Conception d'un environnement de développement pour la résolution de problèmes : apport de l'intelligence artificielle distribuée et application à la vision

Olivier Baujard

THESE

présentée par

Olivier BAUJARD

pour obtenir le titre de DOCTEUR
de l'Université JOSEPH FOURIER - GRENOBLE I

(Arrêté ministériel du 5 juillet 1984)

Spécialité Informatique

Conception d'un Environnement de Développement pour la Résolution de Problèmes :

Apport de l'Intelligence Artificielle Distribuée et Application à la Vision

Thèse soutenue le 12 octobre 1992 devant la commission d'examen

Composition du jury :

Jean-Paul HATON	Président
Yves DEMAZEAU	Rapporteurs
Gérard GIRAUDON	
Jean-Marc CHASSERY	Examinateurs
Yves CHIARAMELLA	
Didier ADELH	
Catherine GARBAY	

Thèse préparée au sein du groupe SIC

Equipe de Reconnaissance de Formes
et de Microscopie Quantitative
Laboratoire TIM3 - USR B 00690 -
Institut IMAG

Obscure fût ma naissance et terriblement lointaine ...

Percy Bysshe SHELLEY

Remerciements

Mes remerciements iront tout d'abord à *Catherine Garbay*, Chargée de Recherche CNRS au laboratoire TIM3, qui m'a encadré tout au long de cette thèse et qui par son dynamisme, son esprit d'ouverture, l'ambiance qu'elle fait règner au sein de son groupe, a su me faire partager son goût pour la recherche.

Je voudrais également exprimer ici ma reconnaissance aux membres du jury :

Jean-Paul Haton, Professeur à l'Université de Nancy I, qui me fait l'honneur de présider ce jury.

Yves Demazeau, Chargé de Recherche CNRS au LIFIA, pour son aide précieuse lors de la rédaction de ce document et *Gérard Giraudon*, Directeur de Recherche INRIA à Sophia-Antipolis, pour l'intérêt qu'il a montré à la lecture de mes travaux. Je les remercie vivement d'avoir accepté d'être les rapporteurs de cette thèse.

Jean-Marc Chassery, Directeur de Recherche CNRS au laboratoire TIM3, pour avoir trouvé le temps, entre deux explosions de moteur, d'examiner cette thèse et de faire partie de ce jury.

Yves Chiaramella, Professeur à l'Université Joseph Fourier, pour l'honneur qu'il me fait en acceptant de faire partie des examinateurs.

Didier Adelh, Directeur de Projet, pour ses remarques toujours judicieuses, et à travers lui la société Alcatel TITN-ANSWARE, pour la grande liberté qui m'a été accordée lors de la réalisation de ce projet de thèse.

Je tiens également à exprimer ma sympathie à toute l'équipe RFMQ :

A l'ensemble du groupe SIC, les compagnons et amis des débuts *Emmanuelle*, *Arturo* et *René-Pierre* pour les discussions passionnées et passionnantes que nous avons sur tous les sujets et pour leur soutien tout au long de ces années, et également *Sylvie*, *Marc* et *Adel*. Merci aussi à tous les stagiaires qui se sont succédés au sein du groupe et qui ont contribué à cette thèse : *Gilles*, *Benoit* et *Etienne*.

Aux anciens : *Franck* camarade de lutte de DEUG, de Licence, de Maîtrise, de DEA, de thèse et qui a fini par échouer sous les bananiers ivoiriens, *Martial* grand amateur d'Adelscot qui fidèle à la philosophie GBM a su s'intéresser à mes travaux et m'intéresser aux siens, *Marisol* pour nos discussions matinales sur nos thèses respectives, *Isabelle* ma partenaire préférée de "lambada", et aux nouvelles : *Christèle* biologiste de caractère mais néanmoins amie, *Paola* qui me pardonnera de souligner la nouvelle défaite du Milan A.C et *Magali (e!)*.

A mes autres compagnons de thèse : le groupe Voronoï (*Raphaël* (Clio RN 1.4), *Etienne* (Clio RN 1.2)), les militaires (*Bruno*, *Philippe* (la quille !) et *Franck* (plus que 11 mois !)), *Pascal* (mais non ! tes blagues ne sont pas toutes mauvaises), *Catherine*, *Edouard*, *Michel* et mes ex-compagnons de bureau (*Jules* (à quand la thèse ?), *Mahmoud* et *Hai-Tao*).

Une pensée pour *Guy*, qui malgré la croissance exponentielle du parc de machines (ah le bon vieux temps où l'on se battait pour **le** dn660 ou **le** dn3000 !), sait garder son calme et sa gentillesse pour répondre aux problèmes de tous.

Et enfin aux responsables : *Gérard.B* (directeur énergique), *Camille*, *Dominique*, *Annick* (je n'oublierais pas la (double) bise pour le pot), *Françoise*, *Yves* (Silicon Iris Indigo) et *Xavier* (un fan d'Arsan) et aux membres du personnel : *Paulette*, *Nicole* (il y a du courrier pour moi ?), *Gérard.C*, *Michèle*, *Victoria*, *Jocelyne*, *Marie-Paule* (Clio S) et *Jean-François* (amateur de moules frites).

Tous, malgré le stress inhérent à un laboratoire de recherche, s'efforcent de faire règner une ambiance amicale.

Enfin, à ma famille pour son soutien indéfectible et sans laquelle cette thèse n'aurait pas été possible.

