l'apprentissage de la tâche sont l'acquisition d'un nouveau plan et sa caractérisation à partir des exemples. C'est à partir de ces plans, acquis au cours de diverses sessions de travail, que la machine est en mesure d'assister l'utilisateur par la reconnaissance du plan poursuivi.

Pour aborder ces deux sujets, notre travail utilise différentes notions issues de l'intelligence artificielle. D'abord, la représentation de la tâche est basée sur un formalisme qui incorpore des aspects fondamentaux pour représenter des actions [Shoham 87]. Il permet d'exprimer des notions de précédence temporelle pour fixer un ordre et, de cette façon, faire ressortir les parallélismes possibles et les séquencements obligatoires entre les actions d'un plan.

Ensuite, une méthode d'apprentissage de concepts à partir d'exemples a été présentée. La machine acquiert les connaissances pour faire une tâche, et le processus de formation des concepts fait évoluer ses connaissances chaque fois que l'utilisateur montre un nouvel exemple de la tâche. La formation d'un concept est réalisée à partir de la recherche des *actions caractéristiques* du concept, et la recherche des *relations descriptives*, c'est-à-dire les associations inhérentes au concept. Pour chaque opération, une stratégie de généralisation différente est exposée : la *généralisation plus spécifique* pour la recherche d'actions caractéristiques [Bergadano & Gunetti 96], et *l'hypothèse la plus spécifique* pour les relations descriptives [Muggleton 95].

Il est intéressant de relever un inconvénient de cette approche d'apprentissage : sa dépendance vis-à-vis de l'utilisateur. L'ordre de présentation des exemples, le nombre d'exemples et la classification d'une tâche –tous les trois décidés par l'utilisateur– sont des aspects déterminants qui influent fortement sur les concepts appris et par conséquent, dans la reconnaissance de la tâche.

Finalement, le problème de la reconnaissance de plans est discuté. Dans notre cas, la reconnaissance se fait à l'aide de deux processus : la construction de la *description récente* –l'ensemble d'événements observés après réduction, c'est-à-dire, dépouillé des actions intermédiaires redondantes– ; et la reconnaissance du plan à partir de la description récente. Pour ce deuxième processus, notre travail utilise les notions de la théorie formelle de la reconnaissance de plans développée par Kautz [Kautz 87]. Cette théorie se base sur la notion d'événements, organisés en *hiérarchies* à partir de relations d'abstraction et de composition. Deux opérateurs sémantiques, la *c-implication* et la *mc-implication*, permettent de tirer les conclusions respectivement d'une observation et d'un ensemble d'observations. Il est important de noter que la méthode de reconnaissance présentée par Kautz n'est pas implantée directement, mais constitue une spécification formelle de l'algorithme de reconnaissance. La mise en œuvre suggéré par Kautz est basée sur l'unification de graphes ET/OU.

Un problème à noter dans la méthode de Kautz est la reconnaissance des plans erronés, laquelle est écartée d'emblée, par l'hypothèse selon laquelle les utilisateurs sont infaillibles, et ne commettent ni erreur ni incohérence [Py 90, Nerzec 93]. Dans notre cas ce problème est moins significatif puisque l'intervention de la machine dépend toujours du consentement de l'utilisateur.

CHAPITRE IV

Un modèle pour le dialogue coopératif

La logique que nous visons tente la modélisation des échanges dialogiques entre l'utilisateur et la machine en utilisant le concept des actes de dialogue. L'idée principale est l'intégration des actes de dialogue dans un cadre logique fondé sur l'action qui permet d'expliquer le déroulement d'un dialogue homme-machine. Ce travail s'inspire des travaux proposés par [Caelen & Villaseñor 97, Cohen & Levesque 90, Halpern & Moses 92, Hoek et al. 94a, Hoek et al. 94b, Linder et al. 94, Prendinger & Schurz 96]. Il contient des éléments d'une logique épistémique –pour la représentation des connaissances–, d'une logique de l'action –pour décrire les actions et les effets produits par ces actions–, et d'une logique dialogique –pour exprimer les engagements, les interruptions et les incompréhensions lors des échanges dialogiques, en fonction des buts du dialogue.

4.1 Concepts de base

La *connaissance* est représentée par l'opérateur ***s*** (savoir), par exemple la formule ***Us φ*** exprime que l'utilisateur sait la proposition ***φ*** (pour faire la distinction entre l'utilisateur et la machine, les deux agents possibles dans notre logique, on utilise les lettres ***U*** pour l'utilisateur et ***M*** pour la machine). Pour *l'action* on introduit la notion *d'événement* : un événement ***Uf α*** est la réalisation de l'action ***α*** par l'utilisateur (ou ***Mf α*** par la machine) et si cet événement a la proposition ***φ*** comme résultat, en utilisant la notation de la logique de l'action, on a la formule suivante ***[Uf α] φ*** (après avoir terminé l'exécution de ***α*** par l'utilisateur, ***φ*** est vraie).

Par la suite, les symboles suivants dénotent :

φ, ϕ :	des propositions
α, β, γ :	des actions
¬, ∧, ∨, ⊃ :	les connecteurs logiques classiques de négation, conjonction, disjonction et implication
U, **M** :	les deux possibles agents, l'utilisateur et la machine

Les modalités sont dénotées à partir des expressions suivantes :

Pour décrire l'action et ses effets.

[Uf α]φ, **[Mf α]φ** : un événement est la réalisation de l'action ***α*** par l'utilisateur (ou la machine) avec comme résultat la proposition ***φ***; autres modalités possibles pour la réalisation de l'action : ***ff, fs, ffs***.

Pour la représentation de connaissances

Us φ, **Ms φ** : l'utilisateur (ou la machine) sait ***φ***

Pour la représentation de l'intention :

Ui φ : l'utilisateur à l'intention de rendre ***φ*** vraie

Figure 4.1. Notations

À travers ce formalisme, on est donc capable de décrire soit une action de base (instruction élémentaire) soit une tâche, c'est-à-dire une séquence d'actions organisée à travers un plan. Il sert aussi à représenter des objets comme résultats de ces actions. Ainsi le *but* de l'utilisateur est-il vu comme l'ensemble d'effets produits par la réalisation des actions. But et objet sont donc deux notions complètement liées. Pour décrire les objets, on va focaliser

notre attention sur la séquence d'actions observées et sur le résultat des actions, dans le contexte du but à atteindre. Pour représenter le but de l'utilisateur, c'est-à-dire son *intention*, on utilise l'opérateur *i*. Ainsi la formule $Ui\ \varphi$ exprime l'intention de l'utilisateur de rendre φ vraie. Il faut noter que seul l'utilisateur est capable d'agir intentionnellement, et non la machine.

4.2 L'acte de dialogue

Un acte de dialogue est un acte qui entraîne un changement (éventuellement nul) de la situation par rapport à la tâche (et/ou de la connaissance en machine sur la tâche) et un changement de la situation du dialogue. C'est pourquoi un acte de dialogue est défini par un événement $[Uf\ \alpha](\varphi_\alpha \wedge \varphi_d)$ où le résultat est l'ensemble des changements de situation par rapport à la tâche φ_α et par rapport à la situation du dialogue φ_d. Les changements par rapport à la tâche sont les effets propres de l'action (φ_α). Et les changements par rapport au dialogue sont les conséquences qui montrent l'avancement du but à chaque tour de parole (φ_d).

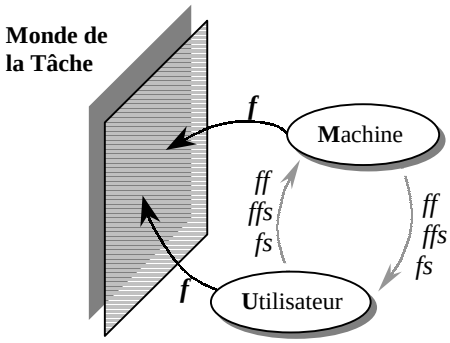


Figure 4.2. Les Actes de Dialogue

Les actes de dialogue sont exprimés à partir des actions de la manière suivante :

Uf α, Mf α	l'utilisateur ou la machine *fait* une action α
Uff α, Mff α	l'utilisateur *fait faire* une action α à la machine (ou vice versa)

Ufs ϕ, Mfs ϕ	l'utilisateur *fait savoir* ϕ à la machine (ou vice versa)
Uffs ϕ, Mffs ϕ	l'utilisateur *fait faire-savoir (=demande)* ϕ à la machine (ou vice versa) [5]

Dans notre cas, un acte de dialogue peut être langagier ou non-langagier, et peut émaner de l'utilisateur ou de la machine. Cependant, la principale différence est qu'on attribue à l'utilisateur une intention, c'est-à-dire que les actions réalisées par l'utilisateur sont orientées vers la recherche d'un objectif. Ainsi on distingue les actes de l'utilisateur de ceux de la machine. Les premiers expriment l'intention de l'utilisateur et, en conséquence, dirigent le dialogue. Au contraire, ceux de la machine sont toujours dépourvus d'intention et subordonnés à l'objectif visé par l'utilisateur.

Les actes de dialogue de la machine sont donc décrits de la manière suivante :

a) Exécution de l'action.

[Mf α]φ

La machine exécute l'action **α** avec comme résultat l'ensemble d'effets **φ.**

b) Réponse à une question.

[Mfs ϕ](Ms Us ϕ)

La machine présente la proposition **ϕ** à l'utilisateur : comme résultat, la machine sait que l'utilisateur connaît maintenant **ϕ.**

c) La machine pose une question à l'utilisateur.

[Mffs ϕ](Ms Us M¬s (ϕ))

La machine fait faire savoir une proposition **ϕ**, cette demande d'information a pour résultat immédiat que la machine sait que l'utilisateur sait que la machine ne connaît pas la valeur de vérité de la proposition **ϕ**.

Par ailleurs, les actes de dialogue de l'utilisateur ont une incidence dans le déroulement du dialogue : c'est l'intention exprimée qui motive la réaction de la machine :

[5] Il est toujours possible de penser à un ensemble de catégories comme : $Uffs_{si}$ quand l'utilisateur pose une question oui/non ou $Uffs_{ref}$ si l'utilisateur pose une question pour connaître la valeur d'un certain paramètre [Colineau 97].

a) Demande d'exécution d'une action

[Uff α](Ms Ui [Mf α])

L'utilisateur fait faire une action **α** à la machine. L'effet de cet acte est la définition de l'intention de l'utilisateur –le but poursuivi– c'est dans cet exemple la réalisation de l'action **α** par la machine.

[Uf α](Ms Ui [Mf α])

Cette autre formule, avec le même effet que la précédente, décrit les actions que l'utilisateur fait en utilisant la manipulation directe. Dans cette situation l'utilisateur a la sensation d'être lui-même l'auteur de l'action, mais c'est réellement la machine qui est derrière l'exécution de l'action. Ainsi, dans la première formule, l'utilisateur demande à la machine la réalisation d'une action **α.** Dans ce cas, la machine est un *partenaire* à qui l'utilisateur délègue la réalisation de la tâche. Par contre, dans la deuxième formule, c'est l'utilisateur lui-même qui réalise l'action. Il a utilisé la machine comme un *outil* pour mener à bien la réalisation de l'action **α**. Par conséquent, une autre distinction entre les deux formules est le type d'action associée. La machine outil est restreinte aux actions de base à la différence de la machine partenaire.

b) L'utilisateur pose une question à la machine

[Uffs ϕ](Ms Ui [Mfs ϕ])

Cet acte de l'utilisateur tombe dans le dialogue en définissant comme but le faire savoir **ϕ** par la machine.

c) Présentation d'une proposition à la machine

[Ufs ϕ](Ms Ui Ms ϕ)

L'utilisateur fait savoir une proposition **ϕ** à la machine. Après cet acte, la machine sait que l'intention de l'utilisateur est une mise à jour de la connaissance en machine par rapport à **ϕ**.

A partir de ces descriptions des actes de dialogue –éléments de base pour les échanges dialogiques– la section suivante montre les axiomes de notre modèle de dialogue coopératif homme-machine.

4.3 Définition des axiomes

Les sections suivantes présentent les axiomes de notre logique. La première montre brièvement les axiomes qui caractérisent la connaissance. Ensuite nous présentons les définitions associées à la convergence du dialogue, afin d'introduire, dans les deux dernières sections, les différents axiomes pour la caractérisation de la coopération et la gestion des échanges dialogiques.

4.3.1 Caractérisation de la connaissance

Les axiomes suivants caractérisent la connaissance en machine : sur les tâches (le savoir-faire requis par la réalisation d'une tâche), et sur les intentions de l'utilisateur (inférées à partir des demandes ou questions posées, indispensables pour la gestion du dialogue, ***Ms φ***). Ils sont écrits de façon générale, c'est-à-dire du point de vue de l'utilisateur ou de la machine[6]. Pour $\mathbf{A} = \{U,M\}$:

(**A.1**)	$\mathbb{A}s\ \varphi \wedge \mathbb{A}s(\ \varphi \supset \phi\ ) \supset \mathbb{A}s\ \phi$	axiome K
(**A.2**)	$\mathbb{A}s\ \varphi \supset \varphi$	on sait seulement des faits vrais
(**A.3**)	$\mathbb{A}s\ \varphi \supset \mathbb{A}s\ \mathbb{A}s\ \varphi$	introspection positive - on sait qu'on sait
(**A.4**)	$\neg\ \mathbb{A}s\ \varphi \supset \mathbb{A}s\ \neg\mathbb{A}s\ \varphi$	introspection négative - on sait qu'on ne sait pas

Cette caractérisation de la connaissance n'a pas l'intention de donner à la machine une conscience propre, il s'agit seulement des mécanismes logiques nécessaires pour réaliser les déductions pertinentes.

4.3.2 La convergence du dialogue

Comme nous l'avons vu auparavant, le dialogue est une suite coordonnée d'actions (langagières et non-langagières) devant conduire à un but (cf. § 2.6.2). Ce but doit être à la fois *atteint* et *satisfait* pour que le dialogue aboutisse à un succès (cf. § 2.6.3.2). Dans notre modèle, le déroulement du dialogue est réglé par l'intention de l'utilisateur, on suppose qu'à un moment donné de l'échange, un acte de dialogue de l'utilisateur

[6] Pour une présentation détaillée voir le système S5 dans [Halpern & Moses 92].

exprime son intention, c'est-à-dire, un but est *posé*. Cette intention de l'utilisateur est interprétée comme l'action souhaitée à exécuter (ou le plan d'action à développer) dont l'effet attendu est le but à atteindre. Si le dialogue se déroule normalement la machine exécute l'action souhaitée et le but est *atteint*. Puis, après l'acceptation par l'utilisateur de l'action exécutée, le but peut être considéré comme *satisfait*. Dans cette version de notre formalisme, nous supposons un seul fil d'activité, c'est-à-dire qu'après que l'utilisateur a posé le but, le dialogue se poursuit jusqu'à la satisfaction ou l'abandon de celui-ci. En conséquence, il n'y pas de buts conditionnés ni différés.

La convergence du dialogue vers un but est annotée de la manière suivante :

- **?** but posé, action (ou plan) à faire
- **+** but atteint, action réalisée sans confirmation de l'utilisateur mais vraisemblablement correcte
- **++** but satisfait, action achevée correcte et confirmée par l'utilisateur
- **@** but abandonné

Cette convergence dépend de la situation et de l'acte même. Examinons cela en détail.

4.3.2.1 But posé par l'utilisateur

Un but est ***posé*** par l'utilisateur quand celui-ci fait soit une demande de réalisation d'une action *Uff,* soit une demande d'information *Uffs*. Ce but est l'état initial du processus. Il est rangé dans une pile et va subir une série de transformations. A l'état final, le but est ***atteint***, si la machine répond *Mfs* ou exécute l'action demandée *Mf,* puis il est ***satisfait*** (c'est-à-dire sorti de la pile) quand l'utilisateur exprime son avis, de manière explicite ou implicite, sur l'adéquation de l'action ou la réponse obtenue. Un but peut-être aussi ***abandonné*** en cours de route par l'utilisateur quand celui-ci communique explicitement son changement d'avis. Pour le cas où l'utilisateur demande la réalisation d'une action *Uff,* on a les notations suivantes :

i) But ***posé***, l'abréviation *?[Mf α]* désigne l'action *[Mf α]φ* comme but posé.

(**A.5**) [Uff α] (Ms Ui [Mf α]φ) $\stackrel{def}{=}$ [Uff α] (? [Mf α])

un but posé est l'effet d'une demande et cet effet est exprimé comme l'intention de l'utilisateur (pour que la machine fasse une action) intégrée aux connaissances en machine.

ii) But ***atteint***, l'abréviation $+[Mf\ \alpha]$ désigne l'action $[Mf\ \alpha]\varphi$ comme but atteint.

(**A.6**) $Ms\ Ui\ [Mf\ \alpha]\varphi \wedge [Mf\ \alpha]\varphi$
$\stackrel{def}{=} ?\ [Mf\ \alpha] \wedge [Mf\ \alpha]\varphi$
$\stackrel{def}{=} +\ [Mf\ \alpha]$

un but posé est atteint quand l'action demandée par l'utilisateur est aussi vraie.

iii) But ***satisfait***, l'abréviation $++[Mf\ \alpha]$ désigne l'action $[Mf\ \alpha]\varphi$ comme but satisfait.

(**A.7**) $Ms\ Ui\ [Mf\ \alpha]\varphi \wedge [Mf\ \alpha]\varphi \wedge$
$(\ Ms\ U\neg i\ [Mf\ \alpha]\varphi \vee (\ Ms\ Ui\ [Mf\ \beta]\gamma \wedge \neg\ relation(\alpha,\beta)))$
$\stackrel{def}{=} +\ [Mf\ \alpha] \wedge (? \neg\ [Mf\ \alpha] \vee (\ ?\ [Mf\ \beta] \wedge \neg\ relation(\alpha,\beta)))$
$\stackrel{def}{=} ++\ [Mf\ \alpha]$

un but posé est satisfait si l'utilisateur accepte l'action comme appropriée, cette acceptation peut être manifestée soit de façon (a) explicite quand l'utilisateur fait savoir qu'il ne garde plus son intention à l'égard de l'action demandée ; soit de façon (b) implicite, quand l'utilisateur demande une nouvelle action qui ne dérive pas de l'action demandée, c'est-à-dire quand la nouvelle action n'est pas en *relation* avec l'action précédente. L'action α est en relation avec l'action β, *relation(α,β)*, si parmi les préconditions de β il existe des postconditions de α. (cf. § 4.3.3.1).

iv) But ***abandonné***, l'abréviation $@[Mf\ \alpha]$ désigne l'action $[Mf\ \alpha]\varphi$ comme but abandonné.

(**A.8**) $Ms\ Ui\ [Mf\ \alpha]\varphi \wedge Ms\ Ui\ [M\neg f\ \alpha]\varphi$
$\stackrel{def}{=} ?\ [Mf\ \alpha] \wedge ?\ [M\neg f\ \alpha]$
$\stackrel{def}{=} @\ [Mf\ \alpha]$

un but posé est abandonné quand l'utilisateur fait savoir qu'il veut que la machine ne fasse pas l'action concernée.

Pour le cas où l'utilisateur pose une question *Uffs*, on a le même état-but. La seule différence est qu'on attend une réponse du type *Mfs*. Une question *posée* par l'utilisateur devient une demande d'information :

(**A.9**) $[\text{Uffs } \phi]\ (\ \text{Ms Ui } [\text{Mfs } \phi]\varphi\ ) \stackrel{def}{=} [\text{Uffs } \phi]\ (\ ?\ [\text{Mfs } \phi]\ )$

4.3.2.2 But déplacé en vue d'un apprentissage

La coopération de la machine est motivée par la connaissance qu'elle possède de l'intention de l'utilisateur et de la tâche à réaliser, et c'est au cours du dialogue que ces connaissances sont communiquées. Dans le cas où la tâche demandée par l'utilisateur est inconnue par la machine, le but initial est *déplacé* de façon à conduire le dialogue dans un contexte d'acquisition car on suppose que l'utilisateur a comme intention élémentaire le transfert de son savoir-faire. Pour ce cas, on a les formules suivantes :

i) But ***déplacé en acquisition***, l'abréviation *?Δ[Uf α]* désigne comme but du dialogue l'acquisition de la tâche α.

(**A.10**) $[\text{Mff } \alpha](\ \text{Ms Ui } [\text{Uf } \alpha]\ \text{Msf } \alpha\ ) \stackrel{def}{=} [\text{Mff } \alpha](\ ?\Delta\ [\text{Uf } \alpha]\ )$
où $\text{Msf } \alpha = \text{Ms plan}(\alpha, \text{préconditions}_\alpha, \text{actions}_\alpha, \text{postconditions}_\alpha)$

ii) But ***en acquisition satisfait***, l'abréviation *++Δ[Uf α]* désigne l'acquisition de la tâche α comme but atteint et satisfait.

(**A.11**) $\text{Ms Ui } [\text{Uf } \alpha]\text{Msf } \alpha \wedge \text{Ms U}\neg\text{i } [\text{Uf } \alpha]\text{Msf } \alpha$
$\stackrel{def}{=} ?\Delta[\text{Uf } \alpha\ ] \wedge \text{Ms U}\neg\text{i } [\text{Uf } \alpha]\text{Msf } \alpha$
$\stackrel{def}{=} ++\Delta\ [\text{Uf } \alpha]$

un but en acquisition est atteint et satisfait quand l'utilisateur fait savoir que la tâche est achevée.

iii) But ***en acquisition abandonné***, l'abréviation *@Δ[Uf α]* désigne un but en acquisition abandonné.

