Mémoire Présenté

par

Madieyna Lamine Fall

à l'Université du Québec à Trois-Rivières

Comme exigence partielle de la
Maîtrise en mathématiques et informatique appliquées

ProtocolBuilder :

« Outil Logiciel de Développement et d'Expérimentation de Protocoles de Communication pour les Systèmes Multiagents »

Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières

JUIN 2004

RESUMÉ

Le document que vous avez sous les yeux présente le travail qui a été effectué au cours d'un mémoire en informatique, plus précisément dans le domaine de l'intelligence artificielle. De là, nous abordons le domaine multiagent, en mettant en avant un thème auquel nous souhaitons nous intéresser plus particulièrement, la communication inter-agent. Ce thème va permettre d'abord de mettre en avant une approche de l'intelligence artificielle autour de laquelle va se construire notre travail.

La communication est un élément principal des sociétés d'agents artificiels, et des normes sont exigées pour réguler la prise de décision distribuée impliquée dans les interactions comme, par exemple, la négociation ou la persuasion. Un protocole de communication spécifie les règles que deux agents communicants ou plus doivent suivre au niveau d'une conversation. Il tiendra compte habituellement de plusieurs expressions alternatives dans chaque situation et l'agent en question doit en choisir une selon sa stratégie.

Dans les sociétés ouvertes d'agent, on ne peut pas supposer que les protocoles et les stratégies de communication s'assortissent toujours parfaitement, vu qu'ils sont typiquement spécifiés par différents concepteurs. Ces anomalies potentielles soulèvent un certain nombre de questions intéressantes, telles que la flexibilité, l'extensibilité, l'interopérabilité et le plus notamment le problème de vérifier la conformité du comportement d'un agent par rapport aux règles décrites par un protocole. Au niveau de ce document, nous décrivons la conception et l'exécution du logiciel appelé ProtocolBuilder. Ce logiciel nous permet d'établir des protocoles de communication et de vérifier la conformité des conversations d'inter-agent par rapport des protocoles de communications spécifiques. ProtocolBuilder devrait avoir les propriétés suivantes:

Fiabilité: mesure la capacité du Logiciel à conserver un comportement conforme a ses spécifications.

Flexibilité: mesure l'aptitude du logiciel à faciliter l'adjonction de nouvelles fonctionnalités ou la modification, voire la suppression, de fonctionnalités existantes (cet

aspect couvre la correction des défauts provenant d'une mauvaise expression des besoins),

Portabilité: mesure l'aptitude du logiciel à minimiser les conséquences d'un changement d'environnement technique (système, matériel...).

Réutilisabilité qui mesure l'aptitude du logiciel à une réutilisation de tout ou partie de ses composants dans le cadre d'un autre projet,

Utilisabilité: mesure l'aptitude d'un logiciel à être d'une utilisation agréable et facile pour l'utilisateur à qui il est destiné,.

ABSTRACT

The document that you have under the eyes presents the work which was carried out during a report in data processing, more precisely in the field of the artificial intelligence. From there, we approach the field multiagent, by proposing a topic in which we wish to be interested more particularly: inter-agent communication. This topic will initially make it possible to propose an approach of the artificial intelligence around which will build our work: «inter-agent communication languages and protocols ».

Communication is one of the key feature of societies of artificial agents, and standards are required to regulate the distributed decision-making involved in interactions such as, for instance, negotiation or persuasion. A protocol specifies the \rules of encounter" governing a dialogue between two or more communicating agents. It specifies which agent is allowed to say what in any given situation. It will usually allow for several alternative utterances in every situation and the agent in question has to choose one according to its strategy.

In open agent societies, communication protocols and strategies cannot be assumed to always match perfectly, because they are typically specified by different designers. These potential discrepancies raise a number of interesting issues, such as flexibility, extensibility, interoperability and most notably the problem of checking that the behavior of an agent is (or will be) conformant to the rules described by a protocol. In this paper, we describe the design and the implementation of the software named ProtocolBuilder. This software enables us to build communication protocols and to verify the compliance of inter-agent conversations towards specific protocols. ProtocolBuilder should have the following properties:

Reliability: measures the ability of the software to preserve a behavior in conformity with the specifications, even in the worth case scenarios.

Flexibility: measures the ability of the software to perform even outside an intended domain.

Portability: measures the ability of the software to perform in any platform.

Usability: measures the ability of the software to be "user-friendly".

Re-usability: measures the ability of the software to import part of its components to be used in many different applications.

DEDICACES

Je dédie ce mémoire à :

- mes parents :
  - ma mère, Awa Diop Fall
  - mon père, Pape Louis Fall

- mes frères et ma sœur :
  - Pape Becaye Fall
  - El hadji Malick Fall
  - Madeleine Fall

REMERCIEMENTS

Mes remerciements vont en direction de :

- Mes parents pour tous leurs sacrifices et leur amour constant tout au long de mes études.

- Mes frères et sœurs Malick, Becaye, Madeleine pour toute leur affection et leurs conseils.

- Le personnel du Service de l'informatique de l'Université du Québec à Trois-Rivières pour leur disponibilité et leur support technique.

- Mes professeurs du département de mathématiques et d'informatique pour leur amitié.

- Chantal Lessard, la secrétaire du comité des études de cycles supérieurs au niveau du département de Mathématiques et Informatique, pour sa disponibilité permanente.

- Mon directeur de maîtrise, le professeur Sylvain Delisle pour sa direction et son support constant.

TABLE DES MATIÈRES

LISTE DES FIGURES *12*

LISTE DES TABLES *14*

Chapitre 1 : INTRODUCTION *16*

1.1 Objectifs **16**

1.2 Notion d'agent **17**
1.2.1 Un exemple mono agent : bonzibuddy (bonzi.com) 19

1.3 Système Multiagent **19**
1.3.1 Un exemple de système multiagent commercialisé : le jeu "QUAKE" 20

1.4 Communication inter-agent **21**
1.4.1 Décomposition et distribution des tâches 22
1.4.1.1 Communication directe 23
1.4.1.2 Communication indirecte 25

1.5 Problèmes au niveau de la communication **27**

1.6 Survol **28**

Chapitre 2 : PROBLEMATIQUE *30*

2.1 Langages et protocoles de communication **31**
2.1.1 Problèmes au niveau de la communication 32
2.1.1.1 Théorie d'agence 32
2.1.1.2 Problèmes au niveau de la sémantique 34
2.1.1.3 Problèmes au niveau de la vérification des conversations 35
2.1.1.4 Problèmes au niveau de l'ontologie 36
2.1.1.5 Complétude des types de message ACL 38
2.1.1.6 Règles Conversation (ou protocoles de communication) 39

2.2 Synthèse de la problématique **43**

Chapitre 3 : ÉTAT DE L'ART *45*

3.1 Langages et Protocoles de communication au sein des outils existants **45**
3.1.1 KQML (Knowledge Query and Manipulation Language) 46
3.1.1.1 Présentation 46
3.1.1.2 Structure des messages KQML 47
3.1.1.3 Désavantages 47
3.1.2 FIPA ACL [Foundation for Intelligent Physical Agents] 48
3.1.2.1 Présentation 48
3.1.2.2 Structure des messages FIPA 49
3.1.3 Actes de discours / Performatifs (Speech acts) 51
3.1.3.1 Liste de performatifs (Actes de discours) du langage KQML 52
3.1.3.2 Liste des actes communicatifs du langage FIPA 54

3.2. Problèmes au niveau des langages de communication **55**
3.2.1 Sémantiques des langages de communication 55
3.2.1.1 Manques au niveau de la Sémantique 56
3.2.1.2 Ontologies partagées 58
3.2.2 En résumé 59

3.3 Règles de conversation **60**
3.3.1 Que sont les règles de conversation ? 61
3.3.1.1 Demande d'action (Request for action) 61
3.3.2 Limites des règles de conversation 62
3.3.2.1 Flexibilité 63
3.3.2.2 Spécification 64

3.4 Nouveaux langages basés sur les engagements et les jeux de dialogue **67**
3.4.1 Un environnement pour les sémantiques des performatifs KQML 67
3.4.1.1 Que sont les notions dialectiques 67
3.4.1.2 Engagements sociaux 67
3.4.1.3 Systèmes dialectiques 69
3.4.2 Protocoles basés sur les « *engagements sociaux* » 73
3.4.2.1 Machines d'engagement (commitment machines) 73
3.4.2.2 Colombetti (Albatross) 78
3.4.2.3 Protocole de proposition (Flores et Kremer) 86
3.4.2.4 Architecture pour la Planification Argumentative du Dialogue (Reed) 90
3.4.2.5 Jeux formels de dialogue (Mac Burney et Parsons) 95
3.4.2.6 Jeux de dialogues (Engagements (Maudet et Chaib-draa)) 100
3.4.2.7 Protocoles de négociation flexible *(Dastani et al)* 105

3.5 Les différents Modèles pour représenter de conversations **108**
3.5.1 Présentation des différentes structures 108
3.5.1.1 Machine à états finis 108
3.5.1.2 Machines à états finis : FSM (Winograd & Flores) 109
3.5.1.3 Graphes de Dooley (Dooley Graph) 110
3.5.1.4 Les réseaux de Pétri (Petri Net) 111
3.5.2 En conclusion 112

Chapitre 4 : SOLUTION PROPOSÉE, logiciel ProtocolBuilder ***114***

4.1 Analyse des caractéristiques voulues du logiciel **114**

4.2 Choix de conception pour ProtocolBuilder **116**
4.2.1 Modèle de cycle de vie adopté au niveau de la conception du logiciel 116
4.2.2 Choix du langage de programmation : java 116
4.2.3 Modélisation des protocoles de communications 118
4.2.4 Choix de conception des conversations 120
4.2.5 Choix de conception des performatifs 121
4.2.6 Choix de conception des paramètres liés aux actes de discours 122
4.2.7 Conception de l'aspect "création de protocoles" 123
4.2.8 Vérification des conversations par rapport aux protocoles crées 123

4.3 Description du fonctionnement de " ProtocolBuilder" 125
4.3.1 Caractéristiques du logiciel 125
4.3.2 Création protocole avec le logiciel « *ProtocolBuilder* » 127
4.3.2.1 Nom et description protocole 127
4.3.2.2 Définir/Sélectionner la liste d'actes communicatifs 128
4.3.2.3 Définir une nouvelle étiquette 129
4.3.2.4 Établir une liste de paramètres pour chaque acte de discours 131
4.3.2.5 Création de la liste des états pour simuler la conversation 132
4.3.2.6 Faire le lien entre les états 133
4.3.2.7 Après avoir fait le lien entre les états 134
4.3.2.8 Fichiers générés 137
4.3.3 Vérification/validation des protocoles crées 139
4.3.3.1 Interface permettant de visualiser les propriétés des protocoles 139
4.3.3.2 Interface permettant la vérification de la conformité 141
4.3.4 Évaluation 145
4.3.4.1 Outils de simulation : «JACK INTELLIGENT AGENTS » 145
4.3.4.2 Simulation du protocole CONTRACTNET 151

Chapitre 5 : CONCLUSION* *166

Chapitre 6 : TRAVAUX FUTURS* *170

Chapitre 7 : REFERENCES* *172

7.1 Bibliographie 172

7.2 Webliographie 175

LISTE DES FIGURES

Figure 1 : Agent « Bonzi » 19
Figure 2: Série d'image du jeu commercial de combat, "QUAKE" 21
Figure 3: Système délégué d'allocation 23
Figure 4: Structure organisationnelle (Client-Server) 24
Figure 5: Décomposition des sous-tâche (partage des tâches) 24
Figure 6: Mémoire partagée globale : Système de tableau noir (BlackBoard) 26
Figure 7: Structure des messages 51
Figure 8: Liste des Actes communicatifs KQML 52
Figure 9 : Liste des actes communicatifs (FIPA ACL 2000) 54
Figure 10: Protocole "demande d'action" 61
Figure 11: Exemple de "sous-dialogue" de négociation (le dialogue d'action). 72
Figure 12: Exemple de diagramme d'interaction du protocole de proposition 87
Figure 13: System architecture Overview 91
Figure 14: The Initiative/Reactive (IR unit) structure 106
Figure 15: Exemple d'échange pour obtenir de l'information 106
Figure 16: Sequential Combination of games 107
Figure 17: Modèle de Winograd/Flores avec des Actes de discours formalisés 109
Figure 18: Graphes de Dooley représentant le tableau précédente (tableau 5) 111
Figure 19: Réseaux de pétri 112
Figure 20: Exemple d'envoi de message 118
Figure 21: Protocole "transitions d'états pour la négociation" 119
Figure 22: Format des messages 120
Figure 23: Format des fichiers contenant les actes de discours 121
Figure 24: Format de fichier contenant les différents paramètres 122
Figure 25: Modélisation de la solution proposée 125
Figure 26: Interface permettant de saisir le nom d'un protocole et sa description 127
Figure 27: Interface permettant de saisir la liste des actes de discours 128
Figure 28: Interface permettant de définir un nouvel ensemble d'actes de discours 130
Figure 29: Interface permettant d'établir la liste des paramètres pour les actes choisis . 131
Figure 30: Interface permettant de saisir la liste des états de la machine à états finis 132
Figure 31: Interface permettant de faire le lien entre les états (machine à états finis) ... 133
Figure 32: Modélisation de la machine à états finis qui dirige les règles de transitions. 134
Figure 33: Symboles représentant les actes de discours au niveau des arcs (FSM) 134
Figure 34: Interfaces montrant les différentes tailles possibles d'une machine 135
Figure 35: Visualisation : positionnement d'une machine 136
Figure 36: Exemple de machine à états finis sauvegardée 137
Figure 37: Fichier permettant de retenir les actes de discours suivi de leurs symboles . 138
Figure 38: Exemple d'éléments de protocole sauvegardé « test_ultime.prot » 138
Figure 39: Interface globale permettant de visualiser les propriétés des protocoles 139
Figure 40: Espace réservé pour l'explication d'un protocole créé 140
Figure 41: Visualisation actes/paramètres stockés d'un protocole créé 141
Figure 42: Sélection d'une conversation existante 142
Figure 43: Conversation sélectionnée 142

Figure 44: Traces de compatibilité (actes/paramètres) 143
Figure 45: ProtocolBuilder : Analyse respect des séquences / FSM (protocole) 144
Figure 46: Logiciel Jack , vue d'ensemble 146
Figure 47: Logiciel Jack , édition de plans 147
Figure 48: Historique d'exécution de plans 148
Figure 49: Protocole FIPA-ContractNet 151
Figure 50: Exécution de la simulation du protocole CONTRACTNET avec "JACK" .. 153
Figure 51: ContractNet, Messages envoyés par le manageur aux différents agents 156
Figure 52: ContractNet, réactions des différents agents face à l'offre du proposeur 156
Figure 53: ContractNet, réactions du proposeur face aux différentes réponses 157
Figure 54: ContractNet, premier conversation entre agents 159
Figure 55: Interface protocolBuilder gérant la première conversation (figure 54) 159
Figure 56: ContractNet, deuxième conversation entre agents 160
Figure 57: Interface protocolBuilder gérant la deuxième conversation (figure 56) 160
Figure 58: ContractNet, troisième conversation entre agents 161
Figure 59: Interface protocolBuilder gérant la troisième conversation (figure 58) 161
Figure 60: Exemple positif de conversation 162
Figure 61: Exemple négatif de conversation 162
Figure 62: Interface ProtocolBuilder gérant la conversation de la figure 61 163
Figure 63: Exemple négatif de conversation 164
Figure 64: Interface ProtocolBuilder gérant la conversation de la figure 63 165

LISTE DES TABLES

Tableau 1: Paramètres des messages FIPA ACL 50
Tableau 2: Ontologie du message FIPA ACL 50
Tableau 3: Signification des actes de discours du langage KQML 53
Tableau 4: Comment le stockage des engagements évolue durant une conversation..... 70
Tableau 5: Les six types principaux de dialogue 71
Tableau 6: Exemple de conversation 110

Chapitre 1 : INTRODUCTION

1.1 Objectifs.....16

1.2 Notion d'agent.....17
1.2.1 Un exemple mono agent : bonzibuddy (bonzi.com).....19

1.3 Système Multiagent.....19
1.3.1 Un exemple de système multiagent commercialisé : le jeu "QUAKE".20

1.4 Communication inter-agent.....21
1.4.1 Décomposition et distribution des tâches.....22

1.5 Problèmes au niveau de la communication.....27

1.6 Survol.....28

Chapitre 1 : INTRODUCTION

Le document que vous avez sous les yeux présente le travail qui a été effectué au cours d'un mémoire en informatique, plus précisément dans le domaine de l'intelligence artificielle. Dans cette introduction, nous allons expliquer la démarche suivie dans le présent mémoire. Dans ce but, nous définissons tout d'abord la notion d'agent intelligent. De là, nous abordons le domaine multiagent, en mettant en avant un thème auquel nous souhaitons nous intéresser plus particulièrement, la communication inter-agent. Ce thème va permettre d'abord de mettre en avant une approche de l'intelligence artificielle autour de laquelle va se construire notre travail: « les langages et protocoles de communication inter-agent ». Puis sera enfin présenté le problème précis qui va retenir notre attention (section 1.4): les problèmes au niveau des langages et protocoles de communication. Le présent chapitre continuera avec une section 1.5 consacrée à quelques points dont il nous semble important de tenir compte dans la conception de l'outil de développement et d'expérimentation de protocoles de communication qui constitue le noyau de mon projet de maîtrise pour ensuite terminer avec la section 1.6 qui constitue un survol des différents chapitres présentés au niveau de ce document.

1.1 Objectifs

Le but de mon projet est de concevoir un environnement de développement et d'expérimentation au niveau des systèmes multiagents tentant de combler plusieurs lacunes identifiées au niveau de la communication inter-agent, plus précisément au sein des protocoles de communications. Ce projet va se matérialiser par la conception et l'implémentation d'un logiciel (*ProtocolBuilder*) qui nous donnera la possibilité de facilement :

1. Créer des protocoles de communication inter-agent de manière flexible/générique. La création de protocoles flexibles pourra être utilisée au niveau de l'exploitation des conversations entre les agents.

2. Vérifier/Valider le bon suivi des protocoles utilisées au niveau des conversations inter-agents. Plusieurs aspects pourront être vérifiés :

   - Vérification de l'exactitude des protocoles au niveau de leurs conceptions.
   - Vérification de la conformité des conversations par rapport aux protocoles.
   - Vérification de l'ordre des messages au sein de la conversation menant directement à un état satisfaisant (état conforme aux spécifications d'un protocole en particulier).
   - Vérification des critères déclaratifs des protocoles (complétude des messages)

Cet environnement permettra d'avoir une meilleure compréhension en soustrayant les détails des messages et des protocoles tout en se basant sur des principes sur lesquels la vérification et la conformité aux protocoles peuvent être rigoureusement analysées à la conception et niveau de l'exécution.

1.2 Notion d'agent

« *Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multiagent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents* ». [Ferber 95]. Les agents possèdent les propriétés suivantes :

- autonomie : les agents fonctionnent sans intervention directe des humains ou d'autres, et ont un certain contrôle de leurs actions et de leur état interne ;

- capacités sociales : les agents agissent avec d'autres agents (et probablement des humains également) par l'intermédiaire d'un certain genre de langage de communication inter-agent.

- réactivité : les agents perçoivent leur environnement, (qui peut être le monde physique, un utilisateur par l'intermédiaire d'une interface utilisateur graphique, une collection d'autres agents, l'INTERNET, ou peut-être le tout combiné), et répondent d'une manière opportune aux différents changements;

- dynamique : les agents n'agissent pas simplement en réponse à leur environnement, ils sont en mesure d'exhiber un comportement dirigé par un but précis en prenant des initiatives.

Il existe plusieurs types d'agents intelligents:

1. Agents Mobiles: statique ou mobile
2. Agents qui collaborent (collaborating agent).
3. Agents qui négocient (negotiating agent).
4. Agents intentionnels (Intentional agent)
5. Agent d'Information

Chaque agent possède

- un ensemble de croyances
- un ensemble d'objectifs (que l'agent tente d'atteindre)
- un ensemble de plans (combinaisons d'actions qui permettent d'obtenir un certain résultat).

1.2.1 Un exemple mono agent : bonzibuddy (bonzi.com)

L'agent Bonzi est un exemple d'agent personnel. Comme on peut voir au niveau de la figure 1, il offre plusieurs services à l'usager, il explore l'Internet, parle, raconte des blagues, cherche des fichiers (browse), effectue des recherches au niveau de l'Internet, gère les courriers électroniques, compare les prix des produits qui intéressent l'usager lui permettant ainsi d'économiser. Bonzi devient de plus en plus intelligent au fur et 'a mesure qu'on « surfe » à travers Internet, identifiant nos goûts et intérêts.

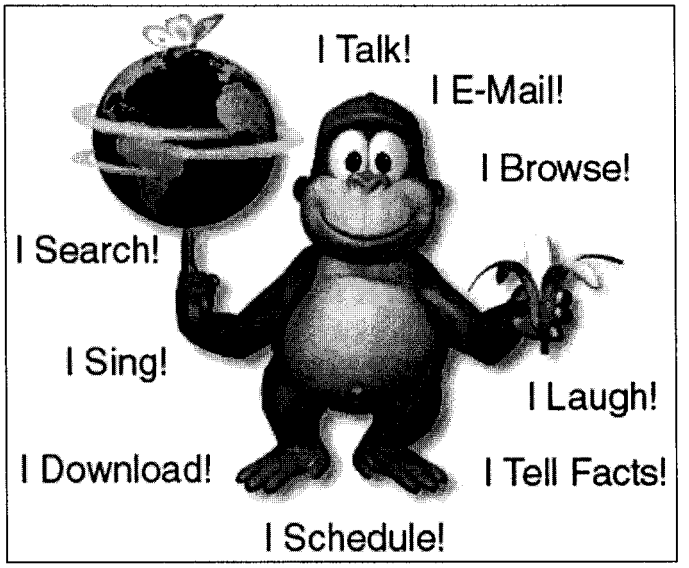


Figure 1 : Agent « Bonzi »

1.3 Système Multiagent

« Un système MultiAgent est une cohorte d'agents intelligents et de personnes capables de se coordonner en se comportant de façon autonome ayant des buts à atteindre, et avec souplesse dans un certain environnement » [Ferber 95]. Un système multiagent pourrait se composer d'un ensemble d'agents intelligents (agents intelligents) d'information qui extraient et intègrent les informations d'un ensemble de sources distribuées au niveau du Web, d'un groupe de personnes travaillant ou jouant ensemble au niveau d'un monde virtuel, ou une combinaison de personnes et d'agents intelligents d'interface (intelligent interface agents) qui fournissent une entreprise virtuelle coordonnée incluse dans le vrai monde. Un système multiagent peut être homogène ou hétérogène. Au niveau d'un système multiagent homogène, tous les agents ont la même

structure interne, ceci incluant les objectifs, les connaissances des domaines, et les actions possibles. Ces agents choisissent leurs prochaines actions en utilisant la même procédure. Par contre, au niveau d'un système multiagent hétérogène, les agents peuvent être hétérogènes en ayant des objectifs différents, des modèles de domaines et des actions différentes.

Un principe essentiel qui permet aux agents ayant leurs propres objectifs d'arriver à collaborer entre eux au niveau d'un système multiagent est la communication. La communication est nécessaire afin de permettre collaboration, négociation, coopération, entre agents.

1.3.1 Un exemple de système multiagent commercialisé : le jeu "QUAKE"

Les jeux d'ordinateur interactifs créent les mondes et les caractères virtuels pour que les personnes agissent l'un sur l'autre dynamiquement. Au niveau du jeu **QUAKE**, le système multiagent est composé de personnages artificiels (ennemis /monstres) qui sont des agents intelligents qui ont pour tâche d'éliminer le joueur en question. Les personnages au niveau du jeu sont des caractères autonomes qui peuvent improviser et réagir au niveau de leur environnement. Chaque agent possède plusieurs clones, des capacités (par exemple le type d'armes qu'ils utilisent, leurs puissances spécifiques, l'aptitude de voler, de nager, ...), qui leur sont propres, communique avec les autres monstres, réagit face à plusieurs stimulus tels que le bruit afin d'éliminer le joueur en question. On peut voir une série d'images provenant du jeu QUAKE au niveau de la figure 2 :

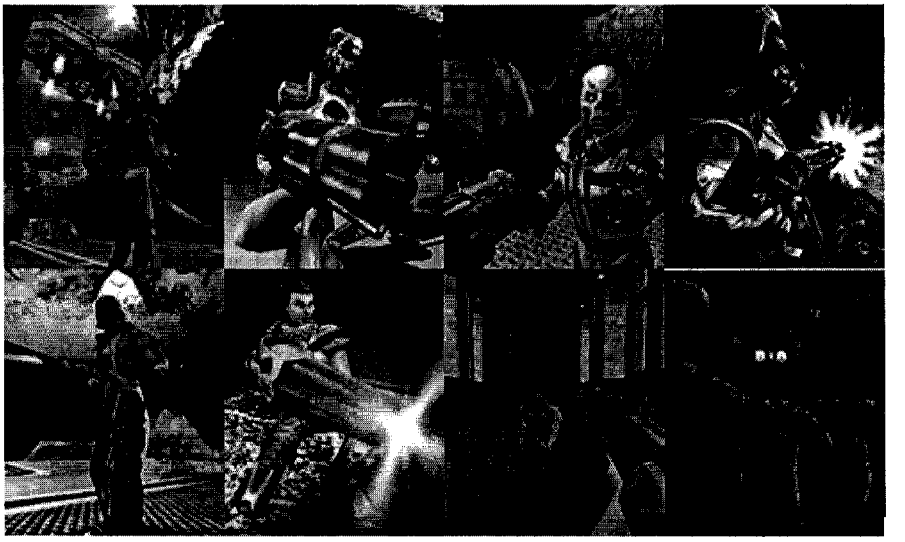


Figure 2: Série d'image du jeu commercial de combat, "QUAKE".

Ainsi au sein de l'intelligence artificielle, les jeux d'ordinateur interactifs représentent un domaine d'application naissant dans lequel le but du niveau humain peut être poursuivi avec succès. Les jeux d'ordinateur interactifs ont des mondes de plus en plus complexes et réalistes et les caractères commandés par ordinateur sont de plus en plus complexes et intelligents.

1.4 Communication inter-agent

Les agents qui fonctionnent coopérativement peuvent partager l'accès aux ressources (telles que l'information ou la connaissance), aussi bien que la contribution vers un but commun. En concevant des agents pour accomplir des tâches spécialisées, et pour travailler coopérativement avec d'autres agents, la complexité des systèmes et des différentes tâches à accomplir peut être réduite. Le partage d'une forme commune de communication est essentiel au succès de la coopération des agents, et il y a beaucoup de solutions disponibles pour établir le lien de communication. Quelques principes important au niveau de la communication au sein des systèmes multiagents sont les principes de décomposition et de distribution des tâches. La communication peut se faire de manière directe ou indirecte

1.4.1 Décomposition et distribution des tâches

La décomposition des tâches peut être faite par le concepteur du système ou par des agent utilisant la planification hiérarchique ("hierarchical planning"). La décomposition de tâche pourrait être faite dans l'espace, basé sur la disposition des émetteurs d'informations ou des points de décision, ou fonctionnellement, selon l'expertise des agents disponibles. Une fois les tâches décomposées, elles peuvent être distribuées. Les Critères de distribution de tâches sont :

- éviter la surcharge des ressources critiques,
- Attribuer les tâches aux agents ayant les compétences adéquates pour traiter les tâches correspondantes,
- Faire en sorte qu'un agent ayant une vue globale puisse donner des tâches à d'autres agents,
- Attribuer des tâches très interdépendantes aux agents à proximité spatiale / sémantique (afin de minimiser les communications).
- Permettre aux tâches urgentes d'être traité en priorité (reassign tasks)

Ainsi, le procédé consiste à identifier et à gérer les rapports entre les divers sous-problèmes (composantes) à résoudre. La possibilité de spécifier et d'établir des relations organisationnelles entre agents aide les concepteurs à faire face à la complexité parce qu'en premier lieu on peut grouper diverses agents dans une composante plus complexe qui peut être alors traitée comme une unité unique d'analyse et, deuxièmement, parce qu'on a la possibilité de décrire des relations complexes entre ces unités, par exemple coopération, fonctionnalités conjointes, etc.

Le partage des tâches peut se faire à travers deux modes de communication : communication directe et indirecte. Les structures organisationnelles utilisent un mode de communication direct, et les approches utilisant le concept de mémoire partagée ("shared memory patterns") utilisent un mode de communication indirect. On effectue la division d'une tâche en sous-tâches (tâches secondaires) pour l'attribution possible de chaque sous-tâche à un autre agent (sachant que chaque sous-tâche doit correspondre aux

qualifications de l'agent choisi). Le but est de rendre les tâches secondaires le plus indépendant possible afin de minimiser la coordination, les données partagées et les ressources partagées.

1.4.1.1 Communication directe

Comme on peut voir au niveau de la figure 3, la communication directe implique que les agents "se connaissent" (identification, localisation) et échange des données en ouvrant des chaînes de communication bout a bout.

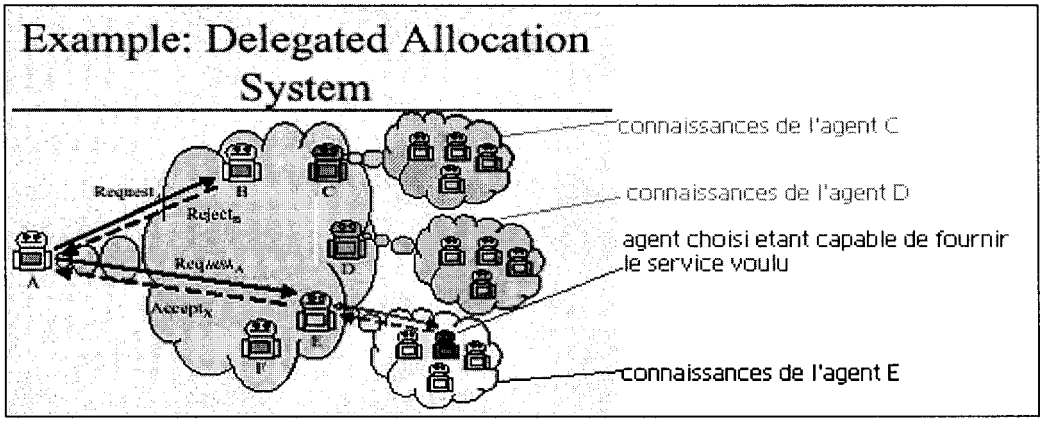


Figure 3: Système délégué d'allocation

La communication inter-agents via l'environnement est basée sur le passage de message, où les agents communiquent en formulant et en s'envoyant des messages individuels. Ainsi pour que des agents puissent communiquer via l'environnement, il faut:

- qu'ils utilisent le même langage de communication (KQML, FIPA ACL, ...).
- qu'ils se servent de protocole de communication (p.e « contract net » pour les enchères).
- qu'ils aient la même ontologie pour que les messages échangés soient compréhensibles des deux bords, afin d'identifier la source du vocabulaire et d'interpréter le contenu du message de la même manière.

Un protocole de coopération inter-agent effectue le partage et la distribution des tâches (« *Task Decomposition/Distribution, task sharing* »). Lorsqu'un agent reçoit une

tâche, il peut soit accomplir cette tâche, ou bien décomposer cette tâche et la distribuer à d'autres agents à travers des structures organisationnelles.

- **Structures organisationnelles**

La décomposition du problème dans des composantes qui s'identifient avec les agents semble une solution naturelle. L'autonomie des agents réduit la complexité du contrôle car le contrôle est localisé dans chaque composante (agent). Les décisions des agents sont prises en partant de la situation locale à l'agent. Les structures organisationnelles sont nécessaires pour structurer le système et éviter les explosions combinatoires du nombre de messages transmis. Au niveau des structures organisationnelles, les sous-agents représentent différentes fonctions et non des ressources parallèles (comme on peut voir au niveau des figures 4 et 5). Faire appel aux sous-agents est comme faire appel à une procédure dans un programme.

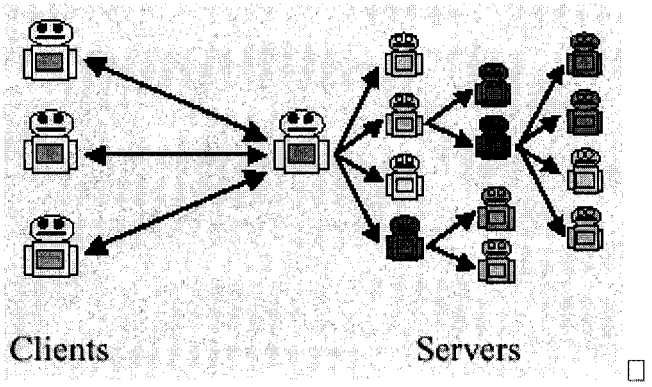


Figure 4: Structure organisationnelle (Client-Server)

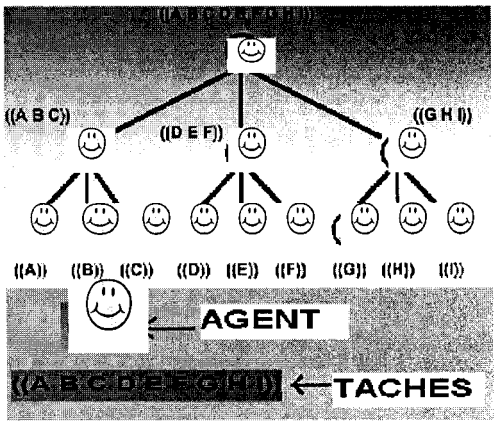


Figure 5: Décomposition des sous-tâche (partage des tâches)

Le rôle des agents en haut de la hiérarchie est d'avertir tous les agents que l'objectif a été atteint. Le rôle de chaque agent traitant les sous-tâches est d'avertir en montant dans la hiérarchie lorsque le sous objectif traité est atteint.

1.4.1.2 Communication indirecte

Cette sous-section présente un modèle architectural définissant un modèle de communication indirect au niveau des agents basé sur l'utilisation d'une mémoire partagé, et de montrer le comportement de ces agents face a cette architecture.

- **Mémoire partagée globale (global shared memory)**

L'utilisation d'une mémoire partagée globale (global shared memory) est une manière de communiquer de manière indirecte. Cette mémoire est connue et accessible par tous les composants logiciels du système et représente le seul moyen de communication pour ces agents. Le principe est que lorsqu'un agent produit de l'information (atteinte d'un objectif par exemple) qui peut être utile pour d'autres agents, il le stocke dans une mémoire (dépôt) partagée, accessible par les autres agents. Les autres agents iront chercher (retreive) leurs informations à partir de cette mémoire partagée (shared repository) si besoin est. Cette approche soulève un problème : quand et comment les agents sont activés afin de lire et d'écrire dans la mémoire partagée? Cette approche présente une forme indirecte de communication où les agents ne se « connaissent » pas. De plus au niveau de cette approche, aucune information n'est fournie concernant l'identification des agents qui stockent les données, ou des agents qui vont utiliser ces données.

- **Système de tableau noir (blackboard)**

Les résultants sont partagés à travers la structure de donnée appelée "tableau noir" (Blackboard system) comme on peut voir au niveau de la figure 6. De Multiple agents peuvent lire et écrire dessus, ainsi les agents écrivent leurs solutions partielles dessus. Les systèmes de «blackboard» peuvent être structurés en hiérarchie. L'exclusion mutuelle est nécessaire, elle empêche des activités concurrentes ce qui peut entraîner une situation de « blocage » (bottleneck). Suite a cela, les résultats partiaux peuvent être récupérés par d'autres agents qui en ont besoins, de plus les hypothèses affichées peuvent être confirmées ou réfutées ce qui peut se révéler utile pour d'autres agents qui peuvent (ou qui seront tentées de) se baser sur ces hypothèses afin d'atteindre leur sous objectifs.

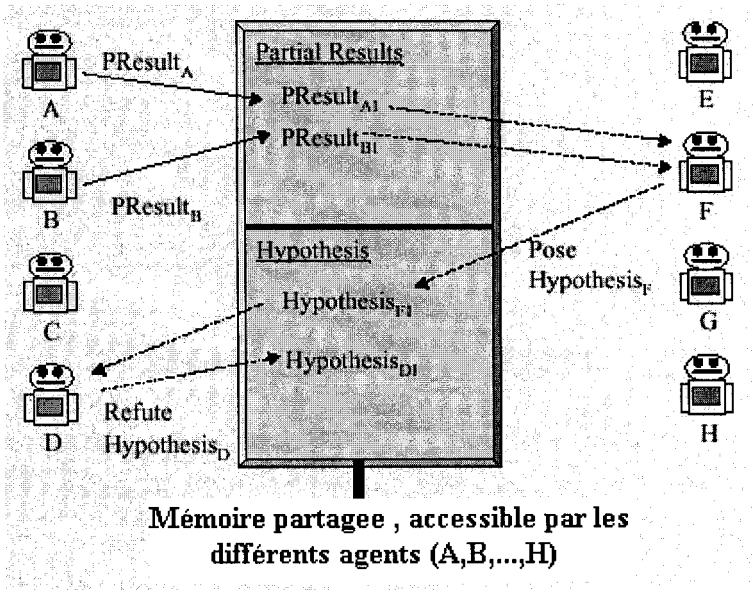


Figure 6: mémoire partagée globale : Système de tableau noir (BlackBoard)

1.5 Problèmes au niveau de la communication

Au niveau des langages de communication, À l'heure actuelle, les deux langages de communication qui font de fait office de standards sont ***KQML*** et ***FIPA-ACL***. Cependant, dû à leurs approches spécifiques, les agents issus de différents environnements ne peuvent pas communiquer. Afin d'établir un consensus, nous devons résoudre plusieurs problèmes restants. L'état actuel de la recherche au niveau des langages de communication pourrait être récapitulé comme suit :

• Manque de sémantique formelle : une sémantique formelle universelle peut aider au niveau de la préservation de la signification des messages à travers différentes plateformes.

• Faible hétérogénéité : les agents issus de différents domaines d'exécution ne peuvent pas encore communiquer parce qu'ils sont confinés dans des contextes rigides.

• Ontologies partagées : une ontologie, qui possède une large couverture concernant son domaine, et qui est extensible, est nécessaire dans beaucoup de langages de communication courants

• Problèmes de flexibilité au niveau des protocoles de communication : on peut voir les protocoles de communication comme une autre manière de structurer le passage de messages au niveau de la communication inter-agent. Les protocoles courants d'interaction et de négociation ont des domaines non définis d'interaction (des buts et des opérations) ou de stratégies de négociation.

De plus, au niveau des outils de développement orientés agent, un grand problème se présente : le problème de standardisation. Nous avons une multitude d'outils de développement d'agent et/ou de systèmes multiagent, chacun utilisant sa propre méthodologie. Ceci rend difficile le développement d'un système multiagent avec plusieurs outils différents, ce qui rend donc difficile le principe de mélange d'outils (« tool mix »), principe important au niveau des outils de développement orienté objet. Présentement, il existe peu ou pas d'outils informatiques permettant de supporter adéquatement le développement et l'expérimentation des divers langages et protocoles de communication dans les systèmes multiagents. Ainsi Les langages de communications

implantés au niveau des différents outils possèdent des failles. Tout d'abord la sémantique de ces langages réfère directement à l'état interne « supposé » d'un agent, via les notions ***d'états mentaux***. Ensuite ils ne permettent pas en pratique de prendre part à des conversations et doivent être utilisés avec des protocoles de communication (i.e. des machines à états finis qui spécifient les enchaînements d'énoncés bien formés). Comme on le verra au niveau du chapitre 3 (État de l'art), afin de dépasser ces limitations, un fort courant actuel tente de fonder des langages de communication sur des notions publiques, comme les ***engagements sociaux***; et des modèles de conversations plus souples que les protocoles, comme par exemple les ***jeux de dialogue.***

1.6 Survol

Le document est divisé de la façon suivante. Dans un premier temps, la problématique de l'approche agent est abordée. Dans cette section, différents problèmes reliés aux Systèmes multiagents sont abordés. Le chapitre 3 se veut une revue de littérature survolant l'univers des systèmes multiagents. Cette section met en relief les principaux travaux du domaine de la communication inter-agent, plus précisément les langages et protocoles de communication. Notamment des langages de communication qui sont plus ou moins standard (KQML, Fipa Acl), et des protocoles de communications fondés sur des notions publiques (engagements sociaux) et sur des modèles plus souples (jeux de dialogue). Le chapitre 4 va présenter l'approche proposée (logiciel *protocolBuilder*) en passant par l'analyse, la conception, la description des services offerts par *protocolBuilder*, pour ensuite se concentrer sur l'évaluation du logiciel. Finalement une conclusion qui effectue une réévaluation de l'atteinte des objectifs fixés au niveau des sections qui précèdent va clôturer ce document.

Chapitre 2 : PROBLEMATIQUE

2.1 Langages et protocoles de communication.....17

2.1.1 Problèmes au niveau de la communication.....18

2.2 Synthèse de la problématique.....24

Chapitre 2 : PROBLEMATIQUE

Ce chapitre effectue la présentation des différents aspects qui représentent des difficultés au niveau de la communication inter-agent. Cette présentation va nous permettre de cibler les aspects qui devront être traités au niveau de l'environnement à développer qui constitue l'objectif de ce mémoire (section 1.1), et de souligner par la même occasion les défis qui nous attendent à ce niveau.

Dans le domaine multiagent, la communication entre les agents autonomes est un secteur provocant de recherche qui implique plusieurs disciplines : philosophie de la langue, psychologie sociale, intelligence artificielle, logiques, mathématiques, etc... Lorsque plusieurs agents logiciels autonomes prennent part à une activité commune (système multiagent), il est crucial qu'ils puissent communiquer afin de négocier, de résoudre des conflits, de coopérer (par exemple, pour coordonner leurs actions). Les langages de communications permettent aux agents de communiquer, partager leurs connaissances avec d'autres agents, et cela malgré certaines contraintes (différences au niveau des plates formes, de systèmes d'exploitation, des langages de programmation,...). Lors de l'échange d'informations qui réunit des notions de haut niveau, cette communication nécessite l'utilisation de langages de communication entre agents (ACL). L'interaction inter-agent va au-delà des échanges de messages. Les problèmes qui y sont associés sont : les modèles d'agents (croyances, buts, représentation du raisonnement), protocoles d'interaction (une politique d'interaction qui guide les agents) et langages d'interactions (langages qui introduisent des types de messages standard que tous les agents interprètent de façon identique). Une manière de structurer le passage de messages au niveau de la communication inter-agent est à travers les protocoles de communication.