SOMMAIRE

INTRODUCTION THESE 1
INTRODUCTION PARTIE A 7

CHAPITRE 1. ETAT DE L'ART 9

I INTRODUCTION 9
II LANGAGES A OBJETS 10
1. LANGAGES DE FRAMES 10
2. LANGAGES D'ACTEURS 11
3. LANGAGES HYBRIDES 13

III SYSTEMES MULTI-AGENTS COGNITIFS 13

1. MODELES D'AGENTS COGNITIFS 14
1.1. EXPERTISE DU DOMAINE 14
1.2. CONNAISSANCES DE SOI ET DES AUTRES 15
1.3. COMMUNICATION ET PROTOCOLES 18
1.4. CONTROLE 20

2. SOCIETES COGNITIVES 21
2.1. DISTRIBUTION 21
2.2. COMMUNICATION 22
2.3. COOPERATION 24

IV CONCLUSION 27

CHAPITRE 2. MOTIVATIONS 29

I INTRODUCTION 29
II MODELE D'AGENT 30
III SOCIETE 31
IV LANGAGE 32
V CONCLUSION 32

CHAPITRE 3. MAPS 35

I INTRODUCTION 35
II ARCHITECTURE LOGICIELLE 35

1. LES AGENTS MAPS 36

1.1. LES RESSOURCES 36
1.1.1. AGENT KS 37
1.1.2. AGENT KP 38

1.2. LES CONNAISSANCES SUR SOI ET SUR LES AUTRES 39
1.2.1. AGENT KS 40
1.2.2. AGENT KP 40

1.3. LES COMMUNICATIONS 41

1.3.1. AGENT KS 41
1.3.2. AGENT KP 43

1.4. LE CONTROLE 43
1.4.1. AGENT KS 44
1.4.2. AGENT KP 47

2. LA SOCIETE D'AGENTS 50
2.1. DISTRIBUTION DES TACHES ET DES COMPETENCES 50
2.2. MODES ET PROTOCOLES DE COMMUNICATION 52
2.3. MODES D'ORGANISATION ET DE COOPERATION 52

III LANGAGE DE PROGRAMMATION 53

1. SYNTAXE DU LANGAGE 54
1.1. LES AGENTS 54
1.2. LES OBJETS 55
1.3. LES ATTRIBUTS 55
1.4. LES CONNEXIONS 55
1.5. LES REGLES 55

2. EXEMPLE D'UTILISATION 57
2.1. PRESENTATION DE L'APPLICATION 57
2.2. LES AGENTS KS 58
2.3. L'AGENT KP 64

3. TECHNIQUES AVANCEES DE PROGRAMMATION 69

IV IMPLANTATION 72
1. LES AGENTS 72
2. LE CONTROLEUR D'APPLICATION 74
3. L'INTERFACE 74
4. LE SERVEUR DE PROCEDURES 76

V CONCLUSION 78

CONCLUSION PARTIE A 79

REFERENCES PARTIE A 81

INTRODUCTION PARTIE B 89

CHAPITRE 1. ETAT DE L'ART 91

I INTRODUCTION 91
II OUTILS DE LA VISION 91

1. INFORMATIONS MANIPULEES 92
2. OUTILS ET TECHNIQUES 93

2.1. OUTILS D'EXTRACTION DE PRIMITIVES 93
2.1.1. OUTILS D'EXTRACTION DES PRIMITIVES REGION 93
2.1.2. OUTILS D'EXTRACTION DES PRIMITIVES CONTOURS 94

2.2. TECHNIQUES DE MANIPULATION 95
2.2.1. FERMETURE DES CONTOURS 96
2.2.2. DECOMPOSITION GEOMETRIQUE 96

2.3. TECHNIQUES D'INTERPRETATION D'IMAGES 97
2.3.1. TECHNIQUES DE MISE EN CORRESPONDANCE 98
2.3.2. TECHNIQUES D'INFERENCE 100
2.3.3. TECHNIQUES ADAPTATIVES 100

3. DISCUSSION 101

III DES OUTILS AUX SYSTEMES DE VISION 101

1. POINT DE VUE ALGORITHMIQUE 102

1.1. COOPERATION 102
1.1.1. REPRESENTATIONS MULTI-NIVEAUX 102
1.1.2. COOPERATION MULTI-NIVEAUX 103
1.1.3. COOPERATION REGION / CONTOUR 104

1.2. FOCALISATION, ADAPTATION ET CONTROLE 105
1.2.1. FOCALISATION 105
1.2.2. ADAPTATION 106
1.2.3. CONTROLE 107

2. POINT DE VUE LOGICIEL 107

2.1. CONNAISSANCES ET REPRESENTATION DES CONNAISSANCES 108
2.1.1. CONNAISSANCES MANIPULEES 108
2.1.2. REPRESENTATION DES CONNAISSANCES 109

2.2. CONTROLE 111
2.2.1. CONTROLE GLOBAL 112
2.2.2. CONTROLE LOCAL 112

3. DISCUSSION 113

IV CONCLUSION 113

CHAPITRE 2. MOTIVATIONS 117

I INTRODUCTION 117
II CONNAISSANCES ET REPRESENTATION DES CONNAISSANCES 118
III CONTROLE 118
1. FOCALISATION ET ADAPTATION 120
2. COOPERATION 121

IV CONCLUSION 121

CHAPITRE 3. KISS 123

I INTRODUCTION 123
II ANALYSE BAS NIVEAU 125

1. FILIERE DE DETECTION DES REGIONS 127
1.1. OBJECTIFS 128
1.2. DESCRIPTION DE LA FILIERE 128
1.3. DISCUSSION 129

2. FILIERE DE DETECTION DES CONTOURS 130
2.1. OBJECTIFS 130
2.2. DESCRIPTION DE LA FILIERE 130
2.3. DISCUSSION 133

3. DISCUSSION 133

III ANALYSE INTERMEDIAIRE 134

1. FILIERE D'INTERPRETATION GEOMETRIQUE 135
1.1. OBJECTIFS 135
1.2. DESCRIPTION DE LA FILIERE 135
1.3. DISCUSSION 136

2. FILIERE D'INTERPRETATION RELATIONNELLE 136
2.1. OBJECTIFS 137
2.2. DESCRIPTION DE LA FILIERE 137
2.3. DISCUSSION 140

3. DISCUSSION 140

IV ANALYSE HAUT NIVEAU 142

1. FILIERE D'INTERPRETATION SEMANTIQUE 144
1.1. OBJECTIFS 144
1.2. DESCRIPTION DE LA FILIERE 144
1.3. DISCUSSION 147

2. FILIERE DE MANIPULATION 147
2.1. OBJECTIFS 147
2.2. DESCRIPTION DE LA FILIERE 147
2.3. DISCUSSION 149

3. DISCUSSION 150

V RESULTATS ET DISCUSSION 150

1. ILLUSTRATION DES ETAPES D'ANALYSE 151
1.1. ANALYSE BAS NIVEAU 151
1.2. ANALYSE INTERMEDIAIRE 153
1.3. ANALYSE HAUT NIVEAU 153