(**A.12**) $\text{Ms Ui } [\text{Uf } \alpha]\text{Msf } \alpha \wedge \text{Ms Ui } [\text{U}\neg\text{f } \alpha]\text{Msf } \alpha$
$\stackrel{def}{=} ?\Delta[\text{Uf } \alpha\ ] \wedge \text{Ms Ui } [\text{U}\neg\text{f } \alpha]\text{Msf } \alpha$
$\stackrel{def}{=} @\Delta\ [\text{Uf } \alpha]$

4.3.2.3 L'incorporation de nouvelles connaissances

La présentation d'une nouvelle proposition à la machine pose un nouveau but, qui indique l'intention de l'utilisateur de modifier la connaissance de la machine. Ce changement peut entraîner la simple acceptation de l'information apportée, ou bien une série d'inférences qui peuvent soit instaurer de nouveaux buts, soit occasionner des incohérences qui devront être résolues à travers des sous-dialogues de réparation. Pour ces situations, on a les formules suivantes :

i)***Nouvelle connaissance à intégrer***, l'abréviation *?(Ms ϕ)* désigne comme but du dialogue l'incorporation de la proposition *ϕ.*

(**A.13**) $[\text{Ufs } \phi](\text{ Ms Ui Ms } \phi\text{ }) \stackrel{def}{=} [\text{Ufs } \phi](\text{ ? (Ms } \phi\text{) })$

ii) ***Connaissance acceptée***, l'abréviation *++(Ms ϕ)* désigne l'acceptation après incorporation de la proposition *ϕ.*

(**A.14**) $[\text{Ufs } \phi](\text{ Ms Ui Ms } \phi\text{ }) \wedge \text{Ms } \phi$
$\stackrel{def}{=} \text{ ?(Ms } \phi\text{)} \wedge \text{Ms } \phi$
$\stackrel{def}{=} \text{ ++ (Ms } \phi\text{)}$

une nouvelle connaissance est acceptée quand celle-ci est passée à la base de connaissance de la machine.

Dans ce cadre, proposé pour la coopération homme-machine, la machine n'a pas d'intention. Néanmoins, il est possible qu'elle prenne une certaine initiative dans le dialogue vis-à-vis de la tâche demandée. Le premier cas se produit quand l'utilisateur enseigne à la machine une nouvelle tâche : c'est le cas d'un but déplacé en acquisition. Le deuxième cas se produit quand la connaissance en machine entre en contradiction avec la proposition affirmée par l'utilisateur : la machine demande alors à l'utilisateur de confirmer ou d'expliciter sa connaissance. Enfin, le troisième cas se produit si la machine ignore une condition préliminaire indispensable pour la tâche à faire (une tâche connue par la machine), ce qui déclenchera une demande d'information dirigée vers l'utilisateur. Dans tous les cas, c'est toujours l'intention de l'utilisateur qui est à l'origine des actes de la machine. On peut seulement

dire que les "initiatives" de la machine sont justifiées non par la tâche elle-même mais par des exigences de coopérativité.

4.3.3 Caractérisation de la coopération.

Le dialogue coopératif est orienté vers la réalisation d'une tâche pour laquelle la machine est considérée comme un assistant qui collabore à l'accomplissement de cette tâche. Dans notre cas, le dialogue est aussi le moyen pour *faire-savoir* une nouvelle tâche. Il s'agit là non de communiquer des informations mais de montrer une série d'actions pour arriver à un but. Par conséquent, l'utilisateur est sollicité à montrer son savoir-faire, et le système est incliné, par une attitude appropriée, à l'acquérir. Les axiomes suivants permettent de structurer le dialogue autour de ces attitudes.

4.3.3.1 L'exécution de la tâche

Si l'utilisateur commande une tâche et que la machine connaît le plan pour la faire, alors la machine est en position de la réaliser. Dans le cas d'une action de base, la machine connaît toujours le plan pour son exécution. Un plan est constitué d'une séquence d'actions, d'un ensemble de préconditions et d'un ensemble de postconditions. On utilise l'abréviation *Msf* α pour la proposition :

$$\text{Ms plan}(\alpha, \text{préconditions}_\alpha, \text{actions}_\alpha, \text{postconditions}_\alpha).$$

Pour l'application d'un plan, il faut couvrir l'ensemble de préconditions avant l'exécution de la séquence d'actions. Et l'ensemble de postconditions est l'ensemble de formules à vérifier après l'exécution de la tâche. Ainsi, dans le cas de l'exécution d'une tâche, le but est étendu pour prendre en compte la vérification des préconditions, l'exécution de la séquence d'actions et la vérification des postconditions.

(**A.15**) $?[\text{Mf } \alpha] \wedge \text{Msf } \alpha \wedge \alpha \in \text{Ab} \supset [\text{Mf } \alpha]\ \varphi$
pour une action de base Ab

(**A.16**) $?[Mf\ \beta] \wedge Ms\ plan(\beta, préconditions_\beta, actions_\beta, postconditions_\beta)$
$\supset ?_{[Mf\ \beta]}$ [Mf vérifier ($préconditions_\beta$), Mf $actions_\beta$,
Mf vérifier ($postconditions_\beta$),
Mf vérifier ($U\neg i$ [Mf β])]

pour une tâche composée

Il faut noter que dans le cas d'une tâche composée, la machine vérifie si sa réponse est pertinente vis-à-vis de l'intention de l'utilisateur. Cette vérification entraîne un sous-dialogue pour conclure la réalisation de la tâche demandée.

Dans la vérification des préconditions, si l'une d'entre elles n'est pas vérifiable mais que l'on connaît une action qui la rend vraie, alors on considère l'action comme un sous-but de la tâche demandée.

(**A.17**) $?[Mf\ vérifier\ \varphi] \wedge M\ \neg s\ \varphi \wedge Msf\ \beta \wedge \varphi \in postconditions_\beta$
$\supset ?_{[Mf\ vérifier\ \varphi]}[\ Mf\ \beta]$

4.3.3.2 L'acquisition d'une tâche

Comme nous l'avons vu, un but posé par l'utilisateur et inconnu par la machine sera momentanément déplacé pour guider le dialogue dans un contexte d'acquisition. Ce déplacement repose sur la supposition que l'utilisateur est toujours disposé à montrer la tâche si la machine l'ignore ou si son savoir-faire est insuffisant. C'est-à-dire, on suppose que l'utilisateur a l'intention, toujours présente, de lui enseigner une tâche pour que la machine l'apprenne :

(**A.18**) $?[Mf\ \alpha] \wedge M\neg\ sf(\alpha) \supset [Mff\ \alpha](\ ?\Delta\ [Uf\ \alpha]\ )$

acquisition d'une tâche

(**A.19**) $\forall\beta\ (\ ++_{\Delta\ [Uf\ \alpha]}\ [Mf\ \beta] \wedge\ ++\Delta\ [Uf\ \alpha] \wedge$
$\wedge\ Ms\ plan(\beta, préconditions_\beta, actions_\beta, postconditions_\beta)$
$\supset Ms\ plan(\alpha, préconditions_\alpha, actions_\alpha, postconditions_\alpha)$
$\wedge (\beta \in actions_\alpha)\ )$

Toute action β satisfaite pendant un dialogue d'acquisition appartient à la séquence pour faire la tâche α.

(**A.20**) $?\Delta\ [Uf\ \alpha] \wedge ?[Mf\ \beta] \wedge M\ \neg sf\ \beta \supset ?\Delta_{\Delta\ [Uf\ \alpha]}\ [Uf\ \beta\ ]$

Acquisition des sous-tâches.

4.3.3.3 Spécialisation d'une tâche

Après avoir demandé la réalisation d'une tâche, l'utilisateur peut la modifier ou la particulariser, et créer soit une nouvelle instance pour cette tâche, soit une toute nouvelle tâche qui reprend les actions de la tâche antérieure. Ce processus de spécialisation est intégré au dialogue à travers les axiomes suivants. En même temps, les axiomes de rectification et d'intégration d'effets de la rectification interviennent pour compléter le déroulement du dialogue (cf. § 4.3.4.1).

L'utilisateur a demandé une action qui tombe sur un élément d'une tâche déjà satisfaite. Cela est la condition initiale pour engager un dialogue de spécialisation. Le premier pas dans ce cas-là est la confirmation de l'intention de l'utilisateur à montrer une nouvelle instance de la tâche :

(**A.21**) $[\text{Uff } \alpha(X)] \wedge \text{Ms élément}(X, Y) \wedge {+}{+}\gamma(Y)$
$\supset ?\,[\text{Mf vérifier}(\text{ Ui } [\text{Uf } \gamma'(Y)](\text{Msf } \gamma')$
$\wedge \text{ spécialisation}(\gamma', \gamma)\,)), \text{Mf } \alpha(X)\,]$

Si la réponse est *affirmative*, la machine sait que :

(**A.22**) $\text{Ms } (\text{Ui } [\text{Uf } \gamma'(Y)](\text{Msf } \gamma')) \wedge \text{Ms spécialisation}(\gamma', \gamma)$
$= ?\Delta\,[\text{Uf } \gamma'(Y)] \wedge \text{Ms spécialisation}(\gamma', \gamma)$

et donc que l'utilisateur souhaite présenter une nouvelle instance de la tâche γ. Dans ce cas on rentre dans un dialogue de spécialisation pour la tâche γ, c'est-à-dire, que le résultat final sera une nouvelle instance γ' qui reprend les actions déjà faites pour la réalisation de γ et intègre les nouvelles rectifications apportées par l'utilisateur.

(**A.23**) $?[\text{Mf } \alpha(X)] \wedge \text{Ms élément}(X, Y) \wedge {+}{+}\gamma(Y)$
$\wedge\ ?\,\Delta[\text{Uf } \gamma'(Y)] \wedge \text{Ms spécialisation}(\gamma', \gamma)$
$\supset {+}\gamma(Y)$

On considère la tâche γ en rectification et en conséquence non satisfaite.

(**A.24**) $\forall\beta($ $(++_{\Delta\,[Uf\,\alpha']}\,[Mf\,\beta] \vee ++_{[Mf\,\alpha]}\,[Mf\,\beta])$
$\wedge$ Ms spécialisation(α', α) $\wedge$ $++\Delta\,[Uf\,\alpha']$
$\wedge$ Ms plan(β, préconditions$_\beta$, actions$_\beta$, postconditions$_\beta$)
$\supset$ Ms plan(α' , préconditions$_{\alpha'}$, actions$_{\alpha'}$, postconditions$_{\alpha'}$)
$\wedge$ ($\beta \in$ actions$_{\alpha'}$))

Après la rectification de la tâche initiale on obtient la nouvelle instance.

La phase finale de la spécialisation d'une tâche est la généralisation. Ce processus trouve les parties communes entre les deux instances et construit un plan général pour la tâche. Le plan général d'une tâche sera notamment utilisé pour la reconnaissance des intentions.

(**A.25**) Msf $\alpha \wedge$ Msf $\alpha' \wedge$ spécialisation(α', α)
$\supset$ Ms plan($\alpha_g/\{\alpha, \alpha'\}$, préconditions$_{\{\alpha, \alpha'\}}$, actions$_{\{\alpha, \alpha'\}}$,
postconditions$_{\{\alpha, \alpha'\}}$)

où $\alpha_g/\{\alpha, \alpha'\}$ est la généralisation des instances α y α'.

Si la réponse est *négative*, l'utilisateur souhaite montrer une nouvelle tâche δ et celle-ci n'est pas en relation de spécialisation avec la tâche antérieure γ. Néanmoins, les actions pour γ sont considérées dans la séquence pour δ. Pour cela on utilise la notation δ/γ :

Ms (Ui [Uf δ/γ](Msf δ)) $\wedge$ Ms $\neg$spécialisation(δ, γ)
= ?Δ [Uf δ/γ] $\wedge$ Ms $\neg$spécialisation(δ, γ)

Dans ce cas on entre dans un dialogue pour apprendre la nouvelle tâche δ qui intercale l'acquisition de nouvelles actions et la rectification des actions initiales empruntées à la réalisation de la tâche γ.

(**A.26**) ?[Mf α(X)] $\wedge$ Ms élément(X, Y) $\wedge$ $++\gamma$(Y)
$\wedge$? Δ[Uf δ/γ(Y)] $\wedge$ Ms $\neg$ spécialisation(δ, γ)
$\supset$ $+\gamma$(Y)

On considère la tâche γ en rectification et en conséquence non satisfaite.

(**A.27**) $\forall\beta$ ((++ $_{\Delta\,[\mathrm{Uf}\,\delta]}$Mf β $\vee$ ++ $_{[\mathrm{Mf}\,\gamma]}$Mf β)
$\wedge$ Ms $\neg$spécialisation(δ, γ) $\wedge$ ++Δ [Uf δ/γ] $\wedge$
$\wedge$ Ms plan(β, préconditions$_\beta$, actions$_\beta$, postconditions$_\beta$)
$\supset$ Ms plan(δ, préconditions$_\delta$, actions$_\delta$, postconditions$_\delta$)
$\wedge$ ($\beta \in$ actions$_\delta$))

La nouvelle tâche est une combinaison de nouvelles actions et d'actions de la tâche de base probablement rectifiées.

4.3.3.4 Reconnaissance de la tâche

Les actions faites par l'utilisateur servent à inférer son but final. Une bibliothèque de plans (vraisemblablement appris à travers plusieurs sessions de travail) est utilisée pour représenter des plans possibles. Ces descriptions sont la base de la *hiérarchie de plans*, composée de deux hiérarchies interconnectées : une hiérarchie *d'abstraction* et une autre de *décomposition*. La hiérarchie d'abstraction est créée à partir d'une relation «est-un». La hiérarchie de décomposition est établie à partir des préconditions et effets de chacun des plans.

En fonction de la hiérarchie de plans, un processus de reconnaissance cherche à établir le but de l'utilisateur. On retrouve un plan candidat quand ses actions sont *comparables* aux actions d'entrée faites par l'utilisateur. Dans le cas où un plan candidat est trouvé, la machine interrompt le déroulement normal du dialogue pour déterminer les intentions de l'utilisateur.

(**A.28**) Ms comparable([δ,δ'], actions$_\beta$) $\supset$? [Mf vérifier(Ui [Uf β])]

Si l'intention de l'utilisateur est bien le plan candidat en question, on confirme si l'utilisateur souhaite l'intervention de la machine.

(**A.29**) Ms Ui [Uf β] $\wedge$ Msf β $\supset$? [Mf vérifier (Ui [Mf β])]

Dans le cas où la machine intervient, la tâche est refaite complètement, c'est-à-dire qu'on élimine les actions faites par l'utilisateur et on refait la tâche. Un dialogue de rectification postérieur saisira la nouvelle instance si des modifications sont effectuées .

4.3.4 La gestion du dialogue

Dans la plupart des situations de dialogue homme-machine on distingue deux sortes de problèmes de dialogue [Luzzati 92] : la compréhension de buts de l'utilisateur, son plan, ses croyances, le contexte, etc., et la gestion du dialogue, la capacité du système de continuer le dialogue. Bien sûr, les deux problèmes sont imbriqués, une incompréhension sera résolue par un dialogue de clarification, ou un échange d'information peut entraîner une incohérence dans le système de croyances.

Dans les sections antérieures nous avons présenté les fondements de notre logique et abordé le problème de la compréhension des buts de l'utilisateur à travers des axiomes qui caractérisent la coopération de la machine dans la tâche poursuivie. Dans les sections suivantes on présente un nouvel ensemble d'axiomes, qui abordent la gestion du dialogue. Ces axiomes définissent la capacité de la machine à continuer le dialogue même en situation d'incompréhension.

La taxonomie suivante, d'après [Bilange 92], présente les différents sortes de problèmes liés au dialogue :

a) *Recouvrement et interruption.* C'est le cas où les deux interlocuteurs parlent en même temps.

b) *Rupture - réparation*. Le dialogue entre dans une situation d'échec : pour en sortir, il peut exister une auto-réparation ou une intervention du deuxième interlocuteur.

- *Ruptures en relation avec les aspects linguistiques (par exemple, le lapsus linguae)*
- *Ruptures liées au domaine (connaissances erronées, compétence du locuteur)*
- *Echec acoustique*

Cette classification est notamment orientée vers les difficultés rencontrées dans la partie parlée du dialogue et laisse de côté une série de problèmes propres au dialogue orienté vers la réalisation d'une tâche. C'est pourquoi, nous proposons une nouvelle taxonomie dont les objets principaux sont les

ruptures liées au domaine. Pour chaque situation on introduit brièvement la stratégie à suivre :

a) *Incompréhension*. Echec acoustique ou ruptures en relation avec les aspects linguistiques. Stratégie : informer de l'incompréhension et demander la répétition.

b) *Rupture - Réparation*. Le dialogue entre dans une série d'échanges pour rectifier le résultat de l'action demandée.

b.1) *Ruptures liées à la tâche*

- *Tâche inconnue*. La machine ne connaît pas la tâche demandée. *Stratégie :* acquisition de la tâche.
- *Rectification* ou réglage (*tunning*) de la tâche. *Stratégie :* rectifier la dernière action réalisée à partir de la nouvelle information précisée.
- *Reprise de l'action.* L'utilisateur reprend l'initiative d'une action précédemment déléguée. *Stratégie :* le but original est abandonné et l'intervention de l'utilisateur est considérée comme un nouveau but.
- *Annulation de l'action*. C'est le cas où l'utilisateur souhaite revenir en arrière et défaire l'action précédente. Dans cette situation on attend tout de suite une nouvelle commande ou l'information pour faire la correction. On peut alors utiliser comme heuristique le fait qu'après la négation il aura la correction et on laissera au planificateur la recherche de la nouvelle commande. *Stratégie :* le but original est abandonné et le nouveau but est construit d'après un processus de planification.

b.2) *Ruptures liées à la structure locale du dialogue*

- *Paires adjacentes*. Dans notre cas, pour la réalisation d'une tâche, l'utilisateur et la machine organisent leurs interventions en couples, au moins dans la plupart des cas (par exemple, à une question correspond une réponse). *Stratégie* : Si la condition de paires adjacentes ne s'accomplit pas, on entre dans un sous-dialogue de clarification qui cherche la confirmation d'abandon ou interruption momentanée du but posé actuel, ou l'adoption d'un nouveau but.

- *Elimination d'ambiguïtés.* Dans le processus de définition de l'acte de dialogue à partir de la phrase écrite ou prononcée, il est possible qu'un sous-dialogue soit déclenché pour enlever des ambiguïtés, par exemple un dialogue pour préciser le référent (on n'aborde pas ce type de cas, mais le formalisme proposé peut être utilisé sans problème).

Les paragraphes suivants décrivent les stratégies et les axiomes appliqués au deux difficultés les plus importantes en cas de rupture : la rectification de la tâche et les paires adjacentes.

4.3.4.1 Rectification de la tâche

Pour ce cas, le dialogue entre dans une série d'échanges pour rectifier ou modifier le résultat d'une action (par exemple, la position d'un objet). Dans cette situation apparaît clairement l'importance d'un cadre qui intègre l'action langagière et gestuelle dans un véritable dialogue.

Le cas général peut être résumé de la manière suivante :

(**A.30**) $\text{Ms Ui [Mf } \alpha]\varphi \wedge [\text{Mf } \alpha]\varphi \wedge \text{Ms Ui [Mf } \beta]\gamma \wedge \text{relation}(\alpha,\beta)$
$= +[\text{Mf } \alpha] \wedge ?_{[\text{Mf } \alpha]}[\text{Mf } \beta] \wedge \text{relation}(\alpha,\beta)$

la machine vient d'atteindre un but (*+[Mf α]*) et l'utilisateur a posé un nouveau but (*?[Mf β]*) qui modifie les effets du but précédent, c'est-à-dire qu'il existe une *relation* entre l'ensemble d'effets **φ** de l'action **α** et les préconditions de la nouvelle action **β**, ainsi le nouveau but est traité comme un sous-but du but précédent ($?_{[Mf\ \alpha]}[Mf\ \beta]$). Par conséquent, le but atteint n'est pas noté comme satisfait et celui-ci et le nouveau but posé seront associés dans une séquence de rectification. C'est-à-dire :

(**A.31**) $+[\text{Mf } \alpha\ ] \wedge +_{+[\text{Mf } \alpha\ ]}[\text{Mf } \beta\ ] \supset +[\text{Mf } \alpha\ ;\ \text{Mf } \beta]$

où le résultat est l'union des effets de **α** et **β** avec la prise en compte des rectifications apportées par **β** sur l'ensemble d'effets de **α**. Au moment où le dernier but en relation est considéré comme satisfait, la séquence de rectification sera aussi considérée comme satisfaite. Ainsi, quand la rectification est satisfaite, un processus de réduction est appliqué (cf. § 3.2).

(**A.32**) $[Mf\ \beta\ ;\ Mf\ \gamma\ ]\ \varphi/\phi \quad \rightarrow_{\text{réduction}}\ [Mf\ \delta\ ]\ \varphi/\phi$

La notation φ/ϕ est l'union des effets de β et γ avec la prise en compte des rectifications apportées par γ sur l'ensemble d'effets de β. Dans le processus de réduction on élimine les actions redondantes, en simplifiant la séquence d'actions mais en respectant les effets souhaités.

