



**HAL**  
open science

# Modélisation cognitive d'interactions humaines dans un cadre de planification multi-agents

Alexandre Pauchet

► **To cite this version:**

Alexandre Pauchet. Modélisation cognitive d'interactions humaines dans un cadre de planification multi-agents. Autre [cs.OH]. Université Paris-Nord - Paris XIII, 2006. Français. NNT: . tel-00125259

**HAL Id: tel-00125259**

**<https://theses.hal.science/tel-00125259>**

Submitted on 18 Jan 2007

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

*N° attribué par la bibliothèque :*

## **THÈSE**

pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITÉ PARIS NORD

Mention : Informatique

présentée par

**Alexandre PAUCHET**

Titre de la thèse :

**Modélisation cognitive d'interactions humaines  
dans un cadre de planification multi-agents**

Directrice de thèse :

Amal EL FALLAH SEGHRUCHNI

Co-encadré par :

Nathalie CHAIGNAUD

Soutenue le 22 Septembre 2006, devant la commission d'examen

### COMPOSITION DU JURY :

Mme	Nathalie	CHAIGNAUD	Encadrante
Mme	Amal	EL FALLAH SEGHRUCHNI	Directrice de thèse
M.	Daniel	KAYSER	Examinateur
Mme	Sylvie	PESTY	Rapporteur
M.	Jean-Paul	SANSONNET	Rapporteur
M.	Henry	SOLDANO	Examinateur



# Préambule

## Résumé

Cette thèse vise à améliorer la conception des systèmes multi-agents, par l'étude et la modélisation des capacités humaines de planification et d'interaction. Elle s'appuie sur une expérimentation psychologique durant laquelle des sujets humains avaient à résoudre un problème de planification en connaissances incomplètes.

Les protocoles collectés au cours de cette expérimentation ont été analysés du point de vue de la planification et du point de vue des interactions. Ainsi, un modèle de la planification humaine et un modèle de l'interaction humaine ont pu être conçus.

Ces deux modèles sont intégrés de façon homogène à une nouvelle architecture d'agent appelée BDIGGY. Parmi les modèles d'agents existants, le modèle BDI (Belief, Desire, Intention) offre un cadre intéressant pour la conception d'agents délibératifs capables de planifier et d'interagir selon leurs états mentaux. Ainsi, l'architecture BDIGGY est née d'une fusion entre le système IGGY (un système pré-existant de planification humaine) et une architecture BDI étendue à la résolution coopérative de problèmes.

Le modèle de l'interaction humaine introduit dans BDIGGY est sur deux niveaux :

1. il s'appuie sur la théorie des actes de langage pour modéliser les énoncés, à l'aide d'un ensemble de performatives (comme c'est le cas dans les ACL<sup>1</sup> classiques) appliquées à des états mentaux ;
2. il utilise un modèle du discours, représenté par des automates temporisés, pour décrire la dynamique des conversations humaines, en remplacement des protocoles de communication jugés trop rigides.

Les niveaux de l'énoncé et du discours sont liés par une sémantique des performatives. Cette sémantique décrit les pré-conditions et les post-conditions portant sur les états mentaux des agents, à la réception et à l'envoi de messages.

Dans BDIGGY, l'interaction, la planification et les connaissances s'entrelacent grâce aux concepts BDI.

L'architecture BDIGGY est validée en comparant au cours d'un test "à la Turing", les protocoles expérimentaux provenant de l'expérimentation psychologique et les protocoles générés par des agents BDIGGY, durant une simulation de la résolution du problème.

---

<sup>1</sup>Langage de Communication entre Agents (Agent Communication Language)

## Abstract

This PhD thesis aims at improving the design of multi-agent systems, thanks to the study and the modelling of human capabilities of planning and interaction. It is based on a psychological experiment where human subjects had to solve a planning problem with incomplete information.

The protocols collected during the experiment has been analyzed from from the planning point of view and from the interaction point of view. Thus, a model of human planning and a model of human interaction are proposed.

The human planning model and the human interaction model are integrated homogeneously into a new agent architecture called BDIGGY. Among existing agent models, the BDI (Belief, Desire, Intention) one offers an interesting framework to design deliberative agents able to plan and interact according to their mental states. Therefore, the BDIGGY architecture is a merging of the IGGY system (a pre-existent human planning model) and the BDI architecture extended to a cooperative problem solving context.

The human interaction model included in BDIGGY is twofold :

1. it is based on the speech act theory to model the utterances, with a set of performatives (as in classical ACL) applied to beliefs and desires ;
2. it uses a discourse model, represented by timed automata, to describe the dynamics of human conversations, instead of rigid communication protocols.

The utterance level and the discourse level are synchronized by a semantics of the performatives. This semantics summarizes the pre-conditions and the post-conditions that should be verified when receiving and sending a message, according to knowledge of the agent.

In BDIGGY, the interaction, planning and knowledge are linked thanks to the BDI concept.

The BDIGGY architecture is validated by comparing, with a "Turing-like" test, the experimental protocols from the psychological experiment and the protocols generated by the BDIGGY agents, during a simulation of the problem solving.

## Remerciements

Je tiens à remercier en tout premier lieu Amal El Fallah Seghrouchni et Nathalie Chaignaud pour leur encadrement, leurs nombreux conseils et leur soutien constant tout au long de ma thèse. Leur complémentarité scientifique m'a été particulièrement bénéfique, tant du point de vue bibliographique que méthodologique. Cette thèse n'aurait sans doute jamais vu le jour sans leur confiance et leur patience.

Je remercie également les rapporteurs Sylvie Pesty et Jean-Paul Sansonnet pour l'intérêt qu'ils ont porté à mon travail, ainsi que pour les discussions passionnantes que nous avons eu sur les ACA<sup>2</sup>. Je tiens également à remercier Daniel Kayser pour m'avoir fait l'honneur de présider mon jury et Henry Soldano pour avoir accepté d'examiner ma thèse.

---

<sup>2</sup>Agents Conversationnels Animés

Par ailleurs, la validation des modèles doit beaucoup à Jean-Philippe Kotowicz, Alain Loisel, Violaine Philippini, Valérie Lourenço et Étienne Côme, mes "experts". Merci pour le temps passé à comparer les protocoles expérimentaux et les protocoles générés. J'espère sincèrement pouvoir leur rendre la pareille à l'avenir.

Il faut aussi souligner l'apport indéniable des outils qu'ont développés Michel Cusenza, Émilien Hamel, Alexandre Pillon, Julien Melano et Aurélien Patry au cours de projets que j'ai encadré à l'INSA de Rouen. Grâce à eux, mon travail d'analyse a été grandement simplifié.

Je remercie également tous les membres du LIPN pour m'avoir accueilli aussi chaleureusement, et plus particulièrement durant ces deux dernières années en tant qu'A.T.E.R.. Il faut aussi souligner l'excellent accueil qui m'a été fait au LITIS - INSA Rouen (anciennement PSI), au cours de mes premières années de thèse.

Merci enfin à toute ma famille et mes amis de m'avoir supporté et aidé, particulièrement à Hassine, mon indispensable co-bureau, et à Violaine, re-lectrice, "experte", vulgarisatrice, soutien moral, etc.



# Table des matières

<b>Introduction</b>	<b>15</b>
<b>I Positionnement</b>	<b>19</b>
<b>1 Projet de l'Agence de Voyage</b>	<b>21</b>
1.1 Cadre expérimental . . . . .	21
1.1.1 Définition du problème . . . . .	21
1.1.2 Mise en place de l'expérimentation . . . . .	22
1.1.3 Protocoles expérimentaux . . . . .	28
1.2 Étude de la planification humaine . . . . .	28
1.2.1 Analyse des protocoles expérimentaux . . . . .	28
1.2.2 Modélisation de planification humaine . . . . .	30
1.2.3 Le système IGGY . . . . .	34
1.3 Conclusion . . . . .	37
<b>2 Contexte dans lequel s'insère le Projet de l'Agence de Voyage</b>	<b>39</b>
2.1 Systèmes multi-agents . . . . .	39
2.1.1 Capacités cognitives/réactives des agents . . . . .	40
2.1.2 Coordination des SMA . . . . .	42
2.1.3 Architectures d'agent cognitif . . . . .	43
2.1.4 Discussion . . . . .	48
2.2 Un modèle de l'énoncé : la théorie des actes de langage . . . . .	49
2.2.1 Fondements de la théorie des actes de langage . . . . .	49
2.2.2 Théorie Austinienne des actes de langage . . . . .	51
2.2.3 Formalisation de la théorie : Searle et Vanderveken . . . . .	52
2.2.4 Théorie des actes de langage et théorie de la pertinence . . . . .	54
2.2.5 Sémantique des actes de langage . . . . .	55
2.2.6 Discussion . . . . .	58
2.3 Modèles de dialogue . . . . .	58
2.3.1 Modèles fondés sur la planification . . . . .	60
2.3.2 Modèles fondés sur la structuration . . . . .	62
2.3.3 Modèles fondés sur l'interaction . . . . .	64
2.3.4 Modèles fondés sur les conventions . . . . .	65
2.3.5 Modèles mixtes . . . . .	66
2.3.6 Discussion . . . . .	67
2.4 ACL et protocoles de communication . . . . .	67
2.4.1 KQML . . . . .	67
2.4.2 FIPA-ACL . . . . .	70

2.4.3	KQML+ . . . . .	71
2.4.4	CoLa . . . . .	74
2.4.5	Représentation des protocoles de communication . . . . .	76
2.4.6	Discussion . . . . .	77
2.5	Conclusion . . . . .	79

## **II Analyse et modélisation des interactions humaines 81**

### **3 Analyse des protocoles expérimentaux du point de vue de l'interaction 83**

3.1	Analyse des énoncés . . . . .	83
3.1.1	Liste de performatives . . . . .	85
3.1.2	Comparaison avec KQML et FIPA-ACL . . . . .	88
3.1.3	Étude statistique . . . . .	89
3.1.4	Discussion . . . . .	90
3.2	Dynamique conversationnelle . . . . .	90
3.2.1	Actes illocutoires, interventions et échanges . . . . .	92
3.2.2	Temporalité des échanges . . . . .	95
3.2.3	Digressions . . . . .	97
3.2.4	Étude statistique . . . . .	98
3.2.5	Discussion . . . . .	100
3.3	Conclusion . . . . .	101

### **4 Modélisation des interactions humaines 103**

4.1	Description formelle du Problème de l'Agence de Voyage . . . . .	103
4.1.1	Syntaxe utilisée . . . . .	104
4.1.2	Termes et ensembles de termes . . . . .	105
4.1.3	Prédicats et ensembles de prédicats . . . . .	106
4.1.4	Actions élémentaires externes . . . . .	106
4.1.5	Contraintes formelles pour la résolution du problème . . . . .	108
4.1.6	Discussion . . . . .	109
4.2	Modélisation de la dynamique conversationnelle . . . . .	109
4.2.1	Automates temporisés . . . . .	109
4.2.2	Modélisation des échanges par automates temporisés . . . . .	110
4.2.3	Automates de demande d'information . . . . .	112
4.2.4	Automates de proposition d'information . . . . .	114
4.2.5	Automates d'envoi spontané d'information . . . . .	116
4.2.6	Automates de traitement des erreurs . . . . .	116
4.2.7	Discussion . . . . .	116
4.3	Modélisation des énoncés . . . . .	118
4.3.1	Description des états mentaux . . . . .	119
4.3.2	Modèle formel d'un énoncé . . . . .	120
4.3.3	Sémantique générale des performatives . . . . .	121
4.3.4	Sémantique des descriptifs . . . . .	122
4.3.5	Sémantique des directifs . . . . .	125
4.3.6	Sémantique des engageants . . . . .	130
4.3.7	Discussion . . . . .	130
4.4	Conclusion . . . . .	131

<b>III</b>	<b>Mise en oeuvre et validation</b>	<b>133</b>
<b>5</b>	<b>Mise en oeuvre des modèles cognitifs de planification et d'interaction : l'architecture BDIGGY</b>	<b>135</b>
5.1	Description de BDIGGY . . . . .	135
5.1.1	Module de perception . . . . .	136
5.1.2	Système IGGY2 . . . . .	138
5.1.3	Interpréteur de plans . . . . .	139
5.1.4	Module de communication . . . . .	145
5.2	Fonctionnement de BDIGGY . . . . .	147
5.2.1	Initialisation . . . . .	147
5.2.2	Perception - Décision - Action . . . . .	148
5.2.3	Liens entre planification et interaction . . . . .	148
5.2.4	Spécificités des dialogues observés . . . . .	151
5.2.5	Réglage du système . . . . .	152
5.2.6	Exemple de fonctionnement . . . . .	152
5.3	Conclusion . . . . .	160
<b>6</b>	<b>Validation</b>	<b>163</b>
6.1	La problématique de la validation . . . . .	163
6.2	Génération de protocoles pour la validation . . . . .	164
6.3	Un test " <i>à la Turing</i> " . . . . .	165
6.4	Conclusion . . . . .	168
	<b>Conclusion générale</b>	<b>169</b>
	<b>Annexes</b>	<b>175</b>
	Annexe A . . . . .	175
	Annexe B . . . . .	185
	Annexe C . . . . .	192
	<b>Bibliographie</b>	<b>212</b>



# Table des figures

1	Projet de l'agence de voyage . . . . .	17
1.1	Expérimentation psychologique . . . . .	23
1.2	Espace de travail des transports aériens . . . . .	25
1.3	Carte du réseau aérien . . . . .	26
1.4	Exemple de protocole expérimental . . . . .	29
1.5	Architecture d'IGGY . . . . .	36
2.1	Architecture BDI de PRS . . . . .	44
2.2	Architecture BDI de Guerra-Hernandez . . . . .	44
2.3	Architecture d'agents d'Allen, Blaylock et Ferguson . . . . .	47
2.4	Graphe d'une requête par Winograd et Flores . . . . .	59
2.5	Protocole FIPA : Request . . . . .	73
2.6	Protocole d'interaction de KQML+ . . . . .	74
2.7	Modélisation par RdPC du protocole FIPA-Inform . . . . .	77
2.8	Exemple de conversation et son Dooley Graph associé . . . . .	78
3.1	Fichier brut d'un trinôme . . . . .	84
3.2	Exemple de chronogramme . . . . .	91
3.3	Regroupement de messages en échanges . . . . .	93
3.4	Imbrication des échanges . . . . .	94
3.5	Exemple de digression . . . . .	99
4.1	Exemple d'automate temporisé . . . . .	110
4.2	Automates de demande d'information . . . . .	113
4.3	Automates de proposition d'information . . . . .	115
4.4	Automates d'envoi spontané d'information . . . . .	117
4.5	Automate de traitement des erreurs . . . . .	117
4.6	Communication entre agents . . . . .	120
5.1	Flux de l'architecture BDIGGY . . . . .	137
5.2	Structure de la mémoire . . . . .	137
5.3	Enchaînements possibles des phases . . . . .	139
5.4	Enchaînements entre interaction et planification . . . . .	150



# Liste des tableaux

1.1	Utilisation des stratégies de planification . . . . .	30
1.2	Critères définissant les personnalités . . . . .	35
2.1	Syntaxe de Cohen et Levesque . . . . .	55
2.2	Syntaxe de Dignum et Weigand . . . . .	57
2.3	Exemple de schéma d'action . . . . .	60
2.4	Liste des performatives de KQML . . . . .	69
2.5	Liste des performatives de FIPA-ACL . . . . .	72
2.6	Performatives de KQML+ . . . . .	72
3.1	Liste des performatives . . . . .	87
3.2	Comparaison avec KQML et FIPA-ACL . . . . .	88
3.3	Répartition statistique des performatives . . . . .	89
3.4	Exemples d'échanges . . . . .	96
3.5	Les différents types d'échanges . . . . .	97
3.6	Répartition statistique des échanges . . . . .	99
6.1	Liste des personnalités observées . . . . .	166
6.2	Résultats du test " <i>à la Turing</i> " . . . . .	167



# Introduction

## Problématique et méthodologie

Dans la vie quotidienne, les interactions homme-machine se multiplient. Les Systèmes Multi-Agents (SMA) hétérogènes, c'est-à-dire composés à la fois d'agents humains et d'agents artificiels, se généralisent. Les humains étant déjà dotés de moyens robustes de communication et de raisonnement, les ordinateurs doivent donc s'adapter à leurs comportements et offrir des interfaces efficaces et agréables.

La conception de ces interfaces peut être améliorée par l'étude, la modélisation et la simulation des interactions et du raisonnement humain dans un cadre coopératif. En cela, nous rejoignons Sabah [Sabah *et al.*, 1997] qui affirme que :

*Pour garantir l'ergonomie des interprétations construites par la machine, i.e. leurs conformités aux attentes des utilisateurs, le fonctionnement du système mis en oeuvre doit être analogue à celui de la cognition humaine.*

La démarche adoptée pour répondre à cette problématique est interdisciplinaire et mêle Psychologie Cognitive (PC) et Intelligence artificielle (IA). La PC fournit les méthodes pour la mise en place d'une expérimentation psychologique. Elle permet de recueillir un échantillon de protocoles expérimentaux qui sont analysés afin d'élaborer un modèle général des comportements coopératifs observés. De plus, elle apporte la rigueur nécessaire à l'analyse de ces protocoles. L'IA, quant à elle, aide à développer des modèles implantables simulés par un programme informatique. La validation de ce programme se fait en comparant les comportements observés et les comportements simulés.

Le but de cette étude est de construire des modèles qui soient satisfaisants à la fois au sens de la PC et de l'IA, comme le définit Mendelsohn [Bastien *et al.*, 1991] :

*Pour les chercheurs en IA, un bon modèle du fonctionnement cognitif est un modèle formalisable par les systèmes existants, sinon, il est inutilisable. Pour les psychologues, un bon modèle doit être expérimentable ; dans le cas contraire, il perd de sa pertinence, par rapport aux faits psychologiques qu'il est censé décrire.*

Ce travail fait suite à un projet financé par le Groupement d'Intérêt Scientifique - Science de la Cognition (GIS-SC) (1997-2000). Ce projet, intitulé "Modélisation de comportements humains de résolution collective de problèmes", impliquait Nathalie Chaignaud, Amal El Fallah Seghrouchni et François Lévy pour leurs connaissances en IA et Anh Nguyen-Xuan et Charles Tijus pour leurs apports en PC. Cette étude avait pour but de mettre en évidence des comportements de planification dans le cadre d'une résolution collective de problèmes et de comparer le

modèle de comportements humains avec des systèmes informatiques et des conceptions théoriques existants.

Ce projet s'appuie sur une expérimentation psychologique durant laquelle des sujets humains ont résolu un problème de planification pour lequel la coopération était indispensable. Cette expérimentation psychologique a permis de récupérer un corpus de *protocoles expérimentaux*, chacun d'eux décrivant le comportement d'un sujet humain au cours de l'expérimentation. Un protocole expérimental est constitué par l'enregistrement des actions réalisées par le sujet et leur verbalisation. Pour cela, il a été demandé aux sujets de décrire oralement toutes leurs actions durant la résolution du problème. C'est cette verbalisation concomitante, recueillie simultanément par un expérimentateur, qui est incluse dans le protocole expérimental de chaque sujet. Ce corpus expérimental a été analysé pour construire un modèle cognitif de planification humaine en contexte coopératif [Chaignaud *et al.*, 2000] étendant IGGY, un modèle cognitif préexistant de planification humaine individuelle [Chaignaud, 1996], [Chaignaud et Levy, 1996].

Cette thèse, quant à elle, vise à observer, modéliser et simuler l'interaction humaine, ainsi que ses liens avec la planification dans un cadre coopératif. Les protocoles expérimentaux sont étudiés du point de vue des interactions entre sujets. Cette analyse permet de construire un modèle de l'interaction humaine. De plus, les modèles de l'interaction et de la planification humaine en contexte coopératif sont intégrés de façon homogène dans une architecture d'agent. Parmi les modèles d'agent existants, l'architecture BDI (Belief, Desire, Intention)<sup>3</sup> offre un cadre intéressant pour développer des agents cognitifs pouvant planifier leurs actions grâce à une représentation de leurs états mentaux. L'architecture BDIGGY que nous proposons est donc fondée sur une fusion du système IGGY, implantation en machine d'un modèle cognitif de résolution individuelle de problèmes, et d'une architecture BDI. Ainsi, ces deux composantes s'entrelacent pour résoudre le problème.

Cette architecture d'agent permet de simuler les capacités de planification et d'interaction au cours de la résolution de problèmes en contexte coopératif. Les modèles cognitifs ainsi que l'architecture ont été validés en comparant les protocoles expérimentaux avec les protocoles issus de la simulation.

Les travaux présentés ici, s'insèrent dans ce que nous avons appelé le "Projet de l'Agence de Voyage", dont le déroulement est décrit FIG. 1. En caractères fins sont représentées les étapes effectuées durant le projet financé par le GIS-SC tandis que celles en caractères gras concernent nos travaux de thèse.

## Plan de la thèse

Ce travail se divise en trois parties contenant chacune deux chapitres. La première partie, consacrée au contexte général de la thèse, décrit le projet de l'agence de voyage dans lequel s'inscrit cette étude (chapitre 1) et présente un ensemble de travaux existants, relatif à notre problématique (chapitre 2). La seconde partie décrit l'analyse de protocoles expérimentaux du point de vue de l'interaction humaine (chapitre 3) et la modélisation des interactions humaines (chapitre 4). La troisième partie introduit une nouvelle architecture d'agent appelée BDIGGY (chapitre 5), dont la mise en oeuvre a permis la validation des modèles cognitifs (chapitre 6).

---

<sup>3</sup>Croyance, Désir, Intention

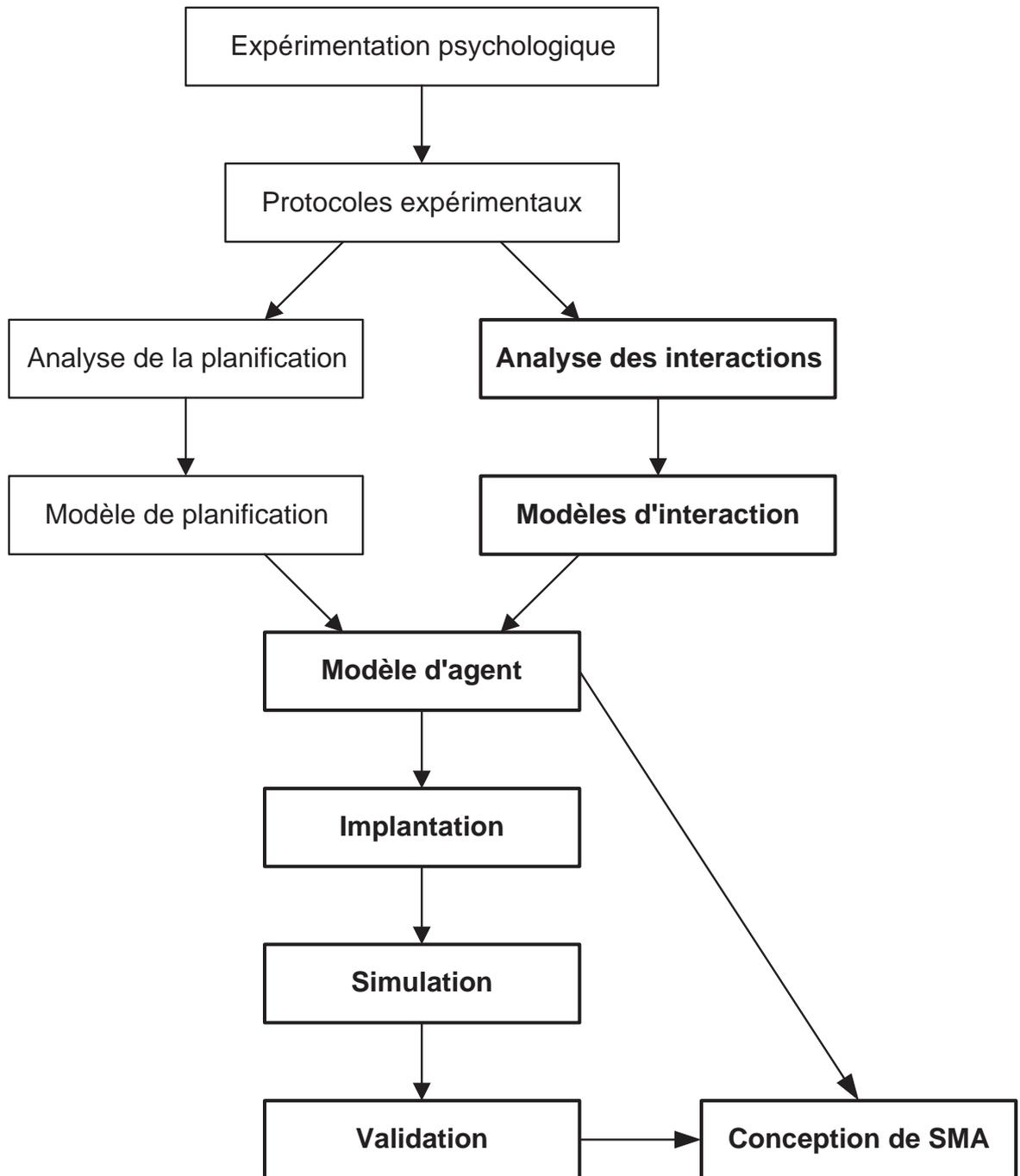


FIG. 1 – Projet de l'agence de voyage

Détaillons maintenant chacun de ces chapitres.

Dans le chapitre 1, le cadre expérimental du Projet de l'Agence de Voyage est approfondi par :

- une description de l'expérimentation psychologique, durant laquelle des sujets humains devaient résoudre un problème de planification en contexte coopératif ;
- une présentation d'un travail antérieur au Projet de l'Agence de Voyage proposant un modèle humain de résolution individuelle de problèmes, pouvant être étendu à la résolution coopérative de problèmes.

Le chapitre 2 présente un récapitulatif des travaux relatifs à la modélisation cognitive des interactions humaines, en contexte coopératif. Les concepts nécessaires à la compréhension des agents et des SMA sont explicités, puis un historique de la théorie des actes de langage (souvent utilisée comme théorie de l'énoncé), est donné, ainsi que divers modèles de dialogue. Enfin, quelques formalismes de représentation des conversations humaines ou logicielles sont exposés.

Le chapitre 3 est consacré à l'analyse des protocoles expérimentaux du point de vue de l'interaction. Cette analyse est faite qualitativement et quantitativement, en se focalisant sur les énoncés pris séparément et sur leur articulation au cours de la conversation.

Le chapitre 4 se concentre sur la modélisation des interactions humaines. Les automates temporisés sont utilisés pour modéliser la dynamique conversationnelle observée durant l'analyse des protocoles expérimentaux. Le modèle de l'énoncé se présente sous la forme d'un système de performatives, dont la sémantique est donnée en termes de croyances, désirs et intentions.

Le chapitre 5 décrit l'architecture d'agent proposée, appelée BDIGGY, qui intègre de manière homogène, les modèles d'interaction et de planification humaine en contexte coopératif. BDIGGY s'appuie à la fois sur une architecture BDI et sur le système IGGY de résolution individuelle de problèmes. Cette architecture nous permet de simuler les comportements humains observés dans les protocoles expérimentaux et de générer informatiquement de nouveaux protocoles.

Dans le chapitre 6, la validation des modèles a pour but de comparer les protocoles expérimentaux et les protocoles générés informatiquement.

La conclusion reprend les principaux points abordés dans cette thèse et discute de la pertinence de nos modèles, au regard des approches existantes. Enfin, quelques perspectives d'approfondissement et d'élargissement de ces modèles sont avancées.

**Première partie**

**Positionnement**



# Chapitre 1

## Projet de l'Agence de Voyage

Le but du Projet de l'Agence de Voyage est de modéliser les comportements humains de résolution coopérative de problèmes en connaissances incomplètes. Il est nécessaire donc d'observer la manière dont les humains raisonnent durant la résolution d'un problème.

Une expérimentation psychologique au cours de laquelle des sujets humains doivent résoudre un problème a donc été mise en place. Elle permet de récupérer un échantillon de protocoles expérimentaux. Ces protocoles ont été analysés afin d'élaborer un modèle général des comportements observés.

Dans ce chapitre, l'expérimentation psychologique sur laquelle s'appuie le projet de l'agence de voyage est détaillée : les caractéristiques des problèmes qui ont été proposés aux sujets, le déroulement de l'expérimentation et le contenu des protocoles expérimentaux récupérés sont présentés.

Les protocoles expérimentaux sont ensuite analysés du point de vue de la planification humaine. Cette étude a permis de construire un modèle de planification humaine en résolution coopérative de problèmes [Chaignaud *et al.*, 2000].

### 1.1 Cadre expérimental

#### 1.1.1 Définition du problème

Le projet de l'agence de voyage s'intéresse à la résolution de problèmes en contexte coopératif et plus particulièrement aux situations où :

- **individuellement**, chaque sujet a son propre problème à résoudre :
  - le sujet est suffisamment habitué à l'environnement du problème pour se concentrer uniquement sur sa résolution,
  - les informations qui lui sont fournies sont incomplètes pour l'obliger à planifier,
  - le sujet doit respecter les contraintes du problème ;
- **collectivement**, les sujets doivent coopérer s'ils veulent résoudre leur problème :
  - les compétences des sujets sont complémentaires,
  - les sujets interagissent par une communication écrite en langage naturel (courriers électroniques).

Dans le problème choisi, trois employés d'une agence de voyage possèdent des compétences différentes. Le premier est spécialisé dans le transport aérien, le second dans le transport ferroviaire et le dernier dans le transport routier (bus et taxis).

Chacun des sujets est chargé d'organiser un voyage pour un client différent. Ces voyages sont caractérisés par :

- une ville de départ en France,
- une ville d'arrivée en France,
- une date et une heure de départ,
- une date et une heure d'arrivée,
- un nombre de voyageurs pour la réservation,
- un budget en Francs<sup>1</sup>.

Un *voyage* peut être décomposé en *étapes* ou *trajets*, reliant deux villes intermédiaires du voyage. Le terme *étape* est utilisé pour décrire une sous-partie du *voyage* et le terme *trajet* quand le voyage n'est pas précisé.

Aucun voyage ne peut se faire via un moyen unique de transport. Ainsi, chaque sujet<sup>2</sup> possède son propre problème à résoudre et participe à la résolution des deux autres problèmes.

Voici la description des trois problèmes qui ont été proposés aux sujets et qu'ils ont résolus au cours de l'expérimentation psychologique.

1. *Problème soumis au sujet en charge des transports aériens* : "Une famille (deux parents et deux adolescents de 17 et 14 ans) habitant à Château-neuf (Loiret - 45) veut partir en vacances à Maguelone (Hérault - 34) dans quatre jours. Son budget est d'environ 3300F.
2. *Problème soumis au sujet en charge des transports ferroviaires* : "Un représentant a un rendez-vous avec un de ses clients qui l'attendra à la gare de Baisieux (Nord - 59) lundi prochain matin à 11 heures. Il partira directement de chez lui de Beaupréau (Maine-et-Loire - 49). Il souhaite partir le matin même et son budget est d'environ 1310F.
3. *Problème soumis au sujet en charge des transports routiers* : "Trois sportifs (de 25 ans) doivent se rendre à une compétition de hockey sur glace au Chatelard (Savoie - 73). Leur départ est prévu vendredi prochain. Ils partiront de Saint-Martin-du-Tertre (Val d'Oise - 95) où leur entraînement a lieu à la patinoire tout près de la gare, mais il ne pourront pas partir avant sa fin à 17h30. Ils souhaitent arriver au Chatelard en début de soirée et demandent à ce que le prix soit au maximum de 1430F.

### 1.1.2 Mise en place de l'expérimentation

Les trois sujets ont été isolés dans des pièces différentes, devant un ordinateur sur lequel tournait une simulation du problème. Ils devaient résoudre simultanément leur propre problème (voir FIG. 1.1).

Toutes les actions (consultations d'horaire, lectures et envois de messages...) des sujets ont été enregistrées par le simulateur dans des fichiers textes. L'activité mentale du sujet n'étant pas observable, il est nécessaire d'en rendre compte par des moyens indirects. Il a donc été

---

<sup>1</sup>L'expérimentation psychologique a été mise en place avant l'avènement de l'euro.

<sup>2</sup>Le terme *sujet* est utilisé pour désigner les sujets humains ayant participé à l'expérimentation alors que le terme *agent* désigne les agents logiciels qui simulent le comportement des sujets humains.

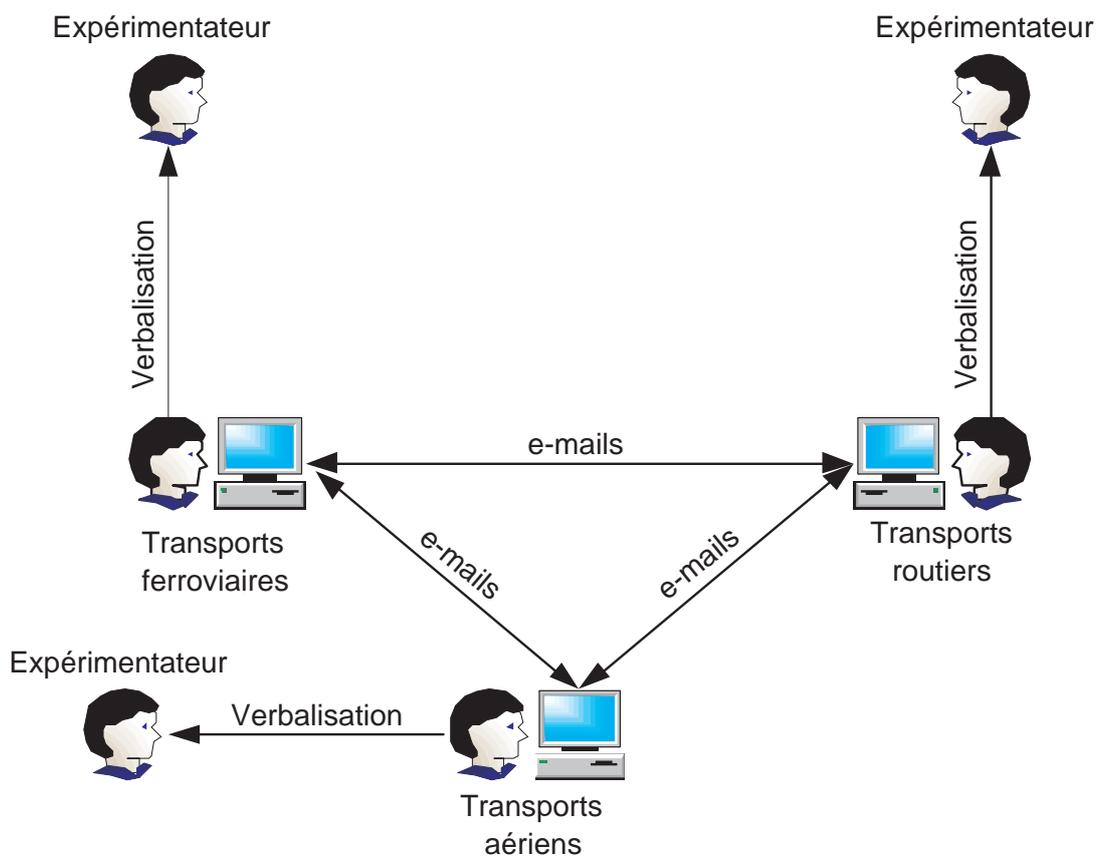


FIG. 1.1 – Expérimentation psychologique

demandé aux sujets de commenter oralement toutes leurs actions au moment de leur réalisation. Cette verbalisation concomitante a été recueillie par un expérimentateur et retranscrite par écrit. Pour leurs communications, les sujets utilisaient un système de courrier électronique et interagissaient en langage naturel.

Pour résoudre leur problème, les sujets disposaient d'une interface (voir FIG. 1.2) conçue pour être la plus intuitive possible.

Cette interface qui représente l'espace de travail du sujet en charge des transports aériens, se divise en quatre parties : une pour consulter les horaires et les prix de trajets (panneau "Horaire/Prix"), une pour mémoriser des trajets intéressants (panneau "Zone de travail"), une pour communiquer avec les autres sujets (panneau "Messagerie") et une dernière pour construire et tester une solution (Panneau "Propositions").

Le panneau "Horaires/Prix" permet d'afficher une liste de trajets par simple appui sur le bouton "Afficher", en ayant préalablement choisi dans les menus déroulants la ville de départ, la ville d'arrivée et la compagnie (excepté pour le sujet en charge des transports ferroviaires qui ne possède pas ce menu déroulant dans son interface). Pour ajouter un trajet intéressant à la "Zone de travail", il suffit de cliquer sur le bouton "Mémoriser" après avoir rempli les champs "Date" et "Nb places".

La "Zone de travail" contient en réalité trois zones de travail (les unes sur les autres), pour pouvoir séparer les étapes concernant chacun des problèmes. La sélection d'un problème se fait par choix entre les champs "Mon Problème", "Pb. ag. X" et "Pb. ag. Y" où X et Y sont les deux autres sujets (ferroviaire et routier pour le sujet en charge des transports aériens). Pour le problème sélectionné, la "Zone de travail" présente la liste des étapes ajoutées depuis le panneau "Horaire/Prix". Le contenu de l'étape sélectionnée dans cette liste est détaillé dans une boîte de texte. Le sujet peut supprimer une étape de la liste (bouton "Supprimer"), réserver une étape ou annuler une réservation précédente (bouton "Réserver/Annuler"), insérer une étape dans un message pour l'envoyer à un autre sujet (bouton "Envoyer") ou bien ajouter une étape à la solution du panneau "Proposition" (bouton "Insérer proposition").

La "Messagerie" contient une liste de tous les messages reçus par le sujet ainsi que le contenu détaillé du message sélectionné de cette liste. Le sujet peut détruire le message (bouton "Supprimer"), répondre à l'expéditeur du message sélectionné (bouton "Répondre") ou créer un nouveau message (Bouton "Nouveau message"). Il peut aussi extraire et ajouter un horaire à la "Zone de travail" si le message reçu contient un horaire attaché (bouton "Mémoriser horaire"),

Le panneau "Proposition" permet de construire et tester une solution. Cette solution est représentée sous forme d'une liste d'étapes dont l'étape sélectionnée est détaillée à part. Cette liste est ordonnée verticalement selon l'enchaînement des étapes. Chacune des étapes peut être déplacée vers le début de la liste (bouton "Depl. vers le haut") ou vers la fin (bouton "Depl. vers le bas"), modifiée (bouton "Modifier") ou encore supprimée (bouton "Supprimer"). Le sujet peut aussi insérer une nouvelle étape à la solution (bouton "Insérer"). Enfin, le bouton "Tester" permet de vérifier la validité d'une solution potentielle et indique le "Prix total" de cette solution.

De plus, une carte des principales villes de France et du réseau correspondant à leur spécialité leur a été fournie (voir FIG. 1.3).

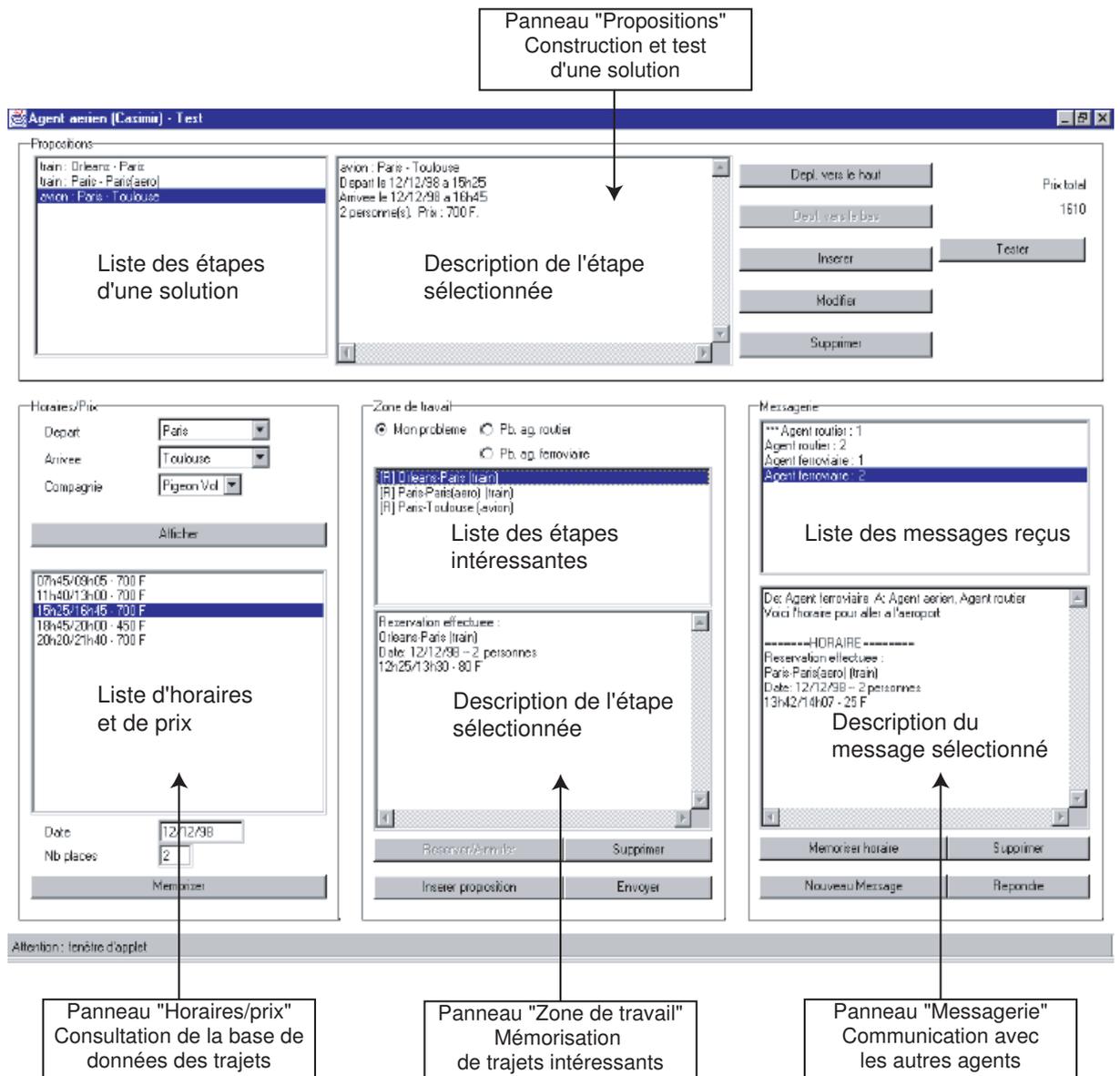


FIG. 1.2 – Espace de travail des transports aériens

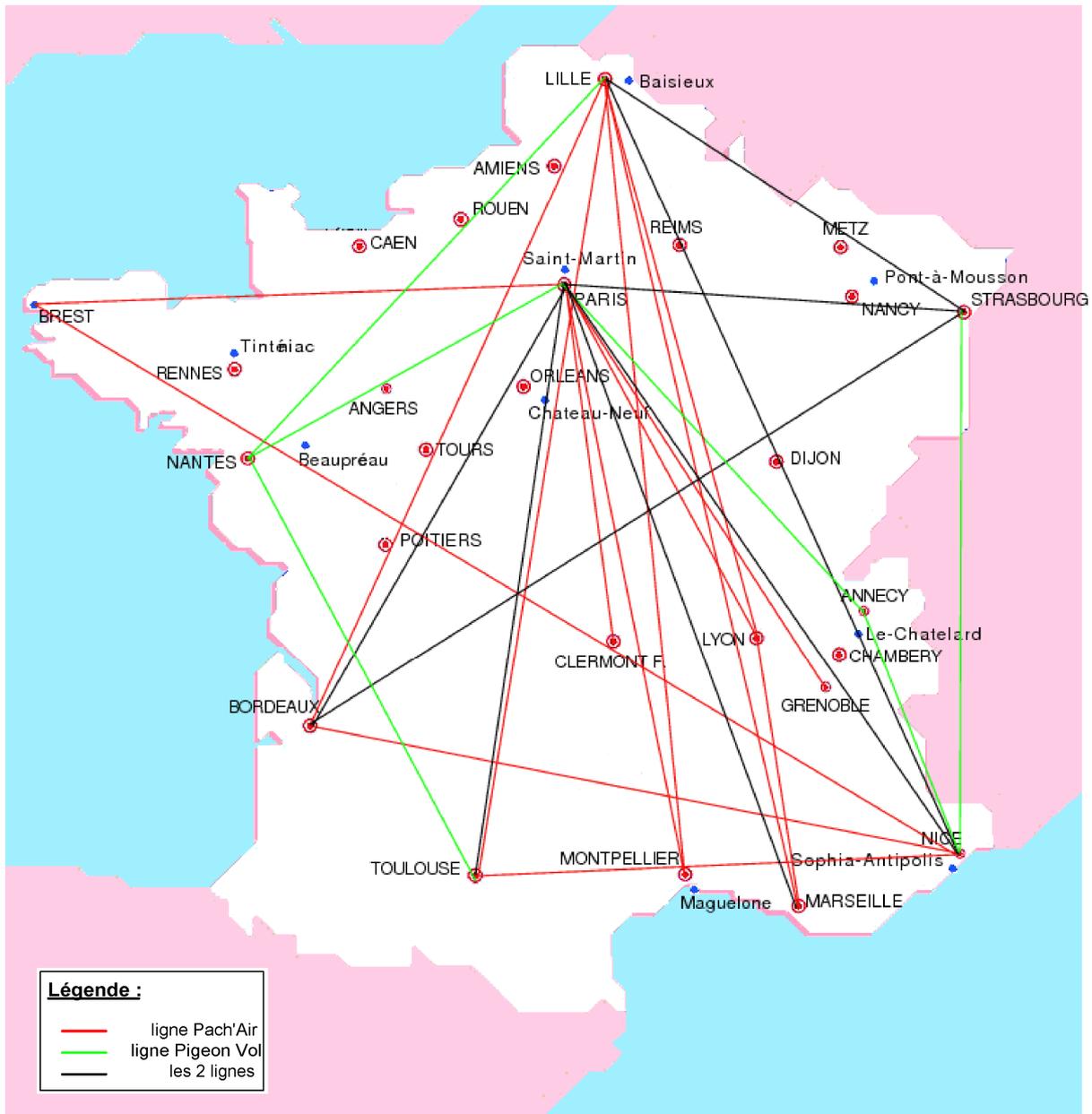


FIG. 1.3 – Carte du réseau aérien

Les sujets se sont familiarisés avec le type de problème et l'interface en résolvant un problème test, avant d'être confrontés à l'expérimentation proprement dite. La consigne suivante leur a été lue afin qu'ils appréhendent correctement la tâche demandée<sup>3</sup> :

*Vous avez bien voulu accepter de passer une expérience. Elle dure une bonne heure. Vous travaillez pour une agence de voyages avec deux autres collègues, dans des locaux différents. Chacun de vous trois est spécialisé dans un moyen de transport : l'un s'occupe des trains, un autre, des avions et le dernier des cars et des taxis.*

*Vous avez chacun une demande de votre client à satisfaire. Les trajets des voyages nécessitent souvent l'utilisation des trois modes de transport (avion, train, route : car et taxi). L'avion et le train permettent des grands déplacements (plus de 30 km) alors que le car ou le taxi ne permettent que de petits déplacements (moins de 30 km).*

*Vous, vous occupez des avions. Parmi vos deux partenaires, l'un s'occupe des trains et l'autre des cars et des taxis.*

*Pour construire le trajet de votre client vous devez communiquer avec vos deux autres collègues afin de :*

- connaître les horaires et tarifs des trains, cars et taxis du voyage dont vous vous occupez,*
- donner à vos collègues les informations dont ils ont besoin sur les avions.*

*Pour communiquer entre vous, vous envoyez et recevez des messages électroniques.*

*Nous allons voir maintenant comment vous pouvez communiquer avec vos collègues et quel est votre espace de travail.*

*Pour cela, nous avons installé un ordinateur,*

À la suite de cette lecture, l'interface a été présentée au sujet, qui devait résoudre un premier problème avec l'aide de l'expérimentateur. Cette phase a duré jusqu'à ce que le sujet soit à l'aise avec cette interface (environ un quart d'heure).

Finalement, le sujet a été informé de son problème, et reçu la consigne suivante :

*Voici la demande d'un nouveau client. Quand vous pensez avoir constitué un voyage qui satisfait le client, vous testez votre proposition.*

*Vous pouvez prendre des notes sur la feuille à votre disposition et vous devez dire tout haut ce que vous faites.*

Le sujet a résolu seul la demande. Cette phase a duré jusqu'à ce que les sujets aient résolu leur trois problèmes ou que des contraintes de temps aient forcé l'arrêt.

Cette expérimentation a été réalisée sur quatorze groupes de trois étudiants de premier cycle. Les quarante deux protocoles obtenus ont été séparés en deux classes distinctes : la première (9x3 protocoles) servant à l'analyse et à la construction de ces modèles et la seconde (5x3 protocoles) servant à la validation de ces modèles.

---

<sup>3</sup>La consigne donnée ici concerne le sujet en charge des transports aériens. Des consignes similaires ont été lues aux deux autres sujets, avec les modifications nécessaires.

### 1.1.3 Protocoles expérimentaux

Cette expérimentation psychologique a permis de récupérer une série de protocoles expérimentaux. Chaque *protocole expérimental* est composé de l'enregistrement des actions effectuées et de leur explicitation orale par le sujet, au moment même de leur réalisation (verbalisation concomitante).

Un exemple de protocole expérimental est présenté FIG. 1.4. On y retrouve, après un préambule décrivant les conditions de l'expérimentation, l'intégralité des actions effectuées par le sujet, numérotées, indexées dans le temps et verbalisées pour certaines.

C'est sur l'analyse de ces protocoles expérimentaux que s'appuient les modèles cognitifs de planification et d'interaction humaine.

## 1.2 Étude de la planification humaine

Cette section décrit l'analyse des protocoles expérimentaux du point de vue de la planification et le modèle cognitif sous-jacent. Celui-ci s'appuie sur un précédent modèle de résolution individuelle de problème appelé IGGY [Chaignaud, 1996], [Chaignaud et Levy, 1996], étendu à la résolution coopérative de problème [Chaignaud *et al.*, 2000].

### 1.2.1 Analyse des protocoles expérimentaux

L'analyse des vingt-sept protocoles à été faite de façon individuelle. Cette analyse des protocoles expérimentaux a montré que les sujets planifiaient bien leurs actions pour résoudre leur problème. Cette planification est faite de manière séquentielle (construction d'un seul plan à la fois) ou parallèle (construction de plusieurs plans à la fois). De plus, l'ordre dans lequel les plans sont conçus varie en fonction du sujet, de manière :

- prospective (du point de départ vers le point d'arrivée),
- rétrospective (du point d'arrivée vers le point de départ),
- centrifuge (en débutant par les étapes centrales du voyage),
- centripète (en débutant par les extrémités du voyage).

Les notions de plans parallèles et de sous-buts parallèles ont été différenciées. Il y a plans parallèles quand le sujet construit plusieurs voyages pour le même problème contenant des étapes différentes. Par contre, des sous-buts parallèles existent pour un plan quand il y a dans ce plan des étapes ayant le même point de départ et le même point d'arrivée (ou des points de départ et d'arrivée situés dans la même ville, comme par exemple la gare et l'aéroport), avec des moyens de transport différents.

Par exemple, les plans

$$P_1 = [(Beaupreau, Angers, train) (Angers, Paris, train)]$$
$$P_2 = [(Beaupreau, Nantes, train) (Nantes, Paris, train)]$$

sont des plans parallèles. Par contre,  $(Beaupreau, Angers, car)$  est un sous-but parallèle pour le plan  $P_1$ . La description formelle des plans parallèles et des sous-buts parallèles est effectuée en 4.1.3.

Enregistrement des actions	Verbalisation concomitante
<p>[09:31:52] Sujet: alexandre  [09:31:52] Rôle: Agent aérien  [09:31:52] Problème: Aller  [09:31:52] Date: Mardi 5 Mai 1998 17:32:43  [09:31:52] 1, Demande horaires Annecy-Annecy (Pach'AIR)  [09:31:55] 2, Demande horaires Annecy-Annecy (Pigeon Vol)  [09:31:59] 3, Demande horaires Annecy-Brest (Pigeon Vol)  [09:32:03] 4, Demande horaires Annecy-Brest (Pach'AIR)  [09:33:45] 5, Nouveau message  [09:34:21] 6, Reçoit message 1 de Agent ferroviaire</p> <p>[09:34:53] 7, Envoie message 1    De: Agent aérien A: Agent ferroviaire    Y a-t-il une gare à Château-Neuf ?  [09:34:57] 8, Affiche message numero 1 de Agent ferroviaire    De: Agent ferroviaire A: Agent aérien    Salut, j'aimerais connaître les prix et les horaires du premier vol NANTES-PARIS pour demain pour 1 personne. merci  [09:35:12] 9, Demande horaires Nantes-Paris (Pach'AIR)  [09:35:15] 10, Demande horaires Nantes-Paris (Pigeon Vol)  [09:35:39] 11, Mémorise horaire  [09:35:39] 12, Ajoute horaire dans panneau Mon problème    Nantes-Paris (avion Pigeon Vol)    Date: Demain -- 1 personne    08h40/09h35 - 595 F  [09:35:43] 13, Envoie horaire Nantes-Paris (avion)  [09:36:05] 14, Reçoit message 1 de Agent routier  [09:36:43] 15, Envoie message 2    De: Agent aérien A: Agent ferroviaire    C'est le plus tôt, le dernier arrive à 20h50, tous à 595F    =====HORAIRE=====    Nantes-Paris (avion Pigeon Vol)    Date: Demain -- 1 personne    08h40/09h35 - 595 F  [09:36:55] 16, Supprime message Agent ferroviaire : 1  [09:36:58] 17, Affiche message numero 2 de Agent aérien    De: Agent aérien A: Agent ferroviaire    C'est le plus tôt, le dernier arrivé 20h50, tous à 595F    =====HORAIRE=====    Nantes-Paris (avion Pigeon Vol)    Date: Demain -- 1 personne    08h40/09h35 - 595 F  [09:37:03] 18, Supprime message (Agent aérien : 2)  [09:37:27] 19, Demande horaires Paris-Annecy (Pach'AIR)  [09:37:29] 20, Demande horaires Paris-Annecy (Pigeon Vol)  [09:37:42] 21, Change panneau : Pb. ag. routier  [09:37:44] 22, Mémorise horaire  [09:37:45] 23, Ajoute horaire dans panneau Pb. ag. routier    Paris-Annecy (avion Pigeon Vol)    Date: Demain -- 1 personne    18h45/20h00 - 450 F  [09:37:46] 24, Envoie horaire Paris-Annecy (avion)  [09:37:57] 25, Reçoit message 3 de Agent ferroviaire  [09:38:03] 26, Reçoit message 2 de Agent routier  [09:38:25] 27, Envoie message 3    De: Agent aérien A: Agent routier    Dois-je réserver ?    =====HORAIRE=====    Paris-Annecy (avion Pigeon Vol)    Date: Demain -- 1 personne    18h45/20h00 - 450 F  [09:38:30] 28, Supprime message *** Agent routier : 1  [09:38:36] 29, Affiche message numero 2 de Agent routier    De: Agent routier A: Agent aérien    Quelle est la duree du vol Paris-Annecy ?  ...  ...</p>	<p>"Si, je mets 4 est-ce qu'il va me mettre le prix..... "</p> <p>"Alors Château-Neuf départ et arrivée Maguelone... Donc Château-Neuf/Orléans en... en... Château-Neuf/Orléans... Donc là... Je me demande si je l'envoie en train directement à Maguelone ou en avion depuis Paris... 3000 F... Donc ils iront en train..."  "Ah! Il faut que je demande s'il y une gare à Château-Neuf, on ne sait jamais.... "</p> <p>"Il est drôle lui... Je vais lui envoyer l'horaire le plus t ôt."</p> <p>" Je vais lui mettre qu'il n'a qu'à faire la différ ence... "</p> <p>...</p>

FIG. 1.4 – Exemple de protocole expérimental

Les principaux résultats de cette analyse sont présentés TAB. 1.1. On y trouve le nombre d'utilisations de chacune des stratégies de planification et le nombre de planifications parallèles entre parenthèses.

	Prospective	Centrifuge	Centripète	Rétrospective
Agents routiers	6(3)	1	1	1(1)
Agents ferroviaires	2(1)	7(4)	0	0
Agents aériens	2(1)	6(3)	0	1

TAB. 1.1 – Utilisation des stratégies de planification

La moitié des sujets a adopté une planification parallèle en construisant des plans prospectifs ou centrifuges. Cette forte utilisation de la planification parallèle est due au long délai des communications faites par envois de messages écrits. Le demandeur a souvent le temps de préparer un nouveau plan en attendant une réponse à une de ses questions, sans pour autant abandonner son idée première.

## 1.2.2 Modélisation de planification humaine

Dans une étude précédente de modélisation cognitive de résolution individuelle de problèmes, Chaignaud a proposé un modèle cognitif implanté par le système IGGY [Chaignaud, 1996], [Chaignaud et Levy, 1996]. L'article [Chaignaud *et al.*, 2000] a montré que ce modèle pouvait être étendu à la planification en contexte coopératif du Problème de l'Agence de Voyage.

Ce modèle est fondé sur l'observation de trois caractéristiques lors de la résolution du problème par les sujets :

- la distinction entre des situations normales et des situations anormales,
- la réduction de l'espace problème en ne considérant pas toutes les contraintes à la fois,
- l'abstraction faite sur les données du problème.

Le modèle construit à partir de cette analyse s'appuie sur les notions de *phase*, *état d'esprit*, *stratégies*, *tactiques* et *observations* pour décrire le comportement des sujets dont les traits de caractère sont représentés par la *personnalité*.

### Phase

Elle permet de différencier les situations *normales* et *anormales*. En effet, la résolution d'un problème ne se fait pas de manière continue car des difficultés peuvent survenir. Deux sortes de difficultés peuvent être rencontrées : les premières sont mineures et ne nécessitent que peu de changements par rapport au plan initial (elles nécessitent l'apport de corrections), alors que d'autres sont plus sérieuses et bloquent la résolution du problème (elles nécessitent une re-planification).

Les différentes phases qui ont été répertoriées sont les suivantes :

- une *phase de planification* pour la construction d'un ou de plusieurs plans à la fois,
- une *phase de correction partielle* d'un plan qui échoue,
- une *phase de vérification* pour faire le point sur la solution partielle construite,

- une *phase de test* permettant de valider ou non une solution complète,
- une *phase de coopération* durant laquelle les sujets interviennent dans la résolution des autres problèmes.

Chaque protocole expérimental est vu comme la succession de ces phases.

## État d'esprit

La décision de réaliser, ou non, une action dépend de l'attention portée aux contraintes du problème. Les sujets satisfont ces contraintes en agissant selon des critères liés à ces contraintes.

Cinq critères se dégagent des contraintes du problème posé.

- Critère d'*horaire* : le sujet fait correspondre les horaires des étapes les unes par rapport aux autres et vérifie que les heures d'arrivée et de départ du voyage satisfont celles souhaitées par le client.
- Critère de *prix* : le sujet porte une attention particulière aux prix de chacune des étapes, pour que le prix du voyage n'excède pas le budget qui lui est imposé.
- Critère de *transfert* : le sujet n'oublie pas les liaisons entre les étapes du voyage, particulièrement entre gare et aéroport d'une même ville.
- Critère de *nombre de personnes* : le sujet s'attache à ce que les étapes du voyage soient prévues pour le bon nombre de voyageurs.
- Critère de *réservation* : le sujet pense à réserver les étapes du voyage.

Les sujets ne prennent pas en compte tous ces critères simultanément mais ils n'en considèrent qu'une partie appelée *état d'esprit*. L'état d'esprit évolue en fonction de la situation (nouvelles informations acquises) et ses changements déclenchent des modifications dans le comportement du sujet.

Par exemple, un état d'esprit  $E = \{prix, horaire, transfert\}$  correspond à un sujet faisant attention, lors du choix d'une étape, à son prix et à ce que les horaires et les liaisons correspondent avec ceux des autres étapes.

## Stratégies

Durant l'analyse des protocoles expérimentaux (voir 1.2.1), certains sujets utilisaient la planification parallèle alors que d'autres construisaient essentiellement des plans séquentiels.

Les stratégies décrivent la manière dont le sujet construit sa solution ainsi que l'ordre dans lequel il va réaliser ses actions. Deux stratégies d'approche sont possibles :

- la *planification parallèle*, quand plusieurs plans sont conçus en parallèles,
- la *planification séquentielle*, quand un seul plan est construit à la fois.

De plus, quatre stratégies de planification ont été utilisées, selon l'ordre de construction des plans :

- la *planification prospective*,
- la *planification rétrospective*,
- la *planification centrifuge*,
- la *planification centripète*.

Dans une phase de planification, les sujets construisent leurs plans en suivant une stratégie d'approche et une stratégie de planification. Dans une phase de correction, les sujets se concentrent sur le plan à corriger, il n'y a donc pas de stratégie associée à cette phase. Enfin, dans une phase de vérification, il existe deux stratégies : vérification du(des) plan(s) ou vérification des messages échangés.

## Tactiques

Dans chacune des phases, la même stratégie peut être réalisée par différentes actions. Les tactiques marquent ces différences.

Durant les phases de planification et de correction, les sujets adoptent des tactiques particulières pour choisir

- les étapes du voyage :
  - le voyage passe par Paris ou ne passe par Paris ;
  - le voyage est optimisé ou est non-optimisé (sans escale inutile) ;
- le moyen de transport :
  - son propre moyen de transport est favorisé ;
  - le moyen de transport le moins cher est favorisé ;
  - le moyen de transport le plus rapide est favorisé ;
  - tous les moyens de transport sont utilisés de façon identique.

Pendant une phase de vérification, les tactiques permettent de vérifier :

- les étapes du voyage :
  - Vérification prospective des étapes ;
  - Vérification rétrospective des étapes ;
  - Vérification centrifuge des étapes ;
  - Vérification centripète des étapes ;
  - Vérification aléatoire des étapes ;
- les messages (reçus et envoyés) :
  - Vérification des messages par ordre chronologique ;
  - Vérification des messages par ordre chronologique inverse ;
  - Vérification des messages d'un sujet spécifique.

## Observations

Le processus de résolution est un enchaînement d'épisodes. Durant la résolution, toute modification d'un ingrédient du modèle (phase, état d'esprit, stratégies ou tactiques) constitue un nouvel épisode. Les caractéristiques de ce nouvel épisode dépendent des raisons de cette modification et de leur lien avec les observations faites sur les données. Ce modèle est donc complété par la notion d'*observations* faites sur l'environnement. Les observations correspondent aux données explicatives concernant la résolution. Elles sont présentes dans la verbalisation lorsque celle-ci est suffisamment riche.

Contrairement aux autres ingrédients, qui représentent les caractéristiques d'un épisode, les observations stigmatisent les causes d'un changement d'épisode. Elles correspondent à une focalisation du sujet sur certains points utiles de la situation.

Elles peuvent avoir une incidence sur :

- la phase (ex : observation d'un blocage dans la planification du voyage, de corrections à apporter à une solution ou de la solution à tester),
- l'état d'esprit (ex : observation d'un dépassement du budget alloué impliquant la prise en compte du critère de prix),
- les stratégies (ex : observation qu'il n'existe pas de gare dans les villes de départ et d'arrivée impliquant une stratégie centrifuge),
- les tactiques (ex : observation qu'il existe une ligne directe entre deux villes impliquant la tactique "voyage optimisé").

Voici la liste des observations utilisées du modèle :

- Observations faites sur les données statiques :
  - *ville-plus-proche(ville1, ville2)* quand le sujet a besoin de remplacer dans son voyage une ville par une autre, proche.
- Observations faites sur le plan :
  - *important(critère)*, avec *critère*  $\in$  {prix, horaire, transfert, réservation, nombre de personnes}, quand le sujet observe qu'il doit tenir compte d'un critère,
  - *à-tester*, quand le sujet pense avoir terminé la résolution de son problème et souhaite tester sa solution,
  - *test-positif*, quand le test de la solution proposée par le sujet est positif,
  - *test-négatif*, quand le test de la solution proposée par le sujet est négatif,
  - *problème(critère, étape)*, avec *critère*  $\in$  {prix, horaire, transfert, réservation, nombre de personnes}, quand le sujet remarque un problème sur une des étapes de sa solution,
  - *corrigé(étape)*, quand le sujet pense avoir résolu un problème observé sur une étape,
  - *étape-à-optimiser(étape)*, quand une étape doit être optimisée,
  - *vérifier-étape(étape)*, quand le sujet pense qu'une étape doit être vérifiée,
  - *étape-vérifié(étape)*, quand le sujet a vérifié une étape,
  - *changer-plan*, en cas de blocage lors de la résolution du problème, lorsque le sujet pense qu'une re planification est nécessaire.
- Observations faites sur les interactions
  - *demande(agent)*, quand le sujet reçoit une demande de la part d'un des autres sujets,
  - *demande-satisfaite(agent)*, quand le sujet pense avoir satisfait une demande d'un autre sujet,
  - *en-attente(agent)*, quand le sujet attend un message de la part d'un autre sujet,
  - *attente-trop-longue(agent)*, quand le sujet attend trop longtemps une réponse,
  - *message(agent)*, quand le sujet reçoit un nouveau message,
  - *à-vérifier-messages*, quand le sujet souhaite vérifier ses messages,
  - *messages-vérifiés*, quand les messages ont été vérifiés.

## Personnalité (ou profil du sujet)

La *personnalité* permet de marquer les différences individuelles entre sujets. Huit traits de caractère la définissent :

- *attentif*, selon la fréquence des erreurs d'inattention que produit le sujet,
- *coopératif*, selon si le sujet coopère ou non avec les autres,
- *économe*, selon l'importance que le sujet attache aux prix des étapes du voyage,

- *bon estimateur*, si le sujet est capable d'évaluer correctement une situation,
- *opportuniste*, selon la capacité du sujet à utiliser les informations à sa portée,
- *optimisateur*, si le sujet cherche à optimiser son plan solution,
- *patient*, selon le délai qu'accepte le sujet entre deux demandes identiques sans réponses,
- *précis*, si le sujet est exhaustif dans ses demandes.

Pour chacun des sujets, ces caractéristiques prennent toutes une valeur parmi *{faiblement, moyennement, très}* (représentée par un entier : 1, 2 ou 3).

TAB 1.2 présente les critères qui nous permettent de valuer les traits de caractères de chaque protocole.

## Fonctionnement du modèle

Pour résumer ce modèle, en fonction de sa personnalité (huit traits de personnalité instanciés) et des observations qu'il fait à un instant  $t$ , le sujet se trouve dans une phase avec un certain état d'esprit (critères présents) et agit selon des stratégies et des tactiques.

L'ANNEXE A contient une analyse de protocole du point de vue de la planification.

### 1.2.3 Le système IGGY

Ce modèle est l'extension d'un précédent modèle de planification conçu dans le cadre d'une résolution individuelle de problèmes : le problème de la chaîne Hi-Fi [Chaignaud, 1996], [Chaignaud et Levy, 1996].

Le modèle de résolution individuelle a fait l'objet d'une implantation en Common Lisp, sous la forme du système IGGY. Il s'agit d'un générateur de protocoles qui prend en entrée un problème à résoudre ainsi que la personnalité du sujet simulé et génère un protocole expérimental (voir FIG. 1.5, tirée de [Chaignaud, 1996]).

IGGY s'articule autour d'une architecture de tableau noir à contrôle hiérarchique sur lequel se greffe un exécuteur. Ce dernier effectue les actions dictées par le tableau noir et génère les protocoles.

Le tableau noir est formé de cinq panneaux thématiques :

- les *données statiques* et les connaissances intuitives liées au problème ainsi que la personnalité du sujet simulé,
- les *données dynamiques* du problème, correspondant aux informations accessibles à l'écran (consultation des bases de données, etc.),
- les *observations* faites sur les données statiques et dynamiques,
- l'*épisode courant*, composée d'une phase, d'un état d'esprit, de stratégies et de tactiques,
- le *squelette du protocole*, donnant un historique des épisodes passés de la résolution.

Un *résumé* du tableau noir, mis à jour à chaque écriture et à chaque exécution d'une action, renseigne le système sur le dernier événement survenu.

Les spécialistes du domaine sont regroupés en cinq familles sous la direction d'un contrôleur du domaine : *Contrôleur Observations*, *Contrôleur Phase*, *Contrôleur État d'esprit*, *Contrôleur Stratégies* et *Contrôleur Tactiques*. Chaque contrôleur d'un domaine possède un nombre fini

Traits de caractère	Valeur	Comportement correspondant
attentif	1	Les envois de message contiennent de nombreuses erreurs de destinataire et de nombreux trajets sont erronés.
	2	Peu d'erreurs de destinataire dans les envois de message et quelques erreurs dans les trajets.
	3	Aucune erreur de destinataire dans les envois de message et peu d'erreurs dans les trajets.
économe	1	Le voyage est construit sans se préoccuper du prix des étapes, sauf en cas de problème.
	2	Le prix des moyens de transport est pris en compte occasionnellement, même sans l'apparition d'un problème pour le prix du voyage.
	3	Le voyage est construit en essayant d'abord les moyens de transport les moins coûteux. Le prix des étapes intervient souvent.
bon estimateur	1	De nombreux tests négatifs précèdent la construction d'une solution correcte.
	2	Quelques tests négatifs avant de trouver une solution.
	3	Une solution est trouvée avec au maximum un seul test négatif.
coopératif	1	Les demandes obtiennent difficilement une réponse.
	2	Les réponses aux demandes sont moyennement rapides.
	3	Les réponses aux demandes sont très rapides.
opportuniste	1	La résolution du problème ne contient ni planification parallèle, ni sous-buts parallèles.
	2	La résolution du problème utilise uniquement des sous-buts parallèles.
	3	La résolution du problème utilise planification parallèle et sous-buts parallèles.
optimisateur	1	Le voyage contient un maximum d'étapes différentes.
	2	Le voyage contient un nombre moyen d'étapes.
	3	Le voyage est construit en minimisant le nombre d'étapes.
patient	1	Le temps entre deux relances est court et le nombre de relances sans réponse est faible.
	2	Le temps entre deux relances et le nombre de relances sans réponse sont moyens.
	3	Le temps entre deux relances est long et le nombre de relances sans réponse est élevé.
précis	1	Le voyage est construit sans aucune précision sur les horaires, sauf en cas de problème.
	2	Les étapes du voyage sont conçues avec quelques précisions sur les horaires.
	3	Toutes les étapes sont construites avec un maximum de précisions sur les horaires.

TAB. 1.2 – Critères définissant les personnalités

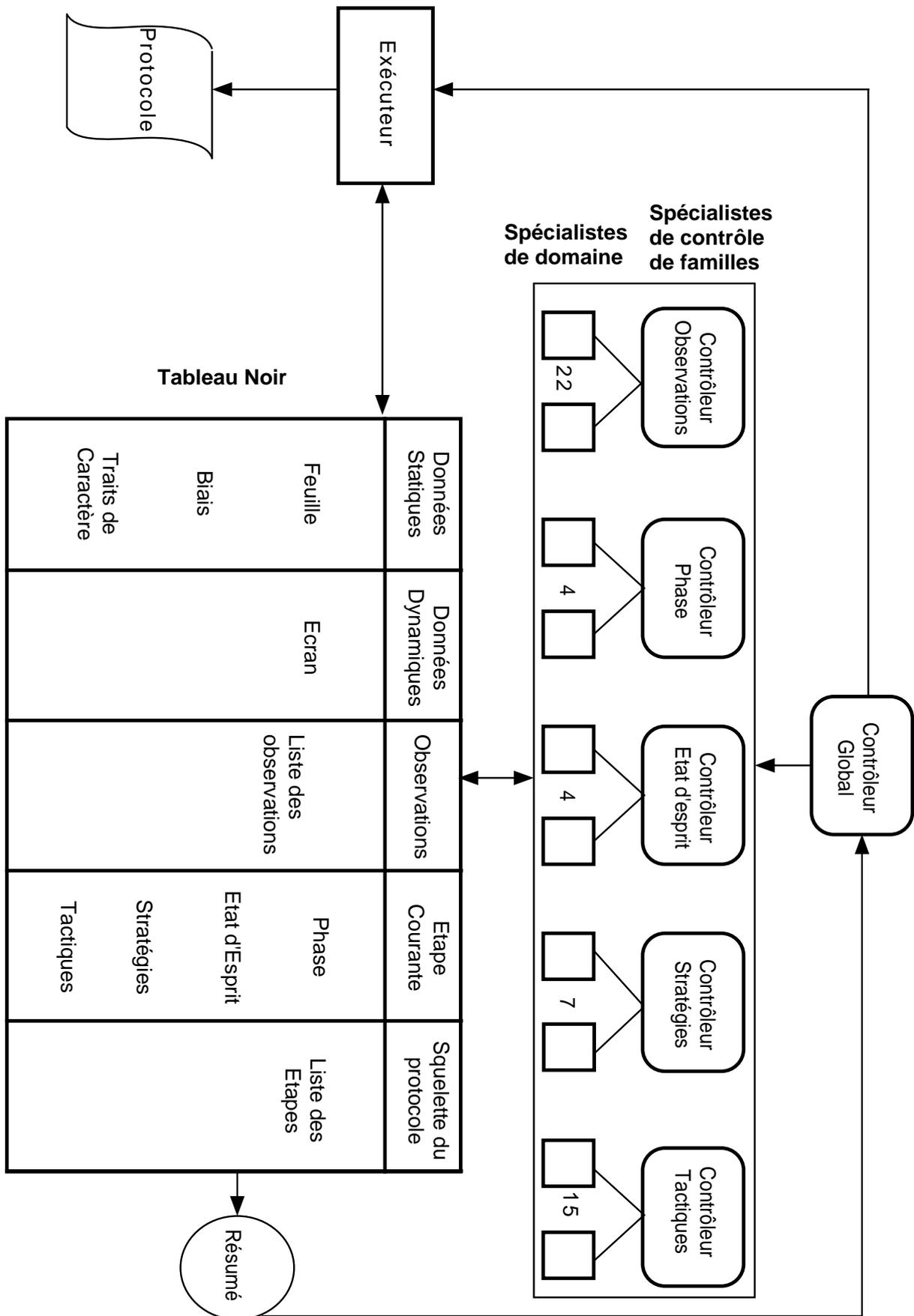


FIG. 1.5 – Architecture d'IGGY

de spécialistes. Les spécialistes représentent chacun une valeur possible dans leur famille. Ils sont activés si les conditions sont réunies dans le tableau noir et proposent leur valeur pour un ingrédient.

Le *contrôleur global* a pour rôle de superviser l'ensemble des contrôleurs de familles.

Enfin, l'*exécuteur* génère les suites d'actions correspondant à l'épisode courant.

## 1.3 Conclusion

Le Projet de l'Agence de Voyage dans lequel s'inscrit cette étude, s'appuie sur un cadre expérimental rigoureux. L'expérimentation psychologique mise en place a fourni un ensemble de traces de résolution de sujets (protocoles expérimentaux) pour un problème de planification humaine en contexte coopératif.

Ce corpus a été analysé du point de vue de la planification humaine. Cette analyse a montré que le modèle proposé dans un cadre de résolution individuelle de problèmes, et implanté sous la forme du système IGGY, s'adaptait bien à la planification en contexte coopératif.

L'analyse qui avait été faite au sein du projet n'avait pas été réalisée sur l'ensemble des 9 trinômes. Nathalie Chaignaud et moi-même avons refait cette analyse conjointement sur l'ensemble des 27 protocoles (9 trinômes). Cela nous a permis de préciser les indices dans les protocoles permettant de faire leur découpage en épisodes puis de préciser les différents ingrédients (phase, état d'esprit, stratégies, tactiques, observations et personnalité).

De plus, ce modèle de résolution coopérative de problèmes n'avait pas fait l'objet d'une implantation avant cette thèse. Pour résoudre le problème de l'agence de voyage, nous avons donc redéveloppé le système IGGY, dont la nouvelle version est appelé IGGY2 pour des raisons de clarté (voir SEC. 5.1.2).



## Chapitre 2

# Contexte dans lequel s'insère le Projet de l'Agence de Voyage

La problématique abordée dans le cadre du Projet de l'Agence de Voyage est double : étudier l'interaction humaine et la planification humaine dans un cadre coopératif.

Cette thèse se focalise plus particulièrement sur l'interaction humaine et sur son articulation avec la planification dans un but de simulation informatique. Ce chapitre regroupe les travaux antérieurs liés à ces thèmes.

Dans un premier temps, il est nécessaire de définir les concepts d'*agent* et de *SMA*, et de fournir quelques exemples d'architectures d'agents relatifs à cette étude. Ensuite, un historique concernant la théorie des actes de langage est proposé. Puis, quelques modèles qui permettent de représenter les dialogues humains sont décrits. Enfin, les principales implantations des théories précédentes, sous la forme de Langage de Communication entre Agents (ACL, Agent Communication Language) et de *protocoles de communication*, sont présentées.