2. RESULTATS ET DISCUSSION 155
2.1. SYSTEME UTILISANT L'ALGORITHME DE FISHER 156
2.2. SEGMENTATION UTILISANT UN ALGORITHME PYRAMIDAL 158

3. IMPLANTATION 161

VI CONCLUSION 163

CONCLUSION PARTIE B 165

REFERENCES PARTIE B 167

INTRODUCTION PARTIE C 177

CHAPITRE 1. APPLICATIONS 179

I INTRODUCTION 179
II LE SYSTEME KIDS 179
1. ARCHITECTURE 179
2. DISCUSSION 181

III LE SYSTEME DE COMPREHENSION DE LA PAROLE 182
1. ARCHITECTURE 182
2. DISCUSSION 184

IV CONCLUSION 185

CHAPITRE 2. ETAT DE L'ART 187

I INTRODUCTION 187
II ENVIRONNEMENTS DE PROGRAMMATION 187

1. LANGAGES DE PROGRAMMATION 187
1.1. EXTENSIONS DE LANGAGES EXISTANTS 187
1.2. LANGAGES SPECIFIQUES 188

2. INTERACTION HOMME-MACHINE 189
3. FONCTIONNALITES ET STYLE D'IMPLANTATION DES PLATE-FORMES 190

III CONCLUSION 193

CHAPITRE 3. MAPS REPARTI 197

I INTRODUCTION 197
II ARCHITECTURE LOGICIELLE 197

1. LES AGENTS MAPS 197

1.1. RESSOURCES 198
1.2. LES CONNAISSANCES SUR SOI ET LES AUTRES 198

1.3. LES COMMUNICATIONS 199
1.3.1. AGENT KS 200
1.3.2. AGENT KP 200

1.4. LE CONTROLE 200
1.4.1. AGENT KS 201
1.4.2. AGENT KP 202

2. LA SOCIETE D'AGENTS 202

2.1. DISTRIBUTION DES TACHES ET DES COMPETENCES 202

2.2. MODES ET PROTOCOLES DE COMMUNICATION.....203
2.3. MODES D'ORGANISATION ET DE COOPERATION.....203

III LANGAGE DE PROGRAMMATION.....204

1. SYNTAXE DU LANGAGE.....204

1.1. LES AGENTS.....204
1.2. LES OBJETS.....205
1.3. LES ATTRIBUTS.....205
1.4. LES CONNEXIONS.....205
1.5. LES REGLES.....205

2. EXEMPLE D'UTILISATION.....210
3. TECHNIQUES AVANCEES DE PROGRAMMATION.....211

IV IMPLANTATION.....214

1. LES AGENTS.....214
2. LE CONTROLEUR D'APPLICATION.....216
3. L'INTERFACE.....217
4. LE SERVEUR DE PROCEDURES.....220

V CONCLUSION.....221

CHAPITRE 4. PERSPECTIVES.....223

I INTRODUCTION.....223
II ETAT DE L'ART.....223

1. MODELES REACTIFS.....223

1.1. REPRESENTATION DE L'AGENT.....224
1.2. L'AGENT ET SON ENVIRONNEMENT.....224
1.3. COMMUNICATION ET PROTOCOLES.....225
1.4. COMPORTEMENT.....226
1.5. DISCUSSION.....227

2. ARCHITECTURES MULTI-COUCHES HETEROGENES.....227

2.1. ARCHITECTURES HETEROGENES.....227
2.2. ARCHITECTURES MULTI-COUCHES.....228
2.3. DISCUSSION.....230

III AMELIORATIONS STRUCTURELLES.....231

1. AGENTS REACTIFS.....231
2. HIERARCHIE MULTI-COUCHES.....233

IV CONCLUSION.....235

CONCLUSION PARTIE C.....237

REFERENCES PARTIE C.....239

CONCLUSION THESE 241

REFERENCES THESE 245

INTRODUCTION THESE

Objectif

L'objectif de cette thèse est la conception d'un environnement de développement pour la résolution de problèmes. Les domaines d'application privilégiés d'un tel environnement seront des domaines complexes tels que la Vision par Ordinateur, le Diagnostic Biomédical ou la Compréhension de la Parole. Ces domaines ont un objectif commun qui concerne l'interprétation symbolique de signaux physiques (image, parole). La résolution implique ici la transformation, le passage de l'information à travers des niveaux de représentation de plus en plus abstraits et selon des points de vue différents. Les systèmes que nous souhaitons créer vont ainsi être complexes, devant représenter et manipuler des connaissances et des tâches hétérogènes, utiliser une approche adéquate pour faire coopérer ces connaissances et ces tâches, et introduire à cette fin un contrôle adapté. Les maîtres mots de tels systèmes sont donc hétérogénéité, intégration, coopération et sélection adaptée. Ces caractéristiques nous ont conduit à étudier les langages à objets et l'Intelligence Artificielle Distribuée, plus particulièrement les systèmes multi-agents dans leur approche coopérative.

Nous avons voulu intégrer à la fois les approches orientées objets et multi-agents, car si l'objet permet l'encapsulation et donc la manipulation d'informations et de tâches à un niveau d'abstraction plus élevé, l'agent apporte un niveau supplémentaire en introduisant l'autonomie et autorisant la distribution du contrôle.

Un environnement de développement MAPS (Multi-Agent Problem Solver) a été développé qui nous a permis de concevoir des applications dans plusieurs domaines. Des critiques ont alors pu être dégagées par ces applications et par une comparaison de MAPS avec des plate-formes en Intelligence Artificielle Distribuée. Ces critiques nous ont amené à définir des améliorations à apporter.

Ces améliorations nous ont alors permis de développer un nouvel environnement, plus proche d'une plate-forme en Intelligence Artificielle Distribuée, seconde étape d'un développement devant nous conduire à la création d'une plate-forme de développement hétérogène et multi-couches. Ces étapes de conception sont résumées dans la figure **1**.