Il est aussi possible de commencer une rectification d'après une proposition à vérifier demandée par l'utilisateur :

(**A.33**) $[Uff\ vérifier(\varphi)](\ Ms\ Ui\ [Mf\ vérifier(\varphi)]\ )$
$= [Uff\ vérifier(\varphi)](\ ?\ [Mf\ vérifier(\varphi)]\ )$

Dans ce cas, si la machine ne peut pas faire la vérification de façon directe, un processus de planification intervient pour retrouver l'action ou la séquence d'actions à exécuter pour atteindre la situation souhaitée.

(**A.34**) $?[\ Mf\ vérifier\ \varphi] \wedge M \neg s\ \varphi \wedge Msf\ \beta \wedge \varphi \in effets_{\beta}$
$\supset\ ?_{[\ Mf\ vérifier\ \varphi]}[\ Mf\ \beta]$

Un troisième cas qui est à l'origine d'un dialogue de rectification est celui où l'utilisateur fait savoir une proposition. Le chemin à prendre dépend de la relation entre la connaissance partagée par l'utilisateur et la connaissance en machine par rapport à la tâche en cours. Si la connaissance partagée par l'utilisateur n'entre pas en contradiction avec la connaissance en machine, la machine accepte la connaissance et confirme son accord.

(**A.35**) $?Ms\ \phi \wedge \neg\ \exists\varphi\ (Ms\ \varphi \wedge (\varphi \supset \neg\ \phi\ ))$
$\supset\ [Mfs\ (Ms\ \phi)]\ (Ms\ Us\ Ms\ \phi)$

Par contre, si la connaissance en machine contredit l'information apportée par l'utilisateur, on suppose qu'il s'agit d'une action indirecte (par exemple, la machine vient de faire une action et l'utilisateur donne des informations qui la rectifient). La machine prend cette information comme un ordre à accomplir, un processus de planification cherche la nouvelle action qui rendra vraie l'information présentée.

(**A.36**) $?Ms\ \phi \wedge \neg\ \exists\varphi\ ((Ms\ \varphi \wedge (\varphi \supset \neg\ \phi\ )) \wedge Msf\ \beta \wedge \phi/\varphi\ \in effets_{\beta}\ ))$
$\supset\ ?\ _{[Ms\ \phi]}\ [Mf\ \beta]$

Bien sûr il est aussi possible de rectifier des actions déjà satisfaites. Cette *reprise de la rectification* d'une action dépend du contexte, et on distingue trois possibilités. La première, se situe quand la reprise se présente pendant l'acquisition d'une tâche. Dans ce cas-là, la rectification de la sous-tâche satisfaite est implicite, c'est-à-dire, on ne demande pas de confirmation pour rectifier l'action en question :

(**A.37**) $?[\text{Mf } \alpha(X)] \wedge {+}{+}[\text{Mf } \beta(X)] \wedge \text{Ms élément}(X, Y) \wedge ? \Delta[\text{Uf } \gamma(Z)] \supset +[\text{Mf } \beta(X)]$

l'action β est définie à nouveau comme atteinte et la rectification reprend normalement.

Les deux autres situations possibles sont la rectification d'une tâche satisfaite pour obtenir une spécialisation de cette tâche, et la création d'une toute nouvelle tâche à partir de la rectification/modification d'une autre tâche. Ces deux cas ont été déjà traités dans le § 4.3.3.3.

4.3.4.2 Paires adjacentes

On retrouve les conditions de paires adjacentes dans deux situations intéressantes : (i) le cas où la machine pose une question et en conséquence attend une réponse ; et (ii) quand l'utilisateur s'engage lui-même à faire une action. Il existe aussi un troisième cas –l'acquisition d'une tâche– quand la machine demande à l'utilisateur de faire une action. Cette dernière situation est déjà traitée dans des paragraphes précédents (cf. § 4.3.2.2 et § 4.3.3.2).

La machine pose une question. Le premier cas est celui où la machine fait une demande d'information. C'est l'ignorance de la machine qui est à l'origine de cette situation. La tâche demandée est connue par la machine mais il manque des données pour l'accomplir, ou bien la machine est incapable de vérifier une précondition de la tâche demandée. La tâche est alors différée jusqu'à l'obtention de l'information. Pour cela on ouvre un sous-dialogue incident fait d'une question posée à l'utilisateur :

$? [\text{ Mf vérifier}_{\text{ref(X)}} (\phi)] \wedge \text{M} \neg \text{s}_{\text{ref(X)}} (\phi) \supset [\text{Mffs}_{\text{ref(X)}} \phi](\text{Ms Us M}\neg\text{s}_{\text{ref(X)}} (\phi))$

après que l'utilisateur ait fait savoir la valeur –à travers un ou plusieurs tours de parole– la machine apprend la valeur, et la vérification peut être satisfaite :

$$? [\text{Mf vérifier}_{ref(X)} (\phi)] \wedge \text{Ms}_{ref(X)} (\phi) \supset ++ [\text{Mf vérifier}_{ref(X)} (\phi)]$$

Si l'utilisateur ne répond pas à la question posée, la machine la répétera puisque la vérification sera toujours insatisfaite.

En général, la machine pose la question puisqu'elle ne connaît pas la proposition ϕ :

(**A.38**) $? [\text{Mf vérifier}(\phi)] \wedge \text{M} \neg\text{s} (\phi) \supset [\text{Mffs } \phi](\text{Ms Us M}\neg\text{s} (\phi))$

demande de la valeur de vérité de ϕ

(**A.39**) $? [\text{Mf vérifier}_{ref(X)} (\phi)] \wedge \text{M} \neg\text{s}_{ref(X)} (\phi) \supset [\text{Mffs}_{ref(X)} \phi](\text{Ms Us M}\neg\text{s}_{ref(X)} (\phi))$

demande de la valeur du référent X

Le but est considéré comme atteint quand la machine connaît la proposition ϕ :

(**A.40**) $? [\text{Mf vérifier}(\phi)] \wedge (\text{Ms} (\phi) \vee \text{Ms} (\neg\phi)) \supset + [\text{Mf vérifier}(\phi)]$

(**A.41**) $? [\text{Mf vérifier}_{ref(X)} (\phi)] \wedge \text{Ms}_{ref(X)} (\phi) \supset + [\text{Mf vérifier}_{ref(X)} (\phi)]$

Par exemple, l'exécution de la tâche β entraîne la séquence suivante :

$$?\text{Mf } \beta \wedge \text{Ms plan}(\beta, \text{préconditions}_\beta, \text{actions}_\beta, \text{postconditions}_\beta) \supset ?_{[\text{Mf } \beta]} [\text{Mf vérifier (préconditions}_\beta) ; \text{Mf actions}_\beta ; \text{Mf vérifier (postconditions}_\beta)]$$

Lors de la vérification des préconditions de l'action, si la valeur d'un paramètre est inconnue, alors la machine pose la question :

$$?_{[\text{Mf } \beta]} [\text{Mf vérifier}_{ref(X)} (\text{condition}_\beta (X))] \wedge \text{M} \neg\text{s}_{ref(X)} \text{condition}_\beta(X) \supset ?_{[\text{Mf vérifier}]}[\text{Mffs}_{ref(X)} \text{condition}_\beta(X)] (\text{Ms Us M}\neg\text{s}_{ref (X)} \text{condition}_\beta(X))$$

Dans la situation où la paire question–réponse n'est pas accomplie, on entre dans une situation d'incompréhension qui est définie par l'impossibilité de vérification de la proposition demandée. Il est possible de profiter des connaissances sur l'utilisateur pour améliorer les actes de dialogue générés, par exemple, si la machine est en train de poser la question pour une deuxième fois, elle peut faire plus explicite sa demande :

$$?[\text{Mf vérifier } \varphi\,] \wedge M \neg s\ \varphi \wedge Ms\ (Us\ (M \neg s\ (\varphi))) \supset [Mfs\ M\neg s\ (\varphi)] \wedge [Mffs\ \varphi]\ Ms\ (Us\ (M \neg s\ (\varphi)))$$

De cette façon, la machine peut préciser l'incompréhension et accentuer le fait qu'elle attend une réponse.

L'utilisateur s'engage à faire une tâche. Ce cas découle directement de l'information présentée par l'utilisateur, ainsi l'acte *[Ufs (Ui [Uf α]φ)]* a pour résultat le but posé :

(**A.42**) $Ms\ (Ui\ [Uf\ \alpha]\varphi\ ) \stackrel{def}{=} ?[Uf\ \alpha]$

Dans cette situation la machine reste passive, en attente de l'intervention de l'utilisateur pour satisfaire le but posé :

(**A.43**) $?[\ Uf\ \alpha] \wedge [Mf\ \alpha]\ \varphi \stackrel{def}{=} ++[Uf\ \alpha]$

Enfin, on a un axiome pour rappeler à l'utilisateur qu'on attend qu'il fasse la tâche :

(**A.44**) $?[\ Uf\ \alpha\ ] \supset [Mfs\ (Ms\ Ui\ [Uf\ \alpha]\varphi\ )]$

4.4 Exemples

Dans cette section on présente un groupe d'exemples avec l'application de notre logique. Les exemples ont été empruntés du corpus recueilli par [Ozkan 94] à l'exception des exemples d'acquisition de la tâche et de reconnaissance qui sont des exemples simulés. Les exemples visent à expliquer l'application de la logique dans les différentes situations présentées auparavant. Pour cette raison on les a simplifiés, évidement un dialogue entier combinerait toutes ces situations.

4.4.1 Exécution d'une tâche

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*dessine une maison* [Uff créer_maison(obj4)]	 ?[Mf créer_maison(obj4)] *but posé (**A.5**)* Ms plan(créer_maison, ∅, [créer_carré ;créer_triangle], {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, noir), holds(T, forme(B, triangle)), holds(T, position(B, (40,70)), holds(T, taille(B, 50), holds(T, couleur(B, rouge) }) ?$_{[Mf\ créer\_maison]}$[Mf créer_carré(obj5), Mf créer_triangle(obj6), Mf vérifier (U¬i [Mf créer_maison(obj4)])] *exécution d'une tâche (**A.16**)*	FF
M	*geste d'exécution* [Mf créer_carré(obj5)]	+[Mf créer_carré(obj5)] Ms élément(obj5, obj4) *but atteint (**A.6**)* ? $_{[Mf\ créer\_maison]}$[Mf créer_triangle(obj6), Mf vérifier (U¬i [Mf créer_maison(obj4)])]] *but posé(**A.5**)*	F
M	*geste d'exécution* [Mf créer_triangle(obj6)]	++[Mf créer_carré(obj5)] *but satisfait (**A.7**)* +[Mf créer_triangle(obj6)] Ms élément(obj6, obj4) *but atteint (**A.6**)*	F
M	*voilà la maison, d'accord ?* [Mffs U¬i [Mf créer_maison(obj4)]]	? $_{[Mf\ créer\_maison]}$[Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*oui* [Ufs U¬i [Mf créer_maison(obj4)]]	+ $_{[Mf\ créer\_maison]}$[Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification atteinte (**A.40**)* ++[Mf créer_maison(obj4)] *but satisfait (**A.7**)*	FS

4.4.2 Acquisition d'une tâche

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*maintenant je je voudrais une maison* [Uff créer_maison(obj4)]	 ?[Mf créer_maison(obj4)] *but posé (**A.5**) et acquisition de la tâche(**A.18**)*	FF
M	*montre-moi* [Mff créer_maison(obj4)] (?Δ [Uf créer_maison(obj4)])	 ?Δ[Uf créer_maison(obj4)] *but déplace en acquisition (**A.10**)*	FF
U	*geste d'exécution* [Uf créer_carré(obj6)]	? $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_carré(obj6)] Ms élément(obj6,obj4) *but posé (**A.5**)* + $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_carré(obj6)] *but atteint (**A.6**)*	F
U	*geste d'exécution* [Uf créer_triangle(obj8)]	? $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_triangle (obj8)] Ms élément(obj8, obj4) *but posé (**A.5**)* ++ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_carré(obj6)] *but satisfait (**A.7**)* + $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_triangle (obj8)] *but atteint (**A.6**)*	F
U	*voilà, une maison* [Ufs U¬i [Uf créer_maison(obj4)]]	? [Ms U¬i [Uf créer_maison(obj4)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U¬i [Uf créer_maison(obj4)]] *connaissance acceptée (**A.14**)* *A partir de la nouvelle information apportée la machine peut inférer :* ++Δ[Uf créer_maison(obj4))] *but en acquisition satisfait (**A.11**)* ++ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Mf créer_triangle(obj8)] *but subordonné satisfait* Ms plan(créer_maison, ∅, [créer_carré ;créer_triangle], {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, noir), holds(T, forme(B, triangle)), holds(T, position(B, (40,70)), holds(T, taille(B, 50), holds(T, couleur(B, rouge) }) *plan acquis pour* créer_maison *(**A.19**)*	FS
M	*ok, l'ensemble est une maison* [Mfs Ms U¬i [Uf créer_maison(obj4)]]	*acceptation de l'information apportée (**A.35**)*	FS

4.4.3 Acquisition d'une tâche en nommant ses parties

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*dessine une maison* [Uff créer_maison(obj4)]	 ?[Mf créer_maison(obj4)] *but posé (**A.5**) et acquisition de la tâche(**A.18**)*	FF
M	*montre-moi* [Mff créer_maison(obj4)] (?Δ [Uf créer_maison(obj4)])	 ?Δ[Uf créer_maison(obj4)] *but déplacé en acquisition (**A.10***	FF
U	*ok, d'abord je vais dessiner les murs* [Ufs Ui ([Uf créer_murs(obj5)] Msf créer_murs(obj5)]	? [Ms Ui ([Uf créer_murs(obj5)] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui ([Uf créer_murs(obj5)] *connaissance acceptée (**A.14**)* *A partir de la nouvelle information apportée la machine peut inférer :* ?Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_murs(obj5))] *acquisition d'une sous-tâche (**A.20**)*	FS
M	*d'accord, montre-moi les murs* [Mfs Ms Ui ([Uf créer_murs(obj5)] Msf créer_murs(obj5)]	*acceptation de l'information apportée (**A.35**)*	FS
U	*geste d'exécution* [Uf créer_carré(obj6)]	? $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] Ms élément(obj6,obj5) *but posé (**A.5**)* + $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] *but atteint (**A.6**)*	F
U	*geste de déplacement* [Uf déplacer(obj6, (40,50)]	?$_{[\text{Mf créer\_carré(obj6)}]}$[Mf déplacer(obj6, (40,50)] *but posé, rectification (**A.30**)* +$_{[\text{Mf créer\_carré(obj6)}]}$[Mf déplacer(obj6, (40,50)] *but atteint (**A.6**)*	F
U	(1) *et ensuite* (2) *je vais dessiner le toit* (1) [Ufs U¬i [Uf créer_murs(obj5)]]	? [Ms U¬i [Uf créer_murs(obj5)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U¬i [Uf créer_murs(obj5)]] *connaissance acceptée (A.14)* *A partir de la nouvelle information apportée la machine peut inférer :* ++Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_murs(obj5)] *but en acquisition satisfait (**A.11**)* +$_{\Delta\ [\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6) ; Mf déplacer(obj6)] *séquence de rectification atteinte (**A.31**)* ++$_{\Delta\ [\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] *réduction appliquée (**A.32**)* Ms plan(créer_murs, ∅, [créer_carré], ∅)	FS

		{holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, rouge) }) *plan acquis pour* créer_murs ***(A.19)***	
	(2) [Ufs Ui ([Uf créer_toit(obj7)]]	? [Ms Ui [Uf créer_toit(obj7)]] *nouvelle connaissance à intégrer* ***(A.13)***	
		++ [Ms Ui [Uf créer_toit(obj7)]] *connaissance acceptée* ***(A.14)***	
		A partir de la nouvelle information apportée la machine peut inférer : ?Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_toit(obj7)] Ms élément(obj7, obj4) *acquisition d'une sous-tâche* ***(A.20)***	
M	*ok, on a fini les murs* [Mfs Ms U¬i [Uf créer_murs(obj5)]] *et après tu vais dessiner le toit* [Mfs Ms Ui [Uf créer_toit(obj7)]]	*acceptation de l'information apportée* ***(A.35)***	FS
U	*geste d'exécution* [Uf créer_triangle(obj8)]	? $_{\Delta\ [\text{Uf créer\_toit(obj7)}]}$ [Uf créer_triangle(obj8)] Ms élément (obj8, obj7) *but posé* ***(A.5)*** +$_{\Delta\ [\text{Uf créer\_toit(obj7)}]}$ [Uf créer_triangle(obj8)] *but atteint* ***(A.6)***	F
U	*voilà, la maison* [Ufs U¬i [Uf créer_maison(obj4)]]	? [Ms U¬i [Uf créer_maison(obj4)]] *nouvelle connaissance à intégrer* ***(A.13)*** ++ [Ms U¬i [Uf créer_maison(obj4)]] *connaissance acceptée* ***(A.14)*** *A partir de la nouvelle information apportée la machine peut inférer :* ++Δ[Uf créer_maison(obj4)] *but en acquisition satisfait* ***(A.11)*** ++Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_toit(obj7)] ++$_{\Delta\ [\text{Uf créer\_toit(obj7)}]}$ [Uf créer_triangle(obj8)] *buts subordonnés satisfaits* Ms plan(créer_toit, ∅, [créer_triangle], {holds(T, forme(A, triangle)), holds(T, position(A, (40,30)), holds(T, taille(A, 50), holds(T, couleur(A, noir) }) *plan acquis pour* créer_toit ***(A.19)*** Ms plan(créer_maison, ∅, [créer_murs, créer_toit], ∅) *plan acquis pour* créer_toit ***(A.19)***	FS
M	*d'accord, l'ensemble est une maison* [Mfs Ms U¬i [Uf créer_maison(obj4)]]	*acceptation de l'information apportée* ***(A.35)***	FS

4.4.4 Reprise implicite de la rectification dans l'acquisition d'une tâche

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*dessine une maison* [Uff créer_maison(obj4)]	?[Mf créer_maison(obj4)] *but posé (**A.5**) et acquisition de la tâche(**A.18**)*	FF
M	*montre-moi* [Mff créer_maison(obj4)] (?Δ [Uf créer_maison(obj4)])	?Δ[Uf créer_maison(obj4)] *but déplace en acquisition (**A.10***	FF
U	*ok, d'abord je vais dessiner les murs* [Ufs Ui ([Uf créer_murs(obj5)] Msf créer_murs(obj5)]	? [Ms Ui ([Uf créer_murs(obj5)] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui ([Uf créer_murs(obj5)] *connaissance acceptée (**A.14**)* *A partir de la nouvelle information apportée la machine peut inférer :* ?Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_murs(obj5))] *acquisition d'une sous-tâche (**A.20**)*	FS
M	*d'accord, montre-moi les murs* [Mfs Ms Ui ([Uf créer_murs(obj5)] Msf créer_murs(obj5)]	*acceptation de l'information apportée (**A.35**)*	FS
U	*geste d'exécution* [Uf créer_carré(obj6)]	? $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] Ms élément(obj6,obj5) *but posé (**A.5**)* + $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] *but atteint (**A.6**)*	F
U	*geste d'exécution* [Uf créer_cercle(obj61)]	++ $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_carré(obj6)] *but satisfait (**A.7**)* ? $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_ cercle (obj61)] Ms element(obj61,obj5) *but posé (**A.5**)* + $_{\Delta[\text{Uf créer\_murs(obj5)}]}$ [Mf créer_ cercle (obj61)] *but atteint (**A.6**)*	F
U	(1) *et ensuite* (2) *je vais dessiner le toit* (1) [Ufs U¬i [Uf créer_murs(obj5)]]	? [Ms U¬i [Uf créer_murs(obj5)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U¬i [Uf créer_murs(obj5)]] *connaissance acceptée (**A.14**)* *A partir de la nouvelle information apportée la machine peut inférer :* ++Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_murs(obj5)] *but en acquisition satisfait (**A.11**)* ++$_{\Delta\ [\text{Uf créer\_murs(obj5)}]}$ [Mf créer_cercle(obj61)] *but subordonné satisfait* Ms plan(créer_murs, ∅, [créer_carré, créer_cercle], ∅) {holds(T, forme(A, carré)),	FS