2.1 Langages et protocoles de communication

Les agents sont supposés communiquer à un niveau élevé en ce sens que les contenus de la communication sont des affirmations sur l'environnement ou sur la connaissance de l'agent. C'est une caractéristique fondamentale qui permet de distinguer la communication entre agents de simples interactions entre entités. On suppose une caractérisation abstraite des agents pour laquelle certaines capacités des agents sont décrites en termes d'attitude mentale de l'agent. Cette spécification caractérise un agent comme étant capable d'être décrit comme s'il possédait des attitudes mentales de :

- **croyance** : l'agent accepte une série de propositions comme étant des affirmations vraies.
- **incertitude** : l'agent n'est pas sûr que certaines propositions soient vraies.
- **intention** : l'agent désire que certaines propositions soient vraies alors que pour l'instant elles sont acceptées comme étant fausses. Un agent qui adopte une intention formera un plan d'action pour transformer l'état du monde indiqué par son choix.

Au niveau du développement orienté agent, la liste des outils de développement est exhaustive [2] : Jade, JAFMAS, AgentTool, AgentBuilder, JACK, ZEUS, ...

Au niveau des outils de développement orienté agent, un grand problème se présente : le problème de standardisation. Nous avons une multitude d'outils de développement d'agent chacun utilisant sa propre méthodologie, or chacun de ces outils formate les résultats de telle sorte qu'il serait impossible de développer un système multiagent avec l'utilisation de plusieurs outils différents. Ce problème rend impossible le « mélange d'outils » (tool mix) qui est un principe important au niveau des outils de développement orienté objet. Par exemple, une application développée par l'intermédiaire de l'outil « JACK » ne pourra pas être réutilisée en utilisant l'outil « ZEUS » malgré le fait que tous deux utilisent le langage de programmation java. Par contre, au niveau des outils de programmation orientée objet, ce problème ne se pose pratiquement pas. Par exemple on peut utiliser le logiciel TOGETHER3 pour effectuer un diagramme de classe puis ensuite utiliser le logiciel RATIONAL ROSE pour

reprendre le même diagramme. Ainsi les aspects vérification et la validation ne sont pas poussés au niveau de la communication inter-agent.

Les agents ne peuvent pas directement prendre part à une conversation juste en échangeant des messages. L'analyse de plusieurs conversations humaines montre qu'il y a souvent une routine suivie (on peut prendre l'exemple des conversations téléphonique).Le niveau mental de la sémantique des actes communicatifs est trop complexe pour déterminer les réponses possibles à un message si on se limite au raisonnement des états mentaux. Un agent doit implémenter des procédures de décisions traçables qui permettront la sélection et la production de messages (au niveau de la communication inter-agent) appropriés à ses intentions : les protocoles de communication (conversation).

2.1.1 Problèmes au niveau de la communication

Dans cette section nous présenterons un ensemble de problèmes qui sont importants au niveau du développement de langages et de théorie de communication agent. Au niveau des langages de communication inter-agent, la présentation va s'effectuer en passant respectivement par les points suivants, notamment la théorie d'agence, la sémantique, l'aspect vérification, les ontologies, la complétude des messages et les protocoles de communication.

2.1.1.1 Théorie d'agence

Un grand problème de recherche au niveau de la communication inter-agent inclut le lien entre la sémantique du langage de communication et la théorie d'agence qui règle et définit le comportement de l'agent. Afin que les messages ACL soient formellement logiques, ces deux théories doivent être alignées. Une théorie d'agence est un modèle formel général qui indique quelles actions un agent peut ou devrait effectuer dans diverses situations [3]. Les théories d'agence pour les agents logiciels sont habituellement basées sur un petit ensemble de primitives dérivées des attitudes de proposition de la philosophie (croyance, désir, et intention qui ont mené à l'architecture BDI) et un ensemble d'axiomes ou de schéma d'axiome qui définissent leurs relations. Une théorie complète d'agence inclut également la stratégie générale de raisonnement

d'un agent, le modèle déductif, la théorie d'action et de causalité, son niveau de satisfaction, de planification et de but, son système de dynamique de croyance et de révision, et ainsi de suite. Le comportement communicatif des agents figure parmi les comportements réglés par une théorie d'agence. Pour cette raison, les théories sémantiques qui définissent la signification d'un message ACL doivent être liées aux entités fournies par la ligne de base de la théorie d'agence des agents. Les versions en cours de KQML et de FIPA-ACL manipulent le lien entre la théorie sémantique et la théorie d'agence en faisant appel à une version simplifiée de la théorie d'acte de la parole du langage naturel (à l'origine développée par Searle (1969)). Dans cette approche, la communication inter-agent est traitée comme type d'action qui affecte le monde de la même manière que les actes physiques affectent le monde. Avec précision, les types de message au niveau de la communication inter-agent sont considérés comme des actes de la parole, qui sont décrits et à leur tour définis en termes de croyance, désirs et intentions(architecture BDI d'un agent: belief, desire, intention). La sémantique de FIPA-ACL est basée sur des notions mentales telles que la croyance et l'intention, et traite la transmission de messages inter-agent comme type d'action. Des disparités entre la théorie d'agence et la théorie sémantique peuvent se produire quand la théorie d'agence autorise les actions communicatives qui ne sont pas exprimables au niveau de la sémantique des langages de communications.

La condition de sincérité au niveau de l'utilisation d'un langage de communication agent est un tel exemple. Les théories sophistiquées d'agence permettent souvent à des agents d'agir avec l'intention de tromper si ça permet de promouvoir leurs buts. Cependant, afin de rendre la sémantique de message aussi utile que possible, la plupart des théories sémantiques ACL (telles que les théories de KQML et de FIPA-ACL) exigent que les agents n'emploient jamais cette condition pour affirmer quelque chose qu'ils ne croient pas eux-mêmes. Ceci est un renforcement du principe analogue des humains : au niveau d'une communication, nous ne supposons pas forcement que nos interlocuteurs nous mentent. Mais ceci rend également possible qu'un agent pourrait désirer communiquer quelque chose tandis que les conditions préalables sémantiques du message ACL l'interdisent de le faire. Pouvons-nous réellement présupposer que les autres agents seront également sincères au niveau de leurs communications ? Vont-ils à la

place dévier de la sémantique priée toutes les fois que ce sera commode ? Un grand problème réside au niveau de la vérification de la sincérité d'un autre agent.

Une autre prétention de ce type implique la capacité d'un agent à observer de manière effective les effets des actions d'un autre agent. Appliqué à la communication inter-agent, il est souvent pris pour acquis que les voies de transmission inter-agent sont exemptées d'erreurs. Les systèmes multiagents supposent par habitude que tous les messages par la suite arrivent à leurs destinataires prévus et ne sont pas interceptés par l'environnement (ou par des acteurs malveillants) en transit. Souvent, on suppose encore que l'ordre des messages qui sont envoyés à la même destination ne change pas pendant le transport. Selon le contexte d'exécution de l'agent, ces prétentions peuvent ne pas être appropriées. De plus, ne pas oublier l'existence de la possibilité, que la simplification des prétentions dans les langages de communication pourrait empêcher certains types de comportement souhaitable ou raisonnable relativement à la théorie d'agence d'un agent.

2.1.1.2 Problèmes au niveau de la sémantique

Les environnements ouverts tels que l'Internet créent des conditions où les systèmes interactifs peuvent librement agir les uns sur les autres. Tandis que les emplacements courants du Web (Internet) sont essentiellement statiques et que l'interaction est conduite par des utilisateurs humains, les développements tels que des services Web, des réseaux point à point (P2P : point to point) et des normes de commerce électronique (« e-business ») permettent d'établir des interactions complexes. La sémantique explicitement partagée permet aux agents de raisonner avec souplesse au sujet de leur communication qui leur permet de réagir dans leur environnement de manière plus efficace. Cependant, comment ces systèmes communiqueront-ils ? En particulier, comment les réalisateurs traiteront-ils la complexité croissante des tâches à traiter, de la *flexibilité* (traitant l'hétérogénéité dans l'environnement) et de *l'adaptabilité* que leurs systèmes sont susceptibles d'exiger ? En outre, comment s'assureront-ils que leurs systèmes communiquent de manière assez significative pour soutenir de véritables interactions automatisées ? Le problème de la sémantique touche plusieurs aspects de la communication, notamment les protocoles de communication [AUML], les langages de

communication [FIPA-ACL][KQML], le langage des contenus des messages (content expressions) [KIF][FIPA-SL] et les ontologies partagées[ONTOL].

2.1.1.3 Problèmes au niveau de la vérification des conversations

Bien que la remarque ait déjà été faite ci-dessus, ça vaut la peine de préciser explicitement que la vérification de la sémantique d'un langage de communication comme la vérification d'une instanciation d'un protocole selon des spécifications de protocoles est un problème sous-estimé. La seule publication jusqu'ici abordant explicitement la question de la vérification de la sémantique d'un langage de communication est celle de Wooldridge (2000). Si la sémantique d'un langage de communication a été donnée dans un certain genre de logique, la vérification pour déterminer si les agents qui emploient ce langage se conforment réellement à la sémantique et devient une preuve de la sémantique du langage à partir de la sémantique des programmes des agents.

Ici nous rencontrons le premier problème. Bien qu'il soit probablement possible de donner une sémantique formelle de n'importe quel programme, habituellement ceci peut seulement être fait en termes d'un certain genre de logique temporelle. Cela n'inclut donc pas des concepts tels que la croyance, l'intention, les buts, etc... Si l'exécution d'un agent était faite strictement sur la base d'une théorie de BDI (belief, desire, intention), il pourrait y avoir une interprétation possible en ces termes, mais cette interprétation serait rarement unique et non ambiguë. Ainsi, il y a deux choix possibles à faire. Soit la vérification d'un langage de communication peut seulement être faite quand les systèmes d'agent se conforment complètement et formellement aux concepts mentaux utilisés dans la sémantique des théories d'agent, soit la sémantique des langages de communication doit également être donnée dans une logique plus simple que les logiques multimodales généralement utilisées (par exemple : KQML et FIPA ACL).

Le deuxième problème, relié à la vérification (mentionné ci-dessus) est celui de la vérification d'un protocole de communication. Même lorsque les spécifications des protocoles sont données par un diagramme à états finis, il est difficile de vérifier si un certain protocole se comporte selon ce diagramme. Le problème devient plus dur quand

des réseaux de Pétri sont employés au niveau de la spécification des protocoles. En raison du parallélisme inhérent qui peut être exprimé au niveau de ce formalisme, le problème de vérification peut seulement être résolu dans certains cas secondaires ordonnés. C'est certainement un problème pratique important pour les systèmes ouverts d'agent. Lorsque les agents décident d'employer un certain protocole pour leur interaction, on veut être sûr qu'ils mettent en application le protocole de la même manière et que l'interaction résultante est conforme aux spécifications. Ceci représente un des piliers de l'industrie de la standardisation et devrait également être un souci principal pour les interactions entre agents.

2.1.1.4 Problèmes au niveau de l'ontologie

Les ontologies ont été développées pour fournir des vocabulaires spécifiques dépendants du domaine d'application pour la communication entre les agents. Une ontologie définit les concepts et les relations qui existent entre les mots d'un vocabulaire formel pour les agents qui l'utilisent. Il faut noter que les agents d'un système multiagent partagent la même ontologie (le même vocabulaire) mais ceci ne veut pas pour autant dire qu'ils possèdent la même base de connaissances.

Un problème qui est étroitement lié à la sémantique des langages de communication est le traitement approprié des ontologies au niveau des langages de communication. FIPA-ACL et KQML incluent un élément qui est employé pour identifier la source de vocabulaire utilisé dans le contenu de message. Ceci est conçu pour rendre ces langages de communication indépendants des vocabulaires particuliers d'application, et pour donner aux destinataires des messages une manière d'interpréter les termes non logiques au niveau du contenu des messages. Dans les spécifications originales de KQML, cet élément a été conçu pour se rapporter à une ontologie indiquée dans Ontolingua (« Ontolingua » est un logiciel qui fournit un environnement de collaboration distribué pour passer en revue, créer, éditer, modifier, et employer des ontologies). Dans *FIPA-ACL*, la sémantique de l'étiquette d'ontologie est efficacement définie pour l'utilisateur. Évidemment, une ontologie qui a une large couverture, concernant son domaine, et extensible, est nécessaire dans beaucoup de langages de communication. Ainsi, une ontologie qui exhibe une couverture de son domaine permet

aux agents multiples de partager leurs connaissances dans plusieurs contextes. Il est cependant très important de garder à l'esprit qu'une large couverture peut mener à une ontologie volumineuse et ainsi les agents peuvent passer beaucoup de temps à essayer de trouver la signification du contenu de leur langage de communication au lieu d'interagir les uns avec les autres. Une bonne ontologie devrait également être extensible afin de permettre aux concepteurs d'ajouter de nouveaux éléments. Finalement, une ontologie doit être dépendante de son domaine ("domain-dépendant") et ses rapports devraient montrer clairement leur pertinence avec ce domaine.

La manière avec laquelle un agent se servirait des spécifications des ontologies de KQML ou de FIPA-ACL pour interpréter des parties peu familières d'un message ACL n'a jamais été définie de manière précise. Le fait d'approvisionner une étiquette d'ontologie ne résout pas le problème de la façon dont les agents acquièrent et emploient la base de connaissance ontologique commune qui est un préalable à une bonne communication. Ceci représente un problème particulièrement aigu dans les systèmes ouverts qui incluent des agents basés dans différents organismes. Au niveau des problèmes liés à l'étude des significations et au raisonnement avec un nouvel ensemble de terminologie, il devrait exister des règles de traduction qui permettent de convertir des termes significatifs d'une ontologie vers une autre. Bien qu'un fonctionnement humain avec des représentants de chaque communauté terminologique puisse souvent hacher hors d'un ensemble de règles satisfaisant, il est presque impossible de construire ces règles entièrement de manière automatique (Klusch 1999, Papazoglou 1997). En conséquence, les agents peuvent seulement entièrement communiquer que s'ils partagent déjà une ontologie commune, ou si un ensemble de règles de préexistence de traduction est fourni. Le problème ontologique général fait toujours l'objet d'une recherche active.

2.1.1.5 Complétude des types de message ACL

Quand les agents hétérogènes agissent les uns sur les autres au moyen d'un langage de communication, la signification d'un tel échange est caractérisée par des actes communicatifs. Selon la philosophie de la langue, tous ces actes tombent dans une des catégories suivantes [5]:

• "*Assertifs*" : actes qui représentent un état de la question, mettent en jeu principalement les croyances du locuteur ou de l'allocutaire concernant l'état des choses assertées et ce à divers degrés.

• "*Directifs*" : actes dont la commande a pour but de demander à l'allocutaire d'exécuter une certaine action.

• "*Engageant*" (commissives) : comme son nom l'indique, actes qui font intervenir la notion d'engagement.

• "*Expressifs*" : actes qui expriment un certain état psychologique, mettent en jeu principalement les émotions.

• "*Déclaratifs*" actes qui expriment des notions d'intentions

• "*Permissives*" : actes qui expriment la permission

• "*Prohibitives*" : actes qui expriment l'interdiction

Toutes ces catégories devraient-elles également être couvertes par la communication inter-agents ? Est ce que seulement quelques unes de ces catégories pourraient suffire ? Quand pouvons-nous dire qu'un ensemble de types de message d'agent est complet ? Vu qu'un langage de communication peut être employé dans des contextes communicatifs arbitraires, un but important est que leur ensemble de base de types de message soit suffisant pour exprimer tous les genres possibles d'intention communicative qui sont permis par la théorie fondamentale d'agence. Sans un ensemble complet de messages, les agents et leurs développeurs peuvent se trouver dans des situations où ils seront forcés d'inventer des significations ad hoc additionnelles pour certains messages issus de la communication inter-agent, avec un déclin propre au niveau de l'interopérabilité. Ainsi par exemple, KQML et FIPA-ACL ont une couverture limitée puisque tous les primitifs sont des actes assertifs ou des actes directifs. Ainsi, au moins la

catégorie importante des actes de discours engageants (commissives) qui sont employés pour donner des engagements à une ligne de conduite, est absente. Dans FIPA-ACL, nous pouvons simuler des actes de discours engageants en utilisant d'autres performatifs. Cependant, il sera difficile d'obtenir la sémantique exacte à laquelle nous nous attendons pour des actes de discours engageants. Ceci signifie que les langages de communication inter-agent populaires (KQML + FIPA) sont incapables d'exprimer toutes les intentions possibles des agents au niveau des théories puissantes d'agence, parce que plusieurs classes de performatifs sont absentes au sein des deux langages. L'effet pratique de ces omissions est limité parce que KQML et FIPA-ACL sont des langages de communication extensibles : les utilisateurs sont libres d'inventer de nouveaux performatifs spécifiques à l'application aussi longtemps qu'ils ne vont pas à l'encontre de l'ensemble standard. Cependant, la sémantique de ces nouveaux performatifs peut être définie différemment par différents groupes et peut avoir comme conséquence le développement de différents dialectes incompatibles de ces langages de communication.

2.1.1.6 Règles Conversation (ou protocoles de communication)

Dans les sections précédentes, nous avons discuté des problèmes de recherches au niveau de la communication. Ces problèmes de communication sont principalement liés à la génération et à l'interprétation de différents messages. Un sujet final que nous adresserons est comment combler le trou qui sépare les différents messages et séquences de messages (au niveau d'une conversation) prolongées (extended), qui surgissent entre les agents. Au niveau du domaine multiagent, chaque agent doit implémenter des procédures qui permettent à chaque agent de choisir et produire des messages qui sont appropriés à ses intentions au sein de la communication. Ce n'est pas purement un problème d'assortir la sémantique ACL aux intentions des agents : excepté dans le plus limité des systèmes multiagent, ces procédures de décision doivent également prendre en considération le contexte des messages antérieurs et d'autres paramètres au niveau des systèmes multiagent. De manière paradoxale, la prise en considération de ce contexte peut réellement simplifier la complexité informatique du choix de message pour un agent. En s'engageant dans des conversations pré planifiées ou stéréotypées, une grande partie

de l'espace de recherche des réponses possibles des agents peut être éliminé, tout en étant toujours conforme à la sémantique des langages de communication inter-agent. Les spécifications de ces conversations sont accomplies par l'intermédiaire des règles de conversation (protocoles de conversation). On a réussi à réunir plusieurs définitions expliquant qu'est-ce qu'un protocole de communication :

1. *Contraintes sur les séquences des messages sémantiquement logiques menant à un but [13].*

2. *Caractéristiques déclaratives qui régissent des communications entre les agents logiciels en utilisant un langage de communication inter-agent [11].*

3. *Ensemble de règles qui guident l'interaction qui a lieu entre plusieurs agents. Ces règles définissent la validité des messages au niveau d'un état quelconque [18].*

Pour compléter les définitions énoncées ci-dessus, Les protocoles de communication représentent les interactions permises. Les protocoles sont essentiels dans les applications telles que le commerce électronique où il est nécessaire de contraindre les comportements des agents. Les protocoles de communication permettent de gérer les séquences (messages) échangées entre agents au sein d'une conversation. En raison de cet avantage informatique, pratiquement tous les systèmes multiagent utilisent un certain type de couche conversationnelle explicite ou implicite. Il y a une grande différence entre la théorie et la pratique dans ce secteur. La terminologie et les approches théoriques sont en cours d'établissement, les approches et les métriques formelles sont jusqu'à présent assez floues, et le rôle de la recherche dans la théorie pragmatique et de discours du langage naturel est en cours d'évaluation. La grande variété des approches qui sont discutées dans cette section montre la nature exploratoire de la recherche. La théorie essaye de trouver un terrain d'entente entre les protocoles complètement fixes (utilisés dans les systèmes répartis) et l'emploi de quelques règles de niveau élevé qui peuvent générer des protocoles prêts à être utilisés. Les protocoles complètement fixes sont habituellement trop rigides pour être employés dans un environnement multiagent où ils deviennent trop complexes (tenant compte de chaque exception possible qui pourrait se produire).

Aspect sémantique au niveau de la conversation

Une des questions importantes au niveau de la recherche dans le domaine multiagent est la question théorique dans le domaine concernant le lien entre la théorie sémantique des langages de communication et la conversation. Les propriétés des conversations inter-agents (flux global de l'information, établissement des engagements), sont des conséquences des différentes significations des messages qui composent la conversation. De plus, les sémantiques conversationnelles sont primaires, et en raison de la dépendance de la sémantique des langages de communication à l'égard du contexte, le même message pourrait avoir des significations légèrement différentes une fois utilisées dans le contexte de différentes conversations inter-agent. Dans le contexte précis de cette optique, un langage de communication est considéré comme une sorte de conversation entre les agents logiciels et pas comme un ensemble d'actes de la parole. Sa sémantique est la sémantique d'une conversation et elle ne peut pas être réduite à la conjonction ou à la composition de la sémantique des actes de la parole (Vongkasem and Chaib-draa 2000). Si on prend la sémantique conversationnelle ou la sémantique des langages de communication comme point de départ, cela affectera les réponses à plusieurs questions, telles que :

- Qu'est-ce qu'une règle de conversation? Que peut-on considérer (et ne pas considérer) comme règle de conversation? Quelles propriétés importantes d'interaction inter-agent les règles de conversation devraient capturer ?

- Comment les règles de conversations se distinguent ? Quand peut-on considérer que deux conversations d'agents sont issues de la même règle ? Existe-t-il des classes d'équivalence intéressante au niveau des règles de conversation ? Y a-t-il un type utile de hiérarchie au niveau des règles de conversation ? Comment prolonger (extend) ces règles ? Comment devraient- elles être composées ? ect.

Aspect représentation des protocoles de communication

Une fois qu'on s'est mis d'accord sur une perspective théorique de base sur le lien entre les règles de conversation et la sémantique des langages de communication, un certain nombre de questions techniques demeure. Par exemple, il y a plusieurs structures différentes possibles au niveau du langage de spécification d'une règle de conversation, en allant des structures de transition (transition net) comme des machines à états finis et des réseaux de Pétri, vers de types divers à base logique, des arbres représentant des sous-tâches, et des systèmes de spécifications de protocoles sous forme de réseau. Ces formalismes changent considérablement au niveau de la rigidité conversationnelle qu'ils nécessitent, des modèles de simultanéité qu'ils soutiennent, de la complexité informatique au niveau leur exécution, et la disponibilité des outils et des techniques pour permettre la vérification des propriétés analytiques des règles représentées au sein de la conversation. Finalement, l'utilisation des règles de conversation (qui guident le comportement des communications inter-agent) engendre une foule de questions pratiques :

- Comment les règles de conversation devraient-elles être implémentées dans le cadre des systèmes multiagent ?
- Les règles de conversation devraient-elles être téléchargées à partir d'une librairie commune préconçue au niveau du « programme interne » des agents, ou dérivées à partir d'axiomes conversationnels au niveau de l'exécution ?
- Comment des règles de conversation peuvent être négociées, et les règles peu familières apprises ?
- Finalement, comment les règles de conversation peuvent-elles être intégrées avec d'autres règles, plans, et règles qui définissent le comportement des agents ?

2.2 Synthèse de la problématique

Au niveau de la communication inter-agent, les agents communiquent dans le cadre d'un protocole de communication précis. Sachant que les agents communiquent dans un but précis qui est de réaliser leurs tâches au niveau du bon suivi des protocoles de communication par les agents communicants, les aspects sémantiques de la vérification (bonne utilisation des actes de discours dans la conversation) et de la validation (respect de l'ordre d'apparition des messages au sein de la conversation) possèdent des lacunes. De plus plusieurs questions, au niveau de l'analyse, de la conception et de l'implémentation optimale des protocoles de communication afin de bien supporter les interactions inter-agents, demeurent.

Ainsi, après avoir présenté l'ensemble des problèmes actuels concernant la communication inter-agent, on compte traiter le sous-problème qui est celui des règles de conversations. Nous allons nous concentrer sur le sous-problème des protocoles de communications (ou règles de conversations), afin de réduire le fossé qui sépare la théorie de la pratique au sein de la communication inter-agent, par l'intermédiaire de la conception et du développement de l'environnement informatique qui constitue l'objectif de mon mémoire de maîtrise (« Outil logiciel de développement et d'expérimentation de protocoles de Communication pour les systèmes multiagents »).

Chapitre 3. ÉTAT DE L'ART

3.1 Langages et protocoles de communication au sein des outils existants.....46

3.1.1 KQML (Knowledge Query and Manipulation Language).....47
3.1.2 FIPA ACL [Foundation for Intelligent Physical Agents].....49
3.1.3 Actes de discours / Performatifs (Speech acts).....52

3.2 Problèmes au niveau des langages de communication.....56

3.2.1 Sémantiques des langages de communication.....56
3.2.2 En résumé.....60

3.3 Règles de conversation.....61

3.3.1 Que sont les règles de conversation ?.....62
3.3.2 Limites des règles de conversation.....63

3.4 Nouveaux langages basés sur les engagements et les jeux de dialogue.....68

3.4.1 Un environnement pour les sémantiques des performatifs KQML.....68
3.4.2 Protocoles bases sur les « *engagements sociaux* ».....74

3.5 Différentes Structures de modélisation de protocoles de conversations.....109

3.5.1 Présentation des différentes structures.....109
3.5.2 En conclusion.....113

Chapitre 3 : ÉTAT DE L'ART

Au niveau de la problématique de ce document (chapitre 2), on a présente non seulement l'ensemble des problèmes qui affectent la communication inter-agent, mais on a également indiqué, au niveau de la synthèse de la problématique (section 2.2 de la problématique), l'aspect qu'on compte exploiter au niveau de la conception et du développement de l'environnement constituant l'objectif de ce mémoire (« protocolBuilder »).

Dans un premier temps, on va effectuer une présentation des différents langages de communication et protocoles associés (section 3.1). En second lieu, nous allons revenir aux différentes failles décelées au niveau de ces langages (section 3.2), aux règles de conversations de manière plus approfondi (section 3.3), suivi de la présentation quelques nouvelles approches prometteuses (sections 3.4), pour finalement effectuer la présentation des différentes structures pouvant être utilisées pour modéliser des protocoles de communication (machines à états finis, graphes de Dooley, réseaux de pétri).

3.1 Langages et Protocoles de communication au sein des outils existants

Lorsque la notion de groupe d'agents est apparue, les recherches sur des mécanismes permettant aux agents de communiquer entre eux se sont amorcées. Celles-ci ont mené à deux langages de communication et de représentation de l'information qui sont devenus des « standards » au niveau des systèmes multiagents : KQML et FIPA ACL. Les langages de communication d'agent (LCA) sont employés pour la communication inter-agent. Ces langages sont basés sur la théorie d'acte de la parole, qui vient de la psychologie. C'est une tentative de formaliser la langue utilisée par les humains de manière à réaliser des tâches journalières comme des demandes, des ordres, des promesses, et ainsi de suite.

3.1.1 KQML (Knowledge Query and Manipulation Language)

3.1.1.1 Présentation

On considère *KQML* comme un langage de traduction, permettant de baser la communication sur un vocabulaire partagé plutôt que sur la sémantique. Autrement dit, *KQML* permet de faire communiquer des entités matériellement hétérogènes possédant des ontologies communes. Plus précisément, KQML est un langage destiné à échanger des informations et des connaissances entre agents. KQML est un format de message et un protocole de manipulation pour soutenir l'exécution mettant en commun les agents au niveau de la communication. KQML fournit un symbolisme de haut niveau dans la description de l'acheminement du message vers son destinataire et dans le contenu du message envoyé à l'agent (par le biais de *KIF*, par exemple).

Le langage *KQML* a été conçu pour permettre l'interaction performative et interrogative entre bases de connaissances. Il a été élaboré par le groupe *THE ARPA KSE* dans l'optique d'une utilisation comme langage d'interaction entre :

- une application et un système intelligent
- plusieurs systèmes intelligents

Pour le partage des connaissances dans un contexte coopératif.

Notamment, KQML définit des performatifs de communication de haut niveau permettant à des agents potentiellement hétérogènes d'échanger des informations structurées. *KQML* consiste principalement en un ensemble extensible de définitions d'actes performatifs. Ces performatifs décrivent les opérations qu'il est possible d'effectuer sur les connaissances contenues par une base de connaissances extérieure.
Les performatifs comportent un substrat sur lequel développer les modèles de plus haut niveau de l'interaction d'inter-agent tels que le contrat net et négociation. Ce langage propose aussi une architecture de base pour le partage d'informations à l'aide de composants spécifiques nommés *Facilitators*, coordonnant les actions des participants à une communication.

3.1.1.2 Structure des messages KQML

La syntaxe de base de *KQML* est la suivante :

```
(KQML-performative
    :sender     <word>
    :receiver   <word>
    :language   <word>
    :ontology   <word>
    :content    <expression> ...)
```

Le langage présente une syntaxe à trois niveaux : communication, message et contenu. Il s'agit en fait d'un protocole de base extensible, mais qui n'inclut pas comme *KIF* des échanges d'informations

Dans l'exemple précédent, on peut reconnaître les trois niveaux auxquels appartiennent les performatifs de *KQML* ;

- `:sender`, `:receiver` définissent la *communication* ;
- `:language`, `:ontology` définissent le *message* ;
- `:content` définit le *contenu.*

3.1.1.3 Désavantages

Un des désavantages de KQML est cependant son orientation vers la communication point à point entre agents (avec potentiellement l'utilisation d'agents facilitateurs pour relayer un message), ce qui le rend peu adapté à une utilisation dans un contexte de canaux de communication partagés, non fiables et possédant une faible bande passante. Étant proche du langage naturel, KQML permet notamment une interface homme-machine plus humaine.

3.1.2 FIPA ACL [Foundation for Intelligent Physical Agents]

3.1.2.1 Présentation

Le langage *ACL* de la *FIPA* se base sur un modèle simple d'abstraction de la communication entre agents. Les agents sont supposés communiquer à un niveau élevé en ce sens que les contenus de la communication sont des affirmations sur l'environnement ou sur la connaissance de l'agent. C'est une caractéristique fondamentale qui permet de distinguer la communication entre agents de simples interactions entre entités.

On suppose une caractérisation abstraite des agents pour laquelle certaines capacités des agents sont décrites en termes d'attitude mentale de l'agent. Cette spécification caractérise un agent comme pouvant être décrit comme s'il possédait des attitudes mentales de croyances, incertitudes et intention. Au niveau de la **croyance,** l'agent accepte une série de propositions comme étant des affirmations vraies. L'agent ayant des **incertitudes** n'est pas sûr que certaines propositions soient vraies. Au niveau de l'**intention, l'agent** désire que certaines propositions soient vraies alors que pour l'instant elles sont acceptées comme étant fausses. Un agent qui adopte une intention formera un plan d'action pour transformer l'état du monde indiqué par son choix.

De plus, les agents comprennent et sont capables d'exécuter certaines actions. Dans un système distribué, un agent sera typiquement capable de remplir complètement ses intentions en influençant d'autres agents afin d'exécuter certaines actions.

Le geste d'influencer les actions d'autres agents est exécuté par une classe spéciale d'actions appelée **actes de communication** (actes de discours/performatifs, section 3.1.3). Un acte de communication est exécuté par un agent en direction d'un autre. Le mécanisme d'exécution d'un acte de communication est précisément celui d'envoyer un message encodant l'acte.

3.1.2.2 Structure des messages FIPA

Un message issu du langage de communication FIPA contient un ensemble d'un ou plusieurs éléments. Plus précisément, les éléments nécessaires pour une communication efficace d'agent changent selon la situation; Le seul élément obligatoire dans tous les messages est le performatif, bien que la plupart des messages issus de la communication inter-agent contiennent les éléments spécifiant l'envoyeur (« ***sender*** »), le récepteur (« ***receiver*** ») et le contenu (« ***content*** »); Si un agent ne reconnaît pas ou est incapable de traiter un ou plusieurs éléments ou leurs valeurs, il peut répondre avec un message « ***not-understood*** » (non-compris) approprié.

Les implémentations spécifiques peuvent inclure des éléments définis par l'usager au niveau du message autres que ceux définis au niveau du tableau 1. Les sémantiques des éléments définis par l'usager ne sont pas définis par FIPA, et aucune interprétation de ceux ci de la part de FIPA n'est nécessaire afin d'être conforme.

Quelques éléments du message peuvent être omis lorsque leur valeur peut être déduite selon le contexte de la conversation. Par contre, FIPA ne définit aucun mécanisme pour traiter de telles situations, ainsi les implémentations qui omettent quelques éléments au niveau du message ne fournissent aucune garantie d'interopérabilité.

L'ensemble complet d'éléments au niveau d'un message FIPA est présenté au niveau du tableau 1 sans tenir compte de leur codage spécifique dans une implémentation. Chaque représentation de message contient une description précise de la syntaxe du code.

Chaque spécification de représentation de message contient des descriptions précises de la syntaxe pour le codage de message ACL (FIPA) basés sur XML, chaînes de caractères et plusieurs autres combinaisons.

Element	Category of Elements
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Tableau 1: Paramètres des messages FIPA ACL

Les termes suivants sont utilisés pour définir l'ontologie et la syntaxe abstraite de la structure des messages FIPA ACL :

Frame : Nom obligatoire de cette entité, et doit être utilisé pour représenter chaque instance de cette classe.

Ontologie : Nom de l'ontologie, dont le domaine du discours inclut leurs éléments décris dans le tableau.

Élément : Identifie chaque composante au sein du « frame ». Le type de l'élément est défini relativement à un codage particulier.

Description : Description en langage naturel des sémantiques de chaque élément.

Valeurs réservées : Liste de constantes définies par FIPA associes a chaque élément. Tous les éléments des messages FIPA partagent le « frame » et l'ontologie au niveau du tableau 2.

Frame	FIPA-ACL-Message
Ontology	FIPA-ACL

Tableau 2: Ontologie du message FIPA ACL

3.1.3 Actes de discours / Performatifs (Speech acts)

Au niveau d'une communication inter-agent, les agents comprennent et sont capables d'exécuter certaines actions. Dans un système distribué, un agent typique sera capable de remplir complètement ses intentions en d'influencer d'autres agents afin d'exécuter certaines actions. Le fait d'influencer les actions d'autres agents est exécuté par les actes de communication. Un acte de communication (acte de discours, performatif) est exécuté par un agent en direction d'un autre. Le mécanisme d'exécution d'un acte de communication est précisément celui d'envoyer un message encodant l'acte. Un message de KQML/FIPA ACL s'appelle un performatif (acte de discours), parce que le message est prévu pour effectuer une certaine action en vertu de l'envoi (la limite est de théorie d'acte de la parole). Le terme "performatif" provient de l'anglais « *to perform* » qui signifie exécuter ou accomplir quelque chose. Ce terme est utilisé pour indiquer que l'énoncé constitue simultanément l'action qui est exprimée. Les agents coopèrent à travers la communication. On suppose qu'au sein de chaque organisation, il existe un ensemble standard d'actes de discours qui définissent les actions communicatives disponibles. Ces discours sont représentés en tant que performatifs de langage de communication inter-agent. Les messages échangés se font par l'intermédiaire des performatifs (actes de discours) du langage de communication inter agent. (Au niveau de l'exemple ci-dessous au niveau de la figure 7, le performatif choisi pour représenter le message est « ***inform.*** » ayant pour but d'informer).

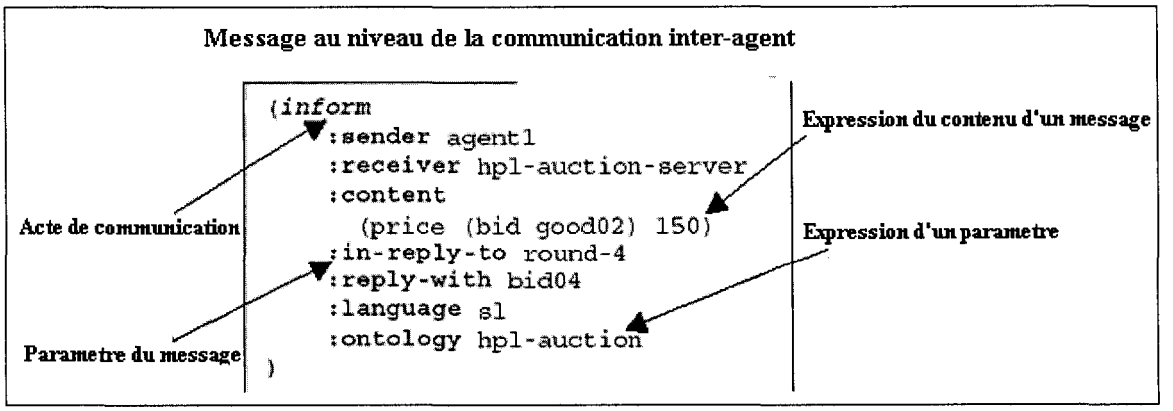


Figure 7: Structure des messages

3.1.3.1 Liste de performatifs (Actes de discours) du langage KQML

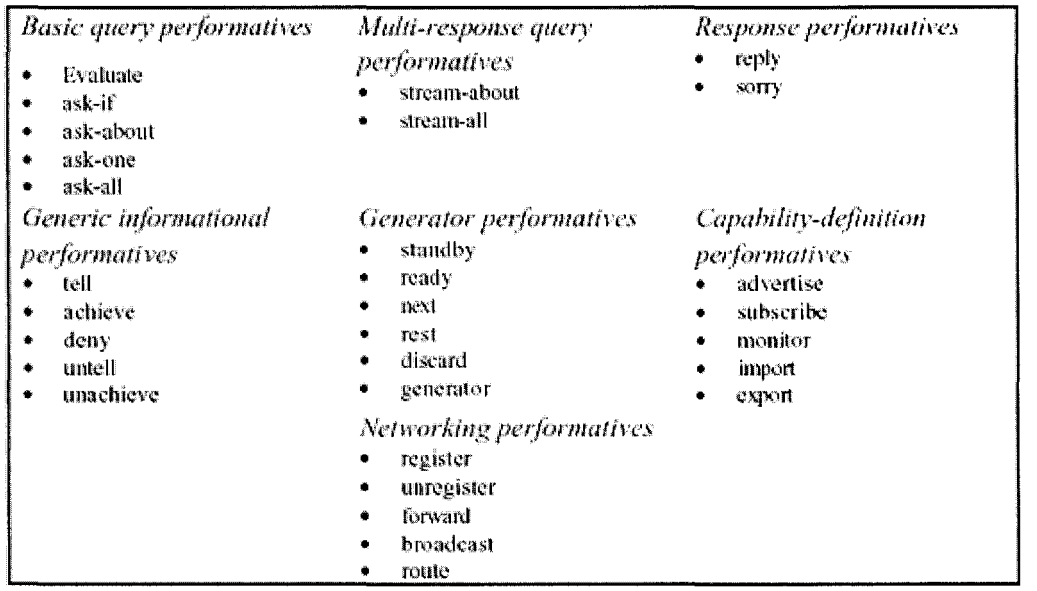


Figure 8: Liste des Actes communicatifs KQML

On peut voir la liste de tous les actes de discours (performatifs) de KQML au niveau de la figure 8. Les actes de KQML peuvent être classifiés en trois grandes catégories :

1. discours - performatifs utilisés pour l'échange d'informations et de connaissances (tel, untell, ask-if, ask-one, ask-all, stream-all, deny, insert, delete-one, delete-all, uninsert, undelete etc.)

2. interconnexion entre les agents pour faciliter l'obtention des informations et des connaissances (broker-one, recommend-one, recruit-one, broker-all, recommend-all, recruit-all, broadcast, register, unregister)

3. exception - des performatifs qui modifient le flux des informations et des connaissances (sorry, standby, error, ready, next, rest, discard etc.)

Basés sur le langage de communication standard KQML, plusieurs performatifs ont été retenus. Selon ce qui suit (table 3) nous allons utiliser les abréviations suivantes :

- **E** : L´agent émetteur,
- **R** : L´agent récepteur
- **C** : Le contenu du message
- **BVC** : La base virtuelle de connaissances

Performatif	Signification
achieve	E veut que R rende quelque chose vrai dans leur environnement
advertise	E fait connaître aux autres agents les capacités disponibles dans le traitement d'une performative
ask-about	E veut connaître toutes les propositions pertinentes dans les BVC de R
ask-if	E veut savoir si la réponse à la question précisée en C se trouve dans la BVC de R
ask-one	E veut que seulement R réponde à sa question C
break	E veut que R interrompe le flux établi par pipe
broadcast	E veut que R transmette à son tour la performative à toutes ses connexions
broker-all	E veut que R collecte toutes les réponses à la performative
broker-one	E veut que R l'aide à répondre à une performative
deny	la performative ne peut pas être appliquée à E
delete-all	E veut que R efface toutes les propositions qui s'apparient avec C de sa BVC
delete-one	E veut que R efface une proposition qui s'apparie avec C de sa BVC
error	E considère le message précédent de R comme mal formé
evaluate	E veut que R évalue (simplifie) C
forward	E veut que R transmette le message vers un autre agent
generator	la même signification que standby de stream-all
insert	E demande à R d'ajouter C à sa BVC
monitor	E veut que R actualise sa réponse à un stream-all
next	E veut la réponse suivante de R à un flux d'une performative
pipe	E veut que R transmette toutes les performatifs futures à un autre agent
ready	E est prêt pour répondre au performatif mentionné précédemment par R
recommend-all	E veut tous les noms des agents qui peuvent répondre à C
recommend-one	E veut le nom d'un agent qui peut répondre à C
recruit-all	E veut que R 'recrute' tous les agents capables de lui répondre à C
recruit-one	E veut que R 'recrute' un agent capable de lui répondre à C
register	E peut transmettre des performatifs à un certain agent
reply	communique une réponse attendue
rest	E veut le reste des réponses de R à une interrogation nommée précédemment
sorry	R ne peut pas fournir plus d'information
standby	E veut que R soit prêt pour répondre à une performative
stream-about	une version réponse multiple de ask-about
stream-all	une version réponse multiple de ask-all
subscribe	E veut que R actualise par la suite sa réponse
tell	E affirme au R que C est dans la BVC de E
transport-address	E associe un nom symbolique à une adresse de transport
unadvertise	l'action contraire de advertise
unregister	l'action contraire de register
untell	E affirme à R que C n'est pas dans la BVC de E

Tableau 3 : Signification des actes de discours du langage KQML

3.1.3.2 Liste des actes communicatifs du langage FIPA

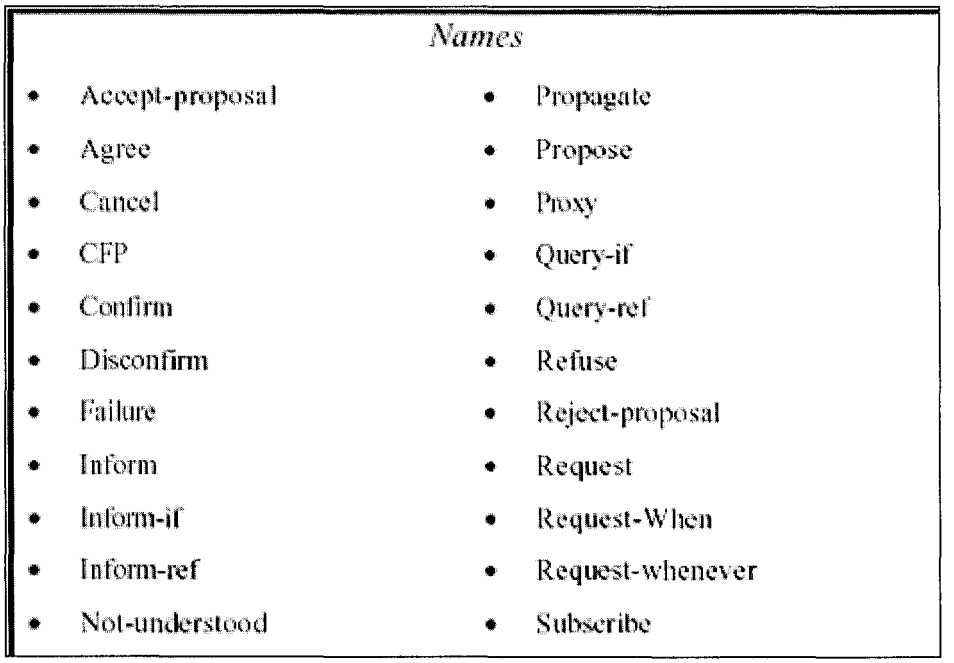


Figure 9 : Liste des actes communicatifs (FIPA ACL 2000)

On peut voir la liste de tous les actes de discours (performatifs) de FIPA au niveau de la figure 9. Basés sur le langage de communication standard FIPA, plusieurs performatifs ont été retenus :

•***Propose (proposer).*** Ceci est employé pour proposer à un agent une sous-tâche à réaliser.