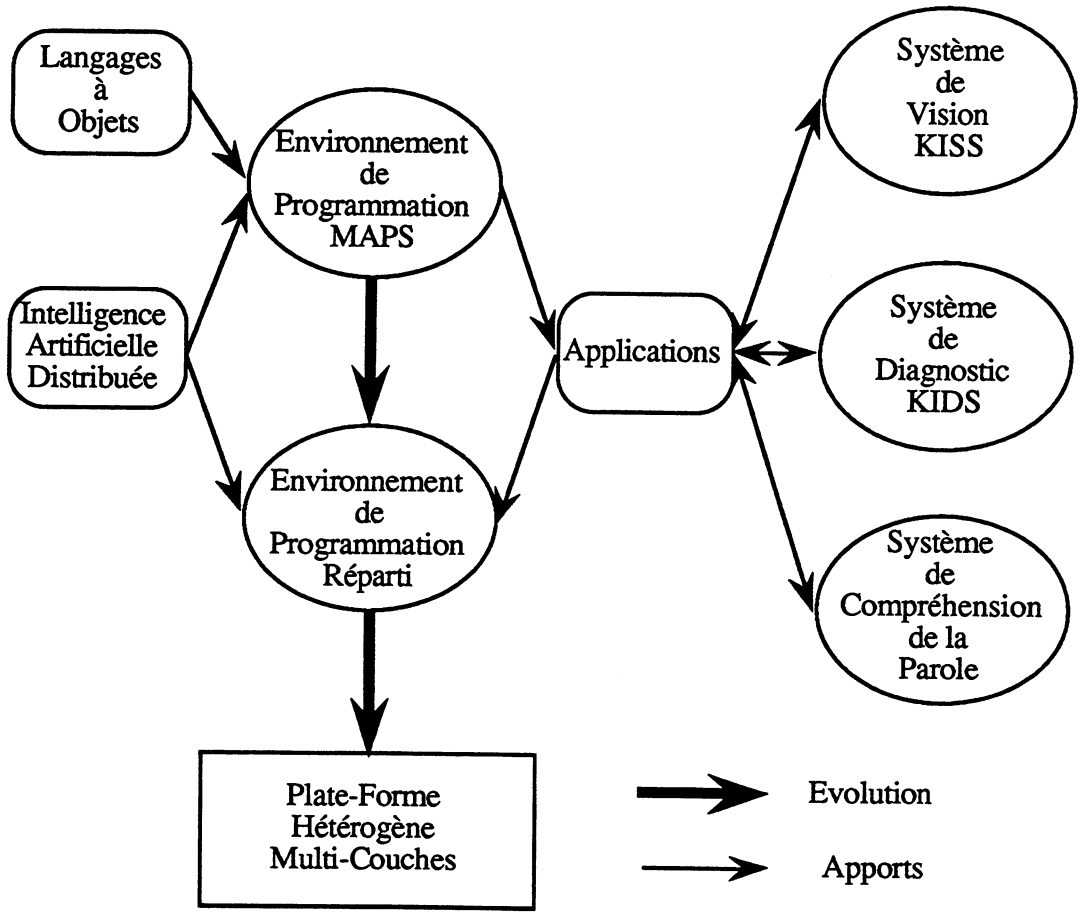


Fig. 1. Evolution de l'environnement de programmation MAPS

Concepts de Base

Notre souci a été d'intégrer des connaissances sous la forme d'objets et d'introduire des raisonnements sur ces objets à l'aide de règles. Il nous a également paru nécessaire d'introduire le concept d'autonomie issu de l'Intelligence Artificielle Distribuée, en distribuant ces objets et ces règles au sein d'entités autonomes. Nous avons finalement voulu conserver la distinction entre des connaissances figuratives et des connaissances opératoires, introduite par nos applications. Deux classes génériques d'agents, de type système expert ont été définies dans ce but. Des agents dédiés à la modélisation et à l'exploitation des connaissances figuratives (agents KS) et des agents dédiés à la modélisation et à l'exploitation des connaissances opératoires (agents KP). Cette distinction permet en outre de distribuer le contrôle propre à chaque type de connaissance au sein de ces agents.

Un langage de programmation hybride, issu de plusieurs formalismes a été introduit pour exploiter ces notions architecturales à un niveau d'abstraction élevé. Dans un souci d'efficacité (en terme de temps d'exécution), nous avons choisi de développer ce langage comme une couche logicielle au dessus du langage C++, fermant ce langage sur des concepts simples. A ce

langage ont également été associés des outils d'aide à la conception afin d'obtenir un environnement de développement performant, permettant le développement d'applications dans divers domaines.

Applications

Cet environnement nous a permis de développer plusieurs applications en Vision par Ordinateur avec le système KISS (Knowledge-based Image Segmentation System), en Diagnostic Biomédical avec le système KIDS (Knowledge-based Image Diagnosis System), en Compréhension de la Parole et en Acquisition des Connaissances, et de valider ainsi nos concepts de base mais aussi de souligner leurs limitations. La conception de ces applications a grandement contribué à la conception de l'environnement MAPS : nous sommes en effet persuadés que la conception d'un environnement de programmation performant repose sur un développement conjoint des concepts architecturaux et des applications. Il convient de souligner ici que la conception des applications de Compréhension de la Parole, de Diagnostic Biomédical et les développements en Acquisition des Connaissances ne rentrent pas dans le cadre de cette thèse.

Evolution

L'analyse critique des applications et une étude approfondie de l'état de l'art des plate-formes de développement en Intelligence Artificielle Distribuée nous ont conduit à définir un certain nombre de besoins d'ordre fonctionnel et d'ordre structurel. Les améliorations fonctionnelles, impliquant l'introduction d'un mode d'exécution parallèle et l'accroissement de l'autonomie des agents ont été intégrées dans un nouvel environnement de développement (MAPS Réparti), pour lequel un nouveau langage de programmation plus proche du langage cible C++ a été défini, garantissant ainsi une meilleure ouverture aux autres langages. Les améliorations structurelles, impliquant la définition d'une architecture hétérogène multi-couches sont quant à elles d'un niveau plus prospectif, bien que connaissant de premiers développements.