### 2.1 Systèmes multi-agents

L'Intelligence Artificielle Distribuée (IAD) est une branche de l'IA. Contrairement à l'IA qui se consacre à la modélisation du comportement d'une seule entité, l'IAD s'intéresse à l'activité conjointe de plusieurs composants à l'intérieur d'un système.

L'IAD revêt de plus en plus d'importance pour sa capacité à aborder des problèmes complexes, aussi bien en résolution de problèmes, qu'en simulation. Elle permet de distribuer l'expertise et donc de diviser les problèmes en sous-problèmes. Cependant, les problèmes complexes ne sont pas toujours aisément divisibles en sous-problèmes indépendants. Ainsi, l'IAD s'appuie sur les SMA constitués de nombreux agents en interaction dynamique avec le monde extérieur, dans le but d'une coopération efficace. Les solutions du problème sont alors le résultat des activités réalisées en commun par les agents. Les SMA offrent, en plus du parallélisme, une certaine robustesse en permettant une re-configuration automatique du système en cas de besoin.

D'après Ferber [Ferber, 1995], un SMA est composé :

- d'un *environnement E* ;

- d'un *ensemble d'objets situés*  $O$  (il est possible de leur associer une position dans  $E$ ) ; ils sont passifs et peuvent être perçus, créés, détruits et modifiés par les agents ;
- d'un *ensemble d'agents*  $A$ , qui sont des objets particuliers représentant les entités actives du système ( $A \subseteq O$ ) ;
- d'un *ensemble de relations*  $R$  unissant des objets (et donc des agents) entre eux ;
- d'un *ensemble d'opérations*  $O_p$  permettant aux agents  $A$  de percevoir, produire, consommer, transformer et manipuler les objets de  $O$  ;
- d'*opérateurs* chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification (que l'on appelle les lois de l'univers).

Toujours d'après Ferber [Ferber, 1995], dans un SMA :

- chaque agent possède des informations ou des capacités de résolution de problèmes limitées, chaque agent a un point de vue partiel ;
- il n'y a pas de contrôle global du système ;
- les données sont décentralisées ;
- le calcul est asynchrone.

Les SMA peuvent être classés suivant de nombreux critères : taille et nombre des agents, mécanismes d'interaction et types de communication, représentation des autres agents et de l'environnement, capacité de coopération de chaque agent, organisation et contrôle dans le SMA, capacité d'apprentissage des agents, etc. Cependant, deux caractéristiques principales se dégagent des SMA.

- Les *capacités cognitives des agents* précisent leurs aptitudes à planifier leurs actions, à raisonner sur les actions et les plans des autres agents et à évaluer l'environnement. En effet, les agents peuvent s'adapter plus aisément s'ils peuvent prédire le comportement des autres agents et les changements dans l'environnement.
- L'*organisation du SMA* définit les liens qui existent entre chaque agent, la structure des communications entre les agents et le degré de coopération.

### 2.1.1 Capacités cognitives/réactives des agents

Il existe de nombreuses tentatives pour définir le concept d'agent. Par exemple, Sycara [Sycara, 1998], donne la définition suivante :

*Un agent est un système logiciel dont les principales caractéristiques sont le caractère situé, l'autonomie, l'adaptativité et la sociabilité :*

**Le caractère situé** (*situatedness*), signifie que l'agent reçoit une certaine forme d'input sensoriel de son environnement qui l'amène à produire certaines actions qui le modifient d'une certaine façon, tout en modifiant cet environnement.

**L'autonomie** signifie que l'agent peut agir sans intervention directe de la part d'une personne ou d'autres agents et qu'il contrôle ses propres actions et son état interne.

**L'adaptativité** signifie que l'agent est capable : (1) de réagir de façon flexible aux changements de son environnement ; (2) de prendre, lorsque nécessaire, des initiatives orientées vers un but ; et (3) d'apprendre, à partir de sa propre expérience, de son environnement et de ses interactions avec d'autres agents.

**La sociabilité** signifie que l'agent est capable d'interagir avec d'autres agents, que ceux-ci soient des personnes ou des agents logiciels.

Cette définition n'est pas totalement satisfaisante car elle distingue agents logiciels et agents humains. Celle de Ferber [Ferber, 1995] semble préférable :

*On appelle agent une entité physique ou abstraite*

- *qui est capable d'agir dans un environnement,*
- *qui peut communiquer directement avec d'autres agents,*
- *qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction qu'elle cherche à optimiser),*
- *qui possède des ressources propres,*
- *qui est capable de percevoir de manière limitée son environnement,*
- *qui dispose d'une représentation partielle de cet environnement (éventuellement aucune),*
- *qui possède des compétences et offre des services,*
- *dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.*

Les agents ont une représentation partielle (éventuellement aucune) de leur environnement et d'eux-même. La complexité de cette représentation varie selon deux écoles : l'école cognitive et l'école réactive.

Les agents *réactifs* ne possèdent ni représentation (ou une représentation minimale) de l'environnement dans lequel ils évoluent, ni mécanisme d'envoi de messages. Leurs capacités fonctionnent sur un mode de type stimulus/action qui peut tout de même être considéré comme une forme de communication. Un SMA constitué d'agents réactifs, possède généralement un grand nombre d'agents et présente un comportement global intelligent. Un agent est considéré comme réactif lorsqu'il agit uniquement en fonction de stimuli (internes et externes). Les *éco-agents* [Ferber et Drogoul, 1992] sont un exemple d'architecture réactive d'agents, utilisable en planification. Ce modèle trouve son inspiration dans le comportement de sociétés d'insectes dans la nature. Un éco-agent est un agent très simple qui cherche toujours à satisfaire ses besoins. S'il est gêné dans son activité, il agresse les autres. Ce modèle peut être utilisé pour la résolution distribuée de problèmes.

À l'opposé, les agents *cognitifs* sont qualifiés "d'intelligents". Ils possèdent une représentation partielle de l'environnement, des buts explicites et sont capables de planifier leur comportement, de mémoriser leurs actions passées, de communiquer par envois de messages, de négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents, mais l'architecture de ces agents est plus complexe à mettre en oeuvre que celle des agents réactifs. Ce type d'architecture est particulièrement intéressant car plus proche des caractéristiques humaines. Des exemples d'architecture cognitive d'agents sont présentés en 2.1.3.

Il existe aussi des architectures *hybrides* qui combinent les deux types d'approches pour répondre au mieux à un problème. Dans une telle architecture, un agent est composé de modules qui gèrent indépendamment la partie réactive et cognitive du comportement de l'agent. Le problème central reste de trouver le mécanisme idéal de contrôle assurant un bon équilibre et une bonne coordination entre ces modules.

## 2.1.2 Coordination des SMA

Il y a essentiellement deux approches de coordination dans les SMA comportant des agents cognitifs. La première approche, appelée *planification multi-agents centralisée*, confie à un agent spécialisé le soin de décider pour régler les conflits à partir de la vue globale qu'il possède des agents. La seconde approche est appelée *planification multi-agents distribuée*. Aucun agent n'a de vue globale des activités du groupe. Dans ce cas, la détection des conflits est beaucoup plus difficile et il importe que chaque agent dispose des capacités de planification prenant en compte les plans des autres agents. Le Problème de l'Agence de Voyage est un bon exemple de problème de planification multi-agents distribuée.

Une structure adéquate pour un SMA doit présenter les caractéristiques suivantes :

- *Couverture* : toutes les actions nécessaires à la solution du problème doivent faire partie des capacités d'au moins un des agents.
- *Connectivité* : les agents doivent pouvoir interagir de façon à ce que les actions puissent s'intégrer dans la solution globale.
- *Capacité* : la couverture et la connectivité doivent pouvoir se réaliser dans les limites des ressources communicationnelles disponibles et avec un degré de fiabilité pré-spécifié.

Dans un SMA, chaque agent contrôle ses actions et ses interactions avec les autres agents. Les notions de coordination et de coopération sont très importantes pour comprendre les mécanismes permettant d'atteindre le résultat recherché. Ces deux concepts décrivent en réalité la même chose, c'est-à-dire les mécanismes qui contraignent les agents à ajuster leur comportement pour participer à une activité commune. De l'extérieur, un agent se coordonne avec les autres, de l'intérieur, l'agent coopère.

Cependant, la coopération n'est pas toujours effective dans les SMA. Comme le souligne Chaib-Draa [Chaib-Draa, 1988], pour résoudre un problème donné,

*[les SMA] sont conçus et implantés comme un environnement et un ensemble d'agents interagissant selon des modes de coopération, de concurrence ou de coexistence.*

Le point crucial des SMA est donc la communication. Comme le dit Ferber [Ferber, 1995] :

*Sans communication, l'agent n'est qu'un individu isolé, sourd et muet aux autres agents, refermé sur sa boucle perception-délibération-action. C'est parce que les agents communiquent qu'ils peuvent coopérer, coordonner leurs actions, réaliser des tâches en commun et devenir ainsi de véritables êtres sociaux.*

Koning et Pesty distinguent deux modes de communication [Koning et Pesty, 2001] :

- La *communication indirecte* via l'environnement correspond à l'utilisation de signaux (*stimuli* via l'environnement) ayant des interprétations fixées. Cette classe de communication s'adresse surtout à des agents réactifs. Ce type de communication ne permet pas d'exprimer de commandes complexes comme les requêtes.
- La *communication directe par passage de message* permet de planifier un acte de communication envers un autre agent. On parle alors de communication intentionnelle. Ce type de communication nécessite des traitements beaucoup plus complexes de la part des intervenants et est donc utilisé par des agents cognitifs.

Les communications dans un SMA composé d'agents cognitifs nécessite des techniques

plus sophistiquées qu'un simple passage de données ou un appel de procédures. Même dans le cas de systèmes coopératifs, il faut tenir compte de l'hétérogénéité des agents et de leur autonomie : leur comportement ne doit dépendre ni d'un contrôle centralisé, ni d'un formatage de leurs communications.

Éviter d'imposer un système d'interaction prédéfini de communication permet de se rapprocher du langage naturel qui, ne l'oublions pas, possède un très grand pouvoir d'expressivité. Outre une satisfaction scientifique indéniable, une grande expressivité permet de se prémunir d'une spécialisation excessive des agents et maintenir possible l'interopérabilité entre différentes architectures d'agent.

De plus, la façon de communiquer d'un agent doit être compatible avec son fonctionnement interne. La capacité de communication d'un agent fait partie intégrante de son modèle cognitif et comportemental. Pour un agent donné, une conversation est un processus dynamique qui met en jeu l'essentiel de ses ressources cognitives.

Actuellement, l'approche classique dans les SMA pour gérer les communications se structure autour de l'utilisation d'un ACL associé à des protocoles de communication. Cette approche est décrite SEC. 2.4.

### 2.1.3 Architectures d'agent cognitif

Les SMA composés d'agents cognitifs capables de communiquer de façon complexe sont les plus pertinents pour notre étude. Ces agents "intelligents" possèdent les caractéristiques suivantes :

- une représentation explicite des connaissances ;
- une structuration de leurs états mentaux ;
- des capacités cognitives : mise à jour des connaissances, planification et autonomie.

Cette partie est consacrée à deux architectures d'agent ayant ces caractéristiques. Il s'agit des agents BDI [Rao et Georgeff, 1995] et de l'architecture d'Allen, Blaylock et Ferguson [Allen *et al.*, 2002].

#### Architectures BDI

Le modèle BDI [Rao et Georgeff, 1995] est issu des travaux sur l'intentionnalité, mené notamment par Bratman [Bratman, 1987], [Bratman *et al.*, 1988].

La première architecture BDI implantée est PRS (Procedural Reasoning System) [Georgeff et Lansky, 1987], réalisée en LISP. PRS a été réimplanté en C++ sous le nom de dMARS (distributed Multi-Agent Reasoning System) [d'Inverno *et al.*, 1997]. Il existe maintenant de nombreuses autres implantations des agents BDI, quelles soient commerciales comme JACK [Howden *et al.*, 2001] ou universitaire comme celle de Guerra-Hernandez [Guerra-Hernandez, 2003].

Tous les systèmes BDI (voir FIG. 2.1, tirée de [d'Inverno *et al.*, 1997], et FIG. 2.2, tirée de [Guerra-Hernandez, 2003]) sont constitués des éléments suivants :

- Les *croyances* (B) représentent la vision que l'agent a du monde. Elles correspondent aux informations que l'agent possède sur l'environnement et sur les autres agents. Elles

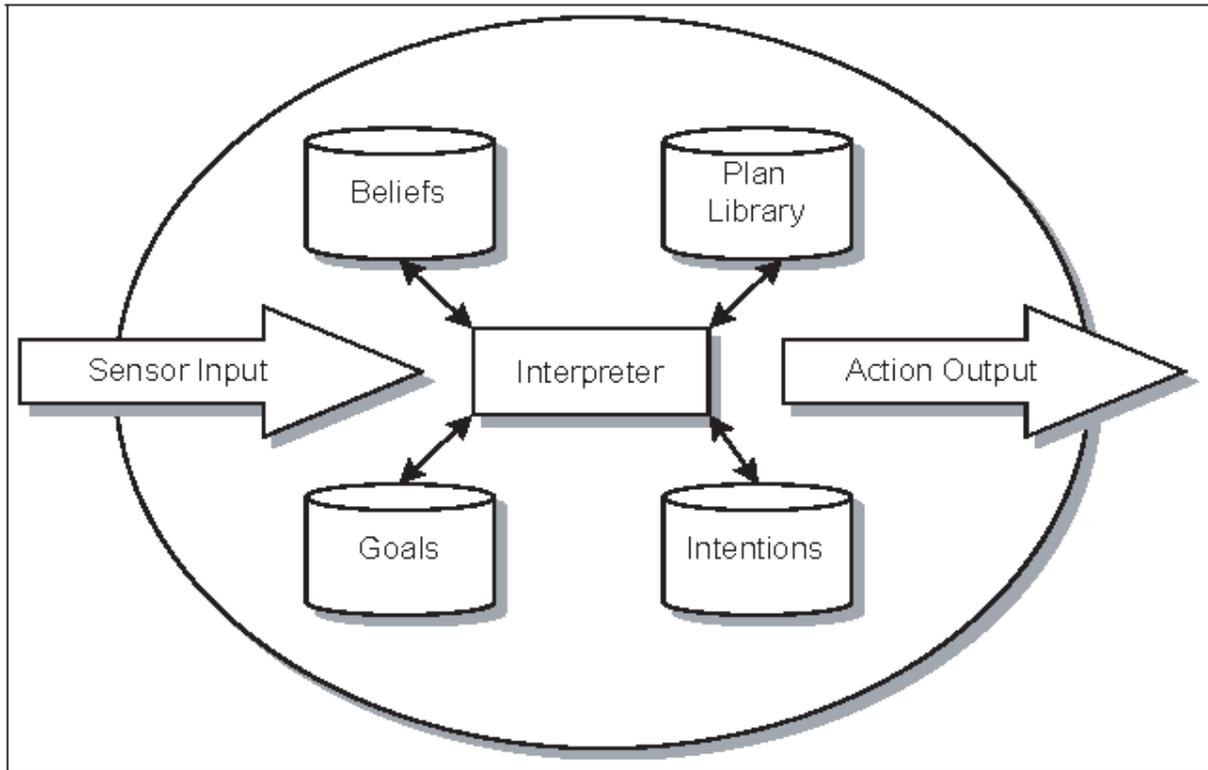


FIG. 2.1 – Architecture BDI de PRS

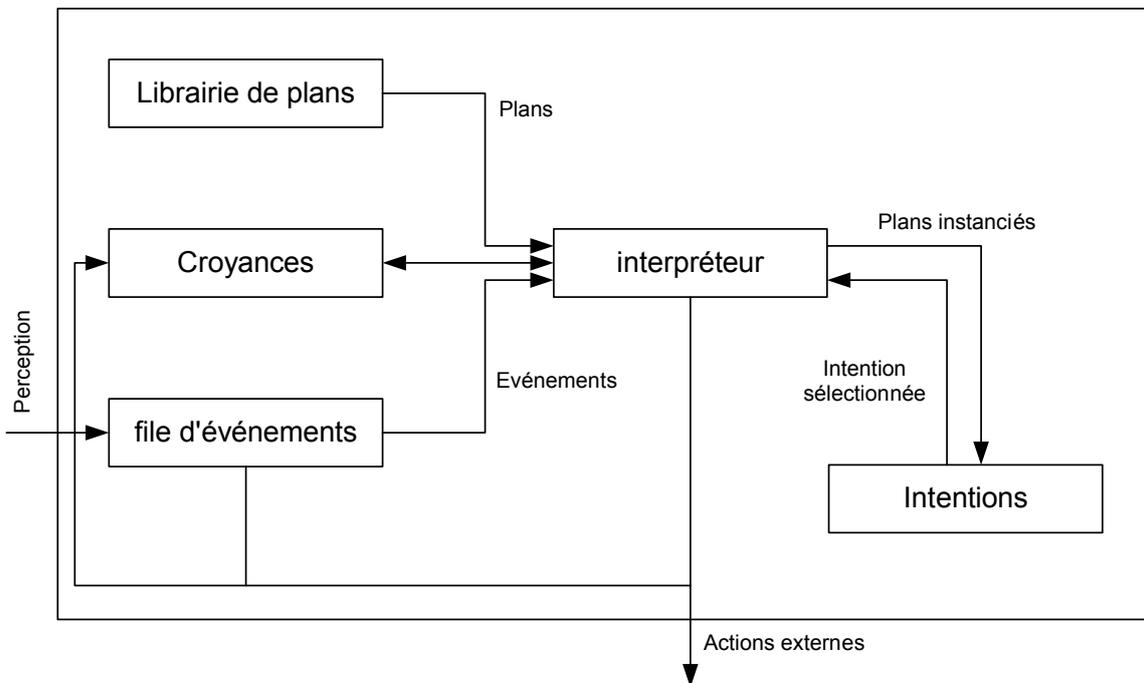


FIG. 2.2 – Architecture BDI de Guerra-Hernandez

peuvent être incorrectes, incomplètes ou incertaines. Les croyances changent au fur et à mesure que l'agent, par sa capacité de perception ou par interaction avec d'autres agents, recueille des informations.

- Les *désirs* (D) ou buts de l'agent, représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs contradictoires. Dans ce cas, il doit choisir parmi ses désirs un sous-ensemble qui soit consistant. Ce sous-ensemble consistant de ses désirs est identifié avec les buts de l'agent. Ils sont représentés comme des plans non encore instanciés pour permettre à l'agent d'atteindre ses buts.
- Les *intentions* (I) d'un agent sont les désirs que l'agent a décidé d'accomplir ou les actions qu'il a décidé de faire pour accomplir ses désirs. Même si tous les désirs d'un agent sont consistants, l'agent peut ne pas être capable d'accomplir tous ses désirs à la fois. Ce sont des piles de plans instanciés.
- Une *file d'événements*<sup>1</sup> stocke les événements (internes ou externes) arrivant dans le système avant qu'ils soient traités par l'agent.
- La *librairie de plans* recense le savoir-faire de l'agent sous la forme de *plans* pré-compilés décrivant les séquences d'actions à effectuer pour réaliser un but.

Le cycle de l'interpréteur BDI fonctionne de la manière suivante :

```
1 B:=B0
2 D:=D0
3 I:=I0
4 répéter
    4.1 obtenir nouvelles perceptions P
    4.2 B:=révision-croyance(B, P)
    4.3 D:=révision-désirs(B, D ,I)
    4.5 I:=filtre-intentions(B, D, I)
    4.6 PE:=plan(B, I)
    4.7 exécuter(PE)
jusqu'à ce que l'agent soit arrêté
```

Dans cet algorithme, B est la pile de croyances, D est la pile de désirs et I la pile d'intentions. Soient B0, D0 et I0 les croyances, désirs et intentions initiaux de l'agent. *révision-croyance* met à jour les croyances de l'agent lorsque celui-ci reçoit de nouvelles perceptions sur l'environnement. P représente l'ensemble des perceptions de l'agent. *révision-désirs* maintient la consistance des désirs en cas de changement dans les croyances de l'agent (on suppose dans ce modèle que l'agent a toujours des désirs consistants). Cette étape utilise la bibliothèque de plans. *filtre-intentions* décide des intentions à poursuivre. *plan* est la fonction qui transforme les plans partiels en plans exécutables, PE étant l'ensemble des plans exécutables.

Le cycle de l'interpréteur BDI commence par actualiser la file d'événements et les croyances de l'agent. Il active alors de nouveaux désirs en sélectionnant les plans de la librairie de plans qui coïncident avec les événements. Un de ces plans est sélectionné pour exécution et une fois instancié, il est placé dans la pile des intentions. Finalement, la première action de la pile d'intentions est exécutée, et ainsi de suite.

---

<sup>1</sup>Queue of events

L'architecture BDI est l'exemple le plus représentatif d'architecture d'agents cognitifs. D'autres architectures que BDI pour les agents cognitifs ont été conçues et utilisées dans diverses applications mais, jusqu'à présent, l'architecture BDI est celle qui est la plus utilisée.

## Architectures TRAINS et TRIPS

Les architectures TRAINS et TRIPS, développées par l'université de Rochester, sont très proches de notre étude car ils traitent de la plupart des thèmes abordés dans cette thèse [Allen *et al.*, 2002], [Allen *et al.*, 2000], [Ferguson et Allen, 1998], [Ferguson *et al.*, 1996] :

- analyse de corpus,
- systèmes dialogiques,
- planification,
- représentation du raisonnement.

Ces quatre axes de recherche sont très liés, puisque les systèmes dialogiques proposés s'appuient sur une analyse de corpus et permettent la planification et la reconnaissance de plans. Sur le long terme, le but est de développer des assistants de planification capables d'interagir en langage naturel.

Le système TRAINS est l'un des premiers systèmes à gérer un dialogue avec un humain, dans le cadre de la résolution collaborative de problèmes. Le projet a permis de constituer plusieurs corpus de dialogues oraux orientés tâche. L'analyse de ces corpus a conduit à la conception de différents systèmes de dialogue appelés TRAINS [Allen *et al.*, 2000], [Ferguson *et al.*, 1996], servant à interroger en langue naturelle une base de données contenant des horaires de trains.

TRIPS (The Rochester Interactive Planning System) est le successeur de TRAINS [Ferguson et Allen, 1998]. Il s'agit encore d'un assistant de planification interagissant avec l'humain, mais TRIPS permet de communiquer par des graphiques (cartes, tableaux, etc.) en plus du langage naturel pour améliorer la compréhension mutuelle.

Allen, Blaylock et Ferguson présentent une architecture d'agent pour la planification collaborative [Allen *et al.*, 2002]. Elle s'articule autour d'un noyau de résolution individuelle de problèmes décrivant le comportement d'un agent autonome isolé, sur lequel se greffent deux niveaux supplémentaires de résolution collaborative et d'interaction (voir FIG. 2.3, tirée de [Allen *et al.*, 2002]).

Le module chargé de la résolution individuelle de problèmes manipule différentes entités.

- La *situation courante*<sup>2</sup> : ensemble des croyances de l'agent.
- Les *objectifs* : buts de l'agent. Ils sont actifs<sup>3</sup> s'ils concernent l'action courante, ou simplement intentionnés<sup>4</sup> s'ils n'ont aucun rapport avec elle.
- Les *recettes intentionnées*<sup>5</sup> : plans qui ont été choisis par l'agent pour atteindre ses objectifs.
- La *bibliothèque de recettes* : ensemble des plans indexés par objectif et situation. La bibliothèque n'est pas statique et peut être étendue par planification, apprentissage, adaptation, etc.

---

<sup>2</sup>intended objectives

<sup>3</sup>active objective(s)

<sup>4</sup>intended objective(s)

<sup>5</sup>intended recipes

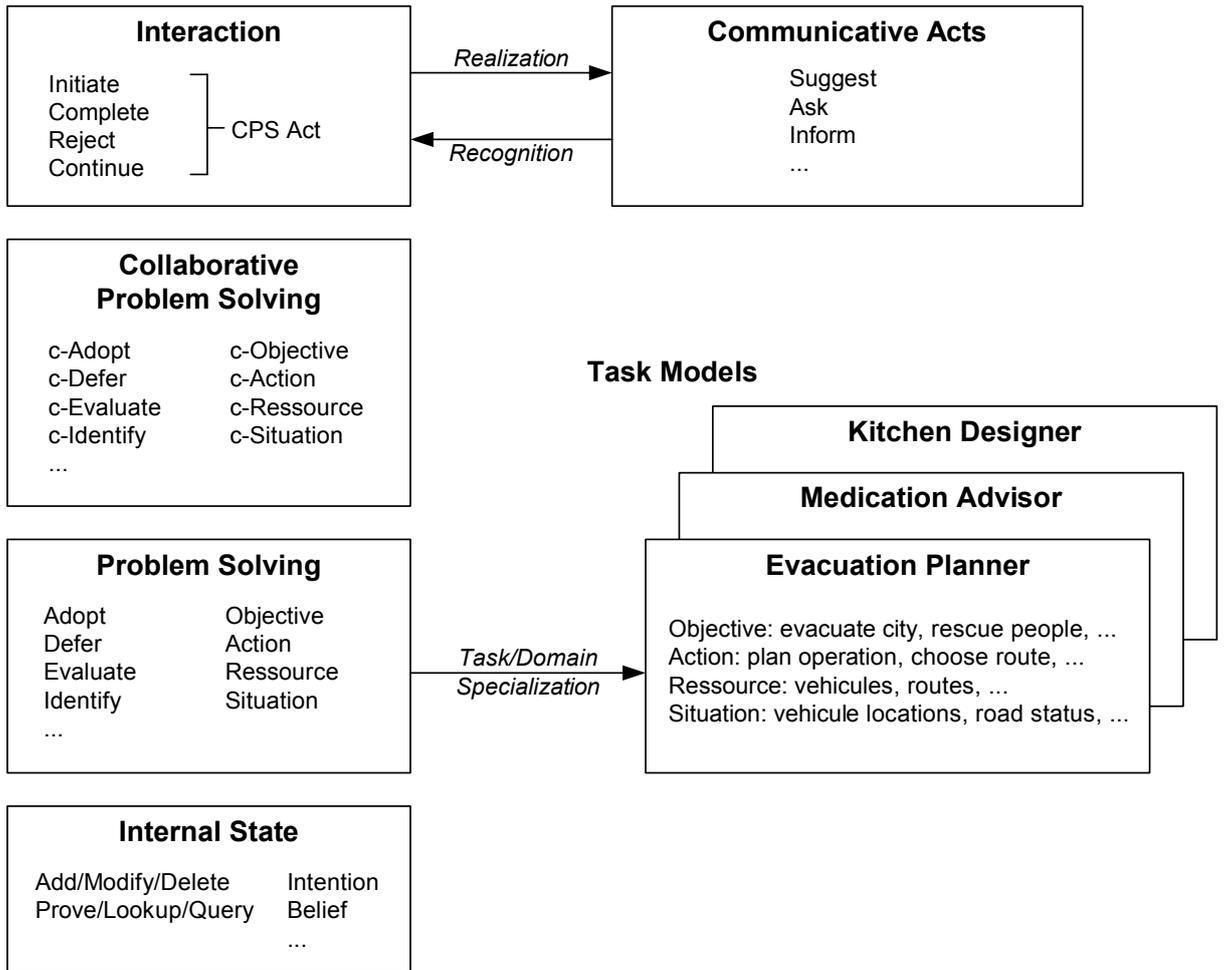


FIG. 2.3 – Architecture d'agents d'Allen, Blaylock et Ferguson

Ces structures de données sont très proches des BDI, sans pour autant être identiques comme le souligne Pollack [Pollack, 1990]. Formellement, une recette est une suite d'actions permettant de réaliser une action plus complexe. Une recette se présente donc sous la forme d'une arborescence appelée graphe de recette (*RGraph*).

Il s'agit d'une architecture générique : le module de résolution de problèmes peut être spécialisé pour décrire n'importe quel modèle de tâches pour un domaine particulier.

Ce module de résolution individuelle de problèmes fonctionne suivant un cycle identique au cycle BDI. Tout d'abord, les objectifs sont déterminés et l'un d'entre eux est activé. Puis, une recette est choisie en fonction de l'objectif actif. Il se peut que la recette ait déjà été invoquée à l'étape précédente sans avoir été réalisée dans son intégralité. Elle est alors réactivée pour être terminée. En suivant la recette active, l'agent peut déterminer quelle est la prochaine action à effectuer. S'il s'agit d'un sous-but, l'agent recommence un processus d'évaluation de ses objectifs. S'il s'agit d'une action élémentaire, elle est effectuée. À ce point, la situation est changée et le cycle reprend à son commencement.

Le module de résolution collaborative de problèmes fonctionne de façon similaire à la résolution individuelle, toute action *conjointe* (faisant intervenir plusieurs agents) pouvant se décomposer en actions individuelles. Il permet de maintenir le lien entre les actions individuelles d'un plan conjoint et les objectifs mutuels. Il génère des actes de résolution collaborative de problèmes que les agents reconnaissent et utilisent pour coordonner leurs actions individuelles.

Le module d'interaction est crucial pour une activité collaborative. Il connecte les actions communicatives réalisées par les agents et les actions de résolution collaborative du problème. Dans le cas d'une communication entre agents logiciels, les actes collaboratifs seront transmis de manière explicite, alors que dans une communication homme-machine en langage naturel, ce module se chargera de la reconnaissance des intentions du locuteur via une inférence de l'acte d'interaction.

Le module de communication utilisé dans cette architecture est similaire à celui de Traum [Traum, 1994]. Il s'agit d'un modèle basé sur la planification dont le but est de reconnaître le plan de l'interlocuteur pour émettre une réponse appropriée. Le modèle de Traum n'est pas détaillé ici mais la section 2.3.1 présente le modèle d'Allen et Perrault [Allen et Perrault, 1980] qui en est très proche.

## 2.1.4 Discussion

D'une façon générale, le Problème de l'Agence de Voyage est un problème de planification multi-agents distribuée, nécessitant coopération entre les agents/sujets<sup>6</sup>.

Les sujets humains peuvent être modélisés au moyen d'agents cognitifs. L'architecture BDI semble être intéressante car elle permet de représenter les états mentaux des agents sous forme de croyances, désirs et intentions. TRAINS et TRIPS, deux architectures d'agent mêlant interaction et planification, utilisent d'ailleurs des principes proches des BDI. Cependant, contrairement à notre approche, TRAINS ne s'appuie pas sur une approche expérimentale pour modéliser la planification humaine.

---

<sup>6</sup>Nous parlons de sujets humains durant l'expérimentation psychologique et d'agents quand il s'agit du modèle (formel ou informatique) d'un sujet.

## 2.2 Un modèle de l'énoncé : la théorie des actes de langage

La théorie des actes de langage est vraisemblablement la théorie dominante pour représenter les unités élémentaires du dialogue au cours d'une conversation. Elle tire sa source des travaux de Reinach [Reinach, 1983] et de Gardiner [Gardiner, 1989], avant d'être formalisée en la théorie classique des actes de langage par Austin [Austin, 1962] puis par Searle [Searle, 1969], [Searle et Vanderveken, 1985] et Vanderveken [Vanderveken, 1988]<sup>7</sup>.

À partir de cette théorie, deux tendances se dégagent, une que suit Vanderveken [Vanderveken, 1999] qui consiste en un prolongement des actes de langage au discours et celle de Sperber et Wilson [Sperber et Wilson, 1989] qui remet en cause certains fondements de la théorie des actes de langage.

### 2.2.1 Fondements de la théorie des actes de langage

Bien qu'Austin soit considéré comme le père fondateur de la théorie des actes de langage, il serait juste de rendre hommage à Reinach, dont les actes sociaux, introduits avant la première guerre mondiale, préfigurent déjà les bases de la théorie des actes de langage.

#### Reinach et les actes sociaux

Selon Reinach [Reinach, 1983], il existe des actes, appelés *actes sociaux*, accomplis uniquement par le langage, et exécutés par le simple fait de dire quelque chose. Ainsi, un acte social :

- est le fait d'un individu *agissant*,
- implique au minimum deux individus différents, le(s) *sujet(s)* et le(s) *récipiendaire(s)*,
- peut être *conditionnel* ou *inconditionnel*,
- peut être accompli par procuration.

La simple énonciation d'une phrase à l'attention d'un interlocuteur, autrement dit la réalisation d'un acte social selon Reinach, crée des obligations et des droits. Ceux-ci s'appliquent au contenu de l'acte social, et s'adressent respectivement au(x) sujet(s) et au(x) récipiendaire(s). La seule condition pour qu'un acte social produise un droit ou une obligation est que le récipiendaire le comprenne. En effet, l'acceptation ou le refus du récipiendaire n'a pas de rôle à jouer dans la création du droit ou de l'obligation pour le sujet.

L'exemple pris le plus souvent par Reinach est celui de la promesse. Si je m'adresse à un ami en lui déclarant : "*Je te promets de venir demain*", il y a création d'un acte social dont le contenu est le fait que je vienne demain. Je me retrouve alors dans l'obligation de venir demain, et mon interlocuteur, le récipiendaire, a le droit de réclamer ma présence.

Cependant, il existe un cas particulier dans lequel l'acceptation ou le refus du récipiendaire a son importance : quand la réponse du récipiendaire fait partie d'une proposition conditionnelle. Par exemple, dans la déclaration : "*Si tu m'accompagnes aujourd'hui, je te promets de venir*

---

<sup>7</sup>La théorie des actes de langage est décrite de façon détaillée par Moeschler dans [Moeschler et Reboul, 1994]. À noter que Reinach (1883-1917) et Gardiner (1879-1963) n'ont vu leurs travaux publiés que fort tard, ce qui explique l'ordre des références bibliographiques.

*demain*", la promesse de venir demain ne crée de droit et d'obligation qu'après acceptation par le récipiendaire.

Un droit ou une obligation ont une durée déterminée, limitée dans le temps par :

- l'accomplissement de l'acte social,
- le renoncement par le récipiendaire à ses droits,
- ou la révocation de l'acte social par le sujet. Cette révocation ne peut se faire que si le sujet possède le pouvoir légal, accordé par le récipiendaire.

## Gardiner

Durant l'entre-deux guerre, Gardiner [Gardiner, 1989] est le premier à introduire le terme d'*acte de langage*. Gardiner subdivise la phrase en deux entités distinctes, le *sujet* et le *prédicat* : "*La prédication consiste à dire quelque chose de quelque chose. Le sujet est la chose dont on parle et le prédicat est ce que l'on en dit.*"

Gardiner se préoccupe plus particulièrement de l'intention du locuteur, qui se reflète dans la structure de la phrase par ses choix de sujet et de prédicat. Cette vision fait ressortir cinq sortes de prédications :

- Tout mot est un prédicat disant quelque chose de l'objet auquel il s'applique. Par exemple, quand une mère dit "*Mon fils*" en désignant son fils.
- Toute phrase en tant que réaction à un stimulus est un prédicat sur ce stimulus. La phrase "*Il pleut*" en est un bon exemple.
- Tout mot employé est un prédicat qui qualifie les mots précédents, comme le mot vite dans "*il marche vite*".
- Dans les phrases de type *sujet+prédicat*, le prédicat dit quelque chose du sujet, comme "*Marie est belle*".
- Tout mot dans une phrase peut être employé de façon prédicative pour dire implicitement quelque chose de l'ensemble de la phrase. La phrase peut alors être considérée comme la contradiction d'une autre opinion. Un exemple est l'énonciation de "*Cette maison est horrible*", avec l'accent mis sur *horrible*. Horrible qualifie, en sus de la maison, l'ensemble de la phrase.

On pourra remarquer que seuls les deux derniers types sont véritablement des prédications intentionnelles de la part du locuteur. Selon Gardiner, le locuteur n'a pas seulement l'intention de communiquer quelque chose, il a aussi un but ultérieur qui n'est pas visible au moment de l'énonciation. La distinction sujet/prédicat doit donc être réservée au niveau du discours. Gardiner propose alors de se limiter à quatre types de phrase permettant de livrer l'intention du locuteur à un niveau où l'intention peut être indiquée dans la phrase elle-même :

- Les *déclarations* : il s'agit de dire quelque chose de quelque chose, d'accomplir une *assertion* qui peut être positive (une *affirmation*) ou négative (une *dénégation*). Dans les déclarations, il est souvent difficile de faire ressortir les intentions du locuteur.
- Les *questions* : la réussite de l'acte dépend de l'existence ou non d'une réponse verbale pertinente de l'interlocuteur. Les questions appelant collaboration (questions fermées auxquelles l'interlocuteur doit répondre par oui ou par non) sont à distinguer de celles nécessitant spécifications (questions ouvertes auxquelles l'interlocuteur doit répondre sur un point particulier).

- Les *requêtes* : la réussite d'une action ne dépend pas uniquement du locuteur mais aussi de la volonté de l'interlocuteur de satisfaire cette requête. Les ordres, les supplications, les prières, les conseils sont des exemples de requête.
- Les *exclamations* : le sujet est le locuteur qui exprime spontanément son humeur ou un désir.

## 2.2.2 Théorie Austinienne des actes de langage

À la suite de ces travaux, dans les années 1950, Austin [Austin, 1962] remarque qu'il existe :

- des énoncés descriptifs, dits *constatatifs*, qui sont susceptibles d'être vrais ou faux,
- des énoncés non-descriptifs dits *performatifs*, qui peuvent être heureux ou malheureux.

L'illusion descriptive est ici remise en cause. Ainsi, l'énoncé :

*"Je baptise ce bateau le Queen Elizabeth."*

est performatif, alors que la prédication suivante :

*"Cette maison est jolie."*

est constatative.

Cependant, l'énonciation d'un performatif ne suffit pas toujours à l'accomplissement de l'action correspondante, il est nécessaire que les circonstances de cette énonciation soient appropriées. Si elles ne le sont pas, l'acte a *échoué*, et l'énoncé est dit malheureux. Pour qu'un énoncé performatif soit heureux, il faut qu'il n'y ait, ni ambiguïté, ni emploi indu du performatif, ni déféctuosité, ni accroc lors de l'exécution de l'acte. De plus, il faut que le locuteur de l'énoncé soit sincère. Austin ajoute à cela deux conditions très générales de succès : "*Il faut que le locuteur ait eu un interlocuteur*" et que cet "*interlocuteur ait compris l'acte et qu'il ait reconnu de quel acte il s'agit*".

La distinction entre énoncé constatatif et énoncé performatif ne se résume pas à la distinction entre dire et faire. Dire qu'un énoncé performatif est heureux, si certaines conditions sont remplies, revient à dire qu'il est heureux si les affirmations qui décrivent ces conditions sont vraies. La vérité/fausseté peut donc toucher les performatifs et le bonheur/malheur peut toucher les constatatifs.

Austin souligne "*que le locuteur de l'énoncé, du fait de son énonciation, accomplit un acte. Un énoncé performatif doit donc pouvoir se ramener à un énoncé comportant un verbe à la première personne du singulier de l'indicatif présent, voix active*".

Le problème se trouve donc déplacé vers la distinction entre énoncés performatifs explicites et implicites. Elle ne peut se faire qu'à l'aide de dispositifs linguistiques ou paralinguistiques particuliers (mode, intonation, adverbes, connecteurs, gestes, circonstances de l'énonciation...). De plus, certains énoncés performatifs explicites peuvent toujours être confondus avec des énoncés constatatifs, à cause de formules ambiguës.

Pour différencier aisément les énoncés performatifs et les énoncés constatatifs, Austin s'appuie sur l'action réalisée par le locuteur lors de l'énonciation d'une phrase. Cette action, baptisée *acte de langage*, se divise en trois actes différents :

- un acte *locutoire*, acte physique d'énonciation d'une phrase appelée *proposition*,
- un acte *illocutoire* qui traduit les intentions du locuteur envers son(ses) interlocuteur(s) (l'acte illocutoire est réalisé en produisant l'énoncé),
- un acte *perlocutoire* qui correspond aux effets réels produits sur l'interlocuteur. Ils peuvent être différents (mais pas nécessairement) de ceux prévus par le locuteur.

Par exemple, si j'ordonne à quelqu'un de lever les mains, l'énonciation de la proposition "Haut les mains" sera l'acte locutoire, cette énonciation produit un acte illocutoire d'ordre et l'acte perlocutoire éventuel sera le fait que l'interlocuteur lève les mains. Un acte locutoire correspond donc à un énoncé, l'acte illocutoire à l'intention que le locuteur place dans cette énonciation et l'acte perlocutoire à ses conséquences.

Cette séparation en trois actes permet de différencier l'information transmise de la manière dont elle est transmise. Ainsi, les énoncés "Allons-nous en !" et "Nous allons partir" portent sur la même proposition (les interlocuteurs vont partir) mais accomplissent un acte illocutoire différent (un ordre et une assertion).

On peut remarquer que dans une affirmation, le locuteur fait un acte illocutoire au même titre que lorsqu'il avertit ou qu'il déclare. Cependant, bien que les énoncés performatifs tout à la fois disent et fassent quelque chose, ils ne sont pas vrais ou faux comme les constatatifs.

### 2.2.3 Formalisation de la théorie : Searle et Vanderveken

Searle poursuit les travaux d'Austin [Searle, 1969]. Il rend la théorie des actes de langage indissociable des théories de l'action et de l'intention. La distinction des énoncés performatifs en trois actes se formalise alors à l'aide de la notation fonctionnelle :  $F(p)$ , où  $p$  représente la proposition et  $F$  la force illocutoire appliquée à cette proposition.

De plus, il ajoute le principe d'exprimabilité, selon lequel *tout ce que l'on veut dire peut être dit*. Cela sous-entend que la force illocutoire d'un énoncé peut toujours être rendue explicite à l'aide d'un verbe performatif.

La reconnaissance du type d'un acte de langage par l'interlocuteur est essentielle. Searle a donc classé les différents types de force illocutoire selon la taxinomie suivante :

- les *descriptifs / assertifs / représentatifs* servent à décrire des faits,
- les *directifs* visent à faire faire quelque chose à l'interlocuteur,
- les *engageants / commissifs / promissifs* obligent le locuteur à réaliser une action,
- les *déclaratifs / déclarations* rendent vrais des faits par leur simple énonciation (par exemple : "*Je vous déclare unis par les liens sacrés du mariage*"),
- les *expressifs* expriment l'état d'esprit du locuteur.

Cette taxinomie tient compte du but illocutoire de chaque type d'acte. Lors d'une énonciation, un locuteur peut avoir de multiples buts perlocutoires mais il a au moins l'intention d'accomplir le but illocutoire de son acte.

L'exhaustivité de cette taxinomie est justifiée par la couverture des quatre directions d'ajustement entre les mots (la proposition) et le monde.

- Pour les descriptifs, les mots s'ajustent au monde.
- Pour les engageants et les directifs, le monde s'ajuste aux mots.
- Pour les déclaratifs, l'énoncé produit un changement du monde rendant la direction d'ajus-

tement à double sens : le monde s'ajuste au contenu propositionnel par le biais de l'énonciation elle-même.

- Pour les expressifs, la direction d'ajustement est vide car l'énoncé et le monde sont indépendants.

Searle définit le *mode d'accomplissement* d'un acte de langage comme la manière d'atteindre le but illocutoire. Dans la langue naturelle, cela se traduit souvent par l'utilisation d'un adverbe. Il précise aussi qu'un locuteur doit respecter des *conditions préparatoires* (dans le cas d'un conseil, le locuteur cherche à aider son interlocuteur) et des *conditions de sincérité*. De plus, la notation fonctionnelle implique que certaines forces illocutoires imposent des conditions sur les contenus propositionnels qui pourraient leur être associés. Par exemple, la proposition associée à un directif doit porter sur des actions futures.

La performance d'un acte de langage est défini par :

- Sa *réussite* (ou *succès*) : les conditions de succès d'un acte de langage sont que le locuteur le réalise avec la bonne force illocutoire et le bon mode d'accomplissement, en respectant les contraintes portant sur la proposition, les conditions préparatoires et les conditions de sincérité.
- Sa *satisfaction* : le monde est transformé conformément aux attentes du locuteur, le contenu propositionnel de l'acte devenant vrai, dans la direction d'ajustement propre à son but illocutoire.

Par exemple, dans le cas de la promesse (un acte engageant), il y aura succès de l'acte s'il est reconnu comme un engageant, et satisfaction de l'acte si le locuteur accomplit bien les actions qu'il a promis de faire.

L'intérêt de la communauté informatique pour la théorie des actes de langage s'est accru quand celle-ci s'est dotée de la logique illocutoire [Searle et Vanderveken, 1985], [Vanderveken, 1988]. Elle vise à formaliser la satisfaction et la réussite d'un acte illocutoire en fonction de la valeur de vérité de son contenu propositionnel. Elle se présente sous une forme récursive rendant compte des relations d'implication entre actes de langage. Par exemple, si une déclaration est réussie, l'assertion de son contenu propositionnel devient vrai : le succès de "Je déclare la séance ouverte" (acte  $a_1$ ) entraîne la satisfaction de l'assertion "La séance est ouverte" (acte  $a_2$ ). En logique illocutoire, cette dépendance s'écrira : *si success( $a_1$ ) alors satisfaction( $a_2$ )*. Vanderveken décrit une sémantique sous la forme d'un langage formel de représentation des actes illocutoires et d'une logique illocutoire du premier ordre [Vanderveken, 1990 1991].

L'introduction de la logique illocutoire dans la théorie des actes de langage permet, par ailleurs, de décrire le mécanisme des *actes illocutoires indirects*. Un acte indirect est un acte illocutoire non littéral, accompli au moyen d'un autre acte qui, lui, est littéral. Quand je dis "Il y a un courant d'air !", la plupart du temps je ne cherche pas seulement à décrire le monde, mais j'essaie de pousser mon interlocuteur à modifier le monde en fermant la fenêtre. Il s'agit là d'un exemple d'acte indirect. L'assertion initiale a permis à l'interlocuteur, grâce au sens commun, de déduire qu'il s'agissait d'une requête indirecte visant à fermer la fenêtre. Lorsque l'interlocuteur ferme la fenêtre, la requête indirecte est réussie et satisfaite, de même que l'assertion initiale grâce à cette requête indirecte.

Searle et Vanderveken ont donc poursuivi les travaux d'Austin sur la théorie des actes de langage, en considérant qu'un énoncé, dès lors qu'il exprime une phrase grammaticale complète, correspond à l'accomplissement d'un acte illocutoire. Ce qui est primordial dans l'interprétation d'un énoncé, ce n'est donc pas la vérité ou la fausseté de la proposition mais l'acte

illocutoire qu'il accomplit. Une même proposition peut être utilisée dans des énoncés différents et accomplir des actes illocutoires différents.

## 2.2.4 Théorie des actes de langage et théorie de la pertinence

Les travaux de Sperber et Wilson [Sperber et Wilson, 1989] sur la pertinence s'appuient sur la notion de coopération définie par Grice [Grice, 1975]. Selon Grice, pour permettre l'échange d'information lors d'un dialogue, les interlocuteurs doivent se conformer au *principe de coopération*, qui se décline sous la forme de maximes (les "maximes de Grice"). Leur transgression va amorcer un processus d'inférence appelé *implicature* pour retrouver le véritable sens des énoncés.

Sperber et Wilson ne conservent parmi les maximes de Grice que celle portant sur la pertinence et l'érige en principe : *le principe de présomption optimale de pertinence*. Dans leur théorie, la pertinence d'un énoncé est évaluée en fonction du rapport entre les effets qu'il produit et les efforts que la production de ces effets demande.

De plus, Sperber et Wilson reviennent sur la théorie classique des actes de langage et critiquent plus particulièrement :

- son caractère anti-vériconditionnaliste, c'est-à-dire qu'elle ne combine pas l'accomplissement des actes et la vérité des propositions,
- le fait que la récupération de l'intention du locuteur est une des conditions de succès de l'acte,
- son aspect conventionnel.

Ces trois remarques portent sur la taxinomie de Searle, pivot de la théorie des actes de langage. Ce n'est pas directement cette classification que Sperber et Wilson remettent en cause, pas plus que sa nécessité au sein de la théorie des actes de langage, mais plutôt le besoin qu'elle introduit, de déterminer l'acte illocutoire accompli pour interpréter un énoncé. Il y aurait, selon eux, deux types d'actes de langage : ceux qui doivent être identifiés par le locuteur et l'interlocuteur pour être accomplis, et ceux qui peuvent être accomplis sans une telle identification.

Le premier groupe contient les actes qualifiés d'institutionnels, nécessitant bien souvent des circonstances bien déterminées pour être accomplis avec succès (comme un statut ou l'appartenance à tel ou tel corps). Le second groupe rassemble plutôt des actes non institutionnels et relève de la linguistique ou de la pragmatique, puisque l'identification des actes n'est pas indispensable pour qu'ils soient accomplis.

À ces deux grandes catégories d'actes de langage, Sperber et Wilson en ajoutent une troisième contenant uniquement les trois actes *dire que*, *dire de* et *demander si*. Cette nouvelle catégorie fait disparaître aussi bien la correspondance terme à terme entre la forme syntaxique et l'acte accompli, que la typologie traditionnelle des actes de langage avec ses conditions de sincérité notamment. Ces trois actes fondamentaux sont ainsi définis :

- "**Dire que** *P*, où *P* est la forme propositionnelle de l'énoncé, c'est communiquer que la pensée exprimée par *P* est entretenue comme une description d'un état de choses réelles".
- "**Dire à l'interlocuteur de** *P*, c'est communiquer que la pensée exprimée par *P* est entretenue comme un état de choses désiré."
- "**Demander si**, c'est communiquer que la pensée exprimée par *P* serait pertinente si elle était vraie dans le cas d'une question fermée, et communiquer qu'il y a une certaine façon

de compléter la pensée exprimée par  $P$  qui la rendrait pertinente si elle était vraie dans le cas d'une question ouverte."

Sperber et Wilson réduisent la taxinomie des actes de langage de la théorie classique, à trois actes de base, qui ne sont pas institutionnels et doivent être identifiés comme tels pour que l'interprétation de l'énoncé soit satisfaisante. Par ailleurs, leur théorie est une théorie *vériconditionnelle* qui combine l'accomplissement des actes avec la vérité des propositions.

## 2.2.5 Sémantique des actes de langage

Il existe de nombreux travaux qui visent à compléter la théorie des actes de langage. Ils s'appuient souvent sur la logique illocutoire de Searle et Vanderveken [Searle et Vanderveken, 1985], [Vanderveken, 1988] et sur une sémantique des actes de langage fournie par Vanderveken [Vanderveken, 1990 1991].

Ces sémantiques visent à représenter l'accomplissement d'un acte de langage au travers de l'engagement illocutoire du locuteur et à décrire formellement la structure des communications intentionnelles. Elles ont en commun d'être basées sur le calcul des situations [McCarthy et Hayes, 1969].

### Apport de Cohen et Levesque

Cohen et Levesque proposent une relecture de la théorie des actes de langage dans laquelle l'accent est mis sur les intentions des acteurs d'une conversation [Cohen et Levesque, 1990b]. Ces intentions sont dérivées de la théorie de l'intention de Bratman [Bratman, 1987]. Ils utilisent une logique modale pour décrire les effets d'un énoncé sur les interlocuteurs [Cohen et Levesque, 1990a]. Cette syntaxe (voir TAB. 2.1) contient des opérateurs pour représenter les attitudes propositionnelles (croyances et buts) et distinguer la séquentialité des actions et des événements.

Prédicat	Description
$(BEL\ x\ p)$	$p$ est une croyance de $x$
$(GOAL\ x\ p)$	$p$ est un but de $x$
$(HAPPENS\ a)$	l'action $a$ sera la prochaine réalisée
$(DONE\ a)$	l'action $a$ vient juste d'être réalisée
$(DECLARATIVE\ s)$	$s$ est un déclaratif
$DECLARATIVE \Rightarrow (BEL\ spkr(TRUE\ s\ e))$	l'énonciation du déclaratif $s$ implique que le locuteur $spkr$ croit l'énoncé vrai en ce qui concerne la suite d'événements $e$ qui vient d'avoir lieu

TAB. 2.1 – Syntaxe de Cohen et Levesque

Grâce à la notion de *but persistant* (but particulier qui ne peut disparaître qu'avec la croyance que ce but est réalisé ou qu'il est impossible) marquant l'engagement d'un agent pour ce but, Cohen et Levesque définissent une *intention* comme un but persistant que l'agent s'engage à réaliser lui-même. Un acte illocutoire est vu comme une *tentative*, de la part du locuteur, pour

réaliser une intention via l'énonciation de cet acte. Cette tentative porte sur n'importe quel type de performative, que ce soit une demande ou une assertion.

La théorie de Cohen et Levesque est un pas important dans la formalisation de la théorie des actes de langage. Cependant, comme le souligne Singh, leur approche soulève quelques problèmes [Singh, 1992]. Elle produit certains résultats contraire à l'intuition : par exemple, si  $p$  est une proposition peu plausible (mais pas impossible) et si  $b$  est le but persistant portant sur  $p$ , d'un agent compétent à son propos,  $b$  sera nécessairement vrai. De plus, Cette approche emprunte quelques raccourcis de construction. Ainsi,

$$(INTEND\ x\ a_1; a_2) \models (INTEND\ x\ a_1)$$

signifie que si un agent  $x$  a l'intention de réaliser une action  $a_2$  après une action  $a_1$ , il a nécessairement l'intention d'effectuer  $a_1$ . Par ailleurs, cette sémantique est conçue pour des agents n'ayant qu'une seule intention à la fois et elle impose une unique politique de révision des intentions pour tous les agents.

## Apport de Singh

Singh distingue trois formes différentes de satisfaction pour les actes de langage [Singh, 1991], [Singh, 1993] :

- la satisfaction *annexe*<sup>8</sup> (*ESAT*) si sa proposition devient vraie, même fortuitement,
- la satisfaction *sans réserve*<sup>9</sup> (*WSAT*) si sa proposition est rendue vraie d'une manière connue et intentionnelle,
- la satisfaction *significative*<sup>10</sup> (*RSAT*) si la proposition devient vraie grâce à la production de l'acte de langage.

La différence entre la satisfaction sans réserve et la satisfaction significative est ténue mais bien réelle. Dans le cas d'un directif, si l'interlocuteur avait de toute façon l'intention de réaliser l'action rendant vraie la proposition transmise par le locuteur, la satisfaction sera sans réserve alors qu'elle sera significative si cette action a été forcée par l'acte de langage.

Le modèle formel est représenté par un arbre des mondes possibles. Le langage formel de description du modèle utilise une logique temporelle et modale à laquelle Singh adjoint, entre autres, un prédicat d'intention (*intends*) et un prédicat de communication (*comm*). La sémantique des formules est donnée relativement à un modèle  $M$  et à une date  $t$  dans ce modèle.  $M \models_t p$  signifie  $M$  satisfait  $p$  à la date  $t$ .

Avec ce type de représentation, Singh décrit les trois modes de satisfaction pour différents types d'acte de langage. Par exemple, voici les conditions de satisfaction pour un assertif :

$$(M \models_{s,t} WSAT(comm(x, y, < assertive, p >))) \text{ iff} \\ (\exists t_e : < S, t, t_e > \in \parallel saysto(x, y, < assertive, p >) \parallel^x \wedge M \models_{s,t_e} p)$$

<sup>8</sup>extensional satisfaction

<sup>9</sup>whole-hearted satisfaction

<sup>10</sup>relevant satisfaction

En résumé, un assertif est satisfait sans réserve si sa proposition est vraie au moment où l'énoncé est émis.

### Apport de Dignum et Weigand

Dignum et Weigand s'appuient à la fois sur la logique illocutoire et sur la logique déontique [Meyer, 1988] pour modéliser les conversations [Dignum et Weigand, 1995]. De la première, ils ne retiennent que les actes primitifs de demande (*DIR*), promesse (*COM*), affirmation (*ASS*) et déclaration (*DECL*) et la seconde leur permet de définir les notions d'*obligation* ( $O_{ij}$ ), d'*interdiction* ( $F_{ij}$ ) et de *permission* ( $P_{ij}$ ) pour une action :

$$O_{ij}(\alpha(i)) = [\overline{\alpha(i)}]Violation_{ij}$$

$$F_{ij}(\alpha(i)) = [\alpha(i)]Violation_{ij} = O_{ij}(\overline{\alpha(i)})$$

$$P_{ij}(\alpha(i)) = \neg[\alpha(i)]Violation_{ij} = \neg F_{ij}(\alpha(i))$$

où  $Violation_{ij}$  est un prédicat spécial indiquant la violation d'un accord passé entre  $i$  et  $j$ . Ainsi, pour l'obligation, l'agent  $i$  est obligé d'effectuer  $\alpha(i)$  sinon il y aura violation de l'accord.

Dans cette syntaxe, le mode d'accomplissement d'un acte de langage est précisé à l'aide d'un des trois indices possibles de charité<sup>11</sup> (c), pouvoir<sup>12</sup> (p) ou autorisation<sup>13</sup> (a). Par exemple,  $DIR_c$  sera une requête alors que  $DIR_a$  sera une requête autorisée (i.e. un ordre légal) et  $DIR_p$  sera un ordre s'appuyant sur une relation de domination.

Tous ces éléments permettent de construire un langage d'actions, les actes de langage étant considérés comme des actions. Pour un acte de langage  $\alpha$ , les conditions préparatoires  $\phi$  et les effets prévus  $\psi$  sont modélisés par le schéma suivant :  $\phi \rightarrow [\alpha]\psi$ .

Axiome	Description
$[COM(i, j, \alpha)]O_{ij}(\alpha)$	À l'émission d'un engageant (commis-sif) appliqué à l'action $\alpha$ , $i$ est dans l'obligation de réaliser $\alpha$ .
$([DIR_p(i, j, \alpha)]O_{ij}(\alpha)) \leftarrow j <_{\alpha} i$	L'émission d'un directif sur $\alpha$ , est conditionnée par l'existence d'une relation de puissance entre $i$ et $j$
$([ASS_a(i, j, \phi)]B(j, \phi)) \leftarrow auth(i, ASS(i, j, \phi))$	À l'émission d'un assertif, pour que l'interlocuteur ajoute $\phi$ à ses croyances, il faut que le locuteur soit autorisé à faire cette assertion

TAB. 2.2 – Syntaxe de Dignum et Weigand

De plus, ils fournissent un protocole de communication appelé CoLa (Communication Language) (voir 2.4.4).

<sup>11</sup>charity

<sup>12</sup>power

<sup>13</sup>authorization

## 2.2.6 Discussion

La théorie des actes de langage ainsi que les théories qui en découlent sont reconnues depuis longtemps dans la communauté scientifique pour leur capacité à modéliser les communications. Elle est bien formalisée par la logique illocutoire de Searle et Vanderveken et grâce à cette formalisation elle a souvent été utilisée pour des applications en interaction (voir 2.4).

Un des reproches qui lui est souvent fait, est d'être une théorie de l'énoncé et non du dialogue. Elle ne tient pas compte du caractère dynamique des discours et des dialogues, tant au niveau de l'enchaînement conversationnel, que des relations existant entre les protagonistes.

Pour améliorer cela, il est nécessaire de considérer les conversations dans leur ensemble et non comme une simple suite d'énonciations isolées. Cependant, les actes illocutoires pouvant être considérés comme des unités de conversation, ils servent de pierre angulaire à de nombreux modèles conversationnels.

## 2.3 Modèles de dialogue

Les actes de langage sont rarement utilisés seuls et leur articulation forme la communication qui peut être synchrone (communication orale) ou asynchrone (courriers électroniques, forums de discussion, etc.). Dans les communications, on distingue le discours mené par un unique locuteur (monologue), le dialogue impliquant deux locuteurs et la conversation réalisée au moins par deux intervenants. Compte tenu des caractéristiques du Problème de l'Agence de Voyage, cette section est consacrée aux modèles de dialogue.

Comme le souligne Pasquier [Pasquier, 2004], ce qui différencie le dialogue du discours, c'est la recherche d'une inter-compréhension. Elle nécessite entre autres choses :

- un langage commun,
- la capacité à détecter les ambiguïtés et les incohérences chez l'interlocuteur,
- la possibilité de vérifier la bonne interprétation des différents actes de communication,
- l'existence d'échanges correctifs,
- la capacité à méta-communiquer (communiquer sur le dialogue lui-même).

Le processus de l'inter-compréhension est complexe et ne résume pas à un simple assemblage d'actes de langage isolés. Par exemple, le graphe d'une simple requête par Winograd et Flores est constitué de plusieurs actes impliquant le demandeur et son interlocuteur [Winograd et Flores, 1986] (voir FIG. 2.4, tirée de [Winograd et Flores, 1986]). Winograd et Flores représentent une requête par un automate à états finis dont les transitions correspondent aux échanges d'acte de langage. Une demande est vue comme une action commune complexe. Sa signification globale est sans commune mesure avec le sens isolé de chacun des actes de langage qui la compose. La modélisation des dialogues n'est donc pas une tâche aisée.

Il existe de nombreux modèles de dialogue dont la majorité utilise les actes de langage comme unité minimale des conversations.

Lehuen distingue dans [Lehuen, 1997] quatre approches de modélisation du dialogue : la *négociation*, la *planification*, la *structuration* et l'*interaction*.

L'approche par la négociation tente de modéliser la recherche d'accords au cours des conver-

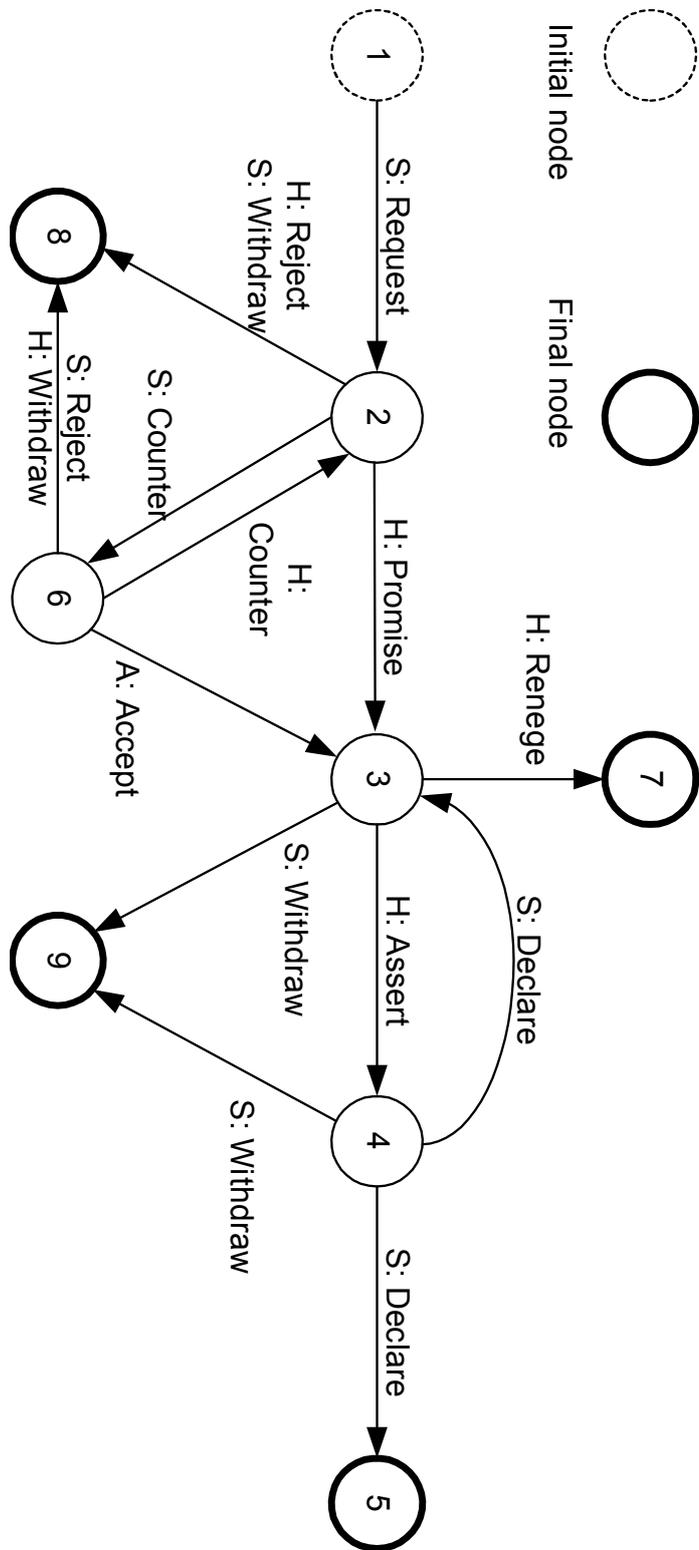


FIG. 2.4 – Graphe d’une requête par Winograd et Flores

sations. L'approche par la planification est fondée sur la reconnaissance de buts et la construction de plans dans les conversations. L'approche par la structuration cherche à dégager et à expliquer la structure des dialogues. Enfin, l'approche par l'interaction s'appuie sur l'interprétation et la production d'actions communicatives durant les dialogues.

Les trois dernières approches sont présentées ici. Elles sont plus proches des besoins rencontrés dans le Projet de l'Agence de Voyage. Un modèle fondé sur la convention et quelques modèles mixtes sont aussi décrits.

### 2.3.1 Modèles fondés sur la planification

Les modèles fondés sur la planification, comme toutes les approches intentionnelles, ont pour idée directrice que la structure du dialogue résulte des intentions des interlocuteurs. Cette idée a été généralisée par Grosz et Sidner qui décomposent le dialogue en trois éléments constitutifs : la *structure linguistique* (les énoncés agrégés en segments de dialogue), la *structure intentionnelle* (ensemble des buts des segments de dialogue : ce sont des sous-buts du but global du dialogue) et l'*état attentionnel* (focus dynamique du dialogue) [Grosz et Sidner, 1986]. Selon eux, la structure linguistique est induite par la structure intentionnelle. En particulier les entrelacements de sous-dialogues traduisent une imbrication de sous-buts.

Pour formaliser la structure intentionnelle, c'est tout naturellement une approche par planification qui s'est imposée. Les modèles de dialogue fondés sur la planification s'appuient sur la capacité du locuteur à reconnaître les plans de son interlocuteur, et à les utiliser pour décider de la réponse la plus pertinente à lui transmettre.

Parmi ces modèles, celui d'Allen et Perrault fait partie des plus aboutis. Il s'appuie sur les notions de *croyances* décrites dans une logique modale et de *schémas d'actions* qui permettent de formaliser les actes de langage et les actions [Allen et Perrault, 1980].

Les schémas d'actions (voir TAB. 2.3) sont représentés grâce au formalisme STRIPS [Fikes et Nilsson, 1971]. Ce sont des règles contenant un nom, un ensemble de paramètres et un ensemble de formules comportant chacune des *pré-conditions*, des *effets* et un *corps*. Le corps décrit l'action sous la forme d'une séquence de sous-actions. Dans l'exemple donné, *L* est le locuteur, *I* l'interlocuteur et *P* la proposition.

<b>INFORMER(L, I, P)</b>	
<i>Préconditions :</i>	L VEUT INFORMER(L, I, P) L SAIT P
<i>Effets :</i>	I SAIT P
<i>Corps :</i>	L DIRE P à I

TAB. 2.3 – Exemple de schéma d'action

Dans ce formalisme, une *occurrence d'action* est un schéma d'action instancié et indexé par le temps. Un *plan* est une séquence linéaire d'occurrences d'action liant un état initial et un état final du monde.

Avec ce modèle, la reconnaissance de plan peut-être considérée comme un processus de planification inversée. Le système construit un plan en partant d'une action pour remonter au but qui l'a motivée. Ceci permet au système de :

- déterminer les buts non implicites de l'interlocuteur,
- prédire ses actions futures,
- anticiper les difficultés pouvant empêcher la réalisation de ses actions.

Pour inférer le plan de l'interlocuteur, Allen et Perrault différencient les règles concernant les *actions* de celles concernant les *connaissances*. L'application successive de ces règles sur les connaissances permet de reconnaître le plan du locuteur.

Par exemple,

$$(I \text{ CROIT } (L \text{ VEUT } ACT)) \Rightarrow (I \text{ CROIT } (L \text{ VEUT } E))$$

est une règle d'inférence concernant une action *ACT* qui signifie que si *L* veut réaliser *ACT*, c'est peut-être qu'il souhaite obtenir l'effet *E*.

De même, la règle d'inférence

$$(I \text{ CROIT } (L \text{ SAITSI } P)) \Rightarrow (I \text{ CROIT } (L \text{ VEUT } P))$$

est une règle concernant les connaissances, telle que si *L* veut savoir si *P* est vrai, il souhaite que *P* soit vrai.

Une fois que le système a inféré tous les buts de son interlocuteur, il peut alors construire un plan permettant d'aider à la réalisation de ce but. La plupart des règles permettant la construction de ce plan sont les règles inverses de celles qui ont servi à la reconnaissance du plan de l'interlocuteur.

Par exemple,

$$(I \text{ VEUT } ACT) \Rightarrow (I \text{ VEUT } P)$$

signifie que si *I* veut réaliser l'action *ACT*, il faut d'abord obtenir la pré-condition *P*.

La reconnaissance de plans au cours d'une conversation permet la construction d'une réponse pertinente. Supposons la réception de l'énoncé "Quand le train de Lyon part-il?". Cette question laisse supposer que le locuteur désire prendre le train de Lyon. Pour cela, il devra se trouver à la gare à la bonne heure, au bon quai. C'est ce type de plan qui peut être inféré à partir de la question initiale. Or, dans cette question, il manque une demande d'information concernant le quai. Pour résoudre ce problème, il suffit de construire un plan dans lequel l'interlocuteur participe en informant le locuteur de l'heure du train et du numéro du quai. Il lui fournira la réponse pertinente suivante : "À 13 heures, quai n°7".

Les modèles fondés sur la planification ont surtout été appliquées aux IHM, dans un cadre mono-agent. Le modèle développé par Balkanski et Hurault-Plantet utilise aussi la reconnaissance et l'élaboration de plans pour comprendre et accomplir des actes communicatifs dans le contexte d'un répondeur téléphonique [Balkanski et Hurault-Plantet, 2000]. Le système TRAINS et son successeur TRIPS sont des systèmes servant à interroger en langue naturelle une base de données [Allen *et al.*, 2000], [Ferguson et Allen, 1998], [Ferguson *et al.*, 1996] (voir 2.1.3). Ces systèmes trouvent leurs fondements dans l'étude approfondie d'un corpus de messages. D'autres systèmes sont plus axés sur le raisonnement et utilisent les plans via les approches logiques comme le système Artimis [Sadek *et al.*, 1997].

## 2.3.2 Modèles fondés sur la structuration

La modélisation des dialogues fondés sur la structuration s'appuie sur la décomposition de l'interaction verbale en unités de différents niveaux. Il s'agit alors de modéliser les règles conversationnelles régissant ces unités.

### Le modèle genevois

Le modèle genevois de Roulet et Moeschler cherche à rendre compte *a posteriori* de la structure des conversations humaines réelles [Roulet *et al.*, 1985]. Il identifie les constituants du discours selon une analyse hiérarchique et une analyse fonctionnelle. Il définit les liens entre ces constituants.

L'analyse hiérarchique décompose l'interaction verbale selon quatre rangs, de l'unité la plus simple à la plus complexe.

- L'*acte de langage* est l'unité minimale des conversations (voir 2.2).
- L'*intervention* est l'articulation supérieure aux actes de langage. Elle peut être *simple* si elle ne contient qu'un seul acte de langage ou *composée* si elle en contient plusieurs.
- L'*échange* est la plus petite unité dialogale. Il est constitué de une à trois interventions, une intervention *initiative* à laquelle on peut adjoindre une intervention *réactive* et éventuellement d'une intervention *évaluative*.
- L'*incursion* est délimitée par la rencontre et la séparation des deux interlocuteurs correspondant au niveau de description le plus général.

Cette terminologie, n'est pas unique. Elle est affinée par Kerbrat-Orechioni qui renomme l'*incursion* en *interaction* et la subdivise en *séquences* [Kerbrat-Orechioni, 1990]. La séquence est constituée de blocs d'échanges reliés par un fort degré de cohérence sémantique et/ou pragmatique.

De plus, Moeschler dote le modèle genevois d'une grammaire générative permettant de développer, grâce à un processus récursif les micro-structures hiérarchiques [Moeschler, 1989].

Le modèle genevois ne se limite pas à la structuration des discours. L'analyse fonctionnelle permet de relier les différents constituants de la structure hiérarchique par des liens fonctionnels illocutoires ou interactifs. Ces fonctions permettent par exemple d'isoler la fonction illocutoire initiative des interventions, qui donne des droits ou impose des contraintes à l'interlocuteur, de la fonction illocutoire réactive, qui renvoie à une intervention antérieure.

C'est l'affectation d'une fonction illocutoire ou interactive qui permet de vérifier la complétude interactionnelle ou interactive du constituant discursif. Ces notions de complétude interactionnelle et interactive sont à mettre en parallèle avec les notions de satisfaction et de réussite de la logique illocutoire.

### La théorie discursive de Vanderveken

Vanderveken étend la théorie des actes de langage aux conversations et aux discours, et dégage ainsi une logique du discours [Vanderveken, 1999]. Il introduit comme unités de base du raisonnement les *états mentaux* et reprend une partie des travaux de l'école genevoise en

décomposant les conversations en *actes illocutoires*, *interventions* et *échanges*. La logique du discours distingue une hiérarchie d'unités structurée sur trois niveaux.

Pour Vanderveken, l'intention discursive dirige les conversations. Les interlocuteurs cherchant à atteindre des buts, même si ces buts sont extra-linguistiques. Il définit alors quatre types de conversation :

- les *conversations à but descriptif* : elles visent à décrire le monde (ajustement des mots aux choses).
- les *conversations à but délibératif* : elles servent à délibérer sur les actions futures que les interlocuteurs s'engagent à faire (ajustement des choses aux mots).
- les *conversations à but déclaratoire* : elles cherchent à transformer le monde par des déclarations (double direction d'ajustement).
- les *conversations à but expressif* : elles expriment les états d'âme et les attitudes des interlocuteurs (direction vide d'ajustement).

Cette typologie est à mettre en parallèle avec celle proposée par Walton et Krabbe, dont les cinq types principaux sont aussi caractérisés par le but global du dialogue [Walton et Krabbe, 1995]. Ils distinguent la *persuasion*, la *négociation*, l'*investigation* (ou *enquête*), la *délibération* et la *recherche d'informations*.

La classification de Vanderveken n'est pas sans rappeler la taxinomie des actes de langage donnée par Searle. Néanmoins, les conversations sont bien plus que de simples séquences finies d'actes de langage. Elles possèdent une structure et des conditions de félicité qui leur sont propres et qui sont étroitement liées à la dynamique du groupe des protagonistes, ainsi qu'à un arrière-plan contextuel.

Par exemple, il se peut que l'interlocuteur d'un acte illocutoire comprenne le sens littéral de cet acte mais ne sache pas s'il doit considérer un fait particulier de l'arrière plan. Pour lever cette ambiguïté, il est amené à demander des précisions au locuteur sur les conditions de l'énoncé s'il ne veut pas prendre de décision en connaissances incomplètes.

De plus, un discours peut faire des digressions, c'est-à-dire contenir des échanges n'étant pas en rapport direct avec le but discursif. Il exige donc l'accomplissement d'un nombre fini d'actes illocutoires essentiels de forme déterminée, appelés *actes-mâtres*. La satisfaction d'un discours dépendant évidemment de la satisfaction des actes maîtres, il est difficile d'attribuer une valeur absolue de satisfaction aux discours. Vanderveken propose alors une logique floue de la satisfaction des discours.

Nous signalons par ailleurs que l'intention discursive n'est pas la seule théorie permettant de décrire les motivations d'une conversation. Par exemple, Pasquier [Pasquier, 2005] utilise pour cela la théorie de la dissonance cognitive. Selon cette théorie, deux cognitions (perception, attitude propositionnelle, etc.) liées peuvent être *consonantes* (l'une entraîne ou supporte l'autre) ou *dissonantes* (l'une entraîne ou supporte le contraire de l'autre). Dans ce cadre, les interactions sont initiées pour réduire une dissonance cognitive, ce qui s'apparente à atteindre un but pour nous. Par exemple, si j'ai besoin d'utiliser une information qui ne m'est pas connue, il y aura dissonance cognitive que j'essaierai de réduire en initiant un dialogue de demande d'information.

### 2.3.3 Modèles fondés sur l'interaction

Bien qu'au cours d'une conversation, il ne soit pas facile d'explicitier l'intention discursive d'un locuteur, les modèles du dialogue fondés sur l'interaction permettent de définir un principe interactionnel. Il s'agit d'une règle locale d'analyse ou de production d'une action communicative, dont les conséquences sont observables dans des corpus de dialogue.

#### Le modèle de Trognon et Brassac

Le modèle de Trognon et Brassac se situe à la jonction des deux cadres théoriques que sont la *logique illocutoire* (voir 2.2) et l'*analyse du discours conversationnel* (voir 2.3.2) [Trognon et Brassac, 1992]. Trognon part d'une similitude entre les notions de *réussite non défectueuse* et de *satisfaction* d'une part, et la *complétude interactive* et la *complétude interactionnelle* d'autre part, pour définir un *principe d'enchaînement conversationnel*.