• ***Counter-Propose (contre-proposition).*** Une contre- proposition est une autre sous-tâche qui satisfait partiellement le but initial d'une proposition. L'utilisation de cet acte de la parole peut avoir comme conséquence une séquence de contre-propositions entre le « proposeur » original et le répondant. Un exemple est:

• ***Accept and Reject (Accepter et rejetter).*** Ceux-ci sont employés pour signaler l'acceptation, respectivement le rejet d'une proposition ou la contre-proposition. Le rejet commence une nouvelle phase de négociation.

• ***Cancel (Annuler).*** Ceci décommande une proposition ou une contre-proposition précédemment admise

• ***Commit (engagement)***. Confirmation positive qu'un agent se met lui-même dans un état qui satisfera une proposition; finit également une phase de négociation ou de renégociation.

• ***De-Commit (annulation d'engagement))***. L'annulation d'un précédent « commit » (d'un engagement précédent).

• ***Satisfy (Satisfaction)***. Un agent annonce qu'un but demandé a été réalisé.

• ***Fail (Echec)***. Un agent informe que l'exécution d'un but commis a échoué.

3.2. Problèmes au niveau des langages de communication

3.2.1 Sémantiques des langages de communication

Au niveau des langages de programmation, les préconditions et postconditions peuvent être exprimés en terme de variables et leurs valeurs avant et après l'action voulue, car les types appropriés d'actions sont limités au niveau de la manipulation des valeurs des variables. Cependant, les actes communicatifs dans un langage de communication ne manipulent pas directement les variables et leurs valeurs. Ils sont conçus pour fonctionner à un niveau plus élevé d'abstraction donné par la théorie d'agence, et mentionnent les primitifs fournis par cette théorie. Par conséquent, les préconditions et les postconditions pour des actes communicatifs sont typiquement exprimées en termes d'attitudes mentales des agents impliqués. Par exemple, au niveau de KQML, la précondition du message « tell » de KQML stipule que l'expéditeur croit ce qu'il indique et qu'il sait que le récepteur veut savoir que l'expéditeur croit également ce qui a été indiqué. La postcondition d'envoyer le message « tell » est que le récepteur peut conclure que l'expéditeur croit le contenu du message. Les sémantiques du langage de communication Fipa-ACL sont basées sur une approche sémantique semblable qui implique la spécification des préconditions « faisables » et les effets rationnellement prévus d'un message.

3.2.1.1 Manques au niveau de la Sémantique

Bien que l'approche de préconditions/postconditions permette de fournir une signification minimale des messages dans un langage de communication, Il est souvent souhaitable de raffiner cette signification minimale de manière plus précise par rapport au contexte, menant ainsi à une tension dans la théorie sémantique des langages de communication. D'une part, il on voudrait que la sémantique soit assez flexible pour être applicables dans toutes les situations où les agents emploient le langage de communication, nous formulons donc des préconditions et des postconditions assez générales allant formellement dans le même sens que les sémantiques. D'autre part, les préconditions et postconditions résultantes sont souvent si abstraites qu'ils sont plus entièrement adéquats pour toutes les situations. De plus, il est souvent très difficile de vérifier si l'état de l'agent satisfait les pré et des postconditions. Ceci est en parti dû au fait que, malgré le fait que nous attribuions par habitude des attitudes mentales aux agents, les agents ne sont pratiquement jamais directement programmés avec ces concepts. Par exemple, comment pourrait-on vérifier, ayant les agents i, j et une proposition p, que : «l'agent i sait que l'agent j veut savoir si l'agent i croit p ? »

- **Sémantiques KQML**

Pour illustrer ces considérations concernant les pré et postconditions, considérons d'abord la sémantique de KQML. Ces sémantiques ont été exprimés en termes de préconditions, de postconditions et de « conditions de complétude » Les préconditions indiquent les états nécessaires pour que l'agent expéditeur envoie un performatif, et pour que l'agent récepteur l'accepte. Si ces préconditions ne « tiennent pas debout », alors une erreur va être générée. Les postconditions, d'autre part, décrivent les états de l'expéditeur après l'envoi réussi d'un performatif, et du récepteur après la réception et traitement du message. Les postconditions concernant l'expéditeur et le récepteur « tiennent debout », sauf si une erreur est générée comme réponse, pour indiquer l'échec de l'envoi ou du traitement du message.

Finalement, un état d'accomplissement indique l'état final, qui correspond généralement a l'atteinte de l'intention qui permet d'initier la conversation. Comme nous pouvons le voir, au niveau de KQML, préconditions, postconditions, et états

d'accomplissement décrivent les états mentaux des agents mais il n'y a aucune sémantique associée à ces états mentaux. Dans ce cas-ci, le « problème de sémantique » est juste transposé des performatifs aux états mentaux des agents.

- **Sémantiques FIPA-ACL**

La sémantique de FIPA ACL est soutenue par un langage formel appelé SL. Ce langage est une logique multimodal mesurée avec les opérateurs modaux pour la croyance, désirs, croyance incertaine, et les buts persistants (une forme d'intention). Elle est employée pour la sémantique de FIPA ACL attribuant à chaque acte (inform, request, ect.) des ensembles de formules de SL décrivant les conditions préalables de faisabilité et les effets rationnels de l'acte. Selon cette sémantique, l'acte d'informer, par exemple, où l'agent i informe l'agent j du contenu p, mène à ce qui suit:

(1) i croit que p se tient;

(2) i ne croit pas vraiment que le récepteur a n'importe quelle connaissance de la vérité p;

(3) i prévoit que l'agent de réception devrait également croire que la proposition est vraie.

Comparativement à KQML, fipa-ACL offre un petit ensemble de primitifs pouvant être combinés. Il possède également une sémantique formelle qui peut supporter l'interopérabilité. Ses limitations tournent autour du fait que sa sémantique minimale, repose seulement sur les états de croyance des agents de communication, n'offre aucun indice sur la façon dont d'impliquer les états mentaux du récepteur. En outre, Fipa-ACL a un contexte fixe avec l'agent expéditeur qui peut être un blocage à l'hétérogénéité (kone 2000).

- **Couverture des domaines**

Un problème relié à la sémantique des langages de communications est le bon traitement des ontologies. KQML et FIPA incluent un élément utilisé pour identifier la source du vocabulaire utilisé dans le contenu du message. Ceci a été conçu pour rendre les langages de communication indépendants des vocabulaires particuliers, et pour donner au récepteur un moyen d'interpréter les termes « non-logiques » au niveau du contenu du message.

Une ontologie possède un grand champ de couverture significative au(x) domaine(s) qu'elle couvre, et est nécessaire au niveau de plusieurs langages de communication. Une ontologie doit exhiber la « couverture de son (ses) domaine(s) afin de permettre aux multiples agents d'augmenter la qualité du partage de connaissances dans des contextes différents. Par contre, une large couverture peut avoir des effets pervers : les agents peuvent passer tout leur temps à chercher la signification du contenu de leur langage au lieu d'interagir entre eux. Une bonne ontologie devrait être extensible afin de permettre aux concepteurs d'ajouter de nouveaux éléments.

- **Agents dans des organisations différentes**

Au niveau des problèmes associés à l'étude des significations et des raisonnements, les ontologies utilisées par les différents agents doivent, d'une manière ou d'une autre, être intégrées. Disons qu'il devrait au minimum exister des « *règles de translation* » qui convertiraient les termes significatifs d'une ontologie à une autre.

Bien qu'un humain travaillant avec les représentatives de chaque communauté terminologique puisse souvent produire un ensemble de règles satisfaisant, il est presque impossible de construire ces règles de manière automatique (Klush 1999; Papazoglou 1997).

Ainsi, les agents ne peuvent communiquer parfaitement que s'ils partagent déjà une ontologie commune, ou si un ensemble de « règles de translation » préexistant est fourni.

3.2.2 En résumé

Pour récapituler nous pouvons reconnaître qu'un certain travail a été effectué sur la sémantique des langages de communication, mais que la sémantique de KQML et de Fipa-ACL est basée sur **l'agence mentale**, c.-à-d., actes de communication décrits en termes de croyance, intentions, désirs, et états mentaux semblables. Cependant, les agents ne sont pratiquement jamais programmés en utilisant directement de tels états mentaux. Ainsi, il est donc presque impossible de vérifier si les messages sont employés correctement par les agents et la différence entre la théorie et la pratique pratiquent dans les langages de communication est très grande.

3.3 Règles de conversation

Les règles de conversation (protocoles de conversation) sont des contraintes générales sur les séquences des messages sémantiquement logiques menant à un but, la cohérence du dialogue est ainsi assurée par ces contraintes [15]. Ceci facilite considérablement la tâche de calculer les différentes réponses possibles à un message donné. Mais l'utilisation de règles de conversations est en théorie intéressante : le lien entre la théorie sémantique du langage de communication et son mode de conversation est une des questions courantes. D'une part, comme expliqué en haut, il semble évident que les propriétés des conversations d'agent à grande échelle, et l'établissement des engagements, sont une conséquence des différentes significations des messages qui composent la conversation.

Dans la même optique, La sémantique des langages de communication est primaire, et chaque propriété conversationnelle dérive logiquement de la composition d'une certaine collection de propriétés sémantiques des différents messages et de leur séquence. D'autre part, il y a un fil significatif de recherches qui prend les séquences conversationnelles pour être sémantiquement primitif, et la signification précise des différents messages est nuancée par leur rôle dans la conversation globale. Fondamentalement, l'idée est de considérer la signification de l'acte communicatif à un niveau objectif, comme donnée par les réponses possibles à un message donné. Dans le même contexte, la sémantique conversationnelle est primaire, et en raison de la dépendance de la sémantique des langages de communications par rapport contexte, le même message pourrait avoir des significations légèrement différentes une fois utilisées dans le contexte de différentes conversations inter-agent. Plus précisément, un langage de communication est vu comme une sorte de conversation entre les agents et non comme ensemble d'actes de parole. Sa sémantique est la sémantique d'une conversation et elle ne peut pas être réduite à la conjonction ou à la composition de la sémantique de ses actes de la parole.

3.3.1 Que sont les règles de conversation ?

Les règles de conversation sont souvent modélisées par l'intermédiaire des machines à états finis (FSMs). Ces modèles sont souvent appelés protocoles. Les états de l'automate tracent les états possibles de la conversation après un message donné par les participants. Des règles soigneusement conçues et fortement complexes de conversation ont été proposées en utilisant ces techniques dans la littérature, et mis en application dans de vraies applications (p.e COOL [1]). L'exemple du protocole ***demande d'action*** (« *request for action* ») illustre ceci.

D'autres formalismes, qui vont être présentés au niveau de la section 3.5 du mémoire, peuvent être employés pour modeler ces règles de conversation:

(a) Réseaux de Pétri particulièrement bien adaptés aux conversations parallélisées (avec plus de deux participants à la conversation),

(b) Graphiques de Dooley, qui peuvent offrir une représentation compacte et précise de la conversation.

3.3.1.1 Demande d'action (Request for action)

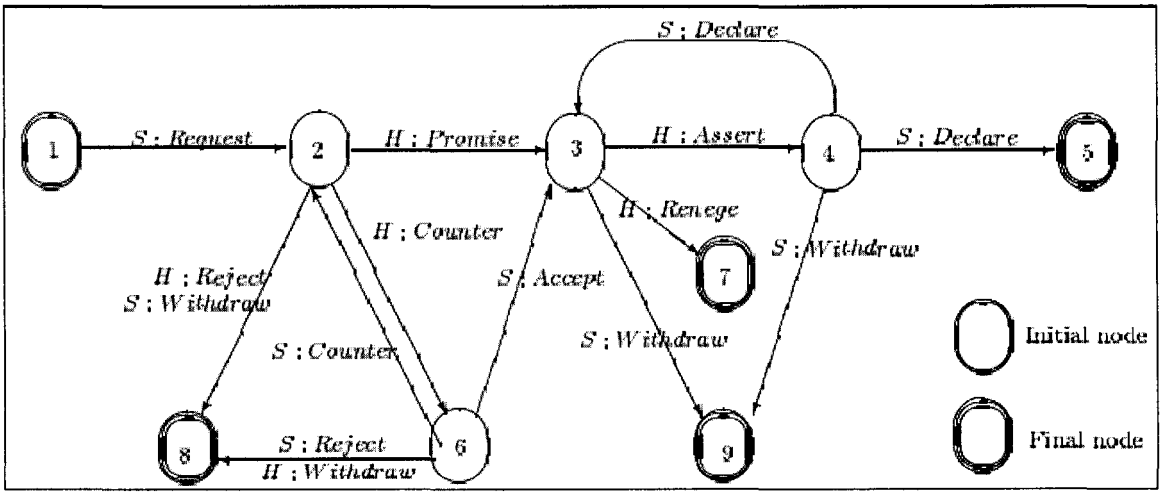


Figure 10: Protocole "demande d'action"

Tentons de décrire le comportement du dialogue prévu avec le protocole demande d'action décrit au niveau de la figure 10. La conversation commence dans l'état initial (1),

par la demande de l'état de l'orateur S. Au niveau de l'état (2), le dialogue peut, avec succès :

- être suivi de la promesse de H de réaliser l'action demandée, ou
- hériter d'un cycle de négociation avec une contre-proposition de H, ou
- échouer avec un rejet (rejection) menant à l'état (8).

Au niveau de l'état (3), le destinataire signalera l'accomplissement de la tâche, ou décidera par la suite de renoncer (menant à l'état final (7)) et S (« Declare ») peut effectuer une bonne évaluation menant à l'état (5) ou mal évaluer le tout menant à l'état (3).

Remarque : Le protocole n'indique rien au sujet de la teneur des actes communicatifs.

3.3.2 Limites des règles de conversation

Pour mieux illustrer les limites des règles de conversations, reprenons l'exemple du protocole *demande d'action.* Pour commencer, comme expliqué ci-dessus, la concordance de la conversation est assurée par les contraintes imposées par le protocole à tous les participants. La conséquence est que tous les messages non prévus par le protocole utilisé ne seront pas considérés. En outre, une fois considéré soigneusement, le protocole semble se composer de différentes phases (ou de petits protocoles), (non identifiés en tant que tels):

Premièrement, les agents essayeront de négocier une tâche pour que H fasse. Ensuite S et H discuteront de l'accomplissement correct de cette tâche. Ces phases (ou petits protocoles) ne sont pas spécifiques au cas particulier de la demande pour l'action. Nous n'avons donc aucune information sur la façon dont les agents ont accepté d'employer un tel protocole.

3.3.2.1 Flexibilité

Le but des règles de conversation est fondamentalement de restreindre le comportement conversationnel des participants, mais il y a un équilibre sensible à trouver entre cet aspect normatif et la flexibilité prévue dans la plupart des communications multi agent. Les différents points peuvent participer à cet objectif:

- Adopter un formalisme qui laisse plus de flexibilité que les machines a état finis, dépendant plus de l'état de la conversation que sur les messages précédents.

- Considérer les messages inattendus (ou exceptionnels) au sein des règles de conversation. L'objectif est ici de donner des suivis appropriés à de tels messages.

- Préférer diverses petites règles de conversation (« protocoles de conversation »), idéalement ceux qui peuvent être bâtis, à un simple large protocole. Le fait de trouver les structures semblables récurrentes du dialogue dans diverses règles de conversation est clairement un argument fort pour ce point. Les possibilités de bâtissage devraient être clairement définies.

- Étudier comment les agents arrivent à parvenir à un accord sur la règle de conversation actuellement utilisée (ce point est crucial si l'objectif précédent est atteint).

3.3.2.2 Spécification

Les spécifications des règles de conversations représentent le deuxième défi important identifié ici. En effet, les caractéristiques des règles de conversations exigent souvent le formalisme ad hoc et sont seulement semi formellement énoncées. Ceci ne permet pas de bien bénéficier des spécifications, ou de formellement vérifier quelques propriétés prévues du modèle. En général, les objectifs sont les suivants:

1. Spécifier des règles de conversation a un niveau d'abstraction élevé. De telles caractéristiques devraient être relativement indépendantes des spécificités des agents impliqués dans la communication, en particulier les états mentaux privés de ces agents

2. Adopter une approche descriptive afin de déclarer explicitement les règles composant les règles de conversation. La clarté et l'expressivité suivent de ce point.

3. Fournir par la suite les propriétés formelles des règles de conversations proposées. En particulier, le problème de l'arrêt des règles de conversations est fortement approprié.

4. Essayer d'optimiser les règles de conversations. Des spécifications précises pourraient en effet permettre d'identifier les raccourcis que pourraient prendre les participants du dialogue au niveau des règles de conversation sans pour autant modifier la signification de l'interaction.

En conclusion, l'utilisation des règles de conversation pour guider le comportement communicatif d'agent engendre une foule de questions pratiques :

- Comment les politiques de conversation devraient-elles être mises en application dans des systèmes d'agent?

- Les règles de conversations devraient-elles être téléchargées d'une bibliothèque commune, préconçue dans le programme de l'agent, ou être dérivées des axiomes conversationnels au temps d'exécution?

- Comment les politiques de conversation peuvent-elles être négociées, et les règles peu familières apprises?

- Et en conclusion, comment les règles de conversation peuvent-elles être intégrées avec les autres plans, et règles qui définissent le comportement d'un agent?

Il y a une alternative à *l'agence mentale* et aux règles de conversation: l'agence sociale. Cette approche prometteuse considère les actes communicatifs en tant qu'élément de l'interaction sociale continue. Dans ce cas-ci, même si nous ne pouvons pas déterminer si les agents ont un état mental spécifique, nous sommes sûrs que les agents qui communiquent suivent quelques lois sociales qui permettent de soutenir des conversations.

3.3.3 Agence sociale

Les concepteurs d'agent ont habituellement supposé que les réseaux des rapports d'engagement et de puissance qui caractérisent le comportement social humain ne sont pas appropriés aux systèmes multiagents. Dans la pratique, cependant, les conventions sociales *idiosyncrasiques* ont fini par être implicitement enfoncées dans des architectures d'agent et des protocoles d'interactions;Par exemple, la norme simple qu'un agent essayera habituellement de répondre à une question (et même effectuer des essais pour trouver la meilleure réponse). Le fait qu'un agent soit coopératif ou pas, dépends en grande partie des engagements (commitments), obligation, et ainsi de suite. Puisque ces concepts sont seulement implicitement incorporés au comportement (et aux protocoles) des agents, le résultat est que les différents systèmes d'agent montrent des incompatibilités significatives dans ce secteur. Plus de recherches devront être effectuées dans la caractérisation de ces concepts communicatifs fondamentaux dans un contexte multiagent. Ceci inclut des concepts tels que les engagements (commitments), obligations, conventions, pouvoir (power) (dans le sens des relations hiérarchiques), et ainsi de suite. Une fois que ces concepts sont clarifiés, il devient alors possible d'établir une sémantique et une pragmatique unifiée des langages de communications qui tiennent compte de ces concepts.

Malgré le fait qu'une évolution s'est fait sentir au niveau de la sémantique des différents actes de la parole dans KQML et Fipa-ACL, on connaît peu au sujet de la sémantique des conversations et des relations entre les actes de la parole et les conversations dont ils font partie. Une sémantique claire des conversations peut faciliter l'extensibilité et le « scalability » des conversations entre les agents dans le sens qu'il peut être plus facile pour un utilisateur de prolonger la conversation existante avec de nouveaux performatifs (actes communicatifs).

3.4 Nouveaux langages basés sur les engagements et les jeux de dialogue

3.4.1 Un environnement pour les sémantiques des performatifs KQML

3.4.1.1 Que sont les notions dialectiques

La dialectique est le champ de la recherche concerné par l'étude du processus interactif de l'argumentation. En particulier, la dialectique étudie les différents types d'erreurs impliquées dans l'argumentation et la dialectique formelle a présenté des notions formelles et des outils pour les traiter. Parmi les diverses notions présentées ou discutées par les dialecticiens formels, deux se sont avérés être de la première importance pour ceux intéressés dans la communication au niveau des systèmes multiagent: « engagement social » (**social commitments**) et «systèmes dialectaux» (**dialectical systems**).

3.4.1.2 Engagements sociaux

Tout d'abord, l'engagement social ne devrait pas être confondu avec l'engagement psychologique, qui capture généralement de la persistance des intentions. De manière cruciale, les engagements sociaux sont des engagements qui lient les membres d'une communauté. Ceci motive naturellement une distinction entre le créancier et le débiteur de l'engagement. Notez que cette notion est différente de celle de la croyance ou l'intention et, plus généralement, les engagements sociaux sont distincts des états privés des agents.

Au niveau des engagements sociaux, une distinction peut être faite entre des « engagements propositionnels » (**propositional commitments**) et des « engagements basés sur l'action » (**action commitments**).

Exemples : Dans cet exemple, A représente l'orateur.

(1) A : Ottawa est la capitale du Canada.

(2) A : Je vais visiter Ottawa, l'été prochain.

On peut dire que (1) est un engagement propositionnel lie l'agent *A* vers l'audience (qu'on appelle B), alors que (2) est un engagement basé sur l'action qui lie *A* envers *B*. Avant de conclure que *A* croit ce qu'il dit, on peut dire que l'agent *A* a un

engagement envers l'audience, et il y a des conséquences reliées à un tel engagement. En particulier, l'audience va certainement pénaliser A s'il effectue une affirmation contraire à la proposition *p* (Capital (Ottawa, Canada)). Dans ce cas précis, l'agent *A* va être considéré comme un agent non-crédible.

En d'autres termes, en maintenant tout simplement (1), *A* est engagé d'une manière qui limite ses actions ultérieures. Plus précisément, dépendant du contexte, *A* peut se retrouver engagé à un certain nombre de choses, par exemple, tenant compte de *p*, défendant que *p* (si remis en question), ne niant pas que p, démontrant que p, discutant que *p*, en prouvant ou en établissant que *p* et ainsi de suite. Finalement, *A* se retrouve engagé à quelques (des ensembles de) stratégies partielles. Mais la raison pour laquelle nous parlons de l'engagement propositionnel est qu' ici, tous les engagements de *A* (comme défini par les stratégies où il est engagé) portent sur la proposition *p*. On pourra dire que *A* est engagé à *p*, signifiant que *A* est engagé à un ensemble de stratégies partielles portant sur *p* (Walton et Krabbe, 1995).

3.4.1.3 Systèmes dialectiques

Les Systèmes Dialectiquess sont des modèles normatifs du dialogue persuasif qui se composent principalement de

(1) D'un stock un mouvement (défi, affirmation, question).
(2) D'un stock d'engagement (CS) pour participant au niveau de la conversation.
(3) D'un ensemble de règles de dialogue qui régulent les mouvements
(4) D'un ensemble de règles d'engagement définissant l'effet des mouvements sur les stocks d'engagements.

Le dialogue se poursuit correctement quand les participants à la conversation se conforment aux règles et par la suite aux extrémités du dialogue quand le partisan a retiré sa thèse, ou (symétriquement) quand l'opposant a concédé à la réclamation de partisan. Ces conditions d'arrêt pourraient être différentes pour d'autres types de dialogue. Les stocks d'engagement des participants reflètent ainsi l'état du dialogue.

Le tableau 1 reflète comment les stocks d'engagement évoluent pour chaque participant (CSp pour P, CSc pour C) durant le bout de dialogue (3), dans style de jeux de dialogue. Notons que les propositions défiées sont stockées dans ce modèle, afin d'éviter quelques dialogues circulaires considérés comme erreurs. Ceci empêche par exemple le joueur P de répondre à C4 en affirmant que le protocole est inutile, un fait a toujours défié à ce point de la discussion.

P1: Le protocole kyoto est inutile
C2 : Pourquoi ?
P3 : parce qu'il doit être largement approuvé pour être utile et la plupart des pays sont peu disposés à le ratifier.
C4 : Pourquoi ces Pays sont peu disposés à le ratifier ?

Tour (Turn)	Joueur (Player)	Action (Move)	CS_P	CS_C
1	P	**assert($\neg u$)**	$\neg u$	$\neg u$
2	C	**challenge($\neg u$)**	$\neg u$	$?\neg u$
3	P	**assert($m \wedge \neg a$)**	$\neg u, m \wedge \neg a$ $m \wedge \neg a \rightarrow \neg u$	$? \neg u, m \wedge \neg a$ $m \wedge \neg a \rightarrow \neg u$
4	C	**challenge(m)**	$\neg u, m \wedge \neg a$ $m \wedge \neg a \rightarrow \neg u$	$? \neg u$ **,?m,** $\neg a$ $m \wedge \neg a \rightarrow \neg u$

Tableau 4: Comment le stockage des engagements évolue durant une conversation

Avec $\neg u$: représente l'affirmation : « Le protocole Kyoto est inutile »

m : « La majorité est requise »

$\neg a$: « La plupart des pays sont peu disposés à le ratifier »

À chaque tour de dialogue, l'ensemble des actions permises est donné par les règles de dialogue. Ces règles de dialogue dépendant de l'action précédente, ou de la conversation, ou de l'état de la conversation détaillée dans les stocks d'engagements de chaque participant.

- **Types de dialogue**

Les systèmes dialectaux peuvent modéliser divers types de dialogue (Walton et Krabbe, 1995). Les différents types de dialogue ont été classifiés selon trois facteurs:

- l'information disponible aux participants,
- -le but du dialogue en tant que tel, et
- -aux différents buts des participants.

Selon ces critères, six types de dialogue ont été identifiés. Le tableau 2 résume ces types, et montre les buts (du dialogue et des participants).

Type de dialogue	But du dialogue	Situation initiale
Persuasion	Résolution de conflits	Points de vues
Négociation	Faire une affaire	conflictuels
Délibération	Prendre une décision	Conflits d'intérêts
Quête d'information	Propagation de la connaissance	Besoin d'action
Enquête	Croissance de la connaissance	Ignorance personnelle
Éristique	Accommodation dans les relations	Ignorance générale
		Antagonisme

Tableau 5: Les six types principaux de dialogue

Ces six types peuvent être raffinés en les sous-types, simplement en spécifiant des conditions plus raffinées sur le type de dialogues (le type de conflit, le degré de rigidité des règles). Ainsi par exemple, un conflit est un sous-type d'une persuasion, où chaque participant essaye de défendre son propre point de vue. Les types du dialogue coïncident avec les systèmes dialectaux ou des jeux dialogues particuliers (Walton et Krabbe). Dans tous les types de dialogue, les deux partis idéalement ont enregistré les états de leurs engagements individuels à n'importe quel point donné de la progression du dialogue (stocks d'engagements).

- **Décalages dialectiques**

Cependant, les dialogues ne sont habituellement pas que d'un seul type de leur commencement à leur fin. Par exemple, il est commun de commencer un dialogue d'enquête, pour se rendre compte pendant le dialogue qu'il y a une controverse en jeu, pour ensuite entrer dans un sous-dialogue de conflit, et pour reprendre par la suite le dialogue d'enquête quand le problème a été résolu. Ceci signifie que la plupart des types de dialogues sont complexes (c'est à dire Composés de différents types). La notion de décalage dialectal a été présentée pour cet aspect (Walton et Krabbe). Un décalage dialectal est un changement du contexte au niveau du dialogue pendant une conversation d'un type de dialogue à un autre. Le décalage pourrait être constructif et approuvé par tous les partis, et c'est un décalage « licit ». Dans ce cas-ci, le deuxième dialogue peut même être fonctionnellement lié au but du dialogue original: c'est un dialogue encastré. Par exemple vu l'exemple préparé au niveau de la figure 11, nous pourrions le voir comme un sous-dialogue de négociation inclus dans une demande pour le dialogue d'action.

S: Va tu ratifier Kyoto?	Début « demande d'action » (request for action)
H: Non, mais on va réduire notre gaz	Début Négociation
S: Peux tu le réduire à 40%	
H: Non, 30%	
S: OK	Fin Négociation
H: On te fera signe lorsque l'objectif sera atteint	
S: Merci	Fin « demande d'action » (request for action)

Figure 11: Exemple de "sous-dialogue" de négociation (le dialogue d'action).

Dans d'autres situations, le décalage pourrait être caché ou inadéquat parce que c'est un type illicite de décalage. Des variations illicites dans notre contexte du dialogue basé sur des argumentations sont fréquemment associées aux erreurs (une erreur peut être vue comme un argument qui semble être correct mais qui ne l'est pas). Dans certains dialogues, nous pourrions avoir un effet de cascade. Dans ce cas-ci, il y a des séries de décalages d'un type de dialogue à un autre, et ainsi de suite. Ainsi, on peut par exemple commencer par une négociation, et puis décaler vers une discussion critique et ensuite décaler vers une négociation qui pourrait décaler à une querelle, ect.

- **Conclusion**

Le lien naturel prévu avec le raisonnement de quelques modèles basés sur l'argumentation est une caractéristique attrayante de cette approche : les agents sont supposés avoir une capacité minimum d'argumentation (e.g donnant des raisons de supporter une croyance ou une intention). Ceci mène à une discussion plus riche que de simples échanges de locutions supportées, plus particulièrement dans les cas de persuasion ou négociation (Parson et Jennings).

En fait, les dialectiques formelles s'avèrent très utiles au niveau de la définition de nouvelles règles de conversations. Une telle influence a permis de produire deux types d'approches : les « protocoles basés sur les engagements » (***commitment-based protocols***) et les « protocoles basés sur les jeux » (***dialogue-game based protocols***).

3.4.2 Protocoles basés sur les « *engagements sociaux* »

3.4.2.1 Machines d'engagement (commitment machines)

Une approche où la synthèse des représentations de machine d'état fini des protocoles est en train d'être développée. D'abord des protocoles de transmission sont modélisés par l'intermédiaire des machines d'engagement (commitment machines). Les machines d'engagements fournissent un contenu aux états et aux actions des protocoles en termes d'engagements sociaux des participants entre eux. Le contenu peut être raisonné plus ou moins par les agents permettant de ce fait l'exécution flexible. Des règles de raisonnement pour capturer l'évolution des engagements par les actions des agents sont fournies. En raison de sa représentation de contenu et de ses règles opérationnelles, une machine d'engagement code efficacement une version systématiquement augmentée d'un protocole qui permet les séquences de base des actions aussi bien que d'autres mouvements légaux afin de s'adapter à des exceptions et à des occasions.

Présentation

Les machines d'état fini sont faciles à opérationnaliser mais elles cachent les contenus des états et mènent à des exécutions rigides. Ceci a motivé le fait de définir une machine d'engagement (CM) en termes d'ensembles états de et actions qui sont données un contenu sémantique déclaratif en termes d'engagement. Une machine d'engagement spécifie :

- Les états possibles dans lesquels un état peut être.
- Les actions utilisées pour la transition d'un état à un autre.
- Les états finaux possibles du protocole.

Le contenu lié à chaque état indique quels engagements sont en vigueur au niveau de cet état en particulier, et le contenu lié à chaque action définit comment les engagements sont affectés par cette action (menant de ce fait à un changement d'état).

Tout comme une machine à état fini, une machine d'engagement à l'état actuel, permet zéro actions ou plus au niveau de chaque état. À la différence d'une machine à état fini, la représentation d'une machine d'engagement n'indique pas un état commençant. Les participants peuvent commencer le protocole d'un état en acceptant les engagements qui sont en vigueur dans cet état. Habituellement, le protocole aura quelques états où aucun engagement n'est en vigueur. Une machine d'engagement a également des états finaux, qui reflètent les états acceptables ou souhaitables d'arrêts du protocole.

D'une manière primordiale, à la différence d'une machine à état fini, les transitions entre états ne sont pas explicitement indiquées. Les contenus des états et des actions sont logiquement représentés. Basé sur le contenu intrinsèque des actions, le nouvel état atteint par l'exécution d'une action à un état particulier peut être logiquement impliqué. Ainsi, au lieu d'indiquer les ordres des actions qui peuvent être effectuées, une machine d'engagement indique simplement les contenus légaux dans le protocole et, parmi ces derniers, les contenus finaux.

Les spécifications d'un protocole des machines d'engagement soulignent que le but d'exécuter le protocole n'est pas simplement d'exécuter certains ordres des actions, mais pour atteindre un état qui représente le résultat d'exécuter ces ordres des actions. À cet effet, nous pouvons fournir les différents chemins qui accomplissent le même but que le chemin original. Ainsi, un protocole peut être modifié (augmenté ou abrégé) en trouvant les chemins alternatifs entre les états.

La spécification d'un protocole d'une machine d'engagement peut se faire de deux façons principales :

- *Exécution. (Run time)* Des spécifications d'un protocole d'une machine d'engagement donnent les états et les conséquences d'effectuer les actions diverses. Au niveau d' une machine d'engagement, un agent qui peut traiter des formules logiques peut calculer les transitions entre les états. À cet égard, le choix des actions est un problème de planification pour chaque agent. C'est-à-dire, à partir des états finaux possibles, l'agent décide d'abord de l'état final désiré, et

implique ensuite logiquement un chemin qui le prendra de l'état actuel à l'état final désiré. En fait, l'agent interprète la machine d'engagement directement au temps d'exécution.

- *compilation. (Compile time)* Pour réduire le calcul prié au temps d'exécution, une machine à états finis peut être synthétisée à partir d'une machine d'engagement. La machine à états finis soustrairait hors des contenus des états et des actions. Une machine à états finis peut être mécaniquement exécutée sans inférence logique explicite. Cette transformation est basée sur la production systématique des chemins entre les paires de contenus. Le contenu dans la machine d'engagement se rapporte aux états dans une machine à état fini. L'ensemble des actions demeure le même. Les transitions entre les états de la machine à états finis suivent des transitions dans la machine d'engagement. L'ensemble des états finaux sera le contenu final dans la machine d'engagement.

Protocoles améliorés (augmentées)

Un des avantages principaux de spécifier des protocoles employant les machines d'engagements est leur facilité de perfectionnement. En ajoutant le nouveau contenu à l'ensemble du contenu, M, et convenablement, à l'ensemble final de contenu, F, le protocole peut être augmenté pour permettre des calculs additionnels. De même, en réduisant les ensembles de contenu M et F, les calculs possibles peuvent être restreints. L'exemple suivant dépeint comment des spécifications existantes de machine d'engagements peuvent être augmentées.

- **Exécution flexible des machines d'engagement**

En ce moment, il est important de redire ce que voulons dire nous par flexibilité. Bien que nous voulions que ces agents décrètent un protocole flexible, nous voulons toujours préserver un ordre qui permettra seulement des conversations significatives. Par

exemple, un négociant ne devrait pas envoyer une citation après envoi des marchandises, et le consommateur ne devrait pas commencer la conversation en envoyant un EPO.

D'une manière primordiale, la flexibilité peut être présentée seulement au point où les significations prévues des actions ne sont pas violées. Les interactions parmi les partis peuvent avoir une signification basée sur leurs engagements entre eux. Quand nous permettons des interactions plus flexibles, nous devons nous assurer que les engagements originaux sont en vigueur ou qu'ils sont changés suivant un accord mutuel. L'exécution d'un protocole alors est contrainte de façon minimale pour satisfaire seulement ceux qui exigent des ordres spécifiques d'exécution et ne fournissent aucune base pour décider sur l'état correct indépendant de l'ordre d'exécution. Les caractéristiques suivantes des machines d'engagement sont des résultats de cet avantage : **Opportunisme** et **composition.**

- Opportunisme

Selon les circonstances, un participant peut choisir de commencer un protocole à partir du milieu pour tirer profit d'une occasion. Une occasion dans ce contexte correspond à un contenu basé engagement (commitment-based) d'un état qui fournit une certaine convenance au participant. En commençant un protocole de cette façon, l'agent accepte les engagements qui sont en vigueur dans cet état particulier. De même, pendant l'exécution d'un protocole, un agent peut pouvoir sauter à travers des étapes immédiatement. Dans notre exemple courant, le négociant peut envoyer une citation des prix de manière proactive.

- Composition

Des protocoles peuvent être combinés dans de plus grands protocoles, aussi longtemps que des extrémités d'un protocole dans un état d'engagement qu'un autre protocole peut accommoder. Ceci peut être applicable dans deux cas, d'abord combiner les protocoles à exécuter les uns après les autres, et en second lieu permettre à un protocole latéral d'être suivi pendant l'exécution du premier protocole.

Un protocole peut se composer en combinant des modules de sous-protocoles à travers un état commun d'engagement. Ce besoin commun d'état doit être un des états finaux de la première machine d'engagement, et un état possible dans la deuxième machine d'engagement. Si l'exécution du premier protocole finit à l'état commun d'engagement, l'exécution peut continuer à partir de cet état actuel particulier dans le deuxième protocole, à condition que l'exécution du protocole ne retourne jamais à un état dans le premier protocole.

Alternativement, les participants d'un protocole peuvent interagir avec des agents qui n'ont pas lieu dans le protocole à travers des protocoles latéraux. Dans ce cas-ci, le point crucial est de s'assurer que l'agent qui entre dans un protocole latéral doit retourner au point où il a quitté au niveau du protocole principal.

Conclusion

Le nouveau formalisme de machines d'engagement a été proposé pour spécifier et exécuter les protocoles dans les systèmes multiagents. Les machines d'engagement fournissent flexibilité en capturant le contenu sémantique des actions au niveau d'un protocole. En spécifiant les protocoles de communication utilisant les engagements, on peut analyser les interactions entre participants à travers la signification intrinsèque de ces interactions. Ces machines peuvent être appliquées durant l'exécution (Run Time), ou la compilation (compile time), dépendant des restrictions au niveau des calculs et de l'architecture des agents.

3.4.2.2 Colombetti (Albatross)

Termes et dénotations

Colombetti a récemment proposé ALBATROSS (langue d'agent basée sur le traitement de sémantiques sociale) (Colombetti, 2000b; Colombetti, 2000a) comme approche dans le même que Singh. Il a en particulier, présenté la notion de pré-engagement comme genre d'engagement conditionnel (par exemple une demande pré-commet («pré-engagement ») l'agent auquel on s'adresse, signifiant que cet agent sera commis en cas d'acceptation). Comme Singh, sa sémantique d'engagement social est également exprimée en limite d'une logique temporelle basée sur CTL *. dans cette logique, l'auteur a présenté un nouveau type d'expressions (α, β, ...), appelé les formules d'action, qui décrivent des types d'action. Si «i » est un nom d'agent et α est une formule d'action, une expression de la forme Diα est une signification de formule d'état que l'agent i (c.-à-d., l'acteur) a juste accomplie l'exécution d'une marque α. Puis, l'auteur a présenté les expressions *mc(i; j; λ)* et *mp(i; j; λ)* comme formules d'action signifiant « make-commitment » (faire un engagement) et « make pre-commitment » (faire un pre-engagement) respectivement. Dans ce cas-ci, l'auteur obtient les axiomes suivants:

1. $D_i mc(i, j, \lambda) \rightarrow C_{ij}\lambda$
2. $D_i mp(j, i, \lambda) \rightarrow P_{ji}\lambda$

Les premiers axiomes signifient que l'acceptation transforme un pré-engagement en engagement (Cij est l'opérateur modal pour l'engagement.). Les deuxièmes axiomes signifient qu'un pré-engagement (comme engagement) devrait persister. Maintenant la question est comment définir la sémantique des messages en termes de logique de Colombetti et en particulier en termes de MC et MP. En fait agir d'une parole comme l'affirmation (« assert ») est exprimée ici comme:

```
(assert
     :sender    i
     :receiver  j
     :content   Lambda)
```

L'envoi d'un tel message implique: Dimc (i; j; λ).De cette façon, les actions permises comme le MC et le MP peuvent être exécutés indirectement, par les messages qui réalisent un répertoire prédéfini de types d'acte de la parole. L'autre manière proposée par Colombetti consiste à donner à des agents plus d'accès directs aux engagements. L'idée est de permettre à un agent de faire un engagement, un pre-engagement, etc. par une déclaration. Ainsi, un agent peut se commettre à la vérité de l'affirmation en envoyant un message de la forme:

```
(declare
     :sender    i
     :receiver  j
     :content   (make-commitment
                 :debtor       i
                 :creditor     j
                 :condition    Lambda))
```

Contrairement au travail de Singh, cette approche n'a pas nécessairement besoin de protocoles additionnels explicites, puisque le protocole lui-même peut être exprimé de pré-engagements. C'est le choix fait dans certains cas, car la règle suivante montre:

$$askIf(x,y,\phi) = request(x,y,Done(y,assert(x,y,\phi)|assert(x,y,\neg\phi)/now+k))$$

C'est élégant et mène à plus de flexibilité, puisque les agents sont censés répondre comme prévu par les engagements, mais ne sont pas obligés à le faire. Toutefois Colombetti, quand il réclame que « *la sémantique du message devrait être indépendante de la nature de la conversation* », cela semble peu disposé à défendre ceci comme politique générale.

Actes De Discours (Speech Acts)

Dans cette section, on définit un certain nombre d'actes de la parole. On suit John Searle [15] en classifiant des actes de discours en cinq catégories: déclarations, affirmations, actes de discours engageants, directifs, et expressifs. Toutes ces catégories sont appropriées à la communication inter-agent, excepté les expressives: au moins dans l'avenir, les agents ne sont pas susceptibles de passer beaucoup d'heure dans la félicitation, faisant des excuses, et ainsi de suite. Pour chacune des quatre catégories restantes, on définirai d'abord un acte de base, c.-à-d., une sorte de zéro-point pour la

catégorie; alors on présenterai des actes plus complexes de la parole pour montrer la flexibilité de la sémantique basée engagement.