Plan de la Thèse

L'environnement de programmation MAPS est présenté dans la partie A, composée de trois chapitres. Le premier chapitre est un état de l'art sur les langages à objets et sur l'approche multi-agents dans un cadre cognitif. En ce qui concerne l'approche multi-agents, l'accent est mis sur l'agent et la société d'agents à un niveau structurel. Cet état de l'art nous pemet de définir des choix qui sont présentés dans un second chapitre. L'implantation de l'environnement MAPS est finalement présentée dans un troisième chapitre.

Le système de Vision par Ordinateur KISS est présenté dans la partie B, composée de trois chapitres. Le premier chapitre est un état de l'art sur la vision. L'accent est mis sur la diversité des outils et des techniques, et sur la complexification algorithmique et logicielle des systèmes de segmentation et de vision. Nos choix quant à la conception du système sont présentés dans le second chapitre, le système KISS étant finalement décrit dans le troisième chapitre.

La partie C, composée de quatre chapitres, présente l'évolution de l'environnement MAPS vers une plate-forme de développement en Intelligence Artificielle Distribuée. Cette évolution est nécessaire, conséquence des besoins des applications de vision mais également de Diagnostic Biomédical et de Compréhension de la Parole, présentées dans le premier chapitre, mais aussi des manques par rapport aux plate-formes de développement en Intelligence Artificielle Distribuée présentées dans le second chapitre. Des améliorations ont ainsi été apportées à l'environnement de programmation qui sont décrites dans le troisième chapitre. Le quatrième chapitre présente finalement les perspectives de l'environnement.

PARTIE A

L'ENVIRONNEMENT DE PROGRAMMATION MAPS

Chapitre 1 : Etat de l'Art

Chapitre 2 : Motivations

Chapitre 3 : MAPS

INTRODUCTION PARTIE A

Notre but est la conception d'un environnement de développement pour la résolution de problèmes. Les domaines d'application privilégiés d'un tel environnement seront des domaines complexes tels que la Vision par Ordinateur, le Diagnostic Biomédical ou la Compréhension de la Parole. Ces applications nécessitent la description et la manipulation de connaissances hétérogènes. Elles démontrent également le besoin de raisonnements complexes nécessitant des techniques de focalisation sur des informations et l'adaptation des traitements à ces informations ainsi que l'introduction de mécanismes de contrôle sophistiqués.

Ces besoins nous ont tout d'abord amené à étudier les langages à objets, abondament utilisés dans les approches à base de connaissances. Ces langages ont en effet été créés pour répondre aux besoins croissants de représenter des connaissances hétérogènes, de manipuler des bases de connaissances complexes ou d'exploiter de nombreux types de données. Ils introduisent des mécanismes tels que l'abstraction de données qui définissent des niveaux d'abstraction supplémentaires par rapport aux langages de programmation classiques. Ils sont également modulaires et garantissent ainsi la lisibilité des programmes et leur réutilisabilité.

Outre les problèmes d'hétérogénéité et de complexité déjà abordés en partie par les langages à objets, l'un des concepts spécifique apporté par l'Intelligence Artificielle Distribuée est la notion d'autonomie. L'agent, au contraire de l'objet peut raisonner sur le message qu'il reçoit. On assiste ainsi un dédoublement du contrôle qui s'exerce de manière interne et externe, qui nous parait indispensable en résolution coopérative de problèmes.

Une étude des langages orientés objets et de l'approche multi-agent dans un cadre cognitif est ainsi présentée dans un premier chapitre. Cet état de l'art nous a permis de dégager un certain nombre de choix, présentés dans un second chapitre, pour l'environnement de développement que nous souhaitons créer. Il nous a en particulier paru intéressant d'intégrer plusieurs concepts issus à la fois des langages à objets et des systèmes multi-agents.

Cet environnement, que nous décrivons dans un dernier chapitre, a été défini dans le souci d'intégrer des connaissances hétérogènes. Ces connaissances ont été introduites sous la forme d'objets associés à des règles d'inférence, permettant ainsi d'effectuer des raisonnements sur les objets. Cet environnement répond également au besoin d'introduire le concept d'autonomie issu de l'Intelligence Artificielle Distribuée, en distribuant ces objets et ces règles au sein d'entités

autonomes (glissement vers le concept d'acteur). Nous avons finalement voulu conserver la distinction entre des connaissances figuratives et des connaissances opératoires, introduite par nos applications.

Deux classes génériques d'agents, de type système expert ont été ainsi définies. Des agents dédiés à la modélisation et à l'exploitation des connaissances figuratives (agents KS) et des agents dédiés à la modélisation et à l'exploitation des connaissances opératoires (agents KP). Cette distinction permet en outre de distribuer le contrôle propre à chaque type de connaissance au sein de ces agents. Ce contrôle va s'exercer de manière interne, introduisant des mécanismes de focalisation sur des informations et des mécanismes d'adaptation des traitements à ces informations. Il va également s'exercer de manière externe et concerner les échanges entre les agents.

Un langage de programmation hybride, issu de plusieurs formalismes a été enfin introduit pour spécialiser ces classes génériques. Ce langage a été développé comme une couche logicielle au dessus du langage C++, fermant ce langage sur des concepts simples. A ce langage ont également été associés des outils d'aide à la conception afin d'obtenir un environnement de développement performant.

CHAPITRE 1

ETAT DE L'ART

I INTRODUCTION

Nos applications privilégiées vont appartenir aux domaines de la Vision par Ordinateur, du Diagnostic Biomédical et de la Compréhension de la Parole. Ces applications, comme toutes les applications complexes, démontrent le besoin de nouveaux outils de programmation pour pallier les limitations algorithmiques des méthodes procédurales. Elles nécessitent ainsi la description et la manipulation d'un ensemble complexe de connaissances souvent hétérogènes, la modélisation du raisonnement humain, l'adaptation des solutions procédurales à des tâches nouvelles, et une explicitation du contrôle. Pour toutes ces raisons, les approches à base de connaissances ont été abondamment employées, et ont en particulier introduit les représentations centrées objets et les langages à objets.