Le point central de ce modèle est la construction d'une interprétation commune des énoncés, par le biais du jeu croisé des conditions de satisfaction d'un acte de langage, relativement à la réussite d'un autre acte de langage avec lequel il est en relation.

Le principe d'enchaînement conversationnel de ce modèle s'articule en trois temps : production d'un énoncé, interprétation de cet énoncé par l'interlocuteur et validation ou non de cette interprétation.

Par ailleurs, Trognon montre que ce principe d'enchaînement permet d'engendrer une structure hiérarchique de dialogue similaire à celle de l'école genevoise [Trognon, 1995].

#### Le modèle de Brassac et Pesty

Le modèle de Trognon et Brassac est repris par Brassac et Pesty [Brassac et Pesty, 1999]. Leur logique interlocutoire consiste en une dialogisation de la sémantique générale des actes de langage.

Les deux auteurs ont adopté une méthodologie de modélisation cognitive et cherchent à :

- "**observer** la conversation [...],
- **modéliser** les mécanismes de perlaboration [...],
- **valider** le modèle des mécanismes de l'interaction langagière [...]."

Dans ce modèle, un même énoncé peut prendre des sens différents selon la situation (le moment) et le sens co-construit peut différer selon les locuteurs. Ainsi, une *stabilisation* de la valeur illocutoire d'un énoncé doit être atteinte.

La principale différence avec le modèle précédent est qu'il ne s'appuie plus sur le principe de la *communicativité* d'un énoncé (tout énoncé est traité par l'auditeur) mais sur celui de la *communicabilité* (l'intention de sens est une production conjointe).

Ce modèle a fait l'objet d'une implémentation sous la forme d'un logiciel de conversation simulant une prise de rendez-vous entre deux agents logiciels.

## Le modèle de Nicolle et Saint-Dizier de Almeida

Le modèle de Nicolle et Saint-Dizier de Almeida fait aussi état d'une co-construction du sens des actes de langage, avec la notion de *valeur interlocutoire* résultant d'une négociation entre les protagonistes de la conversation sur la force illocutoire d'un énoncé [Nicolle et Almeida, 1999].

La force illocutoire dépend toujours de la direction d'ajustement entre les mots et le monde qui est décomposé en monde *physique, social, du soi* et *sémiotique*<sup>14</sup>. Ainsi, le modèle ne distingue plus que quatre forces illocutoires primitives (*assertive, déclarative, directive* et *commissive*).

Les intervenants d'une conversation sont contraints à posséder en leur sein une représentation récursive d'eux-même, de l'autre et d'un terrain commun, dont les niveaux supérieurs ne sont déployés qu'en cas de désaccord.

Un acte illocutoire devient alors *réussi* s'il lui est attribué une valeur interlocutoire et la notion de *satisfaction* est alors rattachée à la valeur interlocutoire.

Le principal défaut de ce modèle est que son implantation paraît difficile.

### 2.3.4 Modèles fondés sur les conventions

Un bon exemple de modèle basé sur les conventions sont les jeux de dialogue [Maudet, 2001].

Les jeux de dialogue tirent sans doute leur essence des systèmes dialectiques initiés par Hamblin [Hamblin, 1970]. Un système dialectique est un système normatif de régulation du dialogue constitué d'un *ensemble de coups*, d'une *liste d'engagements par participant*, d'un *ensemble de règles pour les coups* et d'un *ensemble de règle pour les engagements*. Il faut noter que les systèmes dialectiques ne traitent pas de la production des énoncés, ni du lien entre les états mentaux des agents et leur activité dialogique.

En comparaison, les jeux de dialogue modélisent le comportement des agents lors du dialogue. Ils permettent de spécifier le passage d'un jeu à un autre (conditions de changement de jeux) et le traitement des coups et des états mentaux (règles des jeux).

Il faut noter que les jeux peuvent être imbriqués durant le dialogue. Les agents peuvent donc jouer plusieurs jeux en même temps, ce qui permet de décomposer des problèmes en sous-problèmes dans un dialogue coopératif.

Le changement de jeu est caractérisé par trois types de condition :

- la *condition d'entrée* est l'état activant un jeu ;
- l'*invariant* est l'état qui doit être vérifié tout au long du jeu ;
- la *condition de sortie* est l'état terminant un jeu.

Dans la suite  $\mathcal{C}$  désigne l'ensemble des coups et  $\mathcal{EM}$  celui des états mentaux. Le comportement des agents relativement à un jeu est défini par quatre types de règles :

1. Les *règles de Perception* ( $\mathcal{C} \longrightarrow \mathcal{EM}$ ) mettent en relation les coups joués avec les états

---

<sup>14</sup>le monde des mots

mentaux de l'agent. Elles associent à un coup ou une séquence de coups une formule. Lors de la reconnaissance d'un coup associé à une règle de perception, les états mentaux de la règle sont utilisés pour la mise à jour de la base de connaissances courante. Ces règles sont appliquées après la génération d'un coup.

2. Les *règles de mise à jour des états mentaux* ( $\mathcal{EM} \rightarrow \mathcal{EM}$ ) décrivent le fonctionnement interne de l'agent. À partir d'une expression portant sur les états mentaux courants, la règle produit une combinaison d'états mentaux chez l'agent. Son comportement est le même que pour les règles de perception.
3. Les *règles de génération* servent à produire des coups en fonction des états mentaux de l'agent. La partie gauche d'une règle de génération est une expression portant sur les états mentaux courants de l'agent. La partie droite est le coup généré. Il se peut qu'aucune règle de génération ne puisse être appliquée. Dans ce cas, on considère que l'agent "ne sait pas" quoi dire (c'est différent du coup qui permet à l'agent de terminer le dialogue). C'est un blanc dans la conversation, un "coup vide".
4. Les *règles d'interdiction* servent à filtrer les coups générés. Les règles de génération produisent un ensemble de coups candidats à la génération. Les règles d'interdiction permettent d'éliminer certains de ces coups. Elles correspondent par exemple à des maximes de Grice [Grice, 1975].

Les *règles de perception* permettent de mémoriser les coups. Ensuite, les *règles de mise à jour* travaillent, en parallèle avec les *règles d'inférence* de l'agent pour transformer ses états mentaux. Les règles de génération et d'interdiction construisent un ensemble de coups générables et non-interdits. Finalement, le coup réellement joué sera choisi par une heuristique basée sur une force attribuée aux états mentaux.

Les jeux de dialogue apportent un formalisme plus flexible que celui des automates à états finis. De plus, ils permettent de garder une trace plus riche du dialogue que le simple dernier acte communicatif considéré. Enfin ils permettent de gérer les messages inattendus. Cependant, il leur manque sans doute un petit travail de formalisation, d'implémentation et de validation avant de supplanter totalement les protocoles classiques.

### 2.3.5 Modèles mixtes

Il existe différents modèles qui sont fondés sur plusieurs approches à la fois. Par exemples, le modèle de Luzzati, le modèle COALA de Lehuen ou le modèle de Lemeunier sont fondés sur la structuration et sur l'interaction [Lehuen, 1997], [Lemeunier, 2000], [Luzzati, 1995].

Ces trois modèles visent à modéliser les dialogues finalisés. Ils partent d'une remarque importante de Luzzati.

*Le dialogue peut s'orienter dans deux directions : soit demande d'information et délivrance des renseignements s'enchaînent sans difficulté, et il s'agit d'un dialogue régissant ; soit les demandes de précision, d'explication, de confirmation ou de re-formulation doivent intervenir pour qu'une question ou une réponse soit acceptée, et il s'agit d'un dialogue incident.*

Comme le souligne Lemeunier, les *attentes* d'un système, qui lui permettent de rester sur l'axe du dialogue régissant, sont liées à son activité applicative. La difficulté provient de la

gestion des *non-attendus* : ils sont levés à l'aide d'hypothèses faites sur des attentes structurées hiérarchiquement selon leur fréquence d'apparition.

Ces trois modèles cherchent donc à structurer les conversations en dialogues régissants et en dialogues incidents. Cependant, ils conservent une dimension interactionnelle par la gestion des *attendus* et des *non-attendus* qui sont vus comme des règles locales permettant de produire et interpréter des actions communicatives.

### 2.3.6 Discussion

Tous les modèles de dialogue présentés dans cette section utilisent la théorie des actes de langage pour représenter un énoncé unique. De plus, quelle que soit l'approche adoptée pour modéliser le dialogue (modèles fondés sur la planification, la structuration ou l'interaction), elle reste indissociable des théories de l'action et de l'intention. En effet, toutes les approches tendent à faire ressortir une intention discursive.

La théorie discursive de Vanderveken nous paraît être la plus intéressante car elle étend directement la théorie des actes de langage au discours. Elle présente, de plus, une structuration simple des dialogues en interventions et permet de donner une valeur de satisfaction à un sous-dialogue d'une conversation selon le statut de l'intention discursive [Vanderveken, 1999].

## 2.4 ACL et protocoles de communication

Les implantations en machine des théories précédentes sont nombreuses<sup>15</sup>. Dans cette section, seules les approches les plus significatives sont développées. Que ce soit KQML [Finin *et al.*, 1994], [Labrou et Finin, 1997], FIPA-ACL [FIPA, 2000a], KQML+ [Bouzouba et Moulin, 2001] ou bien CoLa [Dignum et Weigand, 1995], ces approches ont toutes en commun de s'appuyer sur un système de performatives et des protocoles d'interaction.

Pour être exhaustif, il aurait fallu intégrer par exemple MAGMA ou encore COOL. MAGMA est un protocole d'interaction développé par Populaire et Demazeau sous la forme d'un graphe d'états qui permet de gérer un comportement social [Populaire *et al.*, 1993]. COOL est un langage de coordination entre agents conçu par Barbuceanu et Fox [Barbuceanu et Fox, 1995]. Ils utilisent des messages fondés sur les actes de langage et des protocoles de coordination pour les SMA.

### 2.4.1 KQML

Premier apparu parmi les ACL, le langage KQML (Knowledge Query and Manipulation Language) est considéré comme un standard. Il est structuré selon trois couches imbriquées [Finin *et al.*, 1994], [Labrou et Finin, 1997].

1. La *couche de communication* renseigne sur l'identité de l'émetteur et du destinataire et sur la nature de la communication. Cette couche est minimale car KQML ne prend pas en

---

<sup>15</sup>Koning et Pesty présentent les principaux modèles de communication utilisés dans les SMA [Koning et Pesty, 2001]

charge le transport lui-même.

2. La *couche message* précise les informations nécessaires à la compréhension du contenu du message. Elle définit le type d'interaction que les agents pourront avoir.
3. La *couche contenu* est le message en lui-même. KQML ne traite pas cette couche. Il précise seulement où le contenu du message commence et se termine. Si le contenu est opaque pour le destinataire, ce dernier doit rechercher la signification du contenu dans la couche message.

KQML est basé sur la théorie des actes de langage. Il propose une encapsulation des messages dans une performative qui définit l'acte illocutoire. Un message KQML contient alors les paramètres suivants :

- l'*émetteur*,
- le *destinataire*,
- un *identifiant* unique,
- une éventuelle *référence* à un message précédent,
- le *langage de représentation* utilisé par le système,
- l'*ontologie* du système,
- le *contenu* ou *corps* du message.

Les quatre premiers paramètres appartiennent à la *couche de communication*, les deux suivants appartiennent à la *couche message* et le dernier paramètre appartient à la *couche contenu*.

KQML est muni d'une seule règle de conversation : une conversation commence au moment où le premier message est envoyé par un agent à un autre agent et se termine quand ce dernier répond. Pour chaque message, les actes de langage autorisés sont restreints aux performatives. KQML fournit un ensemble de performatives (voir TAB. 2.4) divisé en trois catégories. Dix-huit performatives de discours permettent l'échange d'informations et de connaissances. Sept performatives de régulation des conversations traitent les erreurs et les variantes à la règle de conversation. Enfin, onze performatives d'assistance et de réseau étendent les possibilités de communication à plus de deux agents (non données ici). Pour la description des performatives, *S* est l'émetteur, *R* est le destinataire et *P* est le contenu du message.

KQML est issu d'un projet de la DARPA, le KSE (Knowledge Sharing Effort), initialement prévu comme moyen d'échange d'informations entre programmes à base de connaissance [Finin et Fritzon, 1994]. Sa structure orientée message et la généralité de ses primitives lui permettent d'être utilisé comme langage de communication entre agents. Cependant, il n'est pas exempt de tout reproche. Parmi les imperfections soulevées par Cohen et Levesque et Bouzouba et Moulin, on peut remarquer que la définition du sens de certaines performatives est ambiguë ou vague, que certaines performatives sont mal identifiées et que d'autres manquent [Bouzouba et Moulin, 2001], [Cohen et Levesque, 1995].

Une partie de ces problèmes a été supprimée avec l'adjonction à KQML de différentes sémantiques de ses performatives, en termes de pré-/post-conditions pour le locuteur et l'interlocuteur. Particulièrement, la sémantique proposée par Labrou est basée sur une logique multimodale sophistiquée qui contraint les agents à être conçus selon une architecture BDI [Cohen et Levesque, 1990b], [Finin et Labrou, 1998], [Labrou, 1996], [Labrou et Finin, 1994].

<b>performative</b>	<b>type d'acte</b>	<b>description</b>
ask-if	directif	<i>S</i> veut savoir si <i>P</i> est dans la base de connaissances de <i>R</i> .
ask-all	directif	<i>S</i> veut connaître toutes les instances de <i>P</i> qui soient vraies pour <i>R</i> .
ask-one	directif	<i>S</i> veut connaître une des instances de <i>P</i> qui soit vraie pour <i>R</i> .
stream-all	directif	version multi-réponses de ask-all.
eos	déclaratif	marqueur de fin pour un stream-all.
tell	assertif	<i>P</i> est dans la base de connaissances de <i>S</i> .
untell	assertif	<i>P</i> n'est pas dans la base de connaissances de <i>S</i> .
deny	assertif	la négation de <i>P</i> est dans la base de connaissances de <i>S</i> .
insert	directif	<i>S</i> veut que <i>R</i> ajoute <i>P</i> à sa base de connaissances.
uninsert	directif	<i>S</i> veut que <i>R</i> annule un précédent insert.
delete-one	directif	<i>S</i> veut que <i>R</i> supprime de sa base de connaissances une des instances qui correspond à <i>P</i> .
delete-all	directif	<i>S</i> veut que <i>R</i> supprime de sa base de connaissances toutes les instances qui correspondent à <i>P</i> .
undelete	directif	<i>S</i> veut que <i>R</i> annule un précédent delete-one ou un précédent delete-all.
achieve	directif	<i>S</i> veut que <i>R</i> essaie de rendre <i>P</i> vrai.
unachieve	directif	<i>S</i> veut que <i>R</i> annule un précédent achieve.
advertise	déclaratif	<i>S</i> veut que <i>R</i> sache que <i>S</i> peut envoyer un message de la forme de <i>P</i> .
unadvertise	déclaratif	<i>S</i> veut que <i>R</i> sache que <i>S</i> ne peut plus envoyer de message de la forme de <i>P</i> .
subscribe	directif	<i>S</i> veut que <i>R</i> le prévienne si des changements au(x) réponse(s) d'une de ses demandes interviennent.
error	assertif	<i>S</i> considère que le précédent message de <i>R</i> est erroné.
sorry	assertif	<i>S</i> comprend le précédent message de <i>R</i> mais ne peut pas le satisfaire.
standby	assertif	<i>S</i> informe <i>R</i> qu'il doit attendre qu'il soit prêt avant de lui envoyer la réponse à <i>P</i> .
ready	engageant	<i>S</i> indique à <i>R</i> qu'il a une réponse et qu'il est prêt à l'envoyer à <i>R</i> dès que celui-ci le prévendra.
next	directif	<i>S</i> veut la réponse suivante à une précédente demande qu'il a envoyée à <i>R</i> .
rest	directif	<i>S</i> veut l'ensemble des réponses manquantes à une précédente demande qu'il a envoyée à <i>R</i> .
discard	directif	<i>S</i> ne veut pas de la fin des réponses à une précédente demande qu'il a envoyée à <i>R</i> .

TAB. 2.4 – Liste des performatives de KQML

## 2.4.2 FIPA-ACL

La collaboration internationale au sein de FIPA (Foundation for Intelligent Physical Agents) a permis de spécifier certains standards pour les SMA, notamment un langage standard de communication entre agents, FIPA-ACL [FIPA, 2000a]. Comme KQML, ce dernier est basé sur la théorie des actes de langage : les messages sont vus comme des actes communicatifs car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Les spécifications de FIPA-ACL se composent d'un ensemble de types de message et de leur sémantique (les effets de chaque message sur les états mentaux des interlocuteurs).

FIPA-ACL s'appuie sur différents langages de représentation des connaissances pour décrire le contenu des messages. Par exemple, KIF (Knowledge Interchange Format) possède une sémantique déclarative (il n'a pas besoin d'interpréteur pour comprendre les expressions générées). Il permet de représenter des méta-connaissances en introduisant de nouvelles structures de représentation de connaissances, sans changer le langage [FIPA, 2000b].

Chaque message est considéré comme une action communicative. FIPA-ACL est doté d'une sémantique basée sur celle de Sadek [Sadek, 1991], [Sadek, 1992]. Sa forme actuelle définit des pré-conditions de faisabilité des messages et des post-conditions décrivant les effets rationnels attendus sur les états mentaux des interlocuteurs [Labrou *et al.*, 1999]. Cette sémantique s'appuie sur le langage logique SL (Semantic Language) qui décrit les états mentaux des interlocuteurs comme des *croyances* ( $B$ ), des *désirs* ( $D$ ), des *croyances incertaines* ( $U$ ) et des *intentions* (buts persistants  $PG$ ) [FIPA, 2002b].

Par exemple, la sémantique associée au précédent message est la suivante dans SL :

$\langle i, \text{inform}(j, \phi) \rangle$

FP:  $B_i\phi \wedge \neg B_i(B_i f_j \phi \vee U_i f_j \phi)$

RE:  $B_j\phi$

Dans cette syntaxe, FP sont les pré-conditions et RE les effets à l'envoi pour le locuteur et à la réception pour l'interlocuteur (ici, il n'y a que des effets sur l'interlocuteur). Les  $B_x$  sont des croyances pour l'agent  $x$  et  $U_i f_x$  est une incertitude pour  $x$ . Pour un `inform`, les pré-conditions sont que l'émetteur croit en la proposition qu'il souhaite envoyer et que le destinataire est incertain à ce sujet. À la réception de ce message, le destinataire ajoute le contenu du message à ses croyances.

La sémantique de FIPA-ACL est, comme celle de KQML, alignée sur un modèle de type BDI. Cela signifie qu'elle repose sur une logique multimodale combinée à une théorie de l'action. Par conséquent, seuls les agents conçus dans un style BDI peuvent utiliser FIPA-ACL de façon cohérente.

Au niveau de la syntaxe, FIPA-ACL est aussi très proche de KQML : seul un petit nombre de performatives portent un nom différent dans les deux langages. Ainsi, il maintient l'approche de KQML qui distingue le langage externe et le langage interne. Le langage externe définit la signification prévue du message alors que le langage interne ou le contenu, dénote l'expression à laquelle s'appliquent les croyances, les désirs, et les intentions des interlocuteurs.

FIPA-ACL contient des actes de communication permettant les demandes et transmissions d'information, les négociations, les demandes d'action et la gestion des erreurs. Ces actes de communication peuvent s'utiliser comme *primitives* ou bien être *combinés*.

La liste des performatives de FIPA-ACL est donnée avec une brève description pour chacun d'eux (voir TAB. 2.5).

FIPA fournit également la description normative d'un ensemble de protocoles d'interaction de haut niveau, y compris la demande d'action (voir FIG. 2.5, tirée de [FIPA, 2002a]), l'établissement de contrat (contract-net) et plusieurs types de ventes aux enchères.

Ce protocole fixe l'ordre et le type des communications qui seront échangées entre les agents lors d'une demande d'action. Une demande d'action (`request`) est suivie par une acceptation (`agree`) ou un refus (`refuse`) de la part de l'interlocuteur. En cas d'acceptation, l'interlocuteur notifiera le demandeur du résultat de sa tentative pour effectuer l'action demandée. En cas d'échec, il enverra un message correspondant (`failure`). En cas de réussite, il enverra un message indiquant le résultat de son action ou simplement le fait que l'action a bien été réalisée (`inform`).

### 2.4.3 KQML+

Bouzouba et Moulin font ressortir quatre défauts présents dans KQML, mettant en avant une application perfectible de la théorie des actes de langage [Bouzouba et Moulin, 2001].

1. Les performatifs de KQML se limitent à des assertifs et à des directifs.
2. Le choix des performatifs n'est pas toujours approprié (par exemple un *sorry* n'a rien à voir avec un verbe alors que les performatifs doivent être des verbes).
3. La force illocutoire et le contenu propositionnel des messages de KQML sont indépendants alors qu'ils ne le sont pas dans la théorie des actes de langage.
4. KQML ne permet pas de représenter certaines formulations du langage naturel comme les actes de langage indirects.

Bouzouba et Moulin proposent alors d'étendre KQML vers KQML+ de façon à être plus conforme à la théorie des actes de langage. KQML+ contient huit types de performatives (voir TAB. 2.6).

La forme de ces performatives ressemblent à celle de KQML mais contient quelques paramètres en plus :

(Primitive-performative	
~:sender	?
~:receiver	?
~:expressed-performative	(verbe, degré)
~:expressed-social-relation	?
~:presentation	(verbe/expression, degré)
~:content	contenu propositionnel
~:reply-with	?
~:in-reply-to	?)

Le paramètre `expressed-performative` comprend le verbe performatif réellement exprimé dans la formulation et le degré associé à ce performatif. Ainsi, pour une demande polie, ce paramètre aura la valeur (`demande, polie`). Le paramètre `expressed-social-relation` permet d'exprimer la position sociale, de façon à la différencier du degré associé

<b>performative</b>	<b>type d'acte</b>	<b>description</b>
accept-proposal	directif	<i>S</i> accepte une proposition d'action envoyée précédemment.
agree	engageant	<i>S</i> accepte de réaliser une action demandée.
cancel	engageant	<i>S</i> informe qu'il n'a plus besoin qu'un autre agent réalise une action demandée précédemment.
call-for-proposal	directif	<i>S</i> fait un appel d'offre pour une action.
confirm	directif	<i>S</i> informe qu'une proposition est vraie (ce type de message est envoyé quand <i>S</i> pense que son interlocuteur est incertain à propos de la valeur de la proposition).
disconfirm	directif	<i>S</i> informe qu'une proposition est fausse.
failure	assertif	<i>S</i> informe qu'il a essayé de réaliser un action mais que cette tentative a échoué.
inform	assertif	<i>S</i> informe de la valeur d'un proposition.
inform-if	assertif	<i>S</i> informe qu'une proposition est vraie.
inform-ref	assertif	<i>S</i> informe <i>R</i> de la valeur d'un objet.
not-understood	assertif	<i>S</i> informe <i>R</i> qu'il n'a pas compris une des actions (éventuellement communicative) de <i>R</i> .
propagate	directif	<i>S</i> demande à <i>R</i> de propager un message à un ensemble d'agents défini par <i>S</i> .
propose	engageant	<i>S</i> propose de réaliser une certaine action.
proxy	directif	<i>S</i> demande à <i>R</i> de propager un message à un ensemble d'agents sélectionnés par <i>R</i> .
query-if	directif	<i>S</i> demande si une proposition est vraie ou non.
query-ref	directif	<i>S</i> demande la valeur d'un objet.
refuse	engageant	<i>S</i> refuse de réaliser une action.
reject-proposal	directif	<i>S</i> refuse une proposition d'action.
request	directif	<i>S</i> demande à <i>R</i> de réaliser une action.
request-when	directif	<i>S</i> demande à <i>R</i> de réaliser une action quand une certaine proposition deviendra vraie.
request-whenever	directif	<i>S</i> demande à <i>R</i> de réaliser une action à chaque fois qu'une certaine proposition deviendra vraie.
subscribe	directif	<i>S</i> demande à <i>R</i> de le prévenir des changements de valeur d'un objet.

TAB. 2.5 – Liste des performatives de FIPA-ACL

<i>Performatif</i>	<i>Signification</i>
direct	pour un directif
assert	pour un assertif
declare	pour un déclaratif
commit	pour un engageant
express	pour un expressif
question	pour un interrogatif
accept	pour une acceptation
refuse	pour un refus

TAB. 2.6 – Performatives de KQML+

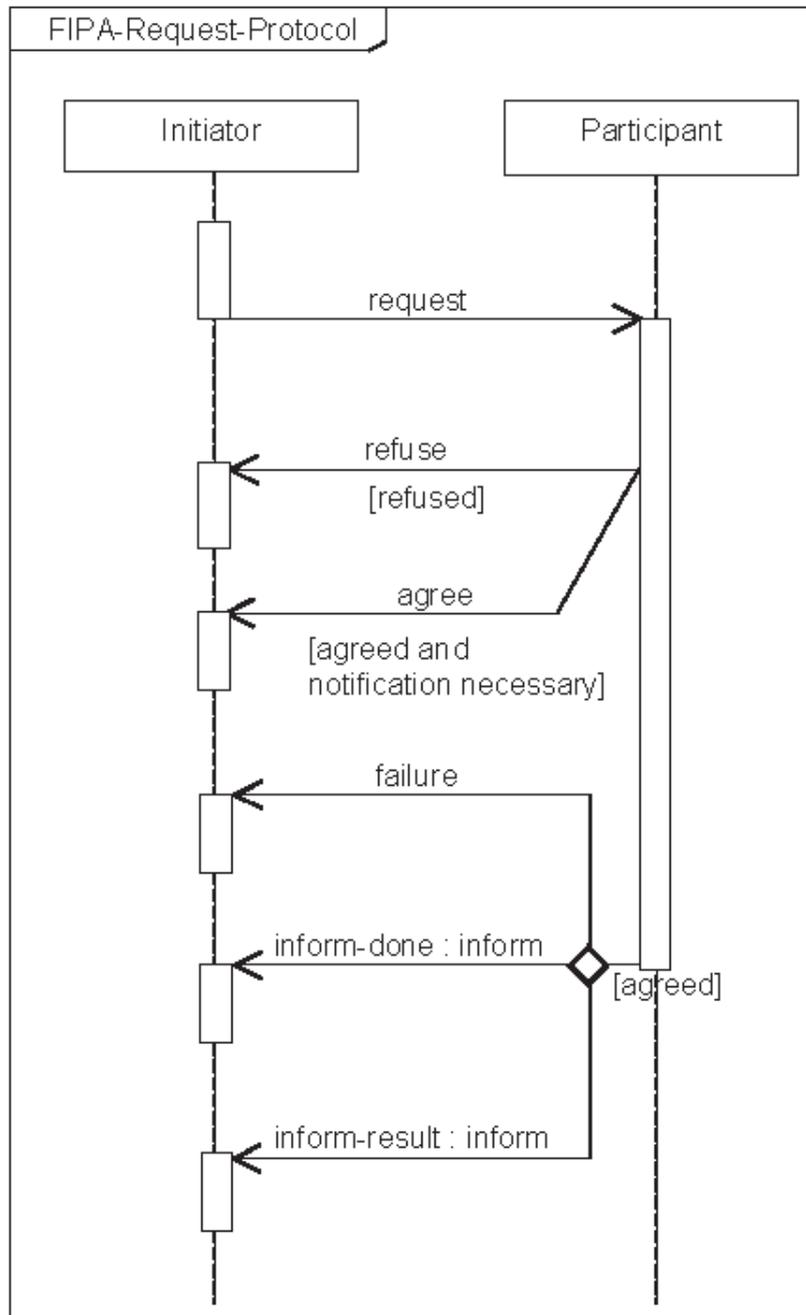


FIG. 2.5 – Protocole FIPA : Request

au performatif. Enfin, certaines attitudes mentales du locuteur peuvent être exprimées grâce au paramètre `presentation`, qui a un rôle de renforcement (positif ou négatif) de l'acte que le locuteur veut accomplir. Par exemple, l'expression "C'est avec grand plaisir que..." sera représentée dans ce paramètre avec la valeur (`plaisir, grand`).

Pour permettre aux agents d'interagir, Bouzouba et Moulin définissent un protocole de communication intermédiaire, c'est-à-dire non figé à la conception (voir FIG. 2.6, tirée de [Bouzouba et Moulin, 2001]).

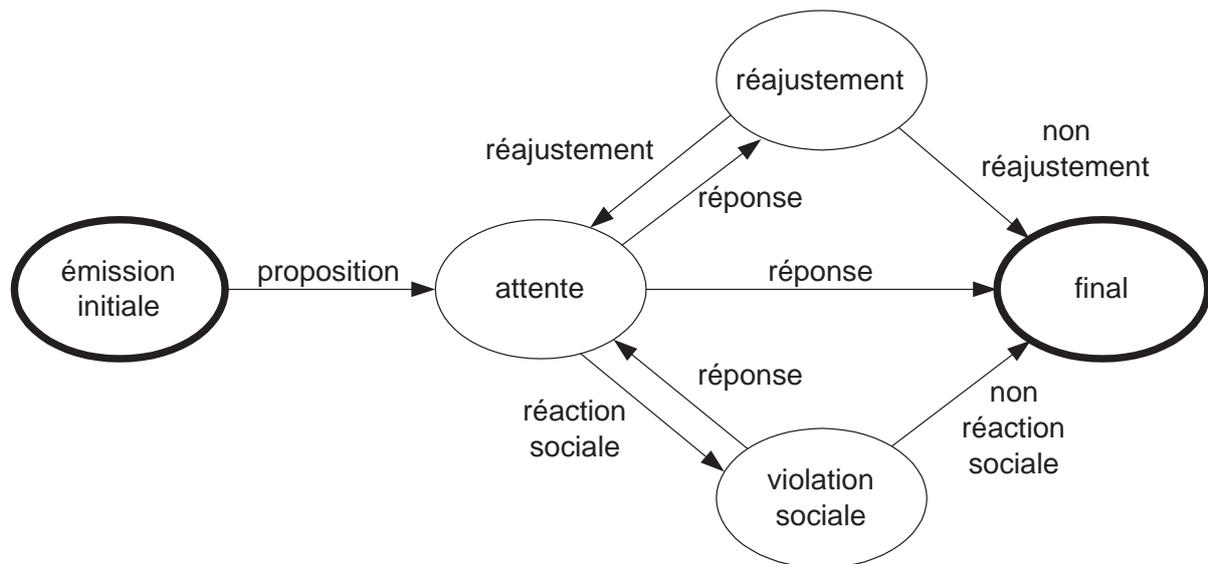


FIG. 2.6 – Protocole d'interaction de KQML+

Le point de vue adopté par Bouzouba et Moulin est proche du nôtre. Cependant, les auteurs se focalisent uniquement sur un modèle de communication. Ils s'inspirent de conversations humaines pour concevoir leurs performatives mais sans utiliser de corpus réel. Les énoncés ne proviennent pas de conversations ayant eu réellement lieu, bien qu'ils soient plausibles.

#### 2.4.4 CoLa

Le langage CoLa de Dignum et Weigand [Dignum et Weigand, 1995] s'appuie sur le modèle décrit en 2.2.5. Ce modèle utilise une syntaxe intégrant à la fois la logique déontique et la logique illocutoire pour représenter les communications entre agents. De la première, ils ne retiennent que les actes primitifs de *demande*, *promesse*, *affirmation* et *déclaration* et la seconde leur permet d'intégrer les notions d'*obligation*, d'*interdiction* et de *permission* pour une action (communicative ou non communicative). De plus, le mode d'accomplissement d'un acte de langage est précisé à l'aide d'un des trois indices possibles de *charité*<sup>16</sup>, *pouvoir*<sup>17</sup> ou *autorisation*<sup>18</sup>.

CoLa est un langage à trois niveaux d'organisation (*messages*, *transactions* et *services*), visant à gérer les communications dans leurs SMA.

<sup>16</sup>charity

<sup>17</sup>power

<sup>18</sup>authorization

Les messages sont décrits grâce à des primitives comme *request*, *assert* et *authorize*. Ces messages ont une sémantique prédéfinie dans le modèle logique. Par exemple, une demande de prix pour une commande s'écrira :

```
request_quotation == quotation.request_c
```

`request_quotation` est une demande sous la graduation de charité (`request_c`), qui correspond à un  $DIR_c$  dans le modèle logique. Cette requête est locale dans une transaction de type `quotation`.

Tous les messages sont regroupés en transactions. Chaque transaction possède un but qui permet de définir si la transaction a été effectuée avec succès ou si elle a échoué. Par exemple, la transaction permettant de demander une autorisation est définie par :

```
transaction get_authorization(Alfa:action)
messages
  request_c(auth(Alfa));
  authorize(Alfa), refuse(authorize(Alfa));
goal
  authorize(Alfa)
end-transaction;
```

Cette transaction générique est une requête de type charité, qui est suivie par l'autorisation demandée ou un refus d'accorder cette autorisation.

Les transactions sont regroupées en services, si ces derniers portent sur le même "objet". Ils précisent comment l'objet peut être créé, supprimé ou modifié. Par exemple, un service de paiement suite à une commande est vu comme portant sur un objet de type obligation.

```
service payment
provided by X
object
  obl(X, pay)
created by
  invoice
removed by
  transfer_money
  expire(date) => {obl(X, pay(fine))}
end-service
```

Ce service donne l'autorisation de réclamer un paiement à une compagnie qui a effectué une transaction. Le service est créé, non pas quand la commande est satisfaite mais au moment de l'émission de la facture (transaction `invoice`). Il est supprimé quand le transfert d'argent est effectué (transaction `transfer_money`). Le non-respect des délais par le client (transaction `expire(date)`) oblige le client à payer une amende.

Dignum et Weigand modélisent les communications comme un contrat, qui suit trois phases distinctes :

1. *établissement des termes du contrat*, sous la forme d'autorisations et d'obligations,

2. *acceptation du contrat*, c'est-à-dire du devis réalisé lors de la phase précédente,
3. *accomplissement du contrat*, c'est la réalisation effective du contrat.

CoLa apparaît comme très intéressant pour gérer les communications dans les SMA. En alliant la logique illocutoire et la logique déontique, CoLa permet de considérer réellement les actes de langage comme actions à part entière et fait le lien entre les communications et l'agenda des agents. En effet, pour les auteurs, l'agenda consiste en l'ensemble des obligations d'un agent. L'ajout d'une obligation dans l'agenda se fait à la réception d'une demande. La suppression d'une action de l'agenda est effectuée quand l'action est réalisée ou si l'obligation est violée. CoLa est, de plus, ouvert car récursif. Il peut donc aisément être enrichi.

Il n'est, par contre, pas exempt de tout reproche. En effet, l'introduction de la logique illocutoire dans la logique déontique n'est ni parfaite ni complète : l'acte illocutoire et l'acte perlocutoire d'un acte de langage sont confondus, tous les actes de langage ne peuvent pas être effectués (par exemple le rappel d'une commande déjà passée) et la graduation en charité-pouvoir-autorisation est sans doute trop peu précise. Elle ne permet pas, par exemple, de différencier la simple demande et la supplication.

### 2.4.5 Représentation des protocoles de communication

De nombreux formalismes permettent de spécifier et de représenter les protocoles de communication ou les dialogues.

Dans FIPA-ACL, les protocoles de communication sont décrits par des diagrammes de séquence UML (voir FIG. 2.5) [FIPA, 2002a; Odell *et al.*, 2000]. Chaque axe vertical correspond à un protagoniste de l'interaction. Tous les échanges de message sont représentés chronologiquement par des flèches partant de l'axe de l'émetteur, joignant l'axe du récepteur et annoté par le message.

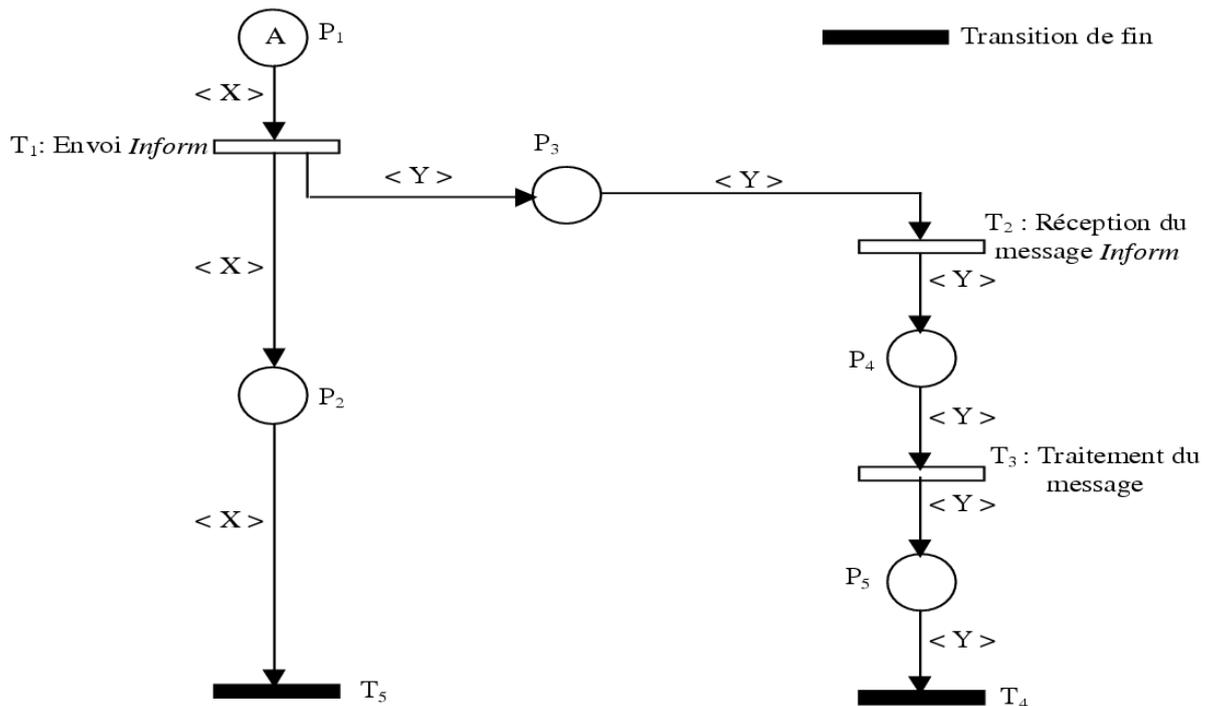
Un protocole peut aussi être décrit par un automate à états finis, dans lequel les noeuds représentent les états du protocole et les transitions sont franchies lorsqu'un acte communicatif est réalisé. Ce type de représentation est utilisé par Winograd et Florès (voir FIG. 2.4) [Winograd et Flores, 1986] ou encore Bouzouba et Moulin (voir FIG. 2.6), [Bouzouba et Moulin, 2001]. Cette approche donne des graphes très expressifs et lisibles.

Mazouzi propose d'analyser les communications dans des SMA à l'aide de réseaux de Petri colorés (RdPC) et utilise le graphe causal généré pour en extraire des instances d'interactions [Mazouzi, 2001]. Cette méthode exploite la relation de causalité entre les échanges communicatifs et non pas une relation temporelle accidentelle. Les RdPC sont un formalisme visuel très expressif qui respectent les principes de causalité, concurrence et non-déterminisme. (voir FIG. 2.7)

Les RdPC permettent une représentation compacte du système grâce à un processus de factorisation mais cette représentation semble moins lisible qu'avec un automate.

Les Dooley Graphs, introduits par Van Dyke Parunak [Parunak, 1996], contiennent la description des échanges, des informations sur la situation courante et les protagonistes d'une conversation, par une représentation des états mentaux (ou états de connaissance) des agents au cours de la communication. Un Dooley Graph (voir FIG. 2.8) se compose :

- de noeuds représentant un participant dans un certain état ; un trait en pointillé relie entre



<b>Transitions :</b>	<b>Places :</b>	<b>Fonctions :</b>
$C(T_1) = C(T_5) = A$	$C(P_1) = C(P_2) = A$	$X(A) = A$
$C(T_2) = C(T_3) = C(T_4) = B$	$C(P_3) = C(P_4) = C(P_5) = B$	$Y(A) = Y(B) = B$

FIG. 2.7 – Modélisation par RdPC du protocole FIPA-Inform

- eux les états successifs d'un même participant (par exemple des états de connaissances différentes ou croissantes) ;
- d'arcs annotés par les messages de la conversation.

Une relation d'un participant vers un autre appelle une réponse directe ou la consultation d'une tierce personne. Quatre types de relation sont possibles pour décrire la conversation : *Respond* (réponse directe à un message), *Reply* (réponse indirecte à un message), *Resolve* (introduit de la sémantique), *Complete* (met fin à l'échange).

Grâce aux Dooley Graphs, les participants à une conversation, leurs états mentaux et les messages échangés sont représentés simultanément.

Toutes ces approches ne tiennent pas compte de la dimension temporelle importante dans la dynamique conversationnelle. Il semble en effet nécessaire de représenter les durées associées aux actions communicatives et de synchroniser les interactions au sein des dialogues. Pour cela, il est possible d'utiliser les versions temporisées des automates et réseaux de *Petri*.

## 2.4.6 Discussion

Les protocoles de communication sont aux modèles de dialogue ce que les ACL sont à la théorie des actes de langage : une implantation selon un certain point de vue.

Cependant, comme l'explique Dignum, les problèmes soulevés par les ACL sont nombreux [Dignum, 2000]. Pasquier liste d'ailleurs ces problèmes [Pasquier, 2004].

Seq	Sndr	Addr	Utterance	Responds to	Replies to	Resolves	Completes
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	

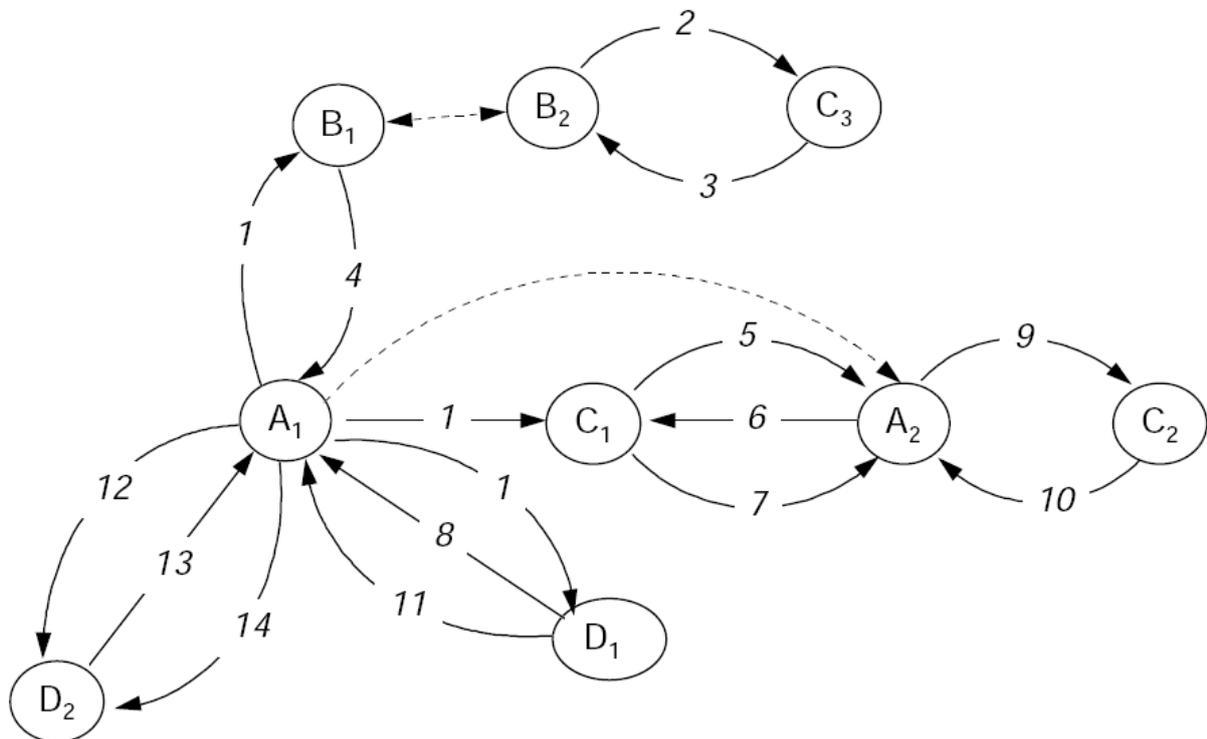


FIG. 2.8 – Exemple de conversation et son Dooley Graph associé

- *Problème de la minimalité sémantique* : les systèmes pré/post-conditions ne sont pas adaptés à toutes les situations.
- *Problème de conformité sémantique* : en pratique il est très difficile, voire impossible, de vérifier si les agents satisfont bien les pré/post-conditions. En effet, même si le problème est bien défini, le calcul de la situation est formellement exponentiel [Dignum, 2000].
- *Problème de l'alignement des sémantiques* : les sémantiques des ACL s'appuient sur certaines hypothèses simplificatrices qui permettent la viabilité du système mais réduisent l'expressivité. Par exemple, les "conditions de sincérité" mettent les agents dans l'impossibilité d'exprimer un mensonge, ou encore la "joignabilité sûre" qui stipule que les agents reçoivent correctement tous les messages ne correspond pas à la réalité des communications.
- *Problème de prise en compte du contexte social* : les obligations, engagements et normes de toutes sortes ne sont pas pris en compte dans les ACL.
- *Problème d'expression de la pragmatique* : les actes communicatifs émis par les agents sont directement liés à leur planification. Le but qui a motivé un acte communicatif devrait faire partie de ses pré-conditions.

KQML, FIPA-ACL et COLA sont parfaitement adaptés aux SMA, ce pour quoi ils sont conçus, mais semblent insuffisants pour prendre en compte les énoncés humains. De plus, la représentation des interactions suivant un protocole pré-déterminé paraît incompatible avec le caractère imprévisible et dynamique des conversations humaines. KQML+, qui propose une extension de KQML pour être plus proche des dialogues humains, paraît, de ce point de vue, plus adapté à notre problématique.

## 2.5 Conclusion

D'un point de vue computationnel, les SMA offrent un cadre intéressant à notre problématique. Plus particulièrement les SMA sans contrôle centralisé, pour lesquels le nombre d'agents est réduit, ne contenant que des agents cognitifs et dont les communications sont asynchrones. Ce type de SMA qualifie exactement le Problème de l'Agence de Voyage.

En ce qui concerne la modélisation informatique des dialogues humains, les ACL existants ont été conçus pour des SMA composés uniquement d'agents logiciels. Nous remarquons l'inadaptation de l'approche fondée sur les protocoles de communication. En effet, en raison de la diversité des réactions humaines devant un énoncé, il paraît difficilement concevable qu'un protocole rigide puisse représenter la structure des conversations humaines et la dynamique conversationnelle.

La théorie des actes de langage est la théorie la plus utilisée pour modéliser les énoncés simples. Ce n'est pas par hasard qu'elle est reconnue depuis longtemps par la communauté scientifique. La logique illocutoire de Searle et Vanderveken lui fournit un formalisme intéressant pour exprimer de façon logique la pragmatique des énoncés. Cependant, il s'agit d'une théorie de l'énoncé et non du dialogue. Elle ne peut donc pas modéliser les dialogues humains dans leur ensemble mais permet d'explicitier les unités de la conversation que sont les énoncés.

Tous les modèles du discours présentés s'appuient sur la théorie des actes de langage. Ils tiennent compte du caractère dynamique des discours et des dialogues, de l'enchaînement conversationnel et des relations existants entre les protagonistes. Nous avons choisi d'utiliser la théorie discursive de Vanderveken pour modéliser les conversations humaines car elle a pour

avantage d'étendre directement la théorie des actes de langage au dialogue. Vanderveken découpe les conversations en interventions et introduit les états mentaux comme unités de base du raisonnement. De plus, il est possible de donner une valeur de satisfaction à un dialogue.

Dans un dialogue, un acte de langage peut être vu comme la transmission d'une partie de l'état mental du locuteur à son interlocuteur. Nous voyons ici l'importance d'une représentation de l'état mental du locuteur, de l'environnement, et de la manière dont ces représentations sont modifiées par les actions des différents protagonistes. Un acte de langage n'est donc rien d'autre que l'expression de l'"état psychologique" du locuteur et peut être transmis, par exemple, sous forme de croyances, désirs et intentions. L'architecture d'agent BDI, qui intègre les états mentaux des agents sous forme de *croyances* (B), *désirs* (D) et *intentions* (I), apparaît comme la plus adaptée pour modéliser les agents résolvant le Problème de l'Agence de Voyage.

## **Deuxième partie**

# **Analyse et modélisation des interactions humaines**



# Chapitre 3

## Analyse des protocoles expérimentaux du point de vue de l'interaction

Cette section est consacrée à l'analyse des protocoles expérimentaux du point de vue des interactions. Seules les communications entre les sujets, au cours de la résolution de leur problème, sont prises en compte.

Cette étude des protocoles expérimentaux est réalisée en deux étapes :

- l'analyse des énoncés pris individuellement, *i.e.* façon dont s'effectue une communication unique entre deux sujets,
- l'analyse de la dynamique conversationnelle, *i.e.* l'agencement des énoncés durant une conversation.

Un exemple d'analyse de protocole du point de vue des interactions est donné en ANNEXE B.

### 3.1 Analyse des énoncés

Cette analyse se concentre sur l'envoi et la réception de messages, sans se préoccuper des actions internes effectuées par le sujet (la recherche d'information dans sa base de données ou les réservations). Les communications de chacun des neuf trinômes ont été fusionnées en un fichier unique appelé *fichier brut* (voir FIG. 3.1).

Dans chaque fichier brut, les mails envoyés par les sujets sont triés chronologiquement selon leur date d'envoi. Un message est décrit par :

- sa date d'envoi,
- son numéro parmi l'intégralité des actions effectuées par le sujet,
- son numéro parmi l'intégralité des messages envoyés par le sujet,
- son émetteur,
- son destinataire,
- son contenu en langage naturel.

Pour les horaires et réservations, le format est imposé par l'interface.

L'intérêt du Projet de l'Agence de Voyage portant sur la résolution coopérative de pro-

Date : Tue May 05  
Agent aérien : priscilla  
Agent ferroviaire : sauvegrain arthur  
Agent routier : jeande marie-aude

**[06:45:32] 2, Envoie message 1**  
| De: Agent aérien A: Agent ferroviaire  
| Y a-t-il un train demain qui va de Chateau-Neuf à Paris ?

**[06:46:23] 25, Envoie message 1**  
| De: Agent ferroviaire A: Agent aérien  
| > Y a-t-il un train demain qui va de Chateau-Neuf à Paris ?  
| Non.

**[06:47:27] 28, Envoie message 2**  
| De: Agent ferroviaire A: Agent routier  
| Y a-t-il des taxis ou des cars de Lille à Baisieux ?

**[06:49:10] 15, Envoie message 2**  
| De: Agent aérien A: Agent ferroviaire  
| Y a-t-il un train qui va de Paris à Montpellier demain ?

**[06:49:39] 7, Envoie message 1**  
| De: Agent routier A: Agent ferroviaire  
| Bonjour, pourrais-je savoir si il y a un train d'Annecy à Le Chatelard aujourd'hui dans la soirée pour trois sportifs ?

**[06:49:56] 33, Envoie message 3**  
| De: Agent ferroviaire A: Agent routier  
| Y a-t-il aussi des taxis ou des cars de Beaupreau à Angers pour demain vers 7h15 ?

**[06:50:21] 11, Envoie message 2**  
| De: Agent routier A: Agent ferroviaire  
| > Y a-t-il des taxis ou des cars de Lille à Baisieux ?  
| Lille aero ou gare ?

**[06:50:37] 37, Envoie message 4**  
| De: Agent ferroviaire A: Agent routier  
| Non, pas à 7h15 mais à 6h30 !

**[06:50:58] 20, Envoie message 3**  
| De: Agent aérien A: Agent routier  
| Y a-t-il un car qui va de Chateau-Neuf à Paris ? Si oui, combien cela couterait pour 2 adultes et 2 enfants ?

**[06:51:40] 43, Envoie message 5**  
| De: Agent ferroviaire A: Agent aérien  
| Oui.  
| =====HORAIRE=====  
| Paris-Montpellier (train)  
| Date: Demain -- 1 personne  
| 10h30/14h39 - 590 F

**[06:51:52] 24, Envoie message 3**  
| De: Agent routier A: Agent ferroviaire  
| Ci-joint, le tarif des taxis car il n'y a pas de cars.  
| =====HORAIRE=====  
| Beaupreau-Angers(gare) ( taxi)  
| Date: Demain -- 1 personne  
| duree : 00h30 - 200 F

**[06:52:32] 23, Envoie message 4**  
| De: Agent aérien A: Agent ferroviaire  
| Combien coûterait le voyage de Paris à Montpellier pour 2 adultes et 2 enfants ?

**[06:52:50] 49, Envoie message 6**  
| De: Agent ferroviaire A: Agent routier  
| > Bonjour, pourrais-je savoir si il y a un train d'Annecy à Le Chatelard aujourd'hui dans la soirée pour trois sportifs ?  
| Non, pas de gare au Chatelard.

**[06:53:15] 31, Envoie message 4**  
| De: Agent routier A: Agent aérien  
| > Y a-t-il un car qui va de Chateau-Neuf à Paris ? Si oui, combien cela couterait pour 2 adultes et 2 enfants ?  
| Il n'y a ni car ni taxi, désolé !

...

FIG. 3.1 – Fichier brut d'un trinôme

blèmes, les messages envoyés par un sujet après la résolution de son problème n'ont pas été pris en compte, à moins qu'ils ne participent à la résolution du problème d'un des autres sujets.

Comme les sujets coopèrent par messages en langage naturel, rédigés librement, l'analyse du contenu des messages n'est pas simple. Cinq types de messages ont été observés :

- les messages de *demande* pour se renseigner sur un trajet ou demander une réservation (ces messages impliquent souvent une réponse de la part de l'interlocuteur) ;
- les messages de *réponse* qui font suite à une demande concernant une information particulière ou une réservation ;
- les messages de *proposition* d'information ou d'horaire ;
- les messages d'*information* envoyés spontanément et n'attendant aucune réponse précise ;
- les messages de *civilités* qui ne font pas directement partie de la résolution du problème, comme les remerciements.

Cette description n'est pas satisfaisante car elle ne permet pas de classer l'intégralité des messages des protocoles expérimentaux. Par exemple, quelle serait la classe d'un message annulant une demande envoyée dans un précédent message ?

Pour rendre compte de ces différences, nous avons décidé de nous appuyer sur les travaux existant sur les ACL. Un des buts fixés est de vérifier si KQML et FIPA-ACL (voir 2.4) sont adéquats pour représenter les interactions humaines des protocoles expérimentaux. Ces derniers exploitent la théorie des actes de langage qui distingue cinq catégories d'actes illocutoires (les *descriptifs*, les *directifs*, les *engageants*, les *déclaratifs* et les *expressifs*) mais ne sont pas forcément adaptés aux conversations humaines puisqu'ils ont été conçus pour la gestion des communications entre agents dans les SMA.

### 3.1.1 Liste de performatives

Tous les messages de chaque fichier brut sont examinés individuellement pour voir s'ils correspondent à l'émission d'une des performatives de FIPA-ACL ou de KQML. Lorsque plusieurs performatives existantes sont possibles, celles en provenance de FIPA-ACL sont privilégiées, en raison de sa sémantique (voir SEC. 2.4.2) sans doute plus intéressante pour la représentation des intentions des sujets en termes d'états mentaux [Sadek, 1991], [Sadek, 1992], [Labrou *et al.*, 1999]. Si aucune performative existante n'est satisfaisante, une nouvelle est créée pour représenter le plus fidèlement possible le message. Dans un deuxième temps, les performatives ont été renommées lorsque cela semblait nécessaire pour que leur appellation soit la plus significative possible.

Les énoncés étant en langue naturelle, ils sont parfois ambigus sortis de leur contexte : il n'est pas toujours possible d'assigner à un message une performative juste en le lisant. Ainsi, dans le fichier brut donné en exemple, le message émis par le sujet en charge des transports ferroviaires au sujet en charge des transports routiers à 06 :50 :37 ("Non, pas à 7h15 mais à 6h30 !") peut être vu comme la réponse à une question ou l'annulation d'une demande remplacée par une autre. Pour lever ce type d'ambiguïté, le(s) message(s) émis précédemment entre les deux sujets ont été pris en compte. Dans le cas de l'exemple donné, c'est la seconde hypothèse qui a été retenue. Cette analyse est grandement simplifiée quand un message est clairement désigné comme une réponse à un envoi précédent (par exemple, le deuxième message du fichier brut répond au premier en l'incluant), mais ce n'est évidemment pas toujours le cas.

De l'analyse des protocoles expérimentaux est née une liste de performatives (voir TAB. 3.1). Celles-ci sont classées selon la taxinomie de Searle [Searle, 1969]. Pour chacune d'elles nous donnons son nom et sa description avec un exemple tiré des protocoles. Les performatives sont données dans l'ordre alphabétique selon la classe à laquelle elles appartiennent. *S* est le locuteur de la performative et *H* son interlocuteur.

Par comparaison avec la classification de Searle, les performatives observées proviennent des trois classes suivantes : les *descriptifs*, les *directifs* et les *engageants*. On notera l'absence des *déclaratifs*, ce qui n'est guère surprenant compte tenu de notre problème qui ne s'y prête pas. De plus, nous considérons, comme Nicolle et Saint-Dizier De Almeida, que la classe des *expressifs* n'a pas lieu d'être, un expressif n'étant qu'un descriptif particulier [Nicolle et Almeida, 1999]. En effet, la description des états d'âme d'un agent est la description d'une partie du monde, même s'il s'agit d'une introspection.

D'autre part, la plupart des messages analysés sont de plusieurs types à la fois et cette combinaison peut être :

- une simple concaténation de plusieurs messages ("*Voilà ton horaire. Dois-je réserver ?*"),
- une combinaison conditionnelle ("*Y a-t-il un car qui aille de Paris à Orléans, si oui quel en est le prix ?*")

Le premier type de message est considéré comme deux messages distincts qui se succèdent (un *reply* suivi d'un *propose*) et le second type de message comme un seul message de type *query*.

Il reste cependant sept énoncés qui ne correspondent à aucune des performatives présentes dans la liste. Les deux premiers messages font suite à un *notUnderstood* :

ROUTIER

> *Es-tu sûr que ce message m'est destiné ?*

*Quel message ?*

FERROVIAIRE

*Le message concernant le trajet Paris Annecy !*

Le sujet Routier semble avoir bien compris le précédent message de l'interlocuteur et demande des précisions. La performative la plus proche de ce type de message serait un *refine* mais ceux-ci ont été réservés uniquement pour des demandes de précision suite à un *query*. De la même façon, la réponse de l'agent ferroviaire ne correspond à aucune performative.

Le troisième message vient en réponse d'un remerciement :

FERROVIAIRE

> *Merci*

*De rien !*

Le quatrième message est une proposition de service ne portant sur aucun horaire :

AERIEN

*Puis je t'aider ? ?*

<b>PERFORMATIVE</b>	<b>DESCRIPTION</b>
<b>Descriptifs</b>	
inform	S envoie spontanément à H une information. Dans le cadre du Problème de l'Agence de Voyage, S informe H de l'état d'avancement de son problème. <i>"Mission accomplie, merci encore."</i>
notUnderstood	S ne comprend pas un des précédents messages de H. <i>"Je n'ai jamais demandé cette réservation."</i>
reply	S répond à une demande de H. Il s'agit ici d'un envoi d'horaire ou de réservation. <i>"Non, il n'y a pas de gare au Chatelard."</i> <i>"Voilà qui est fait."</i>
thank	S remercie H. <i>"Merci beaucoup !"</i>
<b>Directifs</b>	
acceptProposal	S accepte une proposition faite par H. <i>"Oui, réserve-moi cet horaire."</i>
cancel	S demande à H de pas prendre en compte un de ses précédents messages. Il a été observé que les cancels n'étaient appliqués qu'à des messages de type query ou reply. <i>"Je n'ai réservé qu'une personne pour le trajet précédent, je corrige."</i>
query	S demande ou redemande à H de lui envoyer des informations. Il s'agit d'une (re)demande d'horaire ou de réservation. <i>"Combien coûte le trajet Paris-Annecy pour trois ?"</i> <i>"Pourrais-tu me réserver cet horaire ?"</i>
refine	S demande à H plus d'information pour pouvoir satisfaire sa demande. <i>"Lille aéro ou gare ?"</i> <i>"Il y a des cars mais pour quel horaire ?"</i>
refuseProposal	S refuse une proposition faite par H. Pour ce problème, c'est une demande de précision de lieu (entre gare ou aéroport) ou de temps. <i>"Non, ce sont les horaires qui ne collent pas."</i>
<b>Engageants</b>	
propose	S propose à H de lui fournir une information. Ici, il s'agit d'une proposition d'horaire ou de réservation. <i>"Je peux te proposer un trajet Orléans-Montpellier puis Montpellier-Toulouse."</i> <i>"Dois-je réserver ?"</i>

TAB. 3.1 – Liste des performatives

Enfin, les trois derniers messages qui ne peuvent pas être annotés à l'aide d'une des performatives de la liste sont :

FERROVIAIRE	>T'as pas encore fini ?
	Non, j'attends mes réservations du routier
AÉRIEN	Pareil. Je devine qu'elle croule sous le boulot - en fait c'est peinard agent aérien, tout le monde te fout la paix !

et

FERROVIAIRE	J'ai des problèmes avec l'agent routier qui ne répond pas à mes questions, je suis complètement bloquée
-------------	---

Ces messages ont tous été envoyés par des sujets bloqués dans la résolution de leur problème à cause d'une rétention d'information de la part d'un autre sujet. Ces messages servent essentiellement à passer le temps. Ils ne participent pas réellement à la résolution des problèmes mais ne peuvent pas être exclus de l'analyse puisqu'ils ont eu lieu durant cette résolution.

### 3.1.2 Comparaison avec KQML et FIPA-ACL

Contrairement à KQML et à FIPA-ACL qui adoptent un point de vue classique des interactions, en se limitant à un schéma rigide de type question/réponse. Cette analyse se place dans un cadre plus large où chaque demande peut correspondre à plusieurs types de comportements de la part de l'interlocuteur, allant de la réponse ou de la réservation à une absence de réponse.

Hormis deux performatives permettant d'explicitier une question ambiguë (*refine*) et de remercier un sujet (*thank*), toutes les performatives observées dans les protocoles expérimentaux sont présentes dans FIPA-ACL et éventuellement dans KQML. Le TAB. 3.2 présente pour chacune des performatives, son équivalent dans chacun des deux langages, s'il existe.

	KQML	FIPA-ACL
<i>acceptProposal</i>		accept-proposal
<i>cancel</i>	unachieve	cancel
<i>inform</i>		inform
<i>notUnderstood</i>	untell, error	not-understood
<i>propose</i>		propose
<i>query</i>	ask-if, ask-all, ask-one, achieve	query-if, query-ref, request
<i>refine</i>		
<i>refuseProposal</i>		refuse-proposal
<i>reply</i>	tell, deny	inform-ref, inform-if
<i>thank</i>		

TAB. 3.2 – Comparaison avec KQML et FIPA-ACL

KQML ne contient aucune performative permettant de représenter des énoncés humains de type *acceptProposal*, *inform*, *propose*, *refuseProposal* et *thank* présents dans les protocoles expérimentaux. Il en est de même pour une partie des messages de type *cancel* et *reply*, par exemple lors de l’annulation d’une demande ou avec l’envoi d’une réservation. En contrepartie, un grand nombre des performatives de KQML ne sont pas présentes dans la liste puisqu’elles n’ont pas été observées dans les protocoles expérimentaux. Ces observations ne sont guère surprenantes compte tenu des objectifs de KQML qui vise à n’être utilisé qu’avec des SMA non hétérogènes, composés uniquement d’agents logiciels.

Comparativement, il est bien évident que FIPA-ACL permet de couvrir les dialogues humains des protocoles expérimentaux beaucoup plus efficacement que KQML. Cependant, l’ensemble des performatives de FIPA-ACL paraît tout de même insuffisant pour représenter les dialogues humains du fait de l’absence d’une primitive de remerciement (*thank*) et d’une primitive de demande de précision (*refine*).

### 3.1.3 Étude statistique

Chacun des messages des fichiers bruts a été annoté avec la(les) performative(s) nécessaire(s) pour effectuer une étude statistique. Les résultats de cette étude sont présentés dans le TAB. 3.3 qui contient le nombre d’instances et la proportion de chaque type de performative.

Performatives	Nombre d’instances	Proportion
query	474	42,97%
reply	437	39,62%
refine	60	5,44%
thank	30	2,72%
propose	29	2,63%
inform	26	2,36%
cancel	18	1,63%
acceptProposal	10	0,91%
notUnderstood	9	0,82%
refuseProposal	3	0,27%
autres	7	0,63%
total	1103	100%

TAB. 3.3 – Répartition statistique des performatives

Le nombre performatives (1103) observées dans les protocoles expérimentaux ne correspond bien évidemment pas au nombre de messages envoyés (930) puisque certains messages contiennent plusieurs performatives.

On notera le très grand nombre de performatives de type *query* et *reply*, qui forment la colonne vertébrale de tout dialogue de recherche d’information. Bien qu’une demande puisse être suivie de l’envoi de plusieurs réponses qui se complètent (à cause des contraintes liées à l’interface), le nombre de *query* est bien plus conséquent que le nombre de *reply*. Cette différence s’explique par les demandes qui n’obtiennent aucune réponse ou qui nécessitent d’être réitérées, principalement en raison de la communication par courriers électroniques.

La spécificité des dialogues humains se remarque par :

- la présence d’erreurs et/ou d’imprécisions (6,26% des performatives sont émises à la suite de ces imperfections sous la forme de 60 *refine* et 9 *notUnderstood*),
- la diversité des réactions des sujets qui contraste avec les schémas figés utilisés habituellement dans les SMA (*query* sans *reply*, *propose* sans *acceptProposal* ni *refuseProposal*, remerciements non systématiques, etc.)
- et leur relative politesse (30 *thank*).

Les sept "autres" performatives sont celles mentionnées précédemment comme ne correspondant à aucune performative listée.

### 3.1.4 Discussion

Cette première analyse souffre de deux principaux défauts.

- Le premier, dont il est difficile de s’affranchir, est qu’elle est faite *a posteriori*, en ayant l’intégralité des conversations. L’interprétation des énoncés est donc biaisée par les réponses fournies par l’interlocuteur. Ce n’est évidemment pas le cas au cours d’une conversation réelle puisque pour interpréter un énoncé, les protagonistes ne peuvent se référer qu’à cet énoncé et à tous ceux qui ont été émis précédemment.
- Le second est que cette analyse est statique et ne rend pas compte de la dynamique conversationnelle (temporalité des dialogues et multiplicité des comportements observés chez les sujets).

Ainsi, les interactions ne peuvent être considérées comme une simple succession de questions et de réponses suivant un schéma figé, comme dans KQML et FIPA-ACL.

De plus, ces deux langages proposent un ensemble de performatives adapté aux dialogues entre agents logiciels mais ne permettent pas de couvrir l’intégralité des échanges des conversations humaines de demandes d’information.

## 3.2 Dynamique conversationnelle

Comme chaque message des fichiers bruts a été annoté par une performative, les dialogues peuvent être présentés sous la forme de chronogrammes. Les chronogrammes présentés dans cette thèse ont été dessinés à l’aide d’un outil réalisé par Michel Cusenza et Émilien Hamel, dans le cadre d’un projet semestriel à l’Institut National des Sciences Appliquées de Rouen (INSA-Rouen). Leur logiciel permet, à partir d’un protocole expérimental correctement annoté, d’obtenir la représentation en chronogramme des interactions, dans une image au format bitmap. L’exemple de chronogramme proposé FIG. 3.2 correspond au fichier brut de la FIG. 3.1.

Les chronogrammes permettent de visualiser la temporalité et l’enchevêtrement des échanges entre les sujets. Ils se composent d’un axe référençant le temps et de quatre axes pour symboliser les sujets (agent aérien, agent ferroviaire, agent routier et à nouveau agent aérien pour favoriser la lisibilité des échanges<sup>1</sup>). Les messages sont représentés par une flèche partant de l’axe du locuteur, aboutissant à l’axe de l’interlocuteur et dont l’étiquette est la performative qui lui est

<sup>1</sup>Ceci évite d’avoir les messages entre l’agent routier et l’agent aérien qui chevauchent l’axe de l’agent ferroviaire.

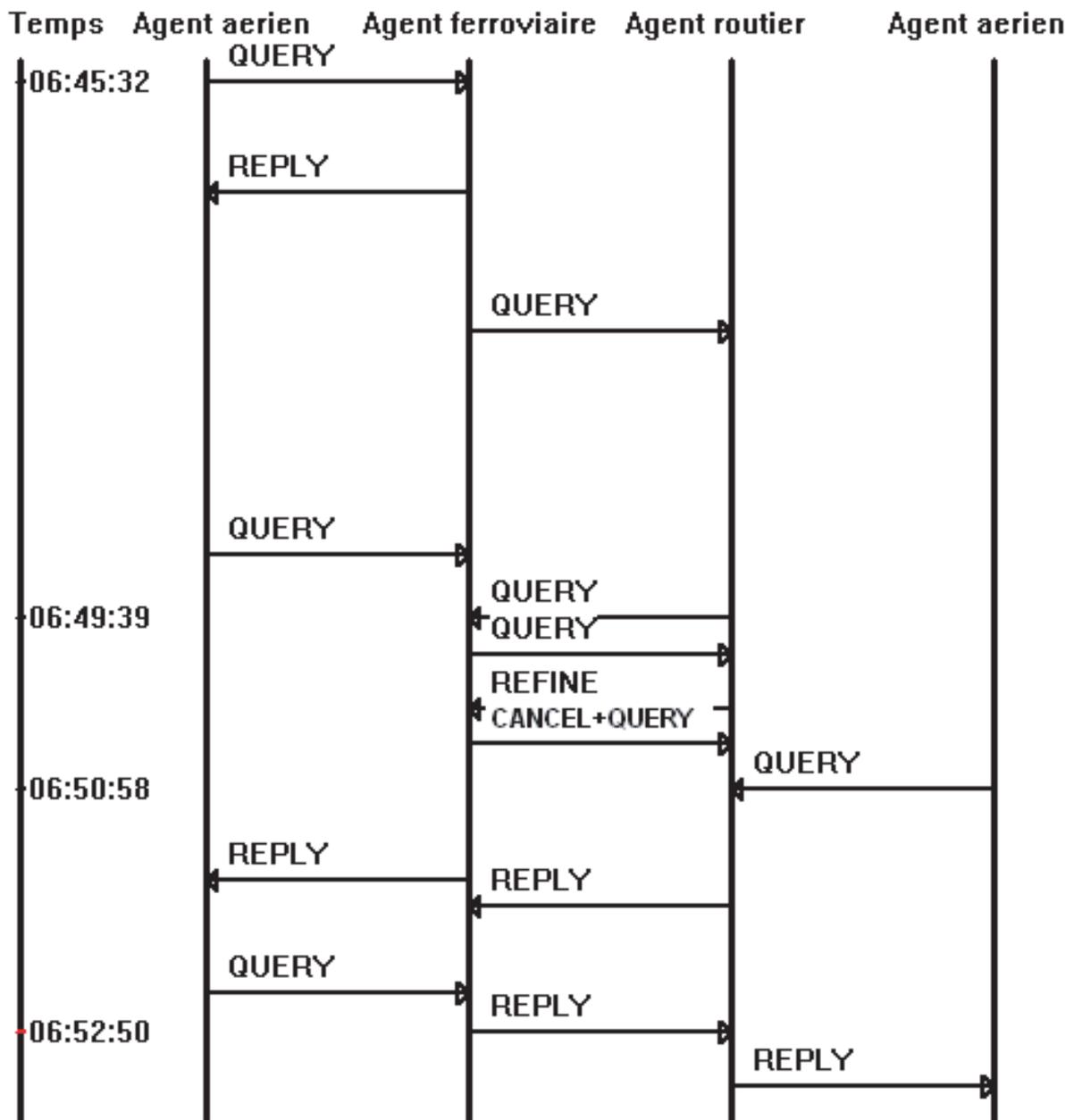


FIG. 3.2 – Exemple de chronogramme

assignée. Chaque message est situé verticalement dans l'axe du temps en fonction du moment de son émission.

L'analyse initiale doit maintenant être reconsidérée à un niveau supérieur, celui du discours. Pour cela, les fichiers annotés par les performatives et leur représentation en chronogrammes forment une bonne base d'étude.

Cette analyse se base sur les travaux de Vanderveken (voir 2.3.2), qui étend la théorie des actes de langage au discours. Il maintient la décomposition des conversations en *actes illocutoires*, introduit les *états mentaux* comme unités de base du raisonnement et regroupe les messages en *échanges*.

### 3.2.1 Actes illocutoires, interventions et échanges

Si on se réfère à la théorie discursive de Vanderveken, chaque message correspond à une *intervention* pouvant contenir un ou plusieurs *actes illocutoires* (annotés par les performatives). Les messages échangés peuvent être regroupés en *échanges*. Pour chacun de ces échanges, une intention discursive est prêtée au sujet initiateur, en fonction de sa première performative. Par exemple, les deux premiers messages (un *query* et un *reply*) peuvent être regroupés dans un même échange. Le locuteur du *query* souhaite que son interlocuteur lui envoie une réponse. L'échange est donc d'intention discursive de type directif. Les autres regroupements possibles du fichier brut, tous du même type, sont présentés FIG. 3.3. Les échanges  $I_2$  et  $I_7$  ne sont pas terminés et se poursuivent après la fin du fichier brut.

Pour effectuer ces regroupements sans qu'ils se chevauchent, certains messages ont dû être déplacés dans le fichier brut. Cela souligne l'imbrication des échanges entre les sujets au cours de la résolution de leur problème, qui se remarque d'autant mieux sur les chronogrammes (voir FIG. 3.4). Cet enchevêtrement n'est pas spécifique aux dialogues humains mais provoqué par l'utilisation de courriers électroniques comme système de communication, ce qui introduit un délai entre le moment où un message est émis et le moment où celui-ci est lu.

Les échanges repérés au cours de l'analyse peuvent être classés dans l'une des quatre catégories suivantes : *demandes d'information*, *propositions d'information*, *envois spontanés d'information* et *traitement des erreurs*. La première a une intention discursive primitive de type *directif*, la seconde d'*engageant*, la troisième de *descriptif* et la dernière de *directif*. Le traitement des erreurs est le seul échange dont l'intention discursive ne correspond pas au type de son message initiateur puisqu'il est envoyé par erreur. Ces échanges ont une intention discursive de type directif car l'intention du sujet recevant le message erroné est que le locuteur l'annule par un *cancel*.

La terminaison des échanges, quant à elle, définit leur satisfaction. Un échange peut être satisfait (réalisation effective du but visé par l'initiateur de l'échange), ou non (réalisation du contraire du but ou non réalisation de ce but). Par ailleurs, la satisfaction d'un échange doit être distinguée de son succès. Un échange est conduit avec succès si son type est bien reconnu par les deux protagonistes mais il ne sera pas obligatoirement satisfait. Sa satisfaction dépendra du comportement de l'interlocuteur de la performative initiatrice.