- **Déclarations**

Une déclaration est un acte de la parole qui provoque un état de la question qui rend son contenu vrai. Les exemples des déclarations sont « *les enchères sont ouvertes* » (employé pour ouvrir une enchère) et « *le prix de cet article est 100 euros* » (employés pour fixer le prix de l'article). Il n'est pas difficile de voir que les divers types d'agents (par exemple, ceux impliqués dans le commerce électronique) devraient pouvoir exécuter des déclarations. Le point d'une déclaration doit provoquer la vérité de ce qu'il s a déclaré. Cependant, afin que la déclaration soit efficace, l'agent de déclaration doit être doté de puissances spécifiques. Par exemple, une enchère ne peut pas être ouverte par un des participants, et le prix d'un article ne peut être fixé par un client. Quand agent X est autorisé à provoquer par déclaration, Voici les axiomes pour des déclarations

(Declare1) $declare(x,\varphi) = speechAct(x,y,declare,\varphi)$.
(Declare2) $Done(x,declare(x,\varphi)) \wedge Empowered(x,\varphi) \rightarrow \varphi$.

- **Affirmations ("Assert")**

Le but d'une affirmation est doit commettre son acteur à la vérité de ce qui est affirmé, relativement à chaque destinataire. L'acte d'affirmer, peut être défini comme suit :

(Assert1) $assert(x,y,\varphi) = speechAct(x,y,assert,\varphi)$,
(Assert2) $Done(e,x,assert(x,y,\varphi)) \rightarrow C(e,x,y,\varphi)$.

Contrairement à affirmer, un acte d'informer présuppose que l'orateur croit son contenu s'attends à ce que les destinataires le croient également. Les conditions de ce type ne sont pas considérées dans la sémantique basée sur les engagements (çommitment-based) d'Albatross. FIPA ACL définit également des affirmations additionnelles, comme la confirmation et l'infirmation.

- **Actes de discours engageants (Commissives)**

Le but d'un acte de discours engageant est de lier l'orateur, relativement à chaque destinataire, à l'exécution d'une action d'un type indiqué dans une date limite. Voici la définition de l'acte de discours engageant de base, *promettre (promising)*:

(Promise1) $promise(x,y,\varphi) = speechAct(x,y,promise,\varphi)$.

(Promise2) $Done(e,x,promise(x,y,Done(x,\alpha/d))) \wedge d > now \rightarrow C(e,x,y,Done(x,\alpha/d))$.

Il apparaît de cette définition qu'un acte de discours engageant peut être remplacé par une affirmation: au lieu de la promesse pour faire α, un agent pourrait tout simplement affirmer qu'il fera α.

- **Directifs**

Le but d'un acte directif est de faire effectuer au(x) destinataire(s) une certaine action dans une date limite. Dans une approche mentale à la communication, on traite typiquement des directives en termes d'intentions. Avec cohérence, on traite des directives en termes d'engagements. Cependant, comme déjà précisé, un agent ne peut pas directement provoquer un engagement d'un autre agent. Pour résoudre ce problème, on compte sur le concept de pre-engagement.

L'acte directif de base, *« demander » (requesting),* peut être défini comme suit:

(Request1) $request(x,y,\varphi) = speechAct(x,y,request,\varphi)$.

(Request2) $Done(e,x,request(x,y,Done(x,\alpha/d))) \wedge d > now \rightarrow PC(e,y,x,Done(x,\alpha/d))$.

- **Accepter et rejeter (Accepting and rejecting)**

En général, un pré-engagement peut être accepté ou rejeté par son débiteur. Accepter est un acte de la parole qui transforme un pré-engagement, dérivant typiquement à partir d'un acte directif, vers un engagement total. Il peut être défini comme suit:

(Accept1) $accept(x,y,\varphi) = speechAct(x,y,accept,\varphi)$.

(Accept2) $Done(x,accept(x,y,\varphi)) \wedge PC(e,x,y,\varphi) \rightarrow C(e,x,y,\varphi)$.

Au contraire, un acte du rejet annule un pre-engagement :

(Reject1) $reject(x,y,\varphi) = speechAct(x,y,reject,\varphi)$,

(Reject2) $Done(x,reject(x,y,\varphi)) \wedge PC(e,x,y,\varphi) \rightarrow Next \neg PC(e,x,y,\varphi)$.

- **Contrats**

Intuitivement, nous estimons qu'au moins dans certains cas une directive peut réellement produire l'engagement d'un destinataire. Le cas le plus évident est quand la directive a la force d'un ordre. Cependant, l'exécution non-défective des directives semblables présuppose l'existence d'une affirmation sociale appropriée. Dans le cas des ordres, par exemple, une affirmation de la subordination doit être établi entre l'orateur et les destinataires. Par exemple, nous pouvons penser à une situation dans laquelle un agent est lié à accepter un pre-engagement en raison d'un accord précédent dans ce sens. Ces genres d'accords préexistants sont appelés des contrats. Plus avec précision, on suppose que les agents X et y peuvent être liés par un contrat pour accepter tous les pré-engagements de x, relativement à y, pour faire une action de type □, et on exprime ceci par l'affirmation Contract (x, y, α). Nous avons l'axiome suivant:

(Contract) $Contract(x,y,\alpha) \wedge PC(e,x,y,Done(x,\alpha/d)) \wedge d > now \rightarrow C(e,x,y,Done(x,\alpha/d))$.

Pour montrer la flexibilité de la sémantique basée engagement, on va maintenant définir quelques types d'actes de parole en plus, à savoir des questions, des questions « wh-questions», et des propositions de oui/non.

- **Questions oui/non (Yes/no)**

Les questions de oui/non (Yes/no) représentent une sous-classe notable des actes directifs, par lesquels un agent invite un autre agent à affirmer si une certaine phrase est vraie ou fausse. On suppose que des réponses aux questions de oui/non doivent être données dans des instants de k, où k peut être vue comme constante globale, ou comme un autre paramètre des questions:

(AssertIf) $assertIf(x,y,\varphi) = (assert(x,y,\varphi) \mid assert(x,y,\neg\varphi))$.
(AskIf) $askIf(x,y,\varphi) =$
$request(x,y,Done(y,assertIf(y,x,\varphi))/now+k))$.

- **Wh-questions**

Avant de définir des wh-questions, il est nécessaire d'établir ce qui compte en tant que valide wh-réponses. Wh-réponse est un acte affirmatif dont le contenu est une affirmation indicative, c.-à-d., une affirmation de la forme

$c = \iota v\ \varphi$.

Où: c est une constante de la sorte appropriée; ι est l'opérateur iota des descriptions définies; v est une variable de la même sorte de c; et φ est une formule contenant v comme seule variable libre. Comme avant, on suppose que des réponses aux wh-questions doivent être données dans des instants k:

(AskWh) $askWh(x,y,\varphi) =$
$request(x,y,Done(y,assert(y,x,c = \iota v\ \varphi)/now+k))$.

Où v est la seule variable libre dans φ, et c est une constante.

- **Propositions**

Une proposition est la conjonction d'une directive et d'un commissive conditionnel. Par exemple, nous pouvons analyser « x propose à y d'acheter le produit i » selon les conditions suivantes:

- X pré-engage y, relativement à x, à transférer la propriété de I à x; et
- X s'engage, relativement à y, à transférer un montant donné d'argent à y, à condition que y de transfère la propriété I à x ou de s'engager à le faire.

En effet, les propositions vont être très communes dans des interactions entre agents. Nous pouvons nous attendre à ce qu'elles apparaissent chaque fois qu'une interaction élémentaire entre deux agents, x et y, comporte l'exécution d'une action par x et d'une autre action par le y. c'est le cas, par exemple, avec des transactions commerciales, où le même événement peut être décrit comme des achats du point de vue d'un client, et comme des ventes du point de vue d'un vendeur.

Afin de traiter correctement avec des propositions, présentons d'abord une manière de traiter des interactions élémentaires. Dans une interaction élémentaire, nous avons un certain nombre de participants, dont chacun possède un rôle indiqué. Alternativement, un rôle est une action ou un ensemble d'actions, effectué par un des participants. Par exemple, acheter le produit i au prix p peut-être défini comme:

(Buy1) $Role(x,buy(x,y,i,p),moneyTransfer(x,y,p))$,

(Buy2) $Role(y,buy(x,y,i,p),propertyTransfer(y,x,i))$.

Les propositions peuvent être définies comme suit :

(Propose1) $propose(x,y,\varphi) = speechAct(x,y,propose,\varphi)$,

(Propose2) $Done(e,x,propose(x,y,Done(x,d,\alpha)))$
$\wedge\ Role(x,\alpha,\beta) \wedge Role(y,\alpha,\gamma) \wedge d > now$
$\rightarrow\ PC(e,y,x,Done(y,d,\gamma))$
$\wedge\ (Done(y,d,\gamma) \vee C(e,y,x,Done(y,d,\gamma))$
$\rightarrow C(e,x,y,Done(x,d,\beta)))$.

Cette définition possède les conséquences suivantes. D'abord, comme une proposition x à y met y dans un état de pre-engagement, il est significatif pour « y » d'accepter une proposition. En second lieu, si « y » accepte la proposition, les deux agents sont engagés

à leurs rôles respectifs dans l'interaction; l'engagement de « x » est provoqué également si « y » exécute directement son rôle sans s'y engager explicitement.

Conclusion

Le niveau auquel la communication est traitée est très abstrait, et il y a un espace considérable à remplir afin de réduire le modèle au niveau de l'exécution. La vue personnelle de l'auteur est que la plupart des agents utilisés dans des applications pratiques n'auront pas un degré élevé d'intelligence. (C.-à-d., capacités déductives en ligne). Par conséquent, on considère le modèle proposé principalement en tant que moyens d'indiquer les agents communicants en différé (off-line). Cependant, beaucoup de travail est nécessaire toujours avant que la proposition de Colombetti puisse être transformée en outil utilisable, et une partie de ce travail sera nécessairement expérimentale. Ses plans courants sont d'appliquer le modèle aux agents marchands (en particulier dans les contextes des enchères) et à une communauté des agents responsable de gérer un bâtiment intelligent.

3.4.2.3 Protocole de proposition (Flores et Kremer)

Au niveau des protocoles basés sur les engagements sociaux de Flores et Kremer, un modèle social pour les logiciels de conversation inter-agents est présenté, basé sur les engagements sociaux et leur négociation. Nous partons du fait que les conversations sont piliers pour soutenir les agents autonomes et hétérogènes, et que la concordance conversationnelle peut être soutenue par des définitions publiques des actes de la parole et de sémantiques compositionnelles.

Un modèle social unifié pour les conversations est indiqué, dans lesquelles la sémantique d'acte de la parole est un produit émergent d'identité, d'utilisation conversationnelle, et d'accomplissement prévu, et où la composition conversationnelle est guidée par des règles d'utilité conversationnelle et de leur application à l'état d'instances conversationnelles. Pour finir, l'efficacité de cette approche nouvelle est montrée en décrivant formellement l'évolution d'une conversation simple pour l'action.

Protocole de proposition (protocol for proposal)

Nous définissons un protocole de base pour la négociation des engagements sociaux. Ce protocole appelé protocole de proposition (PFP), est officieusement décrit comme suit: Le protocole débute avec une proposition d'un expéditeur à un récepteur concurremment pour adopter ou décharger un engagement social indiqué. Soit récepteur répond avec une acceptation, un rejet, ou une contre-offre, ou l'expéditeur publie un retrait ou une contre-offre. Toutes les expressions excepté la contre-offre se terminent cette instance du protocole. Une contre-offre est considéré comme proposition, dans le sens où son expression est suivie de n'importe quelle réponse (actes de la parole) (mais de ces rôles d'expediteur-destinataire sont inversés si le destinataire original est l'orateur de l'expression). Quand une acceptation est publiée, l'orateur et le destinataire s'appliquent simultanément les engagements de proposition à leur historique d'engagements sociaux partagés.

La figure12 montre un exemple limité d'interaction du ***protocole de proposition.***

L'interaction au niveau de cette figure débute par une expression de la part de l'agent A contenant un « **illocutionary point** » PROPOSE. L'agent B peut répondre à cette proposition par une acceptation (ACCEPT), un rejet (REJECT) ou une contre-offre (COUNTER).

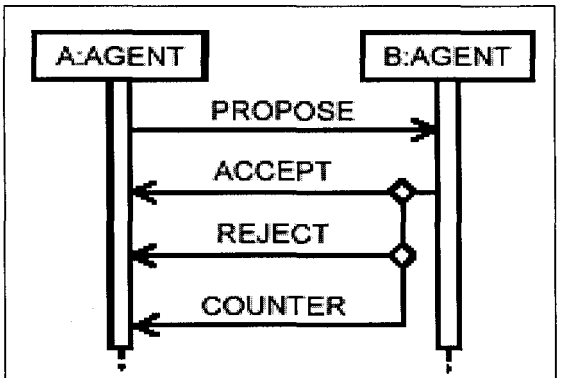


Figure 1. Example Interaction Diagram for the Protocol for Proposals.

Figure 12: Exemple de diagramme d'interaction du protocole de proposition

- **Spécification du protocole "protocole de proposition"**

Pour supporter ce protocole, les spécifications suivantes sont produites (avec leurs effets correspondants) :

- **Spécification 1** : Les agents pourront proposer d'autres agents pour considérer la prise partagée des engagements sociaux.
  a) Une proposition oblige *(engagent)* les agents proposés à répondre
- **Spécification 2:** Les agents proposés pourront accepter une proposition
  a) Une acceptation libère les agents proposés de l'engagement de répondre
  b) Une acceptation (au sein de la fenêtre d'interaction indiquée dans la proposition) réalise la prise partagée des engagements sociaux proposés
- **Spécification 3:** Les agents devraient être en mesure de rejeter une proposition
  a) Une offre libère les agents proposés de l'engagement de répondre.

• **Spécification 4:** Les agents devraient être en mesure d'effectuer une contre-offre au niveau de la proposition.

a) Une contre-offre libère les agents proposés de l'engagement de répondre

b) Une contre-offre engage les agents « *contre-offerts* » à répondre au sein de la fenêtre de l'interaction

• **Spécification 5:** Les agents contre-offerts devraient être en mesure d'accepter une contre-offre.

a) Une acceptation libère les agents contre-offerts de l'engagement de répondre

b) Une acceptation (au sein de la fenêtre d'interaction indiquée dans la contre-offre) réalise la prise partagée des engagements sociaux contre-offerts

• **Spécification 6:** Les agents devraient être en mesure de rejeter une contre-offre

a) Le rejet (rejection) d'une contre-offre libère les agents proposés de l'engagement de répondre.

• **Spécification 7:** Les agents devraient être en mesure d'effectuer un contre-offre à une contre-offre.

a) Une contre-offre libère les agents précédemment contre-offerts de répondre à cette contre-offre au sein de la fenêtre d'interaction.

b) Une contre-offre oblige (*engagent)* les agents contre-offerts à répondre.

- **Engagements conversationnels: Retrait des engagements conversationnels**

Une fois que les agents ont adopté les engagements conversationnels, ils peuvent attendre leur "retrait" une fois qu'une de ces trois conditions suivantes est satisfaite :

- **Condition 1.** Un agent proposé (ou contre-offert) pousse un acte de la parole contenant un ACCEPT avec la même opération sur l'engagement que celui du PROPOSE (ou CONTRE-) précédemment poussé.

- ***Condition 2.*** La proposition où l'agent proposé pousse un acte de la parole contenant un REJET avec la même opération sur l'engagement que celui du PROPOSE précédemment poussé

- ***Condition 3.*** Les agents qui proposent (ou contre-offrent) ou à qui on a proposé (ou contre-offert) pousse un acte de la parole contenant un CONTRE rejetant la même opération sur l'engagement que cela du PROPOSENT (ou CONTRE-) précédemment poussé.

3.4.2.4 Architecture pour la Planification Argumentative du Dialogue (Reed)

Les arguments représentent une occasion pour qu'un système convainque une audience probablement sceptique ou résistante de la véracité de ses propres croyances. Ces capacités sont des composants essentiels d'une bonne communication, facilitant l'explication, l'instruction, la coopération et la résolution de conflit. Une proposition est présentée pour l'architecture d'un système capable de construire ces types d'arguments. La conception de l'architecture se sert de la richesse des arguments naturels, qui, à la différence du langage naturel, est en particulier de convenance par rapport à l'analyse due à ses objectifs et structures claires. L'environnement proposé est basé sur un planificateur hiérarchique de noyau conceptuellement coupé en quatre niveaux de transformation, le niveau le plus élevé étant responsable des guides abstraits, intentionnels et pragmatiques, et le niveau le moins élevé étant responsable de la réalisation en langage naturel. Les niveaux plus élevés auront le contrôle pas simplement de la forme logique de l'argument, mais également des sujets de style et de rhétorique, afin de produire des arguments les plus incontestables et convaincants possibles.

Composantes d'un système argumentatif

L'architecture proposée repose dans un cadre hiérarchique, reflétant les niveaux distincts et inter-reliés de la structure dans des arguments identifiés à travers leur analyse. Il y a une partie de la synthèse des arguments qui se soucie de la résolution de la syntaxe, de l'expression et de la morphologie, comme avec n'importe quelle synthèse de langage naturel. Ceci comporte le niveau le plus bas. Au-dessus de ceci, là repose un niveau responsable de la coordination des relations de phrase et « d'inter-phrase » (« inter-sentence »), structure imposante à un niveau plus abstrait. Ce niveau intermédiaire est également responsable de la manipulation des aspects de communication tels que la focalisation (Grosz et Sidner 1990). En conclusion, aux niveaux pragmatiques plus élevés, de nouvelles machines et techniques sont exigées pour produire des arguments incontestables. Cette génération peut être accomplie par deux niveaux élogieux, dont un manipule les aspects structuraux de l'argument, et les autres modifications complexes à

ces structures et dispositifs stylistiques de contrôle. Le niveau de structure d'argument (AS), est au niveau le plus élevé de l'abstraction, puisqu'il produit la forme logique de l'argument utilisant les structures de données purement intentionnelles, en employant les opérateurs logiques, opérateurs d'erreurs et les opérateurs inductifs. Cette forme est alors augmentée et modifiée par le niveau subalterne de génération d'éloquence (EG), qui utilise l'heuristique basée sur la rhétorique et les paramètres contextuels, et est concerné par des propriétés d'un discours telles que sa longueur, détail, mètre, le réarrangement des sub-arguments, grouper des sub-arguments, la contraction d'enthymème, l'utilisation de répétition, l'allitération, et ainsi de suite. Cette architecture est récapitulée ci-dessous dans Fig.13.

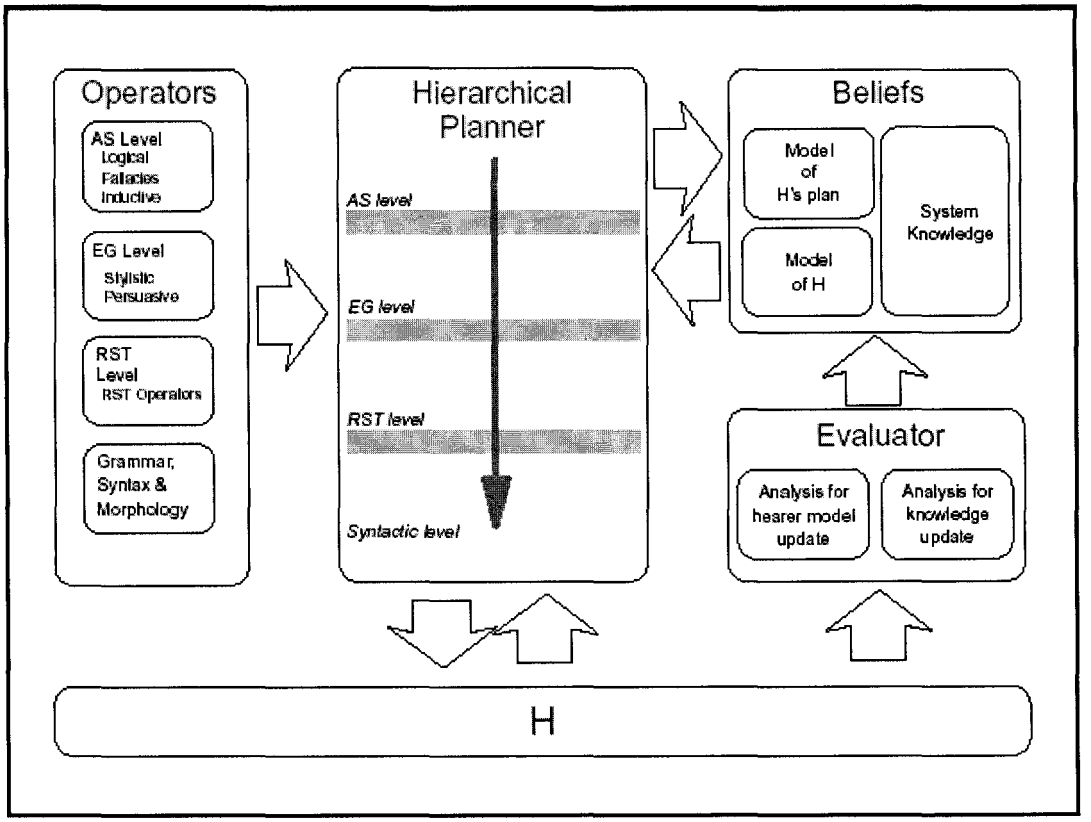


Fig. 1. System architecture overview

Figure 13: System architecture Overview

Composantes de support

- **Modélisation des croyances**

Il est apparu clairement que tous les deux niveaux AS et EG exigent l'accès au modèle de croyance de l'audience, en outre, naturellement, à la croyance du système. Le niveau AS doit pouvoir évaluer à l'avance comment les différentes structures sont susceptibles d'être reçues, et de tenir compte de la croyance de l'audience au niveau de la production de la structure. Le niveau EG utilise le modèle de croyance pour lancer un argument au niveau juste du détail, et pour dépister la pertinence de la croyance utilisée pendant l'argumentation.

- **Planification Hiérarchique**

Le rôle de la planification dans la construction des arguments est significatif: les arguments sont construits pour réaliser des buts et sont établis hors des structures primitives d'argument. Ce procédé de construction inclut la considération des effets des divers composants sur la croyance de l'auditeur, de l'écoulement de l'argument, de son foyer et de sa force et du point auquel chaque partie est soutenue par ce qui la précède. Ces problèmes sont toutes les interprétations, dans le domaine de synthèse d'argument, des problèmes classiques de planification, ainsi il ce n'est pas surprenant que la planification traditionnelle a un rôle à jouer dans ce domaine.

- **Génération de l'Éloquence**

Une des fonctions principales des niveaux AS et EG est le maintien d'une représentation de l'intention derrière des expressions. RST a été critiqué pour sa manipulation restrictive de l'intention (Moore et merlans jaunes 1992), (des jeunes et Moore 1994), un ingrédient indéniablement essentiel dans le discours. Ainsi, les niveaux

AS et EG doivent manœuvrer les structures de données plutôt différentes de ceux utilisés à des niveaux plus bas (qui sembleraient tout à fait raisonnables, vu la différence en structures de données au niveau de RST comparé à des niveaux plus bas). En raison de ces différences importantes, le travail courant se concentre spécifiquement sur la fonctionnalité plus élevée et plus abstraite. La fonctionnalité restante et plus basse exigée réalise les structures intentionnelles abstraites dans des expressions de langage naturel sera fournie par un système tel que LOLITA (Smith et autres 1994), une grande échelle, le système de langage naturel indépendant de domaine, dans lequel on met en application déjà un algorithme de génération de langage naturel qui englobe la responsabilité de résoudre certains problèmes de bas niveau de planification de génération des textes.

- **Interaction**

On prévoit que le système devrait être entièrement interactif, et entrer dans au dialogue de langage naturel dynamique avec un interlocuteur humain. Il y a un certain nombre de problèmes spécifiques à ce but, le plus évidemment, les différences entre le monologue et le dialogue. Le monologue et le dialogue exigent la modélisation des croyances de l'audience; pendant la production d'un monologue cependant, aucune opportunité n'est présentée pour la vérification, la mise à jour et le raffinement du modèle de l'audience, ce qui rend les arguments prolongés plus simples à construire (sans le besoin de modifier le plan selon la nouvelle information concernant la croyance de l'auditeur), mais par conséquent, diminue probablement la capacité à convaincre un auditeur de la proposition à disposition. Cela ne veut pas dire que le monologue est sans intérêt: tout à fait à l'effet contraire, vu que les arguments "à domicile" peuvent être vu comme des monologues de deux interlocuteurs opposes contrairement au vrai dialogue. Il serait probablement plus facile aborder travail sur l'argument monologue-formé à court terme, mais ça ignore beaucoup d'aspects critiques de modélisation qui sont d'intérêt à plus long terme, en particulier, dans le monologue les acceptations classiques de planification de la connaissance parfaite des effets et état d'environnement peut être utilisé, alors qu'avec le dialogue il est nécessaire de considérer les problèmes d'incertitudes, de la connaissance imparfaite (et de la nécessité d'acquérir la connaissance) et de l'échec de plan.

Conclusion

On a présenté un contour de l'architecture exigée pour construire des arguments prolongés du niveau le plus élevé des buts pragmatiques et riches d'intentions à une corde des expressions, en utilisant un planificateur hiérarchique de noyau utilisant les opérateurs structuraux, rhétoriques et heuristiques. L'architecture proposée est le premier cadre clair qui présente une vue informatique des moyens rhétoriques employés par les humains pour créer des arguments incontestables.

Le travail est actuellement en cours : caractériser les paramètres et les opérateurs impliqués aux premières parties de la synthèse des arguments, et relier ces derniers aux aspects des arguments uniques au dialogue, tel que la prise de tour, vérification des mouvements, etc., construites à partir du travail de Levin et de Moore (1977).

3.4.2.5 Jeux formels de dialogue (Mac Burney et Parsons)

Introduction

Mac Burney et Parson présentent un formalisme basé sur la logique pour modéliser des dialogues entre les agents intelligents et autonomes, construits sur une théorie de jeux abstraits de dialogue qui vont être présentés ultérieurement. Le formalisme permet la représentation des dialogues complexes comme séquences des mouvements dans une combinaison des jeux de dialogue, et permet à des dialogues d'être imbriqués entre eux. Le formalisme est informatique et sa nature modulaire permet à différents types de dialogues d'être représentés.

Dialogue et jeux de dialogues

- **Types de dialogues**

Dialogues de recherches d'informations (Information-seeking Dialogues): dialogues où un participant cherche la réponse à quelques questions d'un autre participant, qui est censé par le premier savoir la réponse

Dialogues d'Enquête : Dialogues où les participants collaborent pour répondre à un certaines questions dont la (les) réponse(s) n'est connue d'aucun du participant

Dialogue de persuasion: Inclus un participant cherchant à en persuader d'autres pour accepter une proposition qu'il (ou elle) n'approuve pas actuellement.

Dialogues de négociation : Dialogues où les participants négocient au niveau de la division d'une certaine ressource rare. Ici, le but du dialogue (une division de la ressource acceptable pour tous) peut être en conflit avec les différents buts des participants

Dialogues de délibération: Les participants des dialogues de délibération collaborent pour décider quelles actions ou lignes de conduite devraient être adoptées dans une certaine situation. Ici, les participants partagent la responsabilité de décider la ligne de conduite, où, au moins, ils partagent la bonne volonté de discuter s'ils ont une responsabilité partagée.

Dialogues Éristiques: Dialogues où des participants se disputent verbalement comme produit de remplacement du combat physique, visant à exhaler des réclamations perçues.

La plupart des occurrences réelles de dialogue (humain et agent) impliquent le mélange de ces types de dialogue. Une transaction d'achat, par exemple peut débuter avec la demande d'un acheteur potentiel pour l'information d'un vendeur, procèdent à un dialogue de persuasion, où le vendeur cherche à persuader l'acheteur potentiel de l'importance d'un certain dispositif du produit, et puis la transition vers une négociation, où chaque parti offre de céder quelque chose qu'il désire en échange d'autre chose. Les deux partis peuvent ou peuvent ne pas se rendre compte de la nature différente de leurs discussions à chaque phase, ou des transitions entre les phases.

- **Jeux de dialogue (Dialogue games)**

Les jeux formels de dialogue se passent en interactions entre deux joueurs ou plus, où chaque joueur "se déplace" en formant des expressions, selon un ensemble de règles défini. Des modèles formels de jeux de dialogue ont été présentés pour plusieurs du dialogue atomique dans la typologie de Walton et de Krabbe. Ceux-ci incluent des dialogues de persuasion, dialogues de recherche d'informations (*information-seeking* dialogs) ; négociations; et discussions. En outre, on a proposé des formalismes de jeu pour certaines combinaisons des types atomiques, comme pour la formation des équipes et des intentions collectives, de ce qui composent des combinaisons des dialogues de persuasion et de négociation. En se basant sur ces modèles particuliers de jeu et sur des tentatives de soustraire les jeux généraux, on peut identifier plusieurs types de règles de jeu de dialogue. En supposant d'abord que les sujets de discussion entre les agents peuvent être représentés en une certaine langue logique. Les composants d'un jeu de dialogue sont alors:

Règles de commencement (Commencement Rules): Règles qui définissent les circonstances dans lesquelles le dialogue débute.

Locutions : Règles qui indiquent quelles expressions sont autorisées. Typiquement, les locutions légales permettent à des participants d'affirmer des propositions, permettent à d'autres de remettre en question ou contester des affirmations antérieures, et permettent ceux qui affirment des propositions qui seront plus tard remises en cause ou contestées de justifier leurs affirmations. Les règles de jeu de dialogue peuvent également permettre à des participants d'affirmer des propositions en y affectant des degrés différents d'engagements, par exemple: on peut simplement proposer une proposition, un acte de la parole qui nécessite moins d'engagements qu'une affirmation de la même proposition.

« **Combination rules** »: Règles qui définissent les circonstances dans lesquelles les participants expriment l'engagement à une proposition. Typiquement, l'affirmation d'une réclamation p au cours de la discussion est définie comme indiquant aux autres participants un certain niveau d'engagement, ou à l'appui pour, à la réclamation. Les systèmes de dialogue formel typiques établissent et maintiennent les ensembles publics d'engagements, appelés les stocks d'engagements, pour chaque participant; ces stocks sont habituellement non-monotoniques, dans le sens où les participants peuvent également rétracter des jeux d'engagement, bien que probablement seulement dans des circonstances définies.

Règles de terminaison (Termination rules): Règles qui définissent les circonstances dans lesquelles le dialogue finit.

- **Environnements de dialogue formel**

On présente maintenant un formalisme hiérarchique à trois nivaux pour des dialogues d'agent. Au niveau le plus bas sont les sujets (« topics layer ») qui sont les sujets des dialogues. Au niveau qui suit sont les dialogues eux-mêmes (des instanciations, des persuasions, des enquêtes, de l'ect, et des combinaisons de ces derniers) représentés au moyen de jeux formels de dialogue. Au niveau le plus élevé sont les dialogues de commande (« control dialogues »), où les agents décident quels sont les dialogues à entrer, si le cas se présente.

- Niveau sujet (Topic Layer)

Les « topics » sont des sujets à l'étude par les agents participants. Les matières peuvent se rapporter à de vrais objets du monde ou aux états d'affaires, et le formalisme présenté ci-dessous peut adapter à l'une ou l'autre interprétation.

- Niveau dialogue (Dialogue Layer)

Au niveau suivant dans la hiérarchie nous modélisons les types particuliers de dialogues en utilisant la théorie générique de jeux formels de dialogue (présentés dans section 3.4.2.5).

On examine les composants de cette théorie à leur tour. Premièrement, considérons les règles de commencement. Après, Locutions sont des démarches légales de dialogue entreprises par des participants de dialogue concernant les sujets de discussion, dans un jeu particulier de dialogue. Un tel mouvement peut inclure des affirmations, contestations, justifications, ect. Dans la plupart des jeux de dialogue, ces mouvements se réfèrent aux matières particulières p. que nous supposons que tous les jeux de dialogue contiennent une règle qui affirme que le participant d'un dialogue peut seulement effectuer des locutions dans le dialogue tandis que le dialogue est ouvert. Troisièmement, les règles de combinaison définissent qui différentes circonstances dialogiques.

En outre, quelques règles de combinaison peuvent indiquer pour chaque locution ce que d'autres locutions doivent l'avoir précédé, parce que il à pousser légalement. Ces locutions qui ne font pas constituer de telles conditions préalables avec précision l'ensemble de locutions valides au premier rond du dialogue, et ainsi nous avons une fonction particulière de combinaison qui trace.

- **Couche de contrôle (Control layer)**

Le niveau contrôle cherche à représenter la sélection de types de dialogues spécifiques et les transitions entre ces types.

Conclusion

Le formalisme de dialogue formel proposé permet la représentation de n'importe quel type du dialogue et une diversité large de combinaisons de dialogues et de types de dialogue; Ceci est effectué de manière purement syntactique, et potentiellement génératif ainsi que descriptif. Le formalisme fournit un cadre d'unification simple pour représenter les types disparates de dialogue. En outre, le formalisme présenté est modulaire, de sorte que d'autres types de dialogue puissent être insérés aisément dans le cadre, directement ou en tant que dialogues inclus. Puisque la structure de dialogue formel est un formalisme logique, elle peut également faciliter l'étude des propriétés formelles des protocoles de jeu de dialogue, par exemple leur complexité. La question des stratégies des participant dans des jeux de dialogue est un autre secteur favorable à l'analyse formelle par un formalisme logique.

3.4.2.6 Jeux de dialogues (Engagements (Maudet et Chaib-draa))

Structure des jeux

Nous partageons avec d'autres [24, 23, 6] l'opinion des jeux de dialogue comme structures réglant le mécanisme sous lequel quelques engagements sont discutés par le dialogue. Cependant, à la différence de [9], nous adoptons une approche « basée engagement » stricte dans la structure de jeu et exprimons les règles de dialogue en termes d'engagements. D'autre part, à la différence de [6], nous considérons différents moyens de traiter les structures des jeux, et nous précisons comment dériver tous les autres jeux à partir de quelques jeux de base de dialogue (considérant seulement le degré de force).

Dans notre approche, les jeux sont considérés comme des structures bilatérales définis par des conditions d'entrée (qui doivent être remplies au début du jeu), des conditions de sortie (définissant les objectifs des participants lorsque engagés une fois engagé dans le jeu), et les règle de dialogue. Comme expliqué précédemment, toutes ces notions sont définies en termes d'engagements (probablement conditionnel). Techniquement, les jeux sont conçus comme des structures capturant les différents engagements créés au cours du dialogue.

Pour résumer, nous avons les conditions d'entrée (*Ebd*), de succès (*Sbd*), et de règles de dialogue (*Rbd*) pour chaque jeu. Nous supposons également qu'il y a une sanction constante *sg* pour pénaliser les agents qui ne suivront pas le comportement dialogique prévu (comme décrit dans les règles de dialogue).

Fonte (grounding) des jeux

La question spécifique de la façon dont des jeux sont fondus à travers le dialogue est certainement une des plus délicates. On suppose que les agents peuvent employer quelques actes ("meta-actes) du dialogue pour manipuler la structure de jeu et pour proposer ainsi d'entrer dans un jeu, proposer de stopper le jeu, et ainsi de suite. Ainsi les agents peuvent échanger des messages tels que

Proposer.entrer(Al , Bob, g1)

là où g1 décrit une structure bien formée de jeu (comme détaillé ci-dessus). Ce message est une proposition de l'agent Al à l'agent Bob pour entre dans le jeu g1. Ceci signifie que les jeux peuvent avoir des statut différent: ils peuvent être ouverts, fermés, ou simplement proposés. La manière avec laquelle ce statut est traité dans la pratique est décrite dans un *jeu de contextualisation* qui règle cette communication de méta niveau. Par un premier compte simple de ce jeu, nous pourrait adopter la vue intuitive des jeux simplement ouverts par l'échange réussi d'un ordre de proposer/accepter. Cependant, les choses deviennent plus compliquées si nous voulons tenir compte de différents types de combinaisons (ouvert ou proposé). Tous ces types de structurations sont considérés dans un jeu de contextualisation.

Combinaison de jeux

Comme expliqué précédemment, la possibilité de combiner les jeux est une option très attrayante de l'approche. Nous détaillons maintenant les compositions des jeux que nous employons dans notre cadre. Décrivant ces genres de combinaisons, nous précisons les conditions dans lesquelles ils peuvent être obtenus, et leurs conséquences. Finalement, de telles conditions et conséquences devraient être incluses dans le jeu de contextualisation que nous bâtissons :

1. **Séquencement :** noté g1; g2, qui signifie que g2 débute immédiatement après la terminaison de g1.

   Conditions : jeu g1 est fermé.

   Effets : terminaison du jeu g1 implique entrer dans le jeu g2.

2. **Choix,** signifie que les participants jouent soit au jeu g1 ou g2 de manière non-déterministique. Cette combinaison ne possède aucune spécification et ne possède aucune conséquence.

3. **Pré-séquencement,** signifie que g2 est ouvert pendant que le jeu g1 est proposé.

   Conditions : le jeu g1 est proposé.

   Effets: La terminaison satisfaisante du jeu g1 implique entrer dans le jeu g2.

De tels jeux de pré-sequencement peuvent être joués pour s'assurer que des états d'entrée d'un prochain jeu sont réellement établis, par exemple pour rendre public une position conflictuelle avant d'écrire un jeu de persuasion. Au cas où ce le premier jeu ne serait pas réussi, le deuxième jeu est simplement ignoré.

4. **Imbriquer (Embedding)** Imbriquer, qui signifie que g1 est maintenant ouvert alors que g2 était déjà ouvert.

   Conditions : le jeu g1 est ouvert.

   Effets : Les engagements (conversationnels) des jeux imbriqués sont considérés de manière prioritaire par rapport à ceux du jeu imbriqué. Beaucoup de travail doit être

effectué avec précision pour définir cette notion au sein de ce cadre, mais ceci peut être capturé en contraignant les sanctions liées au jeu imbriqué pour être plus grand que ceux du jeu imbriqué.

CONCLUSION

La question qu'on peut se poser est :
«Les modèles peuvent-ils résoudre les problèmes de flexibilités et de spécification ? ».
Pour répondre à cette question on va traiter les deux aspects qui sont notamment la flexibilité et les spécifications pour voir quels sont les changements que les modèles présentés ont apportés.

- **Flexibilité**

Maintenant l'évolution du dialogue comme suggérée dans notre approche, est un point clé pour rendre protocoles de conversation plus flexible. De ce fait enregistrer les engagements des participants dans un calendrier, comme suggéré dans notre simulateur de jeu (voir ci-dessous), références possibles de marques futures à ces engagements au lieu à de se rapporter juste au mouvement précédent du dialogue. En outre, les engagements publics comme suggérés dans notre approche motive des agents à se conformer à un certain comportement prévu et pour faciliter ainsi la coordination dans le dialogue. Mais les agents restent autonomes et peuvent décider de violer les engagements et de payer les sanctions s'ils trouvent de bonnes raisons de faire ainsi. Ainsi l'approche semble offrir un équilibre intéressant entre la normativité et l'autonomie. Nous avons également employé la composition de petits protocoles de conversation et cela offre d'avantage de flexibilité dans le sens où ces jeux de base peuvent se composer pour n'importe quel autre jeu complexe. En conclusion, le jeu de contextualisation offre la possibilité de passer du jeu en jeu pendant la conversation, mais doit être étudié plus soigneusement.

- **Spécification**

Notre approche est basée sur des engagements publics et par conséquent elle évite de se rapporter aux états internes des agents. Elle emploie également quelques règles déclaratives facilitant de ce fait la conception de la plupart des jeux tout en améliorant la clarté et l'expressivité. Ceci nous pousse à représenter des jeux en XML, une langue explicite standard. En conclusion, notre approche évite le problème de "sur-spécification" de l'exemple de Winograd, et on peut indiquer, par exemple

(a) le cas où la demande de l'initiateur peut être honorée sans acception explicite de l'associé;

(b) le partenaire peut lancer une offre sans la demande explicite de l'initiateur;

(c) l'initiateur peut demander si l'engagement est satisfaisant.

Il est clair que beaucoup de travail doit être effectué concernant quelques aspects fondamentaux de l'approche: les mécanismes des engagements et des sanctions doivent être explorés pour être exploitables par quelques agents, la contextualisation et les combinaisons des jeux auront besoin certainement de corrections lorsque confrontés à d'autres des études de cas. Cependant, par l'essai d'étudier à la lumière d'un exemple classique comment cette approche prometteuse des jeux de dialogues peut être employée, nous estimons que cet article contribue de manière significative au développement des politiques conversationnelles flexible futurs.

3.4.2.7 Protocoles de négociation flexible *(Dastani et al)*

Présentation

Dastani et ses collègues (Dastani et al, 2000) ont proposé une méthodologie pour construire des protocoles flexibles de négociation basés sur des jeux de dialogues, suivant le travail de (Hulstijn, 2000b). Malgré le fait que leurs agents de négociation puissent avoir des intérêts de concurrence, ils partagent un but commun pour coordonner leurs actions. Pour représenter les actions coordonnées, les auteurs ont employé les représentations partielles, appelées les recettes. Un type particulier de ces recettes est constitué par les jeux de dialogue.

La notion principale de leur protocole de négociation est la cohérence. Une expression ou un mouvement dans un dialogue de négociation est logique avec le contexte de dialogue, si :

(i) il adapte un plan qui pourrait réaliser les buts apparents de l'agent, et
(ii) il adapte les règles courantes d'interaction.

Selon l'attitude de l'agent et sur qui a l'initiative, le (i) ou le (ii) précède.

Bien que cette approche se soit concentrée sur la négociation, le cadre est censé être assez générique pour accepter différents types de dialogue. Des expressions sont capturées par des actes de dialogue, classiquement composés de contenu sémantique et de fonction communicative. Un enregistrement conversationnel (contexte de dialogue) garde trace des affirmations et des engagements faits jusqu'ici dans le dialogue. La fonction communicative a une fonction liée-tâche et/ou niveau-interactif. La cohérence est une question de cohérence de niveau-tâche et de niveau-interactif. Tandis que la cohérence niveau-tâche peut impliquer l'inférence complexe de plan basée sur les informations disponibles dans l'enregistrement conversationnel, la cohérence niveau-interactif est mieux est capturée dans des recettes pré-planifiées pour les actions communicatives communes (Hulstijn, 2000a) inspirées par des échanges humain fréquent, plus précisément les jeux de dialogue présentés ci-dessous.

Structure

Structure: Au niveau-interactif, un acte de dialogue est considéré initiatif ou réactif. La structure de base de jeu (échange de base) est une simple unité iniative/réactive (unité IRE, comme représentée sur la figure 14), qui simplement capture qu'un acte initiatif peut être suivi d'une réponse positive, négative, ou reformulée.