Les langages à objets, sous lesquels sont regroupées les représentations centrées objets, répondent en effet au besoin d'outils de coordination pour la création et l'exploitation de nombreux types de données, d'environnements de programmation perfectionnés, d'outils de prototypage rapide, d'outils de représentation de connaissances hétérogènes, d'outils de manipulation et d'exploitation de bases de connaissances et d'outils spécifiquement conçus pour la programmation parallèle **[Masini 90]**. Les principales caractéristiques de ces langages sont l'abstraction de données, la modularité, la réutilisabilité et la lisibilité. L'abstraction de données garantit l'indépendance de l'objet vis-à-vis des messages qu'il reçoit : l'objet décide de la manière de traiter un message en fonction de son implantation physique. La modularité permet de changer la définition d'un objet avec un minimum d'interaction sur les autres. La réutilisabilité permet de bâtir des bibliothèques d'objets et de les utiliser pour créer d'autres objets par spécialisation ou composition. Enfin, l'encapsulation et la modularité renforcent la lisibilité des programmes. Dans un premier temps, nous présentons ainsi les principaux langages à objets existants.

Face à l'expansion et à la croissance des technologies informatiques et à la nécessité de manipuler et de maintenir des bases de connaissances de plus en plus complexes et importantes, il parait également intéressant d'étudier les techniques plus récentes de l'Intelligence Artificielle Distribuée. Ce domaine répond en effet au besoin des applications complexes, d'une distribution de l'"intelligence" parmi plusieurs entités (ou agents) non soumises à un contrôle centralisé, dotées d'une certaine autonomie et devant être capable de planifier, d'agir et de

travailler dans un environnement commun, au prix de conflits éventuels **[Erceau 91]**. Les applications ainsi conçues introduisent une représentation multi-modale des connaissances et l'intégration de divers formalismes de représentation, de schémas de raisonnement multiples et une résolution coopérative des problèmes. Nous présentons ainsi dans un second temps l'approche multi-agents dans un cadre cognitif.

II LANGAGES À OBJETS

Une excellente étude des langages à objets peut être trouvée dans **[Masini 90]**, dont nous nous sommes inspirés pour ce paragraphe. Selon cet ouvrage, les différents langages à objets peuvent être différenciés selon qu'ils offrent uniquement des propriétés d'abstraction de données et d'encapsulation, qu'ils regroupent les objets en classes ou qu'ils définissent la notion d'héritage. Trois grandes familles de langages peuvent par ailleurs être dégagées selon qu'elles privilégient le point de vue structurel (l'objet est un type de données), le point de vue conceptuel (l'objet est le prototype d'un concept) ou le point de vue acteur (l'objet est autonome, actif et se reproduit par copie) : les langages de classes, les langages de frames et les langages d'acteurs. Des combinaisons de langages appartenant à ces familles donnent également naissance aux langages hybrides. Nous considérerons dans ce qui suit les langages de frames, les langages d'acteurs et les langages hybrides, plus proches de nos réalisations, et laisserons de côté les langages de classes plus courants et mieux connus comme Simula **[Birtwistle 73]**, C++ **[Stroustrup 86]**, Eiffel **[Meyer 86]**, Smalltalk-80 **[Goldberg 83]** ou Flavors **[Moon 86]**. Pour ces différents langages, nous donnerons les principales caractéristiques, sans toutefois trop rentrer dans le détail, notre but n'étant pas de faire une étude exhaustive de tous les langages à objets mais plutôt de poser des jalons par rapport à ce que nous souhaitons réaliser.

1. LANGAGES DE FRAMES

Les langages de frames sont les héritiers des réseaux sémantiques et des systèmes de frames de Minsky **[Minsky 75]**, issus de travaux inspirés de la psychologie cognitive. A l'origine, un frame devait constituer un formalisme de représentation général des connaissances qui puisse expliquer l'efficacité et la rapidité des activités mentales humaines, ce formalisme reposant sur la réutilisation dynamique des frames. Le but d'un frame est donc principalement de représenter les connaissances et de les manipuler.

Un frame est ainsi un représentant typique (prototype) d'une catégorie d'objets. Cependant, à la différence des langages de classes, où la classe décrit un ensemble d'objets partageant la même structure (variables d'instances) et le même comportement (méthodes), tout objet est à la fois un

représentant des frames dont il est issu et un générateur de frames plus spécialisés.

Structures et propriétés

Le frame est actuellement défini comme une structure à trois niveaux : cadre-attribut-facette. Un frame est ainsi une entité générique composée d'attributs qui décrivent les différentes propriétés du concept représenté. Les facettes expriment des modalités descriptives ou comportementales, représentant différents points de vue sur l'attribut. L'héritage étant défini, il est possible de construire des hiérarchies de frames.

Les frames n'ont pas de comportement propre et sont manipulés par des primitives (primitives de création, de destruction, d'accès aux attributs par exemple). Des procédures peuvent cependant être attachées localement aux attributs pour définir des méthodes de calcul, de gestion ou d'utilisation. Ces procédures sont des raisonnements locaux et décentralisés et se déclenchent lors des accès aux attributs (méthodes réflexes), les accès les plus courants aux attributs étant la lecture, l'écriture et la suppression. Ce mécanisme définit un style de programmation dirigé par les accès.

Enfin, le frame représentant les informations par défaut caractérisant une famille d'entités, il sert de référence pour la comparaison d'objets à analyser ou classer. Deux catégories de problèmes peuvent être traitées : le filtrage, qui consiste à rechercher parmi un ensemble de frames, ceux qui vérifient certains critères, et la classification, qui consiste à insérer un nouveau frame dans une hiérarchie ou à modifier sa position.

Pour conclure, les langages de frames sont plutôt des langages de représentation des connaissances que des langages de programmation comme les langages de classe. La programmation dirigée par les accès, issue de l'attachement procédural aux attributs (distinction figuratif / opératoire) est cependant une caractéristique intéressante de part l'opportunisme du calcul des attributs.

Différents langages de frames

Parmi les différents langages de frames, citons tout d'abord KRL **[Bobrow 77]**, FRL **[Goldstein 77]**, OWL **[Martin 77]** et UNITS **[Stefik 79]**. Ces précurseurs ont servi de modèles aux langages de représentation, parmi lesquels SRL **[Wright 84]** et Shirka **[Rechenmann 85]**.

2. LANGAGES D'ACTEURS

Les langages d'acteurs sont issus des travaux de Hewitt **[Hewitt 73]**, visant à représenter des connaissances de manière distribuée, sous la forme d'une société d'experts coopérants. Un

acteur peut être ainsi considéré comme un expert vivant en société et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué ou réparti : les connaissances et le comportement sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle, de manière indépendante.