Le TAB. 3.4 présente des exemples d'échanges avec, pour chacun d'eux, son type et sa satisfaction. Il contient :

- trois demandes d'information (deux satisfaites et une non-satisfaite),

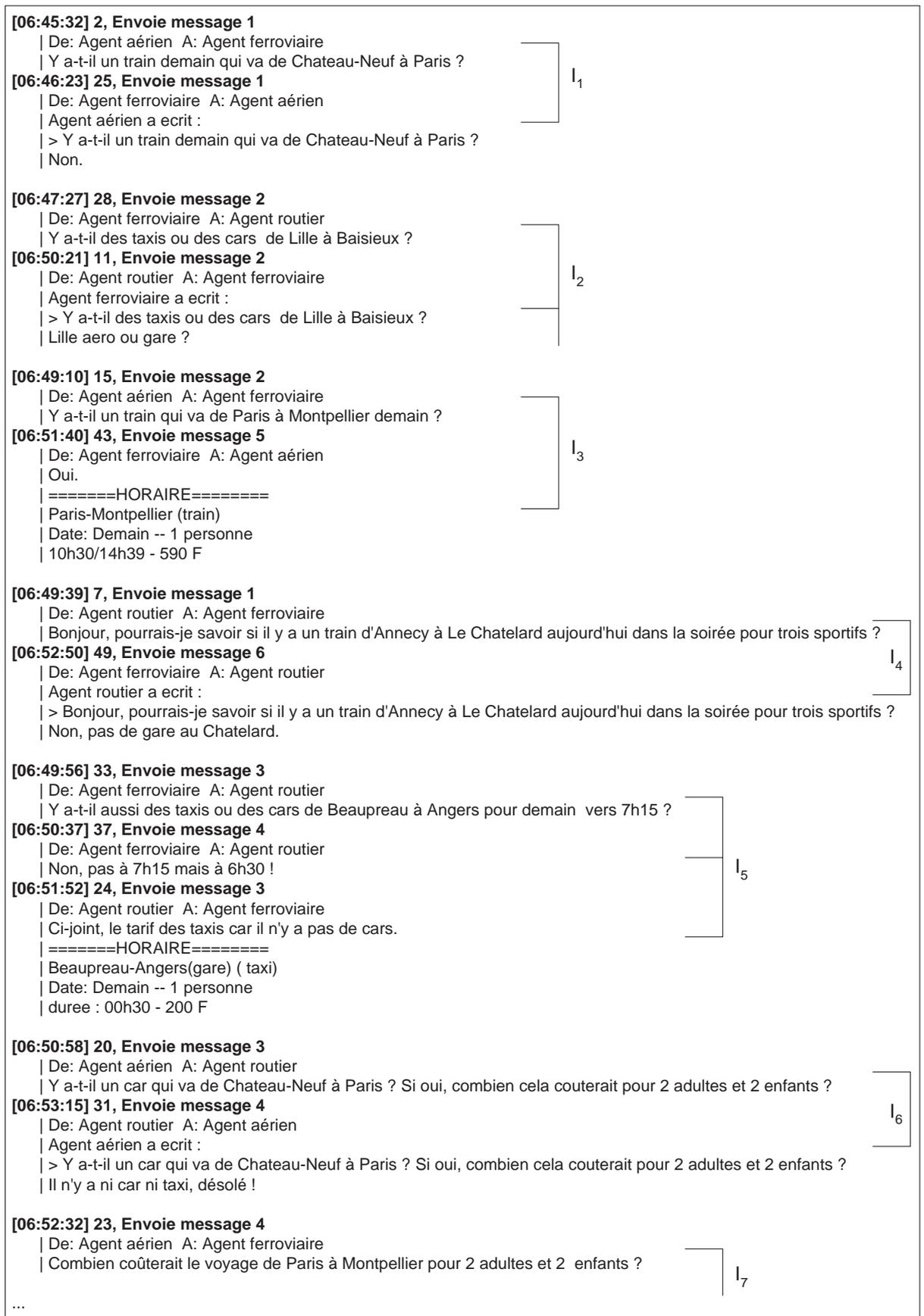


FIG. 3.3 – Regroupement de messages en échanges

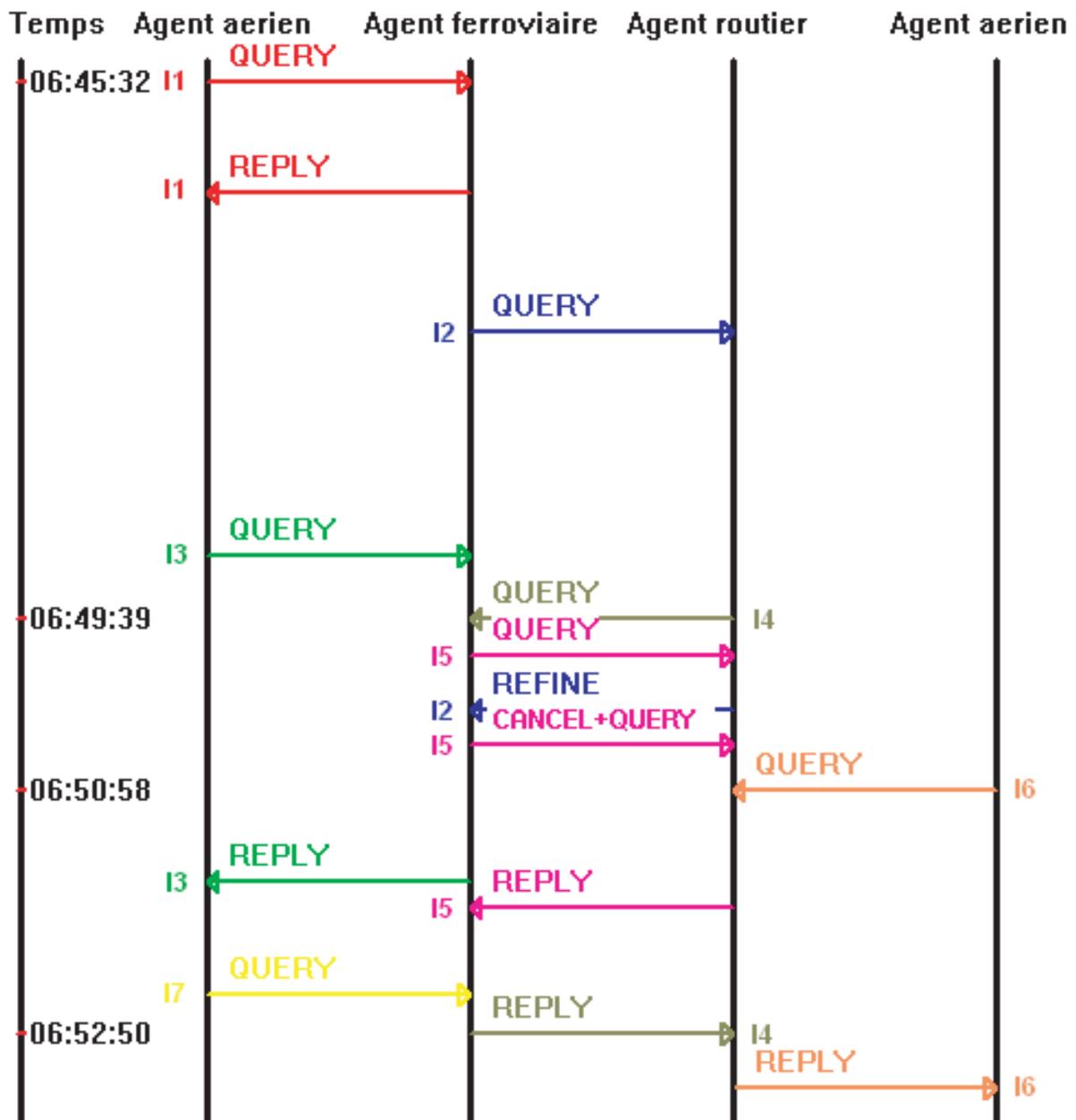


FIG. 3.4 – Imbrication des échanges

- deux propositions d’information (une satisfaite et une non-satisfaite),
- un envoi spontané d’information (toujours satisfait),
- un traitement des erreurs (satisfait).

Les échanges dont le but est de savoir si une ligne existe entre deux villes ou d’obtenir un horaire, un tarif ou une réservation sont tous des demandes d’information. Ils sont satisfaits à partir du moment où ils reçoivent une réponse. De plus, aucun échange de traitement des erreurs qui ne soit pas satisfait n’a été référencé.

La performative initiatrice permet de classer l’échange dans un type particulier, et la performative de clôture, si elle est présente, permet de définir la satisfaction de l’échange. Seul le cas particulier des envois spontanés d’information ne nécessite qu’un seul acte maître, celui de la performative initiatrice qui joue aussi le rôle de clôture. Ils sont toujours satisfaits.

Un résumé des différents types d’échanges observés avec leur intention discursive primitive et leurs performatives initiatrice et de clôture, correspondantes est présenté (voir TAB. 3.5).

### 3.2.2 Temporalité des échanges

Dans les dialogues analysés, le temps est apparu comme un élément prépondérant. Quand un sujet met trop longtemps à répondre à une demande, il peut recevoir un ou plusieurs messages de relance. Un échange peut être considéré comme terminé par les interlocuteurs sans qu’il y ait émission explicite d’une performative de terminaison ; l’échange n’est alors pas satisfait.

L’échange suivant est un exemple dans lequel la demande du sujet en charge du transport ferroviaire n’est pas satisfaite avant deux messages de relance.

FERROVIAIRE : query  
*Combien coûte un trajet Beaupréau-Angers ?*

FERROVIAIRE : query  
*Peux-tu me trouver une correspondance entre Beaupréau et Angers vers 06h ?*

FERROVIAIRE : query  
*Peux-tu répondre à ma demande : existe-t-il une ligne Beaupréau-Angers en car ?*

ROUTIER : reply  
*voici les horaires*  
 =====HORAIRE=====  
*Beaupréau-Angers(gare) (car)*  
*Date : Demain – 1 personne*  
*15h00/15h30 - 40 F*

FERROVIAIRE : query  
*Merci mais peux-tu me dire s’il y a un car Beaupréau-Angers vers 06H ?*

ROUTIER : reply  
*Il n’y a pas de bus avant 9 h.*

Échange	Type	Satisfaction
<p>ROUTIER : query Peux-tu me réserver pour 3 le StMartin-Paris ? FERROVIAIRE : reply</p> <p>voila =====HORAIRE=====</p> <p>Réservation effectuée : SaintMartin-Paris (train) Date : Demain – 3 personnes 18h10/18h35 - 20 F</p>	Demande d'information	Oui
<p>ROUTIER : query Aurais-tu un train Dijon-Annecy, s'il te plaît ? FERROVIAIRE : reply</p> <p>Pas de ligne directe Dijon Annecy.</p>	Demande d'information	Oui
<p>AÉRIEN : query Car Montpellier(12h)-Maguelone ? ROUTIER : refine Est-ce que c'est Montpellier gare ou aéro ? AÉRIEN : cancel</p> <p>On change tout !</p>	Demande d'information	Non
<p>ROUTIER : propose &gt; Y-a-t-il un car reliant l'aéroport à la gare de &gt; Lille, après 9h05 ? Pas de ligne directe, il ne te reste que le taxi. FERROVIAIRE : acceptProposal</p> <p>Vas-y pour le taxi ! ROUTIER : reply</p> <p>Voilà pour le taxi. =====HORAIRE=====</p> <p>Lille(aéro)-Lille(gare) (taxi) Date : Demain – 1 personne Durée : 00h15 - 60 F</p>	Proposition d'information	Oui
<p>FERROVIAIRE : propose Veux-tu réserver ? AÉRIEN : notUnderstood Te réserver quoi, excuse-moi ? FERROVIAIRE : propose Veux-tu réserver, toi, un billet pour Paris-Montpellier ? AÉRIEN : refuseProposal</p> <p>Ce sont les horaires qui ne collent pas.</p>	Proposition d'information	Non
<p>AÉRIEN : inform L'agent aérien a enfin réussi...</p>	Envoi spontané d'information	Oui
<p>FERROVIAIRE : reply</p> <p>voila =====HORAIRE=====</p> <p>Réservation effectuée : SaintMartin-Paris (train) Date : Demain – 3 personnes 18h10/18h35 - 20 F AÉRIEN : notUnderstood</p> <p>Je n'ai jamais demandé cette réservation de Saint-Martin à Paris ! FERROVIAIRE : cancel</p> <p>Désolé c'était une blague !</p>	Traitement des erreurs	Oui

TAB. 3.4 – Exemples d'échanges

Type d'échange	Intention discursive	Performative initiatrice	Performative de clôture
demande d'information	directif	query	reply
proposition d'information	engageant	propose	reply
envoi spontané d'information	descriptif	inform	inform
traitement des erreurs	directif	-	cancel

TAB. 3.5 – Les différents types d'échanges

Voici maintenant un exemple d'échange de type proposition d'information ne recevant aucune réponse :

FERROVIAIRE : propose <i>Voilà. Veux-tu que je te le réserve ?</i> =====HORAIRE===== <i>Paris-Montpellier (train)</i> <i>Date : Demain – 1 personne</i> <i>15h30/19h55 - 590 F</i>	
---	--

De la même façon, les sujets n'utilisent pas systématiquement une performative de type *thank* à la fin d'un échange qui a été satisfaite. Les protagonistes savent que l'échange est terminé puisqu'ils n'en tiennent plus compte. C'est donc qu'ils le considèrent comme fini si après un certain temps il n'a pas été réactivé par l'envoi ou la réception d'un nouveau message.

### 3.2.3 Digressions

Au cours de la conversation, il se peut qu'un des interlocuteur rencontre une difficulté d'interprétation pour un énoncé (un non-attendu pour Lemeunier [Lemeunier, 2000]). La conversation quitte alors son cheminement pour se diriger vers ce que Vanderveken nomme une digression.

Dans les protocoles expérimentaux, la présence effective de dialogues incidents ou digressions, selon que l'on emprunte la terminologie de Luzzati [Luzzati, 1995] ou de Vanderveken [Vanderveken, 1999] a pu être confirmée. Ils s'agit de demandes de précision d'horaire ou de lieu quand une ville possède un aéroport et une gare.

On pourra trouver FIG. 3.5 un exemple de digression. Sur la gauche se trouvent les interactions entre les deux protagonistes et sur la droite les performatives correspondantes regroupées en échanges.

Dans cet exemple, on retrouve deux échanges différents : une demande d'information *A* et une proposition d'information *B* dont la fin n'est pas présentée. Au cours de *A*, les deux protagonistes de la conversation conduisent l'échange courant de demande d'horaire de train comme un dialogue *régissant* ( $A_1$ ). Puis ils initient un dialogue *incident* pour résoudre le problème apparu ( $A_2$ ), c'est-à-dire un horaire non précisé, en recherchant une interprétation commune. Quand cette interprétation commune apparaît, ils réactivent le dialogue régissant, mis en suspens (retour à  $A_1$ ).

Dans les fichiers bruts, les digressions se retrouvent aussi sous une forme différente. En effet, dans les problèmes soumis aux sujets, certaines villes possèdent une gare et un aéroport.

Le sujet en charge des transports routiers a besoin de cette précision pour donner les prix et les horaires de bus ou les prix des taxis. Nous avons ainsi pu observer des dialogues de demande de précision d'une forme similaire à celui de l'échange  $I_2$  présenté précédemment.

```

FERROVIAIRE
Y a-t-il des taxis ou des cars de Lille à Baisieux ?

ROUTIER
Lille aéro ou gare ?

FERROVIAIRE
Gare.

ROUTIER
Voici ton horaire :
=====HORAIRE=====
Lille(gare)-Baisieux (taxi)
Date : Demain – 1 personne
Durée : 00h30 - 180 F

```

Il y a dialogue incident à chaque émission d'une performative de type *refine*.

### 3.2.4 Étude statistique

Les annotations ajoutées aux messages des fichiers bruts ont été complétées pour effectuer une étude statistique des types d'échanges utilisés par les sujets au cours de la résolution de leur problème. Les messages qui n'ont pas pu être annotés ont été retirés lorsqu'ils ne jouaient aucun rôle dans la reconnaissance du type de l'échange auquel ils appartiennent.

Chaque message possède maintenant un label de la forme  $TT - X - n - s - PERF$ , où  $TT \in \{DI, PI, EI, TE, HR\}$  est le type d'échange : *DI* pour une demande d'information, *PI* pour une proposition d'information, *EI* pour un envoi spontané d'information, *TE* pour un dialogue de traitement des erreurs et *HR* si l'échange a lieu après la fin de la résolution du problème. Pour l'étude statistique, la dernière catégorie n'est pas prise en compte.  $X$  est une chaîne de caractères permettant de nommer un échange (par exemple l'échange *Ab*).  $s \in \mathbb{N}$  est le nombre de performatives présentes dans l'échange.  $n \in \mathbb{N}$  est le numéro de la performative dans l'échange courant.  $PERF$  est le type de performative ; c'est cette dernière annotation qui nous a servi à effectuer l'étude statistique sur les performatives (voir 3.1.3).

Les résultats de cette étude sont présentés dans un tableau qui contient le nombre d'instances et la proportion de chaque type d'échange (voir 3.6). Ce tableau est à rapprocher de celui contenant la répartition statistique des performatives (TAB. 3.3 de 3.1.3).

Au cours de l'expérimentation, les protagonistes ont mené 416 échanges. La plupart de ces échanges sont des demandes d'informations (86,06%), ce qu'il était possible de pressentir après l'étude statistique des performatives et la grande proportion de *query* et de *reply* dans les fichiers annotés.

La participation spontanée aux problèmes des autres sujets n'est pas très élevée : seulement 6,42% des échanges sont des propositions d'information. Une seule hypothèse peut être émise

Messages	Performatives	
<b>[08:23:48] 561, Message 39, De: Agent aérien À: Agent ferroviaire</b> Je viens juste d'apprendre qu'il était possible de voyager de Paris à Montpellier en train! Pourrais-tu me donner des horaires?	query	A <sub>1</sub>
<b>[08:28:45] 554, Message 50, De: Agent ferroviaire À: Agent aérien</b> Oui, c'est possible, il y a 7 trains différents entre 8:12 et 18:28 (heure de départ). Est-ce que ça t'intéresse? Pourrais-tu être plus précis sur l'heure de départ? Merci!	refine	
<b>[08:29:52] 598, Message 41, De: Agent aérien À: Agent ferroviaire</b> Oui, 10:15.	query	
<b>[08:33:18] 583, Message 51, De: Agent ferroviaire À: Agent aérien</b> Voici le premier : =====HORAIRE===== Paris-Montpellier (train) Date: Demain -- 1 person 10:30/14:39 - 590 F	reply	
<b>[08:33:49] 589, Message 52, De: Agent ferroviaire À: Agent aérien</b> Et le 2ème, en veux-tu d'autres? =====HORAIRE===== Paris-Montpellier (train) Date: Demain -- 1 person 12:06/16:21 - 590 F	reply + propose	
		A <sub>2</sub>
		B

FIG. 3.5 – Exemple de digression

Échanges	Nombre d'instances	Proportions
demande d'information	358	86,06%
propositions d'information	28	6,73%
envois spontanés d'information	26	6,25%
traitement des erreurs	3	0,72%
autre	1	0,24%
total	416	100%

TAB. 3.6 – Répartition statistique des échanges

pour essayer d'expliquer cette faible proportion : les sujets ont sans doute éprouvé des difficultés à deviner les problèmes posés aux autres sujets humains, ce qui a dû les gêner pour proposer des trajets susceptibles de les intéresser.

La proportion d'envois spontanés d'information (6,25%) n'est pas, en elle-même, significative. Par contre, ces 26 envois spontanés ont été effectués par 72% des sujets en ayant eu l'occasion. En effet, les envois spontanés ne se font que deux fois par sujet (à la fin de la résolution du problème, à chacun des deux autres sujets) et le dernier sujet à résoudre son problème n'a pas l'occasion d'effectuer un envoi car la simulation se termine avant. Il y a donc 26 envois spontanés d'information pour 36 possibles.

Le nombre de demandes d'information est inférieur au nombre de *query* et au nombre de *reply* car une demande peut être répétée et une réponse peut se faire à l'aide de plusieurs messages. De la même manière, une seule proposition d'information a été réitérée (28 propositions d'information pour 29 performatives de type *propose*).

Le seul échange qui n'a pas pu être classé dans l'un des quatre types sus-cités est constitué d'un unique message *thank*. Ce remerciement ne porte pas sur une seule réponse donnée mais sur l'ensemble de l'aide apportée. Il ne peut donc pas être intégré à un échange existant.

Au cours des 416 échanges recensés, les sujets ouvrent 60 dialogues incidents par l'émission d'une performative de type *refine*. Un seul dialogue incident est d'ordre deux, tous les autres étant d'ordre un. Le seul dialogue incident d'ordre deux est celui constitué des deux messages non annoté (voir SEC. 3.1.1).

Pour compléter cette étude statistique, il faut ajouter quelques informations importantes qui n'apparaissent pas dans le tableau précédent.

- Longueur moyenne des échanges : 2,65
- Nombre maximum d'échanges ouverts en même temps par trinôme : 10
- Nombre moyen d'échanges ouverts en même temps par trinôme : 3,51
- Nombre maximum d'échanges ouverts en même temps par sujet : 9
- Nombre moyen d'échanges ouverts en même temps par sujet : 2,60

La longueur moyenne des échanges est faible. Cela s'explique par les nombreux échanges composés uniquement d'une question et de sa réponse. Dans une moindre mesure, les messages initiant un échange mais ne recevant aucune réponse y contribuent aussi.

Le nombre moyen d'échanges ouverts simultanément, que ce soit par trinôme ou par agent, est assez réduit. La plupart du temps, n'ont aucune difficulté pour savoir à quel échange un message appartient. Cependant, les sujets peuvent gérer jusqu'à neuf échanges en même temps, sans éprouver de difficulté.

### 3.2.5 Discussion

Cette seconde analyse permet de faire ressortir la dynamique des conversations humaines. Les messages en relation avec un message initiateur sont regroupés en échanges. Le type de la performative initiatrice indique l'intention discursive de l'échange.

Ces échanges permettent de représenter l'enchaînement des performatives émises et reçues par les sujets durant la résolution de leur problème. Ainsi, toute modélisation de dialogue entre sujets humains, si elle souhaite refléter la réalité le plus fidèlement possible, doit tenir compte

de la temporalité et de l'existence de dialogue incident.

### **3.3 Conclusion**

Cette analyse des protocoles expérimentaux du point de vue de l'interaction s'est concentrée tout d'abord sur les énoncés pris indépendamment les uns des autres. Chaque énoncé a pu être mis sous la forme d'une performative prise dans une liste obtenue par observation des conversations.

La façon dont s'articulent les énoncés les uns par rapport aux autres reflète la dynamique des conversations humaines. Ce travail s'est inspiré de la théorie discursive de Vanderveken, certaines performatives ont été regroupées en échanges.

En raison de l'importance de la temporalité et de l'existence de dialogues incidents, KQML et FIPA-ACL, qui privilégient l'utilisation de protocoles de communication rigides, paraissent inadaptés pour la modélisation de conversations humaines.



# Chapitre 4

## Modélisation des interactions humaines

Ce chapitre décrit les modèles formels permettant de représenter les messages et leurs enchaînements tout en tenant compte de la complexité des dialogues humains et de la dynamique conversationnelle.

Ces modèles respectent les caractéristiques mises en évidence lors de l'analyse des protocoles tout en restant implantables en machine, en vue d'une simulation.

Après avoir défini formellement le Problème de l'Agence de Voyage, le modèle de la dynamique conversationnelle sera proposé avant de donner la représentation des énoncés.

### 4.1 Description formelle du Problème de l'Agence de Voyage

Voici un bref rappel du Problème de l'Agence de Voyage, dont la résolution est modélisée par la suite.

Trois employés d'une agence de voyage possèdent des compétences complémentaires : le premier est spécialisé dans les transports aériens, le second dans les transports ferroviaires et le dernier dans les transports routiers (bus et taxis). Chacun d'eux est chargé d'organiser un voyage pour un client. Ces voyages sont caractérisés par une ville de départ et une ville d'arrivée en France, une date et une heure de départ et une date et une heure d'arrivée, un nombre de voyageurs pour la réservation et un budget.

Un sujet a résolu son problème s'il a réussi à construire un voyage, décomposé en étapes, reliant la ville de départ et la ville d'arrivée, et respectant les contraintes suivantes :

- les villes de départ et d'arrivée sont identiques à celles du problème soumis (contrainte de lieu),
- les dates et heures de départ et d'arrivée respectent les contraintes du problème (contrainte de temps),
- les étapes s'enchaînent de façon à ce que la ville de départ d'une étape corresponde exactement à la ville d'arrivée de l'étape précédente (contrainte de transfert),
- les étapes s'enchaînent de façon à ce que l'heure d'arrivée d'une étape soit inférieure à l'heure de départ de l'étape suivante (contrainte de compatibilité),
- toutes les étapes ont été réservées pour le bon nombre de personnes (contrainte de réservation),

- le coût du voyage est inférieur ou égal au budget alloué (contrainte de prix).

### 4.1.1 Syntaxe utilisée

Le Problème de l'Agence de Voyage et les contraintes à satisfaire peuvent être décrits formellement à l'aide des entités suivantes : les *valeurs*, les *ensembles de valeurs*, les *variables*, les *termes*, les *prédicats* et les *actions*.

Soit *VAL* l'ensemble des *valeurs* que peuvent prendre les variables et les constantes utilisés dans la définition du Problème de l'Agence de Voyage. *VAL* contient toutes les villes, les horaires, les moyens de transport, etc.

Soit *SOV* l'ensemble des ensembles de valeurs. Les *ensembles de valeurs* peuvent porter sur des valeurs numériques (dans ce cas il s'agit d'intervalles) ou sur des valeurs symboliques.

Les *variables* du problème sont les données qui peuvent changer de valeur au cours de la résolution du problème, selon les besoins des agents. Elles prennent leurs valeurs dans *VAL*, tout en conservant le même type tout au long de la résolution. Soit *VAR* l'ensemble des variables du problème. Par exemple,  $x \in TIM$  est une variable de type horaire, pouvant prendre n'importe quelle valeur dans *TIM* au cours de la résolution.

Les *termes* sont les données pouvant être une valeur, un ensemble de valeurs ou une variable. Soit *TER* l'ensemble des termes du problème,  $TER = VAL \cup SOV \cup VAR$ .

Un *prédicat* est la représentation formelle d'une proposition décrivant une partie du monde (*i.e.* de l'interface, du problème ou d'un agent) durant la résolution. Soit *PRE* l'ensemble des *prédicats*. Un nom de prédicat débute par "p" et se présente sous une forme fonctionnelle, appliquée à une série de termes et/ou de prédicats, éventuellement vide. Par exemple, le prédicat  $pStage(Paris\ Orleans\ 10:00\ 11:05\ train\ 1\ 80\ true) \in PRE$  décrit une étape en train non réservée, entre Paris et Orléans, partant à 10h00 et arrivant à 11h05, coûtant 80F pour une personne.

La négation d'un prédicat est représentée par un !, au détriment de l'habituelle négation  $\neg$ , pour pouvoir être utilisée dans les protocoles expérimentaux (fichiers codés en ASCII).

Dans un prédicat, un terme peut être remplacé par ? ou par  $?(x_1 \dots x_n)$  pour indiquer que la valeur est recherchée par un agent. Dans le premier cas, il n'y a aucune contrainte sur la valeur alors que dans le second cas la valeur doit être choisie parmi l'ensemble  $\{x_1, \dots, x_n\}$ . Le prédicat  $pStage(Orleans\ ?\ ?(Paris(gare)\ Paris(aero))\ ?\ ?\ train\ 1\ ?\ false)$  signifie que les horaires et les prix des trains entre Orléans et Paris (si la ligne existe) sont recherchés et qu'il est nécessaire de préciser si la destination exacte est la gare ou de l'aéroport de Paris. Cette substitution de terme permet d'augmenter l'expressivité des messages qui pourront être échangés entre les agents.

Enfin, *ACT* est l'ensemble des *actions* que peuvent effectuer les agents au cours de la résolution du problème. Ces actions peuvent être internes (leur portée se limite à l'agent qui les réalise) ou externes (elles sont effectuées via l'interface) d'une part et élémentaire ou abstraite (elle se subdivise en une série d'actions élémentaires ou abstraites) d'autre part. Un nom d'action débute par "a". Par exemple, l'action abstraite  $aLookForTimetable(Paris\ Orleans\ plane(PigeonVol))$  qui permet de rechercher les horaires d'avions entre Paris et Orléans, pour la compagnie Pigeon-Vol, se décompose en quatre actions élémentaires externes :  $aSelectCityDeparture(Orléans)$ ,  $aSelectCityArrival(Paris)$ ,  $aSelectCompany(PigeonVol)$  et  $aDisplayTimeTables()$ .

## 4.1.2 Termes et ensembles de termes

Soit  $CIT$ , l'ensemble des villes considérées pour le problème. Ces villes sont celles présentes sur les cartes fournies aux sujets et dans les bases de données.  $CIT = \{AixLesBains, Amiens, Amiens(gare), Angers, Angers(gare), Annecy, Annecy(aero), Annecy(gare), Baisieux, Beaupreau, Bordeaux, Brest, Caen, Chambery, Chambery(gare), ChateauNeuf, ClermontFerrand, Dijon, Grenoble, Grenoble(aero), Grenoble(gare), LeChatelard, Lille, Lille(aero), Lille(gare), Lyon, Maguelone, Marseille, Marseille(aero), Marseille(gare), Metz, Montpellier, Montpellier(aero), Montpellier(gare), Nancy, Nancy(gare), Nantes, Nantes(aero), Nantes(gare), Nice, Nice(aero), Nice(gare), Orleans, Orleans(gare), Paris, Paris(aero), Paris(gare), Poitiers, PontAMousson, Reims, Rennes, Rennes(gare), Rouen, SaintMartin, SophiaAntipolis, Strasbourg, Strasbourg(aero), Strasbourg(gare), Tinteniac, Toul, Toul(gare), Toulouse, Toulouse(aero), Toulouse(gare), Tours, Vallauris\}$ .

Soit  $TIM$ , l'ensemble des horaires, écrits sous la forme  $hh:mm$  avec  $0 \leq hh \leq 23$  et  $0 \leq mm \leq 59$ .  $TIM$  contient de plus un marqueur particulier permettant d'exprimer une durée. Ainsi,  $\forall x \in TIM, +(x) \in TIM$  représente la durée  $x$ . Par exemple,  $pStage(Paris(gare) Paris(aero) * +(00:35) taxi 1 100 *)$  indique que le trajet en taxi entre la gare et l'aéroport de Paris dure trente-cinq minutes.

Soit  $MEA$ , l'ensemble des moyens de transport,  $MEA = \{train, plane, plane(PigeonVol), plane(PachAir), bus, coach\}$ .

Pour l'agent aérien, soit l'ensemble des compagnies aériennes  $AER = \{PigeonVol, PachAir\}$ .

Pour l'agent routier, soit l'ensemble des moyens de transport  $ROA = \{bus, coach\}$ .

L'agent ferroviaire ne dispose que d'une seule compagnie sur un seul moyen de transport, le train.

Soit  $AGT$ , l'ensemble des spécialités des agents/sujets,  $AGT = \{air, railway, road\}$ .

Les ensembles de valeurs suivant sont définis :

- $\geq(x) \Leftrightarrow [x; +\infty[$ . Par exemple,  $\geq(08:00)$  représente tous les horaires après 8h00.
- $\leq(x) \Leftrightarrow [0;x]$ . Ainsi pour un prix,  $\leq(100)$  correspond aux prix inférieurs ou égaux à 100F.
- $\geq\langle(x y) \Leftrightarrow [x;y]$ . Par exemple,  $\geq\langle(21:00 22:00)$  désigne les horaire compris entre 21h00 et 22h00.
- $\sim(x) \Leftrightarrow [x-c;x+c]$  avec  $c$  une valeur constante. Cet intervalle sert à représenter la notion *environ*. Par exemple, "environ 500F" s'exprime par  $\sim(500)$ .
- $OR(x_1 \dots x_n)$  correspond à une disjonction de valeurs.
- $*$  remplace n'importe quelle valeur (numérique ou symbolique).

Enfin, pour favoriser la lecture des intervalles d'horaires, soient les espaces de temps suivants et leur intervalle équivalent :

- $early \Leftrightarrow \leq(08:00)$ ,
- $morning \Leftrightarrow \geq\langle(08:00 12:00)$ ,
- $afternoon \Leftrightarrow \geq\langle(12:00 20:00)$ ,
- $evening \Leftrightarrow \geq(20:00)$ .

### 4.1.3 Prédicats et ensembles de prédicats

Une étape est décrite par le prédicat :

$$pStage(C_D C_A T_D T_A M N P R)$$

avec  $C_D \in CIT$  la ville de départ,  $C_A \in CIT$  la ville d'arrivée,  $T_D \in TIM$  l'horaire de départ,  $T_A \in TIM$  l'horaire d'arrivée,  $M \in MEA$  le moyen de transport,  $N \in \mathbb{N}^*$  le nombre de personnes,  $P \in \mathbb{N}^*$  le prix du trajet pour une personne et  $R \in \{true, false\}$  un booléen indiquant si le trajet est réservé (*true*) ou non (*false*).  $STA$  est l'ensemble des étapes. Comme n'importe quel autre prédicat, chaque terme d'une étape peut être remplacée par \* et ?.

Un voyage est représenté par le prédicat :

$$pTravel(S_{1,1}[l\dots lS_{1,n}] \dots S_{t,1}[l\dots lS_{t,m}])$$

avec  $S_{i,j} \in STA, \forall i, j \in \mathbb{N}^*$ . Chacune des étapes considérées a été insérée dans le panneau "Propositions" de l'interface. Les  $S_{k,l}$  pour  $k \in \mathbb{N}^*$  fixé et  $\forall l \in \mathbb{N}^*$  représentent les sous-buts parallèles pour une même étape. Dans le cas où l'agent fait de la planification parallèle, il utilise un prédicat de type *pTravel* pour chacun de ses plans parallèles.  $TRA$  est l'ensemble des voyages.

Un problème est décrit par le prédicat :

$$pProblem(A B)$$

où  $A \in AGT$  et  $B \in \{true, false\}$  avec *true* si le problème est résolu et *false* s'il ne l'est pas. Ce prédicat ne décrit aucunement les contraintes du problème (villes de départ et d'arrivée, horaires imposés, etc.) mais indique seulement si un agent a résolu son problème.  $PRO$  est l'ensemble des problèmes.

Enfin, soit  $MES$  l'ensemble des messages échangés entre les agents durant la simulation. Le format des prédicats de type message est décrit plus précisément en 4.3.

### 4.1.4 Actions élémentaires externes

Voici l'ensemble des actions externes, c'est-à-dire l'ensemble des actions qu'un agent peut effectuer sur l'interface. Ces actions correspondent uniquement à celles réellement effectuées par les sujets humains durant l'expérimentation. Cette liste d'actions n'est donc pas exhaustive par rapport aux possibilités de l'interface.

- **Dans le panneau "Horaire/Prix"**

- `aSelectCityDeparture(C)` avec  $C \in CIT$  : sélectionne une ville de départ pour une étape.
- `aSelectCityArrival(C)` avec  $C \in CIT$  : sélectionne une ville d'arrivée pour une étape.

- `aSelectCompany(C)` avec  $C \in AER \cup ROA$  : sélectionne une compagnie aérienne pour un trajet aérien ou le moyen de transport pour un trajet routier. Cette action est impossible pour l'agent ferroviaire qui ne dispose pas du menu déroulant correspondant dans son interface.
- `aDisplayTimeTables()` : affiche la liste des trajets répondant aux critères fournis dans les menus déroulants servant à choisir une ville de départ, une ville d'arrivée et éventuellement une compagnie.
- `aSelectStageInTimetable(S)` avec  $S \in STA$  : sélectionne un des trajets de la liste du panneau "Horaire/Prix". La liste des trajets est en réalité une liste d'horaires possibles, associés à un prix pour une personne.
- `aKeyInNbTravelers(N)` avec  $N \in \mathbb{N}^*$  : saisit le nombre de voyageurs.
- `aMemorizeFromDatabase()` : ajoute le trajet décrit par les informations provenant des différents éléments du panneau "Horaire/Prix" au problème sélectionné de la "Zone de travail".

- **Dans le panneau "Zone de travail"**

- `aSelectProblem(A)` avec  $A \in AGT$  : sélectionne un des espaces réservés à chacun des problèmes. Dans l'interface, cette sélection se fait par un choix entre "Mon problème", "Pb. ag. X" et "Pb. ag. Y", avec  $X, Y \in AGT$  les deux autres agents.
- `aSelectStageInWorkingPanel(S)` avec  $S \in STA$  : sélectionne un des trajets de la liste de la "Zone de travail".
- `aBookOrCancelStage()` : réserve ou annule la réservation du trajet sélectionné.
- `aDeleteStageFromWorkingPanel()` : supprime le trajet sélectionné de la liste de la "Zone de travail".
- `aInsertStageInPropositionPanel()` : ajoute le trajet sélectionné au panneau "Propositions".
- `aSendStage()` : insère le trajet sélectionné dans un message pour l'envoyer à un autre agent.

- **Dans le panneau "Messagerie"**

- `aSelectMessage(M)` avec  $M \in MES$  : sélectionne un des messages du panneau "Messagerie".
- `aMemorizeFromMessage()` : extrait l'horaire du message sélectionné dans le panneau "Messagerie" et l'ajoute à la "Zone de travail" active.
- `aRemoveMessage()` : supprime le message sélectionné. Cette action correspond à un simple appui sur le bouton "Supprimer" du panneau "Messagerie".
- `aNewMessage()` : crée un nouveau message vide dans une nouvelle fenêtre. Cette action correspond à un simple appui sur le bouton "Nouveau Message" de l'interface.
- `aReply()` : crée un nouveau message dans une nouvelle fenêtre en réponse au message sélectionné. Cette action correspond à un simple appui sur le bouton "Répondre" de l'interface.
- `ASend(M)` avec  $M \in MES$  : envoie le message M. Cette action remplit le contenu du message et appuie sur le bouton "Envoi" se trouvant dans la nouvelle fenêtre ouverte après utilisation de "Nouveau Message" ou de "Répondre".

- **Dans le panneau "Propositions"**

- `aSelectStageInProposalPanel(S)` avec  $S \in STA$  : sélectionne une des étapes de la liste du panneau "Propositions".
- `aMoveUp()` : déplace l'étape sélectionnée vers le haut de la liste.
- `aMoveDown()` : déplace l'étape sélectionnée vers le bas de la liste.
- `aInsert()` : appuie sur le bouton "Insérer".

- `aModify()` : appuie sur le bouton "Modifier".
- `aDeleteFromProposalPanel()` : supprime l'étape sélectionnée dans le panneau "Proposition".
- `aTest()` : lance le test de la proposition.

#### 4.1.5 Contraintes formelles pour la résolution du problème

Les constantes suivantes sont définies pour décrire un problème soumis à un agent :

- $departureCity, arrivalCity \in CIT$  sont les villes de départ et d'arrivée,
- $departureTime, arrivalTime \in TIM$  sont les heures de départ minimale et d'arrivée maximale,
- $nbTravelers \in \mathbb{N}^*$  est le nombre de personnes pour lesquelles il faut réserver le voyage,
- et  $budget \in \mathbb{N}^*$  est le budget alloué pour le voyage.

Soumettre un problème à un agent revient à lui transmettre l'ensemble  $\{departureCity, arrivalCity, departureTime, arrivalTime, nbTravelers, budget\}$  ainsi que la description de la carte des transports dont il se charge. Chaque agent reçoit donc la description de son problème sous la forme d'un  $pStage$ .

- Agent aérien :  $pStage(ChateauNeuf\ Maguelone\ * * * 4 \leq (3300)\ true)$ .
- Agent ferroviaire :  $pStage(Beaupreau\ Baisieux\ * \leq (11 : 00) * 1 \leq (1310)\ true)$ .
- Agent routier :  $pStage(SaintMartin\ LeChatelard\ > (17 : 30) * * 3 \leq (1430)\ true)$ .

Les contraintes qu'une proposition doit vérifier pour être solution d'un problème peuvent maintenant être formellement écrites.

Soit  $pT = pTravel(S_1 \dots S_n)$  la proposition avec  $\forall i \in \mathbb{N}^*, i \leq n, S_i \in STA$  est décrite par  $pStage(C_{i,D} C_{i,A} T_{i,D} T_{i,A} M_i N_i P_i R_i)$ . Si  $pT$  ne se présente pas exactement sous la forme précédente (i.e. si  $pT$  contient des sous-buts) alors  $pT$  ne peut pas être solution du problème. De plus, pour être solution du problème,  $pT$  doit vérifier les contraintes suivantes :

- Contrainte de lieu :  $\begin{cases} C_{1,D} = departureCity \\ C_{n,A} = arrivalCity \end{cases}$
- Contrainte de temps :  $\begin{cases} (departureTime = *) \mid (T_{1,D} \geq departureTime) \\ (arrivalTime = *) \mid (T_{n,A} \leq arrivalTime) \end{cases}$
- Contrainte de transfert :  $\forall i \in \mathbb{N}^*, i > 1, C_{i,D} = C_{i-1,A}$
- Contrainte de compatibilité :  $\forall i \in \mathbb{N}^*, i > 1, T_{i,D} \geq T_{i-1,A}$
- Contrainte de réservation :  $\forall i \in \mathbb{N}^*, \begin{cases} N_i = nbTravelers \\ R_i = true \end{cases}$
- Contrainte de prix :  $(\sum_{i=1}^n P_i) \leq budget$

Par exemple, pour le problème de l'agent ferroviaire, qui est :

$$pStage(Beaupreau\ Baisieux\ * \leq (11 : 00) * 1 \leq (1310)\ true),$$

voici une des solutions :

*pTravel( pStage(Beaupreau Nantes(aero) 07:00 07:35 taxi 1 250 true)  
pStage(Nantes(aero) Lille(aero) 08:05 09:35 avion 1 802 true)  
pStage(Lille(aero) Baisieux 10:00 10:05 taxi 1 170 true) ).*

#### **4.1.6 Discussion**

La syntaxe proposée permet de décrire l'environnement de résolution, les objets manipulés par les agents, même abstraits (voyage, étape, problème, etc.) ainsi que les contraintes qui doivent être vérifiées pour qu'une proposition de voyage soit solution du problème.

Cependant ce modèle n'intègre pas les éléments permettant d'explicitier les mécanismes de résolution. Il faut, pour cela, lui ajouter les objets formels représentant l'état interne des agents : les états mentaux et les outils nécessaires pour modéliser les dialogues humains par courrier électronique.

## **4.2 Modélisation de la dynamique conversationnelle**

Nous souhaitons modéliser la dynamique des conversations humaines observée dans les protocoles expérimentaux ainsi que les problèmes d'interprétation d'un énoncé quand celle-ci n'est pas immédiate pour l'allocataire.

Dans le cadre de notre expérimentation, pour laquelle les messages sont des courriers électroniques, le facteur temps est primordial pour les relances de demande et la terminaison d'un échange. Par exemple, lors d'une demande d'information, si le locuteur ne reçoit pas de réponse suffisamment vite à son goût, il peut réitérer sa demande. De plus, les échanges ne présentant pas d'acte de langage de clôture seront considérés comme terminés après écoulement d'un certain délai.

Par ailleurs, tant que les interlocuteurs ne rencontrent pas de difficultés d'interprétation d'un énoncé (un non-attendu), ils conduisent l'échange courant comme un dialogue régissant. Sinon, ils initient un dialogue incident pour résoudre le problème apparu, en recherchant une interprétation commune. Quand cette interprétation commune apparaît, ils réactivent le dialogue régissant mis en suspens.

Il est donc nécessaire d'avoir un formalisme permettant de représenter la séquentialité et la temporalité des messages envoyés durant la résolution du problème. Il faut aussi conserver des protocoles de communication suffisamment souples pour permettre de simuler les capacités humaines d'interaction.

Parmi les formalismes étudiés, notre choix s'est porté sur l'utilisation des automates temporisés [Alur et Dill, 1994] pour représenter les échanges.

### **4.2.1 Automates temporisés**

Les automates à états finis permettent de modéliser l'ordonnancement temporel d'actions sans donner d'indication quantitative sur le délai séparant deux actions. Les automates temporisés ont été proposés par [Alur et Dill, 1994] pour répondre à ce besoin. Ils permettent par

exemple

- de préciser qu'un automate donné ne reste pas plus de 3 minutes dans un certain état ;
- de modifier par la suite la valeur de la constante 3, sans avoir à transformer la structure du modèle.

Un automate temporisé se compose de deux éléments fondamentaux :

1. un automate à états finis qui décrit les états de contrôle du système et les transitions, supposées instantanées, entre ces états
2. des horloges (ou chronomètres), utilisées pour spécifier des contraintes quantitatives de temps, qui peuvent être associées aux transitions de l'automate à états finis.

Dans ce modèle, les transitions restent instantanées, le "temps" ne s'écoulant qu'entre les transitions.

Les horloges sont des variables à valeurs réelles positives. Dans l'état initial du système, elles valent toutes 0 et progressent ensuite à la même vitesse, de manière synchrone avec le temps. À noter que ce modèle permet aussi de gérer des variables n'étant pas des horloges.

À une transition sont associés trois éléments :

1. une *garde* (aussi appelé *condition de franchissement*), portant sur les valeurs des horloges (ou des variables),
2. une *étiquette* représentant une action, comme dans les automates à états finis,
3. des *actions de remise à zéro* de certaines horloges.

Naturellement, une transition donnée ne peut être franchie que si la condition associée est vérifiée par les valeurs courantes des horloges.

La figure 4.1 présente un exemple d'automate temporisé. La transition initiale met à zéro l'horloge  $t$  de l'automate. La transition entre  $s_1$  et  $s_2$ , est franchie lorsque que 10 unités de temps se sont écoulées, ce qui conduit le système à envoyer une alarme. L'automate est alors fermé, l'état  $s_3$  étant un état terminal.

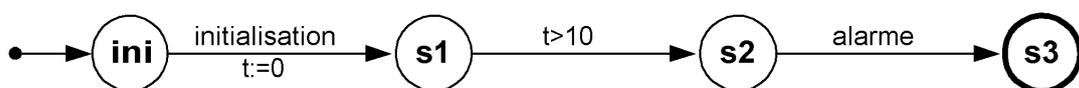


FIG. 4.1 – Exemple d'automate temporisé

Le système fonctionne donc comme s'il possédait une horloge globale, mesurant l'écoulement du temps, et sur laquelle chacune des horloges est synchronisée, entre deux remises à zéro.

## 4.2.2 Modélisation des échanges par automates temporisés

Huit (4x2) automates ont été construits, un pour le locuteur et un pour l'interlocuteur de chaque type d'échange (voir FIG. 4.2, FIG. 4.3, FIG. 4.4 et FIG. 4.5). Chaque échange aurait

pu être représenté avec un automate. Cependant, utiliser deux automates par échange permet de mettre en avant le point de vue de chaque interlocuteur.

Chacun de ces automates respecte le plus fidèlement possible les échanges observés dans les protocoles expérimentaux. La conception des automates temporisés a nécessité de nombreux allers-retours entre les phases d'analyse et de modélisation.

La description formelle des automates temporisés nécessite de compléter la syntaxe présentée dans la section précédente.

Soit  $PER$  l'ensemble des performatives. Elles correspondent à celles observées dans les protocoles expérimentaux et sont décrites en 3.1.1.  $PER = \{inform, reply, notUnderstood, thank, query, acceptProposal, refuseProposal, cancel, refine, propose\}$ .

Soit  $AUT$  l'ensemble des automates temporisés,  $AUT = \{Q_{ini}, Q_{int}, P_{ini}, P_{int}, I_{ini}, I_{int}, E_{ini}, E_{int}\}$ . Les automates temporisés sont nommés  $X_a$ , avec  $X \in \{Q, P, I, E\}$  le type de l'échange ( $Q$  : demande d'information,  $P$  : proposition d'information,  $I$  : envoi spontané d'information et  $E$  : traitement des erreurs) et  $a \in \{ini, int\}$  le protagoniste de l'échange ( $ini$  pour l'initiateur de l'échange et  $int$  pour son interlocuteur). Ainsi  $Q_{ini}$  est l'automate décrivant le comportement de l'agent initiateur d'un échange de type demande d'information et  $Q_{int}$  est son interlocuteur.

Soit  $SQ_{ini}, SQ_{int}, SP_{ini}, SP_{int}, SI_{ini}, SI_{int}, SE_{ini}, SE_{int}$  les ensembles d'états de chacun des automates. La syntaxe est la même que précédemment, c'est-à-dire que les noms des ensembles d'états de chacun des automates est de la forme  $SX_a$ , avec  $X \in \{Q, P, I, E\}$  le type de l'échange et  $a \in \{ini, int\}$  le protagoniste de l'échange. Par exemple,  $SQ_{ini}$  est l'ensemble d'états de l'automate de l'initiateur d'une demande d'information.

Soit  $SAU$  l'ensemble des états des automates,  $SAU = SQ_{ini} \cup SQ_{int} \cup SP_{ini} \cup SP_{int} \cup SI_{ini} \cup SI_{int} \cup SE_{ini} \cup SE_{int}$ .

La situation dans un automate est décrite par le prédicat  $pAutomate(A S)$ , avec  $A \in AUT$  et  $S \in SAU$ . Par exemple,  $pAutomate(Q_{ini} sI)$  signifie que l'agent a activé un automate de l'initiateur d'une demande d'information, et que cet automate se trouve dans l'état  $sI$ .

Les quatre sections suivantes décrivent les automates temporisés de l'initiateur et de son interlocuteur pour chaque type d'échange. L'état initial est représenté par une flèche pointant sur un état  $ini$ . Les états terminaux sont dessinés en cercles gras. Pour chaque état terminal, il est précisé si l'échange est satisfait (*Satisfaction*) ou non (*Unsatisfaction*). Les transitions entre deux états portent deux étiquettes : une étiquette au-dessus de la flèche de transition qui donne les pré-conditions pour changer d'état et une étiquette au-dessous de la flèche qui précise les éventuelles modifications que subissent les variables internes de l'automate. Une transition entre deux états s'effectue :

- si un message est reçu : l'étiquette de la transition est alors de type  $Receive(perf)$  avec  $perf \in PER$  le type de la performativité reçue ;
- si un message est envoyé : l'étiquette de la transition est alors de type  $Send(perf)$  avec  $perf \in PER$  le type de la performativité envoyée ;
- si une condition portant sur une des variables internes de l'automate est vérifiée ;
- si une conjonction ou une disjonction d'éléments d'un des trois types précédents est vérifiée.

Enfin, dans la description des automates,  $A_{ini}$  désigne l'agent initiateur de l'échange et  $A_{int}$  son interlocuteur.

Pour chaque automate, un tableau statistique des fréquences d'apparition de chaque situation (état d'un automate) et de chaque transition durant les conversations des protocoles expérimentaux a été dressé. Les états des automates portent leur nombre d'apparition en étiquette. De même, pour chaque transition entre deux états, sa fréquence observée dans les conversations est précisé.

### 4.2.3 Automates de demande d'information

Les deux automates de demande d'information sont intitulés  $Q_{ini}$  pour celui décrivant le comportement de l'initiateur de l'échange et  $Q_{int}$  pour celui décrivant le comportement de son interlocuteur (voir FIG. 4.2).

Pour la gestion du temps,  $Q_{ini}$  et  $Q_{int}$  contiennent chacun une horloge  $t$  respectant un délai  $tsync$  avant d'effectuer une relance de demande ou de clôturer l'échange. L'automate  $Q_{ini}$  contient, de plus, un compteur  $m$  permettant de dénombrer les demandes d'information effectuées avant de recevoir (éventuellement) une réponse de la part de l'interlocuteur. Si  $m$  est trop élevé ( $m > msup$ ) alors  $A_{ini}$  abandonne l'échange et ferme l'automate.

Dans l'automate  $Q_{ini}$ ,  $A_{ini}$  fait sa demande d'information en envoyant un *query* (état s1). Il peut recevoir une réponse sous la forme d'un ou plusieurs *reply* (état s2). Si cette(ces) réponse(s) est(sont) satisfaisante(s), il peut terminer l'échange sans prévenir  $A_{int}$  ou le remercier par un *thank* (état s3). L'échange est alors satisfait, même sans clôture explicite. Par contre, si la réponse ne satisfait pas  $A_{ini}$ , il abandonne l'échange explicitement en envoyant un *cancel* à  $A_{int}$  (état s6), ou il reformule sa question avec un nouveau *query* (état s1). Par ailleurs  $A_{int}$  peut s'être trompé dans sa réponse et  $A_{ini}$  est prévenu par un *cancel*. En s1, si  $A_{ini}$  attend un temps  $tsync$  sans recevoir de réponse de la part d' $A_{int}$  (état s4), il incrémente son compteur  $m$  et repose sa question auprès d' $A_{int}$  en lui renvoyant un *query*. Si cela arrive plusieurs fois (état s5), il abandonne l'échange qui ne sera pas satisfait. Si  $A_{ini}$  n'a plus besoin de l'information, il envoie un message d'annulation à  $A_{int}$  sous la forme d'un *cancel* (état s6). Si  $A_{ini}$  reçoit de la part d' $A_{int}$  une demande de précision à propos de sa question par un *refine* (état s7), l'échange entre alors dans un dialogue incident de demande d'information. Ce dialogue peut être satisfait par la réception d'un *query* et l'échange courant est réactivé (état s1), ou alors échouer et  $A_{ini}$  doit abandonner sa requête explicitement en envoyant un *cancel* (état s6) ou ne pas le prévenir. Par ailleurs, il se peut qu'à la suite d'événements extérieurs à la conversation,  $A_{int}$  se trouve en mesure de satisfaire à la demande d' $A_{ini}$  avant d'obtenir une réponse à sa demande de précision. Il le fait par l'envoi d'un message de type *reply* (état s2). En s1,  $A_{int}$  peut aussi signifier à  $A_{ini}$  qu'il ne comprend pas sa demande en utilisant un *notUnderstood* (état s8). Dans ce cas,  $A_{ini}$  peut reformuler sa question par un *query* (état s1). Si par contre il s'était trompé dans sa demande, l'échange est abandonné explicitement (envoi d'un *cancel*, état s6) ou non par  $A_{ini}$ , il n'est alors pas satisfait. (FIG. 4.2,  $Q_{ini}$ ).

Parallèlement, dans  $Q_{int}$ ,  $A_{int}$  reçoit la question d' $A_{ini}$  sous la forme d'un *query* (état s1). S'il n'y répond pas assez vite, il reçoit une ou plusieurs demande(s) de relance, toujours sous la forme de *query*, et reste en s1.  $A_{int}$  peut répondre à la question avec autant de *reply* que nécessaire (état s2). Il peut alors être remercié par  $A_{ini}$  (*thank*) ou considérer l'échange comme terminé après écoulement du délai  $tsync$  (état s3). L'échange est alors satisfait. Par contre, si cet(ces) réponse(s) ne satisfait(satisfont) pas  $A_{ini}$ , celui-ci reformule sa question et  $A_{int}$  recevra alors un *query* (état s1).  $A_{ini}$  peut aussi décider d'abandonner sa demande d'information en envoyant un *cancel* (état s6). Dans ce cas, l'échange n'est pas satisfait. Par contre, si  $A_{int}$

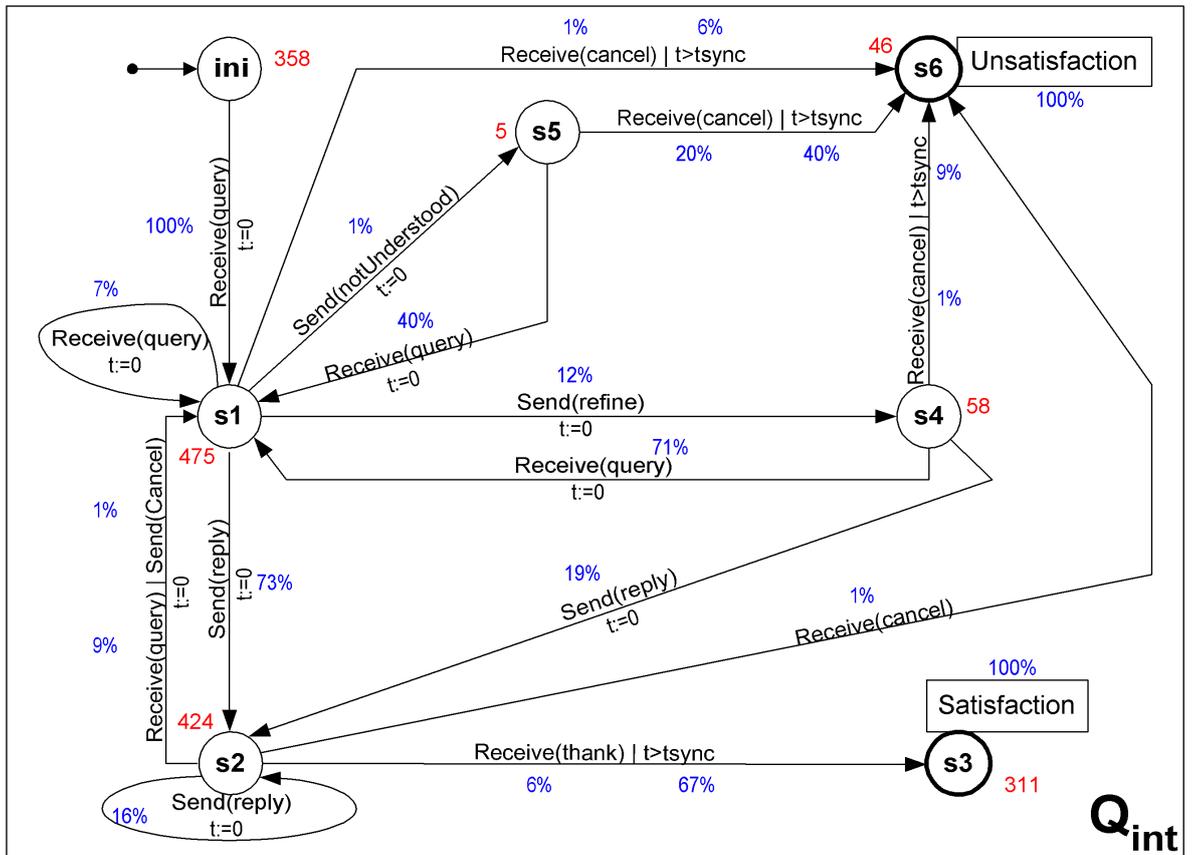
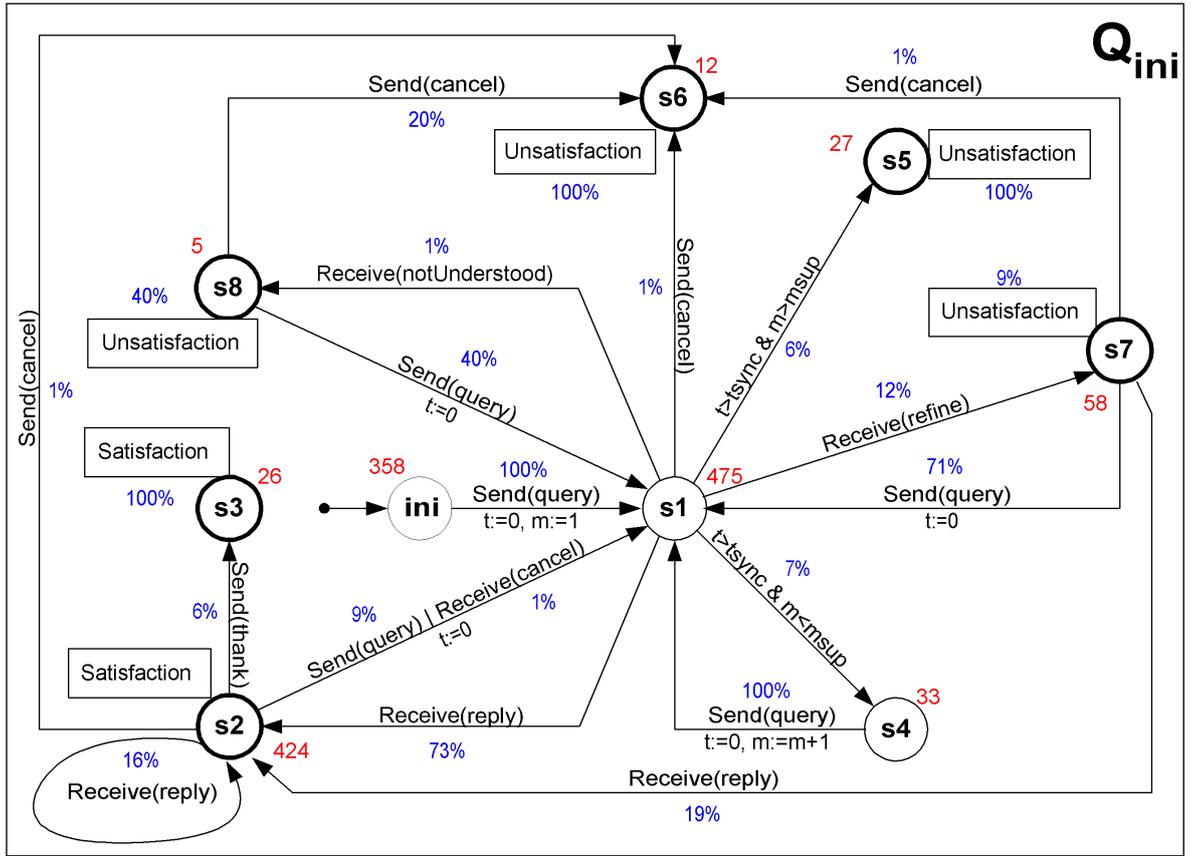


FIG. 4.2 – Automates de demande d'information

estime qu'il s'est trompé en répondant à  $A_{ini}$ , il le prévient par l'envoi d'un *cancel* (état s1). En s1, si  $A_{int}$  pense que la demande ne lui est pas adressée ou bien s'il ne la comprend pas, il le signifie à  $A_{ini}$  par un *notUnderstood* (état s4). De même, toujours en s1, si  $A_{int}$  juge la question incomplète et ne peut pas la satisfaire, il demande par un *refine* à  $A_{ini}$  de préciser les informations manquantes (état s5). Dans les deux cas,  $A_{ini}$  peut reformuler la demande avec un *query* (état s1). Un *refine* étant souvent envoyé alors que les informations manquantes peuvent être obtenues par ailleurs, si  $A_{int}$  peut tout de même satisfaire la demande d' $A_{ini}$  sans l'aide de ce dernier, il lui envoie un *reply*. En s1, s4 et s5, si  $A_{ini}$  abandonne sa demande initiale, il prévient  $A_{int}$  par un *cancel* (état s6) ou ferme l'échange sans l'avertir. Dans ce dernier cas,  $A_{int}$  considérera l'échange comme terminé même sans clôture explicite s'il ne reçoit aucun message d' $A_{ini}$  pendant un délai *tsync* (état s6). Que l'échange soit terminé explicitement ou non, il sera insatisfait. (FIG. 4.2,  $Q_{int}$ ).

#### 4.2.4 Automates de proposition d'information

Les deux automates de proposition d'information sont intitulés  $P_{ini}$  pour modéliser le comportement de l'initiateur de l'échange et  $P_{int}$  pour modéliser le comportement de son interlocuteur (voir FIG. 4.3).

Pareillement à  $Q_{ini}$  et  $Q_{int}$ , l'automate  $P_{ini}$  contient une horloge  $t$  qui permet de le fermer s'il ne subit aucun changement d'état pendant un délai *tsync*.  $P_{int}$  est, quant à lui, un automate à états finis classique.

$A_{ini}$  initie une proposition d'information et ouvre un automate  $P_{ini}$  en envoyant à  $A_{int}$  un message de type *propose*. Si la proposition est refusée explicitement par un *refuseProposal* ou non (écoulement d'un délai *tsync*) l'échange n'est pas satisfait (état s2). Si la proposition n'avait pas été suffisamment explicite,  $A_{int}$  peut recevoir un *notUnderstood* (état s3). Il réitère alors sa proposition à l'aide d'un *propose* plus explicite (état s1). La proposition d'information peut être acceptée par  $A_{int}$ , ce qui se traduit par la réception d'un *acceptProposal* (état s4).  $A_{ini}$  peut alors envoyer l'information proposée à l'aide d'un *reply* (état s6). L'échange est satisfait explicitement avec la réception d'un remerciement (un *thank*) ou implicitement (après écoulement du délai *tsync*). Si la satisfaction de la proposition d'information nécessite plus de précisions,  $A_{ini}$  envoie un *refine* à  $A_{int}$  (état s5) qui lui répond par un *query* avant de retourner à l'état s4. (FIG. 4.3,  $P_{ini}$ ).

À la réception d'un *propose*,  $A_{int}$  active un automate  $P_{int}$  dans l'état s1. S'il juge que la proposition n'est pas intéressante pour résoudre son problème, il peut fermer l'automate directement (état s1), ou bien après avoir envoyé un refus, dans un message de type *refuseProposal*, à  $A_{ini}$  (état s2). Dans les deux cas, l'échange n'est pas satisfait. Si  $A_{int}$  ne comprend pas la proposition, il envoie à  $A_{ini}$  un *notUnderstood* (état s3) qui précisera sa proposition avec un *propose* (état s1). Si l'information proposée est jugée intéressante par  $A_{int}$ , il l'acceptera en envoyant un message de type *acceptProposal* (état s4). Il peut alors recevoir l'information proposé dans un *reply* (état s6). L'échange est satisfait et l'automate sera clôturé par  $A_{int}$  explicitement (état s7), ou en envoyant un remerciement (*thank*) à l'initiateur de l'échange (état s7). Par contre, il se peut que  $A_{ini}$  ait besoin de précisions pour envoyer l'information qu'il a proposée. Il les demande à  $A_{int}$  qui reçoit un *refine* (état s5) et y répond par un *query*. (FIG. 4.3,  $P_{int}$ ).

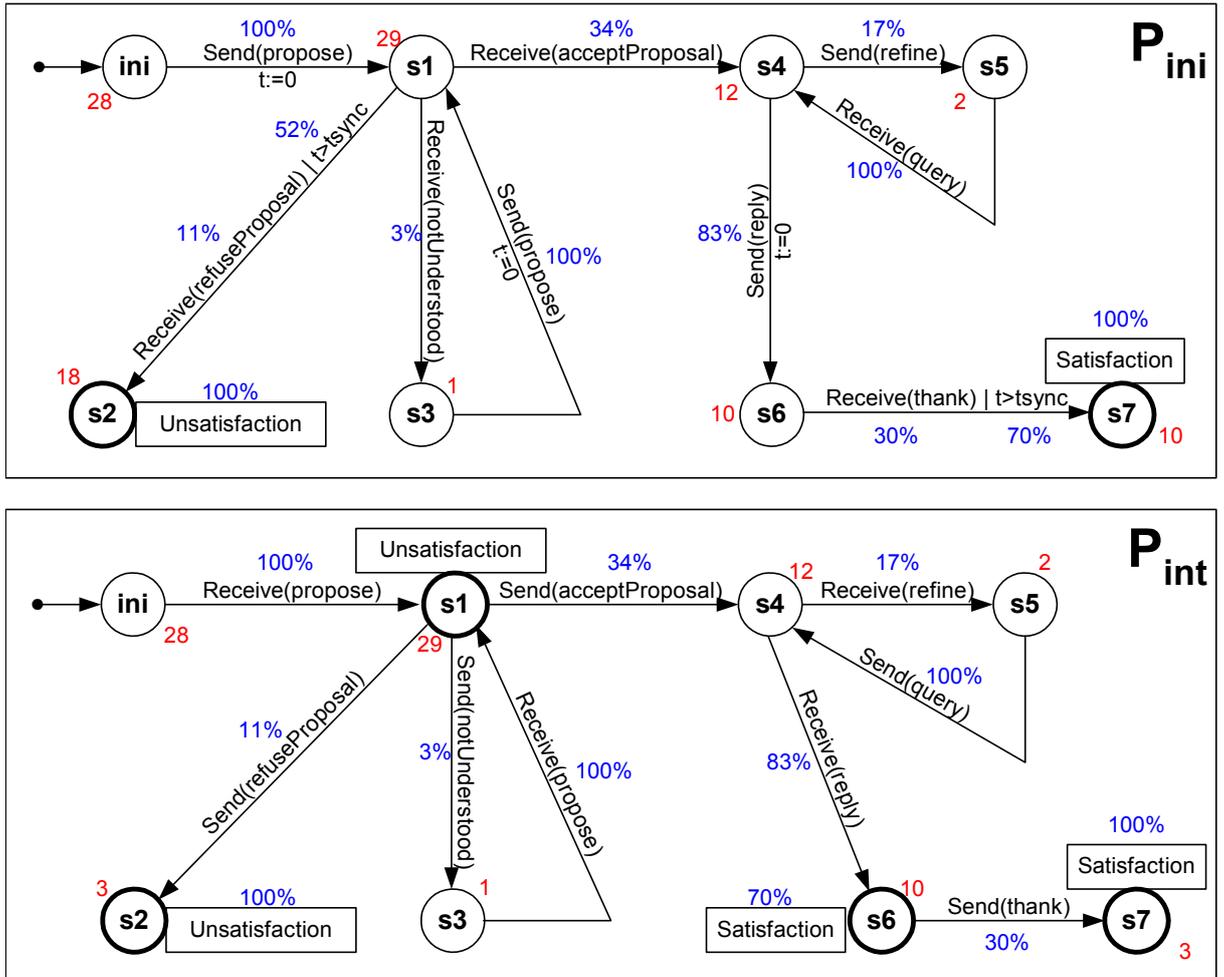


FIG. 4.3 – Automates de proposition d'information

## 4.2.5 Automates d'envoi spontané d'information

Les deux automates d'envoi spontané d'information sont intitulés  $I_{ini}$  pour modéliser le comportement de l'initiateur de l'échange et  $I_{int}$  pour modéliser le comportement de son interlocuteur (voir FIG. 4.4).

$I_{ini}$  et  $I_{int}$  ne contiennent que deux états, un état initial et un état terminal s1. Le premier est activé à l'envoi d'un message de type *query* par l'initiateur de l'échange et le second est activé par la réception de ce message par son interlocuteur.

Comme les deux automates d'envoi spontané d'information ne contiennent qu'une transition, une fois activés ils sont directement fermés. Ce sont les deux seuls automates représentant un échange qui soit toujours satisfait.

## 4.2.6 Automates de traitement des erreurs

Les deux automates de traitement des erreurs sont intitulés  $E_{ini}$  pour modéliser le comportement de l'initiateur de l'échange (l'agent envoyant un *notUnderstood* à la réception d'un message qui ne lui est pas adressé) et  $E_{int}$  pour modéliser le comportement de son interlocuteur (voir FIG. 4.5).

$E_{ini}$  contient une horloge  $t$  qui respecte le délai  $t_{sync}$  avant de clôturer l'automate si  $A_{int}$  n'a reçu aucune fermeture explicite.

$E_{ini}$  est ouvert par  $A_{ini}$  si celui-ci reçoit un message de type *reply*, *acceptProposal*, *refuseProposal*, *cancel*, *refine* ou *thank* alors qu'il ne se trouve pas dans une situation où il est censé en recevoir. Plus précisément, si aucun des automates qu'il a ouvert ne permet de gérer la réception d'un message de ce type, alors il ouvre un automate de traitement des erreurs pour lequel il sera l'initiateur (état s1). Il envoie alors à l'émetteur du message non-attendu un *notUnderstood* (état s2). Celui-ci répond avec un *cancel* permettant d'annuler le message erroné. L'automate de traitement des erreurs est fermé et l'échange est satisfait.

Pour  $A_{int}$ , l'automate de traitement des erreurs  $E_{int}$  est ouvert à la réception d'un message de type *notUnderstood*. Puis  $A_{int}$  envoie à  $A_{ini}$  un *cancel* pour annuler le message fautif (état s2). L'échange est satisfait. (FIG. 4.5,  $E_{int}$ ).

## 4.2.7 Discussion

Les automates temporisés permettent de modéliser finement la diversité et la dynamique des conversations humaines. À tout moment caractéristique d'un échange (un état de l'automate), les attendus sont représentés par des transitions dont l'enchaînement exprime la séquentialité des énoncés.

Chaque état est annoté par une valeur indiquant le nombre de fois où un sujet s'est retrouvé avec certaines attentes durant la conversation dans les protocoles expérimentaux. De même, chaque transition est annotée par une valeur correspondant à la fréquence d'utilisation du message, à partir d'un état de l'automate.

De plus, toutes les échanges observées durant la phase d'analyse et qui ne contiennent aucun message non annoté par une performative (voir 3.1.3) sont correctement représentés par les

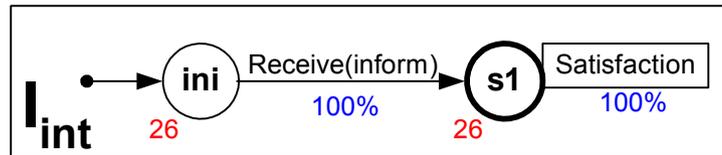
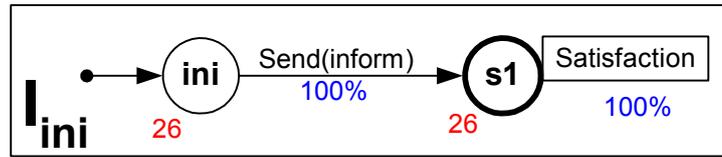


FIG. 4.4 – Automates d'envoi spontané d'information

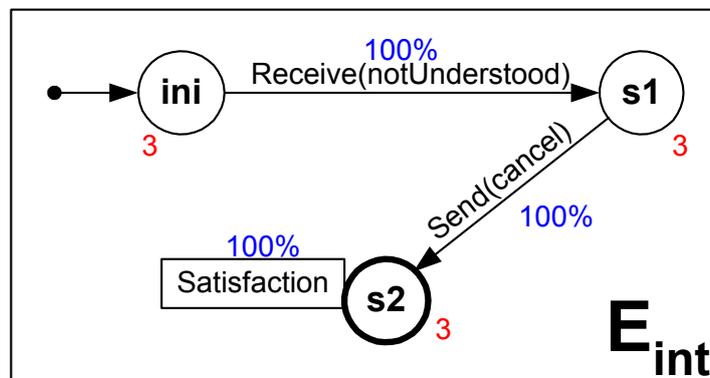
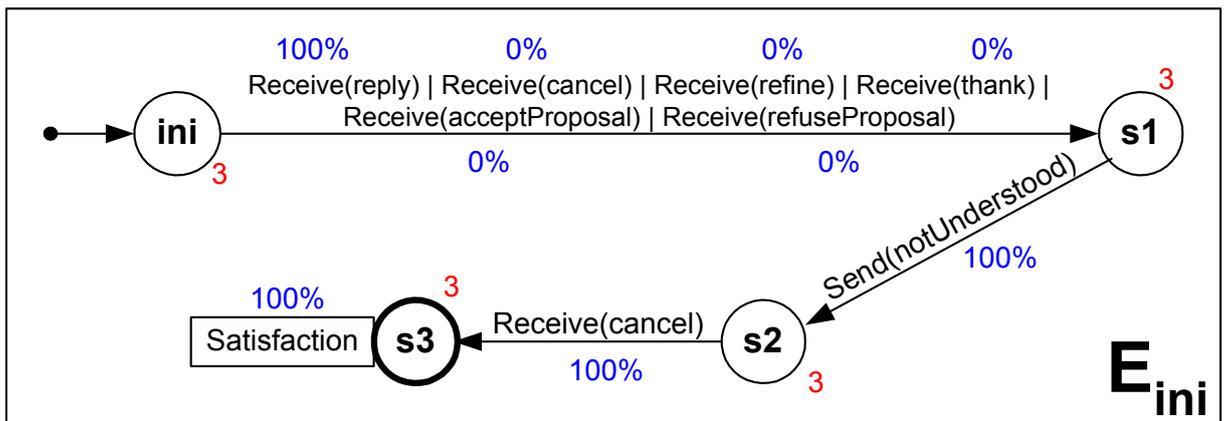


FIG. 4.5 – Automate de traitement des erreurs

automates temporisés.

Les automates temporisés sont plus qu'un simple outil de représentation des échanges observés dans les protocoles expérimentaux. Ils permettent de simuler la génération et l'interprétation de messages.

- La génération : les messages sont produits en suivant rigoureusement l'automate. Lorsqu'un choix se présente, la décision est prise en fonction de l'étape courante et de la personnalité du sujet.
- L'interprétation : les automates permettent de gérer les attendus en fonction des états dans lesquels les interlocuteurs se trouvent. Plusieurs échanges pouvant se dérouler simultanément entre deux mêmes interlocuteurs, ces attendus aident à connaître l'appartenance d'un message à un échange.

La valuation des états et des transitions des automates temporisés permet non seulement de valider la complétude de l'approche mais aussi d'améliorer le choix du franchissement des transitions. En effet, s'il existe plusieurs transitions possibles à partir d'un état, le choix dans les automates sera fait aléatoirement en respectant les fréquences d'utilisation par les sujets humains.

Dans une précédente version des automates temporisés, les dialogues incidents (un *refine* suivi d'un *query*, d'un *reply* ou d'un *cancel*, voir 4.2.3) n'étaient pas représentés à l'intérieur du même automate. La réception et l'envoi d'un *refine* déclenchaient l'ouverture d'un nouvel automate chez les deux interlocuteurs pour gérer le dialogue incident. La valeur de satisfaction de cet automate permettait, en réactivant l'automate gérant le dialogue régissant, d'actualiser son état courant. Nous avons choisi d'abandonner l'utilisation de la récursivité dans les automates temporisés car, tout en étant coûteuse à utiliser, cette solution ne permettait d'inclure en plus dans la modélisation que l'unique dialogue incident de niveau deux (voir 3.2.4). Cependant, nous pensons que la récursivité dans les automates temporisés pourrait être utile pour gérer d'autres types de dialogue que ceux présents dans les protocoles expérimentaux.

D'autre part, il serait sans doute possible d'utiliser des réseaux de Petri à la place des automates temporisés. Ces derniers ont été préférés car ils permettent de modéliser de manière très satisfaisante les protocoles expérimentaux tout en restant très lisibles.

### 4.3 Modélisation des énoncés

Dans la théorie des actes de langage, la notation fonctionnelle des énoncés en  $F(P)$  induit un lien très fort entre la force illocutoire et la proposition à laquelle elle s'applique. De la même manière dans les protocoles expérimentaux, le type de performative associée à un message et le contenu du message sont liés.