$exchange(a, b, \zeta) =$	$initiative(a, b, \eta);$ $[pos\_response(b, a, \zeta)$ $\mid neg\_response(b, a, \zeta)$ $\mid retry(a, b, \xi)],$ where $M_{a,b} \models coherent(\eta, \zeta)$

Figure 14: The Initiative/Reactive (IR unit) structure

La plupart des échanges ont la forme indiquée sur la figure 14 où on permet un échange, étant donné que la contrainte de cohérence sur le contenu sémantique de l'initiative et de la réponse est rencontrée. En d'autres termes, la réponse doit adresser l'initiative. En cas de réponse, la teneur de la réponse, dénotée par « ζ » ci-dessus, est considérée fondée (grounded) (M a;b représente le matériel fondé de la conversation). Par exemple, l'exemple de la figure 15 (pris de (Dastani et autres, 2000)) montre un échange cherchant de l'information.

$information\_seeking(a, b, \psi) =$	$question(a, b, ?\phi); (answer(b, a, \psi)$ $\mid clarification\_question(b, a, ?\chi)),$ where $M_{a,b} \not\models \neg\psi$ $(consistent)$ and $M_{a,b} \not\models \psi$ $(informative)$ and $M_{a,b} \models relevant(?\psi, \phi)$ and $M_{a,b} \models licensed(?\psi, \phi).$

Figure 15: Exemple d'échange pour obtenir de l'information

Cet exemple montre que pour une réponse à une question, les auteurs ont besoin qu'elle soit conforme, instructive, appropriée à la question, et non « sur-instructive » en ce qui concerne la question.

Composition

Composition: Les jeux peuvent être statiquement composés au niveau de la définition par ordonnancement ou enchaînement. D'abord considérons le cas de la combinaison séquentielle (voir la fig. 16)

$$game(a, b, (\eta.\zeta)) = \begin{array}{|l} exchange(a, b, \eta); game(a, b, \zeta); \\ \text{where } M_{a,b} \models coherent(\eta, \zeta) \end{array}$$

Figure 16: Sequential Combination of games

Comme prévu, le résultat du jeu combiné dépend des résultats des jeux le composant. La nature récursive de la définition indique qu'il est possible de combiner autant de jeux qu'on veut. Comme dans le cas de base d'échange, quelques contraintes de cohérence sont énoncées entre le sujet des jeux. Par exemple, pour être combiné, les jeux doivent partager des thèmes de terrain mutuel. Maintenant tournons-nous vers l'enchaînement. Les contraintes exigent que le dernier acte de dialogue (réactif) du premier jeu, soit le premier (initiative) du deuxième jeu. Les exemples canoniques de telles structures d'enchaînement sont *question/réponse/évaluation* ou *proposition/contre-proposition/...* Les jeux peuvent également dynamiquement se composer pendant le dialogue. Des structures IR peuvent être imbriquées, ainsi les dialogues tel que I(IR)R sont bien formés. Cependant, on ne propose aucune contrainte sur de telles combinaisons.

Grounding

« Grounding »: Tandis que les auteurs mentionnent que quelques phases de négociation peuvent être utiles pour déterminer le jeu courant, aucun acte n'est proposé pour traiter cet aspect.

3.5 Les différents Modèles pour représenter de conversations

Actuellement, beaucoup d'efforts de recherche au sujet des interactions inter-agent se sont concentrés sur les bases sémantiques et pragmatiques de différents langages de communication inter-agent basés sur la théorie des actes de discours (Austin 1962; Searle 1969; Cohen & Levesque 1995; Finin,Labrou, & Mayfield 1995; 97 1997; MacEntire & McKay 1998). Cependant, les nouveaux travaux au niveau de la recherche concernant les actes de la parole, exemplifiés par des efforts tels que KAoS(Bradshaw 1996), Dooley Graphs(Parunak 1996), COOL(Barbuceanu & Fox 1995) et MAGMA(Demazeau 1995), essayent de représenter et de raisonner au sujet des relations au sein des conversations. Des formalismes ont été proposés pour modéliser des conversations: FSMs(Barbuceanu & Fox 1995; Chauhan 1997), Dooley graphs(Parunak 1996), Petri Nets(Koning, François, & Demazeau 1998), etc.

3.5.1 Présentation des différentes structures

3.5.1.1 Machine à états finis

La plupart des projets de modélisation de conversations se sont servi des machines a état finis (FSM). Les FSM sont simples, dépeignent l'écoulement des actions/communications de manière intuitive, et permettent plusieurs interactions séquentielles. Les machines à états finis, possèdent une représentation graphique intuitive. Cependant, ils ne sont pas adéquats pour modéliser des interactions plus complexes, particulièrement ceux avec un certain degré de simultanéité/parallélisme (concurrency).

En outre, il n'est pas facile de représenter l'intégration des protocoles. Les machines à état finis (FSM) (ou Statecharts) modélisent le parallélisme uniquement sous forme d'états et de transitions, sans possibilité de modéliser les interactions de manière parallèles (concurrentes). Lorsque ces machines sont employés pour représenter les séquences tolérées d'acte de parole dans les conversations, l'avantage est la facilité à

pouvoir conceptualiser et implémenter, de plus ils peuvent être adéquats pour l'interaction courante de plusieurs de types d'agents simples (Bradshaw, et autres, 1997). Cependant, ils ont limité la capacité de représenter plusieurs types de contraintes concernant les conversations (par exemple, les structures de but ou les informations de plus haut niveau au sujet de la synchronisation et de la sécurité globales).

3.5.1.2 Machines à états finis : FSM (Winograd & Flores)

Une conversation peut être facilement représentée par une machine à état fini. [Winograd et Flores 88] proposent une machine à états finis indépendante au niveau des domaines impliqués dans l'activité humaine, comme sur le schéma qui suit (figure 17). Les parties dans le discours sont marquées A et B. Les arcs représentent des actes de la parole, dans la terminologie employée par Winograd et Flores. Les cercles foncés absorbent des états (d'extrémité).

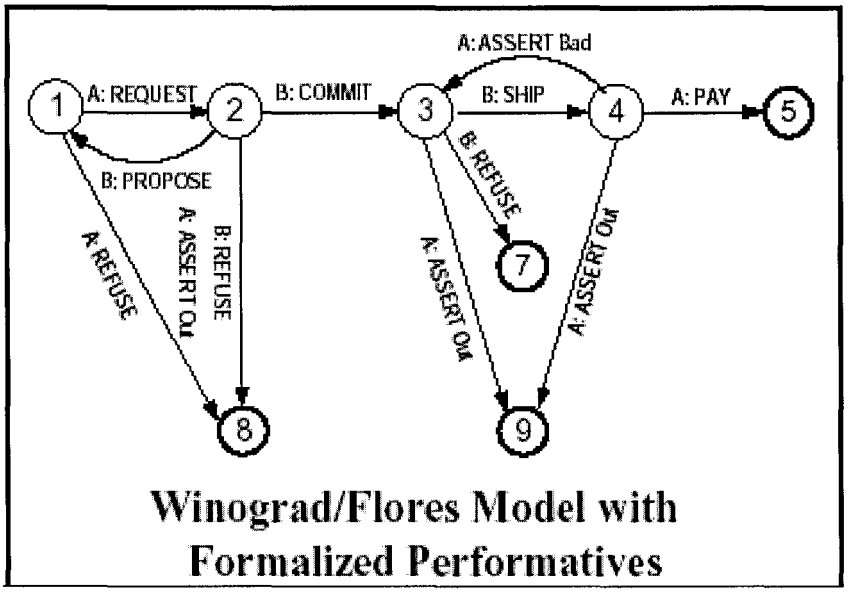


Figure 17: Modèle de Winograd/Flores avec des Actes de discours formalisés

3.5.1.3 Graphes de Dooley (Dooley Graph)

Le graphique de Dooley (Parunak, 1996) est un formalisme alternatif pour des corrélations de visualisation d'agent dans une conversation. Les méthodes orientées objet aiment l'offre d'UML (Booch et autres, 1997), une manière pratique de réduire le fossé entre les utilisateurs et l'analyste en considérant le transfert des messages, pourtant elles concernent uniquement le comportement dynamique des différents objets. Ainsi,le tableau 6, représente un exemple de conversation, suite à cela, la figure 18 représente la modélisation de cette conversation (tableau 6) par l'intermédiaire des *graphes de Dooley.*

Seq	Sndr	Addrs	Utterance	Responds to	Replies to	Re-solves	Com-pletes
1.	A	B,C,D	REQUEST: Please send me 50 widgets at your catalog price by next Thursday.				
2.	B	C	QUESTION: Are you bidding on A's RFQ?	1			
3.	C	B	INFORM: Yes, I am.	2	2	2	
4.	B	A	REFUSE	3	1	1	
5.	C	A	PROPOSE (INFORM + REQUEST): How about 40 widgets at catalog price by next Friday?	1	1		
6.	A	C	REQUEST: Please send me 40 widgets at catalog price by next Friday.	5	5	5	
7.	C	A	COMMIT: I plan to send you 40 widgets at catalog price by next Friday.	6	6	6	
8.	D	A	COMMIT: I plan to send you 50 widgets at catalog price by next Thursday.	1	1	1	
9.	A	C	ASSERT: I've found a better supplier, and am not relying on your COMMIT.	7,8	7		
10.	C	A	REFUSE: I'm abandoning my COMMIT.	9	9		7
11.	D	A	SHIP: Here are your widgets. Please pay me.	1	1		8
12.	A	D	ASSERT + REQUEST: You're five short. Please send the difference.	11	11		
13.	D	A	SHIP: Here are five more widgets. Please pay me.	12	12	12	
14.	A	D	PAY	13	13	13	

Tableau 6: Exemple de conversation

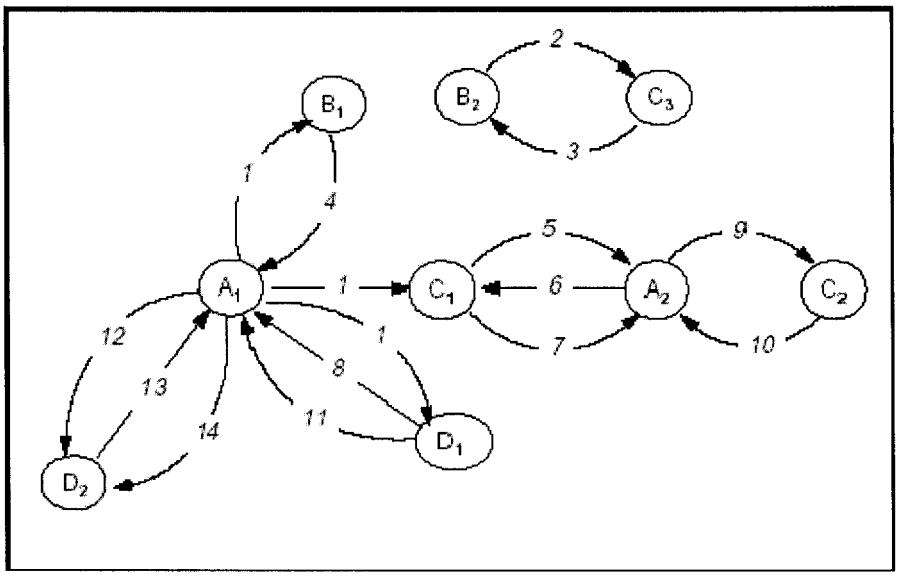


Figure 18: Graphes de Dooley représentant le tableau précédente (tableau 5)

3.5.1.4 Les réseaux de Pétri (Petri Net)

Les réseaux de Pétri sont des modèles bien connus, établis de simultanéité, et peuvent soutenir l'expression d'une plus grande gamme d'interaction. En outre, les réseaux de pétri, comme les machines à état finis (FSMs), ont une représentation graphique intuitive, sont relativement simples à mettre en application, et sont accompagnés d'une variété de techniques et d'outils pour la conception formelle. On peut voir une visualisation des concepts des réseaux de pétri au niveau de la modélisation en se referant à la figure 19. Il reste beaucoup de travail à faire avant qu'un catalogue complet de modèles de conception de réseau de Pétri soit disponible pour être appliqué aux tâches de modélisation pratiques. Nous voyons les domaines les plus importants:

- Définition de modèles et de calibres spécifiques appropriés, de domaine de modèle de présentation.

- Définition de calibres et de modèles de représentation au niveau de domaines spécifiques.

- Collection, identification, abstraction, et catégorisation du corps existant de la connaissance de conception de réseau de Pétri et les modèles subséquents.

- conception d'une langue suffisamment Générale de réseau de Pétri tenant compte de mécanismes puissants pour la composition et la paramétrisation.

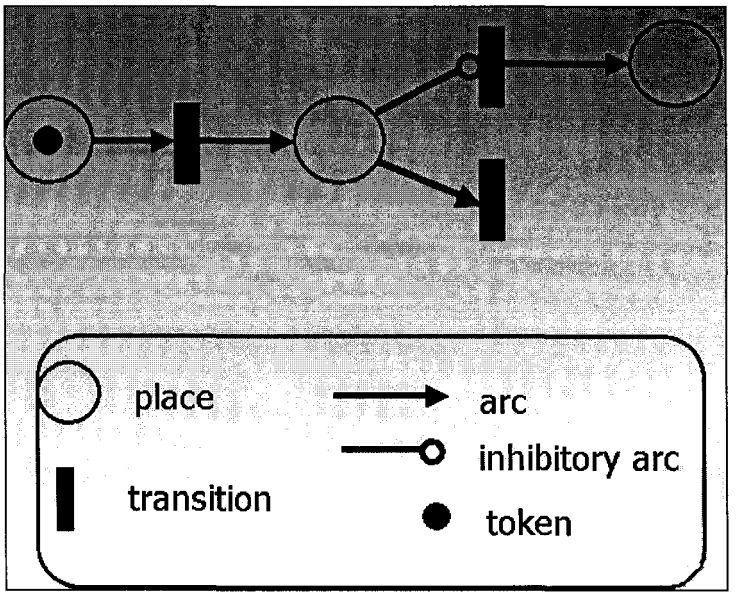


Figure 19: réseaux de pétri

3.5.2 En conclusion

Les approches communes pour modéliser les conversations des agents sont les réseaux de Pétri Colorés (CPN) et les machines à états finis (FSM). Tandis que les réseaux de pétri supportent la modélisation et les analyses qualitatives des interactions complexes, les machines à état finis (FSMs) souffrent de leur absence de simultanéité et de factorisation. Néanmoins cette représentation est employée dans plusieurs approches, au niveau des systèmes multiagents. Jusqu'à présent ici il n'existe aucun consensus permettant d'affirmer quelle est la meilleure manière de définir des spécifications pour des conversations.

Finalement, après être passé à travers les différentes sections de l'état de l'art, on a pu distinguer les différents concepts. Ces différents concepts vont permettre d'établir les spécifications (cahiers de charge) qui seront à la base de l'analyse, de la conception et du développement de notre environnement constituant l'objectif de mon mémoire (***ProtocolBuilder***).

4. SOLUTION PROPOSÉE: logiciel *ProtocolBuilder*

4.1 Analyse des caractéristiques voulues du logiciel.....115

4.2 Choix de conception pour ProtocolBuilder.....116

4.2.1 Modèle de cycle de vie adopté au niveau de la conception du logiciel.....116
4.2.2 Choix du langage de programmation : java.....116
4.2.3 Modélisation des protocoles de communications.....118
4.2.4 Choix de conception des conversations.....120
4.2.5 Choix de conception des performatifs.....121
4.2.6 Choix de conception des paramètres liés aux actes de discours.....122
4.2.7 Conception de l'aspect "création de protocoles".....123
4.2.8 Vérification des conversations par rapport aux protocoles crées.....123

4.3 Description du fonctionnement de " ProtocolBuilder".....125

4.3.1 Caractéristiques du logiciel.....125
4.3.2 Création protocole avec le logiciel « *ProtocolBuilder* ».....127
4.3.3 Vérification/validation des protocoles crées.....139
4.3.4 Évaluation.....145

Chapitre 4 : SOLUTION PROPOSÉE, logiciel ProtocolBuilder

4.1 Analyse des caractéristiques voulues du logiciel

On peut voir au niveau de la section 1.5 de ce document le but de mon projet qui est de réduire les problèmes identifiés au niveau de la communication inter-agent (section 1.4, section 2.1.1, section 3.2) par l'intermédiaire de la conception d'un environnement (logiciel) permettant de créer des protocoles de communication et de vérifier des conversations par rapport aux protocoles crées. Comment concevoir un environnement flexible, maniable, générique qui permet de concevoir des protocoles de communication et par la même occasion d'effectuer la vérification des conversations. Au niveau de la conception des protocoles, notre environnement devrait permettre de créer des protocoles flexibles/génériques. Pour cela, on devrait permettre :

- La création d'un ensemble d'actes de discours,
- La création des paramètres pour être associés à des actes de discours
- La réutilisation (l'importation) des actes existants (déjà créés),
- La réutilisation (l'importation) des paramètres existants (déjà crées),
- L'association de chaque acte de discours à une liste de paramètres.
- Pour limiter les ambiguïtés, permettre d'associer une explication/définition à chaque acte et à chaque paramètre et de visualiser ces définitions si on décide d'utiliser ces actes/paramètres.
- De plus, associer l'ensemble des actes de discours (et leurs paramètres respectifs) à une structure qui permettrait non seulement de modéliser le protocole de communication (conversation) mais également de suivre la trace de la conversation.
- Finalement de sauvegarder le tout dans un format qui pourra être révisé

Au niveau des conversations, on devrait avoir une marque indiquant quel protocole est utilisé, une liste de messages (séquences) représentant les interactions entre agents. Pour avoir une touche de standardisation, chaque message (séquence) devrait avoir une approche similaire aux messages des langages de communication FIPA ACL ou KQML

(langages de communication « standard » au niveau de l'intelligence artificielle). De plus, au niveau de la vérification des conversations, il y aurait deux volets à prendre en compte : d'une part, il faut être sûr que les actes de discours et les paramètres associés à chaque acte soient conformes aux spécifications du protocole de communication choisi, d'autre part il faudrait également faire une vérification du respect de l'ordre permis des séquences par rapport à la spécification du protocole en question (la structure qui permet de suivre la trace de la conversation sera utile à cet effet).

Finalement, pour des fins de qualités, l'environnement devrait avoir des propriétés de :

- Fiabilité : mesure l'aptitude du logiciel à conserver un comportement conforme aux besoins, même dans le cas d'événements non prévus ou non voulus.

- efficacité : l'efficacité mesure l'aptitude d'un logiciel à minimiser la consommation des ressources qu'il utilise (unité centrale, mémoire centrale, place disque, lignes de transmission...),

- flexibilité (maniabilité) : la flexibilité mesure l'aptitude d'un logiciel à être d'une utilisation agréable et facile/maniable pour l'utilisateur à qui il est destiné.

- portabilité : la portabilité mesure l'aptitude du logiciel de s'exécuter sur n'importe quelle plate-forme.

- utilisabilité : mesure l'aptitude être facile/maniable/agréable au niveau de l'utilisation et de la compréhension.

- Testabilité : la testabilité mesure l'aptitude d'un logiciel à faciliter la vérification de son comportement par rapport à des critères de test et de recette.

4.2 Choix de conception pour ProtocolBuilder

4.2.1 Modèle de cycle de vie adopté au niveau de la conception du logiciel

La production d'un logiciel comprend plusieurs étapes et s'étend sur une longue période de temps. Le cycle de vie d'un logiciel, c'est la période de temps qui débute au moment de la définition et du développement d'un produit logiciel et se termine lorsque le produit logiciel n'est plus disponible pour utilisation (Définition de l'IEEE). Au niveau de la réalisation du logiciel *ProtocolBuilder*, le modèle de cycle de vie utilisé tout au long est le modèle *incrémental*. Le modèle de cycle de vie incrémental, prend en compte le fait qu'un logiciel peut-être construit étapes par étapes. Le logiciel est spécifié et conçu dans son ensemble. La réalisation se fait par incrément de fonctionnalités. Chaque incrément est intégré à l'ensemble des précédents et à chaque étape le produit est testé exploité et maintenu dans son ensemble. Ce cycle de vie permet de prendre en compte l'analyse des risques et de faire accepter progressivement un logiciel par les utilisateurs plutôt que de faire un changement brutal des habitudes.

4.2.2 Choix du langage de programmation : java

Java est un nouveau langage de programmation développé au ***Sun Microsystems***, actuellement largement impliqué dans des applications au niveau World Wide Web, évidemment un système ouvert de distribution. Les propriétés les plus importantes de Java sont :

Orientation d'objet : Les programmes de Java se composent des descriptions de classe, définissant l'état et le comportement de la classe elle-même et de ses objets

Indépendance de logiciel d'exploitation, portabilité : ceci est réalisé en compilant le code source de Java au code byte de Java et en l'interprétant sur une machine virtuelle de Java, mise en application pour les logiciels d'exploitation les plus disponibles.

Parallélisme : Les « threads » fournissent des moyens d'exprimer des flux parallèles d'exécution en utilisant le même espace adresse. Les processus sont disponibles aussi.

Interconnexion de réseaux : Une application Java peut communiquer à une autre application sur l'Internet en utilisant le service de « Socket ». Les Sockets mettent en application le protocole de transmission de TCP/IP. En plus de ce mécanisme très simple de communication, une application de Java peut communiquer à une entité de WWW en utilisant le protocole de HTTP mis en application dans la classe d'URL.

Le mécanisme de communication le plus abstrait, actuellement disponible, permet d'adresser les objets à distance et d'appeler à distance leurs méthodes. Ceci peut être fait avec le package RMI (invocation à distance de méthode), fourni par Sun, ou avec une des différentes réalisations de CORBA.

Accès aux bases de données : Le paquet de JDBC permet d'accéder aux bases de données relationnelles que l'espace adresse commun permet pour une manière facile de la communication *inter-threads* : l'utilisation des données partagées. Un mécanisme de synchronisation, est nécessaire dans ce cas-ci. Le système d'exécution de Java fournit donc les moyens des moniteurs, garantissant qu'on permet à seulement un thread à la fois d'accéder à un certain bloc de code.

Création d'interfaces graphiques : Le paquet AWT ou SWING du langage de programmation oriente objet java permet de créer des applications s'exécutant à travers des interfaces graphiques. L'aspect visuel et interactif des interfaces graphiques facilite l'utilisabilité/maniabilité des logiciels.

4.2.3 Modélisation des protocoles de communications

- Choix de structure de modélisation : Modèle machine à états finis

Au sein de mon projet de mémoire, nous considérons quelques restrictions au niveau du genre de dialogues que nous allons traiter. Les dialogues que nous considérerons impliquent seulement deux agents qui de manière séquentielle alternent au niveau du dialogue. Ces restrictions (évitant notamment la simultanéité) vont nous permettre de nous concentrer sur une classe particulière des protocoles, à savoir ceux représentables à l'aide des automates finis déterministes (DFAs), il y a de nombreux exemples à trouver dans la littérature [Pitt et Mamdani, 1999 ; Dignum et Greaves, 2000].

L'interaction entre deux agents (ou plus) peut être modélisée à travers une machine à états finis (FSM) Ainsi, on peut se servir d'une machine à états finis pour modéliser des conversations ou des protocoles de négociations (ou un type de conversation). Les agents interagissent à travers le concept de « conversations ». Au sein d'une conversation, les agents échangent des messages selon des conventions établies au sein des agents qui communiquent, changent d'état et effectuent des actions. La figure 20 représente un exemple de message envoyé par un agent (*agent1*).

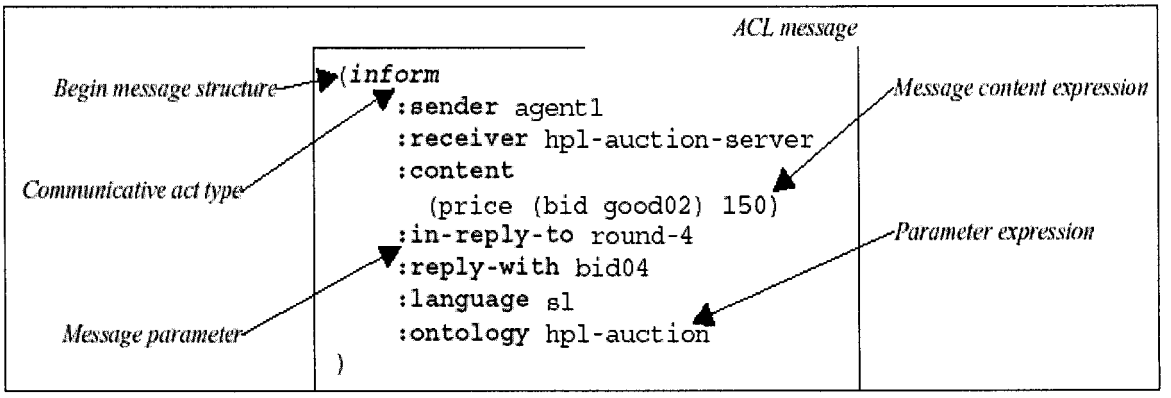


Figure 20: Exemple d'envoi de message

Les différents états d'une machine à états finis représentent les différents états possibles dans d'une conversation. Sachant que nous avons un état initial où la conversation débute et un (ou plusieurs) état(s) final (ou finaux) là où la conversation termine. Chaque arc de notre FSM représente la transmission d'un message. Les messages échangés se font par l'intermédiaire des performatifs (actes de discours) du langage de communication inter agent. (Au niveau de l'exemple de la figure 20, le performatif choisi pour représenter le message est « ***inform.*** » ayant pour but d'informer). La figure 21 représente la modélisation d'un protocole de négociation à travers une structure similaire à la structure de machine à états finis.

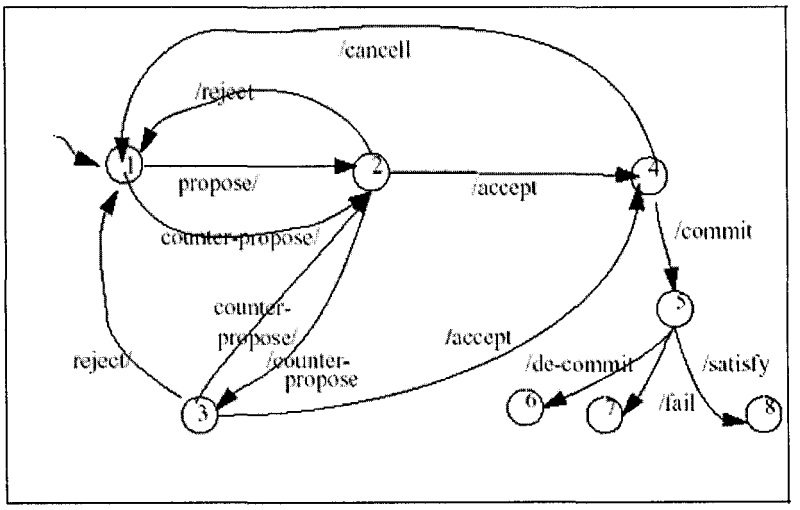


Figure 21: protocole "transitions d'états pour la négociation"

Explications de la figure 21

La Figure 21 utilise la notation <actes de discours reçus> / <actes de discours envoyés> pour marquer les bords au niveau du diagramme. Lorsque la conversation est dans un état donné et qu'un acte de discours est reçu, l'agent effectue un traitement local (ne figurant pas sur le diagramme), envoie l'acte de discours montré, et va vers l'état pointé par la flèche. L'état représente l'état initial et les états 6, 7 et 8 sont des états finaux.

Une conversation débute au niveau de l'état 1 avec la réception de l'agent d'un acte de discours. Si c'est une proposition, l'agent va vers le deuxième état (état 2). Dans cet état, on peut rejeter la proposition ou effectuer une « contre-proposition ».
Si une contre-proposition est lancée (etat3), elle peut être acceptée ou rejetée par l'interlocuteur. Si elle est rejetée, on peut entrer dans un autre cycle « proposition, contre-proposition ». Si une contre-opposition est finalement acceptée, la conversation va vers l'état 4. Ici l'agent peut changer d'avis (annuler, « cancel ») ou se soustraire à effectuer la sous-tâche (« subgoal »). À l'état 5, l'agent peut « report » satisfaction d'engagement, échec, ou non-engagement unilatéral.

Cet exemple de modèle de conversation et de coordination décrit l'échange de messages à partir du point de vue de l'agent qui satisfait les demandes des autres agents. À partir du point de vue de l'agent qui effectue une demande en premier lieu, le modèle de conversation sera différent.

4.2.4 Choix de conception des conversations

Au niveau des conversations, on a choisi comme convention d'établir la spécification des conversations à travers un nombre fini de séquences (message). De plus pour représenter chaque séquence (message), pour des fins de standardisation, on va s'inspirer des modèles déjà existant au niveau de langages de communications qui sont « standards ». La figure 22 représente un format représentant un message applicable au niveau de notre environnement.

```
performatif: ask-all
sender: a
receiver: b
```

Figure 22: format des messages

4.2.5 Choix de conception des performatifs

Au niveau des actes de discours, il faut prévoir la conception et réutilisation des actes de discours lorsqu'on décide de créer un protocole ou de vérifier si le bon performatif a été utilisé au sein de la conversation. Au niveau de notre environnement, on doit pouvoir créer des actes de discours en précisant la catégorie de l'acte crée (p.e FIPA ACL, KQML, ou une autre catégorie imaginée par l'usager), le nom de l'acte et sa signification, importer des actes de discours existants (déjà crées au niveau de la conception d'un autre protocole), de visualiser la signification de chaque acte avant de l'utiliser/l'importer. Une solution serait de stocker tous les actes crées au niveau d'un fichier précisant tout d'abord quelle est la catégorie des actes de discours, ensuite lister la liste des actes de discours de la catégorie suivi de leurs significations (explications) respectives. La figure 23 représente un format de fichier où ces informations (liste des actes de discours, leurs explications respectives, leur catégorie) pourraient être stockées.

```
CATEGORIE: fipakqml
PERFORMATIF: Evaluate   EXPLICATION:       evaluer
PERFORMATIF: ask-if     EXPLICATION:       demander si
PERFORMATIF: ask-about  EXPLICATION:       demander au sujet de
PERFORMATIF: ask-one    EXPLICATION:       demander a un
PERFORMATIF: ask-all    EXPLICATION:       demander à tous
PERFORMATIF: tell       EXPLICATION:       dire
PERFORMATIF: achieve    EXPLICATION:       accomplir
PERFORMATIF: deny       EXPLICATION:       nier
PERFORMATIF: untell     EXPLICATION:       ne pas dire
PERFORMATIF: unachieve  EXPLICATION:       ne pas avoir accompli
```

Figure 23: format des fichiers contenant les actes de discours

4.2.6 Choix de conception des paramètres liés aux actes de discours

Tout comme les actes de discours, on décide de stocker les paramètres des messages pour des fins de réutilisation, d'importation au niveau de la conception d'un protocole de communication, plus précisément au niveau de l'établissement des paramètres qui va être associés aux actes de discours préalablement crées/sélectionnés/importés. Au niveau des paramètres stockés, pour des fins de standardisation par rapport aux concepts de notre environnement, on va suivre le modèle de la représentation applicable des actes de discours (figure 23). On va donc stocker tous les paramètres au niveau d'un fichier précisant tout d'abord la catégorie des paramètres (FIPA ACL, KQML, ou une autre catégorie imaginée par l'usager), ensuite la liste des paramètres suivis de leurs significations respectives.

```
msgType: kqmlMSG

msgParam: performative      details: nom performatif
msgParam: sender            details: nom sender
msgParam: receiver          details: nom(s) recepteur(s)
msgParam: content           details: contenu
msgParam: language          details: language
msgParam: encoding          details: codage
msgParam: ontology          details: ontologie
msgParam: reply-with        details: répondre avec
msgParam: in-reply-To       details: En réponse à
msgParam: reply-by          details: Répondre avec
msgParam: reply-to          details: Répondre à
msgParam: conversation-id   details: ID de la conversation
```

Figure 24: format de fichier contenant les différents paramètres

4.2.7 Conception de l'aspect "création de protocoles"

La création d'un protocole devrait inclure une section qui saisit le nom des protocoles et les explications associées. Les protocoles se servent des actes de discours et de leurs paramètres respectifs pour définir les spécifications voulues des conversations inter-agent. Notre environnement va inclure une partie permettant sélectionner, de spécifier (créer) les actes de discours, les paramètres associés à ces actes, puis de créer une machine à états finis permettant de bien établir l'ordre permis des messages. La création de notre machine à états finis permettra de garder trace des actes de discours utilisés au niveau de chaque transition et de représenter la liste des actes permis au niveau de chaque arc de la machine le tout au sein du protocole (en cours de création). Les actes de discours devraient appartenir à un ensemble pour des fins de réutilisations. Ainsi lorsqu'on crée un protocole, on devrait pouvoir non seulement créer un ensemble (bibliothèque) d'actes de discours, mais on devrait pouvoir également importer les ensembles des actes de discours déjà créés. De plus, au niveau de la création d'un nouvel ensemble d'actes de discours, on devrait pouvoir importer des actes de discours existant afin de les intégrer avec le nouvel ensemble crée. Ensuite, le tout devrait pouvoir être sauvegardé pour des utilisations futures comme la vérification de conformités d'une conversation par rapport à un protocole (déjà crée).

4.2.8 Vérification des conversations par rapport aux protocoles crées

Au niveau de la gestion des protocoles de communication, on doit offrir la possibilité de sélectionner une conversation (fichier ayant comme extension « .conv ») existante, d'identifier et d'ouvrir le protocole utilisé, de visualiser chaque séquence (message) de la conversation. De plus, on devrait être en mesure de visualiser les paramètres (description protocole, liste des actes de discours, leurs explications et leurs listes respectives de paramètres associes) du protocole utilisé par la conversation. Ensuite, on pourra procéder à l'étape de la vérification. L'étape de vérification se divise en deux parties : la vérification de la conformité des messages (validité des actes de

discours et/ou des paramètres associes par rapport au protocole utilise), la vérification du respect de l'ordre des messages au niveau de la conversation par rapport aux actes de discours au niveau de la machine à états finis qui spécifie non seulement les actes de discours tolérés, mais également leur ordre. Notre environnement devrait permettre la création des protocoles et la vérification des conversations par rapport aux protocoles utilisés devrait se faire de manière assez facile, agréable, flexible, maniable au niveau de l'utilisateur.

Conclusion de l'étape d'analyse et de conception

L'étape d'analyse a permis de spécifier les différentes fonctionnalités, les différents services que notre environnement devrait fournir. L'étape de conception est composée de plusieurs sous-sections représentant les structures, les modèles de conceptions, les langages de programmation, les normes établies afin de concevoir notre environnement. Après être passé à travers les différentes étapes d'analyse et de spécification de notre logiciel, nous allons présenter notre environnement dans son ensemble dans la section suivante (4.3 : Description du fonctionnement de ProtocolBuilder).

4.3 Description du fonctionnement de " ProtocolBuilder"

4.3.1 Caractéristiques du logiciel

Le but du projet est de concevoir un environnement flexible de développement et d'expérimentation de protocoles de communication pour les systèmes multiagents, pouvant permettre la conversation inter-agent avec possibilité d'utilisation de protocoles multiples. La figure 25 permet de donner un aperçu des interfaces et des fonctionnalités hypothétiques qui permettent la communication inter-agent par l'intermédiaire de protocoles de communication spécifiées, et la possibilité de l'usager d'ajouter/modifier/sélectionner/retirer des protocoles spécifiques à cette communication, le tout au sein d'un outil logiciel (« ***Protocol Builder*** »).

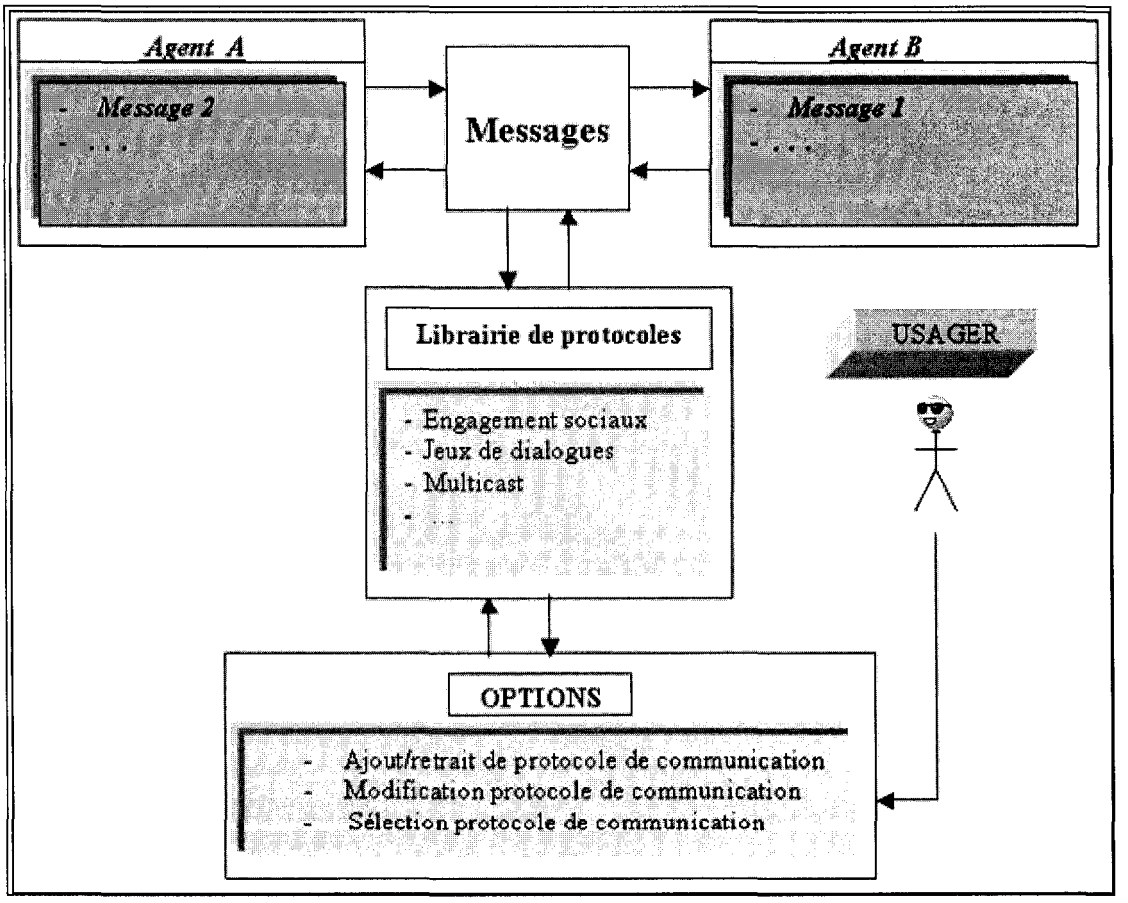


Figure 25: Modélisation de la solution proposée

Dans les sections qui suivent, on va effectuer une présentation des principales parties de ProtocolBuilder :

- Création de protocoles
    - Saisie du nom et la description du protocole en cours de création.
    - Définir /Sélectionner la liste des actes de discours
    - Définir une nouvelle étiquette
    - Établir une liste de paramètres message pour chaque acte de discours
    - Création de la liste des états pour simuler la conversation
    - Faire le lien entre états au niveau d'une machine à états finis
    - Après avoir effectué les liens entre les différents états
    - Fichiers générés

- Vérification/validation des protocoles crées
    - Interface permettant de visualiser les propriétés du protocole
    - Interface permettant la vérification de la conformité

Nous allons ensuite procéder à une évaluation du logiciel ProtocolBuilder. On va tout d'abord effectuer une simulation de conversation entre agents (avec l'outil Jack). Cette simulation va ensuite être évaluée avec ProtocolBuilder:

- Évaluation
    - Outil de simulation "JACK INTELLIGENT AGENTS"
    - Simulation du protocole ContractNet avec l'outil Jack

4.3.2 Création protocole avec le logiciel « *ProtocolBuilder* »

Au niveau de la partie "création de protocoles", dans les sous-sections qui suivent, nous allons présenter les différentes interfaces/fonctionnalités/options permettant de créer un protocole de manière générique/flexible.

4.3.2.1 Nom et description protocole

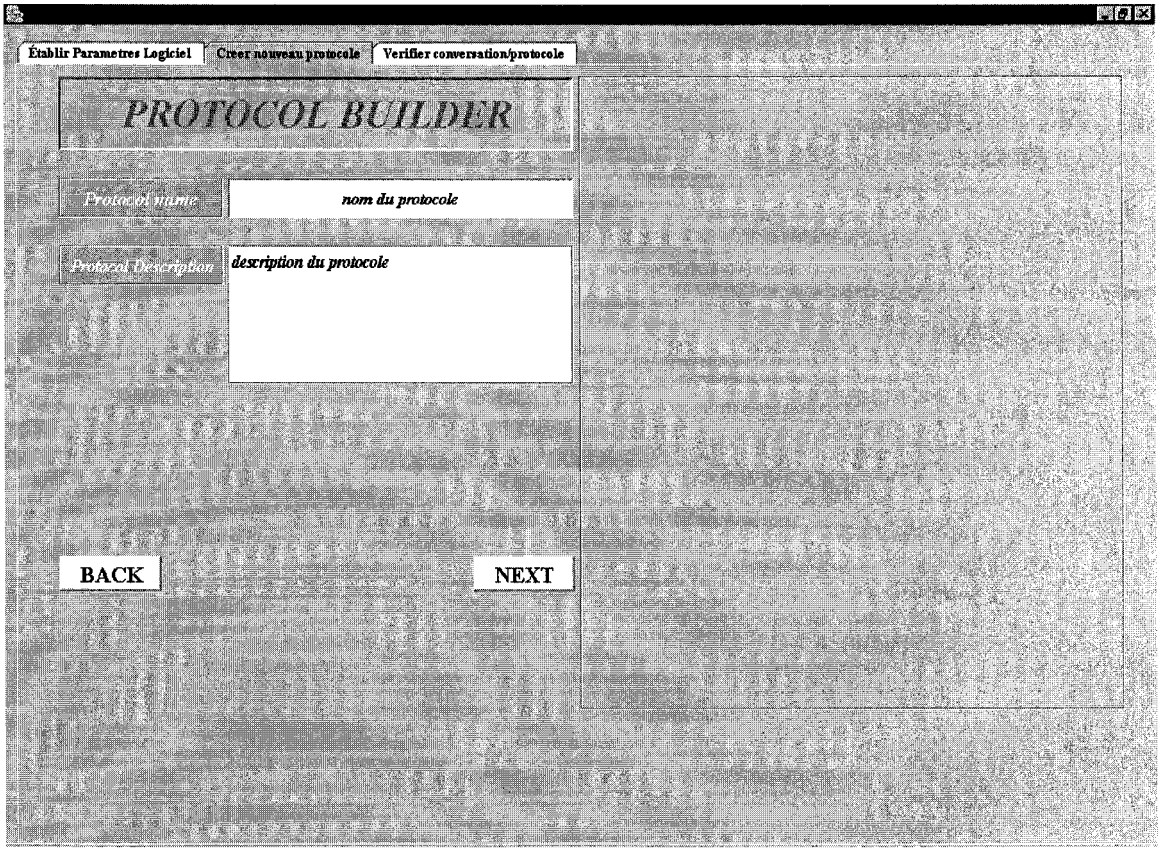


Figure 26: Interface permettant de saisir le nom d'un protocole et sa description

Chaque protocole possède un nom et une description, ainsi la première interface (figure 26) du logiciel *« protocolBuilder »* nous permet de saisir le nom et la description du protocole à créer.