Structures et propriétés

Chaque objet appelé acteur, est un agent actif, autonome et communicant avec les autres librement. Les acteurs évoluent dans un univers dynamique, où des activités se créent, se déroulent et s'achèvent. Un acteur regroupe au sein d'une même entité un ensemble limité de connaissances (des données locales appelées accointances, qui sont les autres acteurs que l'acteur connait directement) et le moyen de les exploiter (un comportement qui définit les actions prises en fonction des événements qui surviennent). Il peut créer un autre acteur par copie de lui-même, la copie étant identique ou différentielle (adjonction de nouvelles caractéristiques) ; ce mécanisme s'apparente à l'instanciation.

Dans un univers d'acteurs, le seul type d'événement qui peut survenir est l'envoi d'un message. Le message est un acteur (acteur message) transmis à un acteur cible par un acteur messager. Un message contient une continuation, c'est-à-dire l'acteur auquel renvoyer le résultat du message (par défaut, l'émetteur du message). Ceci permet deux types de communication : synchrone (quand la continuation est l'émetteur) et asynchrone (quand la continuation est un autre acteur). Notons l'importance de la communication asynchrone qui autorise une plus grande indépendance des acteurs : un acteur peut continuer ses propres traitements sans attendre la complétion des traitements impliqués par les messages qu'il émet.

Un acteur, comme une classe, regroupe donc des données locales et un comportement. Ils diffèrent cependant sur trois points : une classe est un type de données qui décrit des propriétés communes à un ensemble d'instances alors qu'un acteur est un prototype unique et indépendant, créé par copie différentielle d'un autre acteur. Une classe n'a pas de rôle actif mais détient le comportement de ses instances, activés par des messages synchrones, alors qu'un acteur est indépendant et activé par des messages asynchrones. Enfin, il n'existe pas de relation d'héritage pour les acteurs, mais une dépendance mutuelle avec le mécanisme de délégation.

Différents langages d'acteurs

Parmi les différents langages d'acteurs, citons les précurseurs PLANNER **[Hewitt 71]** et CONNIVER **[McDermott 72]** qui donnèrent naissance au langage PLASMA **[Hewitt 77]**. PLASMA et Smalltalk ont par la suite influencé la conception de Act 1 **[Lieberman 81]** et Act 2 **[Theriault 83]** introduisant le mécanisme de la délégation. Citons également ABCL/1

[Yonezawa 86], écrit en Lisp et facile à utiliser à la manière des langages de classes. Citons enfin Actalk, fondé sur Smalltalk et réalisé par Briot **[Briot 88]**, langage particulièrement utilisé en France, par le Laforia notamment.

3. LANGAGES HYBRIDES

Les langages hybrides sont nés des considérations pratiques de l'Intelligence Artificielle avec le besoin de langages permettant à la fois la programmation objet, la programmation logique et la programmation fonctionnelle. Aucune des grandes familles ne permet une telle programmation à elle seule. Les langages hybrides intègrent donc des caractéristiques empruntées à l'une ou l'autre des grandes familles de langages à objets (classes, frames et acteurs). Un langage hybride permet typiquement d'encapsuler des méthodes dans un frame ou d'associer des facettes aux variables d'instances d'une classe, permettant ainsi de bénéficier conjointement des avantages offerts par les classes et les frames. En conclusion, ces langages sont issus des chercheurs en Intelligence Artificielle, qui préfèrent souvent un langage adapté à leurs besoins plutôt que d'utiliser un langage existant.

Différents langages hybrides

Parmi les différents langages hybrides, citons LOOPS **[Bobrow 83]** et les langages apparentés ART **[Clayton 84]** et KEE **[Fikes 85]**. Nous trouvons également les langages développés par les équipes françaises de recherche en Intelligence Artificielle tels MERING **[Ferber 85]**, LORE **[Roche 89]** ou YAFOOL **[Ducournau 88]**. Citons enfin les langages construits au dessus des langages de programmation logique tel OBJLOG **[Duguerdil 87]**.

III SYSTEMES MULTI-AGENTS COGNITIFS

Face aux langages à objets, l'Intelligence Artificielle Distribuée apporte une nouvelle notion, la notion d'agent. Une palette étendue d'agents, de granularité différente, est introduite, allant de la simple procédure aux systèmes plus complexes de type système expert. Un agent diffère d'un objet principalement de part l'autonomie qui lui est conférée ; en cela, il est plus proche de l'acteur. Cependant, alors que les systèmes fondés sur les acteurs nécessitent une programmation d'assez bas niveau, l'expression des agents semble être de plus haut niveau. Nous nous intéressons ici, aux agents et aux sociétés d'agents cognitifs.

La caractéristique essentielle de l'agent cognitif est son autonomie de décision propre par rapport au message reçu. Alors qu'un objet obéit au message qui lui est transmis, un agent va d'abord l'évaluer avant de l'exécuter, car il doit contrôler son comportement en fonction des messages qui lui parviennent et de son état interne. Selon **[Maruichi 90]**, également repris par

[Sueyoshi 91], l'agent est ainsi modélisé comme un objet concurrent qui dispose de son propre interpréteur de message, assez sophistiqué.

La question du modèle d'agent se pose également. Ce modèle doit en effet permettre d'appréhender différentes catégories d'agents (personne, rôle ou groupe), et de construire une grande variété d'applications. Face à ce problème, certains auteurs ont cherché à bâtir un modèle d'agent abstrait et unique. C'est le cas du système PAGES **[Hämmäinen 90]**. Le modèle d'agent est ici fondé sur la notion de classe de formes, une classe spécifiant le comportement des instances, ainsi que leur façon de stocker, de présenter et de traiter l'information. La notion de forme est introduite pour couvrir un large éventail de besoins et pour opérer comme un mécanisme intégrateur : cette notion est pour cela similaire à la notion de carte utilisée dans HyperCard **[Harvey 88]**.

1. MODELES D'AGENTS COGNITIFS

Dans cette partie, nous utilisons la classification proposée par Ferber **[Ferber 91b]** pour décrire la structure des agents de quelques systèmes. Nous abordons ainsi successivement l'expertise du domaine, la modélisation des connaissances sur soi et sur les autres, la communication et ses protocoles, enfin le contrôle et les comportements.