- Un *descriptif* est appliqué à une *croyance* : il permet au locuteur de décrire la manière dont il perçoit le monde, ce qui correspond à ses croyances. Celles-ci s'ajustent au monde en cas de satisfaction du descriptif.
- Un *directif* s'applique à un *désir* du locuteur : il est utilisé quand le locuteur cherche à faire faire quelque chose à l'allocutaire. Il a donc le désir de voir un certain état de fait réalisé et c'est ce désir qu'il transmet.
- Un *engageant* s'applique à un *désir* de l'interlocuteur : le locuteur utilise un engageant pour signifier à l'allocutaire qu'il s'engage à satisfaire un des désirs qu'il lui prête, désir

qu'il transmet pour une éventuelle confirmation.

Cette section présente le modèle formel des énoncés, basé sur un système de performatives comme la plupart des ACL. L'ensemble des performatives retenues est celui observé dans les protocoles expérimentaux et présenté en 3.1.1.

Ce modèle tient compte du lien entre la force illocutoire d'une performative et le contenu auquel elle est appliquée : chaque performative est appliquée, à une croyance pour un descriptif, un désir du locuteur pour un directif ou à un désir de l'interlocuteur pour un engageant. Pour permettre la description d'un énoncé, la syntaxe utilisée pour décrire le modèle formel doit être complétée. Les notions permettant de représenter les états mentaux des agents (croyances, désirs et intentions) et les messages sont ainsi ajoutées.

De plus, pour ne pas occulter l'articulation entre la gestion des énoncés, la dynamique conversationnelle et l'état interne des agents, la sémantique associée à chaque performative est donnée.

### 4.3.1 Description des états mentaux

Usuellement, les croyances, désirs et intentions sont considérées comme des modalités, auxquelles s'ajoutent souvent des prédicats portant sur ces états mentaux. Pour des raisons de simplicité d'écriture, les prédicats et les modalités ne sont pas différenciés. Ainsi, lorsqu'un prédicat portant sur un état mental est vrai, cela signifie que la modalité correspondante existe et réciproquement.

Soit  $DES$  l'ensemble des *désirs* ou *buts*.  $pD \in DES$  est noté :

$$pD(A \delta)$$

avec  $A \in AGT$  et  $\delta \in PRE$ . Cette représentation d'un désir comme appartenant à un agent est nécessaire pour pouvoir différencier les directifs et les engageants. En effet, les premiers s'appliquent à un désir du locuteur alors que les seconds s'appliquent à un désir de l'interlocuteur. Comme les trois problèmes co-existent, les agents peuvent avoir des désirs pour chaque problème. De plus, un désir porte sur un prédicat qui décrit le monde tel que l'agent souhaiterait le voir.

Soit  $BEL$  l'ensemble des *croyances*.  $pB \in BEL$  est notée :

$$pB(\alpha)$$

avec  $(\alpha \in DES) \mid (\alpha \in PRE \text{ et } \alpha \text{ ne contient aucune variable non instanciée})$ . Une croyance porte sur un prédicat (sur la description d'une partie du monde) ou sur un désir pour que l'agent puisse raisonner sur les désirs des autres agents.

Soit  $INT$  l'ensemble des *intentions*.  $pI \in INT$  est notée :

$$pI(\varphi)$$

avec  $\varphi \in ACT$ . Une intention s'applique à une action.

Soit l'ensemble des états mentaux,  $MEN = BEL \cup DES \cup INT$ .

L'état interne d'un agent est décrit par un ensemble de croyances, désirs et intentions. Les agents modifient leurs états mentaux à l'aide des deux actions internes élémentaires suivantes :

- $aAdd(M)$ , avec  $M \in MEN$  : ajoute  $M$  aux croyances, désirs ou intentions de l'agent.
- $aDel(M)$ , avec  $M \in MEN$  : supprime  $M$  des croyances, désirs ou intentions de l'agent.

Avec ces deux actions, les agents peuvent ajouter et supprimer de leurs états mentaux des croyances, désirs et intentions.

### 4.3.2 Modèle formel d'un énoncé

Le problème de la langue naturelle, qui est un axe de recherche à part entière, ne sera pas abordé ici. Dans notre modèle, un énoncé est une performative appliquée à un état mental dont le contenu est un prédicat. Il y a traduction des états mentaux du locuteur lors de l'émission d'un message, et interprétation de ce message par l'allocataire pour le comprendre (voir FIG. 4.6).

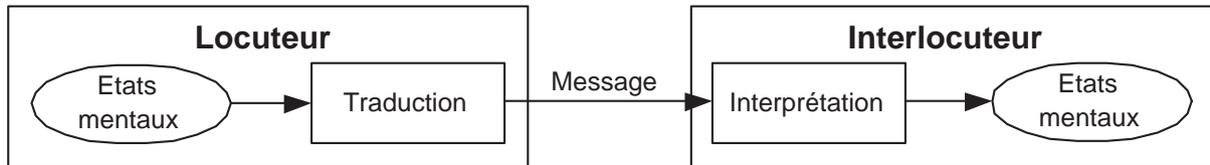


FIG. 4.6 – Communication entre agents

Un message échangé entre deux agents est décrit par le prédicat :

$$pMessage(A_S A_R P O)$$

avec  $A_S \in AGT$  l'agent émetteur,  $A_R \in AGT$  le destinataire,  $P \in PER$  la performative du message,  $O \in MEN$  l'objet auquel s'applique la performative.

Lors de la réception et de l'envoi de messages, les changements d'état dans les automates s'effectuent grâce à l'action interne abstraite  $aUpdate(M)$ , avec  $M \in MES$ . Cette action permet de regarder dans les automates ouverts s'il y en a un dont l'état courant permet l'envoi ou la réception du message en argument. Si c'est le cas, l'état actif de l'automate est modifié en conséquence et l'automate est fermé si nécessaire. Si aucun automate ne correspond, un nouvel automate est ouvert.

Par exemple, si l'agent aérien désire connaître les horaires des trains entre Paris et Angers, cela signifie qu'il possède dans ses états mentaux le désir :

$$pD(air pStage(Paris Angers ?? train * * *))$$

Il enverra alors un message de type *query* contenant ce désir à l'agent ferroviaire :

$$pMessage(air railway query pD(air pStage(Paris Angers ?? train * * *)))$$

Pour que cet acte soit réussi, il faut que l'agent ferroviaire l'interprète comme le désir initial contenu dans les états mentaux de l'agent aérien, et pour qu'il soit satisfait il faut que l'agent ferroviaire transmette à l'agent aérien les horaires du train Paris-Angers.

### 4.3.3 Sémantique générale des performatives

La sémantique des performatives nécessite l'utilisation de trois prédicats n'ayant pas été introduits jusqu'alors.

- $pNotUnderstood(M)$ , avec  $M \in MES$ , décrit un message non compris.
- $pSent(M)$ , avec  $M \in MES$ , devient vrai quand  $M$  est envoyé.
- $pMeans(S)$ , avec  $S \in STA$ , est vrai pour une étape dont le moyen de transport est celui dont l'agent a la charge.

La sémantique associée à chaque performative est présentée SEC. 4.3.4, 4.3.5 et 4.3.6. Pour chaque performative :

- La *syntaxe du message*, exprimée dans le langage logique donné précédemment, décrit la forme du message. Sa forme générale est :  $pMessage(A_S A_R P O)$  avec  $A_S \in AGT$  l'agent émetteur,  $A_R \in AGT$  le destinataire,  $P \in PER$  la performative du message,  $O \in MEN$  l'objet auquel s'applique la performative.  $O$  est une croyance quand la performative est un descriptif, un désir du locuteur quand la performative est un directif et un désir de l'interlocuteur quand la performative est un engageant.
- La *sémantique* associée à la performative est présentée sous la forme d'une règle générique de réduction précisant formellement les pré-conditions à l'envoi et à la réception de la performative ainsi que les actions à effectuer. La sémantique est profondément liée aux automates temporisés.
- Une courte *description* donne une explication informelle sur l'utilisation de la performative. Elle décrit en langage courant la situation à partir de laquelle la performative peut être émise ou reçue et les changements à effectuer sur l'état interne de l'agent qui reçoit ou envoie cette performative.
- Un ou plusieurs *exemple(s)* tiré(s) des protocoles expérimentaux sont traduits sous la forme de prédicat de  $MES$ .

La sémantique est présentée sous la forme d'une règle générique de réduction de l'un des deux types suivants :

$$\begin{array}{c} [PreCond] \frac{T_X(si_1, \dots, si_n) \xrightarrow{Send(per\ formative)} T_X(sf)}{aAdd(pB(pSent(M))); aUpdateTA(M)} \\ \\ \frac{T_X(si_1, \dots, si_n) \xrightarrow{Receive(per\ formative)} T_X(sf)}{a_1; \dots; a_n; aUpdateTA(M)} \end{array}$$

avec  $PreCond$  les préconditions,  $T \in \{Q, P, I, E\}$  et  $X \in \{ini, int\}$  indiquent le type de l'automate,  $si_1, \dots, si_n \in SAU$  les états avant le traitement du message,  $sf \in SAU$  l'état après traitement et  $a_1, \dots, a_n \in ACT$  les actions à réaliser lors du traitement.

Quelques caractéristiques se dégagent de cette sémantique :

- Il n'y a jamais de pré-condition à la réception d'un message de façon à ce que toute réception de message puisse être traitée. Si le message reçu ne s'intègre dans aucun des

échanges en cours, il activera l'ouverture d'un nouvel automate, éventuellement de traitement des erreurs.

- Quels que soient les effets à l'envoi et à la réception des messages, les agents doivent au moins réaliser l'action  $aUpdateTA(M)$ , avec  $M \in MES$  le message reçu ou envoyé, qui permet de mettre à jour les automates.
- Tout envoi de message génère l'action  $aAdd(pB(pSent(M)))$  pour que les agents gardent en mémoire les messages envoyés.
- Contrairement à la réception d'un message, aucune action autre que la mise à jour des automates et la mémorisation de l'envoi n'est effectuée à l'envoi d'un message. Un envoi de message est une simple action élémentaire externe alors qu'une réception de message nécessite la mise à jour des états mentaux de l'agent.

Les performatives sont regroupées selon leur type d'acte illocutoire (descriptifs, directifs et engageants) et classées alphabétiquement.

#### 4.3.4 Sémantique des descriptifs

Tous les descriptifs sont appliqués à des croyances. Leur forme générale est donc :

$$pMessage(A_S A_R performative pB)$$

avec  $A_S, A_R \in AGT$ ,  $performative \in PER$  et  $pB \in BEL$ .

Les quatre types de descriptifs sont les *inform*, les *notUnderstood*, les *reply* et les *thank*.

- **inform**

**Syntaxe :**  $pMessage(A_S A_R inform pB(P))$  avec  $A_S, A_R \in AGT$  et  $P \in PRO$ .

**Sémantique :**

$$\left[ \begin{array}{l} pB(P) \\ pD(A_R P) \end{array} \right] \frac{I_{ini}(ini) \xrightarrow{Send(inform)} I_{ini}(s1)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{I_{int}(ini) \xrightarrow{Receive(inform)} I_{int}(s1)}{aAdd(pB(P)); aUpdateTA(M)}$$

**Description :**  $A_S$  prévient  $A_R$  qu'il a terminé son problème s'il le croit et s'il a le désir d'en informer  $A_R$ . À la réception de l'*inform*,  $A_R$  ajoute à ses croyances le fait que  $A_S$  a résolu son problème.

**Exemple :**

Message	Syntaxe
De : Agent routier A : Agent ferroviaire Mission accomplie pour moi !	$pMessage(road\ railway\ inform\ pB(pProblem(road\ true)))$

- **notUnderstood**

**Syntaxe :**  $pMessage(A_S A_R notUnderstood pB(M'))$  avec  $A_S, A_R \in AGT$  et  $M' \in MES$ .

**Sémantique :**

$$[pB(pNotUnderstood(M'))] \frac{Q_{int}(s1) \xrightarrow{Send(notUnderstood)} Q_{int}(s5)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$[pB(pNotUnderstood(M'))] \frac{P_{int}(s1) \xrightarrow{Send(notUnderstood)} P_{int}(s3)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \begin{array}{l} pD(A_S pCanceled(M')) \\ pB(!pSent(M')) \end{array} \right] \frac{E_{ini}(s1) \xrightarrow{Send(notUnderstood)} E_{ini}(s2)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{ini}(s1) \xrightarrow{Receive(notUnderstood)} Q_{ini}(s8)}{aAdd(pB(pNotUnderstood(M'))); aUpdateTA(M)}$$

$$\frac{P_{ini}(s1) \xrightarrow{Receive(notUnderstood)} P_{ini}(s3)}{aAdd(pB(pNotUnderstood(M'))); aUpdateTA(M)}$$

$$\frac{E_{int}(ini) \xrightarrow{Receive(notUnderstood)} E_{int}(s1)}{aAdd(pB(pD(A_S pCanceled(M')))); aUpdateTA(M)}$$

**Description :** Au cours d'une demande d'information ou d'une proposition d'information,  $A_S$  peut envoyer un *notUnderstood* quand il ne comprend pas un message. Son interlocuteur  $A_R$ , à la réception de ce *notUnderstood*, ajoute à ses croyances la connaissance que le message n'a pas été compris. Par contre, au cours d'un échange de type traitement des erreurs,  $A_S$  ne peut envoyer de *notUnderstood* que s'il a le désir d'annuler un message qu'il n'a pas lui-même envoyé précédemment.  $A_R$  ajoute alors à ses croyances la connaissance du fait qu' $A_S$  désire annuler le message  $M'$ .

**Exemple :**

Message	Syntaxe
De : Agent aérien A : Agent ferroviaire » Agent ferroviaire a écrit : » ===== HORAIRE ===== » Réservation effectuée : » SaintMartin - Paris (train) » Date : Demain - 3 personnes » 18h10/18h35 - 20 F Je n'ai pas demandé cette réservation !	$pMessage(air\ railway\ notUnderstood\ pB(pMessage(railway\ air\ reply\ pStage(SaintMartin\ Paris\ 18:10\ 18:35\ train\ 3\ 20\ true))))$

- **reply**

**Syntaxe :**  $pMessage(A_S A_R reply pB(S)|pB(!S))$  with  $A_S, A_R \in AGT$  and  $S \in STA$ .

**Semantics :**

$$\left[ \frac{pD(A_R S)}{(pB(S)|pB(!S))} \right] \frac{Q_{ini}(s1, s2, s4) \xrightarrow{Send(reply)} Q_{ini}(s2)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \frac{pD(A_R S)}{(pB(S)|pB(!S))} \right] \frac{P_{ini}(s4) \xrightarrow{Send(reply)} P_{ini}(s6)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{ini}(s1, s2, s7) \xrightarrow{Receive(reply)} Q_{ini}(s2)}{aAdd(pB(S))|aAdd(pB(!S)); aUpdateTA(M)}$$

$$\frac{P_{ini}(s4) \xrightarrow{Receive(reply)} P_{ini}(s6)}{aAdd(pB(S))|aAdd(pB(!S)); aUpdateTA(M)}$$

$$\frac{E_{ini}(s1) \xrightarrow{Receive(reply)} E_{ini}(s1)}{aAdd(pB(pD(A_R pCanceled(M))))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *reply* s'il désire  $S$  pour  $A_R$  et s'il a une connaissance appropriée à fournir sur l'étape  $S$ . À la réception d'un *reply*,  $A_R$  ajoute à ses croyances l'étape contenu dans le message.

**Exemples :**

Message	Syntaxe
De : Agent ferroviaire A : Agent aérien Voici ton train : ===== HORAIRES ===== Orléans - Paris (train) Date : Demain - 1 personne 08h25/09h30 - 80 F	$pMessage(railway\ air\ reply\ pB(pStage(Orleans\ Paris\ 08\ :25\ 09\ :30\ train\ 1\ 80\ false)))$
De : Agent routier A : Agent aérien Voici ta réservation : ===== HORAIRES ===== Réservation effectuée : Montpellier(aéro) - Maguelone (car) Date : Demain - 4 personnes 19h30/20h00 - 60 F	$pMessage(road\ air\ reply\ pB(pStage(Montpellier(aero)\ Maguelone\ 19\ :30\ 20\ :00\ car\ 4\ 60\ true)))$

• **thank**

**Syntaxe :**  $pMessage(A_S A_R thank pB(S)|pB(!S))$  avec  $A_S, A_R \in AGT$  et  $S \in STA$ .

**Sémantique :**

$$\left[ (pB(S)|pB(!S)) \right] \frac{Q_{ini}(s2) \xrightarrow{Send(thank)} Q_{ini}(s3)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ (pB(S)|pB(!S)) \right] \frac{P_{ini}(s6) \xrightarrow{Send(thank)} P_{ini}(s7)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{int}(s2) \xrightarrow{Receive(thank)} Q_{int}(s3)}{aUpdateTA(M)}$$

$$\frac{P_{ini}(s6) \xrightarrow{Receive(thank)} P_{ini}(s7)}{aUpdateTA(M)}$$

$$\frac{E_{ini}(ini) \xrightarrow{Receive(thank)} E_{ini}(s1)}{aAdd(pB(pD(A_R pCanceled(M))))}; aUpdateTA(M)}$$

**Description :** Pour pouvoir envoyer un *thank*,  $A_S$  doit avoir dans sa mémoire une étape  $S$  motivant le remerciement. Il n'y a aucune action particulière à effectuer à l'envoi comme à la réception d'un *thank* : l'automate concerné sera fermé dans  $aUpdateTA(M)$ .

**Exemples :**

Message	Syntaxe
De : Agent ferroviaire A : Agent aérien » Il n'y a pas de ligne entre paris et lille. Merci.	$pMessage(railway\ air\ thank\ pB(!pStage(Paris\ Lille\ \*\*train\ \*\*\*)))$
De : Agent ferroviaire A : Agent aérien » C'est réservé. merci	$pMessage(railway\ air\ thank\ pB(pStage(Orleans\ Paris\ 08\ :25\ 09\ :30\ train\ 4\ 80\ true)))$

### 4.3.5 Sémantique des directifs

Tous les directifs sont appliqués à des désirs du locuteur. Leur forme générale est donc :

$$pMessage(A_S A_R performative pD(A_S \phi))$$

avec  $A_S, A_R \in AGT$ ,  $performative \in PER$  et  $\phi \in PRE$ .

Les cinq types de directifs sont les *acceptProposal*, les *cancel*, les *query*, les *refine* et les *notUnderstood*.

- **acceptProposal**

**Syntaxe :**  $pMessage(A_S A_R acceptProposal pD(A_S S))$  avec  $A_S, A_R \in AGT$  et  $S \in STA$ .

**Sémantique :**

$$\left[ \begin{array}{l} pD(A_S S) \\ pB(!pMeans(S)) \\ !pB(S) \\ !pB(!S) \end{array} \right] \frac{P_{int}(s1) \xrightarrow{Send(acceptProposal)} P_{int}(s4)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{P_{ini}(s1) \xrightarrow{Receive(acceptProposal)} P_{ini}(s4)}{aAdd(pB(pD(A_S S))); aUpdateTA(M)}$$

$$\frac{E_{ini}(ini) \xrightarrow{Receive(acceptProposal)} E_{ini}(s1)}{aAdd(pB(pD(A_R pCanceled(M)))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *acceptProposal* s'il a le désir de connaître l'étape  $S$ , s'il n'a aucune connaissance concernant  $S$  dans sa mémoire et si  $S$  ne se fait pas via son propre moyen de transport. À la réception d'un *acceptProposal*,  $A_R$  possède la certitude que  $A_S$  désire bien avoir des informations à propos de l'étape  $S$  et met à jour ses croyances en conséquence.

### Exemples :

Message	Syntaxe
De : Agent routier A : Agent ferroviaire »» Salut, pourrais-je avoir les horaires et les »» prix du train SaintMartin à LeChâtelard »» pour 3 personnes ? » Chambéry-Paris, si tu veux ! Pourquoi pas.	$pMessage(road\ railway\ acceptProposal\ pD(road\ pStage(Chambery\ Paris\ ??\ train\ 3\ ?\ *)))$
De : Agent routier A : Agent ferroviaire » Veux-tu que je te réserve le Chambéry- » Paris ? Oui, réserve-moi cet horaire. ===== HORAIRE ===== Chambéry - Paris (train) Date : Demain - 3 personnes 12h52/15h50 - 370 F	$pMessage(road\ railway\ acceptProposal\ pD(road\ pStage(Chambery\ Paris\ 12\ :52\ 15\ :50\ train\ 3\ 370\ true)))$

- **cancel**

**Syntaxe :**  $pMessage(A_S A_R cancel pD(A_S !M'))$  avec  $A_S, A_R \in AGT$  et  $M' \in MES$ .

### Sémantique :

$$\left[ \begin{array}{l} pD(A_S pCanceled(M')) \\ pB(pSent(M')) \end{array} \right] \frac{Q_{ini}(s1, s2, s7, s8) \xrightarrow{Send(cancel)} Q_{ini}(s6)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \begin{array}{l} pD(A_S pCanceled(M')) \\ pB(pSent(M')) \end{array} \right] \frac{Q_{int}(s2) \xrightarrow{Send(cancel)} Q_{int}(s1)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \begin{array}{l} pD(A_R pCanceled(M')) \\ pB(pSent(M')) \end{array} \right] \frac{E_{int}(s1) \xrightarrow{Send(cancel)} E_{int}(s2)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{ini}(s2) \xrightarrow{Receive(cancel)} Q_{ini}(s1)}{aAdd(pB(pCanceled(M'))); aUpdateTA(M)}$$

$$\frac{Q_{int}(s1, s2, s4, s5) \xrightarrow{Receive(cancel)} Q_{int}(s6)}{aAdd(pB(pCanceled(M'); aUpdateTA(M))}$$

$$\frac{E_{int}(s2) \xrightarrow{Receive(cancel)} E_{int}(s3)}{aAdd(pB(pCanceled(M'); aUpdateTA(M))}$$

$$\frac{E_{ini}(ini) \xrightarrow{Receive(cancel)} E_{ini}(s1)}{aAdd(pB(pD(A_R pCanceled(M))))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *cancel* dès lors qu'il a bien le désir d'annuler un message  $M'$  qu'il a lui même envoyé précédemment. À la réception d'un *cancel*,  $A_R$  ajoute à ses croyances le fait que le message  $M'$  a été annulé.

**Exemples :**

Message	Syntaxe
De : Agent ferroviaire A : Agent routier Non, finalement, je ne veux pas de car à 07h15	$pMessage(railway\ road\ cancel!pMessage(railway\ road\ query\ pD(railway\ pStage(Beaupreau\ Angers\ 07:15\ ?\ car\ ?\ ?\ false)))$
De : Agent routier A : Agent ferroviaire »» Voici le train Paris-Annecy. » es-tu sûr que ce message soit pour moi ? Voilà, j'ai supprimé le message.	$pMessage(road\ railway\ cancel!pMessage(road\ railway\ reply\ pStage(Paris\ Annecy\ 18:16\ 22:04\ train\ 3\ 400\ false)))$

• **query**

**Syntaxe :**  $pMessage(A_S A_R query pD(A_S S))$  avec  $A_S, A_R \in AGT$  et  $S \in STA$ .

**Semantics :**

$$\left[ \begin{array}{l} pD(A_S S) \\ pB(!pMeans(S)) \\ !pB(S) \\ !pB(!S) \end{array} \right] \frac{Q_{ini}(ini, s2, s4, s7, s8) \xrightarrow{Send(query)} Q_{ini}(s1)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \begin{array}{l} pD(A_S S) \\ pB(!pMeans(S)) \\ !pB(S) \\ !pB(!S) \end{array} \right] \frac{P_{int}(s5) \xrightarrow{Send(query)} P_{int}(s4)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{int}(ini, s1, s2, s4, s5) \xrightarrow{Receive(query)} Q_{int}(s1)}{aAdd(pB(pD(A_S S))); aUpdateTA(M)}$$

$$\frac{P_{ini}(s5) \xrightarrow{Receive(query)} P_{ini}(s4)}{aAdd(pB(pD(A_S S))); aUpdateTA(M)}$$

**Description :**  $A_S$  envoie un *query* s'il désire des informations à propos d'une étape  $S$ , si  $S$  utilise un moyen de transport différent de la spécialité d' $A_S$  et si  $A_S$  n'a aucune

connaissance à propos de  $S$ . À la réception du message,  $A_R$  ajoute à ses connaissances la croyance qu' $A_S$  désire une information concernant l'étape  $S$ .

Une message de type query contient au minimum un ?, chaque ? indiquant les champs à remplir obligatoirement. Dans le cas d'un  $?pStage$ , la demande porte sur l'existence de la ligne.

### Exemples :

Message	Syntaxe
De : Agent aérien A : Agent ferroviaire Y a-t-il un train qui va de Orléans à Paris, pour 10h30 au plus tard ?	$pMessage(air\ railway\ query\ pD(air\ ?pStage(Orleans\ Paris\ * \leq(10:30)\ train\ * * *)))$
De : Agent aérien A : Agent routier Pouvez-vous réserver cet horaire pour 4 ? ===== HORAIRES ===== ChâteauNeuf - Orléans(gare) (car) Date : Demain - 1 personne 06h10/06h45 - 24 F	$pMessage(air\ road\ query\ pD(air\ ?pStage(Chateau\ Neuf\ Orleans(gare)\ 06:10\ 06:45\ car\ 1\ 24\ true)))$
De : Agent routier A : Agent aérien Quels sont les horaires des avions Paris- Chambéry ?	$pMessage(road\ air\ query\ pD(road\ pStage(Paris\ Chambéry\ ?\ ?\ avion\ * * false)))$

- refine

**Syntaxe :**  $pMessage(A_S\ A_R\ refine\ pD(A_S\ pD(A_R\ S)))$  avec  $A_S, A_R \in AGT$  et  $S \in STA$ .

### Sémantique :

$$\left[ \begin{array}{l} pD(A_S\ pRefined(S)) \\ pB(!pMeans(S)) \end{array} \right] \frac{Q_{int}(s1) \xrightarrow{Send(refine)} Q_{int}(s4)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\left[ \begin{array}{l} pD(A_S\ pRefined(S)) \\ pB(!pMeans(S)) \end{array} \right] \frac{P_{ini}(s4) \xrightarrow{Send(refine)} P_{ini}(s5)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{Q_{ini}(s1) \xrightarrow{Receive(refine)} Q_{ini}(s7)}{aAdd(pB(pD(A_S\ pRefined(S)))); aUpdateTA(M)}$$

$$\frac{P_{int}(s4) \xrightarrow{Receive(refine)} P_{int}(s5)}{aAdd(pB(pD(A_S\ pRefined(S)))); aUpdateTA(M)}$$

$$\frac{E_{ini}(ini) \xrightarrow{Receive(refine)} E_{ini}(s1)}{aAdd(pB(pD(A_R\ pCanceled(M)))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *refine* lorsqu'il désire voir l'étape  $S$  raffinée et que celle-ci ne correspond pas à son moyen de transport. À la réception d'un *refine*,  $A_R$  ajoute à ses croyances le fait qu' $A_S$  désire voir l'étape  $S$  raffinée.

### Exemples :

Message	Syntaxe
De : Agent routier A : Agent ferroviaire » Y a-t-il des taxis ou des cars de Lille à » Baisieux ? Lille gare ou aéro ?	$pMessage(road\ railway\ refine\ pD(road\ pD(railway\ pStage(?\ (Lille(aero)\ Lille(gare))\ Baisieux\ **\ OR(taxi\ car)\ **\ *)))$
De : Agent routier A : Agent arien » Y a-t-il des cars de Montpellier à » Maguelone ? Il y a des cars mais quel horaire ?	$pMessage(road\ air\ refine\ pD(road\ pD(air\ pStage(Montpellier\ Maguelone\ ??\ car\ **\ *)))$
De : Agent routier A : Agent ferroviaire »» voici ton horaire » je voudrais une réservation svp quelle réservation svp ?	$pMessage(road\ railway\ refine\ pD(railway\ pStage(??? ?\ ?\ ?\ true)))$

- **refuseProposal**

**Syntaxe :**  $pMessage(A_S\ A_R\ refuseProposal\ !pD(A_S\ S))$  avec  $A_S, A_R \in AGT$  et  $S \in STA$ .

### Sémantique :

$$\left[ \begin{array}{l} !pD(A_S\ S) \\ !pB(pD(A_S\ S)) \end{array} \right] \frac{P_{int}(s1) \xrightarrow{Send(refuseProposal)} P_{int}(s2)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{P_{ini}(s1) \xrightarrow{Receive(refuseProposal)} P_{ini}(s2)}{aAdd(pB(!pD(A_S\ S))); aUpdateTA(M)}$$

$$\frac{E_{ini}(s1) \xrightarrow{Receive(refuseProposal)} E_{ini}(s1)}{aAdd(pB(pD(A_R\ pCanceled(M)))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *refuseProposal* après avoir traité le désir proposé par  $A_R$  et ne pas l'avoir intégré à sa pile de désirs. Quand  $A_R$  reçoit le *refuseProposal*, il ajoute à ses croyances la négation du désir qui avait provoqué l'envoi du *propose*.

### Exemples :

Message	Syntaxe
De : Agent aérien A : Agent ferroviaire » Veux-tu d'autres horaires ? Non, les horaires que j'ai choisi me conviennent.	$pMessage(air\ railway\ refuseProposal\ !pD(air\ pStage(Orleans\ Paris\ **\ train\ 4\ **)))$
De : Agent aérien A : Agent ferroviaire » Veux-tu que je réserve Paris-Montpellier ? Non, ce sont les horaires qui ne collent pas.	$pMessage(air\ railway\ refuseProposal\ !pD(air\ pStage(Paris\ Montpellier\ 12\ :06\ 16\ :21\ train\ 1\ 590\ true)))$

### 4.3.6 Sémantique des engageants

Tous les engageants sont appliqués à des désirs du destinataire. Leur forme générale est :

$$pMessage(A_S A_R \text{ performative } pD(A_R \phi))$$

avec  $A_S, A_R \in AGT$ ,  $\text{performative} \in PER$  et  $\phi \in PRE$ .

Le seul engageant observé dans les protocoles expérimentaux est un *propose*.

- **propose**

**Syntaxe :**  $pMessage(A_S A_R \text{ propose } pD(A_R S))$  avec  $A_S, A_R \in AGT$ .

**Sémantique :**

$$\left[ \begin{array}{l} pD(A_S pB(pD(A_R S))) \\ pB(pMeans(S)) \end{array} \right] \frac{P_{ini}(ini) \xrightarrow{Send(propose)} P_{ini}(s1)}{aAdd(pB(pSent(M))); aUpdateTA(M)}$$

$$\frac{P_{int}(ini) \xrightarrow{Receive(propose)} P_{int}(s1)}{aAdd(pB(pD(A_R S))); aUpdateTA(M)}$$

**Description :**  $A_S$  peut envoyer un *propose* portant sur une étape  $S$  dès lors que  $S$  correspond à son moyen de transport et qu'il veut avoir la confirmation que  $A_R$  désire obtenir des informations sur l'étape  $S$ . À la réception de ce *propose*,  $A_R$  ajoute à ses croyances le désir concernant l'étape  $S$ .  $A_R$  ajoutera ce désir à sa pile de désirs si l'étape est jugée utile pour résoudre son problème.

**Exemples :**

Message	Syntaxe
De : Agent ferroviaire A : Agent aérien » ChâteauNeuf - Paris ? Orléans - Paris, si tu veux...	$pMessage(\text{railway air propose } pD(\text{air } pStage(\text{Orleans Paris ? ? train * * *))))$
De : Agent aérien A : Agent ferroviaire Dois-je réserver ? ===== HORAIRES ===== Nantes - Lille (PigeonVol) Date : Demain - 1 personne 07h45/09h03 - 802 F	$pMessage(\text{air railway propose } pD(\text{railway } pStage(\text{Nantes Lille 07 :45 09 :03 avion(PigeonVol) 1 802 true))))$

### 4.3.7 Discussion

Le modèle formel de l'énoncé utilisé ici est très proche de ceux employés dans la plupart des ACL classiques, tels KQML ou FIPA-ACL. Il se présente sous la forme d'une performative appliquée à un état mental dont le type est lié au type de la performative.

La problématique du traitement automatique de la langue naturelle n'est pas abordée car elle correspond à un axe de recherche à part entière. L'approche utilisée ici, qui fait appel à la

traduction des états mentaux, en *pMessage* à l'émission d'un message, et à l'interprétation d'un message en états mentaux à sa réception, n'est pas incompatible avec cette problématique. Il est possible d'intégrer un module se chargeant de faire la traduction et l'interprétation en langue naturelle.

Le modèle de l'énoncé et le modèle de la dynamique conversationnelle sont liés par la sémantique des performatives. Elle décrit formellement les pré-conditions qui doivent être vérifiées avant un envoi de message, en fonction des échanges en cours (représentés par les automates temporisés actifs). Elle définit aussi les actions internes à effectuer par l'émetteur et le destinataire d'un message lorsque celui-ci est traité.

Cependant, bien que ce modèle de l'énoncé couvre de façon très satisfaisante les messages contenus dans les protocoles expérimentaux, il ne permet pas de représenter toutes les nuances des conversations humaines. Cela est dû à l'expérimentation qui permettait surtout d'observer des dialogues liés de recherche d'information. Il serait sans doute intéressant, dans le cadre d'un corpus différent, de s'appuyer sur les remarques de Bouzouba et Moulin lorsqu'ils ont conçu KQML+ [Bouzouba et Moulin, 2001] (voir SEC. 2.4.3), ou bien sur les travaux de Weigand et Dignum [Dignum et Weigand, 1995] et (voir SEC. 2.2.5).

## 4.4 Conclusion

La modélisation des interactions humaines présentée ici fait suite à l'analyse de protocoles expérimentaux issus d'une expérimentation psychologique. Les modèles développés doivent respecter cette phase d'analyse pour permettre de reproduire les comportements observés. Ces modèles ont été conçus après de nombreux allers-retours entre les phases d'analyse et de modélisation. Nous avons pu définir formellement :

- un modèle du Problème de l'Agence de Voyage qui explicite les contraintes à satisfaire pour qu'un voyage soit solution du problème,
- un modèle des énoncés humains, sous la forme de performatives appliquées à des états mentaux,
- un modèle de la dynamique des conversations humaines par automates temporisés,
- une sémantique des performatives qui relie le modèle de l'énoncé et le modèle de la dynamique conversationnelle.

Ces modèles sont satisfaisants car ils couvrent la quasi-totalité des énoncés contenus dans les protocoles expérimentaux et leurs enchaînements. De plus, comme cela sera montré en 6, ils permettent de générer les comportements observés dans les protocoles expérimentaux.

Cependant, ils peuvent difficilement être présentés séparément de l'architecture d'agent à laquelle ils sont intégrés (voir 5) et qui a permis leur validation. Cette intégration a été possible en grande partie grâce aux grandeurs communes qu'ils partagent : les croyances (*B*), désirs (*D*) et intentions (*I*).



## **Troisième partie**

### **Mise en oeuvre et validation**



# Chapitre 5

## Mise en oeuvre des modèles cognitifs de planification et d'interaction : l'architecture BDI<sup>GGY</sup>

Pour développer un système capable de simuler les comportements observés, un SMA composé de trois agents cognitifs a été conçu. L'architecture des agents, appelée BDI<sup>GGY</sup>, intègre les modèles d'interaction et le modèle de planification humaine implanté par le système IGGY2 (version d'IGGY adaptée au Problème de l'Agence de Voyage), de façon la plus homogène possible. Les modèles d'interaction comprennent

- les performatives associées aux états mentaux comme modèle de l'énoncé,
- les automates temporisés pour représenter la dynamique conversationnelle,
- la sémantique des performatives reliant les deux niveaux de l'interaction.

Les croyances, désirs et intentions de type BDI permettent de faire le lien entre les interactions et la planification. En effet, le corps des messages et la sémantique des performatives sont représentés à l'aide de B et D et l'architecture BDI est orientée planification. C'est donc tout naturellement qu'une architecture BDI a été choisie comme base pour les agents de la simulation.

### 5.1 Description de BDI<sup>GGY</sup>

L'architecture d'agent, appelée BDI<sup>GGY</sup>, est utilisée pour résoudre le problème de l'agence de voyage et générer des protocoles "similaires" aux protocoles expérimentaux.

BDI<sup>GGY</sup> se structure autour d'une architecture BDI (voir 2.1.3) et du système IGGY2 (voir IGGY, en 1.2).

BDI<sup>GGY</sup> comprend différents modules (voir FIG. 5.1).

- Un *module de perception* qui observe l'environnement et génère des croyances.
- Un *module de planification humaine*, IGGY2, qui construit des plans abstraits de façon opportuniste ; IGGY2 génère un épisode, c'est-à-dire une phase, un état d'esprit, des stratégies et des tactiques, servant de squelette de plan ;

- Un *interpréteur de plans* qui se décompose en :
  - un *générateur de désirs* interprétant un épisode (plan abstrait) en désirs (morceaux de plan) et encapsulant une représentation des plans courants,
  - un *générateur d'intentions* instanciant un désir en intentions (plans instanciés),
  - un *module d'exécution* effectuant les actions nécessaires à une intention.
- Un *module de communication* qui permet à l'agent d'interagir. Il est appelé par le module de perception pour interpréter les messages reçus et par le module d'exécution pour générer les messages à envoyer. Il contient les modèles de l'énoncé et du dialogue (voir 4.2 et 4.3).

L'architecture BDIGGY est implantée en Java et en Common Lisp. IGGY étant programmé en Common Lisp, a été étendu dans le même langage à la résolution coopérative de problèmes en IGGY2. L'interface utilisée pour l'expérimentation psychologique, par contre, est implantée en Java. Le Common Lisp ne permettant pas de réaliser aisément de la programmation multi-tâches, nécessaire à la conception de SMA, les autres modules de l'architecture BDIGGY sont mis en oeuvre en Java. L'interface entre Java et Common Lisp s'effectue par le biais d'une bibliothèque Java permettant d'interpréter du code Lisp. Cette bibliothèque, appelée *jatha*<sup>1</sup>, a été enrichie pour supporter toutes les fonctions Common Lisp indispensables au fonctionnement d'IGGY2.

En résumé, seul le module IGGY2 est implémenté en Lisp, tous les autres modules de BDIGGY et l'interface sont développés en Java.

### 5.1.1 Module de perception

Le module de perception observe l'interface à chaque étape de la résolution du problème. Il permet à l'agent de mettre à jour ses connaissances sur :

- l'environnement,
- l'état de la résolution de son problème,
- les échanges en cours avec les autres agents.

Toutes les connaissances de l'agent sont stockées dans la *mémoire*, sous forme de croyances appliquées à un prédicat. Par exemple, si l'agent croit qu'il existe une ligne de train reliant Paris et Lille, sa mémoire contiendra la croyance suivante :

*pB(pStage(Paris Lille \* \* train \* \* \*))*

La *mémoire* de l'agent se structure autour de quatre entités distinctes (voir FIG. 5.2).

- La *représentation de l'interface* sert de modèle du monde extérieur. Elle est mise à jour à chaque appel du module de perception. Elle contient la description des panneaux de l'interface.
  - Le panneau "Horaires/Prix" est représenté par une liste de trajets (*pStage*) correspondant aux différents horaires (ou éventuellement par la négation d'un trajet quand il n'existe aucune ligne de transport reliant les deux villes sélectionnées).

<sup>1</sup>Site officiel : <http://jatha.sourceforge.net>

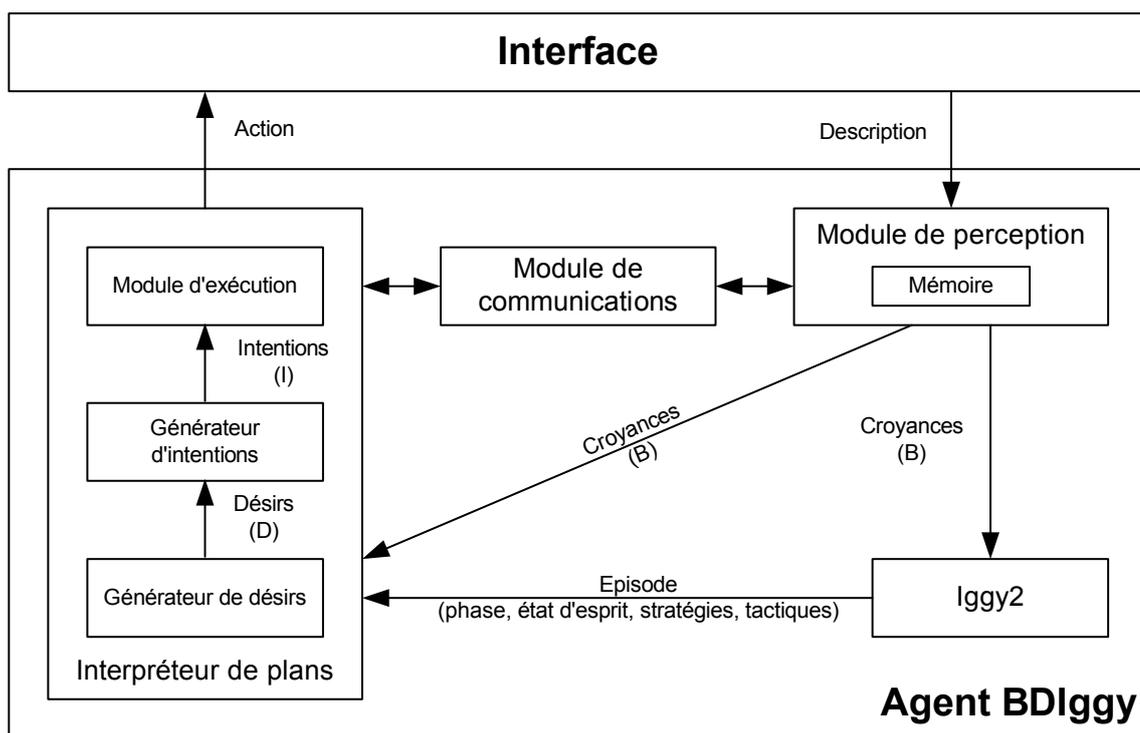


FIG. 5.1 – Flux de l'architecture BDIGGY

Mémoire à court terme	<b>Représentation de l'interface</b>	Mémoire dynamique
	Panneau "Horaires/Prix" Panneau "Zone de travail" Panneau "Proposition" Panneau "Messagerie"	
	<b>Croyances non traitées</b>	
Mémoire à long terme	Croyances sur des désirs à évaluer Croyances sur l'état des échanges en cours	Mémoire dynamique
	<b>Croyances persistantes</b>	
	Informations obtenues par la base de données Informations obtenues auprès des autres agents	Mémoire statique
<b>Représentation du problème</b>		
	Problème à résoudre Représentation de la carte (villes, distances et transports) Connaissances implicites	Mémoire statique

FIG. 5.2 – Structure de la mémoire

- Le panneau "Zone de travail" est décrit par une liste de trajets et un trajet détaillé correspondant à l'étape sélectionnée.
- Le panneau "Proposition" correspond dans la mémoire à un voyage ( $pTravel$ ).
- Le panneau "Messagerie" est dépeint par une liste de nouveaux messages et par un message ( $pMessage$ ) précisant le courrier électronique sélectionné.
- Les *croyances non traitées* caractérisent les connaissances brutes que l'agent ne sait pas encore s'il va les prendre en compte dans son raisonnement. Une croyance non traitée porte sur le désir d'un autre agent ou la description d'un échange.
  - Le *désir d'un autre agent* est reçu dans un message. Ce désir sera ajouté, ou non, aux désirs de l'agent, en fonction de sa disponibilité et de sa volonté de coopérer (selon sa personnalité). Par exemple, si l'agent aérien reçoit une demande d'horaire pour un avion Paris-Montpellier de la part de l'agent routier, il ajoutera à ses croyances non traitées la croyance :  $pB(pD(road\ pStage(Paris\ Montpellier\ ??\ 1\ plane\ *\ *false)))$ .
  - La *description d'un échange* (satisfaction, insatisfaction, demande d'annulation d'un message, etc.) modifie les désirs et les intentions de l'agent. Par exemple, supposons que l'agent aérien ait ouvert un automate de demande d'information ( $Q_{ini}$ ) pour obtenir des horaires de car entre la gare et l'aéroport de Paris. S'il n'obtient aucune réponse de la part de l'agent routier, malgré plusieurs redemandes, il clôture l'automate correspondant et ajoute à ses croyances non traitées la croyance :  $pB(pUnsatisfied(pD(air\ pStage(Paris(gare)\ Paris(aero)\ ??\ coach\ *\ *false))))$ .

La liste des croyances non traitées est modifiée au fur et à mesure de la résolution du problème. De nouvelles croyances non traitées sont ajoutées à la lecture de certains messages ou à la fermeture d'un échange (même non explicite). La liste diminue à chaque fois que l'interpréteur de plans analyse une des croyances non traitées et modifie éventuellement les désirs et intentions de l'agent.

- Les *croyances persistantes* regroupent les connaissances acquises par l'agent au cours de la résolution. Elles synthétisent les informations ayant été utiles à l'agent au moins une fois. Elles portent toujours sur un trajet, sur la négation d'un trajet ou sur l'état de la résolution du problème d'un autre agent. Leur liste augmente toujours au cours du temps, aucune croyance persistante n'étant supprimée durant la résolution du problème.
- Une *représentation du problème* spécifie les informations à la disposition de l'agent dès le début de la résolution du problème. Elle contient la définition du problème à résoudre, la description des cartes (liste des villes, distances entre les villes, lignes existantes, etc.) et quelques connaissances implicites (l'avion et le train doivent être utilisés sur des grands trajets (supérieurs à 30kms), le taxi coûte plus cher que le car, etc.). Cette partie de la mémoire est statique ; c'est une base de données répondant à une requête par la croyance correspondante.

Le module de perception met à jour la mémoire en analysant l'interface à chaque appel.

De plus, à la réception d'un message, le module de perception délègue au module de communication l'interprétation du message. Elle sera faite en fonction des automates ouverts et de la sémantique associée à la performative du message.

### 5.1.2 Système IGGY2

Comme son prédécesseur IGGY, IGGY2 s'articule autour d'une architecture de tableau noir à contrôle hiérarchique que des spécialistes du domaine consultent et remplissent. À chaque

étape de la résolution, IGGY2 retourne un épisode (phase, état d'esprit, stratégies et tactiques) utilisé par l'interpréteur de plans de BDI<sub>GGY</sub>.

Le système IGGY2 est une version d'IGGY, adaptée au Problème de l'Agence de Voyage. Leur fonctionnement général est strictement identique mais IGGY2 implante le modèle de résolution coopérative de problèmes présenté en 1.2.2. Seule la partie dépendante du domaine a été changée. Les différents spécialistes ont donc été reprogrammés. De plus, alors qu'IGGY génère un protocole complet à partir d'un problème à résoudre et de la personnalité du sujet à simuler, IGGY2 s'arrête à chaque étape de la résolution du problème en retournant un épisode.

IGGY2, initialisé avec une personnalité, prend en entrée une description du monde (interface, échanges en cours et connaissances de l'agent). Il génère alors des *observations* sur cette description du monde. Les observations sont une focalisation des connaissances importantes pour l'agent, à un moment de la résolution. Si nécessaire, IGGY2 construit un nouvel épisode, en fonction des observations.

L'enchaînement des phases s'effectue selon les observations et la phase précédente (voir FIG. 5.3).

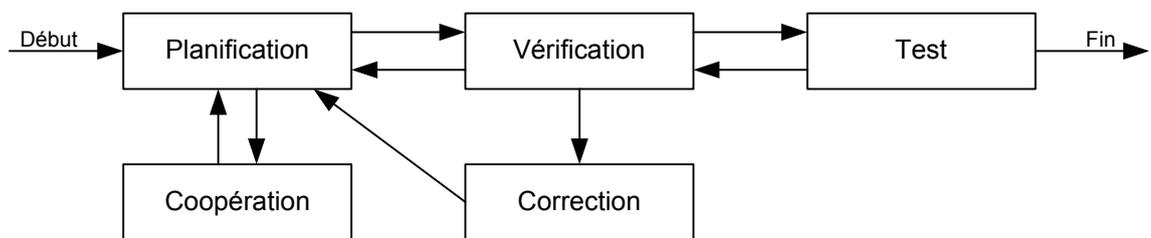


FIG. 5.3 – Enchaînements possibles des phases

Une fois la phase fixée, l'état d'esprit (les critères sur lesquels l'agent se concentre) est construit. Les critères présents dans l'état d'esprit ne peuvent pas être supprimés, mais de nouveaux critères sont ajoutés au fur et à mesure de la résolution, lorsqu'un problème apparaît, ou en fonction de la personnalité à simuler.

Les stratégies et les tactiques pouvant être activées à chaque étape dépendent de la phase choisie. On remarquera que les stratégies et les tactiques ne jouent aucun rôle en phases de coopération (puisque l'agent est dépendant des demandes des autres agents) et de test (une seule action à réaliser).

En retournant un épisode, éventuellement identique au précédent, construit selon les croyances, IGGY2 joue le rôle de la bibliothèque de plans dans les architectures BDI. En effet, les stratégies et tactiques sont la représentation abstraite d'une classe de plans. Par exemple, avec une planification séquentielle et prospective (stratégies), en passant par Paris et en favorisant son moyen de transport (tactiques), un ensemble de trajets peut être défini. Cet ensemble de trajets est traduit en désirs, intentions et actions par l'interpréteur de plans.

### 5.1.3 Interpréteur de plans

L'interpréteur de plans de BDI<sub>GGY</sub> manipule les désirs, intentions et actions de l'agent pour effectuer des commandes directement sur l'interface. Il comprend un générateur de désirs, un générateur d'intentions et un module d'exécution. Le générateur de désirs interprète un épisode

(une phase, un état d'esprit, deux stratégies et des tactiques) transmis par IGGY2 et le traduit en désirs. Ces désirs sont instanciés par le générateur d'intentions en intentions, puis traduites une à une en actions exécutées par le module d'exécution.

## Désirs, intentions et actions

Les désirs de l'agent sont contenus dans la pile de désirs, gérée par le générateur de désirs. Un désir peut être :

- *élémentaire* lorsqu'il peut être instancié directement en intentions ou *abstrait* s'il doit être raffiné en sous-désirs (élémentaires et/ou abstraits) avant ;
- *instancié* s'il a été traduit en intentions ajoutées à la pile d'intentions, ou *non instancié* sinon. Un désir ne peut être instancié que s'il est élémentaire ;
- *satisfait* (respectivement *insatisfait*) si la croyance portant sur le prédicat du désir (respectivement la négation du prédicat du désir) est contenue dans les croyances de l'agent ;
- *réussi* si le désir est satisfait ou insatisfait, *non réussi* si aucune des croyances de l'agent ne porte sur le prédicat du désir ou sur la négation de ce prédicat ;
- *en attente* si la prochaine intention liée au désir est elle-même en attente, *actif* sinon.

Par exemple, pour l'agent aérien,

```
pD(air pTravel(pStage(ChateauNeuf Maguelone * * * 4 <=(3300) true)))  
pD(air pStage(ChateauNeuf Paris * * train 4 ? false))
```

le premier désir est abstrait, alors que le second est élémentaire.

Les intentions de l'agent sont contenues dans la pile d'intentions, manipulée par le générateur d'intentions. Chaque intention est liée au désir qui l'a engendrée et ne peut perdurer lorsque ce désir est réussi (satisfait ou insatisfait) et a disparu de la pile de désirs. Les intentions sont empilées par le générateur d'intentions selon l'ordre des désirs auxquels elles appartiennent. Une intention est composée d'une suite d'actions, élémentaires ou non, dont les arguments sont précisés au moment de leur exécution. Une intention est :

- *active* si la première action de sa liste d'actions est exécutable immédiatement par le module d'exécution ;
- *en attente* si cette première action nécessite un événement particulier, extérieur à l'agent, avant de pouvoir être effectuée (par exemple l'arrivée d'un message).

Les actions listées dans les intentions et effectuées par le module d'exécution sont toutes :

- *externes* car elles n'agissent que sur l'interface ;
- *élémentaires* si elles correspondent à une commande unique sur l'interface ou *abstraites* si elles sont décomposables en plusieurs actions élémentaires. L'exécution d'une action abstraite revient, pour le module d'exécution, à effectuer séquentiellement toutes les actions élémentaires qui la composent.

Une action peut contenir des arguments dont la valeur n'est connue qu'au moment de l'exécution de l'action. Par exemple, si un agent désire réserver un trajet en train après 17h00, l'horaire exact n'est connu qu'au moment de l'affichage du trajet, c'est-à-dire à l'exécution de la réservation et non à la génération du désir ou de l'intention préalables. L'horaire à réserver sera donc un argument de l'action "book", et indiqué au moment de l'exécution de la commande.

Les actions sont transmises par le générateur d'intentions, une fois leurs arguments précisés, afin d'être effectuées par le module d'exécution sur l'interface.

Par exemple, lorsque le désir de l'agent aérien (celui portant sur l'étape ChateauNeuf-Paris en train), est instancié, sa pile d'intentions reçoit l'intention suivante :

```
LookForStage,pStage(ChateauNeuf Paris * * train 4 ? false)
— askDatabase,%stage
```

L'action *askDatabase,%stage* est l'unique action à réaliser pour satisfaire l'intention *LookForStage*. Cette action est abstraite car elle se décompose en une suite d'actions élémentaires (Sélectionner la ville de départ, sélectionner la ville d'arrivée, etc.). L'argument *%stage* ne prend sa valeur qu'au moment de l'exécution de l'action. Ainsi, l'intention peut être modifiée plus facilement (par exemple si une contrainte sur l'horaire est ajoutée).

De la même façon, l'instanciation du désir *pD(railway pStage(Nantes Lille ? ? plane 1 \* false)* chez l'agent ferroviaire conduit à l'ajout sur sa pile d'intentions de l'intention *AskForStage*, qui se décompose en 3 actions élémentaires :

```
AskForStage,air,pStage(Nantes Lille ? ? plane 1 * false)
— sendMessage,%agent,pD(%role %stage)
— waitMessage,%agent,pMessage(%agent %role reply pB(%stage))
  OR pMessage(%agent %role reply pB(!%stage))
— memorize,pMessage(%agent %role reply %stage)
```

La première action de l'intention (*sendMessage*) correspond à l'envoi d'un message contenant un désir de l'agent. La seconde action (*waitMessage*) synchronise l'intention en la mettant en attente d'une réponse de l'interlocuteur. Le contenu exact du message n'est pas connu à l'instanciation du désir et est mis à jour au moment de la réception du message. Enfin, la troisième action (*memorize*) n'est effectuée qu'à la réception d'une réponse et permet d'extraire une étape d'un message. Elle n'a aucun effet si la réponse est la négation d'une étape.

En l'état, cette intention est active, puisque sa première action est exécutable immédiatement par le module d'exécution. Par contre, après l'envoi du *query*, l'intention est mise à jour :

```
AskForStage,air,pStage(Nantes Lille ? ? plane 1 * false)
— waitMessage,%agent,pMessage(%agent %role reply pB(%stage))
  OR pMessage(%agent %role reply pB(!%stage))
— memorize,pMessage(%agent %role reply %stage)
```

et devient en attente, jusqu'à la réception du message permettant de la réactiver.

## Fonctionnement de l'interpréteur de plans

De façon globale, l'interpréteur de plans suit un cycle BDI. Son algorithme de fonctionnement est le suivant :

- 1 Mettre à jour et trier la[les] pile[s] de désirs
- 2 Mettre à jour et trier la[les] pile[s] d'intentions
- [3 Choisir une pile de désirs]
- 4 Si l'épisode a changé
  - 4.1 Mettre à jour l'épisode
  - 4.2 Tant que le désir au sommet de la pile de désirs n'est pas élémentaire alors
    - 4.2.1 Raffiner le désir du sommet de la pile des désirs
  - 4.3 Si le sommet de la pile de désirs n'est pas instancié alors
    - 4.3.1 Instancier le sommet de la pile de désirs en intentions
- 5 Sinon
  - 5.1 Si le sommet de la pile d'intentions est en suspens et que le sommet de la pile de désirs est élémentaire
    - 5.1.1 Instancier le sommet de la pile de désirs en intentions
  - 5.2 Sinon
    - 5.2.1 Tant que le désir du sommet de la pile de désirs n'est pas élémentaire alors
      - 4.2.1.1 Raffiner le sommet de la pile des désirs
    - 5.2.2 Si le désir au sommet de la pile de désirs n'est pas instancié alors
      - 5.2.2.1 Instancier le désir au sommet de la pile de désirs en intentions
- 6 Tant que l'intention au sommet de la pile d'intentions est active
  - 6.1 Exécuter la première action de l'intention du sommet de la pile d'intentions
  - 6.2 Mettre à jour la mémoire
  - 6.3 Mettre à jour la[les] pile[s] de désirs
  - 6.4 Mettre à jour la[les] pile[s] d'intentions
  - 6.5 Si l'intention du sommet de la pile d'intentions est vide
    - 6.5.1 Retirer l'intention du sommet de la pile d'intentions
  - 6.6 Si un désir de la pile de désirs peut être réactivé par l'arrivée d'un nouveau message
    - 6.6.1 Monter le désir réactivable au sommet de la pile
    - 6.6.2 Trier la pile d'intentions

L'interpréteur de plans commence par mettre à jour la pile de désirs et la pile d'intentions. Il s'agit tout d'abord de supprimer tous les désirs réussis (satisfaits et insatisfaits) de la pile des désirs, en fonction des croyances de l'agent. Toutes les intentions liées à un désir supprimé sont retirées de la pile d'intentions. Ensuite, les croyances non-traitées sont inspectées et, selon la phase courante et la personnalité de l'agent, exploitées si elles permettent d'ajouter un désir à la pile de désirs. Une fois utilisée, la croyance est retirée de la liste des croyances non-traitées. Enfin, la pile de désirs est explorée en descendant, et le premier désir actif rencontré est sélectionné. S'il n'y en a pas, c'est le premier désir élémentaire mais non-instancié qui est choisi. Sinon le premier désir abstrait. Le désir sélectionné est placé au sommet de la pile de désirs et la pile d'intentions est triée en fonction de l'ordre de la pile de désirs.

Une fois la pile de désirs et la pile d'intentions actualisées, le fonctionnement de l'interpréteur de plans dépend de l'épisode que lui transmet IGGY2.

- Au premier appel de l'interpréteur de plans et lorsqu'un changement d'épisode intervient (modification d'un des ingrédients), l'interpréteur de plans exécute successivement le générateur de désirs, le générateur d'intentions et le module d'exécution.
- Si l'épisode transmis par IGGY2 est identique à celui reçu précédemment, l'interpréteur de plans peut effectuer un cycle simplifié. Si l'intention au sommet de la pile d'intentions est active, le module d'exécution l'exécute immédiatement. Sinon, il demande au géné-

rateur d'intentions de générer de nouvelles intentions. Ce que ce dernier fait s'il reste des désirs à raffiner dans la pile de désirs, sinon il donne la main au générateur de désirs, pour qu'il génère de nouveaux désirs.

Lorsque le module d'exécution est appelé par l'interpréteur de plans, il effectue une à une les actions des intentions actives empilées sur la pile d'intentions. Entre chaque action, il met à jour la mémoire du module de perception, les désirs de la pile de désirs et les intentions de la pile d'intentions. Si l'arrivée d'un nouveau message peut permettre de réactiver un désir (cette information est gérée en coordination avec les modules de communication et de perception), celui-ci est remonté dans la pile de désirs et la pile d'intentions est triée en conséquence. L'exécution des actions s'arrête dès que l'intention au sommet de la pile d'intention n'est plus une intention active. Cette boucle interne au module d'exécution permet d'exécuter sans interruption une série d'actions élémentaires ne nécessitant pas de changement de phase.

### Sous-buts parallèles

Les sous-buts parallèles sont gérés dans la pile de désirs, à l'intérieur du désir portant sur le voyage à construire. Deux cas différents peuvent se présenter, quand le désir est raffiné :

- si les deux moyens de transport des sous-buts parallèles sont gérés par le même agent, un seul désir portant sur l'étape est utilisé, avec une disjonction de moyens de transport ;
- si les deux moyens de transport des sous-buts parallèles ne sont pas gérés par le même agent, deux désirs portant chacun sur une des deux étapes parallèles sont empilés.

Par exemple, si l'agent ferroviaire a, au sommet de sa pile de désirs, le désir suivant :

```
pD(railway pTravel(...
    pStage(Paris Lille ?? OR(train plane) 1 * false))
    pStage(Lille Baisieux ** OR(coach taxi) 1 ? false))
...))
```

alors, au moment d'être raffiné, ce désir entraîne l'ajout des trois désirs à la pile de désirs : le premier portant sur l'étape en train, le second sur l'étape en avion et le troisième sur l'étape en car ou taxi. La pile de désirs devient :

```
pD(railway pStage(Paris Lille ?? train 1 * false))
pD(railway pStage(Paris Lille ?? plane 1 * false))
pD(railway pStage(Lille Baisieux ** OR(coach taxi) 1 ? false))
pD(railway pTravel(...
    pStage(Paris Lille ?? OR(train plane) 1 * false))
    pStage(Lille Baisieux ** OR(coach taxi) 1 ? false))
...))
```

### Planification parallèle

La planification parallèle est, elle aussi, gérée par l'interpréteur de plans, si la stratégie retournée par IGGY2 est parallèle, et non séquentielle. Dans ce cas, le générateur de désirs

manipule plusieurs piles de désirs, une pile par plan parallèle, au lieu d'une pile unique. De même, le générateur d'intentions contient autant de piles d'intentions que le générateur de désirs compte de piles de désirs. À chaque fois qu'une pile est sélectionnée dans le générateur de désirs, sa pile d'intentions correspondante l'est aussi.

En planification séquentielle, quand un agent est en attente (il a construit une solution dont toutes les étapes sont soit déjà insérées dans son espace de travail, soit dépendantes d'un message d'un autre agent, et qu'au moins une étape est en attente de réponse), il passe en phase de vérification (par le biais d'IGGY2). En planification parallèle, par contre, quand l'agent est en attente, il construit un nouveau plan parallèle. Il gèrera donc autant de plans parallèles que nécessaires pour ne jamais être désoeuvré.

Tant que la planification reste parallèle, la pile de désirs (et la pile d'intentions correspondante) est sélectionnée en parcourant les piles par ordre de création, en prenant la première pile de désirs n'étant pas en attente. Lorsque la planification redevient séquentielle, au milieu de la résolution du problème, le générateur de désirs choisit la pile de désirs la plus avancée, au détriment des autres.

Le choix de l'utilisation de la planification parallèle, de sous-buts parallèles ou d'une simple planification séquentielle s'effectue selon la valeur du champ "opportuniste" de la personnalité de l'agent. Cependant, alors que la distinction entre planification parallèle et séquentielle s'effectue dans IGGY2, l'adoption ou non de sous-buts parallèles est gérée par l'interpréteur de plans.

## Construction d'un voyage

Initialement, le voyage à construire ne comporte qu'une seule étape correspondant au problème à résoudre. Cette étape est éventuellement raffinée en 2 sous-étapes passant par une ville intermédiaire :

- 1 Trier les étapes du voyage selon la stratégie courante
- 2 Prendre la première étape non renseignée de la liste triée
- 3 Si l'étape est connue comme étant impossible
  - 3.1 Modifier l'étape en conséquence
- 4 Sinon si l'étape doit être raffinée
  - 4.1 Raffiner l'étape en 2 sous-étapes
  - 4.2 Remplacer l'étape raffinée dans le voyage par les 2 sous-étapes
  - 4.3 Choisir une des 2 étapes comme l'étape à traiter
- 5 Remplir les champs de l'étape selon la personnalité
- 6 Remplacer l'étape raffinée dans le voyage

Le tri des étapes respecte la stratégie courante (prospective, rétrospective, centrifuge ou centripète). Par exemple, pour un voyage contenant 3 étapes  $\{e_1 e_2 e_3\}$ , on obtient

- en prospectif :  $\{e_1 e_2 e_3\}$
- en rétrospectif :  $\{e_3 e_2 e_1\}$
- en centrifuge :  $\{e_2 e_1 e_3\}$
- en centripète :  $\{e_1 e_3 e_2\}$

Une fois les étapes du voyage triées, la première étape de la liste triée sur laquelle aucune information n'est disponible est sélectionnée. Si cette étape est impossible (pas de ligne joignant

les deux villes avec le moyen de transport, horaires incompatibles, etc.), elle est modifiée en conséquence (changement de moyen de transport, relâchement de contrainte, raffinement d'une étape, etc.) avant d'être testée à nouveau.

Le choix de raffiner ou non une étape en 2 sous-étapes dépend du critère *optimisateur* et du nombre d'étapes du voyage :

- plus un agent est optimisateur, plus souvent il testera des étapes sans les raffiner,
- une étape aura plus de chance d'être raffinée en début de construction d'une solution (quand le voyage contient peu d'étapes), qu'en fin.

Les villes insérées comme étape intermédiaire dans un trajet sont choisies en fonction des tactiques (utilisation ou non de Paris), des connaissances de la base données (distances entre villes) et de la position de l'étape dans le voyage. Par exemple, pour un voyage  $\{... (v_1 v_2) ...\}$  raffiné en  $\{... (v_1 v_i) (v_i v_2) ...\}$ , on aura :

- si  $v_1 = v_d$  avec  $v_d$  la ville de départ du voyage, alors l'étape à raffiner est la première étape et  $v_i = villePlusProche(v_1)$  ;
- si  $v_2 = v_a$  avec  $v_a$  la ville d'arrivée du voyage, alors l'étape à raffiner est la dernière étape et  $v_i = villePlusProche(v_2)$  ;
- si le voyage ne passe pas par Paris alors que la tactique faisant passer par Paris est présente,  $v_i = Paris$  ;
- sinon  $v_i = villeEntre(v_1 v_2)$ .

Dans le cas particulier de la planification parallèle, une liste des villes déjà utilisées dans d'autres plans permet de restreindre la recherche d'une ville plus proche de la ville de départ ou de la ville d'arrivée du voyage. Ainsi, 2 plans parallèles ne seront jamais strictement identiques.

Enfin, le remplissage des champs de l'étape en cours de construction respecte les critères présents dans l'état d'esprit et la personnalité.

## Délégation auprès du module de communication

Le module d'exécution travaille en collaboration avec le module de communication. Lorsqu'il y a un envoi de message à effectuer, le module de communication se charge :

- de rédiger le message en fonction de l'état mental à transmettre et de l'état des automates,
- d'appliquer la sémantique du message,
- de mettre à jour l'état des automates.

Le module d'exécution récupère le message rédigé par le module de communication et l'envoie à l'agent correspondant *via* l'interface.

### 5.1.4 Module de communication

Le module de communication incorpore les modèles de l'interaction humaine construits à partir de l'analyse des protocoles expérimentaux du point de vue de l'interaction (voir 4.2 et 4.3). Il est appelé par :

- le *module de perception* pour interpréter le message sélectionné du panneau "Message-rie" et vérifier si la liste des nouveaux messages ne contient pas de message susceptible

d'être une réponse attendue. Le module de perception récupère aussi les informations sur l'état des échanges influant sur la résolution du problème (fermeture d'un automate temporisé, besoin d'effectuer une relance de demande, satisfaction ou insatisfaction d'un échange, etc.);

- le *module d'exécution* pour rédiger un message à envoyer en fonction de l'état mental et de l'état des automates décrivant les échanges en cours.

À la lecture et à l'écriture des messages, le module de communication parcourt les automates temporisés ouverts (qui caractérisent les échanges en cours) pour savoir si le message appartient à l'un d'entre eux. Pour cela, il utilise l'algorithme suivant :

- 1 Lister les automates ouverts pouvant être mis à jour avec l'émetteur, le destinataire et la performative du message
- 2 Si un seul automate peut être mis à jour avec le message
  - 2.1 Mettre à jour l'automate avec le message
- 3 Sinon si plusieurs automates peuvent être mis à jour avec le message
  - 3.1 Sélectionner l'automate à mettre à jour selon le contenu du message
  - 3.2 Mettre à jour l'automate avec le message
- 4 Si aucun automate ouvert ne correspond au message
  - 4.1 Ouvrir un nouvel automate avec le message
- 5 Si après avoir été mis à jour, l'automate ne peut qu'être fermé
  - 5.1 Supprimer l'automate de la liste des automates ouverts
  - 5.2 Ajouter aux croyances non traitées les croyances relatives à la fermeture de l'automate

Les automates temporisés permettent donc de savoir, dans la grande majorité des cas, à quel échange appartient un message, sans avoir à analyser son contenu. Lorsque la performative, l'émetteur et le destinataire ne suffisent pas à cela, le contenu du message est analysé pour lever l'ambiguïté. En effet, pour chaque échange initié (*i.e.* pour chaque automate ouvert), l'intention discursive est stockée. Il suffit de comparer le contenu du message avec l'intention discursive, grâce à une forme de *pattern matching*, pour supprimer la difficulté d'interprétation. Ce processus étant assez coûteux, il n'est utilisé qu'en cas d'ambiguïté.

Dans le cas où le message n'appartient à aucun échange, un nouvel automate est ouvert en fonction de la performative et des deux participants. Par exemple, un *query* envoyé par un autre agent permettra l'ouverture d'un automate *Qint*, décrivant le comportement de l'interlocuteur d'une demande d'information.

La rédaction du message est elle aussi réalisée par le module de perception, à la demande du module d'exécution. Ce dernier transmet l'état mental à envoyer, l'émetteur et le destinataire, et reçoit le message rédigé. La construction du message se fait grâce à l'algorithme suivant, en fonction de l'état mental à transmettre :

- 1 Si l'état mental est une croyance,
  - 1.1 Si la croyance porte sur un problème alors c'est un inform
  - 1.2 Sinon si la croyance porte sur un message alors c'est un notUnderstood
  - 1.3 Sinon si la croyance porte sur un trajet alors
    - 1.3.1 Si le trajet a été fait par l'agent alors c'est un reply
    - 1.3.2 Sinon c'est un thank
- 2 Sinon si l'état mental est un désir de l'agent
  - 2.1 Si le désir porte sur une étape
    - 2.1.1 Si un automate Pint est ouvert alors c'est un acceptProposal
    - 2.1.2 Sinon c'est un query

- 2.2 Si le prédicat porte sur la négation d'un message alors c'est un `cancel`
- 2.3 Sinon c'est un `refine`
- 3 Sinon si l'état mental est un désir d'un autre agent alors c'est un `propose`
- 4 Sinon l'état mental est la négation d'un désir de l'agent et c'est un `refuseProposal`

Cependant, un envoi de message ne peut pas être effectué à n'importe quel moment.

- Si le message concerne la résolution de son propre problème, il est réalisé en phase de planification.
- S'il porte sur la résolution d'un problème différent du sien, il est réalisé en phase de coopération.
- S'il s'agit d'un message appartenant à un échange de traitement des erreurs, il est réalisé à n'importe quelle phase.

De plus, lorsque l'interpréteur de plans ne peut pas décider seul quel message doit être envoyé, le module de communication se charge du choix, en respectant les fréquences (les annotations des transitions des automates temporisés) observées dans les protocoles de communication. Typiquement, à la fin d'un automate de demande d'information ( $Q_{ini}$ ), en cas de satisfaction, le module de communication tire aléatoirement si un *thank* doit être envoyé.

## 5.2 Fonctionnement de BDIGGY

Après une phase d'initialisation, l'architecture BDIGGY suit, dans son fonctionnement, une boucle de type :

1. *perception* (module de perception),
2. *décision* (le système IGGY2, le générateur de désirs et le générateur d'intentions),
3. *action* (le module d'exécution).

### 5.2.1 Initialisation

La *phase d'initialisation* permet de fixer la personnalité de l'agent, reflet des différents traits de caractère du sujet à simuler, et le problème à résoudre.

Pour fixer la personnalité de l'agent, *attentif*, *coopératif*, *économe*, *bon estimateur*, *opportuniste*, *optimisateur*, *patient* et *précis*, prennent chacun une valeur dans  $\{peu=1, assez=2, beaucoup=3\}$ . Les valeurs données correspondent à des comportements observés durant l'analyse des protocoles expérimentaux.

De plus, un premier désir, correspondant au désir de l'agent de résoudre son problème, est inséré dans la pile des désirs :

$$pD(\text{agent } p\text{Problem}(\text{agent true}))$$

Le problème de l'agent n'est donc pas introduit directement dans ses désirs. Il n'est connu qu'au moment où l'interpréteur de plans raffine le désir initial (généralement à la première étape), après une requête auprès de la partie de sa mémoire contenant la représentation du problème. Le désir raffiné portera sur un voyage :

$$pTravel(pS)$$

où  $pS \in STA$  est l'étape décrivant le problème de l'agent (voir 4.1.5).

## 5.2.2 Perception - Décision - Action

À l'issue de la phase d'initialisation, un cycle complet de type perception, décision, action est effectué. Ce cycle sera répété jusqu'à la fin de la simulation :

- 1 Tant que la simulation n'est pas terminée
- 2.1 Mettre à jour la mémoire dans le module de perception
- 2.2 Générer un épisode dans Iggy2
- 2.3 Effectuer une(des) action(s) externe(s)

Le module de perception met à jour la mémoire de l'agent en fonction de l'interface et des échanges en cours. Il traduit du Java en Lisp les informations de la mémoires utiles à IGGY2.

Le module de planification humaine IGGY2 génère de nouvelles observations sur les croyances de l'agent et construit, si nécessaire, un nouvel épisode.

Le générateur de désir se charge de traduire un épisode en désirs, d'instancier les désirs en intentions et d'effectuer les actions liées aux intentions.

## 5.2.3 Liens entre planification et interaction

Durant la résolution simultanée des trois problèmes, planification et interaction s'entremêlent pour les agents. Chaque agent est dépendant des autres pour obtenir des informations manquantes et doit collaborer. Les liens entre module de communication et l'interpréteur de plans sont visibles au niveau des échanges d'information :

- Dans le sens interpréteur de plans → module de communication, c'est la planification qui dirige l'interaction. C'est le cas le plus courant, l'interaction étant vue comme un outil permettant de récupérer des informations.
- Dans le sens module de communication → interpréteur de plans, c'est l'interaction qui intervient dans le déroulement des actions. Le module de communication agit de façon directive auprès de l'interpréteur de plans pour les relances de demande, les envois de remerciements et la gestion des messages erronés (pour ces deux derniers points, voir aussi la section suivante).

Les échanges d'informations entre module de communication et interpréteur de plans se font la plupart du temps par le biais de croyances ajoutées aux croyances non traitées. Par exemple, si l'agent aérien reçoit de l'agent routier le message

$pMessage(road\ air\ query\ pD(road\ pStage(Paris(airport)\ Lyon(airport)\ (18:00)\ ?\ plane\ *?\ false)))$ ,

alors il ajoutera à ses croyances non-traitées la croyance

$pB(pD(road\ pStage(Paris(airport)\ Lyon(airport)\ (18:00)\ ?\ plane\ *?\ false)))$ .

Répondre ou non à cette demande n'est pas décidé par le module de communication mais par IGGY2 et l'interpréteur de plans. De la même manière sont ajoutées aux croyances non traitées, des croyances portant sur :

- une étape (ou la négation d'une étape) extraite d'un *reply*, transférée ensuite dans les croyances persistantes,
- le désir d'un autre agent, extrait d'un *query*,
- un désir pour une étape, extrait d'un *propose*,
- le désir d'un autre agent d'annuler un message, extrait d'un *cancel*,
- un désir de relancer une demande n'ayant pas reçu de réponse,
- etc.

De plus, la résolution simultanée des trois problèmes complexifie les interactions possibles. En effet, une demande provenant d'un agent ne peut être traitée qu'en phase de coopération, phase qui n'est choisie ni par le module de communication, ni par l'interpréteur de plans, mais par IGGY2. Plusieurs cas peuvent se présenter, la figure 5.4 en présente une liste non exhaustive.

Le cas 1 est le cas le plus courant, lorsque durant une phase de planification, un message arrive en provenance d'un agent dont on attend une réponse. Cette réponse est traitée à l'intérieur de la même phase de planification. Cependant, les messages ne sont pas toujours ce que l'agent croit. Dans le cas 2 par exemple, le message reçu n'est pas la réponse attendue mais est en réalité une demande. Elle n'est traitée qu'à la phase de coopération suivante, décidée par IGGY2.

Dans le cas 3, l'agent est en phase de vérification (de trajets ou de messages). Cette phase n'est interrompue que si tous les trajets ou tous les messages ont été vérifiés, si un problème ou l'arrivée d'un nouveau message est détecté. Ici, l'agent reçoit un message alors qu'il était en attente d'une réponse et passe donc en phase de planification. S'il n'avait pas été en attente, il serait passé en phase de coopération.

Le cas 4 est le cas le plus simple de coopération : l'agent est en phase de planification, lorsqu'il reçoit un message en provenance d'un agent duquel il n'attend aucune information. Il pense donc qu'il s'agit d'une demande. Celle-ci est traitée dès qu'IGGY2 décide de passer en phase de coopération. Par contre, si le message n'est finalement pas une demande (cas 5), comme par exemple un *notUnderstood*, il est traité en phase de planification. Le message nécessite alors un passage par IGGY2 et un changement de phase pour être utilisé.

Enfin, le cas 6 présente l'enchaînement de 2 phases de coopération successives, toutes les deux se terminant par la satisfaction de la demande.