4.3.2.2 Définir/Sélectionner la liste d'actes communicatifs

Comme on peut le voir au niveau de la figure 27, à partir d'une liste d'ensembles communicatifs, on laisse la liberté à l'usager de choisir les actes communicatifs voulus pour représenter le nouveau protocole sachant que les actes de discours permettent d'identifier le type d'action voulu de l'envoyeur par l'intermédiaire du message.

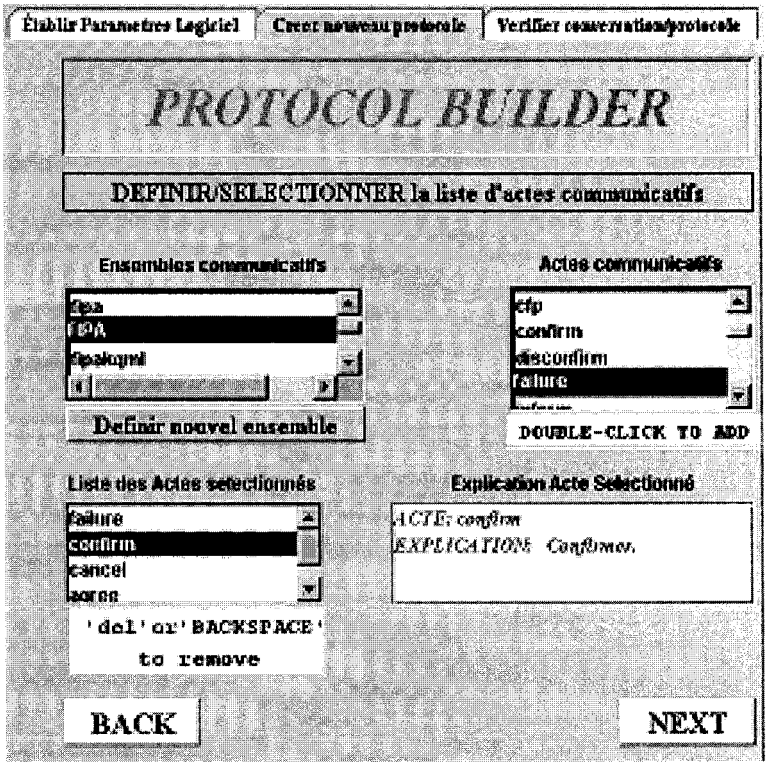


Figure 27: Interface permettant de saisir la liste des actes de discours

Au niveau de la figure 27, on peut identifier les ensembles communicatifs, la liste des actes de discours contenue au niveau de l'ensemble communicatif sélectionné (l'ensemble communicatif FIPA est sélectionné au sein de la figure 27). De plus, on peut non seulement voir la liste des actes sélectionnés au fur et à mesure qu'on les ajoute, mais on peut également voir les détails (explications) de chaque acte sélectionné. Au niveau de notre interface (figure 27), le fait de cliquer sur le bouton "Définir nouvel ensemble" nous mène vers l'interface permettant de définir un nouvel ensemble d'actes de discours. Nous

allons présenter la partie "Définir un nouvel ensemble (une nouvelle étiquette) au niveau de la section qui suit (section 4.3.2.3).

4.3.2.3 Définir une nouvelle étiquette

Au niveau de la spécification des actes de discours, on peut également définir un nouvel ensemble d'actes de discours pour ensuite les intégrer au protocole en cours de création. La figure 28 représente l'interface nous permettant d'effectuer une telle opération. Ainsi, comme la figure 28 le montre, on peut :

- Définir une nouvelle étiquette pour un nouvel ensemble de performatifs

- Importer des primitives existantes qui feront parti du nouvel ensemble. on peut voir par exemple qu'au niveau de la figure 28, la liste de performatifs de FIPA a été importée.

- Ajouter des performatifs au fur et à mesure (avec leur description) au sein de la liste de performatifs représentant la future étiquette.

- Retirer des performatifs qui ont été ajoutés au sein de la liste de performatifs, représentant la future étiquette.

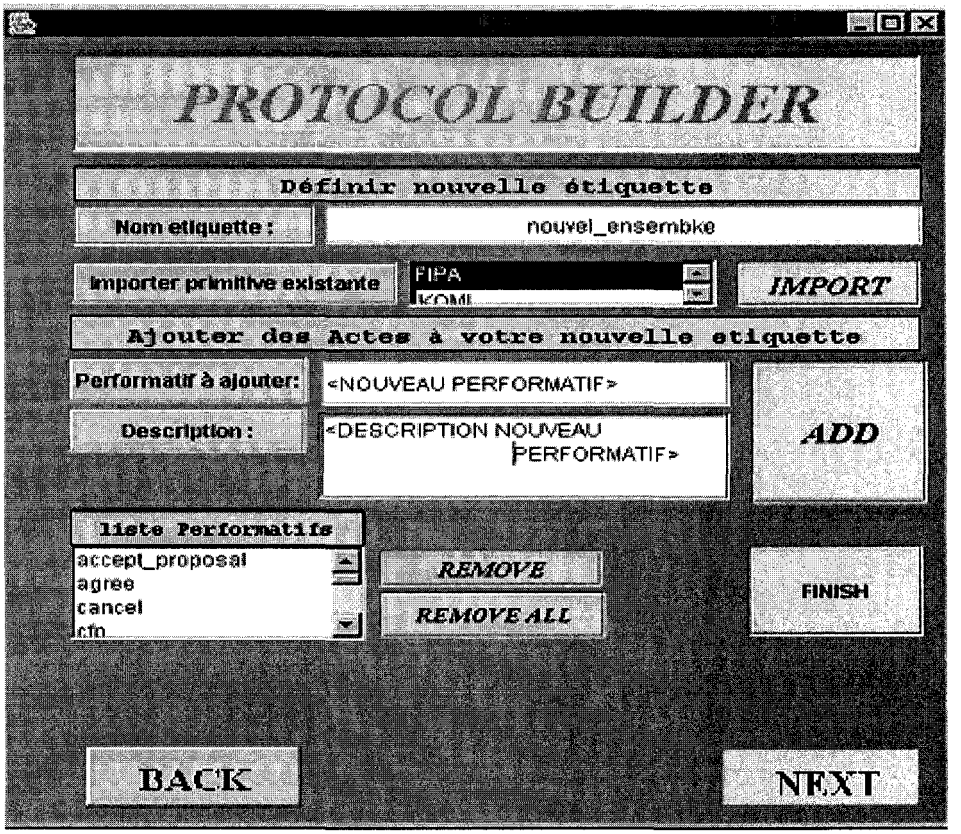


Figure 28: Interface permettant de définir un nouvel ensemble d'actes de discours

Après avoir ajouté un nouvel ensemble d'actes de discours, un fichier représentant cet ensemble est créé. Ainsi, le nouvel ensemble va être ajoute à la bibliothèque d'actes de discours déjà crées. La bibliothèque d'acte de discours en utilisant le principe de réutilisation permet non seulement de spécifier des actes de discours (section 4.3.2.2), de visualiser leurs significations respectives, mais également d'importer des actes de discours pour définir un nouvel ensemble d'actes de discours (sachant que dans un nouvel ensemble, une partie des actes peut déjà être spécifié).

4.3.2.4 Établir une liste de paramètres pour chaque acte de discours

La dimension primaire de l'extension de KQML est au niveau de la définition de nouveaux performatifs (Actes). Les définitions de nouveaux performatifs doivent explicitement décrire tous les paramètres permis, et si applicables, des valeurs par défaut pour les paramètres qui n'apparaissent pas, en particulier des messages. Une définition de performatif peut inventer de nouveaux noms de paramètre. À partir de l'interface de la figure 29,après avoir sélectionné la liste des actes communicatifs, on associe à chaque acte sélectionné une liste de paramètres de messages pouvant être issus de FIPA, KQML, ou bien une catégorie définie par l'usager.

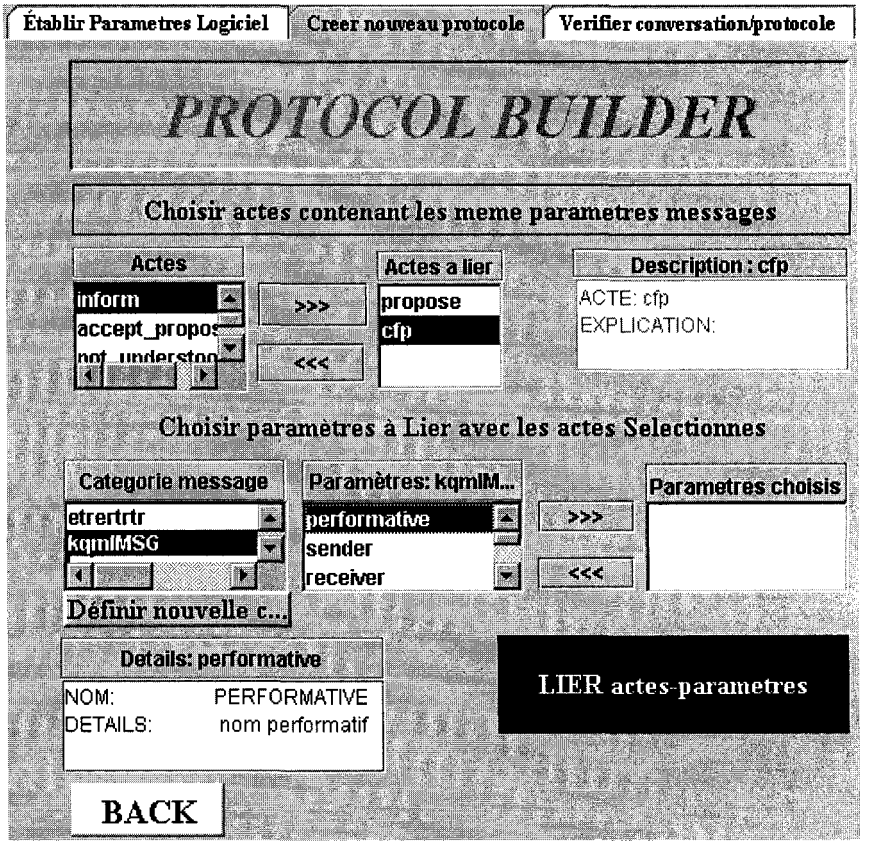


Figure 29: Interface permettant d'établir la liste des paramètres pour les actes choisis

Au niveau de la figure 29, on peut voir les actes sélectionnés (actes "propose" et "cfp"). On peut également voir que les différents paramètres (performative, sender, receiver), qui proviennent de la catégorie "kqmlMSG", sont prêts à être sélectionnés pour être associé (en cliquant sur le bouton "LIER actes-paramètres") à chaque acte de

discours (précédemment sélectionné : propose, cfp) en. De plus, à partir de notre interface (figure 29).

4.3.2.5 Création de la liste des états pour simuler la conversation

À ce niveau, par l'intermédiaire de l'interface représentée au niveau de la figure 30, on donne la possibilité d'établir la liste des états qui vont former une machine à états finis permettant de compléter et mieux visualiser le protocole à bâtir. L'interface au niveau de la figure 30 nous permet non seulement d'ajouter des états, mais de les retirer également. Une fois que la liste des états est établie, on pourra effectuer les liens entre les différents états existant afin de former un protocole.

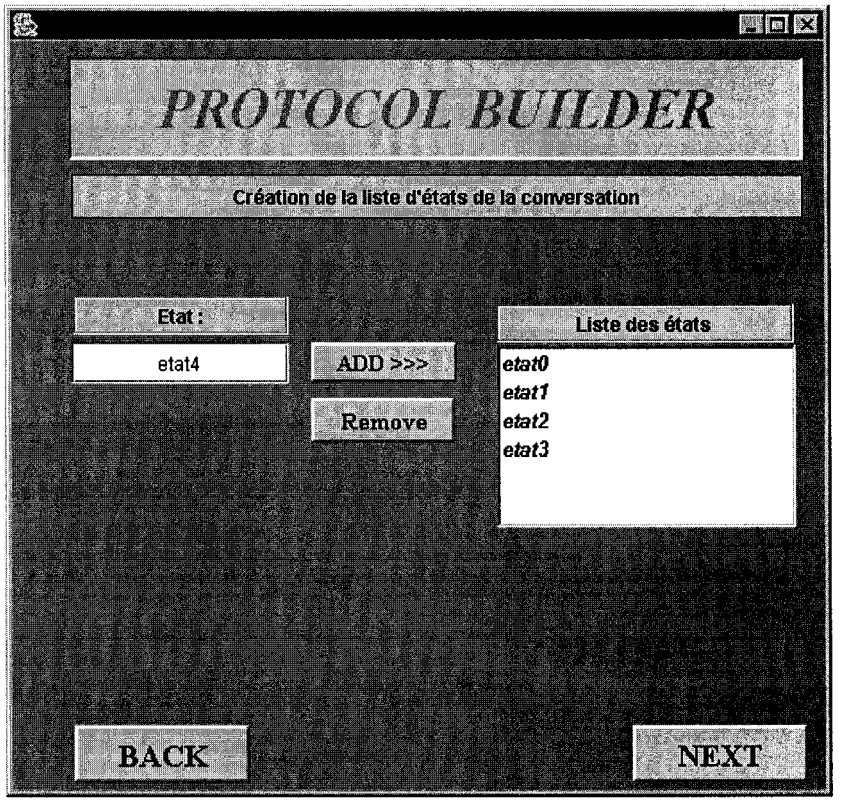


Figure 30: Interface permettant de saisir la liste des états de la machine à états finis

Ainsi, on peut voir au niveau de la figure 30 les quatre états (état0, état1, état2, état3) qui ont été ajoutés à la liste des états. Après avoir établi la liste des états, on accède à l'interface permettant de créer notre machine à états finis qui va gérer l'ordre de succession des différents actes de discours au sein du protocole (figure 31 au niveau de la section 4.3.2.6).

4.3.2.6 Faire le lien entre les états

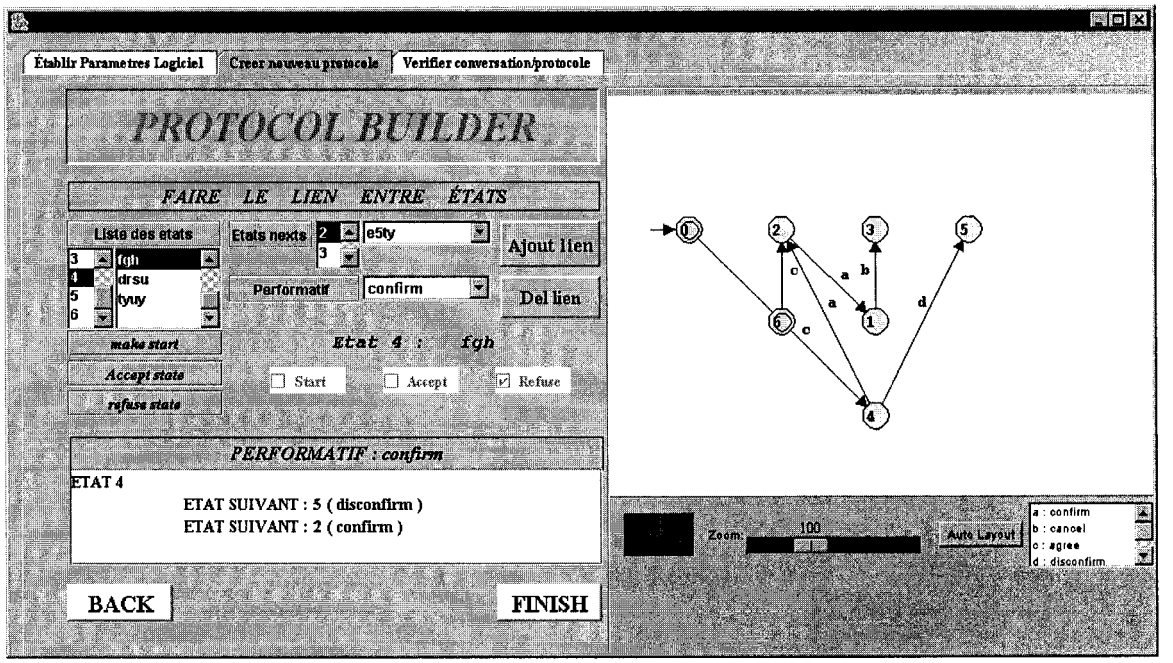


Figure 31: Interface permettant de faire le lien entre les états (machine à états finis)

Apres avoir recueilli toutes les informations (performatifs, paramètres messages (paramètres de chaque acte de discours), listes des états), nous sommes fin prêt à modéliser notre protocole par le biais de notre machine à état fini. Au niveau des états, on peut :

1 Rendre un état acceptant (accept state)

2 Rendre un état refusant (refuse state)

3 Transformer un état non initial en un état initial (initial state)

4 Lier un état avec un autre en spécifiant le performatif utilisé

5 Délier deux états liés (supprimer le lien entre deux états)

6 Finalement, on peut enregistrer notre machine à états finis (FSM), qui pourra éventuellement être visionnée lors d'un traitement futur.

4.3.2.7 Après avoir fait le lien entre les états

La figure 32, représente le résultat, après avoir effectué les liens entre les états tout en spécifiant les performatifs utilisés et les conditions associées au niveau de chaque lien.

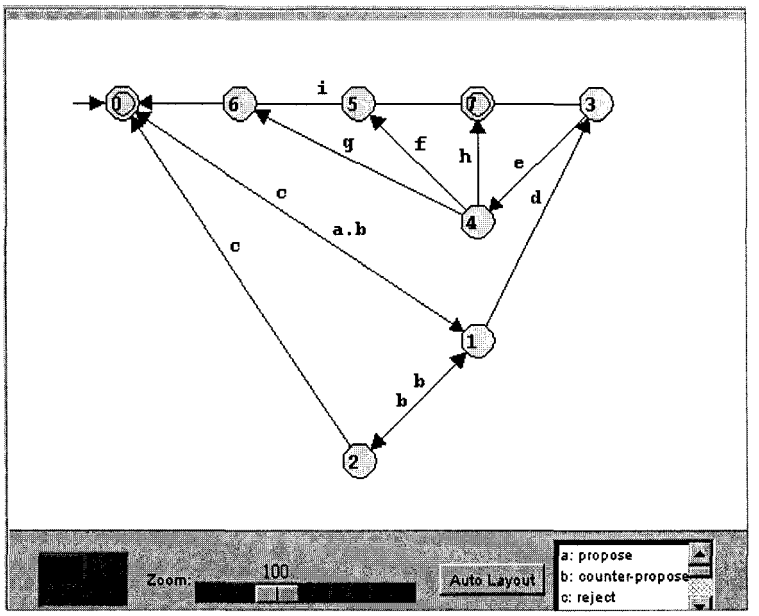


Figure 32: Modélisation de la machine à états finis qui dirige les règles de transitions

- **Transitions FSM**

La figure ci-dessous représente la version modélisée de la machine à états finis qui dirige les règles de transitions (voir figure 32). Nous pouvons ainsi voir les différents actes de discours associés à chaque arc, en associant chaque symbole avec sa définition. On a décidé d'utiliser des symboles pour représenter les différents actes de discours pour ne pas encombrer notre machine à états finis. Par exemple, comme on peut voir au niveau de la figure 32 et plus précisément au niveau de la figure 33, le symbole 'a' représente l'acte « propose », entre l'état 0 et l'état 1, le symbole 'a.b' représente le fait qu'il y a deux transitions possibles, ce qui fait que pour passer de l'état 0 à l'état 1, les actes « propose » et « counter-propose » sont permis).

Figure 33: Symboles représentant les actes de discours au niveau des arcs (FSM)

- Options visualisations de la machine à états finis

ProtocolBuilder nous permet de visualiser notre machine en permettant à l'usager de régler sa taille selon ses convenances. Cette fonctionnalité peut être utile lorsque la machine du protocole est trop (ou bien pas assez) volumineux, elle peut être comparée a l'option d'agrandissement ou de réduction des images ("Zoom") au niveau de plusieurs éditeurs tels que (Paint, Photoshop sur l'environnement Windows). La figure 34 présente deux interfaces qui représentent la même machine à états finis, ce qui les différencie est la taille des machines au niveau de chaque interface:

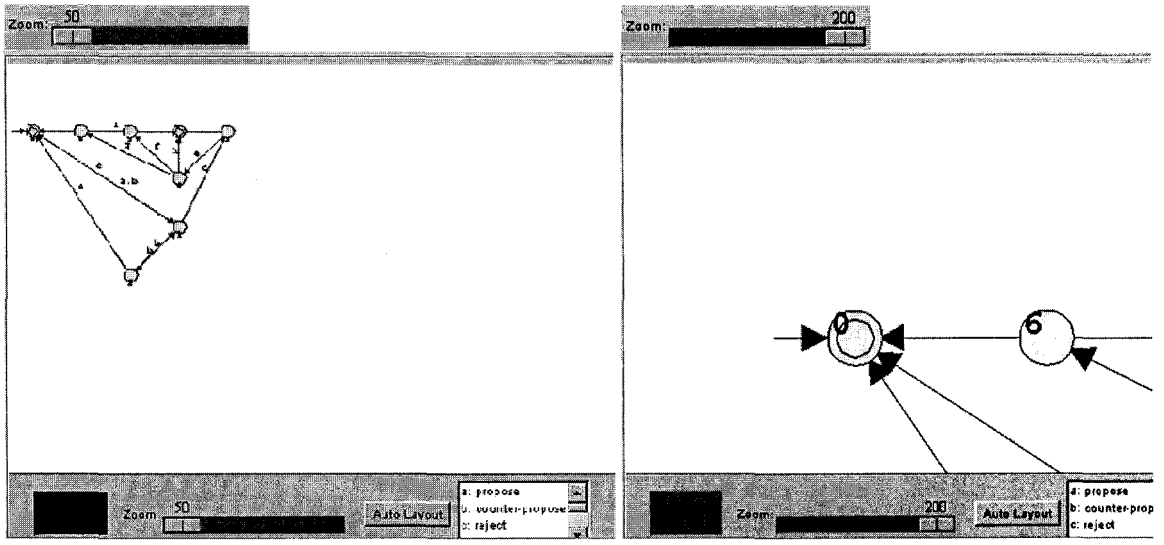


Figure 34: Interfaces montrant les différentes tailles possibles d'une machine

- ***POSITIONNEMENT FSM***

De plus, on peut également modifier notre champ de vision en faisant défiler notre machine. La figure 35 représente une interface qui a subi un tel "défilement". Au niveau de cette figure on ne voit qu'une partie de notre machine car on a positionné notre machine de cette manière (si on en a envie, on peut facilement changer la position de la machine comme bon nous semble):

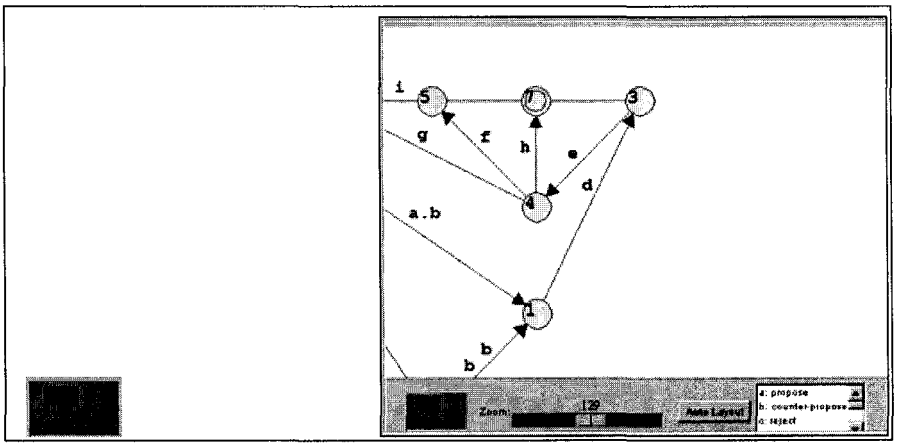


Figure 35:Visualisation : positionnement d'une machine

Après avoir créé notre machine, on peut considérer que l'étape de création de protocole est terminée. Une fois que l'étape de création de protocole est terminée, des fichiers représentant le protocole créé sont générés :

- un fichier représentant les éléments du protocole crée :
  - nom du protocole,
  - description du protocole,
  - actes de discours (performatifs)
  - description des performatifs,
  - paramètres associés aux performatifs
  - description des paramètres associés aux performatifs.

- Un autre fichier représentant la machine à états finis créée pour modéliser le protocole crée.

Les fichiers générés vont être ajoutés à la bibliothèque de protocoles créés. L'existence de cette bibliothèque permet la réutilisation des protocoles au niveau de la vérification de la conformité des conversations par rapport aux protocoles.

4.3.2.8 Fichiers générés

À la fin de la création d'un protocole par le logiciel ***protocolBuilder,*** trois sorties sont générées :

- Une sortie sous forme de fichier texte (figure 36), pour donner les spécificités de la machine à états finis créée dans un fichier portant le nom du protocole avec comme extension « .machine », précisant tous les détails nécessaires pour recréer notre machine avec une autre application. On peut remarquer les les performatifs sont représentés par des symboles de l'alphabet tels que les lettres 'a', 'b', 'c', ...

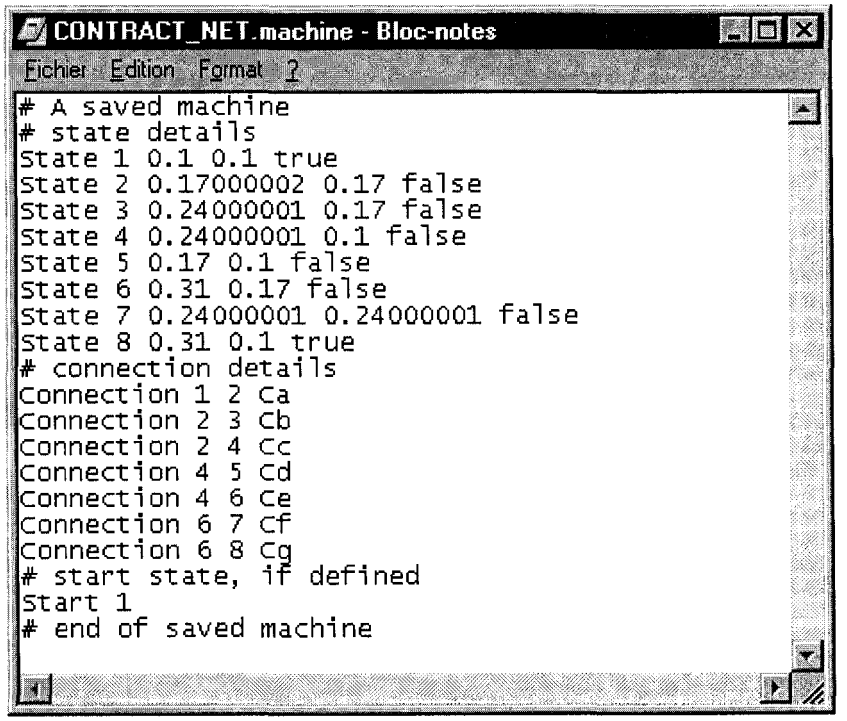


Figure 36: Exemple de machine à états finis sauvegardée

- Une sortie sous forme de fichier texte (figure 37), pour donner les spécificités des symboles utilisés pour représenter les différents actes de discours au niveau de la modélisation des machines à états finis. Par exemple, on peut voir que le symbole ***'b'*** représente le performatif ***« refuse »***.Les fichiers portent l'extension: ".descPerf" (pour "description performatif")

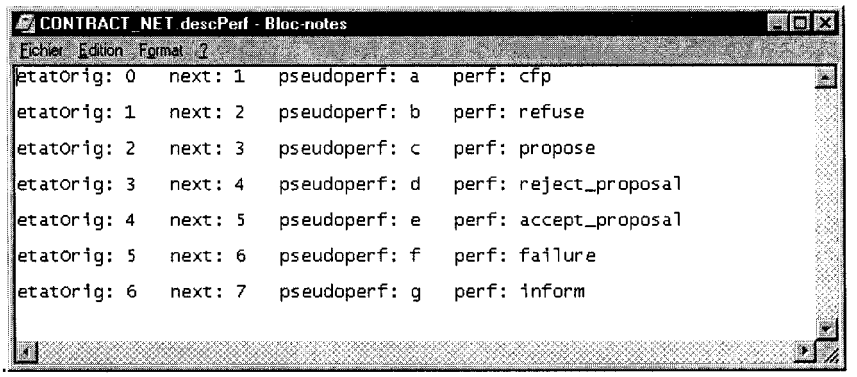


Figure 37: Fichier permettant de retenir les actes de discours suivi de leurs symboles

- Une autre sortie sous forme de fichier texte également (figure 38), pour donner les spécificités du protocole créé, on note les actes de discours et leurs paramètres associés ainsi que leurs descriptions. Les fichiers portent l'extension : « .prot » (pour "protocole").

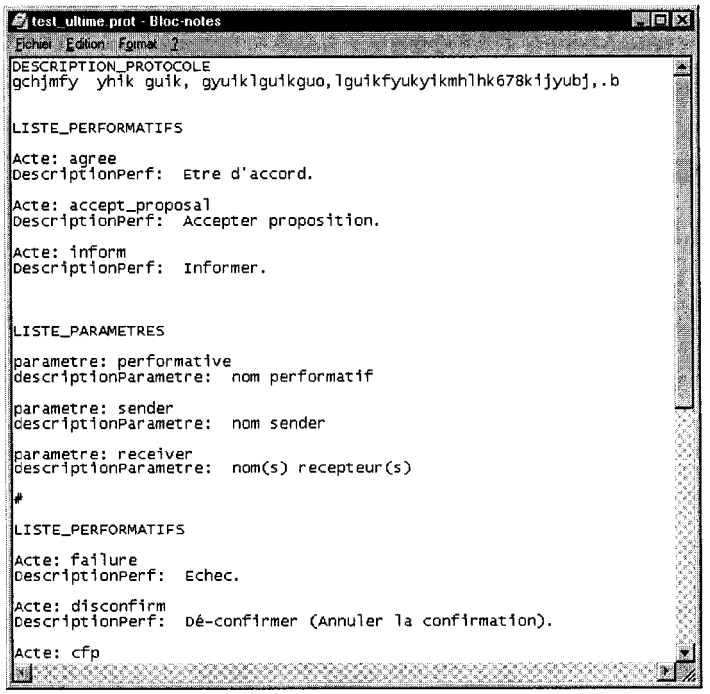


Figure 38: Exemple d'éléments de protocole sauvegardé « test_ultime.prot »

4.3.3 Vérification/validation des protocoles crées

À partir de la section «Vérification conversation protocole » de notre environnement, il est possible de sélectionner une conversation existante. Chaque conversation indique au départ quel protocole est suivi. Ensuite connaissant le protocole suivi, on peut d'une part parcourir les propriétés du protocole utilisé. On peut ainsi obtenir la description protocole,la liste des performatifs, la description de chaque performatif, la liste des paramètres de chaque performatif. D'autre part, on peut passer à la vérification de la conformité des conversations par rapport aux protocoles. La figure 40 représente l'interface permettant de visualiser les différentes propriétés d'un protocole choisi.

4.3.3.1 Interface permettant de visualiser les propriétés des protocoles

La figure 39 représente une interface permettant de visualiser les différentes propriétés des protocoles. Au niveau de cette section, nous allons voir en détail les différents paramètres qui permettent de former le protocole.

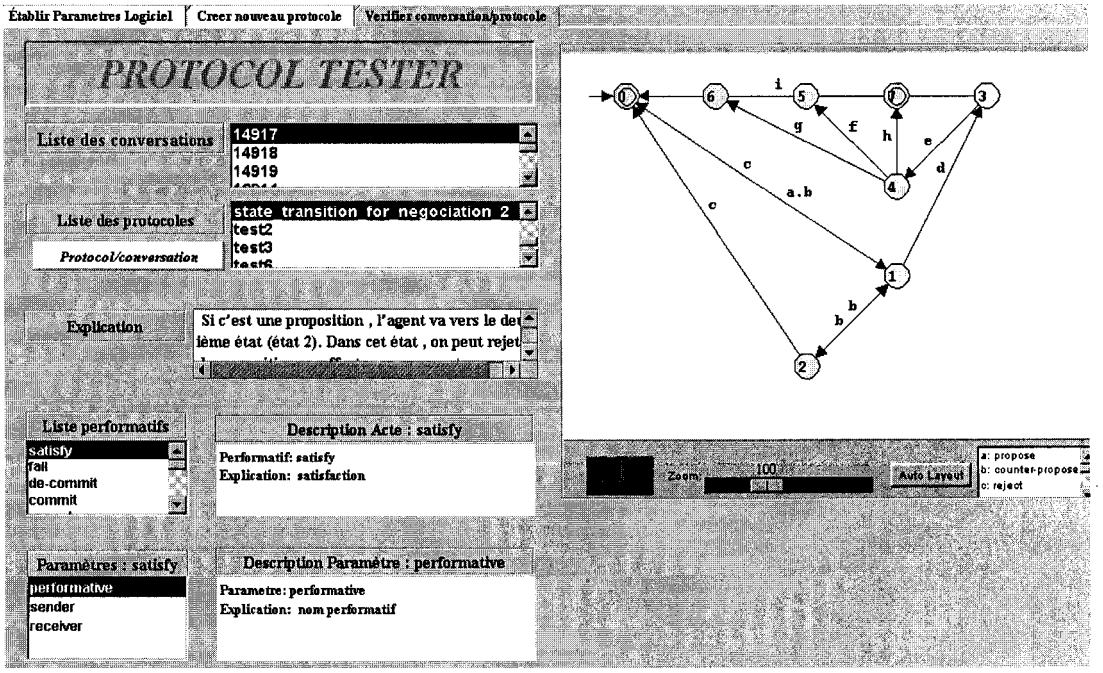


Figure 39: Interface globale permettant de visualiser les propriétés des protocoles

- Au niveau de l'interface permettant de parcourir les propriétés du protocole utilisé, on peut voir les explications de ce protocole au niveau de la figure 40:

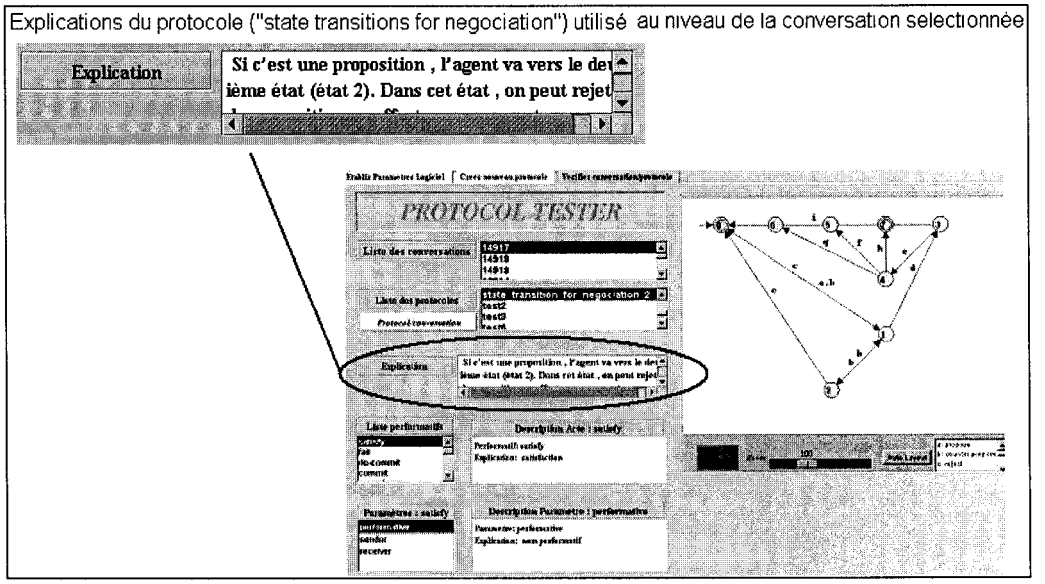


Figure 40: Espace réservé pour l'explication d'un protocole créé

- Toujours au niveau de l'interface permettant de parcourir les propriétés du protocole utilisé, la liste des actes de discours (suivi des descriptions de chaque acte) et les différents paramètres associés à chaque acte (suivi des descriptions de chaque paramètre) au niveau de la figure 41:

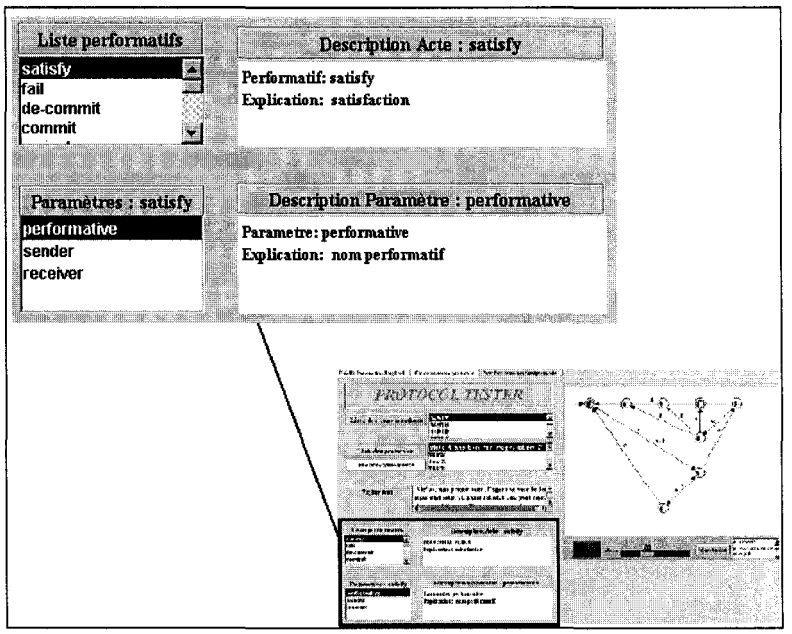


Figure 41: Visualisation actes/paramètres stockés d'un protocole créé

4.3.3.2 Interface permettant la vérification de la conformité

Au niveau de l'interface permettant d'effectuer les différentes vérifications de conformité des conversations, on peut voir plusieurs vérifications différentes. La Possibilité de vérifier la compatibilité des messages (séquences) en fonction du protocole utilisé au niveau de la conversation sélectionnée par rapport aux actes de la conversation et ceux existant au sein du protocole et par rapport à la complétude des paramètres des actes reconnus.

Représentation des conversations (inter-agents) sélectionnée.

Tout d'abord, après avoir sélectionné une conversation, non seulement le protocole utilisé est reconnu, mais le contenu de la conversation est représenté au niveau de notre environnement. Une conversation est composée d'un nombre fini de messages, on représente ces différents messages sous forme de séquences au niveau de notre environnement comme on peut le voir au niveau de la figure 42.Une partie de la conversation divisée en nombre fini de séquences au niveau de la figure 42 peut être retrouvée au niveau de la figure 43. Ainsi afin de démarrer la vérification d'une conversation, l'usager doit tout d'abord sélectionner une conversation (p.e conversation

au niveau de la figure 43). Après avoir sélectionné la conversation voulue, chaque fichier représentant une conversation identifie le protocole suivi. Dans l'exemple de la figure 43, on peut voir que le protocole suivi est le protocole *"state_transition_for_negociation_2"* :

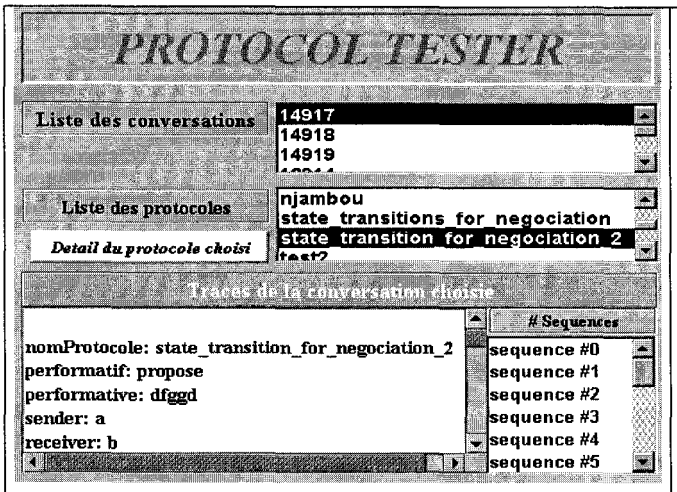


Figure 42: Sélection d'une conversation existante

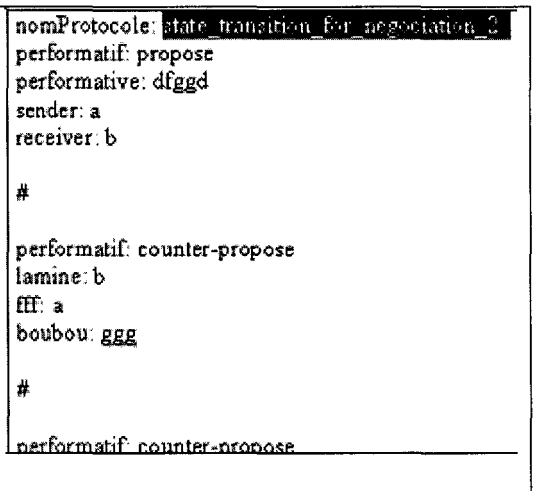


Figure 43: conversation sélectionnée

Vérification compatibilité performatifs/paramètres avec protocole choisi

Au niveau de notre environnement, la conversation sélectionnée est analysée. La liste des messages est stockée sous forme de séquences. Chaque séquence nous permet de retracer les différents actes de discours et leurs paramètres au niveau de chaque message de notre conversation. À partir de la conversation sélectionnée, le protocole utilisé est reconnu, ce qui permet de Visualiser la machine à états finis modélisant les transitions possibles. Une analyse de chaque acte (performatif) et des paramètres respectifs est effectuée.

D'une part, on effectue la vérification de l'appartenance de chaque performatif au sein de la conversation sélectionnée par rapport au protocole qu'elle est supposée suivre. De plus, on vérifie que les paramètres associés à chaque performatif sont complets (qu'il n'y a pas de paramètres en trop, ou manquants par rapport à la spécification des paramètres de l'acte de discours au niveau du protocole).La figure 44 montre non seulement les actes de discours reconnus et non-reconnus (par rapport au protocole), mais montre également les paramètres des actes reconnus, sachant qu'au niveau des paramètres

d'un acte de discours, il se peut qu'il y ait un (ou plusieurs) paramètre(s) qui manque(nt) ou qu'il y ait un (ou plusieurs) paramètre(s) en trop par rapport à la spécification au niveau du protocole utilisé.