	(2) [Ufs Ui ([Uf créer_toit(obj7)]]	holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, rouge), holds(T, forme(B, cercle)), holds(T, position(B, (30,50)), holds(T, taille(B, 25), holds(T, couleur(B, noir)) }) *plan acquis pour* créer_murs ***(A.19)*** ? [Ms Ui [Uf créer_toit(obj7)]] *nouvelle connaissance à intégrer* ***(A.13)*** ++ [Ms Ui [Uf créer_toit(obj7)]] *connaissance acceptée* ***(A.14)*** *A partir de la nouvelle information apportée la machine peut inférer :* ?Δ $_{\Delta[\text{Uf créer\_maison(obj4)}]}$ [Uf créer_toit(obj7)] Ms élément(obj7, obj4) *acquisition d'une sous-tâche* ***(A.20)***	
M	*ok, on a fini les murs* [Mfs Ms U¬i [Uf créer_murs(obj5)]] *et après tu vais dessiner le toit* [Mfs Ms Ui [Uf créer_toit(obj7)]]	*acceptation de l'information apportée* ***(A.35)***	FS
U	*geste d'exécution* [Uf créer_triangle(obj8)]	? $_{\Delta\text{ [Uf créer\_toit(obj7)]}}$ [Uf créer_triangle(obj8)] Ms élément (obj8, obj7) *but posé* ***(A.5)*** +$_{\Delta\text{ [Uf créer\_toit(obj7)]}}$ [Uf créer_triangle(obj8)] *but atteint* ***(A.6)***	F
U	*geste d'exécution* *(on reprend la fenêtre pour la déplacer)* [Uf déplacer(obj61)]	++$_{\Delta\text{[Uf créer\_toit(obj7)]}}$ [Mf créer_triangle (obj8)] *but satisfait* ***(A.7)*** +$_{\Delta\text{[Uf créer\_murs(obj5)]}}$ [Mf créer_cercle(obj61)] *reprise implicite de la rectification* ***(A.37)*** ?Δ [Uf créer_murs(obj5))] *tâche à nouveau en acquisition* ?$_{\text{[Mf créer\_cercle(obj61)]}}$[Mf déplacer(obj61)] *but posé, rectification* ***(A.30)*** +$_{\text{[Mf créer\_cercle(obj61)]}}$[Mf déplacer(obj61)] *but atteint* ***(A.6)***	F
I	*voilà, la maison* [Ufs U¬i [Uf créer_maison(obj4)]]	? [Ms U¬i [Uf créer_maison(obj4)]] *nouvelle connaissance à intégrer* ***(A.13)*** ++ [Ms U¬i [Uf créer_maison(obj4)]] *connaissance acceptée* ***(A.14)*** *A partir de la nouvelle information apportée la machine peut inférer :* ++Δ[Uf créer_maison(obj4))] *but en acquisition satisfait* ***(A.11)*** ++Δ $_{\Delta\text{[Uf créer\_maison(obj4)]}}$ [Uf créer_murs(obj5)] *buts subordonnés satisfaits* +$_{\Delta\text{ [Uf créer\_murs(obj5)]}}$ [Mf créer_cercle(obj61) ; Mf déplacer(obj61)] *séquence de rectification atteinte* ***(A.31)***	FS

		$++_{\Delta\ [Uf\ créer\_murs(obj5)]}$ [Mf créer_cercle(obj61)] *réduction appliquée (**A.32**)* Ms plan(créer_murs, ∅, [créer_carré, créer_cercle], ∅) {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, rouge), holds(T, forme(B, cercle)), holds(T, position(B, (20,55)), holds(T, taille(B, 25), holds(T, couleur(B, noir)) }) *plan acquis pour* créer_murs *(**A.19**)* $++\Delta_{\Delta[Uf\ créer\_maison(obj4)]}$ [Uf créer_toit(obj7))] *buts subordonnés satisfaits* Ms plan(créer_toit, ∅, [créer_triangle], {holds(T, forme(A, triangle)), holds(T, position(A, (40,30)), holds(T, taille(A, 50), holds(T, couleur(A, noir)) }) *plan acquis pour* créer_toit *(**A.19**)* Ms plan(créer_maison, ∅, [créer_murs, créer_toit], ∅) *plan acquis pour* créer_toit *(**A.19*	
M	*ok, voilà une maison* Mfs Ms U¬i [Uf créer_maison(obj4)]]	*acceptation de l'information apportée (**A.35**)*	FS

4.4.5 Reprise explicite de la rectification : spécialisation d'une tâche

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*dessine une maison* [Uff créer_maison(obj4)]	?[Mf créer_maison(obj4)] *but posé (**A.5**)* Ms plan(créer_maison, ∅, [créer_murs, créer_toit], ∅) ? Mf créer_maison(obj4) [Mf créer_murs(obj5), Mf créer_toit(obj6)] Mf vérifier (U¬i [Mf créer_maison(obj4)])] *exécution d'une tâche(**A.16**)* Ms plan(créer_murs, ∅, [créer_carré, créer_cercle], {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, rouge), holds(T, forme(B, cercle)), holds(T, position(B, (30,50)), holds(T, taille(B, 25), holds(T, couleur(B, noir)) }) ? Mf créer_murs(obj5) [Mf créer_carré(obj7); Mf créer_cercle(obj8)] Mf vérifier (U¬i [Mf créer_murs(obj5)])] *exécution d'une tâche (**A.16**)*	FF
M	*geste d'exécution* [Mf créer_carré(obj7)]	+ Mf créer_murs(obj5) [Mf créer_carré(obj7)] *but atteint (**A.6**)* ? Mf créer_murs(obj5) [Mf créer_cercle(obj8), Mf vérifier (U¬i [Mf créer_murs(obj5)])] *continuation but posé (**A.5**)*	F
M	*geste d'exécution* [Mf créer_cercle(obj8)]	++ Mf créer_murs(obj5) [Mf créer_carré(obj7)] *but satisfait (**A.7**)* + Mf créer_murs(obj5) [Mf créer_ cercle(obj8)] *but atteint (**A.6**)*	F
M	*voilà les murs, d'accord ?* [Mffs U¬i [Mf créer_murs(obj5)]]	++ Mf créer_murs(obj5) [Mf créer_ cercle(obj8)] *but satisfait (**A.7**)* ?Mf créer_murs(obj5) [Mf vérifier (U¬i [Mf créer_murs(obj5)])] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*ok* [Ufs U¬i [Mf créer_murs(obj5)]]	+Mf créer_murs(obj5) [Mf vérifier (U¬i [Mf créer_murs(obj5)])] *vérification atteinte (**A.40**)* ++Mf créer_maison(obj4) [Mf créer_murs(obj5)] *but satisfait (**A.7**)* ? Mf créer_maison(obj4) [Mf créer_toit(obj6), Mf vérifier (U¬i [Mf créer_maison(obj4)])] *but posé (**A.5**)*	FS

		Ms plan(créer_toit, ∅, [créer_triangle], {holds(T, forme(A, triangle)), holds(T, position(A, (40,30)), holds(T, taille(A, 50), holds(T, couleur(A, noir)) }) ? $_{\text{Mf créer_toit(obj6)}}$ [Mf créer_triangle(obj9)] Mf vérifier (U¬i [Mf créer_toit(obj6)])] *exécution d'une tâche (**A.16**)*	
M	*geste d'exécution* [Mf créer_ triangle (obj9)]	+ $_{\text{Mf créer_toit(obj6)}}$ [Mf créer_ triangle (obj9)] *but atteint (**A.6**)*	F
M	*voilà le toit, d'accord ?* [Mffs U¬i [Mf créer_toit(obj6)]]	++ $_{\text{Mf créer_toit(obj6)}}$ [Mf créer_ triangle (obj9)] *but satisfait (**A.7**)* ?$_{\text{Mf créer_toit(obj6)}}$ [Mf vérifier (U¬i [Mf créer_toit(obj6)])] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*d'accord* [Ufs U¬i [Mf créer_toit(obj6)]]	+$_{\text{Mf créer_toit(obj6)}}$ [Mf vérifier (U¬i [Mf créer_toit(obj6)])] *vérification atteinte (**A.40**)* ++$_{\text{Mf créer_maison(obj4)}}$ [Mf créer_toit(obj6)] *but satisfait (**A.7**)*	FS
M	*ça y est, d'accord ?* [Mffs U¬i [Mf créer_maison(obj4)]]	? $_{\text{Mf créer_maison(obj4)}}$ [Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*(1) d'accord, (2) mais déplace le cercle à droite* *(1) d'accord* [Ufs U¬i [Mf créer_maison(obj4)]] *(2) mais déplace le cercle à droite* [Uff déplacer(obj8)]	+$_{\text{Mf créer_maison(obj4)}}$ [Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification atteinte (**A.40**)* ++[Mf créer_maison(obj4)] *but satisfait (**A.7**)* ?[Mf vérifier (Ui [Uf créer_murs'(obj5)] (Msf créer_murs') ∧ spécialisation(créer_murs', créer_murs)), Mf déplacer(obj8)] *confirmation de la spécialisation (**A.21**)*	FS, FF
M	*tu veux me montrer autre façon de faire les murs ?* [Mffs (Ui [Uf créer_murs'(obj5)] (Msf créer_murs') ∧ spécialisation(créer_murs', créer_murs)]	?[Mf vérifier (Ui [Uf créer_murs'(obj5)] (Msf créer_murs') ∧ spécialisation(créer_murs', créer_murs))] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*oui* [Ufs (Ui [Uf créer_murs'(obj5)] (Msf créer_murs') ∧ spécialisation(créer_murs', créer_murs)]	++[Mf vérifier (Ui [Uf créer_murs'(obj5)] (Msf créer_murs') ∧ spécialisation(créer_murs', créer_murs))] *vérification atteinte (**A.40**)* ? ∆[Uf créer_murs'(obj5)] Ms spécialisation(créer_murs', créer_murs) *acquisition nouvelle instance(**A.22**)* ? $_{\Delta\text{ [Uf créer_murs'(obj5)]}}$ [Mf déplacer(obj8)] + $_{\text{Mf créer_murs(obj5)}}$ [Mf créer_ cercle(obj8)] *reprise de la rectification pour spécialisation (**A.23**)*	FS
M	*geste d'exécution* [Mf déplacer(obj8)]	+ $_{\Delta\text{ [Uf créer_murs'(obj5)]}}$ [Mf déplacer(obj8)] *but atteint (**A.6**)*	F

		+ $_{\Delta\,[\text{Uf créer\_murs'(obj5)}]}$ [Mf créer_cercle(obj8), Mf déplacer(obj8)] *séquence de rectification atteinte (**A.31**)*	
U	*ok, voilà les nouveaux murs* [Ufs U¬i [Uf créer_murs'(obj5)]]	? [Ms U¬i [Uf créer_murs'(obj5)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U¬i [Uf créer_murs'(obj5)]] *connaissance acceptée (**A.14**)* ++Δ[Uf créer_murs'(obj5))] + $_{\Delta\,[\text{Uf créer\_murs'(obj5)}]}$ [Mf créer_cercle(obj8)] *réduction appliquée (**A.32**)* Ms plan(créer_murs', ∅, [créer_carré ;créer_cercle], {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, rouge), holds(T, forme(B, cercle)), holds(T, position(B, (30,60)), holds(T, taille(B, 25), holds(T, couleur(B, noir)) }) Ms plan(créer_murs$_g$/{créer_murs, créer_murs'}, préconditions$_g$, actions$_g$, postconditions$_g$) *généralisation du plan créer_murs (**A.25**)*	FS
M	*d'accord* Mfs Ms U¬i [Uf créer_murs'(obj5)]]	*acceptation de l'information apportée (**A.35**)*	FS

4.4.6 Reconnaissance d'une tâche

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*d'abord, dessine un carré* [Uff créer_carré(obj1)]	?[Mf créer_carré(obj1)] *but posé (**A.5**)*	FF
M	*geste d'exécution* [Mf créer_carré(obj1)]	+[Mf créer_carré(obj1)] *but atteint (**A.6**)*	F
U	*dessine un cercle* [Uff créer_triangle(obj2)]	++[Mf créer_carré(obj1)] *but satisfait (**A.7**)* ?[Mf créer_triangle(obj2)] *but posé (**A.5**)*	FF
M	*geste d'exécution* [Mf créer_triangle (obj2)]	+[Mf créer_triangle(obj2)] *but atteint (**A.6**)* Ms comparable([créer_carré(obj1), créer_ triangle(obj2)], créer_maison(obj4)) *actions comparables à la tâche* créer_maison *(**A.28**)* ? [Mf vérifier(Ui [Uf créer_maison(obj4)])] *confirmation intentions de l'utilisateur (**A.28**)*	F
M	*tu veux dessiner une maison ?* [Mffs (Ui [Uf créer_maison(obj4)])]	*vérification d'une proposition inconnue (**A.38**)*	FFS
U	*oui* [Ufs (Ui [Uf créer_maison(obj4)])	? [Ms (Ui [Uf créer_maison(obj4)])] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui [Uf créer_maison(obj4)]] *connaissance acceptée (**A.14**)* ? [Mf vérifier(Ui [Mf créer_maison(obj4)])] *confirmation intervention de la machine (**A.29**)*	FS
M	*je peux la faire, je peux essayer ?* [Mffs (Ui [Mf créer_maison(obj4)])]	*vérification d'une proposition inconnue (**A.38**)*	FFS
U	*d'accord* [Ufs (Ui [Mf créer_maison(obj4)])	? [Ms (Ui [Mf créer_maison(obj4)])] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui [Mf créer_maison(obj4)]] *connaissance acceptée (**A.14**)* ?[Mf détruire_carré(obj1), Mf détruire_triangle(obj2), Mf créer_maison(obj4)] *but posé (**A.5**)*	FS
M	*geste d'exécution* [Mf détruire_carré(obj1)]	+[Mf détruire_carré(obj1)] *but atteint (**A.6**)* ?[Mf détruire_triangle(obj2), Mf créer_maison(obj4)] *but posé(**A.5**)*	F
M	*geste d'exécution* [Mf détruire_triangle(obj2)]	+[Mf détruire_triangle(obj2),] *but atteint (**A.6**)* ?[Mf créer_maison(obj4)] *but posé (**A.5**)*	F

		Ms plan(créer_maison, ∅, [créer_carré ;créer_triangle], {holds(T, forme(A, carré)), holds(T, position(A, (40,50)), holds(T, taille(A, 50), holds(T, couleur(A, noir), holds(T, forme(B, triangle)), holds(T, position(B, (40,70)), holds(T, taille(B, 50), holds(T, couleur(B, rouge) }) ?$_{[Mf\ créer\_maison]}$[Mf créer_carré(obj5), Mf créer_triangle(obj6), Mf vérifier (U¬i [Mf créer_maison(obj4)])] *exécution d'une tâche (**A.16**)*	
M	*geste d'exécution* [Mf créer_carré(obj5)]	+[Mf créer_carré(obj5)] Ms élément(obj5, obj4) *but atteint (**A.6**)* ? $_{[Mf\ créer\_maison]}$[Mf créer_triangle(obj6), Mf vérifier (U¬i [Mf créer_maison(obj4)])]] *but posé(**A.5**)*	F
M	*geste d'exécution* [Mf créer_triangle(obj6)]	++[Mf créer_carré(obj5)] *but satisfait (**A.7**)* +[Mf créer_triangle(obj6)] Ms élément(obj6, obj4) *but atteint (**A.6**)*	F
M	*voilà la maison, d'accord ?* [Mffs U¬i [Mf créer_maison(obj4)]]	? $_{[Mf\ créer\_maison]}$[Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification d'une proposition inconnue (**A.38**)*	FFS
U	*oui* [Ufs U¬i [Mf créer_maison(obj4)]]	+ $_{[Mf\ créer\_maison]}$[Mf vérifier (U¬i [Mf créer_maison(obj4)])] *vérification atteinte (**A.40**)* ++[Mf créer_maison(obj4)] *but satisfait (**A.7**)*	FS

4.4.7 Rectification d'une action : vérification d'une condition

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*un rectangle à droite* [Uff créer_rectangle(obj1)] (position(obj1, (x1,y1)))	?[Mf créer_rectangle(obj1)] *but posé (**A.5**)*	FF
M	*geste d'exécution* [Mf créer_rectangle(obj1)] (position(obj1, (x1,y1)))	+[Mf créer_rectangle(obj1)] *but atteint (**A.6**)*	F
U	*tu le mets euh un peu plus bas un peu plus bas aussi* [Uff vérifier(position(obj1, (X, Y-inc)))] *Il faut noter que* ff vérifier *est plus fort que* fs*, le premier étant considéré un ordre –on souhaite arriver à une situation où cette condition est vraie–. Néanmoins, le* fs *peut être considéré comme un acte indirect.*	+[Mf créer_rectangle(obj1)] *but à rectifier (**A.30**)* ?$_{[Mf\ créer\_rectangle(obj1)]}$ [Mf vérifier(position(obj1, (X, Y-inc)))] *rectification posée (**A.33**)* ? $_{[Mf\ vérifier(position())]}$ [Mf déplacer(obj1, (x1,y1-inc))] *nouveau but subordonné résultant (**A.34**)*	FF
M	*geste de déplacement* [Mf déplacer(obj1, (x1,y1-inc)]	+$_{[Mf\ vérifier(position())]}$ [Mf déplacer(obj1, (x1,y1-inc))] *but atteint (**A.6**)* +$_{[Mf\ créer\_rectangle(obj1)]}$ [Mf vérifier(position(obj1, (X, Y-inc)))] *vérification atteinte (**A.40**)*	F
U	*ouais comme ça* [Ufs U ⊣i [Mf créer_rectangle(obj1)]]	++ [Mf créer_rectangle (obj1)] *but satisfait (**A.7**)*	FS

4.4.8 Rectification d'une action : nouvelle information présentée

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*Ensuite, dessine un petit cercle* [Uff créer_cercle(obj3)] (taille(obj3,25))	?[Mf créer_cercle(obj3)] *but posé (**A.5**)*	FF
M	*geste d'exécution* [Mf créer_cercle(obj3)] (taille(obj3,25))	+[Mf créer_cercle(obj3)] *but atteint (**A.6**)*	F
U	*et le mettre à gauche de la de la barre en haut* position(obj4,(x1,_)) ∧ [Ufs position(obj3, (x1-inc,_))]	? [Ms position(obj3, (x1-inc, _))] *nouvelle connaissance à intégrer (**A.13**)* ? $_{[Mf\ créer\_cercle(obj3)]}$ [Mf déplacer(obj3)] *connaissance en contradiction* *réaction pour éliminer la contradiction (**A.36**)* *but posé, rectification (**A.30**)*	FS
M	*geste de déplacement* [Mf déplacer(obj3)] position(obj3, (x1-inc,y1))	+ $_{[Mf\ créer\_cercle(obj3)]}$ [Mf déplacer(obj3)] *but atteint (**A.6**)* + [Mf créer_cercle(obj3), Mf déplacer(obj3)] *séquence de rectification atteinte (**A.31**)*	F
M	*d'accord, on change la position du cercle* [Mfs Ms position(obj3, (x1-inc,y1))]	*acceptation de l'information apportée(**A.35**)*	FS
U	*mais, qui touche la barre* position(obj4,(_,y2)) ∧ [Ufs position(obj3, (_,y2)))]	? [Ms position(obj3, (_,y2))] *nouvelle connaissance à intégrer (**A.13**)* ? $_{[Mf\ créer\_cercle(obj3)]}$ [Mf déplacer(obj3)] *connaissance en contradiction* *action pour éliminer la contradiction (**A.36**)* *but posé, rectification (**A.30**)*	FS
M	*geste de déplacement* [Mf déplacer(obj3)] position(obj3, (x1-inc,y2))	+ $_{[Mf\ créer\_cercle(obj3)]}$ [Mf déplacer(obj3)] *but atteint (**A.6**)* + [Mf créer_cercle(obj3), Mf déplacer(obj3) Mf déplacer(obj3)]] *séquence de rectification atteinte (**A.31**)*	F
M	*d'accord* [Mfs Ms position(obj3, (x1-inc,y2))]	*acceptation de l'information apportée(**A.35**)*	FS
U	*voilà, comme ça* [Ufs U ¬i [Mf créer_cercle(obj3)]]	? [Ms U ¬i [Mf créer_cercle(obj3)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U ¬i [Mf créer_cercle(obj3)]] *connaissance acceptée (**A.14**)* ++[Mf créer_cercle(obj3)] *but satisfait (**A.7**)* ++ [Mf créer_cercle(obj3), Mf déplacer(obj3) Mf déplacer(obj3)]] *but subordonné satisfait* ++ $_{[Mf\ créer\_cercle(obj3)]}$ [Mf créer_cercle(obj3)] *réduction appliquée (**A.32**)*	FS
M	*d'accord* [Mfs Ms U ¬i [Mf créer_cercle(obj3)]]	*acceptation de l'information apportée(**A.35**)*	FS