Tous ces exemples sont supportés par BDI<sub>GGY</sub>. Par contre, notons l'inconvénient d'une boucle de type perception-décision-action car tout événement inattendu (demande à la place d'une réponse par exemple) nécessite un passage supplémentaire pour changer la phase.

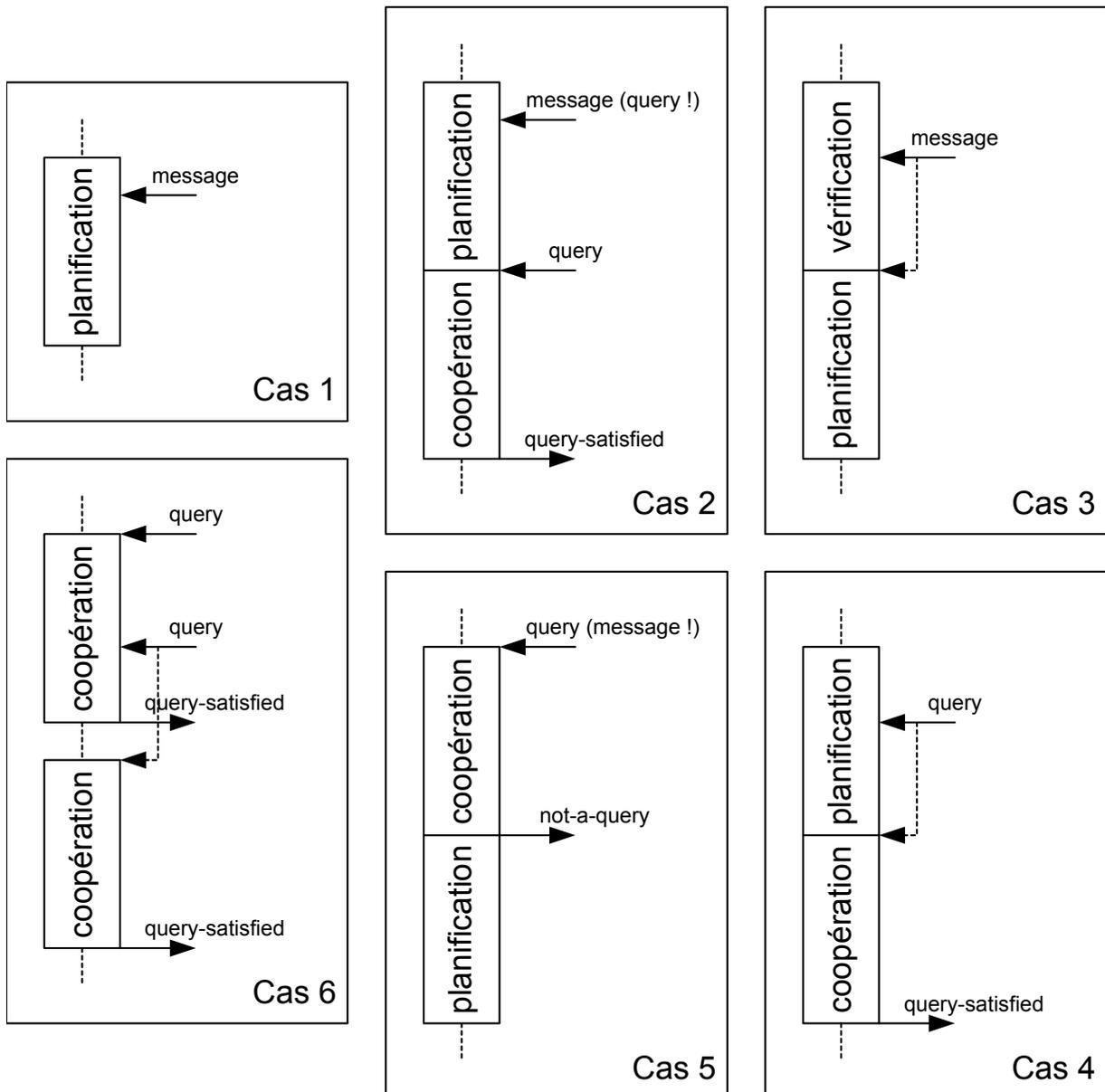


FIG. 5.4 – Enchaînements entre interaction et planification

## 5.2.4 Spécificités des dialogues observés

De nombreuses imprécisions et erreurs jalonnent les échanges observés durant la résolution. Ceux-ci sont simulés dans BDIGGY, au niveau de l'interpréteur de plans.

### Gestion des imprécisions

Les imprécisions dans les messages (absence d'horaire et/ou de prix, trajet demandé sans préciser gare ou aéroport, oubli du nombre de personnes pour une réservation, etc.) sont gérées directement par le générateur de désirs. Selon les critères présents dans l'état d'esprit au moment de la construction d'un trajet et le trait de caractère "précis", les informations recherchées par le sujet seront plus ou moins précises.

Par exemple, pour un agent ferroviaire recherchant un trajet Paris-Lille en avion, un état d'esprit vide entraînera le désir

$$pD(railway?pStage(Paris Lille ** plane ** false))$$

qui correspond à la demande "Y a-t-il une ligne d'avion reliant Paris à Lille".

Par contre, un état d'esprit de type (*price timetable travelers*) permettra d'ajouter à la pile de désirs le désir

$$pD(railway?pStage(Paris Lille morning * plane 1 ? false))$$

qui correspond à la demande "Y a-t-il un avion Paris-Lille partant le matin et à quel prix".

### Gestion des erreurs

Les erreurs sont principalement dues à des envois de message au mauvais destinataire. Le champ destinataire est rempli selon le trait de caractère *attentif*, par l'exécuteur de plan. Cette erreur n'est pas visible au niveau des automates qui représentent les conversations telles que l'agent les a voulues (différentes de la réalité).

L'agent recevant le message erroné initie un dialogue de type  $E_{ini}$ , avec l'envoi d'un *notUnderstood*. À la réception de ce *notUnderstood* chez l'agent responsable de l'erreur, celle-ci est reconnue et traitée par un automate de type  $E_{int}$  permettant d'annuler le mauvais message (envoi d'un *cancel*). De plus, le module de communication effectue un retour en arrière dans l'automate ayant traité le message erroné, de façon à représenter l'état réel des interactions en cours, et non l'état supposé. Enfin, l'interpréteur de plans est informé qu'une série d'intentions n'a pas été effectuée correctement et doit donc être réitérée.

### Gestion des remerciements

L'envoi d'un remerciement (*thank*) est décidé par le module de communication à la réception d'un *reply* jugé satisfaisant. La proportion d'envoi de messages de ce type respecte les

proportions observées dans les protocoles expérimentaux et représentées par les annotations sur les arcs des automates temporisés.

Cependant, les coefficients du système (pourcentage d'erreur d'expéditeur en fonction de la personnalité, temps écoulé entre deux relances, etc.) ont dû être optimisés pour correspondre aux comportements observés, selon chaque personnalité. Ce calibrage a été réalisé uniquement en fonction des 9x3 protocoles analysés. Malheureusement, il n'existe aucune méthode automatique pour cela et ce paramétrage s'est fait manuellement, par tâtonnements successifs. Cette étape est primordiale pour avoir une simulation satisfaisante et sa durée est souvent sous-estimée.

### 5.2.5 Réglage du système

Le système BDIGGY a fait l'objet d'une phase de réglage pour optimiser les paramètres de la simulation. Certains de ces paramètres étaient quantifiables directement par analyse des protocoles expérimentaux. D'autres ont dû être fixés indirectement, en fonction du comportement à modéliser.

Voici une liste non exhaustive de ces paramètres :

- le nombre moyen de relances de demande sans réponse avant d'abandonner un échange,
- le temps moyen entre 2 relances (la simulation ne tient pas compte des délais de planification ou de rédaction des messages),
- les proportions d'utilisation de chaque stratégie et de chaque tactique,
- les nombres moyens d'erreur en fonction de chaque personnalité,
- etc.

### 5.2.6 Exemple de fonctionnement

L'exemple qui suit est tiré d'une trace générée par un agent BDIGGY jouant le rôle de l'agent ferroviaire<sup>2</sup>. Sa personnalité est : *attentif=3, coopératif=3, économe=1, bon estimateur=1, opportuniste=1, optimisateur=1, patient=3, précis=1*.

Au démarrage de la simulation, l'interface est vide, de même que la mémoire et la liste des automates ouverts<sup>3</sup> :

```
Persistent beliefs: []  
Unprocessed beliefs: []
```

```
Opened Automata: []
```

Le module IGGY2 génère l'épisode suivant :

```
Episode:  
Phase = planning  
StateOfMind = (timetable transfert)  
Strategies = (sequential centrifugal)  
Tactics = (Paris notOptimized own)
```

---

<sup>2</sup>Un exemple de protocole complet généré par un agent BDIGGY est donné en ANNEXE C.

<sup>3</sup>Pour ne pas surcharger la trace de fonctionnement de BDIGGY, il a été décidé de n'y faire figurer que les parties de la mémoire qui concerne les croyances persistantes et les croyances non traitées.

Au premier appel de l'interpréteur de plans, la pile de désirs ne contient qu'un unique désir portant sur le problème à résoudre et la pile des intentions est vide :

```
desireStack: [pD(railway pProblem(railway true))]  
intentionStack: []
```

Le générateur de désirs raffine le premier désir de la pile de désirs et y empile de nouveaux désirs. La pile de désirs est unique car la planification se fait de façon séquentielle. De plus, le désir raffiné est Paris-Lille car la stratégie choisie est centrifuge.

```
desireStack: [  
  pD(railway pStage(Paris(gare) Lille(gare) ? <=(11:00) train * * false))  
  pD(railway pTravel(pStage(Baisieux Nantes * * * * * *)  
    pStage(Nantes Paris * * * * * *)  
    pStage(Paris(gare) Lille(gare) ? <=(11:00) train * *  
      false)  
    pStage(Lille Baisieux * * * * * *)))  
  pD(railway pProblem(railway true))  
]
```

Le générateur d'intentions récupère le premier désir de la pile de désirs et l'instancie en intentions.

```
intentionStack: [  
  pI(LookForStage,%stage:  
    askDatabase,pStage(Paris(gare) Lille(gare) ? <=(11:00) train * *  
      false))  
]
```

Le module d'intentions exécute alors la première intention, dont l'action `askDatabase` consiste à interroger la base de données et à ajouter un horaire choisi à la zone de travail. Comme il s'agit d'actions externes, elles apparaissent dans le protocole généré par l'interface :

```
[Ferroviaire] : Demande horaires Paris-Lille ()  
[Ferroviaire] : Memorise horaire  
[Ferroviaire] : Ajoute horaire dans panneau Mon probleme  
Paris-Lille (train)  
Date : Demain – 1 personne  
09h01/10h01 - 250 F
```

La pile d'intentions est alors vide et le cycle perception-décision action est terminé :

```
Intention stack after the execution module:  
intentionStack: []
```

Dans le même temps, l'agent aérien envoie un message à l'agent ferroviaire :

```
[Ferroviaire] : Recoit message 1 de Agent aerien
```

Un nouveau cycle débute avec une mise à jour de la description de l'interface. La description du panneau "Horaires/Prix" contient maintenant la liste des horaires affichés (sous la forme de *pStage*), la description de la "Zone de travail" contient le *pStage* décrivant le trajet Paris-Lille et la description de la "Messagerie" contient un *pMessage* dans la liste des messages non lus. De plus, l'agent ajoute à ses croyances persistantes l'existence d'une ligne de train entre Paris et Lille et un horaire particulier :

```
Persistent beliefs: [
  pB(pStage(Paris(gare) Lille(gare) * * train * * *))
  pB(pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250 false))
]
Unprocessed beliefs: []

Opened Automata: []
```

Concernant le message non lu, l'agent ferroviaire n'ayant aucun échange en cours avec l'agent aérien, il s'agit donc sans doute d'une demande. Le module de perception informe IGGY2 de l'existence de cette demande. Comme l'agent ferroviaire est très coopératif, il décide de passer en phase de coopération :

```
Episode:
  Phase = cooperation
  StateOfMind = ()
  Strategies = ()
  Tactics = ()
```

Dans l'interpréteur de plans, la pile de désirs est mise à jour : le désir du sommet de la pile ayant été satisfait, il est supprimé, et un désir d'ouvrir le nouveau message est ajouté. L'intention correspondante est aussi ajoutée à la pile d'intentions :

```
desireStack: [
  pD(railway pSelectedMessage(pMessage(air railway * *)))
  pD(railway pTravel(pStage(Baisieux Nantes * * * * * *)
    pStage(Nantes Paris * * * * * *)
    pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250
      false)
    pStage(Lille Baisieux * * * * * *)))
  pD(railway pProblem(railway true))
]

activeIntentionStack: [
  pI(SelectUnreadMessage,%agent: selectUnreadMessage,air)
]
```

Au niveau du module d'exécution, cela se traduit par la sélection du message dans l'interface, visible dans le protocole généré par l'interface :

```
[Ferroviaire] : Affiche message numero 1 de Agent aerien
De : Agent aerien A : Agent ferroviaire
pMessage(air railway query pD(air pStage(Paris Montpellier ?? train 4 ? false)))
```

Cette dernière action clôture la seconde boucle. Dans la troisième boucle, la description de la "Messagerie" comporte un nouveau message affiché, traité à l'aide du module de communication. La liste des automates ouverts et les croyances non traitées sont donc modifiées :

```
New message processed: pMessage(air railway query pD(air pStage(Paris
                                Montpellier ? ? train 4 ? false)))

Persistent beliefs: [
  pB(pStage(Paris(gare) Lille(gare) * * train * * *))
  pB(pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250 false))
]
Unprocessed beliefs: [pB(pD(air pStage(Paris(gare) Montpellier(gare)
                            ? ? train 4 ? false)))]

Opened Automata: [Qint(railway,air):s1]
```

IGGY2 maintient la phase de coopération car la demande de l'agent aérien n'est toujours pas satisfaite. Dans la pile de désirs, le désir du sommet ayant été satisfait, il est supprimé, et le désir en provenance de l'agent aérien est ajouté. La liste des croyances non traitées est vidée car la seule croyance qu'elle contenait a été utilisée.

```
Episode:
  Phase = cooperation
  StateOfMind = ()
  Strategies = ()
  Tactics = ()

Persistent beliefs: [
  pB(pStage(Paris(gare) Lille(gare) * * train * * *))
  pB(pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250 false))
]
Unprocessed beliefs: []

desireStack: [
  pD(air pStage(Paris(gare) Montpellier(gare) ? ? train 4 ? false))
  pD(railway pTravel(pStage(Baisieux Nantes * * * * * *)
                    pStage(Nantes Paris * * * * * *)
                    pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250
                          false)
                    pStage(Lille Baisieux * * * * * *)))
  pD(railway pProblem(railway true))
]
```

La pile des intentions contient maintenant deux nouveaux désirs. Le premier permet de récupérer un horaire et le second de l'envoyer à l'agent aérien.

```
intentionStack: [
  pI(LookForStage,%stage:
    askDatabase,pStage(Paris(gare) Montpellier(gare) ? ? train 4 ?
                      false))
  pI(ReplyStage,%agent,%stage:
    wait,pStage(Paris(gare) Montpellier(gare) ? ? train 4 ? false)
    sendStage,air,pB(pStage(Paris(gare) Montpellier(gare) ? ?
                          train 4 ? false)))
]
```

Après la première action, la pile des intentions contient une intention de moins. De plus, l'interface a été modifiée, et la mémoire est donc mise à jour par le module de perception.

```
Persistent beliefs: [  
  pB(pStage(Paris(gare) Lille(gare) * * train * * *))  
  pB(pStage(Paris(gare) Lille(gare) 09:01 10:01 train 1 250 false))  
  pB(pStage(Paris(gare) Montpellier(gare) * * train * * *))  
  pB(pStage(Paris(gare) Montpellier(gare) 10:30 14:39 train 1 590 false))  
]  
Unprocessed beliefs: []
```

L'agent reste en phase de coopération et, comme la pile d'intentions n'est pas vide, il donne la main directement au module d'exécution qui effectue l'intention suivante. La première action de l'intention est une action de synchronisation qui permet de savoir si un horaire satisfaisant le désir a bien été ajouté à la "Zone de travail". C'est effectivement le cas donc l'action est réalisée. L'action d'après consiste à envoyer un message, ce que l'agent fait.

```
[Ferroviaire] : Nouveau message  
[Ferroviaire] : Envoie message 1  
De : Agent ferroviaire A : Agent aerien  
pMessage(railway air reply pB(pStage(Paris Montpellier ?? train 4 ? false)))  
=====HORAIRE=====  
Paris-Montpellier (train)  
Date : Demain - 1 personne  
10h30/14h39 - 590 F
```

La simulation se poursuit ainsi, en suivant à chaque fois un cycle perception-décision-action, jusqu'à la construction d'une solution complète. Pour ne pas surcharger l'écriture, seules les indications indispensables à la compréhension sont maintenant reportées. Le voyage construit par l'agent dans la suite de la simulation est composé de 4 trajets différents : Beaupréau/Nantes en taxi, Nantes/Paris en avion, Paris/Lille en train et Lille/Baisieux en taxi. Toutes ces étapes ont été insérées dans la zone "Zone de travail" de l'agent, qui planifie maintenant une dernière vérification du voyage avant d'insérer les différentes étapes dans le panneau "Proposition":

```
Persistent beliefs: [...]  
Unprocessed beliefs: []
```

```
Communication module: [...]
```

```
Episode:  
  Phase = planning  
  StateOfMind = (travelers timetable transfert price)  
  Strategies = (sequential prospective)  
  Tactics = (Paris notOptimized cheapest)
```

```
activeDesireStack: [  
  pD(railway pLastCheck())  
  pD(railway pTravel(pStage(Beaupreau Nantes(airport) * +(00:35) taxi  
    1 250 false)  
    pStage(Nantes(airport) Paris(airport) 16:45  
      17:45 plane(PigeonVol) 1 595 false)  
    pStage(Paris Lille(station) 09:01 10:01 train 1
```

```

                250 false)
                pStage(Lille(station) Baisieux * +(00:30) taxi 1
                    180 false))
        pD(railway pProblem(railway true))
    ]

activeIntentionStack: [
    pI(AddUnprocessedBelief,%belief:    addBelief,pB(pLastCheck()))
    ]

```

**Cette action ajoute une croyance aux croyances non traitées, de façon à ce que l'interpréteur de plans puisse informer IGGY2 que le voyage est prêt à vérifié :**

```

Persistent beliefs: [...]
Unprocessed beliefs: [pB(pLastCheck())]

Communication module: [...]

```

**La croyance non traitée ajoutée est traitée par IGGY2 à la boucle suivante et permet le passage en phase de vérification :**

```

Episode:
    Phase = checking
    StateOfMind = (travelers timetable transfert price)
    Strategies = (stage)
    Tactics = (retrospective)

activeDesireStack: [
    pD(railway pStageChecked(pTravel(
        pStage(Beaupreau Nantes (airport) * +(00:35) taxi 1 250 false)
        pStage(Nantes (airport) Paris (airport) 16:45 17:45
            plane(PigeonVol) 1 595 false)
        pStage(Paris Lille(station) 09:01 10:01 train 1 250 false)
        pStage(Lille(station) Baisieux * +(00:30) taxi 1 180 false)))
    pD(railway pTravel(pStage(Beaupreau Nantes (airport) * +(00:35) taxi
        1 250 false)
        pStage(Nantes (airport) Paris (airport) 16:45
            17:45 plane(PigeonVol) 1 595 false)
        pStage(Paris Lille(station) 09:01 10:01 train 1
            250 false)
        pStage(Lille(station) Baisieux * +(00:30) taxi 1
            180 false)))
    pD(railway pProblem(railway true))
    ]

activeIntentionStack: [
    pI(CheckStages,%stateOfMind,%tactic,%travel,%first:
        checkNextStage,(travelers timetable transfert price),
        retrospective,
        pTravel(pStage(Beaupreau Nantes (airport) * +(00:35) taxi 1
            250 false)
            pStage(Nantes (airport) Paris (airport) 16:45 17:45
                plane(PigeonVol) 1 595 false)
            pStage(Paris Lille(station) 09:01 10:01 train 1 250 false)
            pStage(Lille(station) Baisieux * +(00:30) taxi 1 180
                false)),true)
    ]

```

BDIGGY boucle dans le module d'exécution qui sélectionne et vérifie les étapes du voyage présentes dans l'interface. Ici, la phase de vérification est interrompue par une erreur d'horaire à l'arrivée du trajet entre Nantes et Paris. Au niveau du protocole généré par l'interface, seul l'affichage des trajets vérifiés est visible :

```
[Ferroviaire] : Affiche horaire
Lille(gare)-Baisieux ( taxi)
Date : Demain – 1 personne
duree : 00h30 - 180 F
```

```
[Ferroviaire] : Affiche horaire
Paris-Lille (train)
Date : Demain – 1 personne
09h01/10h01 - 250 F
```

```
[Ferroviaire] : Affiche horaire
Nantes-Paris (avion Pigeon Vol)
Date : Demain – 1 personne
16h45/17h45 - 595 F
```

Cette erreur est référencée dans les croyances non traitées :

```
Persistent beliefs: [...]
Unprocessed beliefs: [
  pB(pPlanningProblem(timetable-next pStage(Nantes(airport)
    Paris(airport) 16:45 17:45 plane(PigeonVol) 1 595 false)))
]
Communication module: [...]
```

L'agent passe en phase alors de correction pour modifier son voyage. Les corrections sont prises en compte dans la phase de planification qui suit, après notification dans les croyances non traitées que la phase de correction s'est déroulée correctement.

```
Episode:
Phase = correction
StateOfMind = (travelers timetable transfert price)
Strategies = (sequential prospective)
Tactics = (Paris not-optimized cheapest)

activeDesireStack: [
  pD(railway pCorrectedProblem(pPlanningProblem(timetable-next
    pStage(Nantes(airport) Paris(airport) 16:45 17:45
      plane(PigeonVol) 1 595 false))))
  pD(railway pStage(Nantes(airport) Paris(airport) 16:45 17:45
    plane(PigeonVol) 1 595 false))
  pD(railway pTravel(pStage(Beaupreau Nantes(airport) * +(00:35) taxi
    1 250 false)
    pStage(Nantes(airport) Paris(airport) 16:45
      17:45 plane(PigeonVol) 1 595 false)
    pStage(Paris Lille(station) 09:01 10:01 train 1
      250 false)
    pStage(Lille(station) Baisieux * +(00:30) taxi 1
```

```

                                180 false)))
pD(railway pProblem(railway true))

activeIntentionStack: [
  pI (AddUnprocessedBelief, %belief:
    addBelief, pB (pCorrectedProblem (pPlanningProblem (timetable-next
      pStage (Nantes (airport) Paris (airport) 16:45 17:45
        plane (PigeonVol) 1 595 false))))))
  ]

[...]

Persistent beliefs: [...]
Unprocessed beliefs: [
  pB (pCorrectedProblem (pPlanningProblem (timetable-next
    pStage (Nantes (airport) Paris (airport) 16:45 17:45
      plane (PigeonVol) 1 595 false))))
  ]

Communication module: [...]

Episode:
  Phase = checking
  StateOfMind = (travelers timetable transfert price)
  Strategies = (stage)
  Tactics = (retrospective)

activeDesireStack: [
  pD(railway pStage (Nantes (airport) Paris (airport) <=(09:01) ? train 1 ?
    false))
  pD(railway pTravel (pStage (Beaupreau Nantes (airport) * +(00:35) taxi
    1 250 false)
    pStage (Nantes (airport) Paris (airport) 16:45
      17:45 plane (PigeonVol) 1 595 false)
    pStage (Paris Lille (station) <=(09:01) ? train 1 ?
      false)
    pStage (Lille (station) Baisieux * +(00:30) taxi 1
      180 false)))
  pD(railway pProblem(railway true))
]

```

Les phases de vérification et de correction s'enchainent jusqu'à ce que la solution soit jugée satisfaisante par l'agent. Il introduit alors les étapes constitutives du voyage dans le panneau proposition, avant de tester la solution par un appui sur le bouton Tester.

[Ferroviaire] : Affiche horaire  
 Reservation effectuee :  
 Beaupreau-Nantes(gare) ( taxi)  
 Date : Demain – 1 personne  
 duree : 00h45 - 300 F

[Ferroviaire] : Propose horaire [R] Beaupreau-Nantes(gare) (taxi)  
 [Ferroviaire] : Le prix total est : 300

[Ferroviaire] : Affiche horaire  
 Reservation effectuee :

Nantes-Paris (train)  
Date : Demain – 1 personne  
06h28/08h50 - 215 F

[Ferroviaire] : Propose horaire [R] Nantes-Paris (train)  
[Ferroviaire] : Le prix total est : 515

[Ferroviaire] : Affiche horaire  
Reservation effectuee :  
Paris-Lille (train)  
Date : Demain – 1 personne  
09h01/10h01 - 250 F

[Ferroviaire] : Propose horaire [R] Paris-Lille (train)  
[Ferroviaire] : Le prix total est : 765

[Ferroviaire] : Affiche horaire  
Reservation effectuee :  
Lille(gare)-Baisieux ( taxi)  
Date : Demain – 1 personne  
duree : 00h30 - 180 F

[Ferroviaire] : Propose horaire [R] Lille(gare)-Baisieux (taxi)  
[Ferroviaire] : Le prix total est : 945

[Ferroviaire] : Test de la solution : positif

Si l'agent est le dernier à terminer son problème, la simulation s'arrête immédiatement, sinon l'agent passe en phase de coopération pour répondre le plus rapidement possible aux sollicitations des autres agents.

## 5.3 Conclusion

L'architecture BDIGGY que nous proposons est très proche d'une architecture BDI classique. L'interpréteur de plan, qui contient le générateur de désirs, le générateur d'intentions et le module d'exécution, réalise des cycles de façon similaire à un système BDI.

Par rapport à une architecture BDI classique, BDIGGY propose trois changements.

- La queue d'événements est remplacée par le module de perception qui synthétise les changements de l'environnement. Il contient l'ensemble des croyances de l'agent.
- La bibliothèque de plans est remplacée par le module de planification humaine IGGY2. IGGY2 génère des observations qui sont similaires à des croyances, filtrées selon la situation. De plus, les plans sont ici construits de façon opportuniste : les stratégies et les tactiques sont une description abstraite d'une classe de plans.
- Le module de communication permet aux agents d'interagir aisément. Il contient la sémantique des performatives pour traiter les énoncés et les automates temporisés pour gérer les conversations.

Les deux principaux avantages de cette architecture sont, tout d'abord, qu'elle inclut un module de communication pour coopérer avec les autres agents et ensuite que les plans sont générés de façon opportuniste, selon les changements de l'environnement. Cela permet de simuler la construction opportuniste de plans des humains. De plus, la communication et la planification sont ici représentées de façon homogène (par des BDI), dans un même système.



# Chapitre 6

## Validation

Ce chapitre s'inspire largement de la validation faite dans [Chaignaud, 1996].

La validation est un problème crucial. En informatique, il s'agit de prouver la correction d'un programme, alors que notre approche est plus globale. Il s'agit de montrer qu'il est possible de modéliser de façon implantable des comportements humains de résolution de problèmes en contexte coopératif.

Les modèles de la planification et de l'interaction humaine du point de vue de la modélisation cognitive et l'architecture BDIGGY ne peuvent être validés que conjointement ; les processus de raisonnement des modèles ne s'observent que grâce à la simulation par agents BDIGGY.

### 6.1 La problématique de la validation

Une part importante de la modélisation cognitive est de pouvoir mesurer la pertinence de la description proposée des comportements humains. Toutefois, comme l'écrit [Gonzalez, 1991]

*La validation qui, dans son sens le plus étroit, est l'établissement de la preuve de la validité empirique, n'autorise qu'une conclusion définitive d'invalidité ou temporaire de non invalidité.*

Construire des modèles parfaits de l'activité humaine étant impossible, nous ne pouvons pas espérer prouver l'exactitude de nos modèles cognitifs. La seule validation envisageable consiste à comparer un ensemble de protocoles réels (obtenus par l'expérimentation) à un ensemble de protocoles générés (obtenus par simulation). Les différences entre les protocoles réels et les protocoles générés doivent être minimales, voire non observables.

Comme le souligne [Escabarajal, 1991], la validation d'un modèle de simulation de processus peut être considérée sous trois points de vue distincts d'exigence croissante, qu'elle appelle *niveaux*.

- Le premier est celui de l'*identité des sorties*, condition nécessaire mais non suffisante qui postule que la réponse fournie par la machine doit être la même que celle fournie par le sujet.

- La deuxième concerne l'*identité des processus*. Le modèle produit ses réponses de la même manière que le sujet humain : "*la suite des transformations des états de la machine correspond bien à la succession des états du système*".
- Le dernier est celui de l'*invariance des modèles*, pour lequel il s'agit de montrer que les "*éléments fondamentaux du modèle*" ont une portée plus générale dépassant les *contextes d'explication du comportement cognitif humain*.

La validation des modèles cognitifs inclus dans BDIGGY correspond au second niveau, puisque c'est le raisonnement global du sujet qui est modélisé. [Kayser, 1992] remarque d'ailleurs que ce niveau marque la limite entre l'IA qui modélise des résultats, et la modélisation cognitive qui modélise la façon d'y parvenir. Par contre, comme l'explique Kayser :

*On ne peut imposer non plus que tous les états intermédiaires du modèle soient équivalents à des états intermédiaires attestés : modéliser n'est pas dupliquer. Probablement suffit-il à notre critère d'adéquation que certains points intermédiaires jugés particulièrement significatifs pour le sujet humain possèdent un homologue dans le modèle.*

La comparaison entre protocoles réels et protocoles générés doit être faite en utilisant un ensemble d'*observables* mesurant chacun un aspect limité de l'adéquation du modèle au réel. L'idéal est de choisir des approches dont les forces et les faiblesses soient suffisamment complémentaires pour espérer couvrir toutes les facettes du problème. Cependant, en ce qui concerne la validation de BDIGGY proprement dite, seul un test "*à la Turing*" a été effectué.

## 6.2 Génération de protocoles pour la validation

Toute validation nécessite de disposer de données expérimentales de bonne qualité. Dans le cas de BDIGGY, nous avons :

- un ensemble de protocoles réels de référence, obtenus par expérimentation psychologique et représentatifs des comportements observés chez les sujets humains,
- un ensemble de protocoles générés au cours d'une simulation faite par trois agents BDIGGY résolvant simultanément leur problème.

Le premier échantillon a déjà été décrit section 1.1.2 : il s'agit des 5x3 protocoles tirés au hasard parmi le corpus expérimental et écartés préalablement à toute analyse. Cet échantillon est représentatif des données expérimentales et indépendant du processus de modélisation<sup>1</sup>.

Le second échantillon est constitué de protocoles générés à l'aide du système BDIGGY, au cours de simulations de la résolution du problème de l'agence de voyage.

La modélisation des protocoles dépend grandement des paramètres représentant la personnalité du sujet. Pour éviter que les différences entre l'ensemble des protocoles réels et l'ensemble des protocoles générés ne soient dus à ces personnalités, leur distribution statistique dans les protocoles générés doit coïncider avec celle des protocoles humains. Ainsi, une fois le système réglé, 5x3 protocoles humains ont été tirés au hasard parmi l'ensemble des protocoles analysés, afin de servir de référence pour les personnalités utilisées durant la simulation.

<sup>1</sup>Cette méthode permet d'éviter l'effet de *calage* qui revient à utiliser les mêmes protocoles pour construire le modèle et le valider. Le système de simulation "parfait" dans ce cas serait celui qui apprendrait par coeur les protocoles présentés pour ensuite les restituer sans la moindre connaissance cognitive

Comme précisé section 1.2.2, une personnalité est composée de huit traits de caractères (*attentif, économe, bon estimateur, coopératif, opportuniste, optimisateur, patient et précis*) prenant une valeur entre 1 et 3. Le tableau 6.1 recense les personnalités des sujets observées dans les 5x3 protocoles.

BDIGGY a été exécuté 5 fois avec les mêmes personnalités que celles observées dans les 5 trinômes. Cette opération a été exécutée une seconde fois. À l'issue de ce processus, nous disposons donc de deux échantillons de protocoles (5x3 protocoles humains et 2x5x3 protocoles générés) représentatifs respectivement des protocoles obtenus par expérimentation et par simulation. Ces deux échantillons forment le matériau de base à partir duquel notre validation est réalisée.

### 6.3 Un test "*à la Turing*"

Le jeu de l'imitation, souvent appelé test de Turing, a été introduit dans [Turing, 1950]. Ce test célèbre peut être formulé de la façon suivante : supposons qu'une machine M soit programmée pour avoir un comportement *comparable* à celui d'un être humain H pour un certain nombre d'activités manifestement intelligentes. Le test consiste à mesurer cette "comparabilité" en faisant converser de manière aveugle (par le biais d'un terminal par exemple) un autre être humain C soit avec H, soit avec M. Si C n'est pas capable, à l'issue de cette interaction, de déterminer s'il était en relation avec H ou M, on dira que M passe le *test de l'intelligence*.

Un test de Turing n'est évidemment pas envisageable avec des agents BDIGGY, en faisant résoudre le problème de l'agence de voyage par deux sujets humains et un agent BDIGGY et en demandant aux sujets humains, à la fin de l'expérimentation, s'ils étaient en présence d'humains ou d'agents informatiques. Ne serait-ce que parce que BDIGGY n'inclut pas de module permettant de traiter la langue naturelle. Une variante de ce test consiste à demander à des experts d'analyser et de comparer des protocoles réels et des protocoles générés et de voir s'il est possible de séparer aisément les deux classes.

Pour reprendre la classification de [Escabarajal, 1991], ce test "*à la Turing*" se situe entre le niveau d'identité des entrées et celui de l'identité des processus.

Les protocoles des deux échantillons décrits dans la section précédente ont tout d'abord été retraités, pour supprimer tout artefact permettant de les identifier sans étudier les processus de résolution.

- Les en-têtes de tous les protocoles (réels et générés) ont été modifiés. Ils contenaient la date, le nom du sujet/agent et le moyen de transport en charge. Seule cette dernière information a été conservée.
- Les messages des protocoles réels ont été traduits dans la syntaxe des prédicats de type *pMessage* (voir 4.3.2). Aucun langage naturel n'a été conservé.
- Les indications horaires de tous les protocoles ont été supprimées. En effet, une simulation dure entre 2 et 5 minutes alors qu'une résolution par des sujets humains prend entre 1 heure et 1 heure 45.

À partir de ces protocoles modifiés, deux ensembles mêlant protocoles réels et protocoles générés ont ensuite été constitués, avec la répartition suivante :

- *Ensemble 1* : 3 trinômes de protocoles réels tirés au hasard parmi les 5 et 3 trinômes de

	Trinôme 1			Trinôme 2			Trinôme 3		
	aérien	ferroviaire	routier	aérien	ferroviaire	routier	aérien	ferroviaire	routier
attentif	2	2	3	2	2	3	2	3	3
coopératif	3	3	3	2	3	3	2	3	3
économe	2	2	2	2	2	2	2	2	3
bon estimateur	3	1	1	3	2	2	3	2	3
opportuniste	2	3	3	3	3	3	3	1	3
optimisateur	2	2	1	2	2	2	1	3	2
patient	2	2	3	1	3	2	3	2	2
précis	2	3	3	1	2	3	2	2	3

	Trinôme 4			Trinôme 5		
	aérien	ferroviaire	routier	aérien	ferroviaire	routier
attentif	1	1	3	3	1	2
coopératif	2	3	3	2	2	3
économe	2	2	2	3	2	3
bon estimateur	2	1	3	3	2	1
opportuniste	1	1	3	3	3	3
optimisateur	2	2	2	1	1	2
patient	3	3	3	3	3	2
précis	2	1	2	2	2	3

TAB. 6.1 – Liste des personnalités observées

protocoles générés tirés au hasard dans le premier ensemble des 5 trinômes générés.

- *Ensemble 2* : les 2 trinômes de protocoles réels restant et 4 trinômes de protocoles générés tirés au hasard dans le second ensemble des 5 trinômes générés.

Deux ensembles de six trinômes de protocoles sont ainsi obtenus. Ces deux échantillons ont été étudiés par Jean-Philippe Kotowicz (maître de conférence en informatique au LITIS - Rouen), Violaine Philippini (doctorante en chimie au CEA - Orsay), Valérie Lourenço (doctorante en chimie au CEA - Orsay) et Étienne Côme (doctorant en informatique à l'INRETS - Arcueil) dans le but de faire ressortir les différences. Aucun d'entre eux ne connaissaient les protocoles réels avant la phase de validation. Seul Jean-Philippe Kotowicz travaille en modélisation cognitive et est habitué à analyser des corpus d'interaction humaine.

Le tableau 6.2 contient les résultats de la validation. L'ensemble 1 a été analysé par J.-P. Kotowicz et V. Philippini et l'ensemble 2 par J.-P. Kotowicz, V. Lourenço et E. Côme. Chaque trinôme est classé comme composé d'agents BDIGGY ou humains par les "experts", et cette classification apparaît en **bleu** lorsqu'elle est correcte et en **rouge** quand elle est erronée.

		<b>J.-P. Kotowicz</b>	<b>V. Philippini</b>	<b>V. Lourenço</b>	<b>E. Côme</b>
<b>Ensemble 1</b>	Trinôme 1 (BDIGGY)	<b>BDIGGY</b>	<b>Humains</b>	-	-
	Trinôme 2 (Humains)	<b>BDIGGY</b>	<b>BDIGGY</b>	-	-
	Trinôme 3 (BDIGGY)	<b>Humains</b>	<b>BDIGGY</b>	-	-
	Trinôme 4 (Humains)	<b>Humains</b>	<b>Humains</b>	-	-
	Trinôme 5 (Humains)	<b>Humains</b>	<b>BDIGGY</b>	-	-
	Trinôme 6 (BDIGGY)	<b>Humains</b>	<b>Humains</b>	-	-
<b>Ensemble 2</b>	Trinôme 1 (BDIGGY)	<b>Humains</b>	-	<b>Humains</b>	<b>Humains</b>
	Trinôme 2 (Humains)	<b>BDIGGY</b>	-	<b>BDIGGY</b>	<b>Humains</b>
	Trinôme 3 (BDIGGY)	<b>BDIGGY</b>	-	<b>BDIGGY</b>	<b>BDIGGY</b>
	Trinôme 4 (BDIGGY)	<b>BDIGGY</b>	-	<b>BDIGGY</b>	<b>Humains</b>
	Trinôme 5 (BDIGGY)	<b>Humains</b>	-	<b>Humains</b>	<b>BDIGGY</b>
	Trinôme 6 (Humains)	<b>Humains</b>	-	<b>BDIGGY</b>	<b>Humains</b>
		<b>6/12</b>	<b>2/6</b>	<b>2/6</b>	<b>4/6</b>

TAB. 6.2 – Résultats du test "à la Turing"

A la lueur de ces résultats, il apparaît que les "experts" n'ont pas été capable de séparer de façon significative les deux classes de protocoles, puisque leur taux d'erreur est proche de 1/2.

De plus, on notera que certains protocoles sont toujours classés dans la même catégorie, soit de façon correcte (ensemble 1, trinôme 4 et ensemble 2, trinôme 3), soit de façon erronée (ensemble 1, trinômes 2 et 6 et ensemble 2, trinôme 1). Cela laisse supposer qu'il existerait des comportements considérés comme typiques de la machine ou de l'humains par nos "experts". Pour avoir une réelle portée, il faudrait néanmoins effectuer cette classification auprès d'un plus grand nombre d'"experts", et de rechercher le dénominateur commun entre les différents protocoles de chaque catégorie.

D'un point de vue purement pragmatique, il est satisfaisant de constater que certains protocoles générés par des agents BDIGGY sont toujours considérés comme représentatifs de comportements humains.

Même si le test de Turing a été souvent controversé (voir à ce sujet [Searle, 1980]) à cause de son caractère purement comportementaliste, il est nécessaire à la validation de BDIGGY et des

modèles cognitifs. De plus, il permet d'avoir une analyse cognitive réalisée par des personnes extérieures au projet.

## 6.4 Conclusion

L'architecture d'agent BDIGGY a permis de simuler informatiquement la résolution du problème de l'agence de voyage. La validation des modèles de l'interaction humaine et de la planification humaine en contexte coopératif a été faite grâce à un test à *la Turing*.

La classe des protocoles humains et la classe des protocoles générés par simulation n'ayant pas laissé apparaître de différences significatives, les modèles intégrés à BDIGGY, ainsi que l'architecture elle-même, peuvent donc être considérés comme simulant de façon satisfaisante les comportements humains pour le problème donné.

# Conclusion générale

## Bilan

Le travail présenté dans ce document s'appuie sur une méthodologie de modélisation cognitive et s'intègre au sein du projet de l'agence de voyage. Nous cherchons ici à observer, modéliser et simuler les capacités humaines d'interaction et de planification au cours de la résolution d'un problème en contexte coopératif.

Pour cela, nous avons étudié un ensemble de protocoles expérimentaux humains issus d'une expérimentation psychologique réalisée avec des sujets humains. Ce document présente l'analyse et la modélisation cognitive de ces protocoles du point de vue de la planification et des interactions, l'intégration de ces modèles cognitifs au sein d'une architecture d'agent originale et la validation de cette approche grâce à un test "à la Turing".

Du point de vue de la planification, nous reprenons un modèle pré-existant de résolution humaine de problèmes, étendu à la résolution coopérative de problèmes. Le système initial IGGY a aussi été repris et adapté à la résolution coopérative de problèmes pour donner le système IGGY2.

Du point de vue des interactions, les énoncés ont tout d'abord été pris individuellement, pour vérifier leur adéquation avec la théorie des actes de langage. Il a toutefois fallu affiner la liste des performatives et préciser le lien existant entre une force illocutoire et son contenu, de façon à pouvoir représenter le plus fidèlement possible les échanges observés. Cette première analyse a permis de dégager une structure pour les dialogues de demande d'information nécessitant un modèle de la dynamique conversationnelle et plus particulièrement de la temporalité. Nous avons finalement construit un modèle de l'interaction humaine composé d'un système de performatives, comme cela se fait le plus souvent dans les ACL classiques, pour représenter les énoncés, et d'automates temporisés pour gérer l'enchaînement des conversations. Ce modèle est décrit par une logique des prédicats du second ordre qui permet, entre autres choses, de formaliser les contraintes du problème de l'agence de voyage.

Le modèle de l'énoncé présenté est plus proche de la théorie des actes de langage que dans les ACL classiques car il intègre le lien existant entre la force illocutoire et la proposition à laquelle elle s'applique. En effet, toutes les performatives s'appliquent à des objets mentaux, typés en fonction de la classe illocutoire de la performative : les descriptifs s'appliquent à des croyances, les directifs à des désirs du locuteur et les engageants à des désirs de l'interlocuteur.

Les automates temporisés sont utilisés comme formalisme pour introduire la récursivité et la gestion du temps dans la représentation des conversations.

Le lien entre planification et interactions se fait par le biais des concepts BDI. Grâce à eux,

l'architecture BDIGGY englobe nos modèle de la planification humaine et de l'interaction humaine. De plus, BDIGGY améliore l'architecture BDI classique en incluant un module de communication pour générer et interpréter des messages, et permet une construction opportuniste des plans.

Cette architecture a permis de simuler les processus humains de résolution coopérative de problèmes observés dans les protocoles expérimentaux. Les modèles cognitifs et BDIGGY ont été validés en comparant les protocoles expérimentaux et les protocoles générés artificiellement par la simulation, en utilisant les protocoles expérimentaux mis de côté spécialement pour la phase de validation.

## Perspectives

De nombreuses extensions sont possibles aux travaux qui viennent d'être présentés. Nous nous contenterons des principales.

Tout d'abord, le passage d'IGGY2 vers BDIGGY a, certes, permis d'étendre le modèle de planification humaine à la résolution coopérative de problèmes et d'intégrer les interactions mais cela reste insuffisant. Les interactions couvertes sont limitées aux dialogues de recherche d'information par e-mails et à un problème particulier. Il serait intéressant d'explorer d'autres type de problèmes, nécessitant des interactions plus complexes. De plus, les conversations faisant intervenir plus de deux interlocuteurs n'ont pas été abordées, en raison des caractéristiques de l'expérimentation psychologique. Pour étudier ces pistes il faudra bien évidemment mettre en place de nouvelles expérimentations permettant d'améliorer nos modèles.

De plus, il ne faut pas perdre de vue que le problème de la langue naturelle n'a pas été abordé. Nous avons considéré ici qu'il était possible de concevoir un module permettant de traduire un message en énoncé et d'interpréter un énoncé en message. Un tel module existe d'ailleurs partiellement, réalisé par des étudiants<sup>2</sup> durant des projets que nous avons encadrés. Ce module fonctionne en générant des phrases types à partir de *pMessage* et en recherchant des mots clefs dans les énoncés. Cependant, pour être efficace, l'interprétation et la génération de message devraient être traitées en utilisant les informations des automates temporisés. La compréhension d'un énoncé est grandement facilitée quand sa forme est connue. Or, les automates temporisés permettent de représenter les attendus et donc de donner des indications quant à la forme possible des messages. Pour aller plus loin, nous pensons que le décryptage des actes de langages indirects doit pouvoir être effectué grâce à une gestion efficace des attendus des conversation, ce que permettent les automates temporisés.

Par ailleurs, le fonctionnement des automates temporisés peut être amélioré. Notamment en reprenant et en perfectionnant le phénomène de la récursivité pour les dialogues incidents, qui semblait prometteur mais dont l'apport restait limité dans le cadre de cette thèse. Cela nécessitera entre autres choses une formalisation générique de l'utilisation des automates temporisés comme représentation des attendus dans les interactions.

Pour augmenter la flexibilité des automates, une phase d'apprentissage peut être envisagée, selon deux niveaux de complexité :

- en mettant à jour les annotations des arcs et des états dans les automates, selon les types de

---

<sup>2</sup>Alexandre Pillon, Julien Melano, Aurélien Patry, Adeline Lepaul et Guillaume Palomeres.

- messages rencontrés ; il s'agit là d'apprentissage non supervisé, aisé à mettre en oeuvre ;
- en modifiant la structure même des automates par ajout d'arcs, voire en en créant de nouveaux, toujours en fonction des types de messages rencontrés ; cette fois-ci, le processus ne devrait pas pouvoir être automatisé et nécessiterait l'intervention d'un opérateur pour validation (apprentissage supervisé).

Intégrer traitement de la langue naturelle et apprentissage devrait permettre de faire un pas vers nos perspectives majeures sur le long terme : utiliser l'architecture BDIGGY pour construire des SMA hétérogènes, ou au moins des agents conversationnels plus "intelligents" que ceux utilisés actuellement.



# **Annexes**



# Annexe A : Analyse d'un protocole expérimental du point de vue de la planification

Problème à résoudre : ChateauNeuf/Maguelone, 4 personnes, 3300F.

## Protocole expérimental

[04:10:00] Sujet: linda  
[04:10:00] Role: Agent aerien  
[04:10:00] Probleme: Aller  
[04:10:00] Date: Tue May 12 12:10:57 1998  
[04:10:00] 1, Recoit message 1 de Agent routier  
[04:14:06] 2, Affiche message numero 1 de Agent routier  
| De: Agent routier A: Agent aerien  
| quel est le prix du trajet pour trois personnes de Paris àAnnecy pour demain apres 17h30 ?  
[04:15:46] 3, Demande horaires Paris—Annecy (Pach' AIR)  
[04:15:59] 4, Demande horaires Paris—Annecy (Pigeon Vol)  
[04:16:21] 5, Memorise horaire  
[04:16:21] 6, Ajoute horaire dans panneau Mon probleme  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 1 personne  
| 18h45/20h00 — 450 F  
[04:16:33] 7, Supprime horaire  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 1 personne  
| 18h45/20h00 — 450 F  
[04:16:36] 8, Change panneau : Pb. ag. routier  
[04:16:39] 9, Memorise horaire  
[04:16:39] 10, Ajoute horaire dans panneau Pb. ag. routier  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 1 personne  
| 18h45/20h00 — 450 F  
[04:16:41] 11, Change panneau : Mon probleme  
[04:16:42] 12, Change panneau : Pb. ag. routier  
[04:16:45] 13, Envoie horaire Paris—Annecy (avion)  
[04:17:02] 14, Annule.  
[04:17:06] 15, Supprime horaire  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 1 personne  
| 18h45/20h00 — 450 F  
[04:17:12] 16, Memorise horaire  
[04:17:12] 17, Ajoute horaire dans panneau Pb. ag. routier  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 3 personnes  
| 18h45/20h00 — 450 F  
[04:17:15] 18, Envoie horaire Paris—Annecy (avion)  
[04:18:56] 19, Envoie message 1  
| De: Agent aerien A: Agent routier  
| message envoyé  
| =====HORAIRE=====  
| Paris—Annecy (avion Pigeon Vol)  
| Date: Demain — 3 personnes  
| 18h45/20h00 — 450 F  
[04:20:27] 20, Change panneau : Pb. ag. ferroviaire  
[04:20:40] 21, Nouveau message  
[04:23:37] 22, Envoie message 2  
| De: Agent aerien A: Agent ferroviaire  
| j, budget e demande horaire et tarifs pour quatres personnes à destination de toulouse départ de paris budget de 3000 f.  
[04:24:15] 23, Change panneau : Mon probleme  
[04:24:36] 24, Demande horaires Paris—Montpellier (Pigeon Vol)  
[04:24:43] 25, Demande horaires Paris—Montpellier (Pach' AIR)  
[04:26:10] 26, Change panneau : Pb. ag. routier  
[04:26:41] 27, Nouveau message  
[04:27:56] 28, Recoit message 7 de Agent ferroviaire  
[04:28:47] 29, Envoie message 3  
| De: Agent aerien A: Agent routier  
| je demande horaire et tarifs pour qutres personnes: départ aeroport de montpellier vers maguelone. merci  
[04:28:50] 30, Affiche message numero 7 de Agent ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| Désolé ; mais il n'existe pas de ligne Paris—Toulouse  
[04:31:02] 31, Recoit message 8 de Agent routier  
[04:32:13] 32, Affiche message numero 8 de Agent routier  
| De: Agent routier A: Agent aerien  
| voici les horaires bonne chance  
| =====HORAIRE=====  
| Montpellier(aero)—Maguelone ( car)  
| Date: Demain — 4 personnes  
| 13h30/14h00 — 60 F  
[04:32:39] 33, Change panneau : Mon probleme  
[04:32:41] 34, Memorise horaire de message Agent routier : 8  
[04:32:41] 35, Ajoute horaire dans panneau Mon probleme  
| Montpellier(aero)—Maguelone ( car)  
| Date: Demain — 4 personnes  
| 13h30/14h00 — 60 F

[04:32:46] 36, Supprime message Agent routier : 8  
[04:32:58] 37, Supprime message (Agent aerien : 3)  
[04:33:15] 38, Change panneau : Pb. ag. ferroviaire  
[04:33:21] 39, Nouveau message  
[04:35:06] 40, Envoie message 4  
| De: Agent aerien A: Agent ferroviaire  
| demande horaire et tarifs pour quatres  
| personnes de chateau-neuf vers bordeaux.  
| merci  
[04:35:28] 41, Change panneau : Mon probleme  
[04:35:45] 42, Demande horaires Bordeaux-Nice (Pach'  
AIR)  
[04:36:41] 43, Demande horaires Bordeaux-Nice (Pigeon  
Vol)  
[04:36:48] 44, Demande horaires Bordeaux-Nice (Pach'  
AIR)  
[04:37:11] 45, Demande horaires Nice-Montpellier (  
Pach' AIR)  
[04:37:18] 46, Demande horaires Nice-Montpellier (  
Pigeon Vol)  
[04:37:46] 47, Change panneau : Pb. ag. ferroviaire  
[04:37:48] 48, Nouveau message  
[04:39:17] 49, Envoie message 5  
| De: Agent aerien A: Agent ferroviaire  
| demande horaire et tarifs pour quatres  
| personnes depart nice vers montpellier.  
| merci  
[04:40:33] 50, Affiche message numero 1 de Agent  
aerien  
| De: Agent aerien A: Agent routier  
| message envoyé  
| =====HORAIRE=====  
| Paris-Annecy (avion Pigeon Vol)  
| Date: Demain — 3 personnes  
| 18h45/20h00 — 450 F  
[04:40:41] 51, Supprime message (Agent aerien : 1)  
[04:40:55] 52, Recoit message 11 de Agent routier  
[04:40:57] 53, Supprime message (Agent aerien : 2)  
[04:40:59] 54, Affiche message numero 11 de Agent  
routier  
| De: Agent routier A: Agent aerien  
| Peux tu me donner le prix du trajet Paris -  
| Grenoble pour demain à partir de 18h30  
| pour 3 personnes ,merci fabienne  
[04:41:03] 55, Recoit message 11 de Agent ferroviaire  
[04:41:19] 56, Change panneau : Pb. ag. routier  
[04:41:22] 57, Recoit message 12 de Agent ferroviaire  
[04:41:51] 58, Demande horaires Paris-Grenoble (  
Pigeon Vol)  
[04:41:59] 59, Demande horaires Paris-Grenoble (Pach'  
AIR)  
[04:42:21] 60, Memorise horaire  
[04:42:21] 61, Ajoute horaire dans panneau Pb. ag.  
routier  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 18h05/19h05 — 595 F  
[04:42:27] 62, Memorise horaire  
[04:42:27] 63, Ajoute horaire dans panneau Pb. ag.  
routier  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 19h55/20h50 — 595 F  
[04:42:32] 64, Affiche horaire  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 18h05/19h05 — 595 F  
[04:42:33] 65, Envoie horaire Paris-Grenoble (avion)  
[04:44:05] 66, Envoie message 6  
| De: Agent aerien A: Agent routier  
| je souhaite avoir une réponse à mon message  
| de tout à l'heure. merci  
| =====HORAIRE=====  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 18h05/19h05 — 595 F  
[04:44:07] 67, Affiche horaire  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 19h55/20h50 — 595 F  
[04:44:08] 68, Envoie horaire Paris-Grenoble (avion)  
[04:44:39] 69, Envoie message 7  
| De: Agent aerien A: Agent routier  
| réponse obtenue merci.  
| =====HORAIRE=====  
| Paris-Grenoble (avion Pach' AIR)  
| Date: Demain — 3 personnes  
| 19h55/20h50 — 595 F  
[04:44:43] 70, Affiche message numero 11 de Agent  
ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| je te joins un deuxième horaire tout de suite  
| .bye  
| =====HORAIRE=====  
| Nice-Montpellier (train)  
| Date: Demain — 4 personnes  
| 09h40/12h40 — 200 F  
[04:45:29] 71, Affiche message numero 12 de Agent  
ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| voilà  
| =====HORAIRE=====  
| Nice-Montpellier (train)  
| Date: Demain — 4 personnes  
| 14h20/17h20 — 200 F  
[04:46:08] 72, Nouveau message  
[04:46:55] 73, Recoit message 14 de Agent ferroviaire  
[04:47:07] 74, Annule.  
[04:47:16] 75, Affiche message numero 14 de Agent  
ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| désolé mais c'est la seule chose que je peux  
| te proposer pour allaer à Bordeaux, si cela  
| t'intéresse , je te communiquerai des  
| infos sur orléans -Paris  
| =====HORAIRE=====  
| Paris-Bordeaux (train)  
| Date: Demain — 4 personnes  
| 08h01/12h08 — 470 F  
[04:48:05] 76, Supprime message Agent ferroviaire :  
14  
[04:50:25] 77, Nouveau message  
[04:51:57] 78, Envoie message 8  
| De: Agent aerien A: Agent routier  
| demande horaire et tarifs pour quatres  
| personnes chateau-neuf orléan en bus merci.  
[04:52:05] 79, Change panneau : Pb. ag. ferroviaire  
[04:52:11] 80, Nouveau message  
[04:53:56] 81, Recoit message 16 de Agent routier  
[04:54:03] 82, Envoie message 9  
| De: Agent aerien A: Agent ferroviaire  
| demande horaires et tarifs pour quatres  
| personnes:  
| départ orléan — toulouse.  
| départ toulouse— montpellier. merci  
[04:54:10] 83, Affiche message numero 16 de Agent  
routier  
| De: Agent routier A: Agent aerien  
| il y aplusieurs horaires cela coute 24 f peux  
| tu ,me donner l'hoaire exacte ,merci  
[04:54:46] 84, Change panneau : Pb. ag. routier  
[04:54:55] 85, Reponse a Agent routier : 16  
[04:55:46] 86, Envoie message 10  
| De: Agent aerien A: Agent routier  
| Agent routier a ecrit :  
| >il y aplusieurs horaires cela coute 24 f

peux tu ,me donner l'hoaire exacte ,merci  
 | je souhaite avoir les horaires du matin.  
 merci

[04:57:49] 87, Recoit message 17 de Agent ferroviaire  
 [04:58:25] 88, Affiche message numero 17 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | désolé mais il n'y a pas de ligne orléans –  
 toulouse mais je peux te proposer un taret  
 orléans–montpellier puis montpellier –  
 toulouse . En tous les cas voici l'hoaire  
 toulouse –montpellier  
 | =====HORAIRE=====  
 | Toulouse–Montpellier (train)  
 | Date: Demain — 1 personne  
 | 07h40/09h50 – 150 F

[04:58:26] 89, Recoit message 18 de Agent routier  
 [04:59:05] 90, Change panneau : Mon probleme  
 [04:59:07] 91, Memorise horaire de message Agent  
 ferroviaire : 17  
 [04:59:07] 92, Ajoute horaire dans panneau Mon  
 probleme  
 | Toulouse–Montpellier (train)  
 | Date: Demain — 1 personne  
 | 07h40/09h50 – 150 F

[04:59:16] 93, Supprime message Agent ferroviaire :  
 17  
 [04:59:36] 94, Memorise horaire de message \*\*\* Agent  
 routier : 18  
 [04:59:36] 95, Ajoute horaire dans panneau Mon  
 probleme  
 | ChateauNeuf–Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 – 24 F

[04:59:43] 96, Affiche message numero 10 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | Agent routier a ecrit :  
 | >il y a plusieurs horaires cela coute 24 f  
 | peux tu ,me donner l'hoaire exacte ,merci  
 | je souhaite avoir les horaires du matin.  
 merci

[04:59:45] 97, Affiche message numero 18 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | Peux tu me donner une trnche horaire exacte ,  
 | il y a des bus a partir de 6h10 ;merci  
 | =====HORAIRE=====  
 | ChateauNeuf–Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 – 24 F

[04:59:54] 98, Supprime message Agent routier : 18  
 [04:59:59] 99, Supprime message (Agent aerien : 10)  
 [05:00:11] 100, Affiche message numero 16 de Agent  
 routier  
 [05:00:15] 101, Supprime message Agent routier : 16  
 [05:00:17] 102, Affiche message numero 8 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | demande horaire et tarifs pour quatres  
 | personnes chateau–neuf orléan en bus merci.

[05:00:25] 103, Supprime message (Agent aerien : 8)  
 [05:00:26] 104, Affiche message numero 7 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | réponse obtenue merci.  
 | =====HORAIRE=====  
 | Paris–Grenoble (avion Pach' AIR)  
 | Date: Demain — 3 personnes  
 | 19h55/20h50 – 595 F

[05:00:37] 105, Supprime message (Agent aerien : 7)  
 [05:00:39] 106, Affiche message numero 6 de Agent  
 aerien

| De: Agent aerien A: Agent routier  
 | je souhaite avoir une réponse à mon message  
 de tout à l'heure. merci  
 | =====HORAIRE=====  
 | Paris–Grenoble (avion Pach' AIR)  
 | Date: Demain — 3 personnes  
 | 18h05/19h05 – 595 F

[05:00:43] 107, Supprime message (Agent aerien : 6)  
 [05:00:45] 108, Affiche message numero 12 de Agent  
 ferroviaire  
 [05:00:50] 109, Affiche message numero 11 de Agent  
 ferroviaire  
 [05:00:52] 110, Affiche message numero 11 de Agent  
 routier  
 [05:00:58] 111, Supprime message Agent routier : 11  
 [05:00:59] 112, Affiche message numero 5 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | demande horaire et tarifs pour quatres  
 | personnes départ nice vers montpellier.  
 merci

[05:01:04] 113, Affiche message numero 4 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | demande horaire et tarifs pour quatres  
 | personnes de chateau–neuf vers bordeaux.  
 merci

[05:01:07] 114, Affiche message numero 9 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | demande horaires et tarifs pour quatres  
 | personnes :  
 | départ orléan – toulouse.  
 | départ toulouse– montpellier. merci

[05:03:32] 115, Nouveau message  
 [05:04:47] 116, Envoie message 11  
 | De: Agent aerien A: Agent ferroviaire  
 | urgence ou en est mon message: orléan–  
 toulouse. merci

[05:07:14] 117, Recoit message 21 de Agent  
 ferroviaire  
 [05:07:17] 118, Recoit message 22 de Agent routier  
 [05:07:21] 119, Affiche message numero 21 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | Je t'ai déjà répondu qu'il n'y avait pas de  
 | liaison entre orléans et toulouse, mais je  
 | peux te proposer un trajet orléans –  
 | montpellier et montpellier–toulouse .

[05:07:57] 120, Affiche message numero 22 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | peux tu me dire si il existe des vols à moins  
 | de 400 f pour demain apres 18h35 pour paris  
 –Annecy

[05:08:01] 121, Affiche message numero 21 de Agent  
 ferroviaire  
 [05:08:08] 122, Affiche message numero 22 de Agent  
 routier  
 [05:08:22] 123, Affiche message numero 21 de Agent  
 ferroviaire  
 [05:08:23] 124, Reponse a Agent ferroviaire : 21  
 [05:10:57] 125, Envoie message 12  
 | De: Agent aerien A: Agent ferroviaire  
 | Agent ferroviaire a ecrit :  
 | >Je t'ai déjà répondu qu'il n'y avait pas de  
 | liaison entre orléans et toulouse, mais je  
 | peux te proposer un trajet orléans –  
 | montpellier et montpellier–toulouse .  
 | je suis intéresser par le trajet orléan–  
 | montpellier, donne moi les horaires et  
 | tarifs à partir de 10h : 00. merci

[05:11:01] 126, Affiche message numero 22 de Agent

routier  
[05:11:36] 127, Demande horaires Paris—Annecy (Pach' AIR)  
[05:11:43] 128, Demande horaires Paris—Annecy (Pigeon Vol)  
[05:12:13] 129, Reponse a Agent routier : 22  
[05:12:59] 130, Envoie message 13  
| De: Agent aerien A: Agent routier  
| Agent routier a ecrit :  
| >peux tu me dire si il existe des vols à moins de 400 f pour demain apres18h35 pour paris —Annecy  
| il y'a deux vols à plus de 400f.  
[05:13:55] 131, Recoit message 22 de Agent ferroviaire  
[05:14:05] 132, Affiche message numero 22 de Agent ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| désolé , mais il n'existe pas deligne Orléans —montpellier .mais je vais essayer de te trouver un autre trajet ;  
[05:15:29] 133, Recoit message 24 de Agent routier  
[05:17:40] 134, Affiche message numero 24 de Agent routier  
| De: Agent routier A: Agent aerien  
| peux tu me donner le prix du trajet lyon annecy pour demain apres 19h00 ?merci  
[05:18:09] 135, Demande horaires Lyon—Annecy (Pigeon Vol)  
[05:18:14] 136, Demande horaires Lyon—Annecy (Pach' AIR)  
[05:18:20] 137, Demande horaires Lyon—Annecy (Pigeon Vol)  
[05:18:30] 138, Reponse a Agent routier : 24  
[05:18:57] 139, Envoie message 14  
| De: Agent aerien A: Agent routier  
| Agent routier a ecrit :  
| >peux tu me donner le prix du trajet lyon annecy pour demain apres 19h00 ?merci  
| il n'a pas de ligne directe.  
[05:19:12] 140, Supprime message Agent routier : 24  
[05:19:16] 141, Supprime message (Agent aerien : 14)  
[05:19:20] 142, Affiche message numero 13 de Agent aerien  
| De: Agent aerien A: Agent routier  
| Agent routier a ecrit :  
| >peux tu me dire si il existe des vols à moins de 400 f pour demain apres18h35 pour paris —Annecy  
| il y'a deux vols à plus de 400f.  
[05:19:24] 143, Supprime message (Agent aerien : 13)  
[05:19:27] 144, Affiche message numero 22 de Agent routier  
[05:19:31] 145, Supprime message Agent routier : 22  
[05:19:34] 146, Affiche message numero 21 de Agent ferroviaire  
[05:19:37] 147, Affiche message numero 12 de Agent aerien  
| De: Agent aerien A: Agent ferroviaire  
| Agent ferroviaire a ecrit :  
| >Je t'ai déjà répondu qu'il n'y avait pas de liaison entre orléans et toulouse , mais je peux te proposer un trajet orléans — montpellier et montpellier—toulouse .  
| je suis intéresser par le trajet orléan— montpellier, donne moi les horaires et tarifs à partir de 10h : 00. merci  
[05:19:40] 148, Affiche message numero 5 de Agent aerien  
[05:19:45] 149, Supprime message (Agent aerien : 5)  
[05:19:47] 150, Affiche message numero 1 de Agent routier  
[05:19:51] 151, Supprime message Agent routier : 1  
[05:19:53] 152, Affiche message numero 4 de Agent aerien  
[05:19:56] 153, Supprime message (Agent aerien : 4)  
[05:19:57] 154, Affiche message numero 7 de Agent ferroviaire  
[05:20:03] 155, Supprime message Agent ferroviaire : 7  
[05:22:17] 156, Nouveau message  
[05:24:25] 157, Envoie message 15  
| De: Agent aerien A: Agent ferroviaire  
| demande horaires et tarifs pour quatres personnes:  
| —orléans vers Clermont ferrand puis Clermont ferrand vers Montpellier  
| — orléans vers Lyon puis Lyon vers montpellier  
| — orléans vers Grenoble et Grenoble vers montpellier  
[05:27:30] 158, Recoit message 25 de Agent ferroviaire  
[05:27:54] 159, Affiche message numero 25 de Agent ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| Ma pauvre! Il n'y a absolument aucune des lignes que tu as demandées !  
[05:28:12] 160, Supprime message Agent ferroviaire : 25  
[05:28:19] 161, Supprime message (Agent aerien : 15)  
[05:28:23] 162, Supprime message Agent ferroviaire : 22  
[05:28:26] 163, Supprime message (Agent aerien : 12)  
[05:28:28] 164, Nouveau message  
[05:30:43] 165, Envoie message 16  
| De: Agent aerien A: Agent ferroviaire  
| hraire et tarifs départ orléan —paris.  
| départ paris— montpellier. merci  
[05:31:02] 166, Demande horaires Paris—Montpellier ( Pigeon Vol)  
[05:31:09] 167, Demande horaires Paris—Montpellier ( Pach' AIR)  
[05:31:33] 168, Memorise horaire  
[05:31:33] 169, Ajoute horaire dans panneau Mon probleme  
| Paris—Montpellier (avion Pach' AIR)  
| Date: Demain — 4 personnes  
| 11h45/12h40 — 720 F  
[05:31:36] 170, Affiche horaire  
| Toulouse—Montpellier (train)  
| Date: Demain — 1 personne  
| 07h40/09h50 — 150 F  
[05:31:39] 171, Supprime horaire  
| Toulouse—Montpellier (train)  
| Date: Demain — 1 personne  
| 07h40/09h50 — 150 F  
[05:33:34] 172, Recoit message 28 de Agent ferroviaire  
[05:33:39] 173, Affiche message numero 28 de Agent ferroviaire  
| De: Agent ferroviaire A: Agent aerien  
| peux —tu me préciser des horaires ?  
[05:33:48] 174, Affiche message numero 16 de Agent aerien  
| De: Agent aerien A: Agent ferroviaire  
| hraire et tarifs départ orléan —paris.  
| départ paris— montpellier. merci  
[05:34:11] 175, Nouveau message  
[05:36:23] 176, Envoie message 17  
| De: Agent aerien A: Agent ferroviaire  
| précision de l'heure 09h:45 départ orléan vers paris puis à la suite un horaire pour paris— montpellier  
[05:37:37] 177, Recoit message 30 de Agent ferroviaire

[05:37:42] 178, Affiche message numero 30 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | l'autre horaire arrive!  
 | =====HORAIRE=====

[05:38:35] 179, Memorise horaire de message Agent ferroviaire : 30

[05:38:36] 180, Ajoute horaire dans panneau Mon probleme  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F

[05:38:36] 181, Recoit message 31 de Agent ferroviaire

[05:38:41] 182, Affiche message numero 31 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | voilà  
 | =====HORAIRE=====

[05:39:23] 183, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 13h30/14h00 — 60 F

[05:39:35] 184, Memorise horaire de message Agent ferroviaire : 31

[05:39:35] 185, Ajoute horaire dans panneau Mon probleme  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 15h30/19h55 — 590 F

[05:39:43] 186, Affiche horaire  
 | Paris-Montpellier (avion Pach'AIR)  
 | Date: Demain — 4 personnes  
 | 11h45/12h40 — 720 F

[05:39:44] 187, Supprime horaire  
 | Paris-Montpellier (avion Pach'AIR)  
 | Date: Demain — 4 personnes  
 | 11h45/12h40 — 720 F

[05:39:49] 188, Affiche horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 — 24 F

[05:39:52] 189, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F

[05:39:57] 190, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 15h30/19h55 — 590 F

[05:40:00] 191, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 13h30/14h00 — 60 F

[05:40:01] 192, Recoit message 29 de Agent routier

[05:40:06] 193, Affiche message numero 29 de Agent routier  
 | De: Agent routier A: Agent ferroviaire , Agent aerien  
 | j'ai terminé ma mission mes voyageurs sont arrivés à bon port et vous? fabienne

[05:40:25] 194, Change panneau : Pb. ag. routier

[05:40:31] 195, Change panneau : Mon probleme

[05:40:43] 196, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 15h30/19h55 — 590 F

[05:40:47] 197, Nouveau message

[05:42:08] 198, Envoie message 18  
 | De: Agent aerien A: Agent routier  
 | demande horaire podede montpellier vers maguelone en bus pour quatres personnes à partir de 19h:45. merci

[05:43:19] 199, Recoit message 32 de Agent routier

[05:43:25] 200, Affiche message numero 32 de Agent routier  
 | De: Agent routier A: Agent aerien  
 | voici l'horaire  
 | =====HORAIRE=====

[05:43:35] 201, Recoit message 35 de Agent ferroviaire

[05:43:35] 202, Memorise horaire de message Agent routier : 32

[05:43:35] 203, Ajoute horaire dans panneau Mon probleme  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 19h30/20h00 — 60 F

[05:43:40] 204, Affiche message numero 35 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | es-tu toujours en vie agent aérien ?