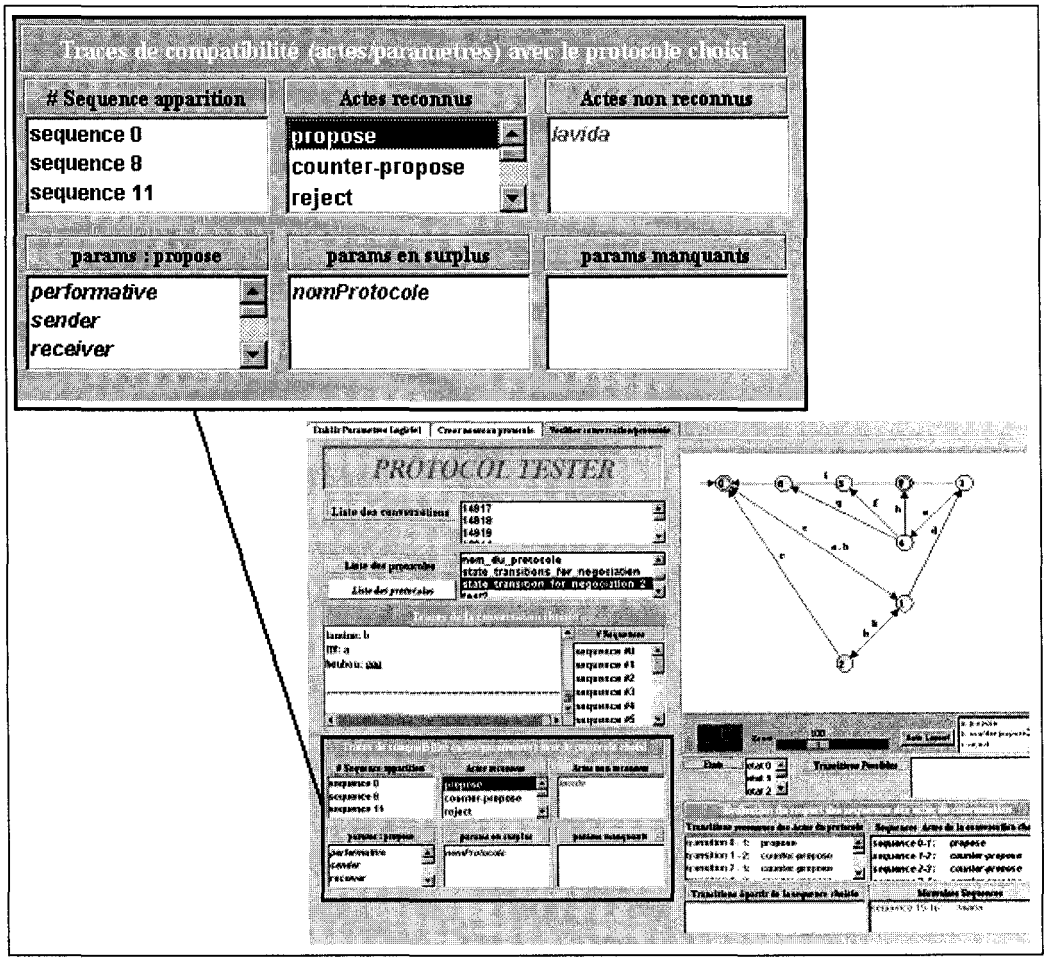


Figure 44: Traces de compatibilité (actes/paramètres)

Vérification du bon suivi du respect de l'ordre permis des performatifs

L'ordre des séquences (représentés par les différents actes de discours) est dicté par la machine à états finis du protocole utilisé. Une analyse du respect des séquences des échanges de messages par rapport aux séquences permises est effectuée. Au niveau de notre machine, une structure de notre environnement nous permet de connaître la liste des transitions possibles en sélectionnant l'état d'origine. De même, à partir d'une séquence de la conversation (« *Séquence Actes de la conversation choisie* »), on peut voir les différentes transitions possibles, selon notre machine. Ainsi, comme le montre la figure 45, on arrive à distinguer toutes les séquences qui respectent l'ordre établi de celles qui ne suivent pas l'ordre spécifié.

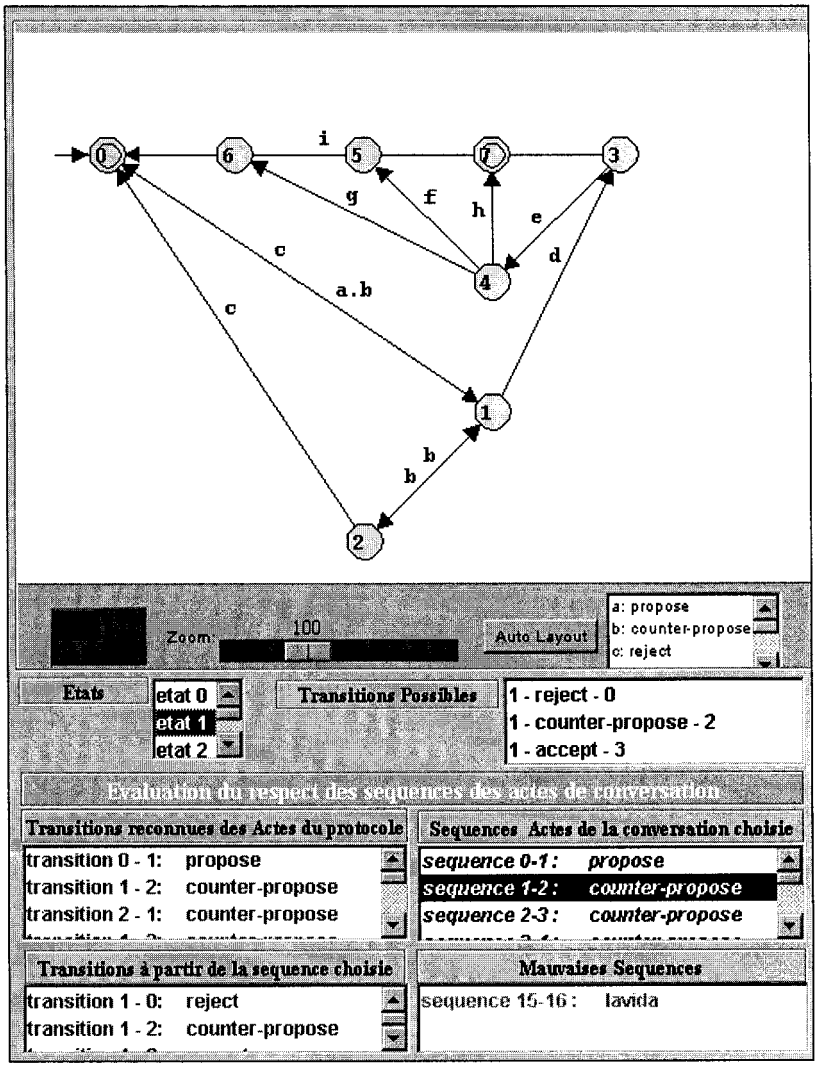


Figure 45: ProtocolBuilder : Analyse respect des séquences / FSM (protocole)

4.3.4 Évaluation

Au niveau de l'évaluation, on a décidé d'utiliser un outil très utilisé au niveau des systèmes multiagents, notamment l'outil Jack [34]. Cet outil va nous permettre de simuler des conversations inter-agents pour ensuite les intégrer avec ProtocolBuilder. On va tout d'abord effectuer une présentation de l'outil Jack [34]. On va ensuite présenter un protocole de communication largement utilisé au niveau du commerce électronique appelé "ContractNet" et effectuer une simulation de conversations inter-agent utilisant ce protocole par l'intermédiaire de l'outil Jack. Ainsi, après avoir utilisé l'outil Jack pour simuler des conversations suivant le protocole ContractNet, on va montrer comment *ProtocolBuilder* reproduit ce protocole de communication et effectue la vérification des différentes conversations (simulées à travers l'outil Jack).

4.3.4.1 Outils de simulation : «JACK INTELLIGENT AGENTS »

JACK™, est un environnement pour concevoir et intégrer des systèmes multiagent de catégorie commerciale, en utilisant une approche basée sur les composants (component-based).

Une vue d'ensemble de l'outil Jack peut-être vue au niveau de la figure 46. JACK™ est basé sur le travail de recherches et de développement des technologies d'agent de logiciel. Le langage de type « agent » JACK est un langage de programmation qui prolonge Java avec des concepts orientés agent, tels que :

- Agents
- Capabilities (possibilités)
- Events (evenements)
- Plans (plans)
- Agent Knowledge Bases (bases de données)
- Resource and Concurrency Management (***gestion de ressource et de simultanéité***)

Jack incorpore une nouvelle suite des outils graphiques visés aux analystes aussi bien qu'aux programmeurs :

- fournis un outil de conception qui permet à des composants d'être présentés au niveau des diagrammes qui représentent différents aspects de la conception d'une application. Ces diagrammes sont automatiquement convertis en code et contours de code. L'outil de conception emploie la facilité « drag and drop » pour manoeuvrer des objets, et fournit des possibilités de manipulations tel que le zoom.

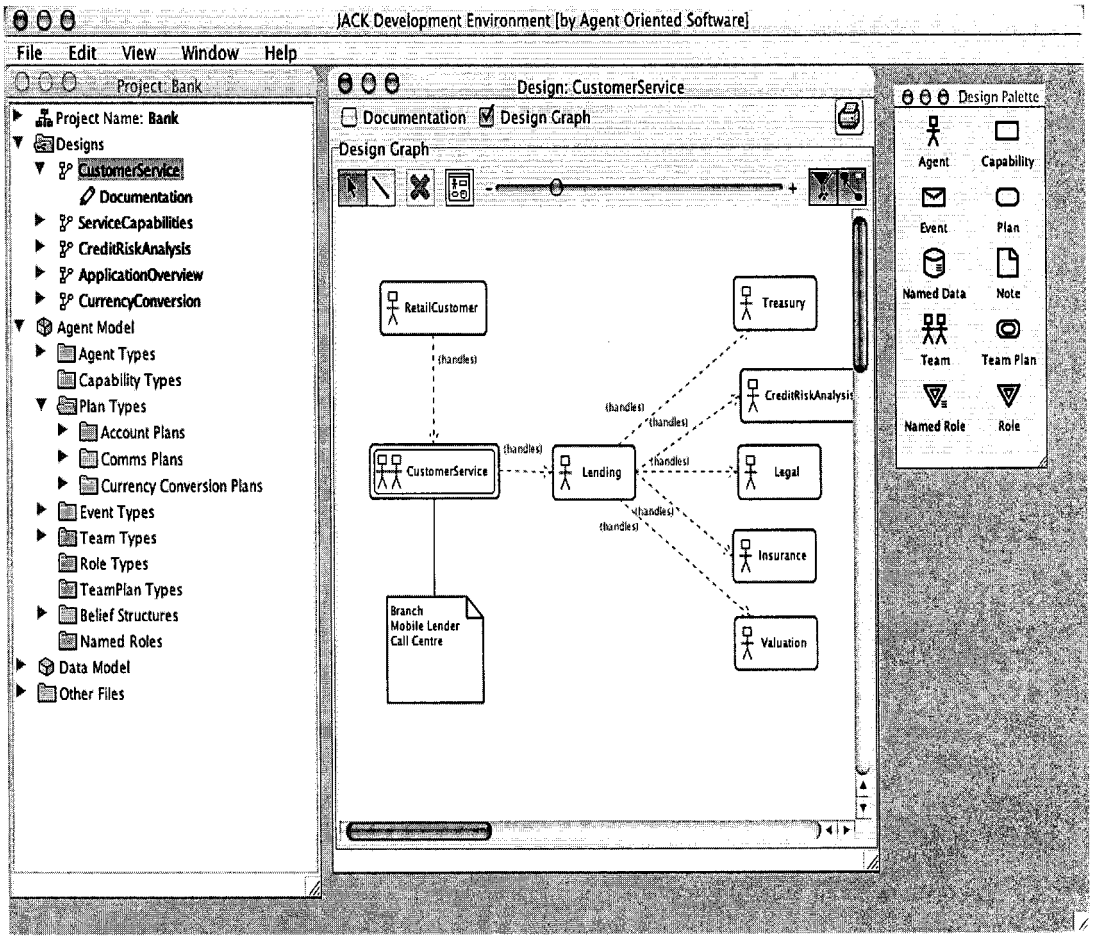


Figure 46: Logiciel Jack , vue d'ensemble

- fournis un outil d'édition de plan (figure 47) permet au raisonnement de plan (*plan reasoning*) d'être présenté en tant que diagramme simple sans devoir sacrifier la puissance du langage fondamentale de JACK/Java. Le raisonnement de plan peut être écrit dans le mode descriptif, permettant aux non-programmeurs de décrire le raisonnement par l'intermédiaire du langage naturel.

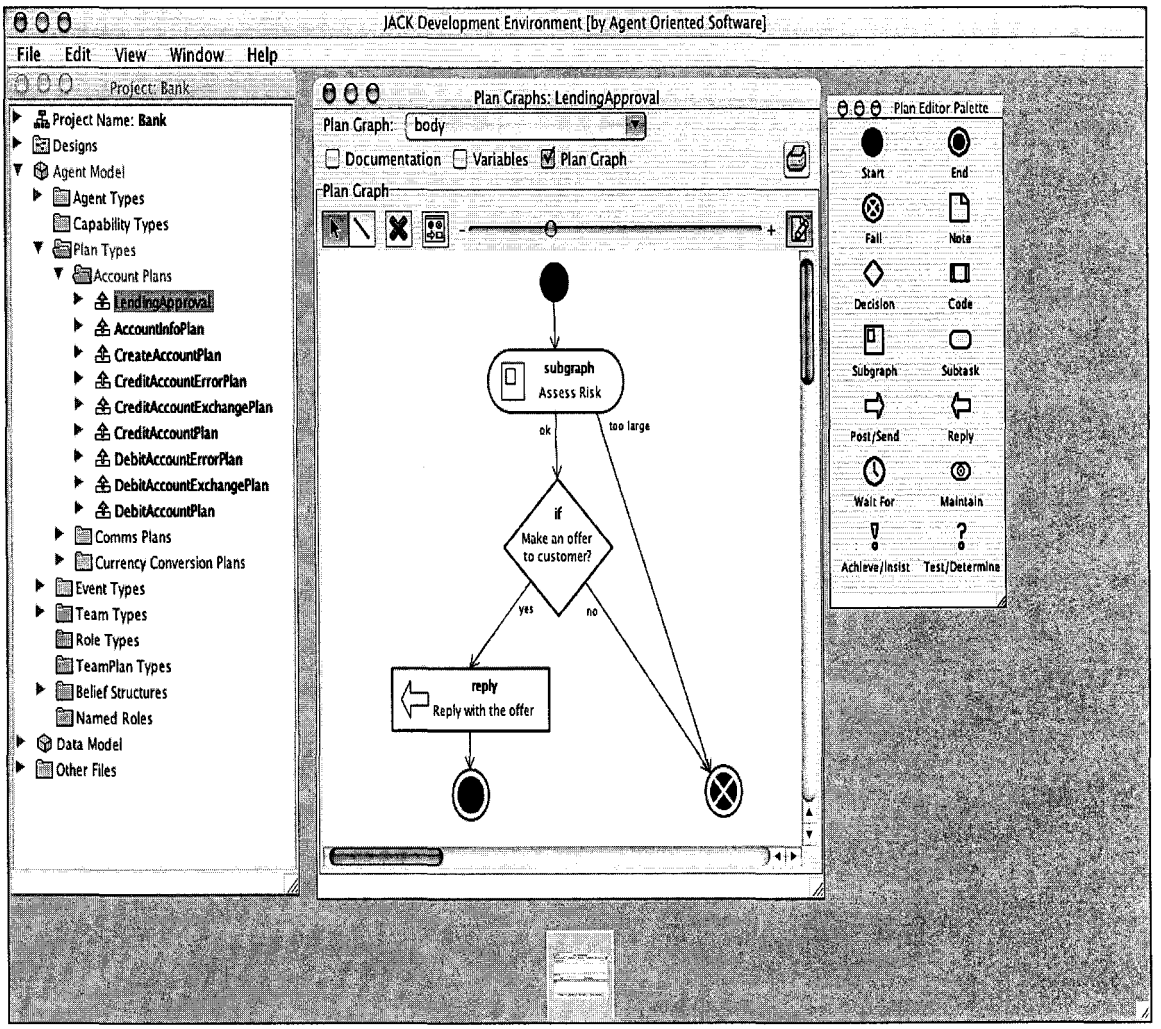


Figure 47: Logiciel Jack , édition de plans

- Au niveau de l'exécution, ces mêmes plans peuvent être tracés graphiquement permettant ainsi à un analyste ou à un programmeur d'observer comment ces plans s'exécutent. Les valeurs des variables utilisées dans ces plans peuvent également être examinées pendant l'exécution, avec l'historique d'exécution de plan (figure 48).

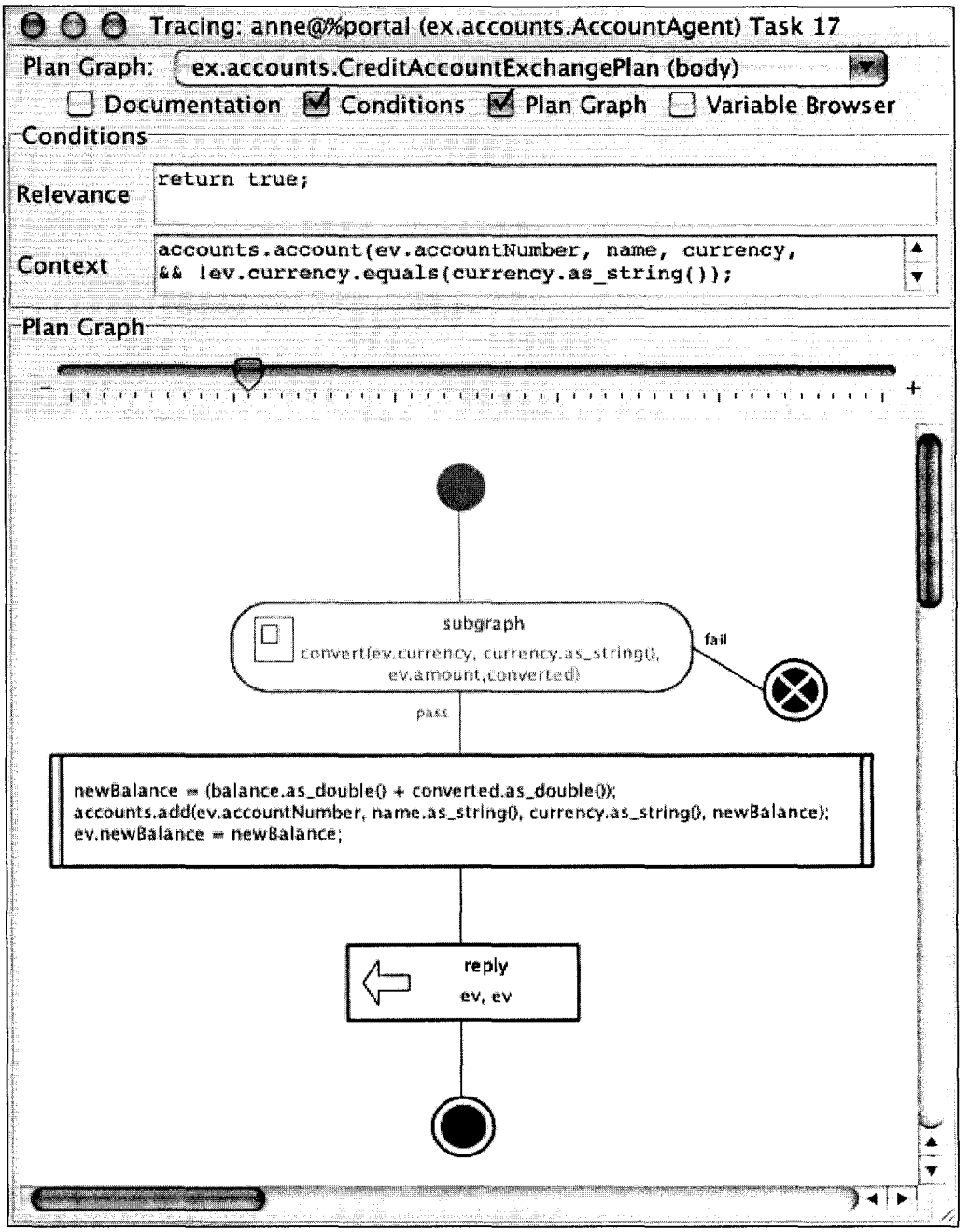


Figure 48: historique d'exécution de plans

Jack (entièrement codé dans le langage ***Java***) présente plusieurs avantages:

- **Flexibilité** : Jack tient compte de plusieurs de types d'agent logiciel, des agents simples (par exemple agents de recherche documentaire) à travers à des agents plus capables en utilisant des modèles sophistiqués de raisonnement

- **Extensibilité** : Jack emploie une philosophie "composante" pour faciliter l'intégration avec d'autres environnements de logiciel ou systèmes existants. S'il y a lieu, la fonctionnalité de JACK's peut être prolongée par l'utilisation des plugins faits sur commande.

- **Familiarité** : Jack est basé sur un ensemble de prolongements simples au langage Java, concernant des concepts orientés agents. La connaissance de JACK aux programmeurs Java réduit non seulement la courbe d'étude mais, comme avec des aides types sûres et orientées objet de Java, de l'approche JACK dans le développement des applications fiables.

- **Portabilité :** Jack est capable de fonctionner sur n'importe quel système sur lequel Java est fourni.

- **Légereté** : Jack est extrêmement léger (**Light-Weight**) et est conçu pour manipuler des milliers d'agents fonctionnant sur relativement le matériel de basse extrémité.

- **Orienté Java (Java-Based)** : Jack permet l'accès à toutes les possibilités présentes et futures de Java, y compris les voies multiples concurrentes (fonctionnant probablement sur des CPUs multiples), à plateformes indépendantes et à bibliothèques de troisième ordre telles que JDBC.

- Interopérabilité : Jack permet l'intégration facile avec des « packages » externes en utilisant l'infrastructure standard, telle que CORBA, RMI, J2EE, EJB, NET, DCOM ou HLA

De plus :

- Les agents de Jack peuvent être organisés en équipes pour modéliser des buts ou pour accomplir des tâches communes réelles.

- JACK a des spécifications de langue et une conception orientée objet visée pour permettre la prolongation facile pour de nouveaux modèles d'agent, tel que l'identification des intentions, ou agents basés « transaction » (transaction-based agents).

- JACK fournit un modèle de communication flexible et « léger ». Par défaut Jack emploie un protocole rapide basé TCP/IP pour communiquer avec les autres agents, systèmes principaux et GUIs. D'autres protocoles peuvent être posés sur ceci ou peuvent remplacer le protocole par défaut entièrement. Ceci peut être employé pour fournir une communication sécuritaire entre les agents.

4.3.4.2 Simulation du protocole CONTRACTNET

Après avoir présenté l'outil Jack, on va tenter d'utiliser cet outil pour effectuer la simulation de conversations entre agents sensée suivre le protocole de communication appliqué au niveau du commerce électronique appelé ContractNet. Nous allons tout d'abord effectuer la présentation du protocole ContractNet (représentée au niveau de la figure 49), on va ensuite simuler des conversations entre agents en suivant le protocole ContractNet. Finalement, nous allons utiliser notre environnement ProtocolBuilder pour reproduire ce protocole et tester/vérifier les conversations issues de la simulation.

Présentation du protocole ContractNet

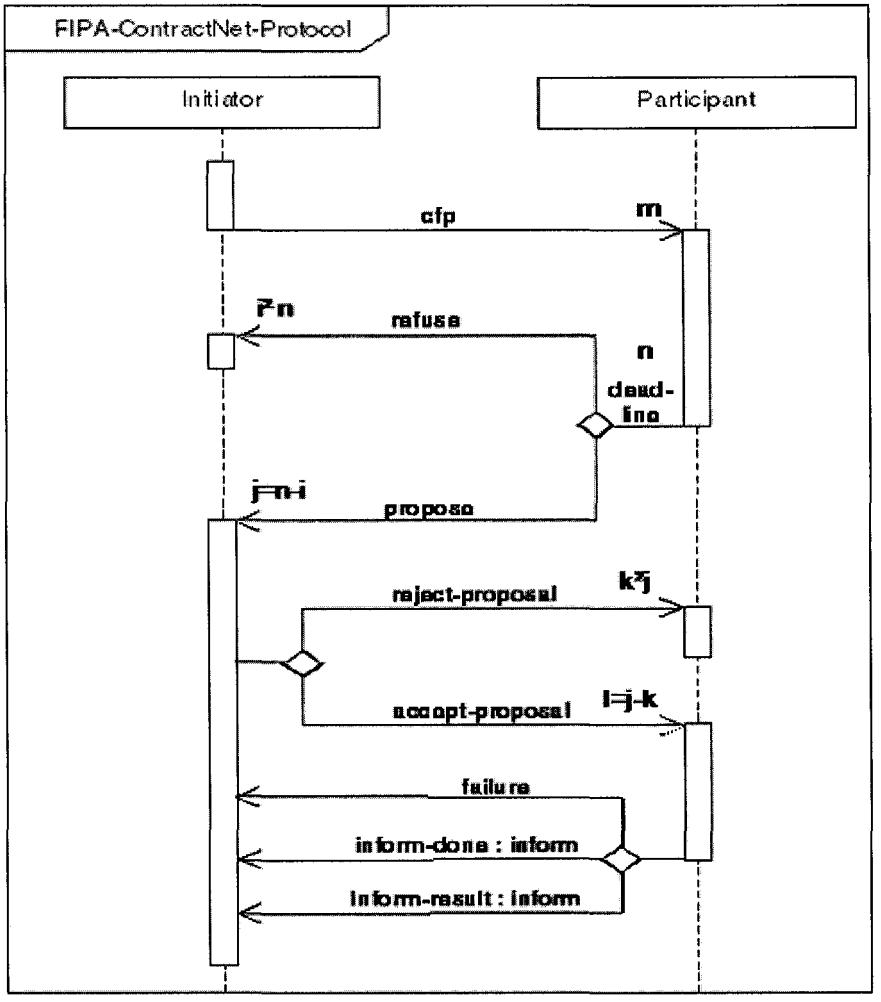


Figure 49: protocole FIPA-ContractNet

L'initiateur sollicite des propositions d'autres agents (participant) en publiant un « acte d'appel aux propositions » (*call for proposals* [FIPA00037]) qui indique la tâche et toutes les conditions que l'initiateur place sur l'exécution de la tâche. Les agents qui reçoivent l'appel aux propositions (*call for proposals* [FIPA00037]) sont considérés comme des entrepreneurs potentiels pouvant générer des propositions (*propose* acts [FIPA00037]) pour accomplir la tâche. La proposition de l'initiateur inclut des conditions préalables au niveau de la tâche, pouvant être le prix, moment où la tâche sera faite, etc... Alternativement, l'entrepreneur peut refuser (*refuse* [FIPA00037]) la proposition effectuée par un agent. Une fois que la date limite passe, l'initiateur évalue toutes les propositions reçues et choisit des agents pour accomplir la tâche (un, plusieurs ou aucun agent peuvent ne pas être choisis). Les agents sélectionnés recevront un acte d'acceptation (*accept-proposal* act [FIPA00037]) et les autres recevront un acte de refus (*reject-proposal* act [FIPA00037]). Les propositions se lient à l'entrepreneur (l'agent), de telle sorte qu'une fois que l'initiateur accepte la proposition, l'entrepreneur acquière un engagement pour accomplir la tâche. Une fois que l'entrepreneur (l'agent) fini d'accomplir la tâche, il envoie un message d'accomplissement a l'initiateur.

Simulation du protocole

Voici un exemple de système multiagent JACK (remerciements à Frédéric Asselin de l'université de LAVAL !). Il implémente le protocole « ContractNet » de FIPA en se basant sur la classe FipaMessage pour transmettre des messages FIPA-ACL. On peut ainsi voir les différents messages que les agents s'échangent suite au message de l'initiateur ("Call For Proposal") au niveau de la figure 50:

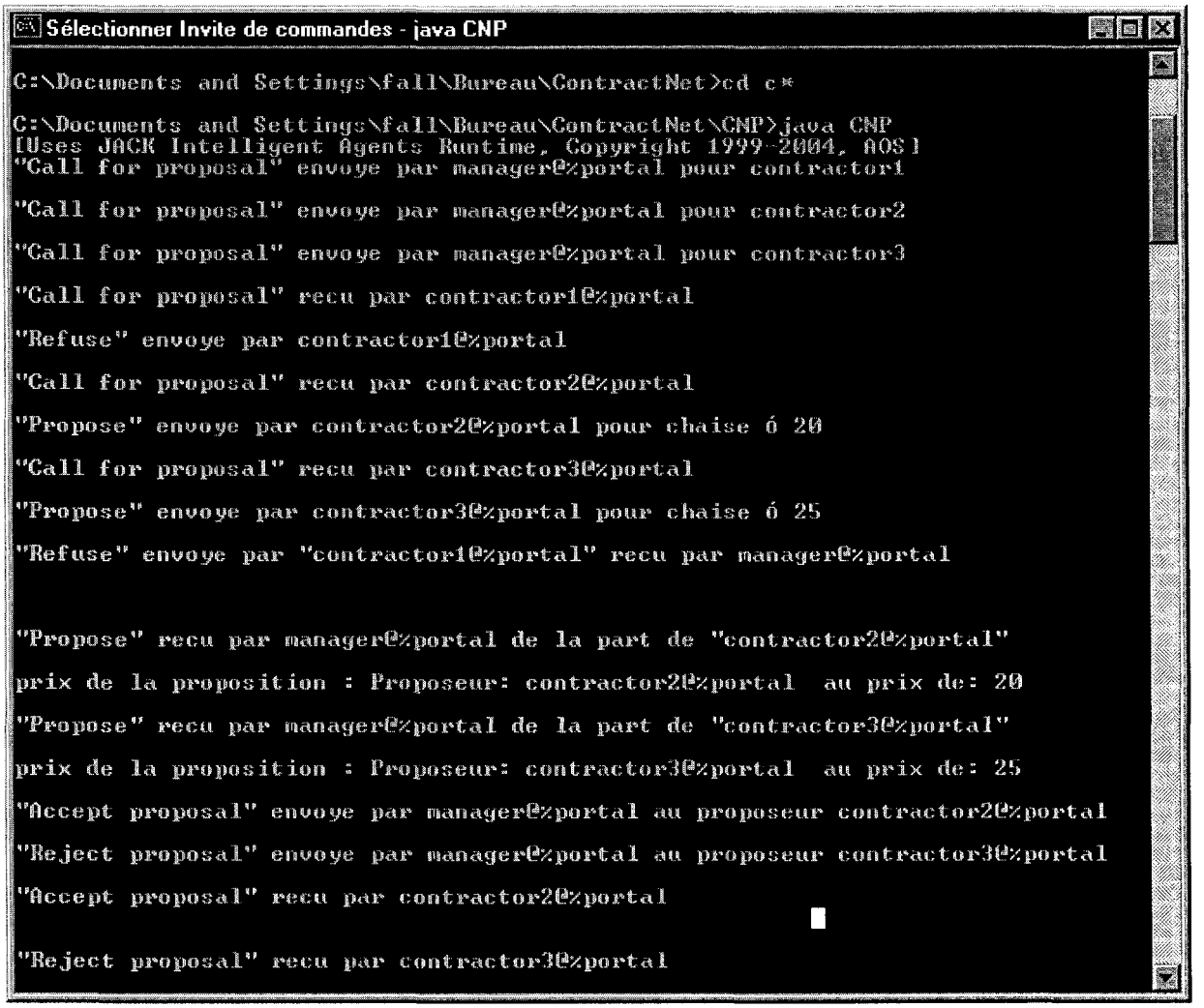


Figure 50: Exécution de la simulation du protocole CONTRACTNET avec "JACK"

Explication de la simulation du Protocole ContractNet effectuée

Nous allons dans un premier temps décomposer et ordonner les différents messages envoyés par les différents agents.

Manager envoie « call for proposal » aux agents contractor1, contractor2, contractor3 concernant la vente aux enchères d'une chaise.

- "Call for proposal" envoyé par manager@%portal pour contractor1
- "Call for proposal" envoyé par manager@%portal pour contractor2
- "Call for proposal" envoyé par manager@%portal pour contractor3

Le manager envoie un message aux différents agents (contractor1, contractor2, contractor3) indiquant qu'il va vendre une chaise.

réaction contractor1 face à l'envoi de manager

- "Call for proposal" reçu par contractor1@%portal
- "Refuse" envoyé par contractor1@%portal

Après avoir reçu le message du manageur ("Call for proposal"), le premier agent (contractor1) lui fait savoir que ça ne l'intéresse pas en lui renvoyant un message (acte de discours "refuse").

réaction contractor2 face à l'envoi de manager

- "Call for proposal" reçu par contractor2@%portal
- "Propose" envoyé par contractor2@%portal pour chaise à 20$

Après avoir reçu le message du manageur ("Call for proposal"), le deuxième agent (contractor2) lui fait savoir que la vente de cette chaise l'intéresse en effectuant une offre d'achat à 20$ (acte de discours "propose").

réaction contractor3

- "Call for proposal" reçu par contractor3@%portal
- "Propose" envoyé par contractor3@%portal pour chaise à 25$

Après avoir reçu le message du manageur ("Call for proposal"), le deuxième agent (contractor3) lui fait savoir que la vente de cette chaise l'intéresse en effectuant une offre d'achat à 25$ (acte de discours "propose").

Réception des réactions des différents agents suite à l'envoi du « call for proposal » par manager

- "Refuse" envoyé par "contractor1@%portal" reçu par manager@%portal
- "Propose" reçu par manager@%portal de la part de "contractor2@%portal", prix de la proposition : Proposeur: contractor2@%portal au prix de: 20$
"Propose" reçu par manager@%portal de la part de "contractor3@%portal", prix de la proposition : Proposeur: contractor3@%portal au prix de: 25$

À ce niveau, le proposeur reçoit toutes les réponses envoyées par les différents agents (l'agent contractor1 qui refuse, l'agent contractor2 qui effectue une offre de 20$ et l'agent contractor3 qui effectue une offre de 25$).

Envoi des réponses du manager

"Accept proposal" envoyé par manager@%portal au proposeur contractor3@%portal
"Reject proposal" envoyé par manager@%portal au proposeur contractor2@%portal

Après avoir reçu les différentes réponses des agents, le Proposeur envoi son verdict final: il décide de vendre sa chaise à l'agent contractor3 qui lui avait fait une offre de 25$ en lui envoyant un message d'acceptation ("Accept proposal"), il indique à l'agent contractor2 qu'il refuse son offre d'achat de 20$ en lui envoyant un refus ("Reject proposal").

Réception des messages envoyés par manager

"Accept proposal" reçu par contractor2@%portal
"Reject proposal" reçu par contractor3@%portal

Modélisation de la Simulation à travers ProtocolBuilder

Pour modéliser la simulation de conversation inter-agent, on doit "formater" les différentes conversations afin qu'elles puissent bien être lues et bien interprétées par ProtocolBuilder.

- **Représentation des messages envoyés**

Tout d'abord, on va assister à la représentation de tous les messages envoyés entre agents au niveau du protocole ContractNet (selon la simulation effectuée avec le logiciel Jack).

Manager (Initiator) envoie « call for proposal » aux agents contractor1, contractor2, contractor3 concernant la vente aux enchères d'une chaise

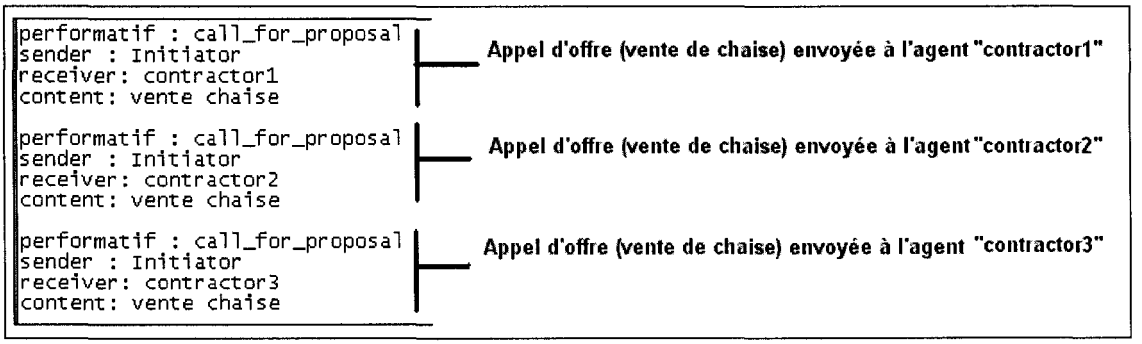


Figure 51: ContractNet, Messages envoyés par le manageur aux différents agents

Réaction des différents agents face à la réception du « call_for_proposal »

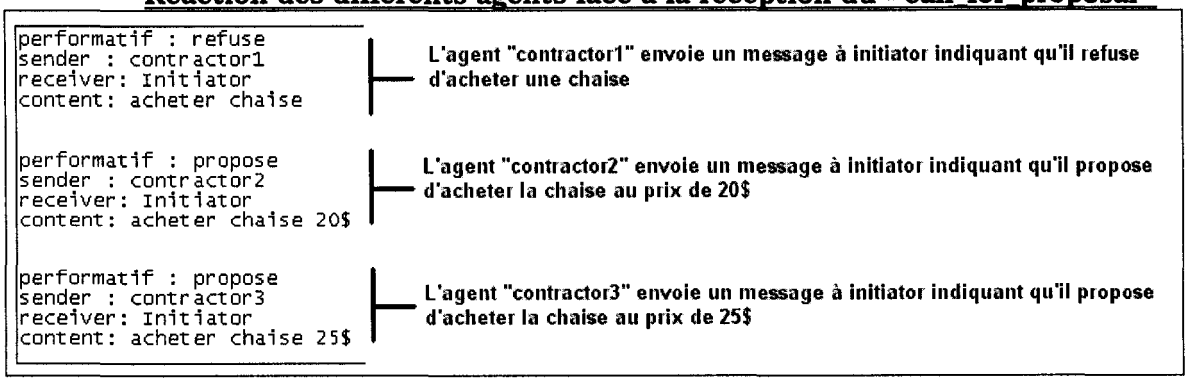


Figure 52: ContractNet, réactions des différents agents face à l'offre du proposeur

Verdict du manager (Initiator) face aux messages (réponses) envoyés par les différents agents :

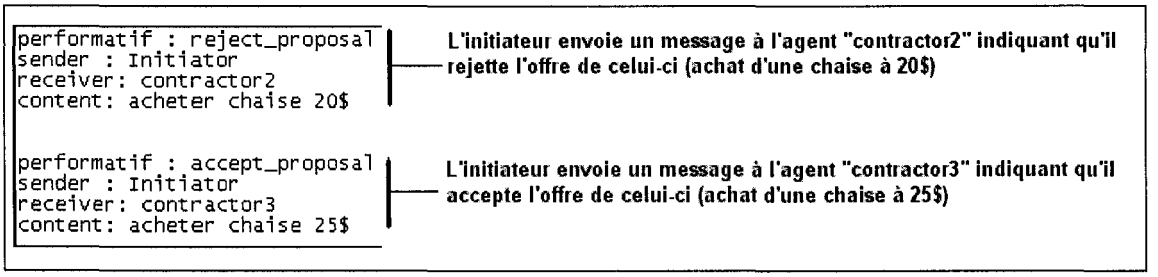


Figure 53: ContractNet, réactions du proposeur face aux différentes réponses

Les différents messages entre les agents peuvent être utilisés au niveau de ProtocolBuilder. Après avoir créé le protocole ContractNet au sein de notre environnement, on peut vérifier la conformité des différents messages.

- Représentation de la simulation du protocole ContractNet à travers ProtocolBuilder

Le premier fichier "achatChaise1" au niveau de la figure 54 représente l'ensemble des messages envoyés entre les agents proposeur (Initiator) et contractor1. Ce fichier contient deux messages : au niveau du premier message, le proposeur effectue un appel d'offres (cfp), au niveau du deuxième message, l'agent contractor1 indique qu'il n'est pas intéressé en envoyant un refus (refuse).

Le deuxième fichier "achatChaise2" au niveau de la figure 56 représente l'ensemble des messages envoyés entre les agents proposeur (Initiator) et contractor2. Cette conversation contient trois messages: tout d'abord le proposeur effectue son appel d'offres (vente de chaise), ensuite l'agent contractor2 témoigne son intérêt en faisant une offre de 20$ pour l'achat de la chaise, finalement le proposeur réplique en déclinant l'offre de l'agent contractor2.

Finalement au niveau de la figure 58 représente l'ensemble des messages envoyés entre les agents proposeur (Initiator) et contractor3. Cette conversation contient également trois messages : tout d'abord le proposeur effectue son appel d'offre (vente de chaise), ensuite l'agent contractor3 témoigne son intérêt en faisant un offre de 25$ pour l'achat de la chaise, finalement le proposeur réplique en acceptant l'offre de l'agent contractor3.

Les différentes figures sont présentées par paires de figures (figures 54-55, figures 56-57, figures 58-69, figures 60-61) logiquement reliées. Les paires de figures représentent d'un côté des exemples de conversations qui sont supposés suivre un protocole, et d'autre part la vérification/validation du bon suivi de ce protocole à travers ProtocolBuilder. Les figures 55, 57, 59 et 61 montrent de quelle manière notre environnement gère ces messages (respectivement des figures 54, 56, 58 et 60) par rapport au protocole qu'ils utilisent (ContractNet). Comme prévu, après avoir divisé notre fichier en plusieurs séquences au niveau de notre environnement, on effectue les vérifications de conformités des actes de discours et de leurs paramètres par rapport au protocole utilisé (CONTRACT_NET comme mentionné à la première ligne du fichier au niveau de la figure 54). De plus, notre environnement effectue également la vérification du respect de l'ordre permis des séquences (représenté par la machine à états finis au niveau des figures 55,57, 59 et 61).

Les paires de figures 54-55, 56-57, 58-59 représentent des scénarios qui marchent : les conversations représentées au niveau des figures 54, 56, 58 sont conformes au protocole qu'ils suivent (exactitude des actes de discours, complétude des paramètres, bon suivi de l'ordre permis des messages/séquences). La paire de figures 60-61 représente un scénario qui ne marche ni au niveau de l'exactitude des actes de discours, ni au niveau de la complétude des paramètres, ni au niveau du bon suivi de l'ordre permis des messages.

Les figures qui suivent 60, 61, 62 et 63 représentent des scénarios de conversations qui mettent en valeur la non-conformité des messages (actes de discours, paramètres associés) et du bon suivi de l'ordre permis des séquences (messages) par rapport des protocoles.