4.4.9 Rupture et reprise de l'action

Agent	Actes de dialogue	Etat de buts	Type d'acte
U	*alors un petit triangle* [Uff créer_triangle(obj4)] taille(obj4, x1)	?[Mf créer_triangle(obj4)] *but posé (**A.5**)*	FF
M	*geste d'exécution* [Mf créer_triangle(obj4)] taille(obj4, x1)	+[Mf créer_triangle(obj4)] *but atteint (**A.6**)*	F
U	*en haut hein* [Uff vérifier(position(obj4, (_,y1)))]	+[Mf créer_triangle(obj4)] *but à rectifier (**A.30**)* ?$_{[Mf\ créer\_triangle(obj1)]}$ [Mf vérifier(position(obj4, (_, y1)))] *rectification posée (**A.33**)* ? $_{[Mf\ vérifier(position())]}$ [Mf déplacer(obj4, (x1,y1))] *nouveau but subordonné résultant (**A.34**)*	FF
M	*geste de déplacement* [Mf déplacer(obj4, (x1,y1))]	+ $_{[Mf\ vérifier(position())]}$ [Mf déplacer(obj4, (x1,y1))] *but atteint (**A.6**)* +$_{[Mf\ créer\_triangle(obj4)]}$ [Mf vérifier(position(obj4, (_, y1)))] *vérification atteinte (**A.40**)*	F
U	*non pas comme ça* [Ufs Ui [M ¬f déplacer(obj4, (x1,y1))]]	? [Ms Ui [M ¬f déplacer(obj4, (x1,y1))]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui [M ¬f déplacer(obj4, (x1,y1))]] *connaissance acceptée (**A.14**)* @$_{[Mf\ créer\_triangle(obj4)]}$[Mf déplacer(obj4, (x1,y1))] *but abandonné (**A.8**)*	FS
M	*ok, j'arrête* [Mfs Ms Ui [M ¬f déplacer(obj4, (x1,y1))]]	*acceptation de l'information apportée(**A.35**)*	FS
U	*attends je vais le faire* [Ufs Ui [Uf déplacer(obj4, (x2,y2))]]	? [Ms Ui [Uf déplacer(obj4, (x2,y2))]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms Ui [Uf déplacer(obj4, (x2,y2))]] *connaissance acceptée (**A.14**)* ?$_{[Mf\ créer\_triangle(obj4)]}$[Uf déplacer(obj4, (x2,y2))] *but posé de l'utilisateur (**A.42**)* *rectification (**A.30**)*	FS
M	*ouais vas y* [Mfs Ms Ui [Uf déplacer(obj4, (x2,y2))]]	*acceptation de l'information apportée(**A.35**)*	FS
U	*création d'un carré* [Uf créer_carré(obj5)]	++[Mf créer_carré(obj5)] *but posé (**A.5**)* ++[Mf créer_carré(obj5)] *but satisfait (**A.7**)*	F
M	*ok, mais j'attends que tu déplaces le triangle* [Mfs Ms Ui [Uf déplacer(obj4, (x1,y1))]]	?$_{[Mf\ créer\_triangle(obj4)]}$[Uf déplacer(obj4, (x2,y2))] *rappel but engagé (**A.44**)*	FS
U	*geste de déplacement*	?$_{[Mf\ créer\_triangle(obj4)]}$[Mf déplacer(obj4, (x2,y2))]	F

	[Uf déplacer(obj4, (x2,y2))]	*but posé (**A.5**)* ++$_{[Mf\ créer\_triangle(obj4)]}$[Uf déplacer(obj4, (x2,y2))] *but satisfait (**A.7**)* + [Mf créer_triangle(obj4), Mf déplacer(obj4, (x2,y2))] *séquence de rectification atteinte (**A.31**)*	
U	*ouais, voilà comme ça* [Ufs U ¬i [Mf créer_triangle(obj4)]]	? [Ms U ¬i [Mf créer_triangle(obj4)]] *nouvelle connaissance à intégrer (**A.13**)* ++ [Ms U ¬i [Mf créer_triangle(obj4)]] *connaissance acceptée (**A.14**)* ++[Mf créer_triangle(obj4)] *but satisfait (**A.7**)* ++[Mf créer_triangle(obj4), Mf déplacer(obj4, (x2,y2))] *but subordonné satisfait (**A.7**)* ++ $_{[Mf\ créer\_cercle(obj3)]}$ [Mf créer_triangle(obj4)] *réduction appliquée (**A.32**)*	FS
M	*d'accord* [Mfs Ms U ¬i [Mf créer_triangle(obj4)]]	*acceptation de l'information apportée(**A.35**)*	FS

CHAPITRE V

Mise en œuvre informatique

Dans ce chapitre, nous présentons le système informatique qui met en œuvre notre modèle de dialogue coopératif. Pour illustrer notre recherche, nous avons opté pour une application de conception de dessin, ICPdraw, qui permet de dessiner à l'écran des objets quelconques, à partir d'une palette d'objets de base. Nous avons choisi cette application pour sa disponibilité et sa simplicité d'interaction. L'objectif de ce système informatique est alors de valider notre modèle sur une situation de conception dans un contexte multimodal. Nous commencerons par une description générale de ICPdraw, ensuite nous présenterons les processus spécifiques de notre système.

5.1 Description d'ICPdraw

ICPdraw [Wretö & Caelen 89] est une application de dessin (type MacDraw™) à manipulation directe, dans lequel la communication homme-machine est multimodale. L'utilisateur dispose d'une palette d'outils graphiques et de menus de fonctions. Il peut activer ces fonctions par la parole ou l'écriture ou le geste (manipulation de la souris). Le logiciel prévoit l'ensemble des fonctions habituelles nécessaires au dessin : sélection d'objets par pointage ou entourage ou désignation verbale, déplacement des objets par la voix ou par "dragage" avec la souris, changement des coloris, etc. La figure ci-dessous montre l'interface de l'application ICPdraw.

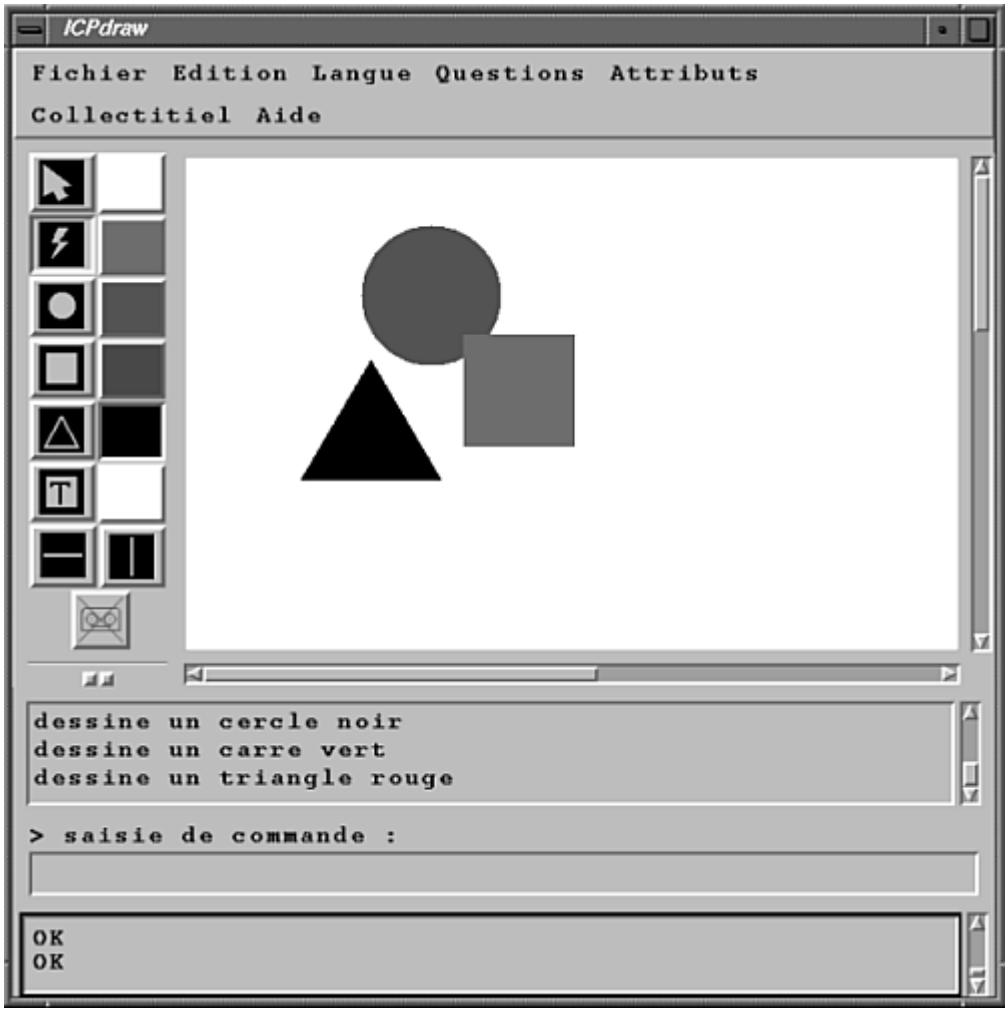


Figure 5.1. L'application ICPdraw

Il est composé d'une fenêtre découpée en deux zones. La première pour le travail graphique et la deuxième pour l'interaction écrite. Eventuellement, une autre fenêtre est utilisée pour visualiser les résultats de la compréhension de la parole.

Il est important de noter que ce travail s'insère dans le cadre d'étude de l'équipe GEOD au laboratoire CLIPS. Le travail de l'équipe englobe différents sujets pour aborder l'utilisation de la parole dans les interfaces homme-machine. Notre travail est centré dans la gestion du dialogue pour une tâche de conception. D'autres aspects, comme la reconnaissance de la parole spontanée et l'analyse linguistique, de même que la génération d'actes de langage, ont été abordés par d'autres membres de l'équipe (cf. [Colineau 97, Imberdis & Caelen 97, Kurdi 98]). Ainsi les modules de reconnaissance et de génération d'actes de langage ont été simplifiés intentionnellement dans notre système, pour offrir uniquement des fonctionnalités minimales requises.

5.2 Description générale de notre système

Notre système informatique est un système de DHM qui maintient une boucle d'interaction avec un utilisateur de ICPdraw. L'entrée du système est l'acte de dialogue : soit des actions directes non-langagières (par exemple une action avec la souris), soit des actions langagières (par exemple l'ordre « dessine une maison »). Cette interaction entre l'utilisateur et ICPdraw doit s'améliorer par l'usage. Le système acquiert de nouveaux dessins qui s'intègrent à son répertoire. De cette façon, ils peuvent être reproduits dans des sessions de travail ultérieures.

Le système est constitué de trois sous-systèmes : le sous-système de gestion du dialogue, le sous-système de formation de concepts et le sous-système de reconnaissance de la tâche (voir la figure 5.2). Le sous-système de gestion du dialogue, *le gestionnaire du dialogue*, est la partie principale du système. Il a en charge la convergence du dialogue. A chaque acte de dialogue provenant de l'utilisateur, il calcule son effet et construit la réponse. Le sous-système de formation de concepts a en charge l'apprentissage de nouvelles tâches. Après que l'utilisateur a exécuté une nouvelle tâche, le sous-système reçoit la

séquence d'actions associée. Celle-ci constitue alors un nouvel exemple de la tâche en question, et il est ajouté à l'ensemble d'exemples associé à la tâche. Un processus d'induction tente de retrouver l'information inhérente au concept à partir de ce nouvel ensemble. Le troisième sous-système a en charge la reconnaissance d'une tâche d'après l'observation des actions effectuées par l'utilisateur.

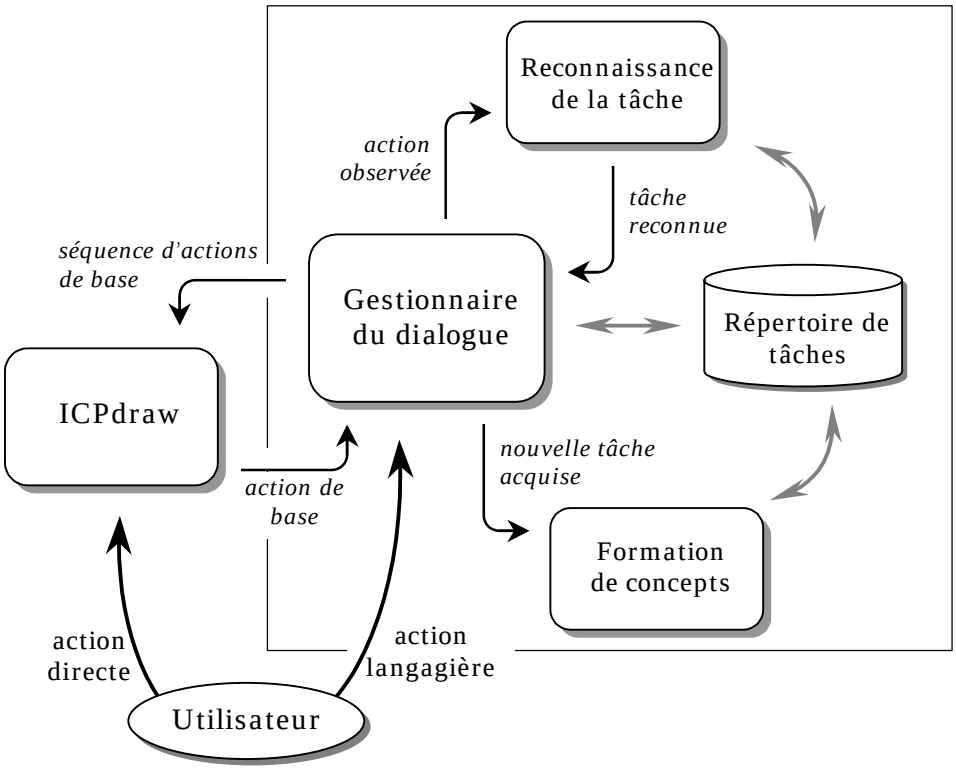


Figure 5.2. Modules du système

La figure 5.3 montre l'interface de notre système. L'utilisateur établit l'interaction à travers deux fenêtres. La première est l'aire de travail de ICPdraw, sur laquelle l'utilisateur voit le dessin en construction. A travers cette fenêtre, l'utilisateur peut donner des ordres directs en utilisant la souris et la palette des actions de base. Ou bien, il peut demander une action –de base ou composée– de manière langagière à travers une deuxième fenêtre liée directement au gestionnaire du dialogue. Dans cette fenêtre sont aussi présentées les différentes réponses ou questions du système. Eventuellement, on utilise, pour chaque sous-système, une fenêtre affichant les traces de ses inférences et ses résultats partiels.

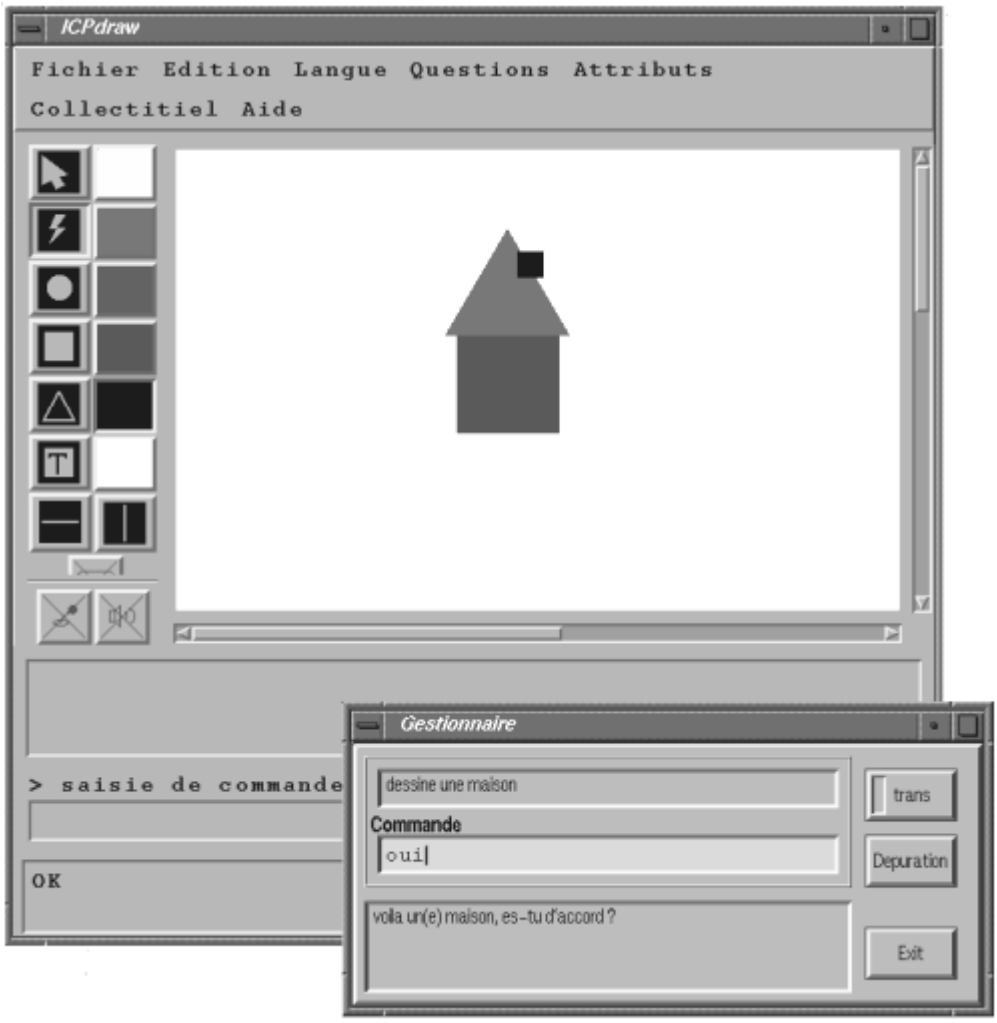


Figure 5.3. Interface de notre système

Le système est écrit en C++ et en Prolog[7] sous un environnement UNIX[8]. Le travail sous UNIX nous a permis de profiter d'un certain nombre de facilités de mise en œuvre non négligeables. Ainsi, la communication entre les différentes instances de ICPdraw, soit sur la même machine soit sur une machine distante, est réalisée à l'aide de *sockets* –le gestionnaire du dialogue est considéré comme une instance quelconque–. La communication des sous-systèmes de formation de concepts et de reconnaissance avec le gestionnaire est réalisée à travers des *pipes nommés*. Il s'agit alors de quatre processus. Le gestionnaire du dialogue, ICPdraw, le sous-système de formation de concepts et le sous-système de reconnaissance. Le gestionnaire et ICPdraw sont

[7] XSB Prolog version 1.8, XSB research group , SUNY at Stony Brook.
[8] sur une machine Silicon Graphics système IRIX 5.3

interdépendants et conservent une forte communication. Les sous-systèmes de formation de concepts et de reconnaissance reçoivent des informations et rendent leurs résultats au gestionnaire mais ils sont effectivement des processus indépendants. Chaque sous-système a son propre interprète Prolog. Cependant, le cas du gestionnaire est spécial, il est construit en C++ avec un interprète Prolog comme une classe de C++. Finalement, nous avons utilisé la version la plus récente du système Progol[9] (cf. § 3.4.2) dans le sous-système de formation de concepts, et la bibliothèque de fonctions graphiques XForms[10] pour simplifier la construction de fenêtres sous XWindows.

5.2.1 Représentation des actions

Un élément fondamental dans le système est l'action. Il existe deux classes : l'action de base et l'action composée ou tâche. Les actions de base sont des actions propres du monde de dessin : création d'un objet graphique, modification de sa taille ou de sa position, etc. Dans le contexte de ICPdraw les actions de base possibles sont la création, la destruction et la modification de propriétés des objets graphiques primitifs. Les objets graphiques primitifs sont le carré, le triangle, le cercle et le trait. Ainsi, un dessin complexe est réalisé à partir de ces actions de base. Une action de base est représentée comme un événement qui survient à un instant donné. Un événement réunit toute l'information pour décrire une action de base :

```
occurs( Temps,
   événement(  action(créer),  référence(Obj1),
               forme(cercle),  position(260,850),
               taille(65),  couleur(rouge)  ))
```

Comme nous l'avons vu, chaque action a un ensemble de préconditions et de postconditions qui permettent de l'enchaîner à une autre action. Ainsi, nous sommes capables de représenter des actions composées à travers de séquences d'actions de base –ou plans– (cf. § 3.2).

Un autre élément essentiel est l'acte de dialogue. Un acte de dialogue est une action qui a une visée communicationelle. Cette action a pour effet une

[9] Progol version 4.4, Machine Learning Group, University of York.
[10] XForms Library V0.88 © T.C. Zhao et M. Overmars

modification de la situation par rapport au dessin et/ou une modification de la situation du dialogue (cf. § 2.6.5 et § 4.2). Un acte de dialogue est une action, et sa représentation suit le schéma « préconditions, corps, postconditions ». La figure 5.4 montre le schéma d'un acte de dialogue sous la syntaxe Prolog utilisée.

```
%%  L'acte de dialogue :
%%     ( Precondition1 ET Precondition2 ) ∧
%%         [Uff Action] (Postcondition1 ET Postcondition2)
%%
%%  est transforme en syntaxe Prolog comme suit
%%      [uff Action] / ( Precondition1, Precondition2) -
%%                          ( Postcondition1, Postcondition2 )
```

Figure 5.4. L'acte de dialogue en Prolog

5.3 Processus d'interprétation d'actes de dialogue

Le gestionnaire du dialogue a en charge l'interprétation des actes de dialogue. Chaque intervention de l'utilisateur est composée d'un ou plusieurs actes de dialogue. L'interprétation correcte de ces actes dépend de l'état du dialogue. Celui-ci est géré à travers une pile de *buts*. Un but est l'action immédiate souhaitée. Lorsqu'un nouveau but est *posé*, il est rangé dans la pile. Si le but est *atteint*, alors il est marqué mais il reste dans la pile jusqu'à qu'il soit *satisfait*. Un but peut aussi être retiré de la pile s'il est *abandonné* (cf. § 4.3.2).

La boucle d'interprétation est divisée en trois pas. La figure 5.5 montre ces trois pas. Après la réception de l'acte de dialogue, le premier pas met en rapport l'acte reçu et les buts présents dans la pile. Il traite la confirmation implicite (s'il n'y a pas de relation entre le but atteint immédiat et l'acte de dialogue reçu alors on considère le but comme satisfait, cf. § 2.6.4 et § 4.3.2.1). Par contre, s'il existe une relation entre le but précédent et l'acte de dialogue reçu, il traite la demande de rectification. Un processus de réduction sera appliqué au moment où la rectification est finie (cf. § 4.3.4.1). Le deuxième pas de la boucle d'interprétation fait les déductions pertinentes d'après l'état de la pile et l'acte reçu. Pour faire ces déductions, le savoir et le savoir-faire de la machine entrent en jeu (cf. § 4.3.3). Par exemple, si l'utilisateur demande la réalisation d'une tâche et si la machine connaît un

plan pour la faire alors un nouveau but, composé de la séquence d'actions pour faire la tâche, est rangé dans la pile. Le dernier pas de la boucle d'interprétation réalise les actions calculées, autrement dit, ce pas transforme et exécute les actes de dialogue de la machine (cf. § 4.3.4.2).