[05:43:57] 205, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 13h30/14h00 — 60 F

[05:43:59] 206, Supprime horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 13h30/14h00 — 60 F

[05:44:07] 207, Envoie horaire ChateauNeuf-Orleans ( gare) (car)

[05:44:36] 208, Envoie message 19  
 | De: Agent aerien A: Agent routier  
 | reserve moi cette horaire merci  
 | =====HORAIRE=====

[05:44:39] 209, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F

[05:44:40] 210, Envoie horaire Orleans-Paris (train)

[05:45:02] 211, Envoie message 20  
 | De: Agent aerien A: Agent ferroviaire  
 | réservé moi cette horaire  
 | =====HORAIRE=====

[05:45:03] 212, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F

[05:45:03] 212, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 15h30/19h55 — 590 F

[05:45:04] 213, Envoie horaire Paris-Montpellier ( train)

[05:45:11] 214, Recoit message 34 de Agent routier

[05:45:24] 215, Envoie message 21  
 | De: Agent aerien A: Agent ferroviaire  
 | reserver moi cette horaire  
 | =====HORAIRE=====

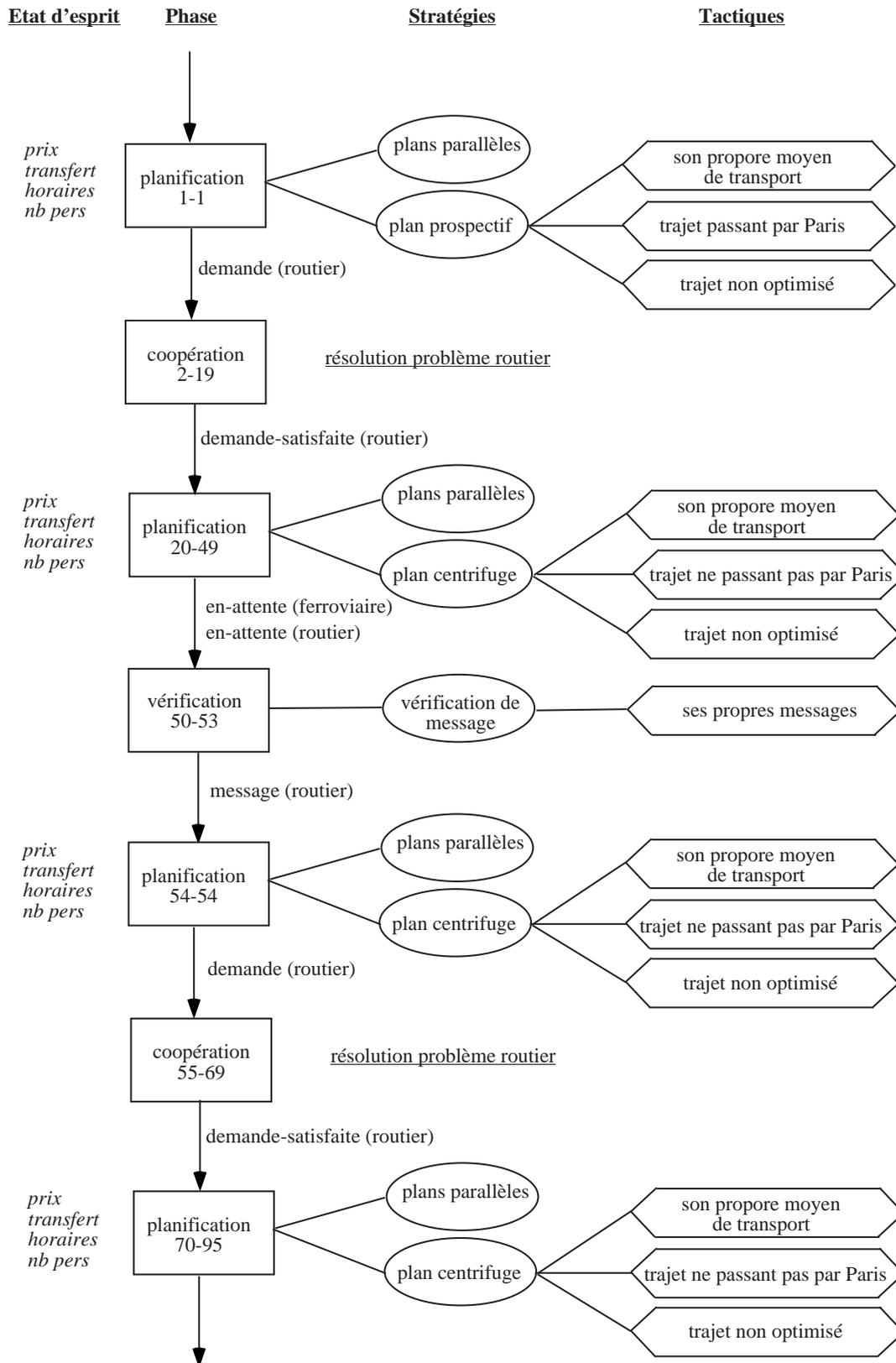
[05:45:29] 216, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 15h30/19h55 — 590 F

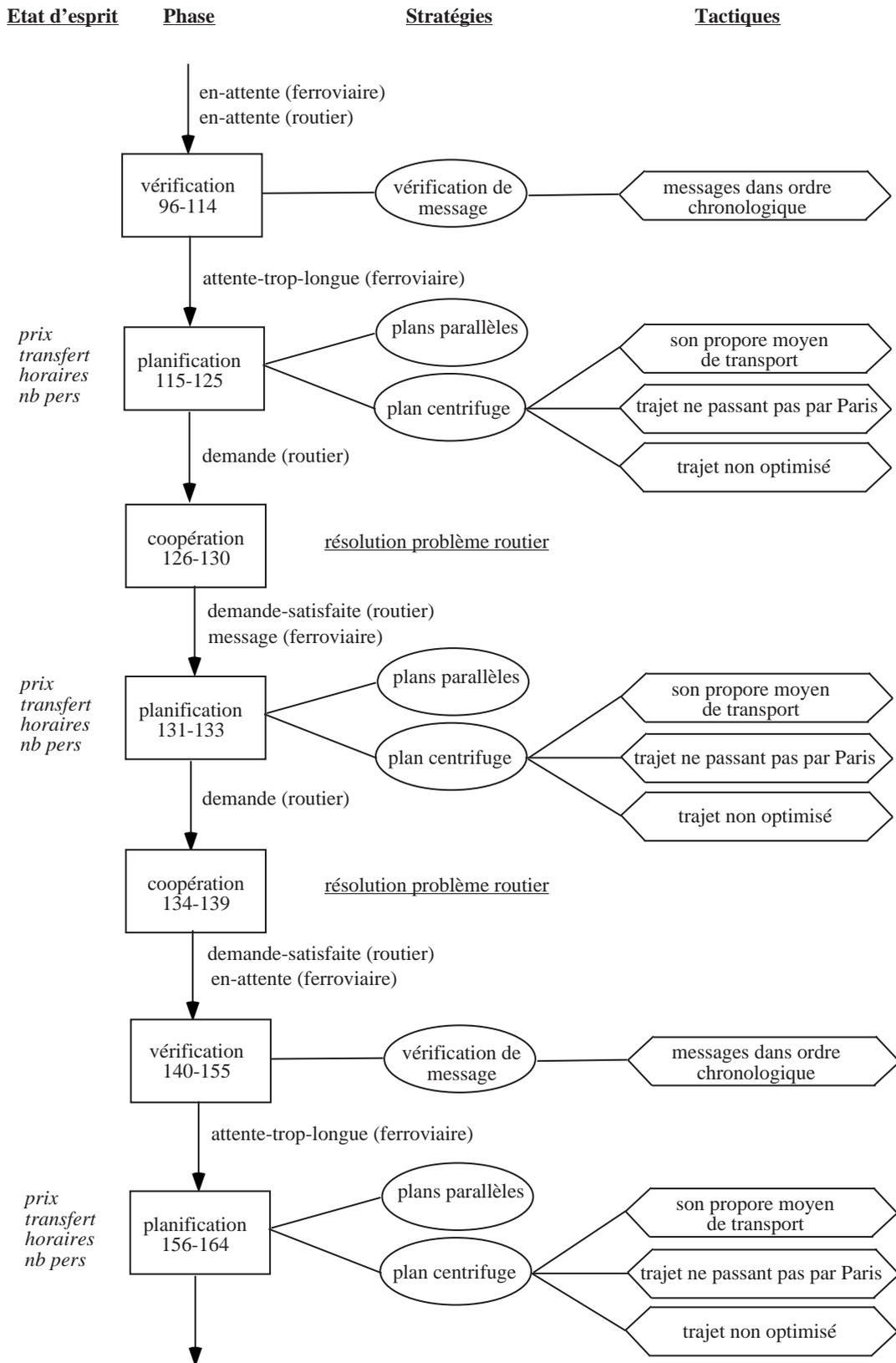
[05:45:29] 216, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)

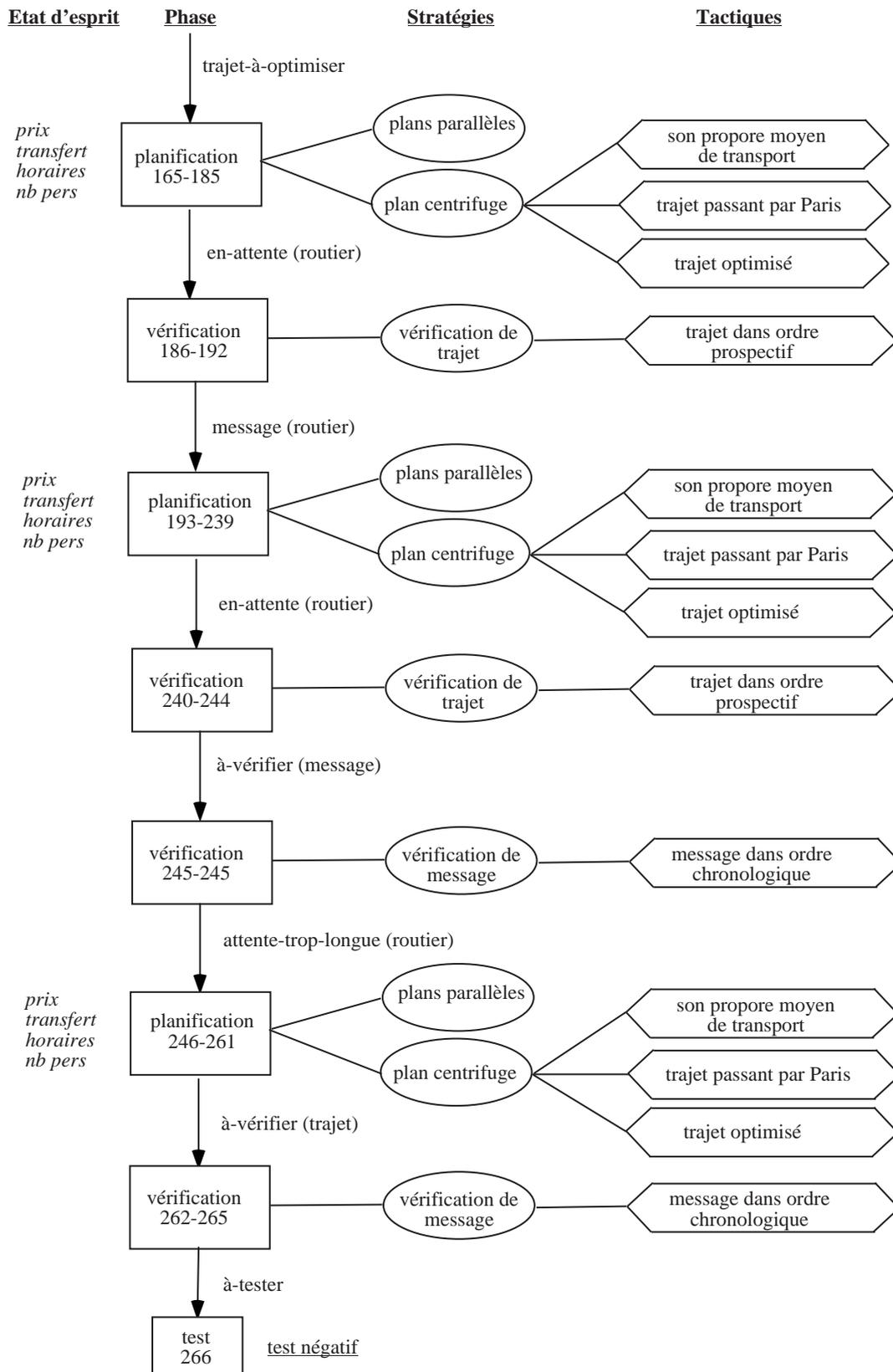
| Date: Demain — 4 personnes  
 | 19h30/20h00 — 60 F  
 [05:45:42] 217, Affiche message numero 34 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | voila c'est gratuit  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 — 24 F  
 [05:45:47] 218, Memorise horaire de message Agent  
 routier : 34  
 [05:45:47] 219, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 — 24 F  
 [05:45:50] 220, Propose horaire [R] ChateauNeuf-  
 Orleans(gare) (car)  
 [05:45:50] 221, Le prix total est : 96  
 [05:45:50] 222, Nouveau trajet en 0  
 | Reservation effectuee :  
 | car : ChateauNeuf - Orleans(gare)  
 | Depart le Demain a 09h10  
 | Arrivee le Demain a 09h45  
 | 4 personne(s). Prix : 24 F.  
 [05:45:55] 223, Supprime message Agent routier : 34  
 [05:46:00] 224, Recoit message 36 de Agent  
 ferroviaire  
 [05:46:03] 225, Affiche message numero 36 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | voilà  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Orleans-Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [05:46:06] 226, Memorise horaire de message Agent  
 ferroviaire : 36  
 [05:46:06] 227, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | Orleans-Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [05:46:10] 228, Propose horaire [R] Orleans-Paris ( train)  
 [05:46:10] 229, Le prix total est : 416  
 [05:46:10] 230, Nouveau trajet en 1  
 | Reservation effectuee :  
 | train : Orleans - Paris  
 | Depart le Demain a 12h25  
 | Arrivee le Demain a 13h30  
 | 4 personne(s). Prix : 80 F.  
 [05:46:13] 231, Nouveau message  
 [05:46:53] 232, Recoit message 37 de Agent  
 ferroviaire  
 [05:47:10] 233, Envoie message 22  
 | De: Agent aerien A: Agent routier  
 | je voudrais l'horaire de montpellier gare  
 vers maguelone.  
 [05:47:16] 234, Affiche message numero 37 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | voilà  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [05:47:24] 235, Memorise horaire de message Agent  
 ferroviaire : 37  
 [05:47:24] 236, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [05:47:25] 237, Propose horaire [R] Paris-Montpellier  
 (train)  
 [05:47:25] 238, Le prix total est : 2776  
 [05:47:25] 239, Nouveau trajet en 2  
 | Reservation effectuee :  
 | train : Paris - Montpellier  
 | Depart le Demain a 15h30  
 | Arrivee le Demain a 19h55  
 | 4 personne(s). Prix : 590 F.  
 [05:47:40] 240, Affiche horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 — 24 F  
 [05:47:42] 241, Supprime horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 09h10/09h45 — 24 F  
 [05:47:47] 242, Supprime horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F  
 [05:47:48] 243, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 19h30/20h00 — 60 F  
 [05:47:53] 244, Affiche horaire  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [05:48:00] 245, Affiche message numero 22 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | je voudrais l'horaire de montpellier gare  
 vers maguelone.  
 [05:48:11] 246, Nouveau message  
 [05:48:32] 247, Recoit message 35 de Agent routier  
 [05:49:12] 248, Envoie message 23  
 | De: Agent aerien A: Agent routier  
 | je précise l'horaire du message précédent à  
 partir de 19h:55  
 [05:49:18] 249, Affiche message numero 35 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | Agent aerien a ecrit :  
 | >je voudrais l'horaire de montpellier gare  
 vers maguelone.  
 | Peut tu me donner l'horaire approximative ,  
 merci  
 [05:50:35] 250, Recoit message 36 de Agent routier  
 [05:50:39] 251, Affiche message numero 36 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | il n'a pas de bus à cette heure -ci ;mais ci  
 -joint le prix du taxi  
 | =====HORAIRE=====  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 01h00 — 450 F  
 [05:51:46] 252, Reponse a Agent routier : 36  
 [05:52:13] 253, Envoie message 24  
 | De: Agent aerien A: Agent routier  
 | Agent routier a ecrit :  
 | >il n'a pas de bus à cette heure -ci ;mais ci  
 -joint le prix du taxi

| réservé moi cette horaire merci  
 [05:52:42] 254, Recoit message 37 de Agent routier  
 [05:52:46] 255, Affiche message numero 37 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | voila  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Montpellier(gare)–Maguelone ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 01h00 – 450 F  
 [05:52:49] 256, Memorise horaire de message Agent  
 routier : 37  
 [05:52:49] 257, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | Montpellier(gare)–Maguelone ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 01h00 – 450 F  
 [05:52:50] 258, Propose horaire [R] Montpellier(gare)  
 –Maguelone ( taxi)  
 [05:53:14] 259, Propose horaire [R] Montpellier(gare)  
 –Maguelone ( taxi)  
 [05:53:36] 260, Le prix total est : 3226  
 [05:53:36] 261, Nouveau trajet en 3  
 | Reservation effectuee :  
 | taxi : Montpellier(gare) – Maguelone  
 | Depart le Demain a 19h55  
 | Arrivee le Demain a 20h55  
 | 1 personne(s). Prix : 450 F.  
 [05:53:42] 262, Affiche trajet [R] car : ChateauNeuf  
 – Orleans(gare)  
 [05:53:46] 263, Affiche trajet [R] train : Orleans –  
 Paris  
 [05:53:48] 264, Affiche trajet [R] train : Paris –  
 Montpellier  
 [05:53:49] 265, Affiche trajet [R] taxi : Montpellier  
 (gare) – Maguelone  
 [05:53:54] 266, Test de la solution : negatif  
 [05:54:03] 267, Affiche trajet [R] train : Paris –  
 Montpellier  
 [05:54:04] 268, Affiche trajet [R] train : Orleans –  
 Paris  
 [05:54:05] 269, Affiche trajet [R] car : ChateauNeuf  
 – Orleans(gare)  
 [05:54:07] 270, Affiche trajet [R] taxi : Montpellier  
 (gare) – Maguelone  
 [05:56:25] 271, Test de la solution : negatif  
 [05:56:39] 272, Affiche trajet [R] car : ChateauNeuf  
 – Orleans(gare)  
 [05:56:40] 273, Affiche trajet [R] train : Orleans –  
 Paris  
 [05:56:41] 274, Affiche trajet [R] train : Paris –  
 Montpellier  
 [05:56:41] 275, Affiche trajet [R] taxi : Montpellier  
 (gare) – Maguelone  
 [05:56:54] 276, Fin de l' experimentation

## Analyse







# Annexe B : Analyse d'un trinôme de protocoles expérimentaux du point de vue des interactions

Chaque message a été traduit dans la syntaxe des *pMessage* : les traductions précèdent les messages en langage naturel et sont repérés par la forme *@@pMessage(...)*.

Les messages sont regroupés en échanges et indexés par une annotation de la forme *##XX\$A\$i\$n\$PERF*. *XX* ∈ {*DI*, *PI*, *EI*, *TE*, *HR*} indique le type d'échange (*DI* pour une demande d'information, *PI* pour une proposition d'information, *EI* pour un envoi spontané d'information, *TE* pour un traitement d'erreur et *HR* quand l'échange est hors résolution). *A* ∈ {*A*, *B*, ..., *Z*, *AA*, ..., *ZZ*, *AAA*, ...} dénombre les échanges. *n* correspond au nombre total de messages de l'échange et *i* au numéro du message dans l'échange. *PERF* est la performative du message.

```

Date : Tue May 12
Agent air : linda
Agent ferroviaire : kermorgant catherine
Agent routier : Fabienne

[04:09:59] 7, Envoie message 1 ##DISA$1$2$QUERY
@@pMessage(road air query pD(road pStage(Paris Ancecy
(> 17:30) * avion 3 ? false)))
  | De: Agent routier A: Agent air
  | quel est le prix du trajet pour trois
    personnes de Paris à Ancecy pour demain
    apres 17h30 ?
[04:11:33] 9, Envoie message 2 ##DISB$1$2$QUERY
@@pMessage(road railway query pD(road pStage(StMartin
Paris (> 17:30) * train 3 ? false)))
  | De: Agent routier A: Agent ferroviaire
  | quel est le prix du trajet saint -martin à
    Paris pour trois personnes demain après 17
    h30 ?
[04:12:14] 27, Envoie message 1 ##DISC$1$6$QUERY
@@pMessage(railway road query pD(railway pStage(
Beaupreau Angers * * * * ? false)))
  | De: Agent ferroviaire A: Agent routier
  | combien coute un trajet Beaupreau-Angers ?
[04:13:48] 33, Envoie message 2 ##DISB$2$2$REPLY
@@pMessage(railway road reply pB(pStage(StMartin
Paris 18:10 18:35 train 3 20 false)))
  | De: Agent ferroviaire A: Agent routier
  | voilà
  | =====HORAIRE=====
  | SaintMartin-Paris (train)
  | Date: Demain — 3 personnes
  | 18h10/18h35 — 20 F
[04:15:34] 35, Envoie message 3 ##DISD$1$4$QUERY
@@pMessage(railway road query pD(railway pStage(Lille
Baisieux * * * * ? false)))
  | De: Agent ferroviaire A: Agent routier
  | Combien coute le transfert de Lille à
    Baisieux.
[04:17:12] 41, Envoie message 3 ##DISD$2$4$REPLY
@@pMessage(road railway reply pB(pStage(Lille(aero)
Baisieux * (+ 00:30) taxi 1 170 false)))
  | De: Agent routier A: Agent ferroviaire
  | voila l'heure bonne chance
  | =====HORAIRE=====
  | Lille(aero)-Baisieux ( taxi)
  | Date: Demain — 1 personne
  | duree : 00h30 — 170 F
[04:18:56] 19, Envoie message 1 ##DISA$2$2$REPLY
@@pMessage(air road reply pB(pStage(Lille(aero)
Baisieux 18:45 20:00 avion 3 450 false)))
  | De: Agent air A: Agent routier
  | message envoyé
  | =====HORAIRE=====
  | Paris-Ancecy (avion Pigeon Vol)
  | Date: Demain — 3 personnes
  | 18h45/20h00 — 450 F
[04:19:08] 41, Envoie message 4 ##DISD$3$4$QUERY
@@pMessage(railway road query pD(railway ?pStage(
Lille(gare) Baisieux * * * * ? false)))
  | De: Agent ferroviaire A: Agent routier
  | merci , mais je voudrai avoir la liaison
    Lille (gare)-Baisieux .
[04:21:04] 66, Envoie message 4 ##DISD$4$4$REPLY
@@pMessage(road railway reply pB(pStage(Lille(gare)
Baisieux * (+ 00:30) taxi 1 180 false)))
  | De: Agent routier A: Agent ferroviaire
  | voilà
  | =====HORAIRE=====
  | Lille(gare)-Baisieux ( taxi)
  | Date: Demain — 1 personne
  | duree : 00h30 — 180 F
[04:22:44] 69, Envoie message 5 ##DISE$1$2$QUERY
@@pMessage(road railway query pD(railway ?pStage(
Paris Ancecy (> 17:30) * train 3 ? false)))
  | De: Agent routier A: Agent ferroviaire

```

| combien coute le trajet Paris –Annecy pour  
 trois personnes demain après 17h30 ? merci  
 [04:23:37] 22, Envoie message 2 ##DISF\$1\$2\$QUERY  
 @@pMessage(air railway query pD(air ?pStage(Paris  
 Toulouse ? \* train 4 ?(< 3000) false)))  
 | De: Agent air A: Agent ferroviaire  
 | j, budget e demande horaire et tarifs pour  
 quatres personnes à destination de toulouse  
 départ de paris budget de 3000 f.  
 [04:24:20] 60, Envoie message 5 ##DISG\$1\$2\$QUERY  
 @@pMessage(railway road query pD(railway ?pStage(\*  
 Baisieux \* 10:10 taxi 1 \* true)))  
 | De: Agent ferroviaire A: Agent routier  
 | peux –tu me réserver un taxi pour mon client  
 à son arrivée à Baisieux à 10h10 pour  
 demain, merci  
 [04:25:32] 66, Envoie message 6 ##DIS\$E\$2\$2\$REPLY  
 @@pMessage(railway road reply pB(pStage(Paris Annecy  
 18:06 22:04 train 3 400 false)))  
 | De: Agent ferroviaire A: Agent routier  
 | voilà  
 | =====HORAIRE=====  
 | Paris–Annecy (train)  
 | Date: Demain — 3 personnes  
 | 18h06/22h04 – 400 F  
 [04:25:42] 83, Envoie message 6 ##DISG\$2\$2\$REPLY  
 @@pMessage(railway road query pD(railway ?pStage(  
 lille (gare) Baisieux \* (+ 00:30) taxi 1 180 true)  
 ))  
 | De: Agent routier A: Agent ferroviaire  
 | et voilà  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Lille (gare)–Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 – 180 F  
 [04:27:55] 73, Envoie message 7 ##DISF\$2\$2\$REPLY  
 @@pMessage(railway air reply pB( pStage(Paris  
 Toulouse \* \* train \* \* false)  
 | De: Agent ferroviaire A: Agent air  
 | Désolé ; mais il n'existe pas de ligne Paris  
 –Toulouse  
 [04:28:47] 29, Envoie message 3 ##DISH\$1\$4\$QUERY  
 @@pMessage(air road query pD(air ?pStage(Montpellier(  
 aero) Maguelone ? ? \* 4 ? true)))  
 | De: Agent air A: Agent routier  
 | je demande horaire et tarifs pour qutres  
 personnes: départ aéroport de montpellier  
 vers maguelone. merci  
 [04:28:47] 91, Envoie message 7 ##DISGb\$1\$3\$QUERY  
 @@pMessage(road railway query pD(road pStage(Annecy  
 LeChatelard 21:00 \* train 3 ? false)))  
 | De: Agent routier A: Agent ferroviaire  
 | Combien coute le trajet Annecy – le  
 chatelard pour trois personnes demain a  
 partir 21h00 ,merci  
 | =====HORAIRE=====  
 | Paris–Annecy (train)  
 | Date: Demain — 3 personnes  
 | 18h06/22h04 – 400 F  
 [04:31:01] 110, Envoie message 8 ##DISH\$2\$4\$REPLY  
 @@pMessage(road air reply pB(pStage(Montpellier(aero)  
 Maguelone 13:30 14:00 car 4 60 false)  
 | De: Agent routier A: Agent air  
 | voici les horaires bonne chance  
 | =====HORAIRE=====  
 | Montpellier(aero)–Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 13h30/14h00 – 60 F  
 [04:31:35] 83, Envoie message 8 ##DISGb\$2\$3\$REPLY + 2  
 e reply \*\*\*\*\*  
 @@pMessage(railway road reply pB( pStage(Annecy  
 LeChatelard \* \* train \* \* false)))

@@pMessage(railway road reply pB(pStage(Annecy  
 Chambéry \* \* train \* \* false)))  
 | De: Agent ferroviaire A: Agent routier  
 | désolé ,mais je n'ai pas de trajet Annecy–Le  
 chatelard, je n'ai qu'une liaison Annecy–  
 chambéry.  
 [04:34:00] 115, Envoie message 9 ##DISGb\$3\$3\$THANK  
 @@pMessage(road railway thank pB(pStage(Annecy  
 Chambéry \* \* train \* \* false)))  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a ecrit :  
 | >désolé ,mais je n'ai pas de trajet Annecy–Le  
 chatelard, je n'ai qu'une liaison Annecy–  
 chambéry.  
 | merci beaucoup  
 [04:34:18] 89, Envoie message 9 ##DISC\$2\$6\$QUERY  
 @@pMessage(railway road query pD(railway ?pStage(  
 Beaupreau Angers 06:00 \* \* \* \* false)))  
 | De: Agent ferroviaire A: Agent routier  
 | peux–tu me trouver une correspondance entre  
 Beaupréau et Angers vers 06h ?  
 [04:35:06] 40, Envoie message 4 ##DISI\$1\$2\$QUERY  
 @@pMessage(air railway query pD(air pStage(  
 ChateauNeuf Bordeaux ? ? train 4 ? false)))  
 | De: Agent air A: Agent ferroviaire  
 | demande horaire et tarifs pour quatres  
 personnes de chateau–neuf vers bordeaux.  
 merci  
 [04:36:26] 122, Envoie message 10 ##TESJ\$1\$3\$REPLY +  
 2e reply \*\*\*\*\*  
 @@pMessage(road railway reply pB( pStage(Beaupreau  
 Angers 06:00 \* car \* \* false)))  
 @@pMessage(road railway reply pB(pStage(Paris Annecy  
 18:06 22:04 train 3 400 false)))  
 | De: Agent routier A: Agent ferroviaire  
 | il n'y a pas de car à cette heure ci mais il  
 ya un taxi je joins le prix et la durée du  
 trajet :  
 | =====HORAIRE=====  
 | Paris–Annecy (train)  
 | Date: Demain — 3 personnes  
 | 18h06/22h04 – 400 F  
 [04:38:46] 98, Envoie message 10 ##TESJ\$2\$3\$NOT–  
 UNDERSTOOD  
 @@pMessage(railway road notunderstood pB(pMessage(  
 road railway reply pB( pStage(Beaupreau Angers  
 06:00 \* car \* \* false))))  
 | De: Agent ferroviaire A: Agent routier  
 | es–tu sur que ce message m'est destiné?  
 [04:39:17] 49, Envoie message 5 ##DISK\$1\$3\$QUERY  
 @@pMessage(air railway query pD(air pStage(Nice  
 Montpellier ? ? train 4 ? false)))  
 | De: Agent air A: Agent ferroviaire  
 | demande horaire et tarifs pour quatres  
 personnes départ nice vers montpellier.  
 merci  
 [04:40:55] 129, Envoie message 11 ##DISL\$1\$1\$QUERY  
 @@pMessage(road air query pD(road pStage(Paris  
 Grenoble 18:30 \* avion 3 ? false)))  
 | De: Agent routier A: Agent air  
 | Peux tu me donner le prix du trajet Paris –  
 Grenoble pour demain à partir de 18h30  
 pour 3 personnes ,merci fabienne  
 [04:41:03] 108, Envoie message 11 ##DISK\$2\$3\$REPLY  
 @@pMessage(railway air reply pB(pStage(Nice  
 Montpellier 09:40 12:40 train 4 200 false)))  
 | De: Agent ferroviaire A: Agent air  
 | je te joins un deuxième horaire tout de suite  
 .bye  
 | =====HORAIRE=====  
 | Nice–Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 09h40/12h40 – 200 F

[04:41:21] 112, Envoie message 12 ##DISK\$3\$3\$REPLY  
 @@pMessage(railway air reply pB(pStage(Nice  
 Montpellier 14:20 17:20 train 4 200 false)))  
 | De: Agent ferroviaire A: Agent air  
 | voilà  
 | =====HORAIRE=====  
 | Nice-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 14h20/17h20 — 200 F

[04:41:41] 132, Envoie message 12 ##NSSJ\$3\$5\$NON-  
 SUPPORTE  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a ecrit :  
 | >es-tu sur que ce message m'est destiné?  
 | quel message ?

[04:42:55] 122, Envoie message 13 ##NSSJ\$4\$5\$NON-  
 SUPPORTE  
 | De: Agent ferroviaire A: Agent routier  
 | le message concernant le trajet Paris Annecy  
 !

[04:44:05] 66, Envoie message 6 ##DISH\$3\$4\$QUERY  
 @@pMessage(air road query pD(air ?pStage(\* \* \* \* \*  
 \* \*)))  
 | De: Agent air A: Agent routier  
 | je souhaite avoir une réponse à mon message  
 | de tout à l'heure. merci  
 | =====HORAIRE=====  
 | Paris-Grenoble (avion Pach'AIR)  
 | Date: Demain — 3 personnes  
 | 18h05/19h05 — 595 F

[04:44:21] 149, Envoie message 13 ##TESJ\$3\$3\$CANCEL  
 @@pMessage(road railway cancel pD(road pMessage(road  
 railway reply pB(pStage(Paris Annecy 18:06 22:04  
 train 3 400 false)))  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a ecrit :  
 | >le message concernant le trajet Paris Annecy  
 !  
 | j'ai supprimé le message voilà

[04:44:39] 69, Envoie message 7 ##DISH\$4\$4\$CANCEL  
 @@pMessage(air road cancel pD(air pMessage(air road  
 query pD(air ?pStage(Montpellier(aero) Maguelone  
 ? \* 4 ? true)))  
 | De: Agent air A: Agent routier  
 | réponse obtenue merci.  
 | =====HORAIRE=====  
 | Paris-Grenoble (avion Pach'AIR)  
 | Date: Demain — 3 personnes  
 | 19h55/20h50 — 595 F

[04:46:30] 161, Envoie message 14 ##DISM\$1\$2\$QUERY  
 @@pMessage(road railway query pD(road ?pStage(  
 StMartin LeChatelard \* \* train \* \* false)))  
 | De: Agent routier A: Agent ferroviaire  
 | Peux tu me dire si il existe une ligne entre  
 | saint-martin et le chatelard? fabienne

[04:46:55] 131, Envoie message 14 ##DISI\$2\$2\$REPLY ##  
 PISb\$1\$1\$PROPOSE  
 @@pMessage(railway air reply pB(pStage(Paris Bordeaux  
 08:01 12:08 train 4 470 false))  
 @@pMessage(railway air propose pD(air pStage(Orleans  
 Paris \* \* train \* \* false))  
 | De: Agent ferroviaire A: Agent air  
 | désolé mais c'est la seule chose que je peux  
 | te proposer pour aller à Bordeaux, si cela  
 | t'intéresse, je te communiquerai des  
 | infos sur orléans-Paris  
 | =====HORAIRE=====  
 | Paris-Bordeaux (train)  
 | Date: Demain — 4 personnes  
 | 08h01/12h08 — 470 F

[04:50:30] 135, Envoie message 15 ##DISM\$2\$2\$REPLY ##  
 DISC\$3\$6\$QUERY  
 @@pMessage(railway road reply pB(pStage(StMartin  
 LeChatelard \* \* train \* \* false)))

[04:51:57] 78, Envoie message 8 ##DISN\$1\$5\$QUERY  
 @@pMessage(air road query pD(air pStage(ChateauNeuf  
 Orleans ? ? car 4 ? false)))  
 | De: Agent air A: Agent routier  
 | demande horaire et tarifs pour quatres  
 | personnes chateau-neuf orléan en bus merci.

[04:52:04] 185, Envoie message 15 ##DISC\$4\$6\$REPLY  
 @@pMessage(road railway query pB(pStage(Beaupreau  
 Angers(gare) 15:00 15:30 car 1 40 false)))  
 | De: Agent routier A: Agent ferroviaire  
 | voici les horaires  
 | =====HORAIRE=====  
 | Beaupreau-Angers(gare) ( car)  
 | Date: Demain — 1 personne  
 | 15h00/15h30 — 40 F

[04:53:46] 149, Envoie message 16 ##DISC\$5\$6\$QUERY  
 @@pMessage(railway road query pD(railway ?pStage(  
 Beaupreau Angers 06:00 \* car \* \* false)))  
 | De: Agent ferroviaire A: Agent routier  
 | merci, peux-tu me dire s'il y a un car  
 | Beaupréau-angers vers 06H

[04:53:56] 193, Envoie message 16 ##DISN\$2\$5\$REFINE  
 @@pMessage(road air refine pD(air pStage(ChateauNeuf  
 Orleans ? \* car 4 24 false)))  
 | De: Agent routier A: Agent air  
 | il y a plusieurs horaires cela coute 24 f peux  
 | tu ,me donner l'hoaire exacte ,merci

[04:54:03] 82, Envoie message 9 ##DISO\$1\$6\$QUERY ##  
 DISP\$1\$2\$QUERY  
 @@pMessage(air railway query pD(air pStage(Orleans  
 Toulouse ? ? train 4 ? false)))  
 @@pMessage(air railway query pD(air pStage(Toulouse  
 Montpellier ? ? train 4 ? false)))  
 | De: Agent air A: Agent ferroviaire  
 | demande horaires et tarifs pour quatres  
 | personnes:  
 | départ orléan — toulouse.  
 | départ toulouse — montpellier. merci

[04:55:01] 201, Envoie message 17 ##DISC\$6\$6\$REPLY +  
 2e reply \*\*\*\*\*  
 @@pMessage(road railway reply pB(pStage(Beaupreau  
 Angers (< 09:00) \* car \* \* false)))  
 @@pMessage(road railway reply pB(pStage(Beaupreau  
 Angers(gare) 15:00 15:30 car 1 40 false)))  
 | De: Agent routier A: Agent ferroviaire  
 | il n'y apas de bus avant 9 h  
 | =====HORAIRE=====  
 | Beaupreau-Angers(gare) ( car)  
 | Date: Demain — 1 personne  
 | 15h00/15h30 — 40 F

[04:55:46] 86, Envoie message 10 ##DISN\$3\$5\$QUERY  
 @@pMessage(air road query pD(air pStage(ChateauNeuf  
 Orleans matin ? car 4 24 false)))  
 | De: Agent air A: Agent routier  
 | Agent routier a ecrit :  
 | >il y a plusieurs horaires cela coute 24 f  
 | peux tu ,me donner l'hoaire exacte ,merci  
 | je souhaite avoir les horaires du matin.  
 | merci

[04:57:49] 158, Envoie message 17 ##DISO\$2\$6\$REPLY ##  
 PISOb\$1\$1\$PROPOSE ##DISP\$2\$2\$REPLY + 2e propose  
 \*\*\*\*\*  
 @@pMessage(railway air reply pB(pStage(Orleans  
 Toulouse \* \* train \* \* false)))

```

@@pMessage(railway air propose pD(air pStage(Orleans
Montpellier * * train * * false)))
@@pMessage(railway air propose pD(air pStage(
Montpellier Toulouse * * train * * false)))
@@pMessage(railway air reply pB(pStage(Toulouse
Montpellier 07:40 09:50 train 1 150 false)))
| De: Agent ferroviaire A: Agent air
| désolé mais il n'y a pas de ligne orléans -
toulouse mais je peux te proposer un trajet
orléans-montpellier puis montpellier -
toulouse . En tous les cas voici l'horaire
toulouse -montpellier
| =====HORAIRE=====
| Toulouse-Montpellier (train)
| Date: Demain — 1 personne
| 07h40/09h50 - 150 F
[04:58:26] 213, Envoie message 18 ##DISN$4$5$REFINE
##DISN$5$5$REPLY + reply *****
@@pMessage(road air refine pD(air pStage(ChateauNeuf
Orleans ? ? car 4 24 false)))
@@pMessage(road air reply pB(pStage(ChateauNeuf
Orleans (> 06:10) * car 4 24 false)))
@@pMessage(road air reply pB(pStage(ChateauNeuf
Orleans 09:10 09:45 car 4 24 false)))
| De: Agent routier A: Agent air
| Peux tu me donner une trnche horaire exacte ,
il y a des bus a partir de 6h10 ;merci
| =====HORAIRE=====
| ChateauNeuf-Orleans(gare) ( car)
| Date: Demain — 4 personnes
| 09h10/09h45 - 24 F
[04:59:10] 161, Envoie message 18 ##DISCb$1$2$QUERY
@@pMessage(railway road query pD(railway pStage(
Beaupreau Angers(gare) * * taxi 1 ? false)))
| De: Agent ferroviaire A: Agent routier
| Très bien , peux -tu alors me donner le prix
d'un taxi pour ce même trajet. Bye
[05:00:47] 224, Envoie message 19 ##DISCb$2$2$REPLY
@@pMessage(road railway reply pB(pStage(Beaupreau
Angers(gare) * (+ 00:30) taxi 1 200 false)))
| De: Agent routier A: Agent ferroviaire
| voila le tarif salut
| =====HORAIRE=====
| Beaupreau-Angers(gare) ( taxi)
| Date: Demain — 1 personne
| duree : 00h30 - 200 F
[05:02:41] 171, Envoie message 19 ##DISQ$1$2$QUERY
@@pMessage(railway road query pD(railway ?pStage(
Beaupreau Angers(gare) * (+ 00:30) taxi 1 200
true)))
| De: Agent ferroviaire A: Agent routier
| Peux-tu me réserver un taxi pour ce trajet .
Merci beaucoup .
[05:02:42] 227, Envoie message 20 ##DISR$1$2$QUERY
@@pMessage(road railway query pD(road ?pStage(Paris
Annecy * (> 18:35) * train * * false)))
| De: Agent routier A: Agent ferroviaire
| Peux tu me dire si il y aun train Paris -
Annecy après 18h35 merci
[05:03:31] 233, Envoie message 21 ##DISQ$2$2$REPLY
@@pMessage(road railway reply pB(pStage(Beaupreau
Angers(gare) * (+ 00:30) taxi 1 200 true)))
| De: Agent routier A: Agent ferroviaire
| y a pas de quoi aurevoir
| =====HORAIRE=====
| Reservation effectuee :
| Beaupreau-Angers(gare) ( taxi)
| Date: Demain — 1 personne
| duree : 00h30 - 200 F
[05:03:49] 179, Envoie message 20 ##DISR$2$2$REPLY
@@pMessage(railway road reply pB(pStage(Paris Annecy
20:06 23:54 train 1 400 false)))
| De: Agent ferroviaire A: Agent routier
| mais bien sur !
| =====HORAIRE=====
| Paris-Annecy (train)
| Date: Demain — 1 personne
| 20h06/23h54 - 400 F
[05:04:47] 116, Envoie message 11 ##DISO$3$6$QUERY
@@pMessage(air railway query pD(air pStage(Orleans
Toulouse ? ? train 4 ? false)))
| De: Agent air A: Agent ferroviaire
| urgence ou en est mon message: orléan-
toulouse. merci
[05:07:14] 189, Envoie message 21 ##DISO$4$6$REPLY
##PI$Ot$1$2$PROPOSE + 2e propose *****
@@pMessage(railway air reply pB( pStage(Orleans
Toulouse * * train * * false)))
@@pMessage(railway air propose pD(air pStage(Orleans
Montpellier * * train * * false)))
@@pMessage(railway air propose pD(air pStage(
Montpellier Toulouse * * train * * false)))
| De: Agent ferroviaire A: Agent air
| Je t'ai déjà répondu qu'il n'y avait pas de
liaison entre orléans et toulouse , mais je
peux te proposer un trajet orléans -
montpellier et montpellier-toulouse .
[05:07:17] 244, Envoie message 22 ##DIS$S$1$2$QUERY
@@pMessage(road air query pD(road ?pStage(Paris
Annecy (> 18:35) ? avion * (< 400) false)))
| De: Agent routier A: Agent air
| peux tu me dire si il existe des vols àmoins
de 400 f pour demain apres18h35 pour paris
-Annecy
[05:10:57] 125, Envoie message 12 ##DISO$5$6$Query
faux est déjà termine ***** ##
PI$Ot$2$2$acceptproposal *****
@@pMessage(air railway accept pD(air pStage(Orleans
Montpellier (> 10:00) ? train * ? false)))
| De: Agent air A: Agent ferroviaire
| Agent ferroviaire a ecrit :
| >Je t'ai déjà répondu qu'il n'y avait pas de
liaison entre orléans et toulouse , mais je
peux te proposer un trajet orléans -
montpellier et montpellier-toulouse .
| je suis intéresser par le trajet orléan-
montpellier , donne moi les horaires et
tarifs à partir de 10h : 00. merci
[05:12:59] 130, Envoie message 13 ##DIS$S$2$2$REPLY
@@pMessage(air road reply pB(pStage(Paris Annecy (>
18:35) * avion * (> 400) false)))
| De: Agent air A: Agent routier
| Agent routier a ecrit :
| >peux tu me dire si il existe des vols àmoins
de 400 f pour demain apres18h35 pour paris
-Annecy
| il y'a deux vols à plus de 400f.
[05:13:18] 256, Envoie message 23 ##DIST$1$2$QUERY
@@pMessage(road railway query pD(road pStage(Paris
Lyon (> 18:35) * train 3 ? false)))
| De: Agent routier A: Agent ferroviaire
| Peux tu me dire le prix du trajet Paris Lyon
en train demain à partir de 18h35 pour 3
personnes ? merci
[05:13:55] 199, Envoie message 22 ##DISO$6$6$REPLY
***** (Ot)
@@pMessage(railway air reply pB( pStage(Orleans
Montpellier * * train * * false)))
| De: Agent ferroviaire A: Agent air
| désolé , mais il n'existe pas deligne
Orléans -montpellier .mais je vais essayer
de te trouver un autre trajet ;
[05:14:55] 205, Envoie message 23 ##DIST$2$2$REPLY
@@pMessage(railway road reply pB(pStage(Paris Lyon
20:31 22:37 train 3 350 false)))
| De: Agent ferroviaire A: Agent routier

```



@@pMessage(railway air reply pB(pStage(Orleans Paris 12:25 13:30 train 1 80 false)))  
 | De: Agent ferroviaire A: Agent air  
 | l'autre horaire arrive!  
 | =====HORAIRE=====

[05:38:35] 278, Envoie message 31 ##DI\$Y\$5\$5\$REPLY  
 @@pMessage(railway air reply pB(pStage(Paris Montpellier 15:30 19:55 train 1 590 false)))  
 | De: Agent ferroviaire A: Agent air  
 | voilà  
 | =====HORAIRE=====

[05:40:01] 347, Envoie message 29 ##EI\$HH\$1\$1\$INFORM ##HR\$HHb\$1\$4\$QUERY @pMessage(road railway inform pB(pProblem (road true)))  
 | De: Agent routier A: Agent ferroviaire  
 | j'ai terminé ma mission mes voyageurs sont arrivés à bon port et vous? fabienne

[05:40:01] 347, Envoie message 29 ##EI\$ZZZ\$1\$1\$INFORM ##HR\$HHb\$1\$4\$QUERY  
 @@pMessage(road air inform pB(pProblem (road true)))  
 | De: Agent routier A: Agent air  
 | j'ai terminé ma mission mes voyageurs sont arrivés à bon port et vous? fabienne

[05:40:07] 281, Envoie message 32 ##HR\$HHb\$2\$5\$QUERY  
 | De: Agent ferroviaire A: Agent routier  
 | as-tu fini l'expérience .

[05:41:01] 284, Envoie message 33 ##EI\$HHt\$1\$1\$INFORM @pMessage(railway road inform pB(pProblem (railway true)))  
 | De: Agent ferroviaire A: Agent routier  
 | mon passager est arrivé à bon port (et entier)!!!

[05:41:13] 352, Envoie message 30 ##HR\$HHb\$3\$4\$REPLY  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a écrit :  
 | >as-tu fini l'expérience .  
 | oui je viens de t'envoyer un message agent ferrovière

[05:41:46] 355, Envoie message 31 ##HR\$HHb\$4\$4\$INFORM \*\*\*\*\*  
 @@pMessage(road railway inform pB(pProblem (road true)))  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a écrit :  
 | >mon passager est arrivé à bon port (et entier)!!!  
 | le mien aussi na

[05:42:08] 198, Envoie message 18 ##DI\$AA\$1\$2\$QUERY  
 @@pMessage(air road query pD(air pStage(Montpellier Maguelone (> 19:45) ? car 4 \* false)))  
 | De: Agent air A: Agent routier  
 | demande horaire pode montpellier vers maguelone en bus pour quatres personnes à partir de 19h:45. merci

[05:42:34] 290, Envoie message 34 ##HR\$BB\$1\$2\$QUERY  
 | De: Agent ferroviaire A: Agent routier  
 | as-tu des nouvelles de notre collègue qui est dans les nuages (airs)?

[05:43:18] 364, Envoie message 32 ##DI\$AA\$2\$2\$REPLY  
 @@pMessage(road air reply pB(pStage(Montpellier(aero) Maguelone 19:30 20:00 car 4 60 false)))  
 | De: Agent routier A: Agent air  
 | voici l'horaire  
 | =====HORAIRE=====

[05:43:35] 292, Envoie message 35 ##HR\$I\$1\$1\$QUERY  
 | De: Agent ferroviaire A: Agent air  
 | es-tu toujours en vie agent aérien ?

[05:44:13] 367, Envoie message 33 ##HR\$BB\$2\$2\$REPLY  
 | De: Agent routier A: Agent ferroviaire  
 | Agent ferroviaire a écrit :  
 | >as-tu des nouvelles de notre collègue qui est dans les nuages (airs)?  
 | oui elle est perdue dans les nuages

[05:44:36] 208, Envoie message 19 ##DI\$CC\$1\$2\$QUERY  
 @@pMessage(air road query pD(air ?pStage(ChateauNeuf Orleans(gare) 09:10 09:45 car 4 24 true)))  
 | De: Agent air A: Agent routier  
 | reserve moi cette horaire merci  
 | =====HORAIRE=====

[05:45:02] 211, Envoie message 20 ##DI\$DD\$1\$2\$QUERY  
 @@pMessage(air railway query pD(air ?pStage(Orleans Paris 12:25 13:30 train 1 80 true)))  
 | De: Agent air A: Agent ferroviaire  
 | réservé moi cette horaire  
 | =====HORAIRE=====

[05:45:11] 374, Envoie message 34 ##DI\$CC\$2\$2\$REPLY  
 @@pMessage(road air reply pB(pStage(ChateauNeuf Orleans(gare) 09:10 09:45 car 4 24 true)))  
 | De: Agent routier A: Agent air  
 | voilà c'est gratuit  
 | =====HORAIRE=====

[05:45:24] 215, Envoie message 21 ##DI\$EE\$1\$2\$QUERY  
 @@pMessage(air railway query pD(air ?pStage(Paris Montpellier 15:30 19:55 train 1 590 true)))  
 | De: Agent air A: Agent ferroviaire  
 | reserver moi cette horaire  
 | =====HORAIRE=====

[05:46:00] 303, Envoie message 36 ##DI\$DD\$2\$2\$REPLY  
 @@pMessage(railway air reply pB(pStage(Orleans Paris 12:25 13:30 train 4 80 true)))  
 | De: Agent ferroviaire A: Agent air  
 | voilà  
 | =====HORAIRE=====

[05:46:52] 310, Envoie message 37 ##DI\$EE\$2\$2\$REPLY  
 @@pMessage(railway air reply pB(pStage(Paris Montpellier 15:30 19:55 train 1 590 true)))  
 | De: Agent ferroviaire A: Agent air  
 | voilà  
 | =====HORAIRE=====

[05:47:10] 233, Envoie message 22 ##DI\$FF\$1\$4\$QUERY  
 @@pMessage(air road query pD(air pStage(Montpellier(gare) Maguelone ? ? car \* \* false)))  
 | De: Agent air A: Agent routier  
 | je voudrais l'horaire de montpellier gare vers maguelone.

[05:48:32] 379, Envoie message 35 ##DI\$FF\$2\$4\$REFINE

@@pMessage(road air refine pD(air pStage(Montpellier(gare) Maguelone ? \* car \* \* false)))  
 | De: Agent routier A: Agent air  
 | Agent air a ecrit :  
 | >je voudrais l'horaire de montpellier gare vers maguelone.  
 | Peux tu me donner l'horaire approximative , merci

[05:49:12] 248, Envoie message 23 ##DI\$FF\$3\$4\$QUERY  
 @@pMessage(air road query pD(air pStage(Montpellier(gare) Maguelone (> 19:55) ? car \* \* false)))  
 | De: Agent air A: Agent routier  
 | je précise l'horaire du message précédent à partir de 19h:55

[05:50:35] 387, Envoie message 36 ##DI\$FF\$4\$4\$REPLY + 2e reply \*\*\*\*\*  
 @@pMessage(road air reply pB( pStage(Montpellier(gare) Maguelone (> 19:55) ? car \* \* false)))  
 @@pMessage(road air reply pB(pStage(Montpellier(gare) Maguelone \* (+ 01:00) taxi 1 450 false)))  
 | De: Agent routier A: Agent air  
 | il n'a pas de bus à cette heure -ci ;mais ci -joint le prix du taxi  
 | =====HORAIRE=====  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 01h00 - 450 F

[05:52:13] 253, Envoie message 24 ##DI\$GG\$1\$2\$QUERY  
 @@pMessage(air road query pD(air ?pStage(Montpellier(gare) Maguelone \* (+ 01:00) taxi 1 450 true)))  
 | De: Agent air A: Agent routier  
 | Agent routier a ecrit :  
 | >il n'a pas de bus à cette heure -ci ;mais ci -joint le prix du taxi  
 | réservé moi cette horaire merci

[05:52:42] 394, Envoie message 37 ##DI\$GG\$2\$2\$REPLY  
 @@pMessage(road air reply pB(pStage(Montpellier(gare) Maguelone \* (+ 01:00) taxi 1 450 true)))  
 | De: Agent routier A: Agent air  
 | voila  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 01h00 - 450 F

# Annexe C : Trinôme de protocoles générés par BDIGGY

**Agent aérien**  
 careful=3 cooperative=3 estimator=3 opportunistic=3  
 optimizer=1 patient=3 precise=3 thrifty=3

Problème à résoudre : ChateauNeuf/Maguelone, 4 personnes, 3300F.

```
[14:08:28] Sujet: John Aipu
[14:08:28] Role: Agent aerien
[14:08:28] Probleme: Aller
[14:08:28] Date: Mon Apr 24 14:08:28 CEST 2006
[14:08:28] 1, Nouveau message
[14:08:28] 2, Envoie message 1
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Orleans Paris ? ? train * * false)))
[14:08:28] 3, Recoit message 1 de Agent ferroviaire
[14:08:29] 4, Affiche message numero 1 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air query pD(railway ?pStage
    (Beaupreau Baisieux * <=(11:00) plane * *
    *)))
[14:08:30] 5, Nouveau message
[14:08:30] 6, Envoie message 2
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway reply pB(!pStage(
    Beaupreau Baisieux * * plane * * *)))
[14:08:30] 7, Recoit message 2 de Agent ferroviaire
[14:08:31] 8, Affiche message numero 2 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Orleans
    Paris 12:25 13:30 train 1 80 false)))
  | =====HORAIRE=====
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 12h25/13h30 — 80 F
[14:08:31] 9, Memorise horaire de message Agent
ferroviaire : 2
[14:08:31] 10, Ajoute horaire dans panneau Mon
probleme
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 12h25/13h30 — 80 F
[14:08:32] 11, Reponse a Agent ferroviaire : 2
[14:08:32] 12, Envoie message 3
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Paris Montpellier morning ? train * ? false
    )))
[14:08:34] 13, Nouveau message
[14:08:34] 14, Envoie message 4
  | De: Agent aerien A: Agent routier
  | pMessage(air road query pD(air ?pStage(
    ChateauNeuf Orleans morning ? OR(coach taxi
    ) * * false)))
[14:08:35] 15, Nouveau message
[14:08:35] 16, Envoie message 5
  | De: Agent aerien A: Agent routier
  | pMessage(air road query pD(air ?pStage(
    Montpellier Maguelone * afternoon coach * *
    false)))
[14:08:36] 17, Reponse a Agent ferroviaire : 2
[14:08:36] 18, Envoie message 6
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Orleans Paris morning ? train * ? false)))
[14:08:37] 19, Reponse a Agent ferroviaire : 2
[14:08:37] 20, Envoie message 7
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Paris Marseille morning ? train * ? false))
  )
[14:08:37] 21, Recoit message 3 de Agent routier
[14:08:38] 22, Affiche message numero 3 de Agent
routier
  | De: Agent routier A: Agent aerien
  | pMessage(road air reply pB(pStage(ChateauNeuf
    Orleans(station) 10:10 10:45 coach 1 24
    false)))
  | =====HORAIRE=====
  | ChateauNeuf-Orleans(gare) ( car)
  | Date: Demain — 1 personne
  | 10h10/10h45 — 24 F
[14:08:38] 23, Memorise horaire de message Agent
routier : 3
[14:08:38] 24, Ajoute horaire dans panneau Mon
probleme
  | ChateauNeuf-Orleans(gare) ( car)
  | Date: Demain — 1 personne
  | 10h10/10h45 — 24 F
[14:08:40] 25, Recoit message 5 de Agent ferroviaire
[14:08:40] 26, Recoit message 4 de Agent routier
[14:08:41] 27, Affiche message numero 5 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Paris
    Montpellier 10:05 14:54 train 1 590 false))
  )
  | =====HORAIRE=====
  | Paris-Montpellier (train)
  | Date: Demain — 1 personne
```

| 10h05/14h54 - 590 F  
 [14:08:41] 28, Memorise horaire de message Agent  
 ferroviaire : 5  
 [14:08:41] 29, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Montpellier (train)  
 | Date: Demain - 1 personne  
 | 10h05/14h54 - 590 F  
 [14:08:42] 30, Affiche message numero 4 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(Montpellier  
 (airport) Maguelone 12:30 13:00 coach 1 60  
 false)))  
 | =====HORAIRE=====  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain - 1 personne  
 | 12h30/13h00 - 60 F  
 [14:08:43] 31, Memorise horaire de message Agent  
 routier : 4  
 [14:08:43] 32, Ajoute horaire dans panneau Mon  
 probleme  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain - 1 personne  
 | 12h30/13h00 - 60 F  
 [14:08:43] 33, Recoit message 6 de Agent ferroviaire  
 [14:08:44] 34, Affiche message numero 6 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Orleans  
 Paris 08:25 09:30 train 1 80 false)))  
 | =====HORAIRE=====  
 | Orleans-Paris (train)  
 | Date: Demain - 1 personne  
 | 08h25/09h30 - 80 F  
 [14:08:45] 35, Recoit message 7 de Agent ferroviaire  
 [14:08:46] 36, Affiche horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 1 personne  
 | 10h10/10h45 - 24 F  
 [14:08:47] 37, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain - 1 personne  
 | 12h25/13h30 - 80 F  
 [14:08:48] 38, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain - 1 personne  
 | 10h05/14h54 - 590 F  
 [14:08:51] 39, Reponse a Agent ferroviaire : 6  
 [14:08:51] 40, Envoie message 8  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air pStage(  
 Paris Montpellier >=(13:30) ? train 4 ?  
 false)))  
 [14:08:51] 41, Recoit message 10 de Agent ferroviaire  
 [14:08:52] 42, Affiche message numero 7 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Paris  
 Marseille 10:30 14:39 train 1 590 false)))  
 | =====HORAIRE=====  
 | Paris-Marseille (train)  
 | Date: Demain - 1 personne  
 | 10h30/14h39 - 590 F  
 [14:08:53] 43, Affiche message numero 10 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air query pD(railway ?pStage  
 (Nantes Baisieux \* <=(11:00) plane \* \*  
 false)))  
 [14:08:54] 44, Recoit message 11 de Agent ferroviaire  
 [14:08:55] 45, Nouveau message  
 [14:08:55] 46, Envoie message 9  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway reply pB(!pStage(Nantes  
 Baisieux \* \* plane \* \* \*)))  
 [14:08:56] 47, Affiche message numero 11 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Paris  
 Montpellier 15:30 19:55 train 4 590 false))  
 )  
 | =====HORAIRE=====  
 | Paris-Montpellier (train)  
 | Date: Demain - 4 personnes  
 | 15h30/19h55 - 590 F  
 [14:08:56] 48, Memorise horaire de message Agent  
 ferroviaire : 11  
 [14:08:56] 49, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Montpellier (train)  
 | Date: Demain - 4 personnes  
 | 15h30/19h55 - 590 F  
 [14:08:58] 50, Affiche horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 1 personne  
 | 10h10/10h45 - 24 F  
 [14:08:59] 51, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain - 1 personne  
 | 12h25/13h30 - 80 F  
 [14:09:00] 52, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain - 1 personne  
 | 10h05/14h54 - 590 F  
 [14:09:00] 53, Supprime horaire  
 | Paris-Montpellier (train)  
 | Date: Demain - 1 personne  
 | 10h05/14h54 - 590 F  
 [14:09:00] 54, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain - 4 personnes  
 | 15h30/19h55 - 590 F  
 [14:09:01] 55, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain - 1 personne  
 | 12h30/13h00 - 60 F  
 [14:09:03] 56, Nouveau message  
 [14:09:03] 57, Envoie message 10  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air pStage(  
 Montpellier(airport) Maguelone >=(19:55) ?  
 coach 4 \* false)))  
 [14:09:05] 58, Recoit message 16 de Agent ferroviaire  
 [14:09:05] 59, Affiche message numero 16 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air query pD(railway ?pStage  
 (Nantes Lille morning \* plane \* \* false)))  
 [14:09:06] 60, Demande horaires Nantes-Lille (Pach'  
 AIR)  
 [14:09:06] 61, Demande horaires Nantes-Lille (Pigeon  
 Vol)  
 [14:09:06] 62, Memorise horaire  
 [14:09:06] 63, Ajoute horaire dans panneau Mon  
 probleme  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain - 1 personne  
 | 07h45/09h05 - 802 F  
 [14:09:06] 64, Envoie horaire Nantes-Lille (avion)  
 [14:09:06] 65, Envoie message 11  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway reply pB(pStage(Nantes  
 Lille 07:45 09:05 plane(PigeonVol) 1 802  
 false)))  
 | =====HORAIRE=====

```

    | Nantes-Lille (avion Pigeon Vol)
    | Date: Demain — 1 personne
    | 07h45/09h05 — 802 F
[14:09:07] 66, Recoit message 11 de Agent routier
[14:09:07] 67, Recoit message 12 de Agent routier
[14:09:09] 68, Affiche message numero 11 de Agent
routier
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(!pStage(
        Montpellier(airport) Maguelone >=(19:55) *
        coach 4 * false)))
[14:09:10] 69, Reponse a Agent routier : 11
[14:09:10] 70, Envoie message 12
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(
        Montpellier(airport) Maguelone * * taxi 4 *
        false)))
[14:09:11] 71, Affiche message numero 12 de Agent
routier
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(Montpellier
        (airport) Maguelone 19:30 20:00 coach 4 60
        false)))
    | =====HORAIRE=====
    | Montpellier(aero)-Maguelone ( car)
    | Date: Demain — 4 personnes
    | 19h30/20h00 — 60 F
[14:09:12] 72, Recoit message 14 de Agent routier
[14:09:13] 73, Affiche message numero 14 de Agent
routier
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(Montpellier
        (airport) Maguelone * +(00:30) taxi 4 300
        false)))
    | =====HORAIRE=====
    | Montpellier(aero)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 00h30 — 300 F
[14:09:13] 74, Memorise horaire de message Agent
routier : 14
[14:09:13] 75, Ajoute horaire dans panneau Mon
probleme
    | Montpellier(aero)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 00h30 — 300 F
[14:09:15] 76, Affiche horaire
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:09:16] 77, Affiche horaire
    | Orleans-Paris (train)
    | Date: Demain — 1 personne
    | 12h25/13h30 — 80 F
[14:09:17] 78, Affiche horaire
    | Paris-Montpellier (train)
    | Date: Demain — 4 personnes
    | 15h30/19h55 — 590 F
[14:09:18] 79, Affiche horaire
    | Montpellier(aero)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 00h30 — 300 F
[14:09:20] 80, Reponse a Agent routier : 14
[14:09:20] 81, Envoie message 13
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(
        Montpellier(station) Maguelone * * taxi * *
        *)))
[14:09:23] 82, Affiche horaire
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:09:23] 83, Recoit message 16 de Agent routier
[14:09:24] 84, Affiche horaire
    | Orleans-Paris (train)
    | Date: Demain — 1 personne
    | 12h25/13h30 — 80 F
[14:09:25] 85, Affiche message numero 16 de Agent
routier
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(Montpellier
        (station) Maguelone * +(01:00) taxi 4 450
        false)))
    | =====HORAIRE=====
    | Montpellier(gare)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 01h00 — 450 F
[14:09:25] 86, Memorise horaire de message Agent
routier : 16
[14:09:25] 87, Ajoute horaire dans panneau Mon
probleme
    | Montpellier(gare)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 01h00 — 450 F
[14:09:27] 88, Affiche horaire
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:09:29] 89, Reponse a Agent routier : 16
[14:09:29] 90, Envoie message 14
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(
        ChateauNeuf Orleans(station) 10:10 10:45
        coach 4 24 false)))
[14:09:31] 91, Supprime horaire
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:09:31] 92, Affiche horaire
    | Orleans-Paris (train)
    | Date: Demain — 1 personne
    | 12h25/13h30 — 80 F
[14:09:31] 93, Recoit message 17 de Agent routier
[14:09:33] 94, Nouveau message
[14:09:33] 95, Envoie message 15
    | De: Agent aerien A: Agent ferroviaire
    | pMessage(air railway query pD(air ?pStage(
        Orleans Paris 12:25 13:30 train 4 80 false)
        ))
[14:09:34] 96, Affiche message numero 17 de Agent
routier
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(ChateauNeuf
        Orleans(station) 10:10 10:45 coach 4 24
        false)))
    | =====HORAIRE=====
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 4 personnes
    | 10h10/10h45 — 24 F
[14:09:34] 97, Memorise horaire de message Agent
routier : 17
[14:09:34] 98, Ajoute horaire dans panneau Mon
probleme
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 4 personnes
    | 10h10/10h45 — 24 F
[14:09:36] 99, Recoit message 19 de Agent ferroviaire
[14:09:37] 100, Affiche horaire
    | Orleans-Paris (train)
    | Date: Demain — 1 personne
    | 12h25/13h30 — 80 F
[14:09:37] 101, Supprime horaire
    | Orleans-Paris (train)
    | Date: Demain — 1 personne
    | 12h25/13h30 — 80 F
[14:09:37] 102, Affiche horaire
    | Paris-Montpellier (train)

```

| Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:09:38] 103, Affiche horaire  
 | Montpellier(gare)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 01h00 — 450 F  
 [14:09:39] 104, Affiche message numero 19 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Orleans  
 Paris 12:25 13:30 train 4 80 false)))  
 | =====HORAIRE=====  
 | Orleans—Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:09:39] 105, Memorise horaire de message Agent  
 ferroviaire : 19  
 [14:09:39] 106, Ajoute horaire dans panneau Mon  
 probleme  
 | Orleans—Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:09:42] 107, Affiche horaire  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:43] 108, Affiche horaire  
 | Orleans—Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:09:44] 109, Affiche horaire  
 | Paris—Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:09:45] 110, Affiche horaire  
 | Montpellier(gare)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 01h00 — 450 F  
 [14:09:46] 111, Affiche horaire  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:46] 112, Propose horaire ChateauNeuf—Orleans(  
 gare) (car)  
 [14:09:46] 113, Le prix total est : 96  
 [14:09:46] 114, Nouveau trajet en 0  
 | car : ChateauNeuf — Orleans(gare)  
 | Depart le Demain a 10h10  
 | Arrivee le Demain a 10h45  
 | 4 personne(s). Prix : 24 F.  
 [14:09:46] 115, Affiche horaire  
 | Orleans—Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:09:46] 116, Propose horaire Orleans—Paris (train)  
 [14:09:46] 117, Le prix total est : 416  
 [14:09:46] 118, Nouveau trajet en 1  
 | train : Orleans — Paris  
 | Depart le Demain a 12h25  
 | Arrivee le Demain a 13h30  
 | 4 personne(s). Prix : 80 F.  
 [14:09:46] 119, Affiche horaire  
 | Paris—Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:09:46] 120, Propose horaire Paris—Montpellier ( train)  
 [14:09:46] 121, Le prix total est : 2776  
 [14:09:46] 122, Nouveau trajet en 2  
 | train : Paris — Montpellier  
 | Depart le Demain a 15h30  
 | Arrivee le Demain a 19h55  
 | 4 personne(s). Prix : 590 F.  
 [14:09:46] 123, Affiche horaire  
 | Montpellier(gare)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 01h00 — 450 F  
 [14:09:46] 124, Le prix total est : 3226  
 [14:09:46] 125, Nouveau trajet en 3  
 | taxi : Montpellier(gare) — Maguelone  
 | Depart le Demain a 19h55  
 | Arrivee le Demain a 20h55  
 | 4 personne(s). Prix : 450 F.  
 [14:09:46] 126, Propose horaire Montpellier(gare)—  
 Maguelone (taxi)  
 [14:09:47] 127, Test de la solution : negatif  
 [14:09:48] 128, Affiche horaire  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:50] 129, Nouveau message  
 [14:09:50] 130, Envoie message 16  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(  
 ChateauNeuf Orleans(station) 10:10 10:45  
 coach 4 24 true)))  
 [14:09:52] 131, Affiche message numero 16 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(  
 ChateauNeuf Orleans(station) 10:10 10:45  
 coach 4 24 true)))  
 [14:09:52] 132, Supprime message (Agent aerien : 16)  
 [14:09:53] 133, Affiche message numero 15 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air ?pStage(  
 Orleans Paris 12:25 13:30 train 4 80 false)  
 ))  
 [14:09:53] 134, Supprime message (Agent aerien : 15)  
 [14:09:54] 135, Recoit message 21 de Agent routier  
 [14:09:54] 136, Affiche message numero 17 de Agent  
 routier  
 [14:09:56] 137, Affiche message numero 21 de Agent  
 routier  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(ChateauNeuf  
 Orleans(station) 10:10 10:45 coach 4 24  
 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:56] 138, Memorise horaire de message Agent  
 routier : 21  
 [14:09:56] 139, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:58] 140, Affiche horaire  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:58] 141, Supprime horaire  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:58] 142, Affiche horaire  
 | Reservation effectuee :  
 | ChateauNeuf—Orleans(gare) ( car)  
 | Date: Demain — 4 personnes  
 | 10h10/10h45 — 24 F  
 [14:09:59] 143, Recoit message 24 de Agent

```

ferroviaire
[14:09:59] 144, Affiche horaire
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:01] 145, Affiche message numero 24 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air query pD(railway ?pStage
(Nantes Lille 07:45 09:05 plane(PigeonVol)
  | 802 true)))
[14:10:02] 146, Affiche horaire
  | Nantes-Lille (avion Pigeon Vol)
  | Date: Demain — 1 personne
  | 07h45/09h05 — 802 F
[14:10:02] 147, Reserve horaire [R] Nantes-Lille (
avion)
[14:10:02] 148, Envoie horaire [R] Nantes-Lille (
avion)
[14:10:02] 149, Envoie message 17
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway reply pB(pStage(Nantes
Lille 07:45 09:05 plane(PigeonVol) 1 802
true)))
  | =====HORAIRE=====
  | Reservation effectuee :
  | Nantes-Lille (avion Pigeon Vol)
  | Date: Demain — 1 personne
  | 07h45/09h05 — 802 F
[14:10:03] 150, Reponse a Agent ferroviaire : 24
[14:10:03] 151, Envoie message 18
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air ?pStage(
Orleans Paris 12:25 13:30 train 4 80 true))
)
[14:10:06] 152, Affiche horaire
  | Reservation effectuee :
  | ChateauNeuf-Orleans(gare) ( car)
  | Date: Demain — 4 personnes
  | 10h10/10h45 — 24 F
[14:10:08] 153, Recoit message 25 de Agent
ferroviaire
[14:10:09] 154, Affiche message numero 25 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Orleans
Paris 12:25 13:30 train 4 80 true)))
  | =====HORAIRE=====
  | Reservation effectuee :
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:09] 155, Memorise horaire de message Agent
ferroviaire : 25
[14:10:09] 156, Ajoute horaire dans panneau Mon
probleme
  | Reservation effectuee :
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:11] 157, Affiche horaire
  | Reservation effectuee :
  | ChateauNeuf-Orleans(gare) ( car)
  | Date: Demain — 4 personnes
  | 10h10/10h45 — 24 F
[14:10:12] 158, Affiche horaire
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:12] 159, Supprime horaire
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:12] 160, Affiche horaire
  | Reservation effectuee :
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:13] 161, Affiche horaire
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:15] 162, Reponse a Agent ferroviaire : 25
[14:10:15] 163, Envoie message 19
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air ?pStage(
Paris Montpellier 15:30 19:55 train 4 590
true)))
[14:10:18] 164, Affiche message numero 1 de Agent
aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
Orleans Paris ? ? train * * false)))
[14:10:18] 165, Supprime message (Agent aerien : 1)
[14:10:18] 166, Recoit message 27 de Agent
ferroviaire
[14:10:19] 167, Affiche message numero 2 de Agent
aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway reply pB(!pStage(
Beaupreau Baisieux * * plane * * *)))
[14:10:19] 168, Supprime message (Agent aerien : 2)
[14:10:20] 169, Affiche message numero 27 de Agent
ferroviaire
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Paris
Montpellier 15:30 19:55 train 4 590 true)))
  | =====HORAIRE=====
  | Reservation effectuee :
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:20] 170, Memorise horaire de message Agent
ferroviaire : 27
[14:10:20] 171, Ajoute horaire dans panneau Mon
probleme
  | Reservation effectuee :
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:22] 172, Affiche horaire
  | Reservation effectuee :
  | ChateauNeuf-Orleans(gare) ( car)
  | Date: Demain — 4 personnes
  | 10h10/10h45 — 24 F
[14:10:23] 173, Affiche horaire
  | Reservation effectuee :
  | Orleans-Paris (train)
  | Date: Demain — 4 personnes
  | 12h25/13h30 — 80 F
[14:10:24] 174, Affiche horaire
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:24] 175, Supprime horaire
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:24] 176, Affiche horaire
  | Reservation effectuee :
  | Paris-Montpellier (train)
  | Date: Demain — 4 personnes
  | 15h30/19h55 — 590 F
[14:10:25] 177, Affiche horaire
  | Montpellier(gare)-Maguelone ( taxi)
  | Date: Demain — 4 personnes

```

| duree : 01h00 - 450 F  
 [14:10:27] 178, Nouveau message  
 [14:10:27] 179, Envoie message 20  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(Montpellier(station) Maguelone \* +(01:00) taxi 4 450 true)))  
 [14:10:28] 180, Recoit message 25 de Agent routier  
 [14:10:29] 181, Affiche message numero 25 de Agent routier  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air inform pB(pProblem(road true)))  
 [14:10:31] 182, Recoit message 27 de Agent routier  
 [14:10:32] 183, Affiche message numero 27 de Agent routier  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(Montpellier(station) Maguelone \* +(01:00) taxi 4 450 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:32] 184, Memorise horaire de message Agent routier : 27  
 [14:10:32] 185, Ajoute horaire dans panneau Mon probleme  
 | Reservation effectuee :  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:34] 186, Affiche horaire  
 | Reservation effectuee :  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 4 personnes  
 | 10h10/10h45 - 24 F  
 [14:10:35] 187, Affiche horaire  
 | Reservation effectuee :  
 | Orleans-Paris (train)  
 | Date: Demain - 4 personnes  
 | 12h25/13h30 - 80 F  
 [14:10:36] 188, Affiche horaire  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain - 4 personnes  
 | 15h30/19h55 - 590 F  
 [14:10:37] 189, Affiche horaire  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:37] 190, Supprime horaire  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:37] 191, Affiche horaire  
 | Reservation effectuee :  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:38] 192, Affiche horaire  
 | Reservation effectuee :  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 4 personnes  
 | 10h10/10h45 - 24 F  
 [14:10:38] 193, Affiche trajet car : ChateauNeuf - Orleans(gare)  
 [14:10:38] 194, Supprime trajet car : ChateauNeuf - Orleans(gare)  
 [14:10:38] 195, Le prix total est : 3130  
 [14:10:38] 196, Propose horaire [R] ChateauNeuf-Orleans(gare) ( car)  
 [14:10:38] 197, Le prix total est : 3226  
 [14:10:38] 198, Nouveau trajet en 1  
 | Reservation effectuee :  
 | car : ChateauNeuf - Orleans(gare)  
 | Depart le Demain a 10h10  
 | Arrivee le Demain a 10h45  
 | 4 personne(s). Prix : 24 F.  
 [14:10:38] 199, Deplace trajet bas: [R] car : ChateauNeuf - Orleans(gare)  
 [14:10:38] 200, Affiche horaire  
 | Reservation effectuee :  
 | Orleans-Paris (train)  
 | Date: Demain - 4 personnes  
 | 12h25/13h30 - 80 F  
 [14:10:38] 201, Affiche trajet train : Orleans - Paris  
 [14:10:38] 202, Supprime trajet train : Orleans - Paris  
 [14:10:38] 203, Le prix total est : 2906  
 [14:10:38] 204, Propose horaire [R] Orleans-Paris (train)  
 [14:10:38] 205, Le prix total est : 3226  
 [14:10:38] 206, Nouveau trajet en 2  
 | Reservation effectuee :  
 | train : Orleans - Paris  
 | Depart le Demain a 12h25  
 | Arrivee le Demain a 13h30  
 | 4 personne(s). Prix : 80 F.  
 [14:10:38] 207, Deplace trajet bas: [R] train : Orleans - Paris  
 [14:10:38] 208, Affiche horaire  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain - 4 personnes  
 | 15h30/19h55 - 590 F  
 [14:10:38] 209, Affiche trajet train : Paris - Montpellier  
 [14:10:38] 210, Supprime trajet train : Paris - Montpellier  
 [14:10:38] 211, Le prix total est : 866  
 [14:10:38] 212, Propose horaire [R] Paris-Montpellier (train)  
 [14:10:38] 213, Le prix total est : 3226  
 [14:10:38] 214, Nouveau trajet en 3  
 | Reservation effectuee :  
 | train : Paris - Montpellier  
 | Depart le Demain a 15h30  
 | Arrivee le Demain a 19h55  
 | 4 personne(s). Prix : 590 F.  
 [14:10:38] 215, Deplace trajet bas: [R] train : Paris - Montpellier  
 [14:10:38] 216, Affiche horaire  
 | Reservation effectuee :  
 | Montpellier(gare)-Maguelone ( taxi)  
 | Date: Demain - 4 personnes  
 | duree : 01h00 - 450 F  
 [14:10:38] 217, Affiche trajet taxi : Montpellier(gare) - Maguelone  
 [14:10:38] 218, Supprime trajet taxi : Montpellier(gare) - Maguelone  
 [14:10:38] 219, Le prix total est : 2776  
 [14:10:38] 220, Le prix total est : 3226  
 [14:10:38] 221, Nouveau trajet en 3  
 | Reservation effectuee :  
 | taxi : Montpellier(gare) - Maguelone  
 | Depart le Demain a 19h55  
 | Arrivee le Demain a 20h55  
 | 4 personne(s). Prix : 450 F.  
 [14:10:38] 222, Propose horaire [R] Montpellier(gare) - Maguelone (taxi)  
 [14:10:39] 223, Test de la solution : positif  
 [14:10:39] 224, Fin de l' experimentation

**Agent ferroviaire**  
 careful=3 cooperative=3 estimator=1 opportunistic=2  
 optimizer=3 patient=3 precise=3 thrifty=1

Problème à résoudre : Beaupreau/Baisieux, 1 personne, arrivée avant 11h00, 1310F.

```
[14:08:28] Sujet: Samantha Lo
[14:08:28] Role: Agent ferroviaire
[14:08:28] Probleme: Aller
[14:08:28] Date: Mon Apr 24 14:08:28 CEST 2006
[14:08:28] 1, Nouveau message
[14:08:28] 2, Recoit message 1 de Agent aerien
[14:08:28] 3, Envoie message 1
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air query pD(railway ?pStage
    (Beaupreau Baisieux * <=(11:00) plane * *
    *)))
[14:08:29] 4, Affiche message numero 1 de Agent
  aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Orleans Paris ? ? train * * false)))
[14:08:30] 5, Demande horaires Orleans-Paris ()
[14:08:30] 6, Memorise horaire
[14:08:30] 7, Ajoute horaire dans panneau Mon
  probleme
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 12h25/13h30 - 80 F
[14:08:30] 8, Envoie horaire Orleans-Paris (train)
[14:08:30] 9, Recoit message 1 de Agent routier
[14:08:30] 10, Envoie message 2
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Orleans
    Paris 12:25 13:30 train 1 80 false)))
  | =====HORAIRE=====
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 12h25/13h30 - 80 F
[14:08:30] 11, Recoit message 2 de Agent aerien
[14:08:31] 12, Affiche message numero 1 de Agent
  routier
  | De: Agent routier A: Agent ferroviaire
  | pMessage(road railway query pD(road pStage(
    SaintMartin Paris >=(17:30) * train * ?
    false)))
[14:08:32] 13, Demande horaires SaintMartin-Paris ()
[14:08:32] 14, Memorise horaire
[14:08:32] 15, Ajoute horaire dans panneau Mon
  probleme
  | SaintMartin-Paris (train)
  | Date: Demain — 1 personne
  | 18h10/18h35 - 20 F
[14:08:32] 16, Envoie horaire SaintMartin-Paris (
  train)
[14:08:32] 17, Recoit message 2 de Agent routier
[14:08:32] 18, Envoie message 3
  | De: Agent ferroviaire A: Agent routier
  | pMessage(railway road reply pB(pStage(
    SaintMartin Paris 18:10 18:35 train 1 20
    false)))
  | =====HORAIRE=====
  | SaintMartin-Paris (train)
  | Date: Demain — 1 personne
  | 18h10/18h35 - 20 F
[14:08:32] 19, Recoit message 3 de Agent aerien
[14:08:34] 20, Affiche message numero 2 de Agent
  routier
  | De: Agent routier A: Agent ferroviaire
  | pMessage(road railway query pD(road pStage(
    Paris Lyon * * train * ? false)))
[14:08:35] 21, Demande horaires Paris-Lyon ()
[14:08:35] 22, Nouveau message
[14:08:35] 23, Envoie message 4
  | De: Agent ferroviaire A: Agent routier
  | pMessage(railway road reply pB(pStage(Paris
    Lyon * * train 1 350 false)))
[14:08:36] 24, Affiche message numero 2 de Agent
  aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway reply pB(!pStage(
    Beaupreau Baisieux * * plane * * *)))
[14:08:36] 25, Recoit message 6 de Agent aerien
[14:08:37] 26, Memorise horaire
[14:08:37] 27, Ajoute horaire dans panneau Mon
  probleme
  | Paris-Lyon (train)
  | Date: Demain — 1 personne
  | 05h32/07h32 - 350 F
[14:08:37] 28, Recoit message 7 de Agent aerien
[14:08:38] 29, Affiche message numero 3 de Agent
  aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Paris Montpellier morning ? train * ? false
    )))
[14:08:39] 30, Demande horaires Paris-Montpellier ()
[14:08:39] 31, Memorise horaire
[14:08:39] 32, Ajoute horaire dans panneau Mon
  probleme
  | Paris-Montpellier (train)
  | Date: Demain — 1 personne
  | 10h05/14h54 - 590 F
[14:08:40] 33, Envoie horaire Paris-Montpellier (
  train)
[14:08:40] 34, Envoie message 5
  | De: Agent ferroviaire A: Agent aerien
  | pMessage(railway air reply pB(pStage(Paris
    Montpellier 10:05 14:54 train 1 590 false))
    )
  | =====HORAIRE=====
  | Paris-Montpellier (train)
  | Date: Demain — 1 personne
  | 10h05/14h54 - 590 F
[14:08:41] 35, Affiche message numero 6 de Agent
  aerien
  | De: Agent aerien A: Agent ferroviaire
  | pMessage(air railway query pD(air pStage(
    Orleans Paris morning ? train * ? false)))
[14:08:42] 36, Affiche horaire
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 12h25/13h30 - 80 F
[14:08:42] 37, Demande horaires Orleans-Paris ()
[14:08:42] 38, Memorise horaire
[14:08:42] 39, Ajoute horaire dans panneau Mon
  probleme
  | Orleans-Paris (train)
  | Date: Demain — 1 personne
  | 08h25/09h30 - 80 F
[14:08:43] 40, Envoie horaire Orleans-Paris (train)
[14:08:43] 41, Recoit message 5 de Agent routier
[14:08:43] 42, Envoie message 6
  | De: Agent ferroviaire A: Agent aerien
```

| pMessage(railway air reply pB(pStage(Orleans  
 Paris 08:25 09:30 train 1 80 false)))  
 | =====HORAIRE=====  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 08h25/09h30 — 80 F  
 [14:08:44] 43, Affiche message numero 7 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air pStage(  
 Paris Marseille morning ? train \* ? false))  
 )  
 [14:08:45] 44, Demande horaires Paris-Marseille ()  
 [14:08:45] 45, Memorise horaire  
 [14:08:45] 46, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Marseille (train)  
 | Date: Demain — 1 personne  
 | 10h30/14h39 — 590 F  
 [14:08:45] 47, Envoie horaire Paris-Marseille (train)  
 [14:08:45] 48, Recoit message 6 de Agent routier  
 [14:08:45] 49, Envoie message 7  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Paris  
 Marseille 10:30 14:39 train 1 590 false)))  
 | =====HORAIRE=====  
 | Paris-Marseille (train)  
 | Date: Demain — 1 personne  
 | 10h30/14h39 — 590 F  
 [14:08:46] 50, Affiche message numero 5 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Lyon >=(18:35) \* train \* ? false)))  
 [14:08:47] 51, Affiche horaire  
 | Paris-Lyon (train)  
 | Date: Demain — 1 personne  
 | 05h32/07h32 — 350 F  
 [14:08:47] 52, Demande horaires Paris-Lyon ()  
 [14:08:47] 53, Memorise horaire  
 [14:08:47] 54, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Lyon (train)  
 | Date: Demain — 1 personne  
 | 20h31/22h37 — 350 F  
 [14:08:47] 55, Envoie horaire Paris-Lyon (train)  
 [14:08:47] 56, Envoie message 8  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Lyon 20:31 22:37 train 1 350 false)))  
 | =====HORAIRE=====  
 | Paris-Lyon (train)  
 | Date: Demain — 1 personne  
 | 20h31/22h37 — 350 F  
 [14:08:48] 57, Affiche message numero 6 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(  
 Lyon Chambery afternoon \* train \* \* false))  
 )  
 [14:08:50] 58, Demande horaires Lyon-Chambery ()  
 [14:08:50] 59, Nouveau message  
 [14:08:50] 60, Envoie message 9  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(!pStage(Lyon  
 Chambery \* \* train \* \* \*)))  
 [14:08:51] 61, Nouveau message  
 [14:08:51] 62, Envoie message 10  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air query pD(railway ?pStage  
 (Nantes Baisieux \* <=(11:00) plane \* \*  
 false)))  
 [14:08:51] 63, Recoit message 8 de Agent aerien  
 [14:08:52] 64, Affiche message numero 8 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air pStage(  
 Paris Montpellier >=(13:30) ? train 4 ?  
 false)))  
 [14:08:52] 65, Recoit message 7 de Agent routier  
 [14:08:53] 66, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 10h05/14h54 — 590 F  
 [14:08:53] 67, Demande horaires Paris-Montpellier ()  
 [14:08:53] 68, Memorise horaire  
 [14:08:53] 69, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:08:53] 70, Envoie horaire Paris-Montpellier (  
 train)  
 [14:08:53] 71, Envoie message 11  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Paris  
 Montpellier 15:30 19:55 train 4 590 false))  
 )  
 | =====HORAIRE=====  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:08:55] 72, Affiche message numero 7 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Chambery \* \* train \* ? false)))  
 [14:08:55] 73, Recoit message 9 de Agent aerien  
 [14:08:56] 74, Demande horaires Paris-Chambery ()  
 [14:08:56] 75, Nouveau message  
 [14:08:56] 76, Envoie message 12  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambery \* \* train 1 370 false)))  
 [14:08:57] 77, Affiche message numero 9 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway reply pB(!pStage(Nantes  
 Baisieux \* \* plane \* \* \*)))  
 [14:08:58] 78, Recoit message 8 de Agent routier  
 [14:08:58] 79, Nouveau message  
 [14:08:58] 80, Envoie message 13  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Beaupreau Nantes early \* OR(coach  
 taxi) \* \* false)))  
 [14:08:59] 81, Affiche message numero 8 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Chambery \* evening train \* ? false)))  
 [14:09:00] 82, Demande horaires Paris-Chambery ()  
 [14:09:00] 83, Memorise horaire  
 [14:09:00] 84, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Chambery (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:00] 85, Envoie horaire Paris-Chambery (train)  
 [14:09:00] 86, Envoie message 14  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambery 18:06 21:13 train 4 370 false)))  
 | =====HORAIRE=====  
 | Paris-Chambery (train)  
 | Date: Demain — 4 personnes

```

    | 18h06/21h13 - 370 F
[14:09:01] 87, Recoit message 9 de Agent routier
[14:09:01] 88, Affiche message numero 9 de Agent
routier
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway refine pD(road pD(
        railway ?pStage(Beaupreau ??(Nantes(station
        ) Nantes(airport)) early * OR(coach taxi) *
        * false)))
[14:09:02] 89, Reponse a Agent routier : 9
[14:09:02] 90, Envoie message 15
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road query pD(railway ?
        pStage(Beaupreau Nantes(airport) early * OR
        (coach taxi) * * false)))
[14:09:03] 91, Demande horaires Nantes-Baisieux ()
[14:09:05] 92, Nouveau message
[14:09:05] 93, Envoie message 16
    | De: Agent ferroviaire A: Agent aerien
    | pMessage(railway air query pD(railway ?pStage
        (Nantes Lille morning * plane * * false)))
[14:09:05] 94, Recoit message 10 de Agent routier
[14:09:06] 95, Affiche message numero 10 de Agent
routier
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway reply pB(pStage(
        Beaupreau Nantes(airport) * +(00:35) taxi 1
        250 false)))
    | =====HORAIRE=====
    | Beaupreau-Nantes(aero) ( taxi)
    | Date: Demain - 1 personne
    | duree : 00h35 - 250 F
[14:09:06] 96, Memorise horaire de message Agent
routier : 10
[14:09:06] 97, Ajoute horaire dans panneau Mon
probleme
    | Beaupreau-Nantes(aero) ( taxi)
    | Date: Demain - 1 personne
    | duree : 00h35 - 250 F
[14:09:06] 98, Recoit message 11 de Agent aerien
[14:09:07] 99, Affiche message numero 11 de Agent
aerien
    | De: Agent aerien A: Agent ferroviaire
    | pMessage(air railway reply pB(pStage(Nantes
        Lille 07:45 09:05 plane(PigeonVol) 1 802
        false)))
    | =====HORAIRE=====
    | Nantes-Lille (avion Pigeon Vol)
    | Date: Demain - 1 personne
    | 07h45/09h05 - 802 F
[14:09:07] 100, Memorise horaire de message Agent
aerien : 11
[14:09:07] 101, Ajoute horaire dans panneau Mon
probleme
    | Nantes-Lille (avion Pigeon Vol)
    | Date: Demain - 1 personne
    | 07h45/09h05 - 802 F
[14:09:08] 102, Nouveau message
[14:09:08] 103, Envoie message 17
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road query pD(railway ?
        pStage(Lille Baisieux * * taxi * * false)))
[14:09:10] 104, Recoit message 13 de Agent routier
[14:09:10] 105, Affiche message numero 13 de Agent
routier
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway reply pB(pStage(Lille(
        station) Baisieux * +(00:30) taxi 4 180
        false)))
    | =====HORAIRE=====
    | Lille(gare)-Baisieux ( taxi)
    | Date: Demain - 4 personnes
    | duree : 00h30 - 180 F
[14:09:10] 106, Memorise horaire de message Agent
routier : 13
[14:09:10] 107, Ajoute horaire dans panneau Mon
probleme
    | Lille(gare)-Baisieux ( taxi)
    | Date: Demain - 4 personnes
    | duree : 00h30 - 180 F
[14:09:13] 108, Affiche horaire
    | Nantes-Lille (avion Pigeon Vol)
    | Date: Demain - 1 personne
    | 07h45/09h05 - 802 F
[14:09:14] 109, Affiche horaire
    | Beaupreau-Nantes(aero) ( taxi)
    | Date: Demain - 1 personne
    | duree : 00h35 - 250 F
[14:09:15] 110, Le prix total est : 250
[14:09:15] 111, Nouveau trajet en 0
    | taxi : Beaupreau - Nantes(aero)
    | Depart le Demain a 07h10
    | Arrivee le Demain a 07h45
    | 1 personne(s). Prix : 250 F.
[14:09:15] 112, Propose horaire Beaupreau-Nantes(aero
) (taxi)
[14:09:15] 113, Affiche horaire
    | Nantes-Lille (avion Pigeon Vol)
    | Date: Demain - 1 personne
    | 07h45/09h05 - 802 F
[14:09:15] 114, Propose horaire Nantes-Lille (avion)
[14:09:15] 115, Le prix total est : 1052
[14:09:15] 116, Nouveau trajet en 1
    | avion : Nantes - Lille
    | Depart le Demain a 07h45
    | Arrivee le Demain a 09h05
    | 1 personne(s). Prix : 802 F.
[14:09:15] 117, Affiche horaire
    | Lille(gare)-Baisieux ( taxi)
    | Date: Demain - 4 personnes
    | duree : 00h30 - 180 F
[14:09:15] 118, Le prix total est : 1232
[14:09:15] 119, Nouveau trajet en 2
    | taxi : Lille(gare) - Baisieux
    | Depart le Demain a 09h05
    | Arrivee le Demain a 09h35
    | 4 personne(s). Prix : 180 F.
[14:09:15] 120, Propose horaire Lille(gare)-Baisieux
(taxi)
[14:09:17] 121, Test de la solution : negatif
[14:09:17] 122, Recoit message 15 de Agent routier
[14:09:18] 123, Affiche message numero 15 de Agent
routier
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway query pD(road pStage(
        Paris Chambery >=(18:35) ? train * * *))
[14:09:19] 124, Affiche horaire
    | Paris-Chambery (train)
    | Date: Demain - 4 personnes
    | 18h06/21h13 - 370 F
[14:09:19] 125, Demande horaires Paris-Chambery ()
[14:09:19] 126, Memorise horaire
[14:09:19] 127, Ajoute horaire dans panneau Mon
probleme
    | Paris-Chambery (train)
    | Date: Demain - 4 personnes
    | 20h06/23h00 - 370 F
[14:09:19] 128, Envoie horaire Paris-Chambery (train)
[14:09:19] 129, Envoie message 18
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road reply pB(pStage(Paris
        Chambery 20:06 23:00 train 4 370 false)))
    | =====HORAIRE=====
    | Paris-Chambery (train)
    | Date: Demain - 4 personnes
    | 20h06/23h00 - 370 F

```

[14:09:21] 130, Affiche horaire  
 | Lille(gare)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 180 F

[14:09:22] 131, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:23] 132, Affiche horaire  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F

[14:09:24] 133, Affiche trajet taxi : Beaupreau — Nantes(aero)

[14:09:24] 134, Supprime trajet taxi : Beaupreau — Nantes(aero)

[14:09:24] 135, Le prix total est : 982

[14:09:24] 136, Le prix total est : 1232

[14:09:24] 137, Nouveau trajet en 1  
 | taxi : Beaupreau — Nantes(aero)  
 | Depart le Demain a 07h10  
 | Arrivee le Demain a 07h45  
 | 1 personne(s). Prix : 250 F.