1. **Premier ensemble de figure (54 et 55)**

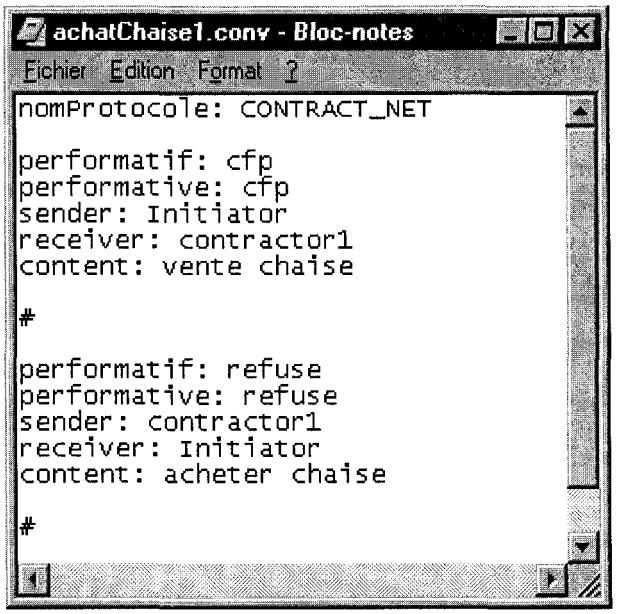


Figure 54: ContractNet, premier conversation entre agents

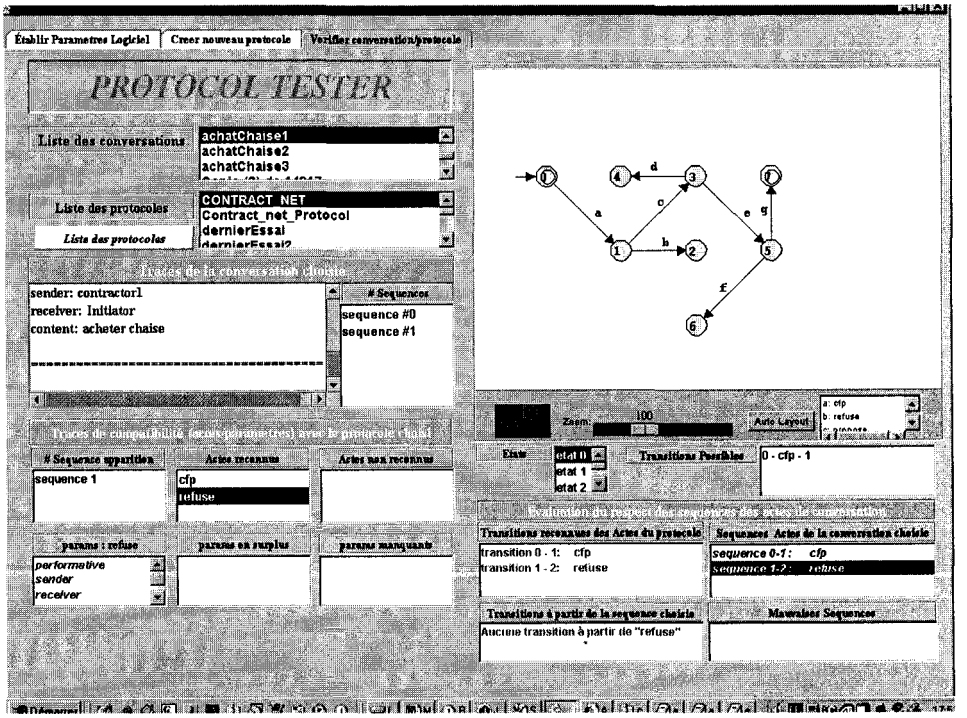


Figure 55: Interface protocolBuilder gérant la première conversation (figure 54)

2. **Deuxième ensemble de figures (figures 56 et 57)**

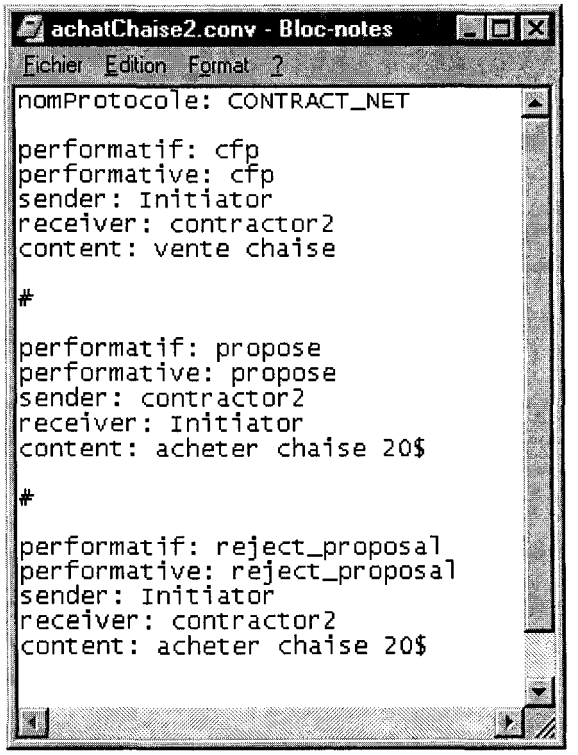


Figure 56: ContractNet, deuxième conversation entre agents

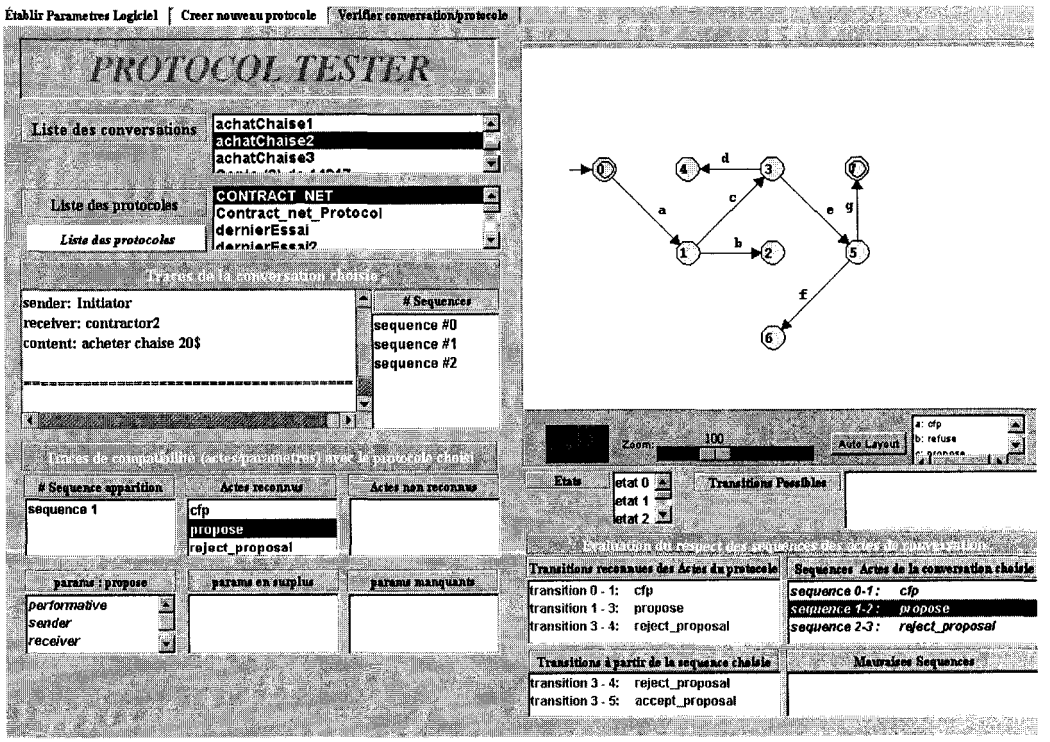


Figure 57: Interface protocolBuilder gérant la deuxième conversation (figure 56)

3. Troisième ensemble de figures (figures 58 et 59)

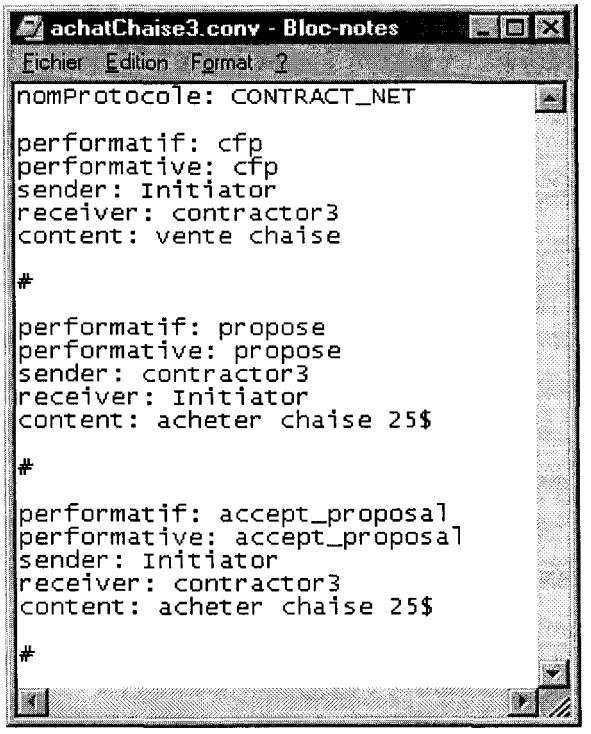


Figure 58: ContractNet, troisième conversation entre agents

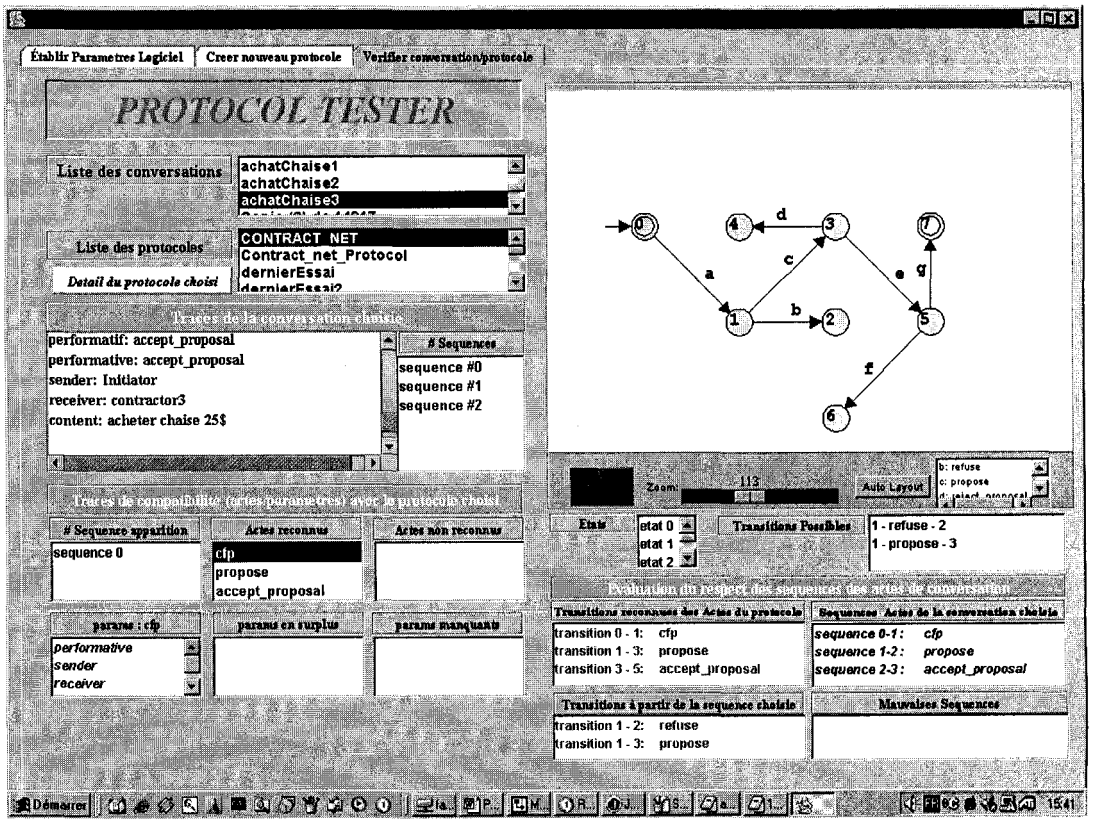


Figure 59: Interface protocolBuilder gérant la troisième conversation (figure 58)

4. Quatrième ensemble de figures (figures 60 et 61) : exemple "négatif" (conformité des messages)

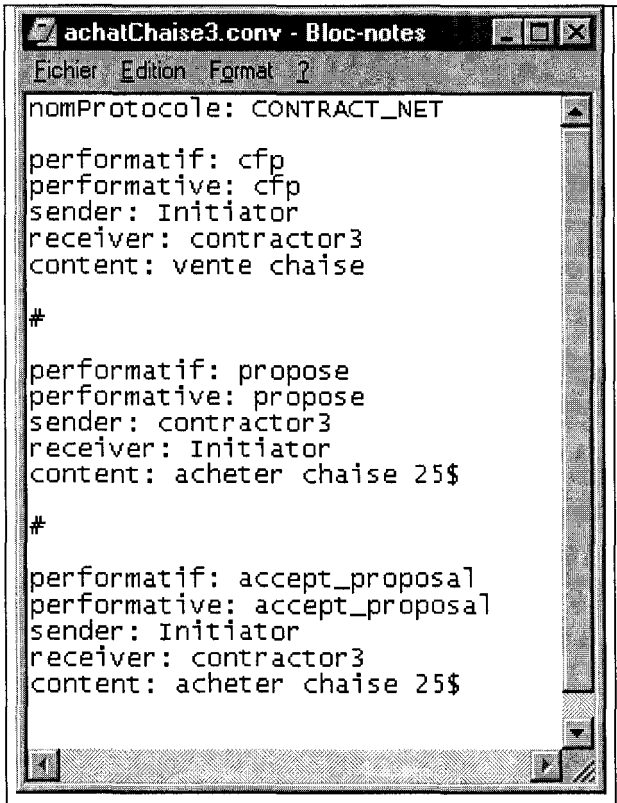


Figure 60: Exemple positif de conversation

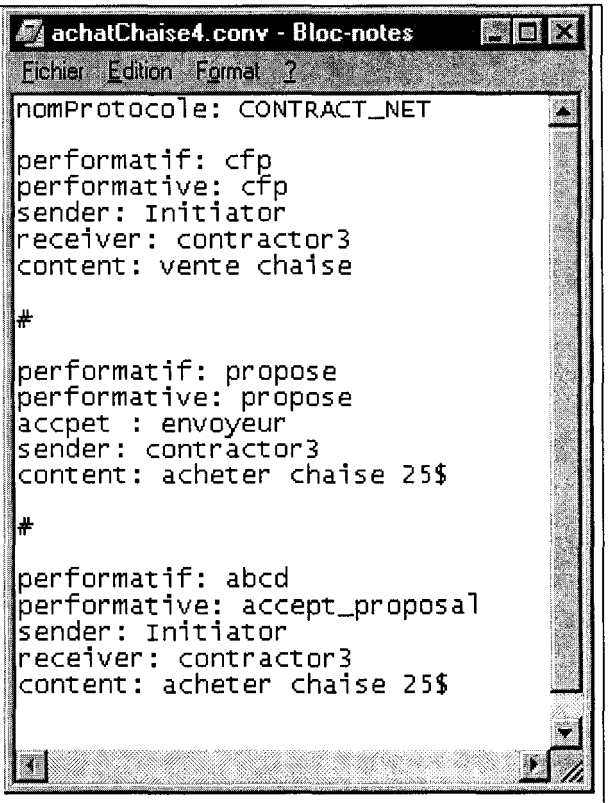


Figure 61: Exemple négatif de conversation

La figure 60 (représentation de la figure 58) représente une conversation utilisant le protocole ContractNet. Cette conversation est totalement conforme aux spécifications du protocole au niveau des différents actes de discours utilisés (cfp, propose, accept_proposal) et des différents paramètres associés à chaque acte.

La figure 61 représente la même conversation représentée au niveau de la figure 60 modifiée. Cette conversation n'est pas totalement conforme aux spécifications du protocole et cette non-conformité réside au niveau de la conformité des messages (actes de discours et paramètres):

- Tout d'abord au niveau de la deuxième séquence (dirigée par l'acte de discours "propose"), il y a le paramètre "receiver"qui manque

- Ensuite, toujours au niveau de la deuxième séquence, il y a le paramètre "accept" n'appartenant pas aux spécifications qui est "en trop".

- Finalement, les spécifications indiquent que l'acte de discours de la troisième séquence (message) devrait être l'acte "accept_proposal" comme précisé au niveau de la figure 60, or au niveau de la figure 61, l'acte de discours de la troisième séquence est "abcd".

Comme on peut voir au niveau de la figure 62, ProtocoleBuilder a pu non seulement détecter l'acte de discours "abcd" non conforme aux spécifications, mais a pu également détecter le paramètre manquant "receiver" et détecter le paramètre en trop "accept".

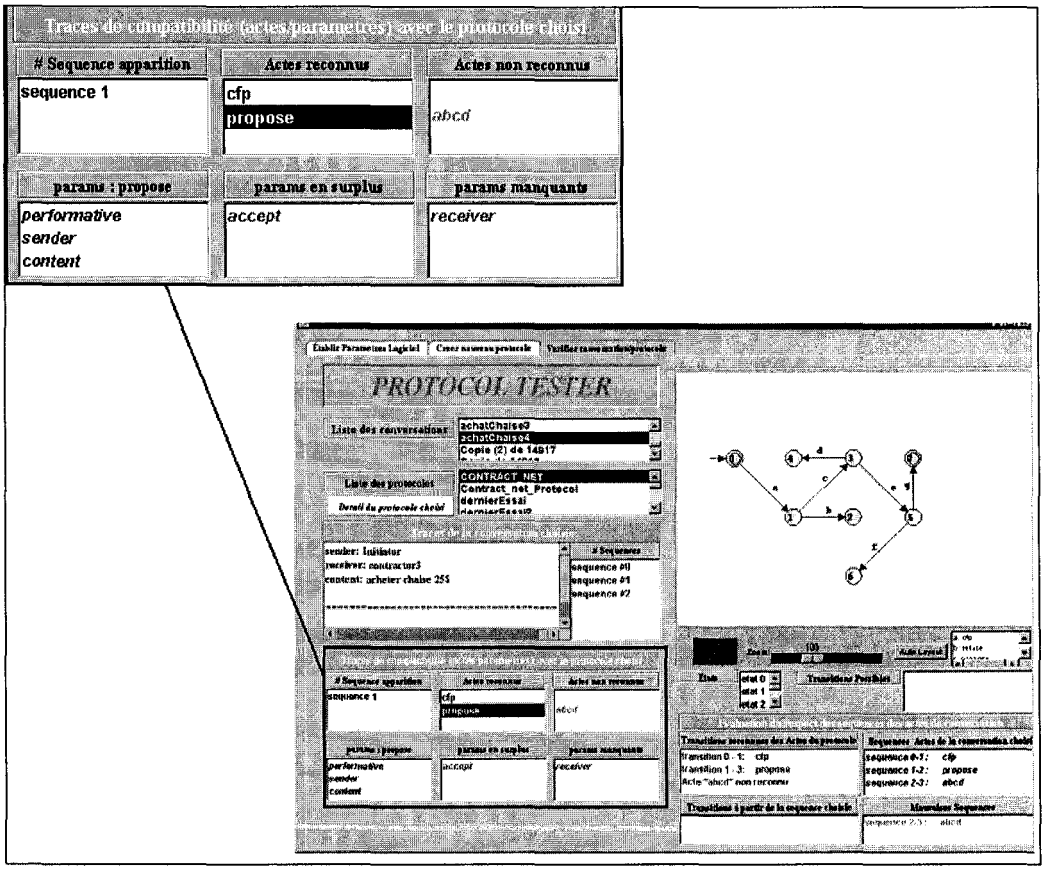
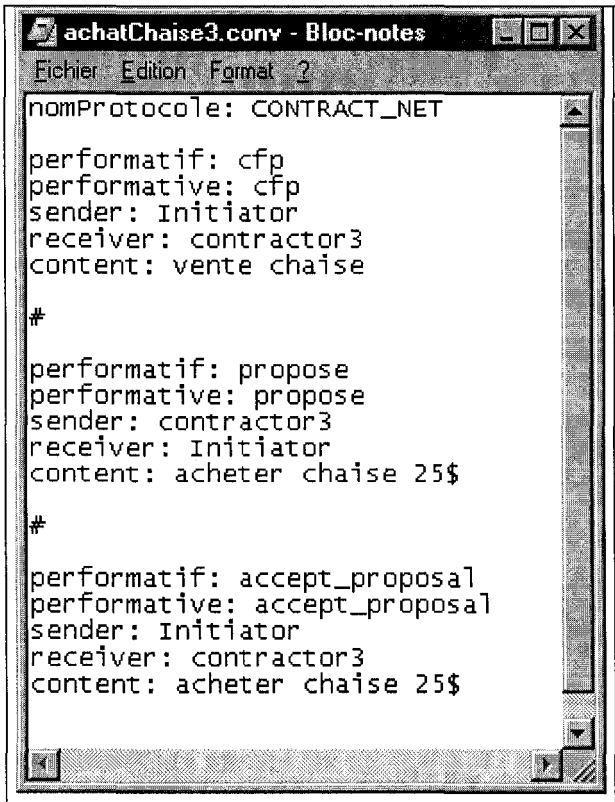


Figure 62: Interface ProtocolBuilder gérant la conversation de la figure 61

5. **Cinquième ensemble de figures (figures 61 et 62) : exemple "négatif" (conformité des messages)**



Exemple positif de conversation (Figure 60)

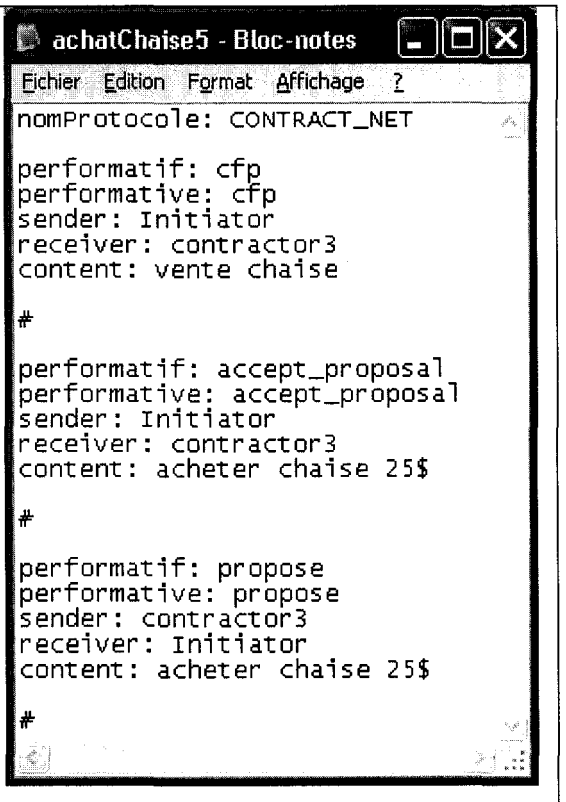


Figure 63: Exemple négatif de conversation

Au niveau du cinquième ensemble de figures, nous retrouvons la figure 60 (conversation définie précédemment) et la figure 63 qui représente la figure 60 modifiée. Comme indiqué précédemment, la conversation représentée au niveau de la figure 60 est totalement conforme aux spécifications (du protocole ContractNet) non seulement au niveau de la complétude des messages (actes de discours, paramètres), mais également au niveau du bon suivi de l'ordre des messages. La conversation représentée au niveau de la figure 63 respecte partiellement les spécifications : la complétude des messages est totalement respectée, mais le bon suivi de l'ordre des messages ne l'est pas. Au niveau de la figure 63, on peut remarquer que la deuxième et la troisième séquence ont été inversées. Ainsi, l'acte de discours de la deuxième séquence est "accept_proposal" au lieu d'être "propose" selon les spécifications et, de même, l'acte de discours de la troisième séquence est "propose" au lieu d'être "accept_proposal". La figure 64 montre comment ProtocolBuilder détecte ces inconformités.

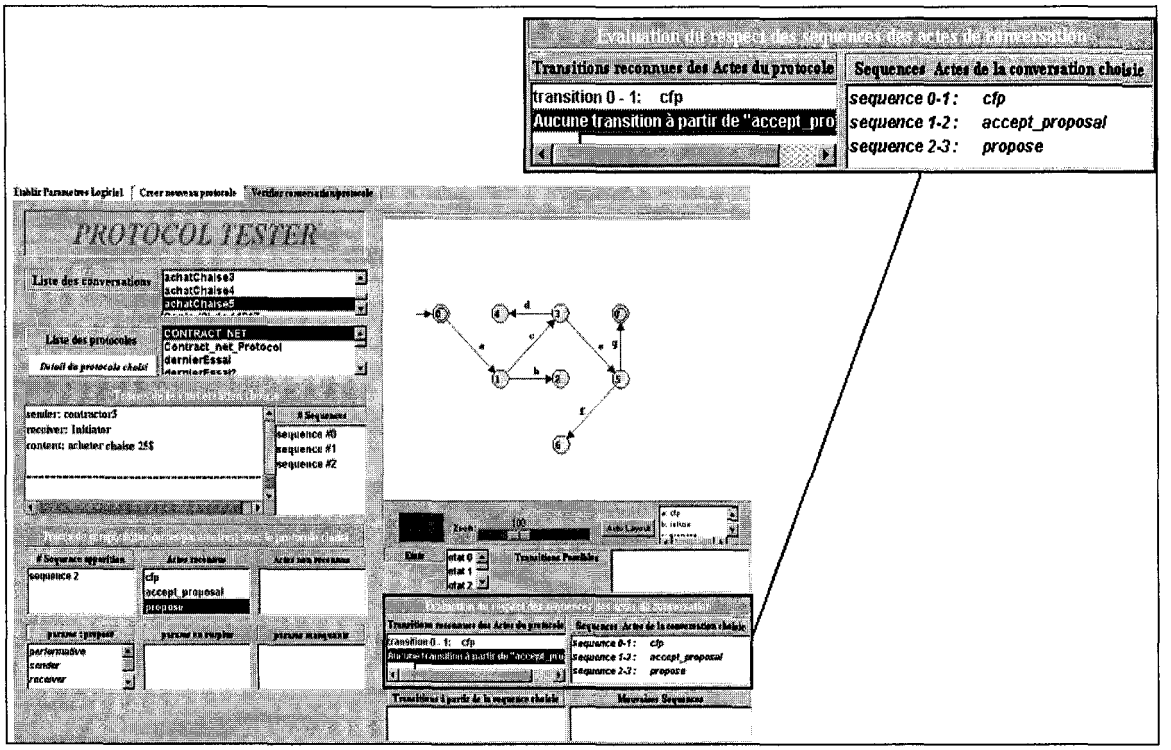


Figure 64: Interface ProtocolBuilder gérant la conversation de la figure 63

Chapitre 5 : CONCLUSION

Dans le cadre du domaine de l'intelligence artificielle, on a survolé les différents aspects liés à la communication inter-agent (langages de communication, sémantiques, failles au niveau des outils existants permettant la communication inter-agent). On a proposé une approche qui permettrait la résolution des failles identifiées, notamment par l'intermédiaire du logiciel "*ProtocolBuilder*". Le logiciel développé (« ProtocolBuilder ») permet de créer des protocoles de communication de manière générique, flexible (selon les désirs de l'usager). Ce logiciel permet également de vérifier l'exactitude des conversations (inter-agent) qui suivent un protocole déjà créé (par l'intermédiaire des actes de discours et leurs paramètres, sans oublier les séquences respectives d'apparition de ces actes au sein de la conversation).

Pour récapituler, le but de mon projet était de concevoir un environnement de développement et d'expérimentation au niveau des systèmes multiagents tentant de combler plusieurs lacunes identifiées au niveau de la communication inter-agent, plus précisément au sein des protocoles de communications. Comme voulu, notre environnement permet non seulement de créer des protocoles de communication, mais également de vérifier l'exactitude et le bon suivi des conversations par rapport aux protocoles (supposé être) suivis au niveau des conversations. La création des protocoles se fait de manière flexible/générique. Cette approche a été évaluée en passant par une série de tests allant de la création/évaluation de protocoles fictifs à la reproduction/évaluation de protocoles existants (exemple: protocole "*ContractNet* ", section 4.3.4.2). Ainsi, comme indiqué au niveau des spécifications (section 4.1), notre environnement *protocolBuilder* possède les caractéristiques suivantes :

- L'environnement permet la spécification, via des interfaces graphiques, de protocoles de communications :

  - spécification du nom des protocoles, suivi de leurs explications.
  - spécification des actes de discours, suivi de leurs significations.
  - spécification des paramètres associés aux actes de discours, suivi de leurs significations respectives.
  - spécification de l'ordre des séquences toléré par l'intermédiaire de la structure des machines à état finis (notre environnement permet une création interactive et intuitive des différentes machines).

- L'environnement permet la vérification, via des interfaces graphiques, de la conformité des conversations par rapport aux protocoles de communications :

  - vérification de l'exactitude des actes de discours utilises au niveau de la conversation par rapport au protocole utilise au sein de cette conversation.
  - Vérification de la complétude des paramètres "accompagnant chaque acte de discours" (utilisé au niveau de la conversation) par rapport au protocole suppose être suivi au sein de cette conversation.

- L'environnement est composé de "bibliothèques":

  - bibliothèque d'actes de discours (fichiers ayant comme extension ".per"),
  - bibliothèque de paramètres pouvant être associés aux différents actes(fichiers ayant comme extension ".param").
  - bibliothèque de structures permettant de générer la partie visuelle des machines à états finis appartenant à des protocoles déjà crées (fichiers ayant comme extension ".machine" pour les machines à états finis et ".descPerf" pour les actes de discours au niveau de chaque arc).

Ces bibliothèques augmentent dynamiquement au fur et à mesure que l'on crée des protocoles, des actes de discours, des paramètres. Ces bibliothèques sont très utiles lorsqu'il s'agit de créer un protocole, ou de vérifier si un protocole est bien suivi au niveau d'une conversation. Plus précisément, ces bibliothèques sont utiles :

- au niveau de l'utilisation des actes de discours et/ou des paramètres existants.
- au niveau de la sauvegarde des protocoles après avoir fini de les spécifier
- Au niveau de la vérification de la complétude des actes de discours et des paramètres utilisés au niveau des conversations :

  - Est-ce que cet acte fait parti de l'ensemble des actes de discours permis au niveau du protocole utilise ?

  - En supposant qu'un acte de discours est reconnu comme faisant parti du protocole, est-ce que les paramètres associes respectivement à chaque acte de discours sont complet/conformes ?

  - Est ce qu'il en manque?

  - Est-ce qu'il y en a en trop ?

- au niveau de la vérification du bon suivi d'une conversation par rapport à un protocole (plus précisément le protocole utilisé au sein de la conversation) : est-ce que l'ordre des actes de discours au sein de la conversation est conforme à l'ordre spécifié au niveau du protocole utilisé ?
- Après être passé à travers les différentes vérifications (conformité des actes de discours/paramètres, bon suivi de l'ordre des messages) au niveau des conversations, permet la validation du bon suivi d'un protocole.

L'environnement témoigne par l'intermédiaire de ses différents modes de création (d'actes de discours, de paramètres associés) d'une bonne **extensibilité**. L'environnement témoigne par l'intermédiaire de l'utilisation des différentes bibliothèques d'une bonne **réutilisabilité**. L'environnement met l'accent sur la facilité et la simplicité de l'exploitation des différents protocoles en permettant une navigation simple et intuitive à l'intérieur des différentes interfaces afin de maximiser l'aspect "**utilisabilité**"

(l'utilisabilité mesure l'aptitude être facile/maniable/agréable au niveau de l'utilisation et de la compréhension). Notre environnement a entièrement été développé à l'aide du langage de programmation orientée objet Java, ainsi d'après les propriétés de développement de ce langage (section 4.2.2), ProtocolBuilder bénéficie également d'une bonne "**interopérabilité**" et d'une bonne "**portabilité**" (la portabilité mesure l'aptitude du logiciel de s'exécuter sur n'importe quelle plate-forme). De plus, on peut utiliser plusieurs outils [42] (AgentTool, Jackal, Jive, JAFMAS) autre que l'outil Jack (section 4.3.4.1) pour simuler des conversations inter-agents pour ensuite les évaluer avec ProtocolBuilder. Après avoir effectué une série de vérification, créé un ensemble de protocoles et évalué les différents facteurs (caractéristiques voulues de notre environnement) établis au niveau de la spécification, on peut conclure que l'ensemble des objectifs de mon mémoire a été atteint.

Nous avons fait ce travail dans l'optique de fournir un outil permettant de combler certaines lacunes au niveau de la communication inter-agent, notamment l'aspect vérification et validation du bon suivi des protocoles de communication au niveau des conversations inter-agent. L'environnement développé ne permet pas pour l'instant de gérer des conversations très complexes utilisant le principe de simultanéité. Par contre, non seulement il fournit tous les éléments de base nécessaires pour la création et l'expérimentation de protocoles de communication de manière flexible/générique, mais peut représenter une base pour de futurs travaux.

Chapitre 6 : TRAVAUX FUTURS

La modélisation des protocoles au sein de notre approche s'est effectuée à travers la structure des machines à états finis. Ainsi la création d'un protocole, crée par ProtocolBuilder, ne peut se faire qu'en série (aucun ensemble d'action concurrente n'est représenté, tout se fait en série). Au niveau des travaux futurs, on peut entrevoir plusieurs voies d'avancement envisageables pour compléter le projet :

- Envisager une modélisation suivant une structure telle que les réseaux de pétri, permettant ainsi la création d'un protocole pourra inclure l'aspect concurrent (permettant ainsi de considérer l'exécution de plusieurs actions concurrentes) au niveau de la création d'un protocole et le bon suivi (vérification/validation) des conversations face à ces protocoles.

- Malgré le fait qu'une évolution s'est fait sentir au niveau de la sémantique des différents actes de la parole dans KQML et Fipa-ACL, on connaît peu au sujet de la sémantique des conversations et des relations entre les actes de la parole et les conversations dont ils font partie. On devrait pouvoir capturer le contenu sémantique des actions au niveau d'un protocole en analysant les interactions entre les participants à travers la signification intrinsèque de ces interactions sachant qu'une sémantique claire des conversations peut faciliter l'extensibilité et le niveau des conversations en tenant compte de la performance et la fiabilité qui sont affectées par l'ajout de plusieurs agents de types différents au sein du système multiagent.

- Les dialogues peuvent prendre plusieurs formes : persuasion, négociation, délibération, quête d'information, ... Au niveau des conversations, les dialogues ne sont habituellement pas que d'un seul type de leur commencement à leur fin. Par exemple, il est commun de commencer un dialogue d'enquête, pour se rendre compte pendant le dialogue qu'il y a une controverse en jeu, pour ensuite entrer dans un sous-dialogue de conflit, et pour reprendre par la suite le dialogue

d'enquête quand le problème a été résolu. Une voie d'avancement serait d'envisager le concept de création de protocoles compliqués à partir de plusieurs sous-protocoles. Ce principe permettrait non seulement d'obtenir une plus grande abstraction des protocoles, mais également une diminution de la complexité au niveau de la vérification/validation des conversations qui emploient ce concept de décalages dialectiques (voir section 3.4.1.3 : Systèmes dialectiques).

Chapitre 7 : REFERENCES

7.1 Bibliographie

[1] Barbuceanu M. et Fox. M. S. (1995) "***Cool***: A language for describing coordination in multi agent systems".
URL: http://www.eil.utoronto.ca/aac/papers/mihai-ICMAS95.pdf

[2] Bouzouba, K. et Moulin, B.*(1998) KQML+ : pour que les agents s'échangent des actes de discours de la conversation humaine.*
URL: http://dein.ucs.br/profs/aribeiro/IAWDAIMAS.pdf

[3] Chaib-draa, B. et Dignum, F. (2002), Trends in Agent Communication Language
URL: http://www.cs.uu.nl/people/dignum/papers/Clarticle1.pdf

[4] Chaib-draa, B. et Vanderveken, D. (1998), Agent Communication Language: A Semantics based on the Success, Satisfaction and Recursion.
URL: http://www.damas.ift.ulaval.ca/publications/ATAL-98.pdf

[5] Chaib-draa, B., Moulin B. et Delisle S., *Analyse et Simulation de conversations: De la théorie des actes de discours aux systèmes multiagents, Lyon : L'interdisciplinaire*

[6] Chaib-draa, B., Labrie, M.-A., et Maudet, N. (2003). Request for action reconsidered as a dialogue game based on commitments.
URL: http://www.damas.ift.ulaval.ca/publications/requestforactionV2.pdf

[7] Colombetti, M. (2000), A commitment-based approach to agent speech acts and conversations.
URL: http://wwwis.win.tue.nl/ac2000/fpapers/colombetti.pdf

[8] Cortadella J., Kishinevsky M., Lavagno L., et Yakovlev A., "Deriving Petri Nets from Finite Transition Systems"
URL: http://www.lsi.upc.es/~jordicf/publications/pdf/tc98_synpn.pdf

[9] Dignum F., Greaves M. (2000), Issues in Agent Communication: An Introduction, Issues in Agent Communication.
URL: http://www.agents.org.au/20000728Dignum-aclissues00.pdf

[10] Ferber, J.,1999. Multi-Agent Systems : an Introduction to Distributed Artificial Intelligence. Reading, MA, Addison-Wesley, `Paper: ISBN 0-201-36048-9`

[11] Finin T., Scott R, Chen Y., Labrou Y., Peng Y. 1999: Colored Petri Nets for Conversations Modeling.
URL:
http://citeseer.ist.psu.edu/rd/60889562,270968,1,0.25,Download/http://citeseer.ist.psu.edu/cache/papers/cs/10856/http:zSzzSzwww.csee.umbc.eduzSz~jklabrouzSzpublicationszSzijcai99acl.pdf/cost99using.pdf

[12] Flores R. A. et Kremer R. C. (2001). Bringing coherence to agent conversations
URL: http://www.auml.org/auml/supplements/Flores-AOSE2001.pdf

[13] Flores R. A. et Kremer, R. C. (2004), A Principled Modular Approach to Construct Flexible Conversation Protocols.
URL: http://www.damas.ift.ulaval.ca/~flores/FloresKremerAI04.pdf

[14] Garneau T. (2003): "DMAS Builder : Un environnement de développement de Systèmes mutli-Agents Totalement distribués.
URL: http://www.uqtr.ca/dmasbuilder/

[15] Greaves M., Holback H., et Bradshaw J. (1999), What Is a Conversation Policy?
URL: http://www.boeing.com/special/agents99/greaves.pdf

[16] Hinchey M., Truszkowski W. et Rouff C., Innovative Concepts for Agent-Based Systems: First International Workshop on Radical Agent Concepts, Wrac 2002, McLean, Va, Usa, January 16-18, 2002

[17] Kone M.T., Shimazu A., et Nakajima T. *The state of the art in agent communication languages.* Knowledge and information systems, 2:259 284, 2000.

[18] Koning J-L., Huget M-P, Wei J., et Wang X.(2003) Interaction Protocol Engineering in Multiagent Systems. MATES 2003 Tutorial
URL: http://www.netobjectdays.org/pdf/03/papers/tutorial/huget.pdf

[19] Krabbe E. Symposium on Argument and Computation Group: Argument and Computational Societies

[20] Labrou Y., Finin T. (1997). Semantics and conversations for an agent communication language
URL: http://www.cs.umbc.edu/~jklabrou/publications/ijcai97.pdf

[21] Labrou Y., Finin T. et Peng Y. (1999), The current landscape of Agent Communication Languages.
URL: http://www.umbc.edu/~finin/papers/ieee99.pdf

[22] Lalanda P. (1998). Shared repository pattern. Proceedings of Pattern Languages of Programming.
URL: http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P24.pdf

[23] Maudet N. et Evrard F., Les jeux: des structures pour l'engagement dans le dialogue. *Information - Interaction - Intelligence*, (Hors série Modèles formels de l'interaction):135-156, 2002.

[24] McBurney P., Parsons S: Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents
URL: http://www.sci.brooklyn.cuny.edu/~parsons/publications/downloads/journals/jolli.ps.gz

[25] Parunak H. V. D., Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis
URL: http://www.erim.org/~vparunak/dooldesn.pdf

[26] Ribeiro, A. et Demazeau, Y. (1998). A Dynamic Interaction Model for Multi-Agent Systems
URL: http://dein.ucs.br/profs/aribeiro/IAWDAIMAS.pdf

[27] Singh, M. P. (1998), Agent communication languages: rethinking the principles.
URL: http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/computer-acl-98.pdf

[28] Willmott, S., Dale J. et Charlton P. (2002). Agent Communication Semantics for Open Environments.
URL: http://icwww.epfl.ch/publications/documents/IC_TECH_REPORT_200240.pdf

[29] Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley and Sons Ltd. ISBN 0 47149691X

7.2 Webliographie

[30] Spécification de la structure des messages FIPA ACL:
http://www.fipa.org/specs/fipa00061/XC00061D.pdf
http://jmvidal.cse.sc.edu/talks/agentcommunication/structure.html

[31] UMBC KQML Web :
http://www.cs.umbc.edu/kqml/

[32] langages communication Agent :
http://www.ryerson.ca/~dgrimsha/courses/cps720/acl.html

[33] XML et Communication Agent :
http://www.ryerson.ca/~dgrimsha/courses/cps720/aclxml.html

[34] Outil "JACK-INTELLIGENT AGENTS" :
www.agent-software.com

[35] Java et programmation orientée objet :
www.java.sun.com

[36] "THE FINITE STATE MACHINE EXPLORER", Matt Chapman :
http://www.belgarath.demon.co.uk/java/fsme.html

[37] Les différents Types d'agent :
http://www.magma.ca/~mrw/agents/types-classification.html

[38] Agents mobiles :
http://www.soi.city.ac.uk/~dk581/projectreport.doc
http://www.guill.net/reseaux/ids.html#4.2
http://www.cs.dartmouth.edu/~dfk/papers/kotz:future2/

[39] programmation orientée objet VS programmation orientée agent:
http://www.jfipa.org/publications/AgentOrientedSoftwareEngineering/

[40] Système multiagent:
http://agents.usc.edu/agents/agents/main.htm
http://www.cs.bham.ac.uk/~axe/aitb/Lecture5a.pdf
http://cdps.umcs.maine.edu/Papers/1996/AUV96-slides.ps
http://www.ipd.bth.se/mfe/teaching/papers/paper_2001_croms.pdf

[41] Exemple de systèmes (monoagent et multiagent) :
- **monoagent commercial** : www.bonzi.com
- **multiagent commercial** : www.quake.com

[42] Systèmes Multiagent :
http://www.multiagent.com/Software/Tools_for_building_MASs/