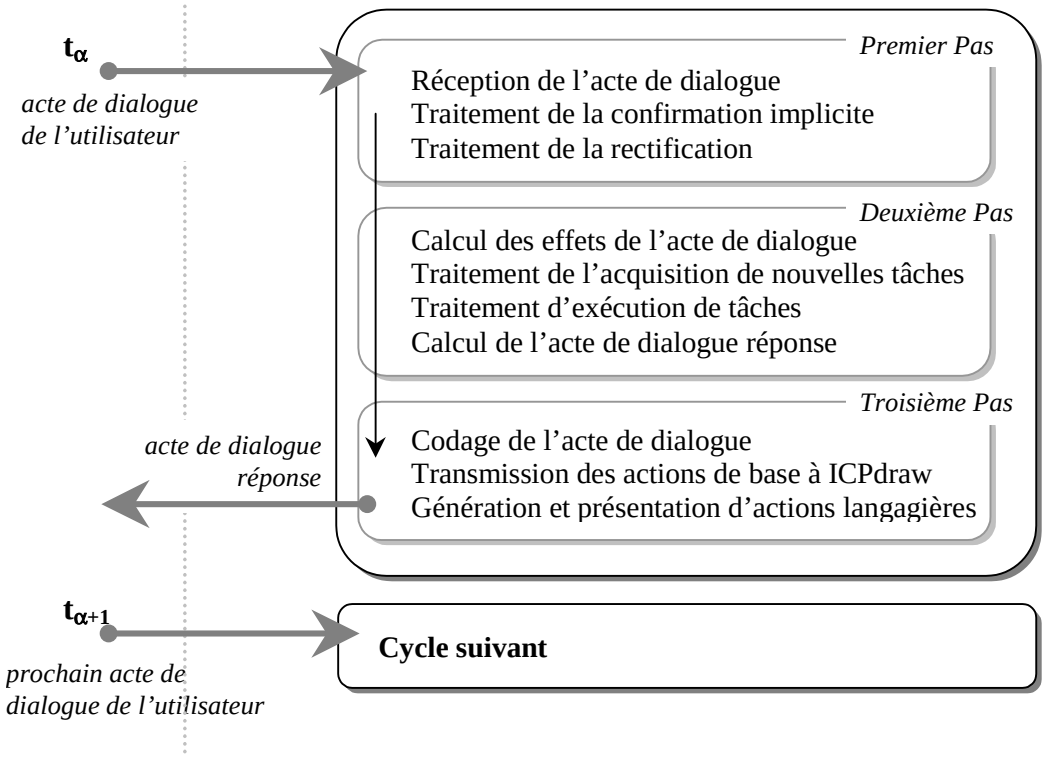


Figure 5.5. Processus d'interprétation d'un acte de dialogue

Les deux premiers pas dans la boucle d'interprétation sont implémentés à travers des ensembles de règles qui correspondent aux axiomes définis dans notre modèle. Dans le premier pas les règles se déclenchent à partir de conditions vérifiées sur l'état de la pile des buts et sur l'acte du dialogue reçu. Dans le deuxième pas les règles prennent en considération les conditions sur la pile de buts et sur le savoir et savoir-faire de la machine –soit pour apprendre une nouvelle tâche soit pour en reproduire une autre déjà connue–. Ce deuxième pas a aussi des règles qui considèrent le cas où la tâche en cours est reconnue. L'ensemble des différentes situations dialogiques abordées est présenté dans la section 4.4.

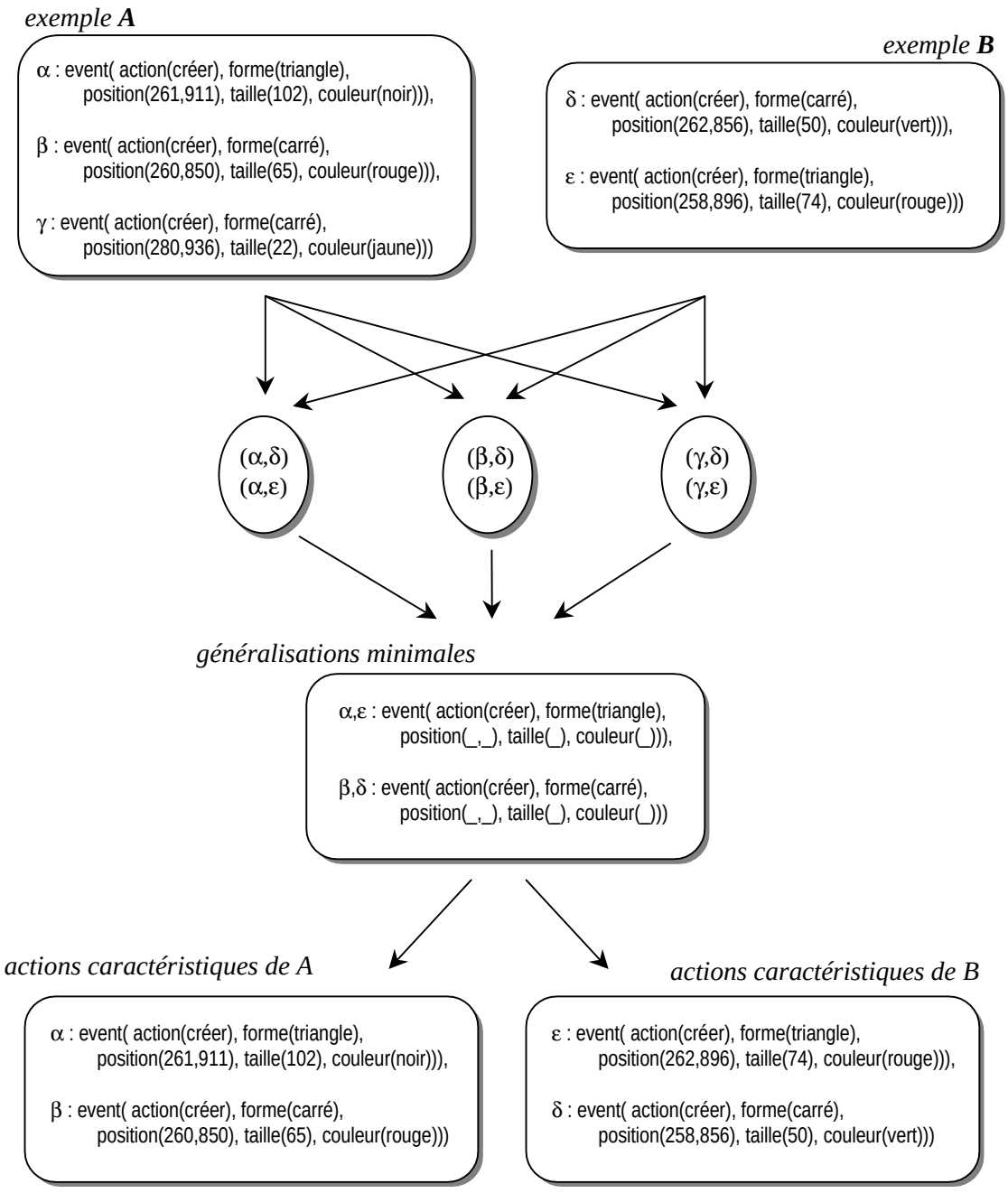


Figure 5.6. Le calcul des actions *caractéristiques*

5.4 Processus de formation de concepts

La formation de concepts est subordonnée au gestionnaire du dialogue. Après que le gestionnaire a acquis un nouvel exemple d'une tâche, il envoie cette description au processus de formation de concepts. La caractérisation d'un concept est réalisée en deux pas : le calcul des actions *caractéristiques* et la recherche des relations *descriptives* du concept (cf. § 3.4.3).

Le premier pas retrouve les actions caractéristiques à partir de l'ensemble d'exemples de la tâche. Cette opération est basée sur la méthode de la généralisation la plus spécifique, laquelle est appliquée à chaque paire possible d'actions entre deux exemples donnés. Ces généralisations résultantes sont évaluées pour arriver à obtenir les généralisations minimales, c'est-à-dire les généralisations avec un minimum de perte d'information. Les actions à l'origine des généralisations minimales sont considérées être les actions caractéristiques du concept. Par exemple, la figure 5.6 montre le calcul des actions caractéristiques entre deux séquences d'actions pour faire le dessin d'une maison.

Le deuxième pas tente d'établir les relations qui décrivent le concept, c'est-à-dire des relations vérifiées parmi les actions caractéristiques de chaque exemple de la tâche. Cet ensemble de relations descriptives est obtenu par le calcul de l'hypothèse la plus spécifique en utilisant Progol (cf. § 3.4.2). La construction de cette hypothèse présuppose une connaissance d'arrière-plan qui explicite les possibles relations entre deux actions. Par exemple nous avons défini des relations spatiales comme « au dessous de » ou « à droite de » qui sont établies par rapport aux positions des objets dessinés. La figure 5.7 montre les résultats de la recherche des relations descriptives pour les actions caractéristiques calculées par les exemples A et B.

Le concept obtenu est gardé dans le répertoire des tâches, de la même manière que l'ensemble d'exemples de la tâche. Si un nouvel exemple de la tâche est présenté, le processus de formation de concepts recommence, pour faire évoluer la description du concept.

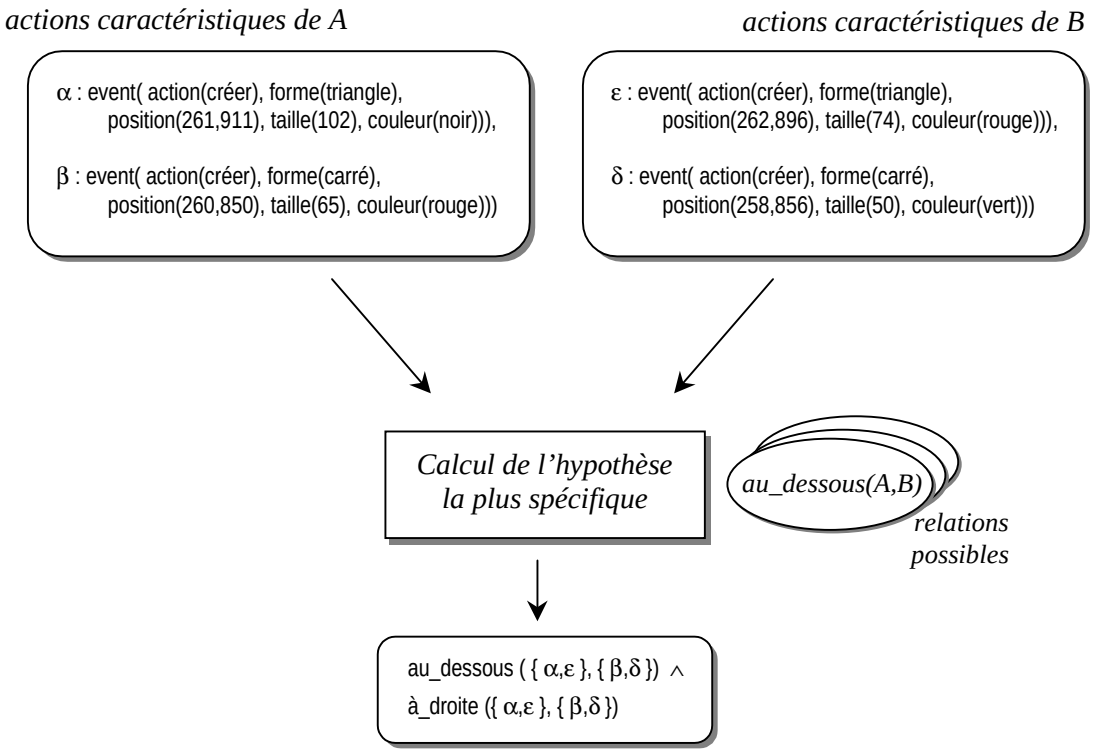


Figure 5.7. Calcul des relations descriptives

5.5 Processus de reconnaissance de la tâche

Le sous-système de reconnaissance de la tâche est aussi subordonné au gestionnaire du dialogue. Il reçoit du gestionnaire les actions observées réalisées par l'utilisateur. Le processus de reconnaissance se fait en deux phases : la construction de la description *récente* et la recherche *d'hypothèses* (cf. § 3.5).

La description récente est l'ensemble d'actions le plus simple possible qui décrit le dessin en cours. Après avoir intégré une action observée à la description récente, un processus de réduction élimine les actions intermédiaires redondantes. Si l'intégration d'une nouvelle observation n'affecte pas les observations précédentes, alors l'ensemble d'hypothèses calculé auparavant est utilisé pour la recherche des nouvelles hypothèses. Par contre, si l'intégration de la nouvelle observation entraîne des changements sur la description récente, alors la recherche d'hypothèses recommence d'après ce nouvel ensemble d'observations réduites.

La recherche d'hypothèses est un processus incrémental : pour chaque action observée le processus tente de trouver une ou plusieurs *explications*. Chaque nouvelle explication est associée aux hypothèses antérieures, pour former de nouvelles hypothèses, qui couvrent toute la séquence d'actions observées. Lorsque le processus de reconnaissance arrive à une seule hypothèse alors le système considère la tâche comme reconnue. La connaissance sur les tâches connues est représentée à travers une hiérarchie d'abstraction et composition. A partir de cette hiérarchie, le processus de reconnaissance trouve des explications à l'action observée. Une explication est représentée par un graphe *et/ou*. Une hypothèse est un graphe *et/ou* qui combine plusieurs explications. L'association d'une explication et d'une hypothèse doit vérifier les relations descriptives entre les actions concernées.

Supposons que la machine connaît les séquences pour faire les dessins d'un bus et d'une maison. Elle connaît au moins deux exemples de chaque tâche. La hiérarchie d'abstraction et composition pour ce cas-là est la suivante :

```
%% hiérarchie abstraction

bus(A) :> evenementFinal // [partie(1,A)/B, partie(2,A)/C, au_dessous(C,B),
              a_droite(C,B)].
maison(A) :> evenementFinal // [partie(1,A)/B, partie(2,A)/C, au_dessous(C,B)].

%% hiérarchie composition

bus(A) <: créer(carré, B) // [partie(1,A)/B, action(B)/créer, forme(B)/carré,
              position(B)/[C,D], taille(B)/E, couleur(B)/F].
bus(A) <: créer(cercle, B) // [partie(2,A)/B, action(B)/créer, forme(B)/cercle,
              position(B)/[C,840], taille(B)/D, couleur(B)/E].

maison(A) <: créer(triangle, B) // [partie(1,A)/B, action(B)/créer,
              forme(B)/triangle, position(B)/[C,D], taille(B)/E, couleur(B)/F].
maison(A) <: créer(carré, B) // [partie(2,A)/B, action(B)/créer, forme(B)/carré,
              position(B)/[C,D], taille(B)/E, couleur(B)/F].
```

Notation :
- **:>** relation est-un (hiérarchie d'abstraction)
- **<:** relation partie-de (hiérarchie de composition)

chaque nœud est représenté par le type de nœud et sa description
- type **//** description

où la description est formée d'un ensemble de rôles
- nom-du-rôle **/** valeur

Figure 5.8. Hiérarchies d'abstraction et composition pour les tâches « dessiner bus » et « dessiner maison »

D'après ces hiérarchies, le processus de construction d'une nouvelle hypothèse est montré dans la figure ci-dessous.

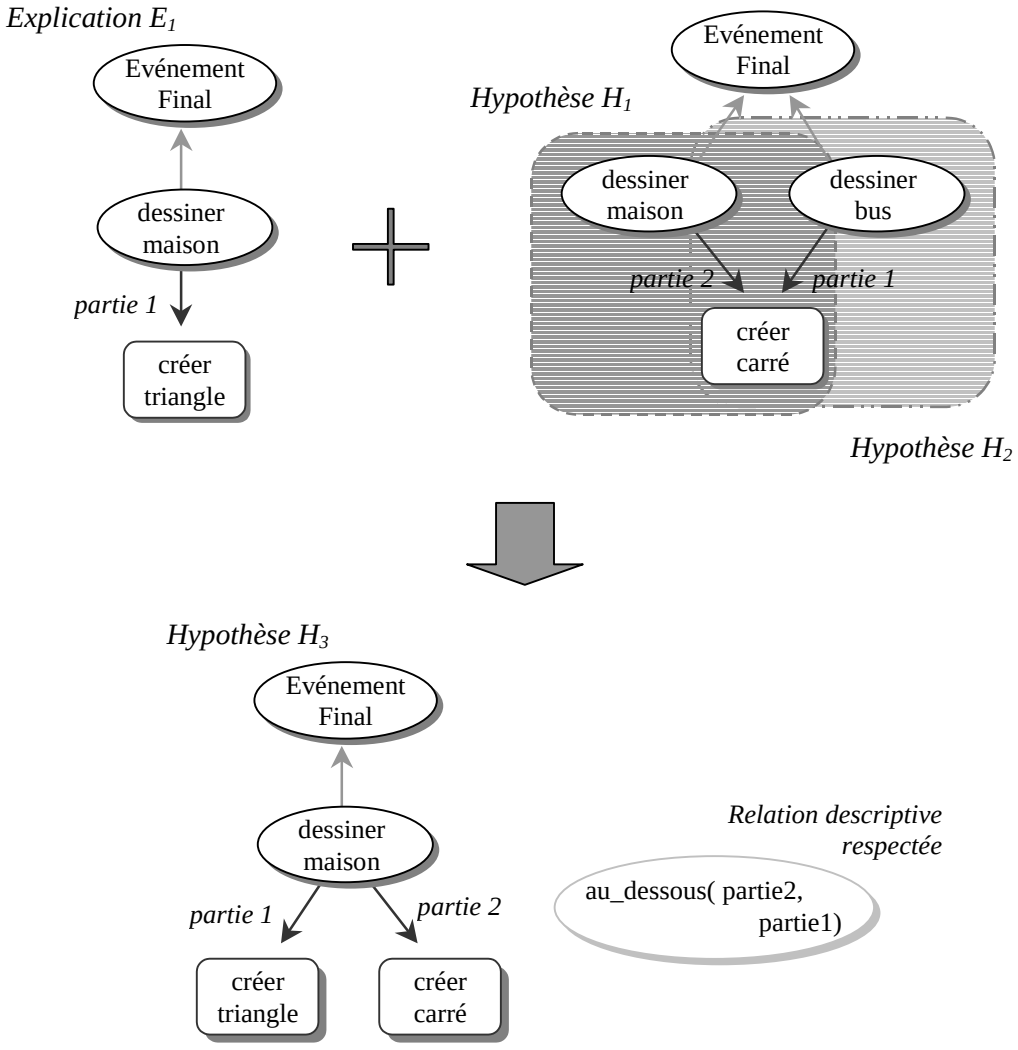


Figure 5.9. Construction d'une nouvelle hypothèse

Supposons qu'une première action observée est le dessin d'un carré. Donc, deux hypothèses sont possibles, H_1 et H_2. La seconde action observée est le dessin d'un triangle. Le système calcule une explication E_1 puis il associe cette explication aux hypothèses H_1 et H_2 pour arriver à la nouvelle hypothèse, H_3. L'association de l'explication E_1 et de H_1 a échoué car la séquence pour dessiner un bus ne comporte pas le dessin d'un triangle. L'association de l'explication E_1 et de H_2 est possible puisque le dessin d'une

maison implique le dessin d'un triangle et les relations descriptives sont respectées.

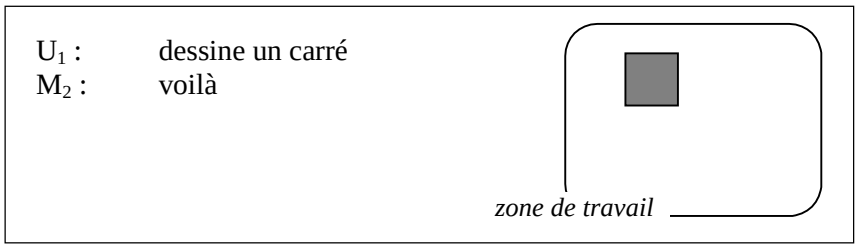
Bien sûr, le sous-système de formation de concepts est en charge de la construction d'une nouvelle hiérarchie chaque fois que le répertoire de tâches est mis à jour, soit pour l'acquisition d'une nouvelle tâche, soit pour la présentation d'un nouvel exemple d'une tâche déjà répertoriée.

5.6 Exemple

Pour illustrer le fonctionnement de notre système nous allons nous servir de l'exemple suivant. Son intérêt principal est de montrer l'intervention de la machine quand une tâche est reconnue. Nous supposons que l'utilisateur a présenté au système plusieurs exemples de la tâche « dessiner maison » et de la tâche « dessiner bus ». Ainsi, nous supposons une connaissance de départ telle que nous la retrouvons dans la hiérarchie de la figure 5.8.