[14:09:24] 138, Propose horaire Beaupreau-Nantes(aero) ( taxi)

[14:09:24] 139, Deplace trajet bas: taxi : Beaupreau — Nantes(aero)

[14:09:24] 140, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:24] 141, Affiche trajet avion : Nantes — Lille

[14:09:24] 142, Supprime trajet avion : Nantes — Lille

[14:09:24] 143, Le prix total est : 430

[14:09:24] 144, Propose horaire Nantes-Lille (avion)

[14:09:24] 145, Le prix total est : 1232

[14:09:24] 146, Nouveau trajet en 2  
 | avion : Nantes — Lille  
 | Depart le Demain a 07h45  
 | Arrivee le Demain a 09h05  
 | 1 personne(s). Prix : 802 F.

[14:09:24] 147, Deplace trajet bas: avion : Nantes — Lille

[14:09:24] 148, Affiche horaire  
 | Lille(gare)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 180 F

[14:09:24] 149, Affiche trajet taxi : Lille(gare) — Baisieux

[14:09:24] 150, Supprime trajet taxi : Lille(gare) — Baisieux

[14:09:24] 151, Le prix total est : 1052

[14:09:24] 152, Le prix total est : 1232

[14:09:24] 153, Nouveau trajet en 2  
 | taxi : Lille(gare) — Baisieux  
 | Depart le Demain a 09h05  
 | Arrivee le Demain a 09h35  
 | 4 personne(s). Prix : 180 F.

[14:09:24] 154, Propose horaire Lille(gare)-Baisieux ( taxi)

[14:09:25] 155, Test de la solution : negatif

[14:09:27] 156, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:28] 157, Affiche horaire  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F

[14:09:29] 158, Affiche trajet taxi : Beaupreau — Nantes(aero)

[14:09:30] 159, Supprime trajet taxi : Beaupreau — Nantes(aero)

[14:09:30] 160, Le prix total est : 982

[14:09:30] 161, Le prix total est : 1232

[14:09:30] 162, Nouveau trajet en 1  
 | taxi : Beaupreau — Nantes(aero)  
 | Depart le Demain a 07h10  
 | Arrivee le Demain a 07h45  
 | 1 personne(s). Prix : 250 F.

[14:09:30] 163, Propose horaire Beaupreau-Nantes(aero) ( taxi)

[14:09:30] 164, Deplace trajet bas: taxi : Beaupreau — Nantes(aero)

[14:09:30] 165, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:30] 166, Affiche trajet avion : Nantes — Lille

[14:09:30] 167, Supprime trajet avion : Nantes — Lille

[14:09:30] 168, Le prix total est : 430

[14:09:30] 169, Propose horaire Nantes-Lille (avion)

[14:09:30] 170, Le prix total est : 1232

[14:09:30] 171, Nouveau trajet en 2  
 | avion : Nantes — Lille  
 | Depart le Demain a 07h45  
 | Arrivee le Demain a 09h05  
 | 1 personne(s). Prix : 802 F.

[14:09:30] 172, Deplace trajet bas: avion : Nantes — Lille

[14:09:30] 173, Affiche horaire  
 | Lille(gare)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 180 F

[14:09:30] 174, Affiche trajet taxi : Lille(gare) — Baisieux

[14:09:30] 175, Supprime trajet taxi : Lille(gare) — Baisieux

[14:09:30] 176, Le prix total est : 1052

[14:09:30] 177, Le prix total est : 1232

[14:09:30] 178, Nouveau trajet en 2  
 | taxi : Lille(gare) — Baisieux  
 | Depart le Demain a 09h05  
 | Arrivee le Demain a 09h35  
 | 4 personne(s). Prix : 180 F.

[14:09:30] 179, Propose horaire Lille(gare)-Baisieux ( taxi)

[14:09:31] 180, Test de la solution : negatif

[14:09:33] 181, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:33] 182, Recoit message 15 de Agent aerien

[14:09:35] 183, Affiche message numero 15 de Agent aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air ?pStage(Orleans Paris 12:25 13:30 train 4 80 false))

[14:09:36] 184, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F

[14:09:36] 185, Demande horaires Orleans-Paris ( )

[14:09:36] 186, Memorise horaire

[14:09:36] 187, Ajoute horaire dans panneau Mon probleme  
 | Orleans-Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F

[14:09:36] 188, Envoie horaire Orleans-Paris (train)

[14:09:36] 189, Envoie message 19  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Orleans

Paris 12:25 13:30 train 4 80 false)))  
 | =====HORAIRE=====

[14:09:37] 190, Nouveau message  
 [14:09:37] 191, Envoie message 20  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Lille(airport) Baisieux \* \* taxi \* \*  
 \*)))

[14:09:39] 192, Recoit message 18 de Agent routier  
 [14:09:39] 193, Affiche message numero 18 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(Lille(  
 airport) Baisieux \* +(00:30) taxi 4 170  
 false)))

| =====HORAIRE=====

| Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 170 F

[14:09:39] 194, Memorise horaire de message Agent  
 routier : 18  
 [14:09:39] 195, Ajoute horaire dans panneau Mon  
 probleme  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 170 F

[14:09:44] 196, Reponse a Agent routier : 18  
 [14:09:44] 197, Envoie message 21  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Lille(airport) Baisieux \* +(00:30)  
 taxi 1 170 true)))

[14:09:47] 198, Recoit message 19 de Agent routier  
 [14:09:47] 199, Affiche message numero 19 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(Lille(  
 airport) Baisieux \* +(00:30) taxi 1 170  
 true)))

| =====HORAIRE=====

| Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F

[14:09:47] 200, Memorise horaire de message Agent  
 routier : 19  
 [14:09:47] 201, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F

[14:09:48] 202, Recoit message 20 de Agent routier  
 [14:09:49] 203, Affiche message numero 20 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(  
 SaintMartin Paris 18:10 18:35 train 3 20  
 true)))

[14:09:50] 204, Affiche horaire  
 | SaintMartin-Paris (train)  
 | Date: Demain — 1 personne  
 | 18h10/18h35 — 20 F

[14:09:50] 205, Demande horaires SaintMartin-Paris ( )  
 [14:09:50] 206, Memorise horaire  
 [14:09:50] 207, Ajoute horaire dans panneau Mon  
 probleme  
 | SaintMartin-Paris (train)  
 | Date: Demain — 3 personnes  
 | 18h10/18h35 — 20 F

[14:09:50] 208, Reserve horaire [R] SaintMartin-Paris  
 (train)  
 [14:09:50] 209, Envoie horaire [R] SaintMartin-Paris  
 (train)  
 [14:09:50] 210, Envoie message 22  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(  
 SaintMartin Paris 18:10 18:35 train 3 20  
 true)))

| =====HORAIRE=====

| Reservation effectuee :  
 | SaintMartin-Paris (train)  
 | Date: Demain — 3 personnes  
 | 18h10/18h35 — 20 F

[14:09:53] 211, Affiche horaire  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 170 F

[14:09:53] 212, Supprime horaire  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 170 F

[14:09:54] 213, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:09:55] 214, Recoit message 22 de Agent routier  
 [14:09:56] 215, Affiche message numero 22 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(  
 Chambery LeChatelard >=(23:00) \* train 3 \*  
 false)))

[14:09:57] 216, Memorise horaire  
 [14:09:57] 217, Ajoute horaire dans panneau Mon  
 probleme  
 | SaintMartin-Paris (train)  
 | Date: Demain — 3 personnes  
 | 20h10/20h35 — 20 F

[14:09:57] 218, Nouveau message  
 [14:09:57] 219, Envoie message 23  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(!pStage(  
 Chambery LeChatelard \* \* train \* \* \*)))

[14:09:59] 220, Nouveau message  
 [14:09:59] 221, Envoie message 24  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air query pD(railway ?pStage(  
 Nantes Lille 07:45 09:05 plane(PigeonVol)  
 1 802 true)))

[14:10:01] 222, Affiche horaire  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F

[14:10:02] 223, Recoit message 17 de Agent aerien  
 [14:10:03] 224, Recoit message 18 de Agent aerien  
 [14:10:04] 225, Affiche message numero 17 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway reply pB(pStage(Nantes  
 Lille 07:45 09:05 plane(PigeonVol) 1 802  
 true)))

| =====HORAIRE=====

| Reservation effectuee :  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F

[14:10:04] 226, Memorise horaire de message Agent  
 aerien : 17  
 [14:10:04] 227, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :

| Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:06] 228, Affiche message numero 18 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air ?pStage(  
 Orleans Paris 12:25 13:30 train 4 80 true))  
 )  
 [14:10:07] 229, Recoit message 23 de Agent routier  
 [14:10:07] 230, Affiche horaire  
 | Orleans-Paris (train)  
 | Date: Demain — 1 personne  
 | 12h25/13h30 — 80 F  
 [14:10:07] 231, Demande horaires Orleans-Paris ()  
 [14:10:07] 232, Memorise horaire  
 [14:10:07] 233, Ajoute horaire dans panneau Mon  
 probleme  
 | Orleans-Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:10:08] 234, Reserve horaire [R] Orleans-Paris (  
 train)  
 [14:10:08] 235, Envoie horaire [R] Orleans-Paris (  
 train)  
 [14:10:08] 236, Envoie message 25  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Orleans  
 Paris 12:25 13:30 train 4 80 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Orleans-Paris (train)  
 | Date: Demain — 4 personnes  
 | 12h25/13h30 — 80 F  
 [14:10:09] 237, Affiche message numero 23 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(  
 Paris Chambéry 20:06 23:00 train 3 370 true  
 )))  
 [14:10:10] 238, Affiche horaire  
 | Paris-Chambéry (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:10:10] 239, Demande horaires Paris-Chambéry ()  
 [14:10:10] 240, Memorise horaire  
 [14:10:10] 241, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Chambéry (train)  
 | Date: Demain — 3 personnes  
 | 20h06/23h00 — 370 F  
 [14:10:10] 242, Reserve horaire [R] Paris-Chambéry (  
 train)  
 [14:10:10] 243, Envoie horaire [R] Paris-Chambéry (  
 train)  
 [14:10:10] 244, Envoie message 26  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambéry 20:06 23:00 train 3 370 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Paris-Chambéry (train)  
 | Date: Demain — 3 personnes  
 | 20h06/23h00 — 370 F  
 [14:10:12] 245, Affiche horaire  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F  
 [14:10:13] 246, Affiche horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:13] 247, Supprime horaire  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:13] 248, Affiche horaire  
 | Reservation effectuee :  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:15] 249, Affiche horaire  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:10:15] 250, Recoit message 19 de Agent aerien  
 [14:10:17] 251, Affiche message numero 19 de Agent  
 aerien  
 | De: Agent aerien A: Agent ferroviaire  
 | pMessage(air railway query pD(air ?pStage(  
 Paris Montpellier 15:30 19:55 train 4 590  
 true)))  
 [14:10:18] 252, Affiche horaire  
 | Paris-Montpellier (train)  
 | Date: Demain — 1 personne  
 | 10h05/14h54 — 590 F  
 [14:10:18] 253, Demande horaires Paris-Montpellier ()  
 [14:10:18] 254, Memorise horaire  
 [14:10:18] 255, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:10:18] 256, Reserve horaire [R] Paris-Montpellier  
 (train)  
 [14:10:18] 257, Envoie horaire [R] Paris-Montpellier  
 (train)  
 [14:10:18] 258, Envoie message 27  
 | De: Agent ferroviaire A: Agent aerien  
 | pMessage(railway air reply pB(pStage(Paris  
 Montpellier 15:30 19:55 train 4 590 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Paris-Montpellier (train)  
 | Date: Demain — 4 personnes  
 | 15h30/19h55 — 590 F  
 [14:10:19] 259, Nouveau message  
 [14:10:19] 260, Envoie message 28  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Beaupreau Nantes(airport) \* +(00:35)  
 taxi 1 250 true)))  
 [14:10:21] 261, Affiche message numero 1 de Agent  
 aerien  
 [14:10:21] 262, Recoit message 24 de Agent routier  
 [14:10:21] 263, Supprime message Agent aerien : 1  
 [14:10:23] 264, Affiche message numero 24 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(  
 Beaupreau Nantes(airport) \* +(00:35) taxi 1  
 250 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:10:23] 265, Memorise horaire de message Agent  
 routier : 24  
 [14:10:23] 266, Ajoute horaire dans panneau Mon  
 probleme  
 | Reservation effectuee :  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F

[14:10:25] 267, Affiche horaire  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F  
 [14:10:26] 268, Affiche horaire  
 | Reservation effectuee :  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:27] 269, Affiche horaire  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:10:27] 270, Supprime horaire  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:10:27] 271, Affiche horaire  
 | Reservation effectuee :  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:10:28] 272, Affiche trajet taxi : Beaupreau —  
 Nantes(aero)  
 [14:10:28] 273, Supprime trajet taxi : Beaupreau —  
 Nantes(aero)  
 [14:10:28] 274, Le prix total est : 982  
 [14:10:28] 275, Le prix total est : 1232  
 [14:10:28] 276, Nouveau trajet en 1  
 | Reservation effectuee :  
 | taxi : Beaupreau — Nantes(aero)  
 | Depart le Demain a 07h10  
 | Arrivee le Demain a 07h45  
 | 1 personne(s). Prix : 250 F.  
 [14:10:28] 277, Propose horaire [R] Beaupreau-Nantes(  
 aero) (taxi)  
 [14:10:28] 278, Deplace trajet bas: [R] taxi :  
 Beaupreau — Nantes(aero)  
 [14:10:28] 279, Affiche horaire  
 | Reservation effectuee :  
 | Nantes-Lille (avion Pigeon Vol)  
 | Date: Demain — 1 personne  
 | 07h45/09h05 — 802 F  
 [14:10:28] 280, Affiche trajet avion : Nantes — Lille  
 [14:10:28] 281, Supprime trajet avion : Nantes —  
 Lille  
 [14:10:28] 282, Le prix total est : 430  
 [14:10:28] 283, Propose horaire [R] Nantes-Lille (  
 avion)  
 [14:10:28] 284, Le prix total est : 1232  
 [14:10:28] 285, Nouveau trajet en 2  
 | Reservation effectuee :  
 | avion : Nantes — Lille  
 | Depart le Demain a 07h45  
 | Arrivee le Demain a 09h05  
 | 1 personne(s). Prix : 802 F.  
 [14:10:28] 286, Deplace trajet bas: [R] avion :  
 Nantes — Lille  
 [14:10:28] 287, Affiche horaire  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h30 — 170 F  
 [14:10:28] 288, Affiche trajet taxi : Lille(gare) —  
 Baisieux  
 [14:10:28] 289, Supprime trajet taxi : Lille(gare) —  
 Baisieux  
 [14:10:28] 290, Le prix total est : 1052  
 [14:10:28] 291, Le prix total est : 1222  
 [14:10:28] 292, Nouveau trajet en 2  
 | Reservation effectuee :  
 | taxi : Lille(aero) — Baisieux  
 | Depart le Demain a 09h05  
 | Arrivee le Demain a 09h35  
 | 1 personne(s). Prix : 170 F.  
 [14:10:28] 293, Propose horaire [R] Lille(aero)-  
 Baisieux (taxi)  
 [14:10:28] 294, Recoit message 26 de Agent routier  
 [14:10:30] 295, Test de la solution : positif  
 [14:10:31] 296, Affiche message numero 26 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway inform pB(pProblem(road  
 true)))  
 [14:10:39] 297, Fin de l' experimentation

<b>Agent Routier</b> careful=1 cooperative=3 estimator=3 opportunistic=1 optimizer=1 patient=3 precise=1 thrifty=3
--

Problème à résoudre : SaintMartin/LeChalelard, 3 personnes, départ après 17h30, 1430F.

```

[14:08:28] Sujet: Bob Leponge
[14:08:28] Role: Agent routier
[14:08:28] Probleme: Aller
[14:08:28] Date: Mon Apr 24 14:08:28 CEST 2006
[14:08:28] 1, Demande horaires SaintMartin-Paris(gare) (taxi)
[14:08:28] 2, Demande horaires SaintMartin-Paris(aero) (taxi)
[14:08:28] 3, Demande horaires Paris(gare)-Paris(aero) (taxi)
[14:08:28] 4, Memorise horaire
[14:08:28] 5, Ajoute horaire dans panneau Mon probleme
    | Paris(gare)-Paris(aero) ( taxi)
    | Date: Demain — 1 personne
    | duree : 00h45 — 300 F
[14:08:29] 6, Demande horaires SaintMartin-Paris(gare) (car)
[14:08:30] 7, Nouveau message
[14:08:30] 8, Envoie message 1
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway query pD(road pStage(SaintMartin Paris >=(17:30) * train * ? false))
[14:08:31] 9, Demande horaires Chambery(gare)-LeChatelard (car)
[14:08:31] 10, Memorise horaire
[14:08:31] 11, Ajoute horaire dans panneau Mon probleme
    | Chambery(gare)-LeChatelard ( car)
    | Date: Demain — 1 personne
    | 19h10/19h40 — 40 F
[14:08:32] 12, Nouveau message
[14:08:32] 13, Envoie message 2
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway query pD(road pStage(Paris Lyon * * train * ? false)))
[14:08:32] 14, Recoit message 3 de Agent ferroviaire
[14:08:33] 15, Affiche message numero 3 de Agent ferroviaire
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road reply pB(pStage(SaintMartin Paris 18:10 18:35 train 1 20 false)))
    | =====HORAIRE=====
    | SaintMartin-Paris (train)
    | Date: Demain — 1 personne
    | 18h10/18h35 — 20 F
[14:08:34] 16, Memorise horaire de message Agent ferroviaire : 3
[14:08:34] 17, Ajoute horaire dans panneau Mon probleme
    | SaintMartin-Paris (train)
    | Date: Demain — 1 personne
    | 18h10/18h35 — 20 F
[14:08:34] 18, Recoit message 4 de Agent aerien
[14:08:35] 19, Memorise horaire
[14:08:35] 20, Ajoute horaire dans panneau Mon probleme
    | Chambery(gare)-LeChatelard ( car)
    | Date: Demain — 1 personne
    | 19h10/19h40 — 40 F
[14:08:35] 21, Recoit message 5 de Agent aerien
[14:08:35] 22, Recoit message 4 de Agent ferroviaire
[14:08:36] 23, Affiche message numero 4 de Agent aerien
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(ChateauNeuf Orleans morning ? OR(coach taxi) * * false)))
[14:08:37] 24, Demande horaires ChateauNeuf-Orleans(gare) (car)
[14:08:37] 25, Memorise horaire
[14:08:37] 26, Ajoute horaire dans panneau Mon probleme
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:08:37] 27, Envoie horaire ChateauNeuf-Orleans(gare) (car)
[14:08:37] 28, Envoie message 3
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(ChateauNeuf Orleans(station) 10:10 10:45 coach 1 24 false)))
    | =====HORAIRE=====
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain — 1 personne
    | 10h10/10h45 — 24 F
[14:08:38] 29, Affiche message numero 5 de Agent aerien
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(Montpellier Maguelone * afternoon coach * * false)))
[14:08:40] 30, Demande horaires Montpellier(gare)-Maguelone (car)
[14:08:40] 31, Demande horaires Montpellier(aero)-Maguelone (car)
[14:08:40] 32, Memorise horaire
[14:08:40] 33, Ajoute horaire dans panneau Mon probleme
    | Montpellier(aero)-Maguelone ( car)
    | Date: Demain — 1 personne
    | 12h30/13h00 — 60 F
[14:08:40] 34, Envoie horaire Montpellier(aero)-Maguelone (car)
[14:08:40] 35, Envoie message 4
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(Montpellier (airport) Maguelone 12:30 13:00 coach 1 60 false)))
    | =====HORAIRE=====
    | Montpellier(aero)-Maguelone ( car)
    | Date: Demain — 1 personne
    | 12h30/13h00 — 60 F
[14:08:41] 36, Affiche message numero 4 de Agent ferroviaire
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road reply pB(pStage(Paris Lyon * * train 1 350 false)))
[14:08:42] 37, Reponse a Agent ferroviaire : 4
[14:08:42] 38, Envoie message 5
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway query pD(road pStage(Paris Lyon >=(18:35) * train * ? false)))
[14:08:44] 39, Memorise horaire
[14:08:44] 40, Ajoute horaire dans panneau Mon

```

probleme  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 1 personne  
 | 19h30/20h00 — 60 F  
 [14:08:45] 41, Reponse a Agent ferroviaire : 4  
 [14:08:45] 42, Envoie message 6  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(  
 Lyon Chambéry afternoon \* train \* \* false))  
 )  
 [14:08:47] 43, Affiche message numero 1 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 SaintMartin Paris >=(17:30) \* train \* ?  
 false)))  
 [14:08:47] 44, Supprime message (Agent routier : 1)  
 [14:08:47] 45, Recoit message 8 de Agent ferroviaire  
 [14:08:48] 46, Affiche message numero 5 de Agent  
 routier  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Lyon >=(18:35) \* train \* ? false)))  
 [14:08:48] 47, Supprime message (Agent routier : 5)  
 [14:08:50] 48, Affiche message numero 8 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Lyon 20:31 22:37 train 1 350 false)))  
 | =====HORAIRE=====  
 | Paris-Lyon (train)  
 | Date: Demain — 1 personne  
 | 20h31/22h37 — 350 F  
 [14:08:50] 49, Memorise horaire de message Agent  
 ferroviaire : 8  
 [14:08:50] 50, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Lyon (train)  
 | Date: Demain — 1 personne  
 | 20h31/22h37 — 350 F  
 [14:08:50] 51, Recoit message 9 de Agent ferroviaire  
 [14:08:51] 52, Affiche message numero 9 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(!pStage(Lyon  
 Chambéry \* \* train \* \* \*)))  
 [14:08:52] 53, Reponse a Agent ferroviaire : 9  
 [14:08:52] 54, Envoie message 7  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Chambéry \* \* train \* ? false)))  
 [14:08:54] 55, Affiche horaire  
 | SaintMartin-Paris (train)  
 | Date: Demain — 1 personne  
 | 18h10/18h35 — 20 F  
 [14:08:55] 56, Affiche horaire  
 | Chambéry(gare)-LeChatelard ( car)  
 | Date: Demain — 1 personne  
 | 19h10/19h40 — 40 F  
 [14:08:56] 57, Recoit message 12 de Agent ferroviaire  
 [14:08:57] 58, Affiche message numero 12 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambéry \* \* train 1 370 false)))  
 [14:08:58] 59, Reponse a Agent ferroviaire : 12  
 [14:08:58] 60, Envoie message 8  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Chambéry \* evening train \* ? false)))  
 [14:08:58] 61, Recoit message 13 de Agent ferroviaire  
 [14:08:59] 62, Affiche message numero 13 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Beaupreau Nantes early \* OR(coach  
 taxi) \* \* false)))  
 [14:09:00] 63, Recoit message 14 de Agent ferroviaire  
 [14:09:01] 64, Nouveau message  
 [14:09:01] 65, Envoie message 9  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway refine pD(road pD(  
 railway ?pStage(Beaupreau ??(Nantes(station  
 ) Nantes(airport)) early \* OR(coach taxi) \*  
 \* false))))  
 [14:09:02] 66, Affiche message numero 14 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambéry 18:06 21:13 train 4 370 false)))  
 | =====HORAIRE=====  
 | Paris-Chambéry (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:02] 67, Memorise horaire de message Agent  
 ferroviaire : 14  
 [14:09:02] 68, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris-Chambéry (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:02] 69, Recoit message 15 de Agent ferroviaire  
 [14:09:03] 70, Recoit message 10 de Agent aerien  
 [14:09:04] 71, Affiche message numero 15 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Beaupreau Nantes(airport) early \* OR  
 (coach taxi) \* \* false)))  
 [14:09:05] 72, Demande horaires Beaupreau-Nantes(aero  
 ) (car)  
 [14:09:05] 73, Demande horaires Beaupreau-Nantes(aero  
 ) (taxi)  
 [14:09:05] 74, Memorise horaire  
 [14:09:05] 75, Ajoute horaire dans panneau Mon  
 probleme  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:09:05] 76, Envoie horaire Beaupreau-Nantes(aero)  
 (taxi)  
 [14:09:05] 77, Envoie message 10  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(  
 Beaupreau Nantes(airport) \* +(00:35) taxi 1  
 250 false)))  
 | =====HORAIRE=====  
 | Beaupreau-Nantes(aero) ( taxi)  
 | Date: Demain — 1 personne  
 | duree : 00h35 — 250 F  
 [14:09:06] 78, Affiche message numero 10 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air pStage(  
 Montpellier(airport) Maguelone >=(19:55) ?  
 coach 4 \* false)))  
 [14:09:07] 79, Affiche horaire  
 | Montpellier(aero)-Maguelone ( car)  
 | Date: Demain — 1 personne  
 | 12h30/13h00 — 60 F  
 [14:09:07] 80, Demande horaires Montpellier(aero)-  
 Maguelone (car)  
 [14:09:07] 81, Memorise horaire  
 [14:09:07] 82, Ajoute horaire dans panneau Mon  
 probleme  
 | Montpellier(aero)-Maguelone ( car)

| Date: Demain — 4 personnes  
 | 19h30/20h00 — 60 F  
 [14:09:07] 83, Nouveau message  
 [14:09:07] 84, Envoie message 11  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(!pStage(  
 Montpellier(airport) Maguelone >=(19:55) \*  
 coach 4 \* false)))  
 [14:09:07] 85, Envoie horaire Montpellier(aero)—  
 Maguelone (car)  
 [14:09:07] 86, Envoie message 12  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(Montpellier  
 (airport) Maguelone 19:30 20:00 coach 4 60  
 false)))  
 | =====HORAIRE=====  
 | Montpellier(aero)—Maguelone ( car)  
 | Date: Demain — 4 personnes  
 | 19h30/20h00 — 60 F  
 [14:09:08] 87, Recoit message 17 de Agent ferroviaire  
 [14:09:08] 88, Affiche message numero 17 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road query pD(railway ?  
 pStage(Lille Baisieux \* \* taxi \* \* false)))  
 [14:09:09] 89, Demande horaires Lille(gare)—Baisieux  
 (taxi)  
 [14:09:09] 90, Demande horaires Lille(aero)—Baisieux  
 (taxi)  
 [14:09:09] 91, Demande horaires Lille(gare)—Baisieux  
 (taxi)  
 [14:09:09] 92, Memorise horaire  
 [14:09:09] 93, Ajoute horaire dans panneau Mon  
 probleme  
 | Lille(gare)—Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 180 F  
 [14:09:10] 94, Envoie horaire Lille(gare)—Baisieux ( taxi)  
 [14:09:10] 95, Envoie message 13  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(Lille(  
 station) Baisieux \* +(00:30) taxi 4 180  
 false)))  
 | =====HORAIRE=====  
 | Lille(gare)—Baisieux ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 180 F  
 [14:09:10] 96, Recoit message 12 de Agent aerien  
 [14:09:11] 97, Affiche message numero 12 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(  
 Montpellier(airport) Maguelone \* \* taxi 4 \*  
 false)))  
 [14:09:12] 98, Demande horaires Montpellier(aero)—  
 Maguelone (taxi)  
 [14:09:12] 99, Memorise horaire  
 [14:09:12] 100, Ajoute horaire dans panneau Mon  
 probleme  
 | Montpellier(aero)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 300 F  
 [14:09:12] 101, Envoie horaire Montpellier(aero)—  
 Maguelone (taxi)  
 [14:09:12] 102, Envoie message 14  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(Montpellier  
 (airport) Maguelone \* +(00:30) taxi 4 300  
 false)))  
 | =====HORAIRE=====  
 | Montpellier(aero)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 00h30 — 300 F  
 [14:09:14] 103, Affiche horaire  
 | SaintMartin—Paris (train)  
 | Date: Demain — 1 personne  
 | 18h10/18h35 — 20 F  
 [14:09:15] 104, Affiche horaire  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:17] 105, Nouveau message  
 [14:09:17] 106, Envoie message 15  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road pStage(  
 Paris Chambery >=(18:35) ? train \* \* \*)))  
 [14:09:19] 107, Recoit message 18 de Agent  
 ferroviaire  
 [14:09:19] 108, Affiche message numero 18 de Agent  
 ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris  
 Chambery 20:06 23:00 train 4 370 false)))  
 | =====HORAIRE=====  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes  
 | 20h06/23h00 — 370 F  
 [14:09:19] 109, Memorise horaire de message Agent  
 ferroviaire : 18  
 [14:09:19] 110, Ajoute horaire dans panneau Mon  
 probleme  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes  
 | 20h06/23h00 — 370 F  
 [14:09:21] 111, Recoit message 13 de Agent aerien  
 [14:09:22] 112, Affiche message numero 13 de Agent  
 aerien  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(  
 Montpellier(station) Maguelone \* \* taxi \* \*  
 \*)))  
 [14:09:23] 113, Demande horaires Montpellier(gare)—  
 Maguelone (taxi)  
 [14:09:23] 114, Memorise horaire  
 [14:09:23] 115, Ajoute horaire dans panneau Mon  
 probleme  
 | Montpellier(gare)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 01h00 — 450 F  
 [14:09:23] 116, Envoie horaire Montpellier(gare)—  
 Maguelone (taxi)  
 [14:09:23] 117, Envoie message 16  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(Montpellier  
 (station) Maguelone \* +(01:00) taxi 4 450  
 false)))  
 | =====HORAIRE=====  
 | Montpellier(gare)—Maguelone ( taxi)  
 | Date: Demain — 4 personnes  
 | duree : 01h00 — 450 F  
 [14:09:25] 118, Affiche horaire  
 | SaintMartin—Paris (train)  
 | Date: Demain — 1 personne  
 | 18h10/18h35 — 20 F  
 [14:09:26] 119, Affiche horaire  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:26] 120, Supprime horaire  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes  
 | 18h06/21h13 — 370 F  
 [14:09:26] 121, Affiche horaire  
 | Paris—Chambery (train)  
 | Date: Demain — 4 personnes

```

    | 20h06/23h00 - 370 F
[14:09:27] 122, Affiche horaire
    | Chambéry(gare)-LeChatelard ( car)
    | Date: Demain - 1 personne
    | 19h10/19h40 - 40 F
[14:09:28] 123, Affiche horaire
    | SaintMartin-Paris (train)
    | Date: Demain - 1 personne
    | 18h10/18h35 - 20 F
[14:09:28] 124, Propose horaire SaintMartin-Paris (
    train)
[14:09:28] 125, Le prix total est : 20
[14:09:28] 126, Nouveau trajet en 0
    | train : SaintMartin - Paris
    | Depart le Demain a 18h10
    | Arrivee le Demain a 18h35
    | 1 personne(s). Prix : 20 F.
[14:09:28] 127, Affiche horaire
    | Paris-Chambéry (train)
    | Date: Demain - 4 personnes
    | 20h06/23h00 - 370 F
[14:09:28] 128, Propose horaire Paris-Chambéry (train
    )
[14:09:28] 129, Le prix total est : 1500
[14:09:28] 130, Nouveau trajet en 1
    | train : Paris - Chambéry
    | Depart le Demain a 20h06
    | Arrivee le Demain a 23h00
    | 4 personne(s). Prix : 370 F.
[14:09:28] 131, Affiche horaire
    | Chambéry(gare)-LeChatelard ( car)
    | Date: Demain - 1 personne
    | 19h10/19h40 - 40 F
[14:09:28] 132, Propose horaire Chambéry(gare)-
    LeChatelard (car)
[14:09:28] 133, Le prix total est : 1540
[14:09:28] 134, Nouveau trajet en 2
    | car : Chambéry(gare) - LeChatelard
    | Depart le Demain a 19h10
    | Arrivee le Demain a 19h40
    | 1 personne(s). Prix : 40 F.
[14:09:29] 135, Recoit message 14 de Agent aerien
[14:09:29] 136, Test de la solution : negatif
[14:09:30] 137, Affiche message numero 14 de Agent
    aerien
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(
    ChateauNeuf Orleans(station) 10:10 10:45
    coach 4 24 false)))
[14:09:31] 138, Affiche horaire
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain - 1 personne
    | 10h10/10h45 - 24 F
[14:09:31] 139, Demande horaires ChateauNeuf-Orleans(
    gare) (car)
[14:09:31] 140, Memorise horaire
[14:09:31] 141, Ajoute horaire dans panneau Mon
    probleme
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain - 4 personnes
    | 10h10/10h45 - 24 F
[14:09:31] 142, Envoie horaire ChateauNeuf-Orleans(
    gare) (car)
[14:09:31] 143, Envoie message 17
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(ChateauNeuf
    Orleans(station) 10:10 10:45 coach 4 24
    false)))
    | =====HORAIRE=====
    | ChateauNeuf-Orleans(gare) ( car)
    | Date: Demain - 4 personnes
    | 10h10/10h45 - 24 F
[14:09:34] 144, Affiche horaire
    | SaintMartin-Paris (train)
    | Date: Demain - 1 personne
    | 18h10/18h35 - 20 F
[14:09:35] 145, Affiche horaire
    | Paris-Chambéry (train)
    | Date: Demain - 4 personnes
    | 20h06/23h00 - 370 F
[14:09:36] 146, Affiche horaire
    | Chambéry(gare)-LeChatelard ( car)
    | Date: Demain - 1 personne
    | 19h10/19h40 - 40 F
[14:09:37] 147, Recoit message 20 de Agent
    ferroviaire
[14:09:38] 148, Affiche message numero 20 de Agent
    ferroviaire
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road query pD(railway ?
    pStage(Lille(airport) Baisieux * * taxi * *
    *)))
[14:09:39] 149, Demande horaires Lille(aero)-Baisieux
    (taxi)
[14:09:39] 150, Memorise horaire
[14:09:39] 151, Ajoute horaire dans panneau Mon
    probleme
    | Lille(aero)-Baisieux ( taxi)
    | Date: Demain - 4 personnes
    | duree : 00h30 - 170 F
[14:09:39] 152, Envoie horaire Lille(aero)-Baisieux (
    taxi)
[14:09:39] 153, Envoie message 18
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway reply pB(pStage(Lille(
    airport) Baisieux * +(00:30) taxi 4 170
    false)))
    | =====HORAIRE=====
    | Lille(aero)-Baisieux ( taxi)
    | Date: Demain - 4 personnes
    | duree : 00h30 - 170 F
[14:09:40] 154, Affiche horaire
    | Chambéry(gare)-LeChatelard ( car)
    | Date: Demain - 1 personne
    | 19h10/19h40 - 40 F
[14:09:40] 155, Demande horaires Chambéry(gare)-
    LeChatelard (car)
[14:09:40] 156, Memorise horaire
[14:09:40] 157, Ajoute horaire dans panneau Mon
    probleme
    | Chambéry(gare)-LeChatelard ( car)
    | Date: Demain - 3 personnes
    | 19h10/19h40 - 40 F
[14:09:41] 158, Demande horaires Chambéry(gare)-
    LeChatelard (taxi)
[14:09:41] 159, Memorise horaire
[14:09:41] 160, Ajoute horaire dans panneau Mon
    probleme
    | Chambéry(gare)-LeChatelard ( taxi)
    | Date: Demain - 3 personnes
    | duree : 00h25 - 200 F
[14:09:43] 161, Affiche horaire
    | SaintMartin-Paris (train)
    | Date: Demain - 1 personne
    | 18h10/18h35 - 20 F
[14:09:44] 162, Recoit message 21 de Agent
    ferroviaire
[14:09:46] 163, Affiche message numero 21 de Agent
    ferroviaire
    | De: Agent ferroviaire A: Agent routier
    | pMessage(railway road query pD(railway ?
    pStage(Lille(airport) Baisieux * +(00:30)
    taxi 1 170 true)))
[14:09:47] 164, Affiche horaire
    | Lille(aero)-Baisieux ( taxi)
    | Date: Demain - 4 personnes

```

| duree : 00h30 - 170 F  
 [14:09:47] 165, Demande horaires Lille(aero)-Baisieux (taxi)  
 [14:09:47] 166, Memorise horaire  
 [14:09:47] 167, Ajoute horaire dans panneau Mon probleme  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain - 1 personne  
 | duree : 00h30 - 170 F  
 [14:09:47] 168, Reserve horaire [R] Lille(aero)-Baisieux (taxi)  
 [14:09:47] 169, Envoie horaire [R] Lille(aero)-Baisieux (taxi)  
 [14:09:47] 170, Envoie message 19  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway reply pB(pStage(Lille(airport) Baisieux \* +(00:30) taxi 1 170 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | Lille(aero)-Baisieux ( taxi)  
 | Date: Demain - 1 personne  
 | duree : 00h30 - 170 F  
 [14:09:48] 171, Reponse a Agent ferroviaire : 21  
 [14:09:48] 172, Envoie message 20  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(SaintMartin Paris 18:10 18:35 train 3 20 true)))  
 [14:09:50] 173, Recoit message 16 de Agent aerien  
 [14:09:50] 174, Recoit message 22 de Agent ferroviaire  
 [14:09:52] 175, Affiche message numero 16 de Agent aerien  
 | De: Agent aerien A: Agent routier  
 | pMessage(air road query pD(air ?pStage(ChateauNeuf Orleans(station) 10:10 10:45 coach 4 24 true)))  
 [14:09:54] 176, Affiche horaire  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 1 personne  
 | 10h10/10h45 - 24 F  
 [14:09:54] 177, Demande horaires ChateauNeuf-Orleans(gare) (car)  
 [14:09:54] 178, Memorise horaire  
 [14:09:54] 179, Ajoute horaire dans panneau Mon probleme  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 4 personnes  
 | 10h10/10h45 - 24 F  
 [14:09:54] 180, Reserve horaire [R] ChateauNeuf-Orleans(gare) (car)  
 [14:09:54] 181, Envoie horaire [R] ChateauNeuf-Orleans(gare) (car)  
 [14:09:54] 182, Envoie message 21  
 | De: Agent routier A: Agent aerien  
 | pMessage(road air reply pB(pStage(ChateauNeuf Orleans(station) 10:10 10:45 coach 4 24 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | ChateauNeuf-Orleans(gare) ( car)  
 | Date: Demain - 4 personnes  
 | 10h10/10h45 - 24 F  
 [14:09:55] 183, Nouveau message  
 [14:09:55] 184, Envoie message 22  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(Chambery LeChatelard >=(23:00) \* train 3 \* false)))  
 [14:09:56] 185, Affiche message numero 22 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(SaintMartin Paris 18:10 18:35 train 3 20 true)))  
 | =====HORAIRE=====  
 | Reservation effectuee :  
 | SaintMartin-Paris (train)  
 | Date: Demain - 3 personnes  
 | 18h10/18h35 - 20 F  
 [14:09:56] 186, Memorise horaire de message Agent ferroviaire : 22  
 [14:09:56] 187, Ajoute horaire dans panneau Mon probleme  
 | Reservation effectuee :  
 | SaintMartin-Paris (train)  
 | Date: Demain - 3 personnes  
 | 18h10/18h35 - 20 F  
 [14:09:57] 188, Recoit message 23 de Agent ferroviaire  
 [14:09:58] 189, Affiche message numero 23 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(!pStage(Chambery LeChatelard \* \* train \* \* \*)))  
 [14:10:00] 190, Affiche horaire  
 | Chambery(gare)-LeChatelard ( car)  
 | Date: Demain - 1 personne  
 | 19h10/19h40 - 40 F  
 [14:10:00] 191, Demande horaires Chambery(gare)-LeChatelard (car)  
 [14:10:00] 192, Memorise horaire  
 [14:10:00] 193, Ajoute horaire dans panneau Mon probleme  
 | Chambery(gare)-LeChatelard ( car)  
 | Date: Demain - 3 personnes  
 | 19h10/19h40 - 40 F  
 [14:10:01] 194, Affiche horaire  
 | Chambery(gare)-LeChatelard ( taxi)  
 | Date: Demain - 3 personnes  
 | duree : 00h25 - 200 F  
 [14:10:03] 195, Affiche horaire  
 | SaintMartin-Paris (train)  
 | Date: Demain - 1 personne  
 | 18h10/18h35 - 20 F  
 [14:10:03] 196, Supprime horaire  
 | SaintMartin-Paris (train)  
 | Date: Demain - 1 personne  
 | 18h10/18h35 - 20 F  
 [14:10:03] 197, Affiche horaire  
 | Reservation effectuee :  
 | SaintMartin-Paris (train)  
 | Date: Demain - 3 personnes  
 | 18h10/18h35 - 20 F  
 [14:10:05] 198, Affiche horaire  
 | Paris-Chambery (train)  
 | Date: Demain - 4 personnes  
 | 20h06/23h00 - 370 F  
 [14:10:07] 199, Reponse a Agent ferroviaire : 23  
 [14:10:07] 200, Envoie message 23  
 | De: Agent routier A: Agent ferroviaire  
 | pMessage(road railway query pD(road ?pStage(Paris Chambery 20:06 23:00 train 3 370 true)))  
 [14:10:09] 201, Affiche horaire  
 | Reservation effectuee :  
 | SaintMartin-Paris (train)  
 | Date: Demain - 3 personnes  
 | 18h10/18h35 - 20 F  
 [14:10:10] 202, Recoit message 26 de Agent ferroviaire  
 [14:10:12] 203, Affiche message numero 26 de Agent ferroviaire  
 | De: Agent ferroviaire A: Agent routier  
 | pMessage(railway road reply pB(pStage(Paris

```

    Chambéry 20:06 23:00 train 3 370 true)))
| =====HORAIRE=====
| Reservation effectuee :
| Paris-Chambéry (train)
| Date: Demain — 3 personnes
| 20h06/23h00 — 370 F
[14:10:12] 204, Memorise horaire de message Agent
ferroviaire : 26
[14:10:12] 205, Ajoute horaire dans panneau Mon
probleme
| Reservation effectuee :
| Paris-Chambéry (train)
| Date: Demain — 3 personnes
| 20h06/23h00 — 370 F
[14:10:15] 206, Affiche horaire
| Reservation effectuee :
| SaintMartin-Paris (train)
| Date: Demain — 3 personnes
| 18h10/18h35 — 20 F
[14:10:16] 207, Affiche horaire
| Paris-Chambéry (train)
| Date: Demain — 4 personnes
| 20h06/23h00 — 370 F
[14:10:16] 208, Supprime horaire
| Paris-Chambéry (train)
| Date: Demain — 4 personnes
| 20h06/23h00 — 370 F
[14:10:16] 209, Affiche horaire
| Reservation effectuee :
| Paris-Chambéry (train)
| Date: Demain — 3 personnes
| 20h06/23h00 — 370 F
[14:10:17] 210, Affiche horaire
| Chambéry(gare)-LeChatelard ( taxi)
| Date: Demain — 3 personnes
| duree : 00h25 — 200 F
[14:10:19] 211, Reserve horaire [R] Chambéry(gare)-
LeChatelard (taxi)
[14:10:19] 212, Recoit message 28 de Agent
ferroviaire
[14:10:20] 213, Affiche message numero 28 de Agent
ferroviaire
| De: Agent ferroviaire A: Agent routier
| pMessage(railway road query pD(railway ?
pStage(Beaupreau Nantes(airport) * +(00:35)
taxi 1 250 true)))
[14:10:21] 214, Affiche horaire
| Beaupreau-Nantes(aero) ( taxi)
| Date: Demain — 1 personne
| duree : 00h35 — 250 F
[14:10:21] 215, Reserve horaire [R] Beaupreau-Nantes(
aero) (taxi)
[14:10:21] 216, Envoie horaire [R] Beaupreau-Nantes(
aero) (taxi)
[14:10:21] 217, Envoie message 24
| De: Agent routier A: Agent ferroviaire
| pMessage(road railway reply pB(pStage(
Beaupreau Nantes(airport) * +(00:35) taxi 1
250 true)))
| =====HORAIRE=====
| Reservation effectuee :
| Beaupreau-Nantes(aero) ( taxi)
| Date: Demain — 1 personne
| duree : 00h35 — 250 F
[14:10:24] 218, Affiche horaire
| Reservation effectuee :
| SaintMartin-Paris (train)
| Date: Demain — 3 personnes
| 18h10/18h35 — 20 F
[14:10:25] 219, Affiche horaire
| Reservation effectuee :
| Paris-Chambéry (train)
| Date: Demain — 3 personnes
| 20h06/23h00 — 370 F
[14:10:26] 220, Affiche horaire
| Reservation effectuee :
| Chambéry(gare)-LeChatelard ( taxi)
| Date: Demain — 3 personnes
| duree : 00h25 — 200 F
[14:10:27] 221, Affiche horaire
| Reservation effectuee :
| SaintMartin-Paris (train)
| Date: Demain — 3 personnes
| 18h10/18h35 — 20 F
[14:10:27] 222, Affiche trajet train : SaintMartin —
Paris
[14:10:27] 223, Supprime trajet train : SaintMartin —
Paris
[14:10:27] 224, Le prix total est : 1520
[14:10:27] 225, Propose horaire [R] SaintMartin-Paris
(train)
[14:10:27] 226, Le prix total est : 1580
[14:10:27] 227, Nouveau trajet en 1
| Reservation effectuee :
| train : SaintMartin — Paris
| Depart le Demain a 18h10
| Arrivee le Demain a 18h35
| 3 personne(s). Prix : 20 F.
[14:10:27] 228, Deplace trajet bas: [R] train :
SaintMartin — Paris
[14:10:27] 229, Affiche horaire
| Reservation effectuee :
| Paris-Chambéry (train)
| Date: Demain — 3 personnes
| 20h06/23h00 — 370 F
[14:10:27] 230, Affiche trajet train : Paris —
Chambéry
[14:10:27] 231, Supprime trajet train : Paris —
Chambéry
[14:10:27] 232, Le prix total est : 100
[14:10:27] 233, Propose horaire [R] Paris-Chambéry (
train)
[14:10:27] 234, Le prix total est : 1210
[14:10:27] 235, Nouveau trajet en 2
| Reservation effectuee :
| train : Paris — Chambéry
| Depart le Demain a 20h06
| Arrivee le Demain a 23h00
| 3 personne(s). Prix : 370 F.
[14:10:27] 236, Deplace trajet bas: [R] train : Paris
— Chambéry
[14:10:27] 237, Affiche horaire
| Reservation effectuee :
| Chambéry(gare)-LeChatelard ( taxi)
| Date: Demain — 3 personnes
| duree : 00h25 — 200 F
[14:10:27] 238, Affiche trajet car : Chambéry(gare) —
LeChatelard
[14:10:27] 239, Supprime trajet car : Chambéry(gare) —
LeChatelard
[14:10:27] 240, Le prix total est : 1170
[14:10:27] 241, Le prix total est : 1370
[14:10:27] 242, Nouveau trajet en 2
| Reservation effectuee :
| taxi : Chambéry(gare) — LeChatelard
| Depart le Demain a 23h00
| Arrivee le Demain a 23h25
| 3 personne(s). Prix : 200 F.
[14:10:27] 243, Propose horaire [R] Chambéry(gare)-
LeChatelard (taxi)
[14:10:27] 244, Recoit message 20 de Agent aerien
[14:10:28] 245, Test de la solution : positif
[14:10:28] 246, Nouveau message
[14:10:28] 247, Envoie message 25
| De: Agent routier A: Agent aerien
| pMessage(road air inform pB(pProblem(road

```

```

    true)))
[14:10:28] 248, Nouveau message
[14:10:28] 249, Envoie message 26
    | De: Agent routier A: Agent ferroviaire
    | pMessage(road railway inform pB(pProblem(road
      true)))
[14:10:29] 250, Affiche message numero 20 de Agent
aerien
    | De: Agent aerien A: Agent routier
    | pMessage(air road query pD(air ?pStage(
      Montpellier(station) Maguelone * +(01:00)
      taxi 4 450 true)))
[14:10:31] 251, Affiche horaire
    | Montpellier(gare)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 01h00 — 450 F
[14:10:31] 252, Reserve horaire [R] Montpellier(gare)
-Maguelone (taxi)
[14:10:31] 253, Envoie horaire [R] Montpellier(gare)-
Maguelone (taxi)
[14:10:31] 254, Envoie message 27
    | De: Agent routier A: Agent aerien
    | pMessage(road air reply pB(pStage(Montpellier
      (station) Maguelone * +(01:00) taxi 4 450
      true)))
    | =====HORAIRE=====
    | Reservation effectuee :
    | Montpellier(gare)-Maguelone ( taxi)
    | Date: Demain — 4 personnes
    | duree : 01h00 — 450 F
[14:10:39] 255, Fin de l' experimentation

```



# Bibliographie

- [Allen *et al.*, 2000] J. Allen, G. Ferguson, B.W. Miller, E.K. Ringger, et T. Sikorski-Zollo. Dialogue systems : From theory to practice in TRAINS-96. Dans *Handbook of Natural Language Processing*, éditeurs Robert Dale, Hermann Moisl, et Harold Somers, pages 347–376, Édinburgh, Écosse, 2000.
- [Allen *et al.*, 2002] J. Allen, N. Blaylock, et G. Ferguson. A problem solving model for collaborative agents. Dans *Actes d'Autonomous Agents and MultiAgent Systems (AAMAS'02)*, pages 774–781, Bologne, Italie, 2002.
- [Allen et Perrault, 1980] J. Allen et R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15 :143–178, 1980.
- [Alur et Dill, 1994] R. Alur et D. L. Dill. A theory of timed automata. *Theoretical computer science*, 126 :183–235, 1994.
- [Austin, 1962] J.L. Austin. *How to Do Things with Words*. Oxford University Press, Oxford, 1962.
- [Balkanski et Hurault-Plantet, 2000] C. Balkanski et M. Hurault-Plantet. Cooperative requests and replies in a collaborative dialogue model. *International Journal of Human-Computer Studies*, 53(6) :915–968, 2000.
- [Barbuceanu et Fox, 1995] M. Barbuceanu et M.S. Fox. COOL : a language for describing coordination in multiagent systems. Dans *Actes d'International Conference on Multi-Agent Systems (ICMAS'95)*, pages 17–24, San Francisco, USA, 1995.
- [Bastien *et al.*, 1991] C. Bastien, J.-P. Caverni, P. Mendelsohn, et G. Tiberghien. *Psychologie Cognitive : Modèles et Méthodes*. PUG, 1991.
- [Bouzouba et Moulin, 2001] K. Bouzouba et B. Moulin. Les interactions distribuées : Approche basée sur le point de vue personnel des agents. Dans *Actes des Modèles Formels de l'Interaction (MFI'01)*, pages 15–30, Toulouse, France, 2001.
- [Brassac et Pesty, 1999] C. Brassac et S. Pesty. Simuler la conversation : un défi pour les systèmes multiagents. Dans *Analyse et simulation de conversations. De la théorie des actes de discours aux systèmes multiagents*, chapitre 9, pages 317–345. L'interdisciplinaire, 1999.
- [Bratman *et al.*, 1988] M.E. Bratman, D.J. Israel, et M.E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4 :349–355, 1988.
- [Bratman, 1987] M.E. Bratman. *Intention, plans and practical reason*. Harvard University Press, 1987.
- [Chaib-Draa, 1988] B. Chaib-Draa. Interaction between agents in routine, familiar and unfamiliar situation. *Cooperative Information Systems*, 5(1) :1–25, 1988.
- [Chaignaud *et al.*, 2000] N. Chaignaud, I. Cheikhrouhou, et A. Nguyen-Xuan. Generalisation of a cognitive model for problem solving in incomplete information. Dans *Actes d'International Conference on Cognitive Modeling (ICCM'00)*, pages 58–69, 2000.

- [Chaignaud et Levy, 1996] N. Chaignaud et F. Levy. Common-sense reasoning : experiments and implementation. Dans *Actes d'European Conference on Artificial Intelligence (ECAI'96)*, pages 604–608, 1996.
- [Chaignaud, 1996] N. Chaignaud. *Étude cognitive et informatique de la résolution d'un problème : analyse, modélisation et implantation*. Thèse de doctorat, Université Paris Nord, 1996.
- [Cohen et Levesque, 1990a] P. R. Cohen et H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.
- [Cohen et Levesque, 1990b] P. R. Cohen et H. J. Levesque. Performatives in a rationally based speech acts theory. Dans *Actes du Meeting of The Association for Computational Linguistic*, pages 79–88, Pittsburgh, Pensylvanie, 1990.
- [Cohen et Levesque, 1995] P.R. Cohen et H.J. Levesque. Communicative actions for artificial agents. Dans *Actes d'International Conference on Multi-Agent Systems (ICMAS'95)*, pages 65–72, San Francisco, USA, 1995.
- [Dignum et Weigand, 1995] F. Dignum et H. Weigand. Modelling communication between cooperative systems. Dans *Actes de Conference on Advanced Information Systems Engineering (CAISE'95)*, pages 140–153, 1995.
- [Dignum, 2000] F. Dignum. Issues in agent communication : An introduction. *Lecture note in Artificial Intelligence*, pages 1–16, 2000.
- [d'Inverno et al., 1997] M. d'Inverno, D. Kinny, M. Luck, et M. Wooldridge. A formal specification of dmars. Note Technique 72, Australian Artificial Intelligence Institute, Carlton, Victoria, 1997.
- [Escabarajal, 1991] M.-C. Escabarajal. *À propos de la validité des modèles de simulation de processus*, pages 427–442. PUG, 1991.
- [Ferber et Drogoul, 1992] J. Ferber et A. Drogoul. Using reactive multi-agent systems in simulation and problem solving. Dans *Distributed Artificial Intelligence : Theory and Praxis*, pages 53–80. Kluwer Academic, 1992.
- [Ferber, 1995] J. Ferber. *Les Systèmes Multi-Agents : vers une intelligence collective*. Inter-Éditions, 1995.
- [Ferguson et al., 1996] G. Ferguson, J. Allen, et B. Miller. TRAINS-95 : Towards a mixed-initiative planning assistant. Dans *Actes d'Artificial Intelligence Planning Systems (AIPS'96)*, pages 70–77, Édimbourg, Écosse, 1996.
- [Ferguson et Allen, 1998] G. Ferguson et J. Allen. TRIPS : An intelligent integrated problem-solving assistant. Dans *Actes d'Autonomous Agents and Artificial Intelligence (AAAI-98)*, pages 567–573, Madison, Wisconsin, 1998.
- [Fikes et Nilsson, 1971] R.E. Fikes et N.J. Nilsson. STRIPS : a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2 :189–208, 1971.
- [Finin et al., 1994] T. W. Finin, R. Fritzson, D. McKay, et R. McEntire. KQML as an agent communication language. Dans *Actes de Conference on Information and Knowledge Management (CIKM'94)*, Gaithersburg, USA, 1994.
- [Finin et Fritzson, 1994] T.W. Finin et R. Fritzson. Specification of the KQML agent communication language plus example agent policies and architectures. Rapport technique, DARPA Knowledge Sharing Initiative External Interface Working Group, 1994.
- [Finin et Labrou, 1998] T. W. Finin et Y. Labrou. Semantics for an ACL. Dans *Agents, Architecture and Languages*, Berlin, Allemagne, 1998. Springer-Verlag.

- [Gardiner, 1989] A.H. Gardiner. *Langage et acte de langage. Aux sources de la pragmatique*. Presses universitaires de Lille, 1989.
- [Georgeff et Lansky, 1987] M.P. Georgeff et A.L. Lansky. Reactive reasoning and planning. Dans *Actes d'Artificial Intelligence (AAAI'87)*, pages 667–682, Seattle, USA, 1987.
- [Gonzalez, 1991] M. Gonzalez. *Sur les pratiques de validation de modèles en psychologie cognitive*, pages 443–464. PUG, 1991.
- [Grice, 1975] H. P. Grice. Logic and conversation. Dans *Syntax and Semantics : Speech acts*, volume 3, chapitre Logic and Conversation. Academic Press, 1975.
- [Grosz et Sidner, 1986] B. J. Grosz et C. L. Sidner. Attention, intentions and the structure of discourse. Dans *Computational Linguistics*, volume 12, pages 175–204. MIT Press, 1986.
- [Guerra-Hernandez, 2003] A. Guerra-Hernandez. *Apprentissage d'agents rationnels BDI dans un univers Multi-Agents*. Thèse de doctorat, Université de Paris13, 2003.
- [Hamblin, 1970] C. Hamblin. *Fallacies*. Methuen, 1970.
- [Howden *et al.*, 2001] N. Howden, R. Rönquist, A. Hodgson, et A. Lucas. JACK intelligent agents - summary of an agent infrastructure. Dans *Workshop Infrastructure for Agents, MAS, and Scalable MAS*, 2001. <http://www.agent-software.com>.
- [Kayser, 1992] D. Kayser. Intelligence artificielle et modélisation cognitive : objectifs et évaluation. Dans *Intellectica*, 13-14, pages 223–240. 1992.
- [Kerbrat-Orechioni, 1990] C. Kerbrat-Orechioni. *Les interactions verbales*. Armand Colin, 1990.
- [Koning et Pesty, 2001] J.-L. Koning et S. Pesty. Chapitre 3 - modèles de communication. Dans *Principes et architecture des systèmes multi-agents*, éditeurs J.-P. Briot et Y. Demazeau, pages 109–138. Hermès, 2001.
- [Labrou *et al.*, 1999] Y. Labrou, T. Finin, et Y. Peng. Agent communication languages : The current landscape. *IEEE Intelligent Systems*, 14(2) :45–52, 1999.
- [Labrou et Finin, 1994] Y. Labrou et T.W. Finin. A semantics approach for KQML - a general purpose communication language for software agents. Dans *Conférence on Information and Knowledge (CIKM'94)*, pages 447–455, Gaithersburg, 1994.
- [Labrou et Finin, 1997] Y. Labrou et T.W. Finin. A proposal for a new KQML specification. Rapport technique cs-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, Maryland, 1997. Available at <http://www.cs.umbc.edu/kqml/papers>.
- [Labrou, 1996] Y. Labrou. *Semantics for an Agent Communication Language*. Thèse de doctorat, Université de Maryland, 1996.
- [Lehuen, 1997] J. Lehuen. *Un modèle de dialogue dynamique et générique intégrant l'acquisition de sa compétence linguistique - Le système COALA*. Thèse de doctorat, Université de Caen, 1997.
- [Lemeunier, 2000] T. Lemeunier. *L'intentionnalité communicative dans le dialogue homme-machine en langue naturelle*. Thèse de doctorat, Université du Maine, 2000.
- [Luzzati, 1995] D. Luzzati. *Le dialogue verbal homme-machine*. Masson, 1995.
- [Maudet, 2001] N. Maudet. *Modéliser les conventions des interactions langagières : la contribution des jeux de dialogue*. Thèse de doctorat, Université Paul Sabatier - Toulouse III, 2001.

- [Mazouzi, 2001] H. Mazouzi. *Ingénierie des protocoles d'interaction : des systèmes distribués aux systèmes multi-agents*. Thèse de doctorat, Université Paris IX - Dauphine, 2001.
- [McCarthy et Hayes, 1969] J. McCarthy et P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. Dans *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.
- [Meyer, 1988] J.-J.C. Meyer. A different approach to deontic logic : deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1) :109–136, 1988.
- [Moeschler et Reboul, 1994] J. Moeschler et A. Reboul. *Dictionnaire encyclopédique de pragmatique*, chapitre 1 : Théorie des actes de langage, pages 43–78. Seuil, 1994.
- [Moeschler, 1989] J. Moeschler. *Modélisation du dialogue*. Hermès, 1989.
- [Nicolle et Almeida, 1999] A. Nicolle et V. Saint-Dizier De Almeida. Vers un modèle des interactions langagières. Dans *Analyse et simulation de conversations : De la théorie des actes de discours aux systèmes multi-agents*, chapitre 4, pages 133–169. L'Interdisciplinaire, 1999.
- [Odell et al., 2000] J. Odell, H. Van Dyke Parunak, et B. Bauer. Representing agent interaction protocols in UML. Dans *AOSE*, pages 121–140, 2000.
- [Parunak, 1996] H. Van Dyke Parunak. Visualizing agent conversations : using enhanced doo-ley graphs for agent design and analysis. Dans *Actes d'International Conference on Multi-Agent Systems (ICMAS'96)*, pages 275–282, 1996.
- [Pasquier, 2004] P. Pasquier. Modèles des dialogues entre agents cognitifs : un état de l'art. Dans *Cognito - Cahiers Romains de Sciences Cognitives*, volume 1.4, pages 79–138. Cognito - ISSN 1267-8015, 2004.
- [Pasquier, 2005] P. Pasquier. *Aspects cognitifs des dialogues entre agents artificiels : l'approche par la cohérence cognitive*. Thèse de doctorat, Université Laval, Canada, 2005.
- [Pollack, 1990] M. E. Pollack. Plans as complex mental attitudes. Dans *Intentions in Communication*, éditeurs P.R. Cohen, J. Morgan, et M. E. Pollack, pages 77–104. MIT Press, 1990.
- [Populaire et al., 1993] P. Populaire, Y. Demazeau, O. Boissier, et J. Sichman. Description et implémentation de protocoles de communication en univers multiagent. Dans *Actes des Journées Francophones de l'Intelligence Artificielle Distribuée et des Systèmes multi-agents (JFIAD SMA'93)*, pages 242–252, Toulouse, France, 1993.
- [Rao et Georgeff, 1995] A. S. Rao et M. Georgeff. BDI agents : from theory to practice. Dans *Actes d'International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, San Francisco, USA, 1995.
- [Reinach, 1983] A. Reinach. *The A priori foundations of the civil law*, pages 1–142. Aletheia 3, 1983.
- [Roulet et al., 1985] E. Roulet, A. Auchlin, J. Moeschler, C. Rubattel, et M. Schelling. *L'articulation du discours en français*. Peter Lang, 1985.
- [Sabah et al., 1997] G. Sabah, J. Vivier, A. Vilnat, J.-M. Pierrel, L. Romary, et A. Nicolle. *Machine, langage et dialogue*. L'Harmattan, 1997.
- [Sadek et al., 1997] D. Sadek, P. Brétier, et F. Panaget. ARTIMIS : Natural dialogue meets rational agency. Dans *Actes d'International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1030–1035, Yokohama, Japon, 1997.

- [Sadek, 1991] D. Sadek. Dialogue acts are rational plans. Dans *Actes d'ESCA/ETRW Workshop on the Structure of Multimodal Dialogue*, pages 1–29, Maratea, Italie, 1991.
- [Sadek, 1992] D. Sadek. A study in the logic of intention. Dans *Actes de Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 462–473, Cambridge, Massachusetts, 1992.
- [Searle et Vanderveken, 1985] J.R. Searle et D. Vanderveken. *Foundations of illocutionary logic*. Cambridge University Press, 1985.
- [Searle, 1969] J.R. Searle. *Speech Acts – An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.
- [Searle, 1980] J.R. Searle. Minds, brains, and programs. Dans *Behavioral and brain sciences*, volume 3, pages 417–424. Cambridge university press, 1980.
- [Singh, 1991] M. P. Singh. Towards a formal theory of communication for multiagent systems. Dans *Actes d'International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 69–74, Sidney, Australie, 1991.
- [Singh, 1992] M. P. Singh. A critical examination of the cohen-levesque theory of intention. Dans *Actes d'European Conference on Artificial Intelligence (ECAI'92)*, 1992.
- [Singh, 1993] M. P. Singh. A semantics for speech acts. *Annals of Mathematics and Artificial Intelligence*, 8(I-II) :47–71, 1993.
- [Sperber et Wilson, 1989] D. Sperber et D. Wilson. *La pertinence. Communication et cognition*. Minuit, 1989.
- [Sycara, 1998] K.P. Sycara. The many faces of agents. *AI Magazine*, 19(2) :11–12, 1998.
- [FIPA, 2000a] FIPA. Communicative act library specification. Rapport technique, Foundation for Intelligent Physical Agents, 2000. Available at <http://www.fipa.org>.
- [FIPA, 2000b] FIPA. KIF content language specification. Rapport technique, Foundation for Intelligent Physical Agents, 2000. Available at <http://www.fipa.org>.
- [FIPA, 2002a] FIPA. FIPA request interaction protocol specification. Rapport technique, Foundation for Intelligent Physical Agents, 2002. Available at <http://www.fipa.org>.
- [FIPA, 2002b] FIPA. SL content language specification. Rapport technique, Foundation for Intelligent Physical Agents, 2002. Available at <http://www.fipa.org>.
- [Traum, 1994] D.R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. Thèse de doctorat, Université de Rochester, 1994.
- [Trognon et Brassac, 1992] A. Trognon et C. Brassac. L'enchaînement conversationnel. Dans *Cahier de linguistique française*, volume 13, pages 76–116. Cazaubon, 1992.
- [Trognon, 1995] A. Trognon. Structures interlocutoires. Dans *Cahier de linguistique française*, volume 17, pages 79–98. Cazaubon, 1995.
- [Turing, 1950] A.M. Turing. Computing machinery and intelligence. Dans *Mind*, volume 59, pages 433–460. 1950.
- [Vanderveken, 1988] D. Vanderveken. *Les actes de discours. Essai de philosophie du langage et de l'esprit sur la signification des énonciations*. Pierre Mardaga, 1988.
- [Vanderveken, 1990 1991] D. Vanderveken. *Meaning and Speech Acts*, volume 2 : Formal Semantics of Success and Satisfaction. Cambridge University Press, 1990-1991.
- [Vanderveken, 1999] D. Vanderveken. La structure logique des dialogues intelligents. Dans *Analyse et simulation de conversations. De la théorie des actes de discours aux systèmes multi-agents*, chapitre 2, pages 61–100. L'interdisciplinaire, 1999.

- [Walton et Krabbe, 1995] D. Walton et E. Krabbe. *Commitment in Dialogue*. Suny Press, 1995.
- [Winograd et Flores, 1986] T. Winograd et F. Flores. *Understanding Computers and Cognition : A New Foundation for Design*. Ablex, 1986.