U_1 : dessine un carré
M_2 : voilà
U_3 : <*avec la souris l'utilisateur dessine un triangle au-dessous du carré*>
M_4 : ok
U_5 : <*avec la souris l'utilisateur déplace le carré au-dessous du triangle*>
M_6 : ok,
es-tu en train de dessiner une maison ?
U_7 : oui
M_8 : ok, alors je vais la faire, es-tu d'accord ?
U_9 : oui
M_{10} : <*réalisation du dessin*>
voilà une maison,
es-tu d'accord ?
U_{11} : oui
M_{12} : d'accord
…

Ce dialogue est analysé pas à pas ci-après :



Interventions U_1, M_2. On suppose qu'il n'existe aucun autre but dans la pile. L'acte reçu provoque le rangement d'un nouveau but. Une solution pour le but posé est connue par la machine. Elle génère alors la réponse et considère le but comme atteint. La réponse est double : le dessin d'un carré et l'attestation langagière « voilà ».

- *Réception de l'acte de dialogue :*

```
[uff occurs(612999.0000,
          événement( action(créer), référence(_h384), forme(carré),
          position(260, 850), taille(65), couleur(rouge)))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 13 : l'utilisateur demande une tâche → but posé rangé dans la pile

règle 22 : solution du but posé connue (action de base) →
génération acte réponse, but atteint

```
[mf occurs(612999.0000, événement(action(créer), référence(_h878),
          forme(carré), position(260, 850), taille(65), couleur(rouge)))]
```

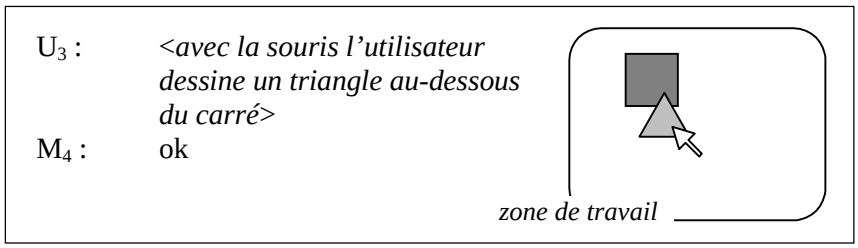
règle 32 : exécution de l'acte réponse →
transmission d'actions à ICPdraw et au sous-système de reconnaissance

- *Calcul d'hypothèses.*

Comme résultat du sous-système de reconnaissance, la liste d'hypothèses a deux éléments « dessiner bus » et « dessiner maison » :

```
([noeud( créer(carré, _h1547),
         bus(_h2147),
          [partie(1, _h2147)/_h1547, action(_h1547)/créer, forme(_h1547)/carré,
          position(_h1547)/[260, 850], taille(_h1547)/65, couleur(_h1547)/rouge]),
  noeud( bus(_h2147),
         événementFinal,
          [partie(1, _h2147)/_h1547, partie(2, _h2147)/_h2333,
          au_dessous(_h2333, _h1547), adroite(_h2333, _h1547)])
 ],

 [noeud( créer(carré, _h1547),
         maison(_h2201),
           [partie(1, _h2201)/_h1547, action(_h1547)/créer, forme(_h1547)/carré,
          position(_h1547)/[260, 850], taille(_h1547)/65, couleur(_h1547)/rouge]),
  noeud( maison(_h2201),
         événementFinal,
          [partie(1, _h2201)/_h1547, partie(2, _h2201)/_h2445,
          au_dessous(_h1547, _h2445)])
 ])
```



Interventions U_3 , M_4. Le gestionnaire du dialogue reçoit la description de l'action faite par l'utilisateur.

- *Réception de l'acte de dialogue :*

```
[uf occurs(730999.0000, événement(action(créer), référence(317),
          forme(triangle), position(264, 774), taille(50),
          couleur(vert)))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 11 : l'utilisateur a réalisé une action → but atteint rangé dans la pile

règle 32 : exécution de l'acte réponse →
transmission d'actions à ICPdraw et au sous-système de reconnaissance

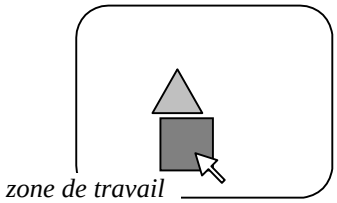
- *Calcul d'hypothèses.*

La liste d'hypothèses a trois possibilités : « dessiner bus » et « dessiner maison » pour la première observation et « dessiner maison » pour la deuxième. Bien que la séquence pour dessiner une maison comporte le dessin d'un carré et d'un triangle, le système de reconnaissance considère ces deux actions comme indépendantes puisque la relation descriptive n'est pas respectée :

```
([noeud( créer(carré, _h2233),
         bus(_h2328),
          [partie(1, _h2328)/_h2233, action(_h2233)/créer, forme(_h2233)/carré,
          position(_h2233)/[260, 850], taille(_h2233)/65, couleur(_h2233)/rouge]),
  noeud( bus(_h2328),
         événementFinal,
          [partie(1, _h2328)/_h2233, partie(2, _h2328)/_h2318,
          au_dessous(_h2318, _h2233), adroite(_h2318, _h2233)])
 ],
 [noeud( créer(carré, _h2233),
         maison(_h2240),
          [partie(1, _h2240)/_h2233, action(_h2233)/créer, forme(_h2233)/carré,
          position(_h2233)/[260, 850], taille(_h2233)/65, couleur(_h2233)/rouge]),
  noeud( maison(_h2240),
         événementFinal,
          [partie(1, _h2240)/_h2233, partie(2, _h2240)/_h2234,
          au_dessous(_h2233, _h2234)])
 ],
 [noeud(créer(triangle, 317),
        maison(_h2476),
          [partie(2, _h2476)/317, action(317)/créer, forme(317)/triangle,
          position(317)/[264, 774], taille(317)/50, couleur(317)/vert]),
  noeud(maison(_h2476),
        événementFinal,
          [partie(1, _h2476)/_h2597, partie(2, _h2476)/317,
          au_dessous(_h2597, 317)])
 ])
```

U_5 : *<avec la souris l'utilisateur déplace le carré au-dessous du triangle>*

M_6 : ok,
es-tu en train de dessiner une maison ?



Interventions U_5 , M_6. L'utilisateur déplace le carré. Le sous-système de reconnaissance établit une possibilité de but et un sous-dialogue d'assistance est déclenché.

- *Réception de l'acte de dialogue :*

```
[uf occurs(854999.0000, événement(action(modifier), référence(316),
          forme(carré), position(263, 727), taille(65), couleur(rouge)))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 11 : l'utilisateur a réalisé une action → but atteint rangé dans la pile

règle 32 : exécution de l'acte réponse →
transmission d'actions à ICPdraw et au sous-système de reconnaissance

règle 44 : conclusion à partir de nouvelles connaissances (tâche reconnue) →
but posé rangé dans la pile

```
?[mf occurs(A, événement( action(vérifier),
          holds(B, ?(uf occurs(1813998.0000,  événement(action(créer),
          référence(_h4668), forme(maison), position(260, 850),
          taille(65), couleur(rouge)))) ]
```

règle 22.1 : solution du but posé connue (action vérifier) →
génération acte réponse, but atteint

```
[mffs (?[uf occurs(1813998.0000, événement(action(créer),
          référence(_h4668), forme(maison), position(260, 850),
          taille(65), couleur(rouge))] )]
```

règle 34 : machine pose une question → transmission d'actions à ICPdraw

- *Calcul d'hypothèses.*

La nouvelle observation est en rapport avec une observation précédente ; alors le processus réduction est appliqué. A partir du nouvel ensemble d'observations, le système considère la tâche « dessine maison » comme reconnue.

```
([noeud( créer(triangle, 317),
         maison(_h4429),
          [partie(2, _h4429)/317, action(317)/créer, forme(317)/triangle,
          position(317)/[264, 774], taille(317)/50, couleur(317)/vert]),
  noeud( créer(carré, 316),
         maison(_h4429),
          [partie(1, _h4429)/316, action(316)/créer, forme(316)/carré,
          position(316)/[263, 727], taille(316)/65, couleur(316)/rouge]),
  noeud( maison(_h4429),
         événementFinal,
          [partie(1, _h4429)/316, partie(2, _h4429)/317, au_dessous(316,
          317)])
])
```

U_7 :	oui
M_8 :	ok, alors je vais la faire, es-tu d'accord ?

Interventions U_7 , M_8. L'utilisateur confirme qu'il est en train de faire une maison. Alors la machine demande la confirmation pour prendre en charge la tâche.

- *Réception de l'acte de dialogue :*

```
[ufs occurs(1265998.0000, événement(action(partager), holds(_h284, ?[uf
          occurs(1813998.0000, événement(action(créer), référence(_h267),
          forme(maison), position(260, 850), taille(65), couleur(rouge)))]
          )))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 12 : l'utilisateur fait savoir → but posé rangé dans la pile

règle 21.1 : solution du but posé connue (intégration de l'information) →
génération acte réponse, but atteint

règle 33 : exécution de l'acte réponse (confirmation acceptation de l'information) →
transmission d'actions à ICPdraw

règle 45 : conclusion à partir de nouvelles connaissances (prise en charge) →
but posé rangé dans la pile

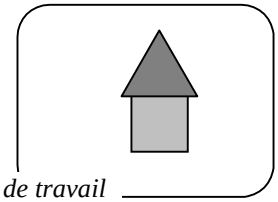
```
?[mf occurs(A, événement( action(vérifier),
          holds(B, ?[mf occurs(1813998.0000,  événement(action(créer),
          référence(_h4668), forme(maison), position(260, 850),
          taille(65), couleur(rouge)))])))  ]
```

règle 22.1 : solution du but posé connue (action vérifier) →
génération acte réponse, but atteint

```
[mffs (?[mf occurs(1813998.0000, événement(action(créer),
          référence(_h4668), forme(maison), position(260, 850),
          taille(65), couleur(rouge))) ])]
```

règle 34 : machine pose une question → transmission d'actions à ICPdraw

U_9 :	oui
M_{10} :	<*réalisation du dessin*> voilà une maison, es-tu d'accord ?



Interventions U_9 , M_{10}. L'utilisateur accepte l'intervention de la machine. Elle exécute la séquence d'actions du premier exemple de la tâche et demande l'avis de l'utilisateur.

- *Réception de l'acte de dialogue :*

```
[ufs occurs(1265998.0000, événement(action(partager), holds(_h284, ?[mf
          occurs(1813998.0000, événement(action(créer), référence(_h267),
          forme(maison), position(260, 850), taille(65), couleur(rouge)))]
          )))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 12 : l'utilisateur fait savoir → but posé rangé dans la pile

règle 21.1 : solution du but posé connue (intégration de l'information) →
génération acte réponse, but atteint

règle 33 : exécution de l'acte réponse (confirmation acceptation de l'information) →
transmission d'actions à ICPdraw

règle 45 : conclusion à partir de nouvelles connaissances (prise en charge) →
but posé rangé dans la pile

```
?[mf occurs(_h4665, événement(action(créer), référence(_h4668),
          forme(maison), position(260, 850), taille(65),
          couleur(rouge))))]
```

règle 23 : solution du but posé connue (action composée) →
but posé rangé dans la pile

```
?[mf occurs(_h887, événement(action(créer), _h891, forme(carré),
          position(283, 839), taille(50), couleur(vert))),
  mf occurs(_h910, événement(action(créer), _h914, forme(triangle),
          position(283, 877), taille(82), couleur(rouge))),
  mf occurs(_h933, événement(action(vérifier),
          holds(_h941, +[mf occurs(1813998.0000, événement(action(créer),
          référence(_h959), forme(maison), position(260, 850), taille(65),
          couleur(rouge)))] ]
```

règle 22 : solution du but posé connue (action de base) →
génération acte réponse, but atteint

```
[mf occurs(_h887, événement(action(créer), _h891, forme(carré),
          position(283, 839), taille(50), couleur(vert)))]
```

règle 32 : exécution de l'acte réponse →
transmission d'actions à ICPdraw et au sous-système de reconnaissance

règle 22 : solution du but posé connue (action de base) →
génération acte réponse, but atteint

```
[mf occurs(_h910, événement(action(créer), _h914, forme(triangle),
          position(283, 877), taille(82), couleur(rouge)))]
```

règle 32 : exécution de l'acte réponse →
transmission d'actions à ICPdraw et au sous-système de reconnaissance

règle 22.1 : solution du but posé connue (action vérifier) →
génération acte réponse, but atteint

```
[mffs (+[mf occurs(1813998.0000, événement(action(créer), référence(_h959),
          forme(maison), position(260, 850), taille(65), couleur(rouge))]]
```

règle 34 : machine pose une question → transmission d'actions à ICPdraw

U_{11} :	oui
M_{12} :	d'accord
…	

Interventions U_{11} , M_{12}. L'utilisateur est satisfait de la réalisation faite par la machine. Il accepte le dessin proposé par la machine.

- *Réception de l'acte de dialogue :*

```
[ufs occurs(1265998.0000, événement(action(partager), holds(_h284, +[mf
          occurs(1813998.0000, événement(action(créer), référence(_h267),
          forme(maison), position(260, 850), taille(65), couleur(rouge)))]
          )))]
```

- *Calcul des effets de l'acte de dialogue :*

règle 12 : l'utilisateur fait savoir → but posé rangé dans la pile

règle 21.1 : solution du but posé connue (intégration de l'information) → génération acte réponse, but atteint

règle 33 : exécution de l'acte réponse (confirmation acceptation de l'information) → transmission d'actions à ICPdraw

règle 42 : conclusion à partir de nouvelles connaissances (réalisation acceptée) → but satisfait

5.7 Premières observations et conclusions

Nous avons présenté les différents composants de notre système informatique. Le cadre d'application est simple, mais il est suffisamment riche pour nous permettre de valider les idées développées dans notre modèle de dialogue. Le système est opérationnel et il est capable d'établir un dialogue coopératif sur la base d'une tâche de dessin. L'utilisateur montre différents dessins et la machine acquiert et organise cette connaissance pour assister l'utilisateur dans des sessions de travail ultérieures.

Le travail réalisé, ne comporte pas d'évaluation systématique. Celle-ci est actuellement en cours. Nous en donnons ci-après un aperçu.

Parmi les premières observations, on note l'influence de l'ordre des tâches planifiées par l'utilisateur dans la formation d'un concept. Dans notre cas, la formation est empirique, elle procède de façon incrémentale et elle se réalise

uniquement sur des exemples positifs. Ainsi, la formation d'un concept est dépendante de l'ordre de présentation des exemples. Si l'utilisateur montre deux exemples qui sont proches l'un de l'autre (par le nombre d'actions ou le type d'actions), alors le concept résultant est plus approprié que dans le cas où les deux exemples sont plus éloignés. Il faut donc donner à l'utilisateur des moyens pour intervenir dans la construction du concept, notamment dans la constitution de l'ensemble d'actions caractéristiques.

Une autre constatation est le besoin de rendre disponible pour l'utilisateur les résultats du sous-système de reconnaissance. L'utilisateur doit pouvoir consulter les hypothèses calculées et celles-ci doivent être présentées de façon à provoquer la meilleure réaction possible de l'utilisateur. Cette présentation doit montrer les hypothèses possibles en cours, et mettre en évidence l'ensemble d'actions qui est à l'origine de chaque hypothèse. Il faut donner cette information à l'utilisateur pour faire intervenir la machine le plus rapidement possible, et ainsi éliminer les cas où la reconnaissance a besoin d'un nombre considérable d'observations pour aboutir à une hypothèse. C'est le cas de la reconnaissance entre tâches qui sont très proches, ou bien quand le nombre d'actions caractéristiques dans l'ensemble de tâches est élevé.

Enfin, il faut noter que pour la réalisation informatique nous avons adapté le modèle de dialogue proposé, mais en respectant les idées sur lesquelles il s'appuie. L'implémentation est un système de règles qui découle directement des axiomes de notre logique.

Une deuxième version du système envisage l'intégration des idées développées par d'autres membres de l'équipe pour l'amélioration de sa compétence linguistique. Cet apport aux modules de reconnaissance et de génération d'actes de dialogue permettra notamment d'élargir l'utilisation de notre modèle.

CHAPITRE VI

Conclusions et Perspectives

Ce travail s'est concentré sur un aspect du dialogue homme-machine, celui où la machine prend un rôle d'assistant pour coopérer à la réalisation d'une tâche. Pour aborder ce problème, nous avons proposé de donner à la machine des capacités pour interagir avec l'utilisateur et établir, en dialoguant, *quoi* faire et *comment* faire. Ainsi, compte tenu du caractère spécifique que nous voulions modéliser, nous avons développé –à partir d'une étude de différents modèles de dialogue homme-machine– un modèle de DHM finalisé basé sur la logique modale. Nous avons, en outre, adapté l'existant pour l'acquisition et la formation de concepts, ainsi que pour la reconnaissance de la tâche poursuivie par l'utilisateur. Ces capacités d'adaptation sont centrales pour l'incorporation de la coopération dans notre modèle de dialogue. Enfin, nous avons montré la faisabilité de nos idées en réalisant un système informatique.

La finalité de la recherche que nous avons engagée consiste à améliorer l'interaction entre l'homme et la machine dans un cadre actionnel, c'est-à-dire

pour des situations où la collaboration homme-machine est centrée vers l'exécution d'une séquence d'actions. La base de cette interaction est le dialogue. Pour améliorer cette interaction nous avons donné à la machine des capacités d'adaptation lui permettant d'assister l'utilisateur dans ce cadre actionnel. Nous avons essayé cette amélioration en proposant un modèle de dialogue coopératif qui n'est pas seulement orienté vers la co-construction des actions mais aussi vers l'apprentissage de nouvelles tâches.

Notre modèle de dialogue est basé sur la logique modale qui prend en compte la représentation des savoirs et des savoir-faire de la machine. Ces savoirs sont à la base de l'attitude coopérative de la machine. Pour aborder cette représentation, notre modèle intègre des aspects d'une logique épistémique et d'une logique de l'action. Pour rendre compte de l'aspect dialogique, notre modèle adapte les idées de [Caelen 95, Vernant 97] et considère le dialogue comme un échange d'actes de dialogue qui convergent pour la réalisation d'une tâche. Ainsi, un acte de dialogue est une action qui a un effet actionnel et un effet communicationnel. L'effet communicationnel a une incidence sur la convergence du dialogue vers le but instauré par l'utilisateur. Par ailleurs, l'effet actionnel a une incidence sur la tâche même. Le modèle proposé est capable de rendre compte de ces deux effets pour diriger le dialogue vers l'accomplissement de la tâche.

Bien que notre modèle ne couvre pas tous les phénomènes liés au dialogue homme-machine dans une situation de conception, il apporte des réponses dans cette voie. Il intègre, sur un même cadre actionnel, les interventions langagières et non-langagières. Il aborde le problème de la construction de connaissances communes à travers le dialogue, à la différence de beaucoup de systèmes de dialogue dont l'objectif est la recherche d'informations.

Parmi les phénomènes qui n'ont pas été abordés par notre travail nous avons :

- L'incompréhension. Il faut inclure des axiomes qui prennent en charge la résolution de l'incompréhension. Il est tout à fait possible de faire intervenir toute une série d'échanges dialogiques pour résoudre une ambiguïté due à une incompréhension.

- La négociation et l'argumentation. L'utilisateur doit pouvoir agir sur

le processus de formation de concepts. C'est lui qui doit valider le concept résultant et, s'il le faut, il doit intervenir pour le modifier. Donc des stratégies de négociation doivent être intégrées.

Notre travail a permis de contribuer à la vaste problématique de l'apprentissage. A partir d'une représentation adéquate des actions pour faire une tâche, nous avons défini un processus de réduction qui, en éliminant les actions redondantes d'un plan, arrive à un ensemble minimal d'actions sans altérer le résultat. Ce processus est utilisé au moment de l'acquisition d'une tâche comme dans la construction de la description récente pour la reconnaissance. C'est à partir de ces exemples normalisés que le processus de formation de concepts obtient des résultats. Pour cela, une nouvelle inspection des exemples extrait les actions caractéristiques, et une recherche sur ces actions caractéristiques trouve les relations inhérentes qui décrivent le concept.

Pourtant, comme nous l'avons vu, cet apprentissage est dépendant de l'utilisateur –sans que lui en soit vraiment conscient–, c'est pourquoi il faut lui donner les moyens pour intervenir dans le processus de formation d'un concept.

Enfin, pour la reconnaissance de la tâche nous avons mis en place un processus dynamique. Chaque nouvelle observation engendre une explication qui est couplée aux hypothèses des observations précédentes. La base des connaissances associée à la reconnaissance est mise à jour chaque fois que l'utilisateur présente de nouvelles tâches au cours des différentes sessions de travail. Cependant, dans une situation de conception, les intentions de l'utilisateur sont souvent imprécises et les buts émergent au cours de la réalisation de la tâche. Pour cela, d'après les premières observations que nous avons faites, le sous-système de reconnaissance a des problèmes pour aboutir rapidement à une réponse. Il faut donc mettre à disposition de l'utilisateur les résultats partiels de la reconnaissance de tâches pour déléguer, le plus tôt possible, le travail à la machine.

Le travail accompli dans notre thèse a donc montré la faisabilité d'un système coopératif : un système qui s'adapte grâce à l'interaction avec l'utilisateur, et qui a pour base d'interaction, le dialogue en langage naturel.

Références bibliographiques

[Andrès 95] Andrès, M. *Etude des stratégies de prise de décision d'une machine dans le contexte d'un dialogue enfant-machine. CEDRE Compère En Dialogue de Référence avec un Enfant*. Thèse de l'Université de Caen, 1995.

[Allen 84] Allen, J.F. Towards a general theory of action and time. *Artificial Intelligence* 23, pp 123-154, 1984.

[Allen & Perrault 80] Allen, J. et Perrault R. Analyzing intentions in utterances, *Artificial Intelligence* 15, pp 143-178, North-Holland Publishing Company. 1980.

[Austin 70] Austin, J.L. *Quand dire c'est faire*. Edition du Seuil, Paris, 1970.

[Bellalem & Romary 95] Bellalem, N. et Romary, L. *Langue et geste pour le dialogue homme-machine finalisé*. In [Caelen & Zreik 95].

[Bergadano & Gunetti 96] Bergadano, F. et Gunetti, D. *Inductive Logic Programming*. The MIT Press, 1996.

[Bilange 92] Bilange, E. *Dialogue personne-machine : modélisation et réalisation informatique*. Paris : Hermès, 1992.

[Bobrow et al. 77] Bobrow, D., Kaplan, R., Kay, M., Norman, D., Thompson, H. et Winograd, T. GUS, A Frame-Driven Dialog System. *Artificial Intelligence*, 8, pp 155-173. North-Holland Publishing Company. 1977.

[Bratko et al. 91] Bratko, I., Muggleton, S., Varsek, A. Learning Qualitative Models of Dynamics Systems. In L.A. Birnbaum et G.C. Collins (eds), *Proceedings of the Eighth International Workshop on Machine Learning* (ML91), CA, USA, 1991.

[Buntime 88] Buntime, W. Generalized Subsumption and Its Applications to Induction and Redundancy. *Artificial Intelligence*, Vol. 36. Elsevier Science Publishers. 1988.

[Caelen 95] Caelen, J. Vers une logique dialogique. *Séminaire International de Pragmatique*, Jérusalem, 1995.

[Caelen & Zreik 95] Caelen, J. et Zreik, K.(eds.) *Le Communicationnel pour Concevoir*. Europia Productions, 1995.

[Caelen & Villaseñor 97] Caelen, J. et Villaseñor, L. Dialogue Homme-Machine et Apprentissage. In *Apprentissage par l'interaction.* K. Zreik, (ed.) pp 83-117. Europia Productions. Paris, 1997.

[Carbonell 89] J.G. Carbonell. Introduction : paradigms for Machine Learning. *Special volume on Machine Learning. Artificial Intelligence*, Vol. 40. Elsevier Science Publishers. 1989.

[Carbonell et al. 83] Carbonell, J.G., Michalski, R.S. et Mitchell, T.M. *An Overview of Machine Learning*. In [Michalski et al., 83].

[Carbonell 86] Carbonell, J.G. Derivational analogy. A theory of reconstructive problem solving and expertise acquisition, *in Machine Learning : an Artificial Intelligence Approach*, vol. 2, Michalski, R.S., Carbonell, J.G. et Mitchell, T.M. eds., 1986.

[Chevallier 92] Chevallier, R. *Mise en œuvre d'un modèle dynamique de dialogue dans un Tuteur Intelligent*, Thèse de l'Université du Mans. 1992.

[Clark & Wilkes-Gibbs 86] Clark, H. H. et Wilkes-Gibbs, D. Referring as a collaborative process. *Cognition*, 1986, Vol. 22, pp 1-39, 1986. Réimprimé dans *Intentions in Communication*. P.R. Cohen, J. Morgan et M. Pollack (eds.). MIT Press. 1990.

[Cohen & Perrault 79]Cohen, P. R. et Perrault, C. R. Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, 3(3) : 177-212, 1979.

[Cohen & Levesque 84] Cohen, P. R. et Levesque, H. J. Speech Acts and Rationality. *10th International Conference on Computational Linguistics*. Coling84, 2-6 Juillet 1984, Stanford University, California, EU. 1984.

[Cohen & Levesque 90a] Cohen, P. R. et Levesque, H.J. Persistence, Intention and Commitment. In *Intentions in Communication.* P. R. Cohen, J. Morgan et M. E. Pollack. (eds.) The MIT Press. 1990.

[Cohen & Levesque 90b] Cohen, P. R. et Levesque, H. J. Rational interaction as the basis for communication. In *Intentions in Communication*. P.R. Cohen, J. Morgan et M. Pollack. (eds.) MIT Press. 1990.

[Colineau 97] Colineau, N. *Etude des marqueurs discursifs dans le dialogue finalisé.* Thèse en Sciences Cognitives, Université Joseph Fourier, Grenoble, 1997.

[Cypher 91] Cypher, A. EAGER : Programming repetitive tasks by exemple. *Proceedings of CHI 91*. ACM. 1991.

[De Raedt 92] De Raedt, L. *Interactive Theory Revision. An Inductive Logic Programming Approach*. Academic Press. 1992.

[Dean & McDermott 87] Dean, T.L. et McDermott, D.V. Temporal Database Management. *Artificial Intelligence* 32 : 1-55, 1987.

[Dean & Wellman 90]Dean T.L et Wellman, M.P. *Planning and Control*. Morgan Kaufan Publishers, 1990.

[Dent et al. 92] Dent, L., Boticario, J., McDermott, J., Mitchell, T., et Zaborowski, D. A Personal Learning Apprentice. *Proceedings of the 10th Natioinal Conference on Artificial Intelligence*. San Jose,CA. AAAI Press. 1992.

[Deshayes 98]Deshayes, P. Peut-on produire des modèles d'opération de la conception ? Perspectives et enjeux. *Forum des Arts de l'Univers Scientifique et Technologique, VIIèmes Rencontres.* Toulouse, octobre 1998.

[Fikes & Nilsson 71] Fikes, R. E. et Nilsson, N. J. STRIPS : A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*. 2, pp 189-208. North-Holland Publishing Company. 1971.

[Furukawa et al. 97] Furukawa, K., Murakami, T., Ueno, K., Ozaki, T. et Shimazu, K. On a Sufficient Condition for the Existence of Most Specific Hypothesis in Progol. *Proceedings of the 7th International Workshop en Inductive Logic Programming ILP-97*, Prague Czech Republic. Nada Lavrac et Saso Dzeroski (eds.). LNAI ; 1297. Springer.

[Ghiglione & Trognon 93] Ghiglione, R. et Trognon, A. *Où va la pragmatique ? de la pragmatique à la psychologie sociale*. Presses Universitaires de Grenoble. 1993.

[Goodman & Litman 92] Goodman, B. et Litman, D. On the Interaction between Plan Recognition and Intelligent Interfaces. *User Modeling and User-Adapted Interactions*. Vol 2; pp 83-115, Kluwer Academic Publishers. 1992.

[Grau et al. 94] Grau B., Sabah, G. et Vilnat A. Control in Man-Machine Dialogue, *THINK-5, special issue on man-machine interfaces*,Vol. 3, pp 32-55. 1994.

[Grosz et al. 89] Grosz, B., Pollack, M. et Sidner, C. Discourse. In *Foundations of Cognitive Science* M. Posner. (ed.) pp 437-468. MIT Press, 1989.

[Grosz & Kraus 96] Grosz, B. J. et Kraus, S. 1996. Collaborative Plans for Complex Group Action. *Artificial Intelligence.* 86 (1996) 269-357. Elsevier Science 1996.

[Grosz & Sidner 90] Grosz, B. J. et Sidner, C. L. Plans for discourse. In *Intentions in Communication*. P.R. Cohen, J. Morgan et M. Pollack. (eds.) MIT Press. 1990. pp 417-444.

[Halpern & Moses 92]Halpern, J.Y. et Moses, Y. A guide to completeness and complexity for modal logics og knowledge and belief. *Artificial Inteligence* 54, 319-379. Elsevier Science Publishers. 1992.

[Heeman & Hirst 95] Heeman, P.A. and Hirst, G. Collaborating on Referring Expressions. *Computational linguistics.* 1995, Vol. 21, No. 3, pp 351- 382, MIT Press.

[Helander et al. 97] Helander, M.G., Landauer, T.K. et Prabhu, P.V.(eds.) *Handbook of Human-Compuer Interaction*. *Second Edition*. Elsevier, 1997.

[Hoek et al. 94a] van der Hoek, W., van Linder, B. et Meyer, Ch. J.-J. A Logic of Capabilities (extended abstract). *Proceedings of the Third International Symposium of Logic Foundations of Computer Science. LFCS, 94.* A. Nerode et Yu. V. Matiyasevich. (eds.) St. Petersburg, Russie, julliet 11-14, Springer-Verlag, 1994.

[Hoek et al. 94b] van der Hoek, W., van Linder, B. et Meyer, Ch. J.-J. 1994. Unraveling Nondeterminism : On having the Ability to Choose (extended abstract). *Proceedings of the sixth International Conference on Artificial Intelligence : Methodology, Systems, Applications AIMSA'94.* P. Jorrand et V. Sgurev. (eds.) Sofia, Bulgarie, septembre 21-24, 1994. World Scientific. 1994.

[Imberdis & Caelen 97] Imberdis, L et Caelen, J. Génération d'actes illocutoires pour le dialogue. *Génération Automatique de Textes (GAT`97),* Grenoble, pp181-196. 1997.

[Karsenty & Pachet 95] Karsenty-Aflalo, S. et Pachet F. *Un Mécanisme Hiérarchique de Répétition et Prédiction de Tâches.* Septièmes journées sur l'ingénierie de l'Interaction Homme-Machine. Cépaduès Editions. 1995.

[Kautz 87] Kautz, H.A. *A Formal Theory of Plan Recognition*. Technical Report 215, University of Rochester, NY, 1987.

[Kautz 90] Kautz, H. A circumscriptive theory of plan recognition. In *Intentions in Communication*. P.R. Cohen, J. Morgan et M. Pollack. (eds.) MIT Press. 1990.

[Krause 93] Krause, J. *A Multilayered Empirical Approach to Multimodality : Towards Mixed Solutions of Natural Language and Graphical Interfaces.* In [Maybury 93].

[Kurdi 98] Kurdi, M. Z. *Reconnaissance de la parole par concepts : Vers un analyseur robuste des dialogues oraux spontanés*. Mémoire DEA en Science Cognitives. 1998.

[Laird et al. 86] Laird, J.F., Newell, A. et Rosenbloom, P.S. *Soar : An Architecture for General Intelligence*. Report STAN CS 86-1140, Department of Computer Science, Stanford University. 1986.

[Lang 95] Lang, K. NewsWeeder : Learning to filter news. *Proceeding of the 12th International Conference on Machine Learning*. California. EU. Prieditis, A. Russell, S. (eds.) San Francisco, CA US : Morgan Kaufmann. 1995.

[Langley 97] Langley, P. Machine Learning for Adaptive User Interfaces. *KI-97, Advances in Artificial Intelligence, Proceedings of 21st Annual German Conference on Artificial Intelligence*. Germany, septembre 9-12, pp 53-62, Springer, 1997.

[Leake 96] Leake, D.B. CBR in Context : the present and future. In *Case-Based Reasoning Experiences, Lessons, & Future Directions*. D. Leake (eds) AAAI Press /MIT Press. 1996.

[Lebahar 98] Lebahar, J-C. La CAO, révélatuer des structures profondes de l'homme-concepteur. *Forum des Arts de l'Univers Scientifique et Technologique, VIIèmes Rencontres.* Toulouse, octobre 1998.

[Lehuen 97] Lehuen, J. *Un modèle de dialogue dynamique et générique intégrant l'acquisition de sa compétence linguistique. Le système COALA*. Thèse en Informatique de l'Université de Caen, 1997.

[Li 94] Li, H. *Machine Learning of Design Concepts*, Computational Mechanics Publications, 1994.

[Linder et al. 94] van Linder, B., van der Hoek, W. et Meyer, Ch. J.-J. Communicating Rational Agents. *Proceedings of the 18th German Annual Conference on Artificial Intelligence KI-94 : Advances in Artificial Intelligence.* B. Nebel et L. Dreschler-Fischer. (eds.) Saarbrücken, Allemagne, septembre 18-23, 1994. Springer-Verlag 1994.

[Litman & Allen 87] Litman, D. et Allen, J.F. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science*. no. 11, pp 163-200, 1987.

[Litman & Allen 90] Litman, D. et Allen, J. F. Discourse processing and commonsense plans. In *Intentions in Communication*. P.R. Cohen, J. Morgan et M. Pollack. (eds.) MIT Press. 1990. pp 365-388.

[Lochbaum 93] Lochbaum, K. E. *A collaborative planning approach to discourse understanding.* Harvard University, 1993. Technical Report TR-20-93.

[Lochbaum 94] Lochbaum, K.E. *Using Collaborative Plans to Model the Intentional Structure of Discourse*. Thèse de Harvard University, 1994.

[Luzzati 92] Luzzati, D. Un Modèle dynamique pour le dialogue Homme-Machine. *Du dialogue, Recherches sur la philosophie du langage*, Vrin, Paris, n°14, pp 295-314, 1992.

[Maybury 93] Maybury, M.T. *Intelligent Multimedia Interfaces.* AAAI/MIT Press, USA. 1993.

[McCarthy & Hayes 69] McCarthy, J. et Hayes, P.J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Readings in Artificial Intelligence*, (Tioga, Palo Alto, CA ; 1981) 1969.

[McCarthy 80] McCarthy, J. Circumscription : a form of non-monotonic reasoning. *Artificial Intelligence*, vol 13, pp 27-39. 1980.

[McCarthy 84] McCarthy, J. Application of circumscription to formalizing common sense knowledge. *Proceedings of the AAAI Workshop on non-monotonic reasoning*, 1984.

[McDermott 82] McDermott, D.V. A temporal logic for reasoning about processes and plans. *Cognitive Science* 6, pp 101-155, 1982.

[Michalski 83] Michalski, R.S. *A Theory and Methodology of Inductive Learning*. In [Michalski et al., 83].

[Michalski et al. 83] Michalski, R.S., Carbonell, J.G. et Mitchell, T.M.. *Machine Learning. An Artificial Intelligence Approach.*, Vol 1, Tioga Publishing Co, USA. 1983.

[Mille & Napoli 97] Mille A. et Napoli, A. Aspects du raisonnement à partir de cas. *Sixièmes Journées Nationales du PRC-GDR Intelligence Artificielle*. Pesty S. et Siegel P. (eds) Hermes, 1997.

[Motoda & Yoshida 97]Motoda, H. et Yoshida, K. Machine Learning Techniques to Make Computers Easier to Use. *Proceedings of IJCAI-97*, 1997.

[Muggleton 95] Muggleton, S. Inverse Entailment and Progol. *New Generation Computing*. Vol. 13, pp 245-286. 1995.

[Nerzec 93] Nerzec, P. *Erreurs et échecs dans le dialogue oral homme-machine. Détection et Réparation*. Thèse en Informatique. Université de Rennes 1. 1993.

[Nicolle 96] Nicolle, A. Compèrobot, ou comment une machine dialogue-t-elle avec des enfants. In *Le dialogue homme-machine, problèmes psychologiques*. Europia Productions. Paris, 1996.

[Nicolle & Vivier 97] Nicolle, A. et Vivier, J. Apprentissage et Dialogue humain/humain, humain/machine, machine/machine. In *Apprentissage par l'interaction.* K. Zreik, (ed.) pp 61-82. Europia Productions. Paris, 1997.

[Ogden & Bernick 97] Ogden, W.C. et Bernick, P. *Using Natural Language Interfaces*. In [Helander et al. 97].

[Ozkan 94] Ozkan, N. *Analyses communicationnelles de dialogues finalisés.* Thèse en Sciences Cognitives de l'Institut National Polytechnique de Grenoble. 1992.

[Pazzani et al. 96] Pazzani, M. Muramatsu, J. et Billsus, D. Syskill et Webert : Identifying interesting web sites. *Proceedings of the Thirteenth National Conferente of the AAAI*. Portland, AAAI Press. 1996.

[Piaget 64] Piaget, J. *Development and Learning. Piaget Rediscovered*, Ripple R.E. and Roccastle V.N. (eds.), School of Education, Cornell University, Ithaca, New York, 1964.

[Pierrel & Romary 97] Pierrel, J-M. et Romary, L. *Quelles références dans les dialogues homme-machine ?*. In [Sabah et al. 97].

[Prendinger & Schurz 96] Prendinger, H. et Schurz, G. Reasoning about Action and Change. A Dynamic Logic Approach. *Journal of Logic, Language, and Information*, 5:209-245, 1996.

[Py 90] Py, D. *Reconnaissance de plan pour l'aide à la démonstration dans un tuteur de la géométrie*. Thèse en Informatique. Université de Rennes 1. 1990.

[Quinlan 86]Quinlan, J.R. Induction of decision trees. *Machine Learning*. vol. 1(1), pp 81-106. 1986.

[Quinlan 93]Quinlan, J.R. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA. 1993.

[Rosenbloom & Laird 86] Rosenbloom, P.S. et Laird, E.R. Mapping Explanation-Based Generalization onto Soar. *Proceedings of the American Association of Artificial Intelligence AAAI-86*, 1986.

[Roulet et al. 85] Roulet, E., Auchlin, A., Moeschler, J., Rubattel, C., Schelling, M. *L'articulation du discours en français contemporain*, Peter Lang, Berne, 1985.

[Sabah 89] Sabah, G. *L'intelligence artificielle et le langage*. Hermès, Paris. 1989.

[Sabah 97] Sabah, G. Apprentissage et traitement automatique des langues. In *Apprentissage par l'interaction.* K. Zreik, (ed.) pp 31-59. Europia Productions. Paris, 1997.

[Sabah et al. 97] Sabah, G., Vivier, J., Vilnat, A., Pierrel J-M., Romary, L. et Nicolle, A. *Machine, Langage et Dialogue.* L'Harmattan, Paris, 1997.

[Sadek 90] Sadek, D. Logical task modelling for man-machine dialogue. *AAAI-90 Proceedings. Eighth National Conference on Artificial Intelligence.* pp 970-5 vol.2

[Sadek 96] Sadek, D. Le dialogue homme-machine : de l'ergonomie des interfaces à l'agent intelligent dialoguant. *Nouvelles Interfaces Homme-Machine*, série ARAGO, Observatoire Français des Technologiques Avancées. Paris, 1996.

[Searle 72] Searle, J.R. *Les actes de langage : essai de philosophie du langage* (trad. francaise par H. Pauchard). Paris, Hermann, 1972.

[Searle & Vanderveken 85] Searle, J.R. et Vanderveken, D. *Foundation of illocutionary logic*. Cambridge University Press. 1985.

[Shoham 87] Shohlam, Y. Temporal logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence* 33 : 89-104, 1987.

[Song & Cohen 96] Song, F. et Cohen, R. A Strengthened Algorithm for Temporal Reasoning about Plans. *Computational Intelligence*, Vol. 12, No. 2, pp 331-356. 1996.

[Trognon & Brassac 92] Trognon, A. et Brassac, C. L'enchaînement conversationnel. *Les cahiers de Linguistique Française*. No. 13, pp 76-107. 1992.

[Vanderveken 88] Vanderveken, D. *Les actes de discours*. Mardaga éd. Bruxelles, 1988.

[Vanderveken 90] Vanderveken, D. *Meaning and speech acts*. (vol. 1 et 2). Cambridge University Press, 1990.

[Veloso & Carbonell 93] Veloso, M.M. et Carbonell, J.G. Derivational Analogy in PRODIGY : Automating Case Acquisition, Storage and Utilization. In J.L. Kolodner (ed). *Case-Based reasoning*. Kluwer Academic Publishers, 1993.

[Vernant 92]Vernant, D. Approche Actionnelle et Modèle Projectif du Dialogue Informatif. *Du dialogue, Recherches sur la philosophie du langage*, Vrin, Paris, n°14, pp 295-314, 1992.

[Vernant 97]Vernant, D. *Du discours a l'action : études pragmatiques*. Presses Universitaires de France, Paris, 1997.

[Vilnat 97] Vilnat, A. *Quels processus pour les dialogues homme-machine ?*. In [Sabah et al. 97].

[Vivier & Nicolle 97] Vivier, J. et Nicolle, A. *Questions de méthode en dialogue homme-machine : l'expérience Compèrobot*. In [Sabah et al. 97].

[Wilensky 83]Wilensky, R. *Planning and Understanding. A computational approach to human reasoning*. Reading, Addison-Wesley Publishing Company, 1983.

[Wilensky et al. 88] Wilensky, R., Chin, D., Luria, M., Martin, J., Mayfield, J. et Wu, D. The Berkeley UNIX Consultant Project. *Computational Linguistics* Vol. 14, No. 4. 1988.

[Winograd 71] Winograd, T. *Procedures as a presentation for data in computer program for understanding natural language*. MIT, Cambridge. PhD. Thesis. 1971.

[Winograd & Flores 86] Winograd, T. et Flores, C.F. Understanding Computers and Cognition. Norwood NJ, Ablex. 1986.

[Wretö & Caelen 89] Wretô, J. et Caelen, J. *ICPdraw. Rapport final du projet ESPRIT Multiworks n°2105*. 1989.

[Zanello 97] Zanello, M.L. *Contribution à la connaissance sur l'utilisation et la gestion de l'interface multimodale*. Thèse Sciences Cognitives, Université Joseph Fourier, Grenoble, 1997.