1.1. EXPERTISE DU DOMAINE

L'expertise de l'agent lui confère sa compétence pour un domaine d'application donné. Elle regroupe les connaissances révélées par le comportement et qui sont propres à l'agent indépendamment de la société **[Pleiad 92]**. Ces connaissances peuvent être qualifiées de connaissances cognitives ou d'autoconnaissances **[Bessière 83]**.

Elle est en général codée sous forme de règles comme dans **[Woolridge 91]**. Cet auteur s'intéresse à la distribution des mécanismes d'inférence utilisés pour établir des hypothèses et leur valeur de vérité. Dans le système MACE **[Gasser 87]**, les agents comprennent une mémoire et des règles qui en régissent l'exploitation. Ce système est utilisé pour modéliser du parallélisme de bas niveau (un système distribué de règles de production, chaque règle étant un agent).

Dans le système PANDORA II **[Maruichi 90]**, l'expertise est codée sous formes de couples attributs-valeurs et de méthodes. Une autre partie de l'expertise est également codée par les règles de l'interpréteur. Sueyoshi **[Sueyoshi 91]** propose une application de ce système à la coordination de robots. Un agent est ici responsable d'une tâche qu'il doit entreprendre, et pour laquelle il doit communiquer avec les autres. Une tâche est représentée par un diagramme de transition d'état, qui définit le comportement de l'agent. Celui-ci peut modifier son

comportement en fonction des actions des autres, par observation de leur comportement.

Dans le système MAGES **[Bouron 91]**, des agents de granularité différente peuvent coexister et communiquer dans un environnement. Les agents de granularité élevée peuvent travailler comme des systèmes à base de tableau noir individuels, et interviennent comme des résolveurs de problèmes spécialisés. Ces agents utilisent alors des "frames" et des règles pour représenter la connaissance.

Enfin, dans le système PAGES **[Hämmäinen 90]**, la résolution se fait par migration de formes. Toutes les opérations spécifiques du domaine sont définies dans des classes de formes et toutes les données résident dans des instances de forme persistantes. Une forme est définie par un ensemble de champs associant des couples attributs-valeurs, et s'apparente à la notion de "frame". Des facettes de type "règles" sont attachées à ces formes, et déclenchées par la réception d'événements. Ces règles codent l'expertise du domaine : le domaine considéré est ici un travail de type secrétariat, il implique la gestion des rendez-vous, la gestion du courrier et le suivi des activités courantes. Différentes formes sont utilisées pour modéliser différents types de personnel (secrétaire par exemple) ou groupes de personnes (département des ventes par exemple). D'autres formes représentent différents types de messages, attachés à des tâches ou au transfert de données précises.

L'expertise du domaine peut donc être modélisée de façon plus ou moins complexe : de la simple base de faits et de règles aux hiérarchies d'objets ou de frames associées à des règles (systèmes MAGES et PAGES). Notons que cette dernière approche est similaire à celle introduite par les langages hybrides ART **[Clayton 84]** et KEE **[Fikes 85]**.

Nous nous intéressons à la conception d'applications dans des domaines différents. Le mécanisme de représentation des connaissances doit donc être à la fois indépendant d'un domaine d'application et, suffisamment riche et général pour permettre d'appréhender des connaissances hétérogènes. Un mode de représentation fondé sur une hiérarchie d'objets (ou de frames) associée à des règles complexes nous semble ainsi le plus à même de respecter à la fois le besoin de complexité et le besoin d'indépendance vis-à-vis de l'application.

1.2. CONNAISSANCES DE SOI ET DES AUTRES

Cette partie de l'agent lui permet de gérer ses relations avec les autres agents et est donc étroitement liée à la partie responsable des communications et des protocoles de communication. L'agent doit en effet modéliser le rôle, la compétence, la localisation ou les plans des autres agents pour pouvoir interagir avec eux. Ces connaissances portant sur les interactions, les relations avec les autres, la participation de l'agent dans la société, peuvent être alors qualifiées de connaissances sociales ou de paraconnaissances **[Bessière 83]**.

La manière la plus simple d'introduire des connaissances sur les autres est de la coder directement dans les règles d'expertise de l'agent, en indiquant par exemple à quel destinataire transmettre une information. Cette approche est utilisée par **[Maruichi 90]** et **[Hämmäinen 90]**. Dans ce dernier système, les agents utilisent diverses connaissances, sur eux-mêmes et sur les autres, pour interpréter les messages qui leur parviennent. Les connaissances nécessaires et leur utilisation diffèrent selon le type de message (il s'agit ici de messages en langue naturelle, de forme peu structurée) :

- en cas de "message social", il convient d'identifier l'envoyeur et les personnes ou groupes à qui retransmettre le message et donc d'identifier les rôles des groupes et personnes importantes ;
- en cas de "message économique", le message est filtré en fonction de sa taille et du type d'envoi (envoi personnel ou envoi en grand nombre), ce qui implique la connaissance de l'influence de ces paramètres sur la manière dont il sera perçu ;
- en cas de "message cognitif", le message est filtré selon son contenu, ce qui implique la connaissance des centres d'intérêt et des compétences des autres agents.

Dans une seconde approche, les connaissances sur les autres agents sont codées sous la forme des listes de leurs accointances. Dans le système MACE **[Gasser 87]** par exemple, chaque agent possède une liste de modèles des autres et de lui-même. Le modèle d'un agent est constitué de son nom, sa classe, son adresse, son rôle ou sa relation à l'agent initial ("Self" modélise l'agent initial, "Creator" modélise le créateur de l'agent initial, "Org-member" modélise un membre d'une organisation supervisée par l'agent initial, "My-org-manager" modélise le manager de l'organisation dont fait partie l'agent initial, "Co-worker" modélise un membre du même groupe). Il définit également sa compétence (une liste de buts et les méthodes pour les atteindre, en termes de procédures explicites ou des agents capables de les atteindre), ses buts et les plans pour les atteindre (un ensemble partiellement ordonné de buts ou de compétences, entrelacés avec des points de rendez-vous).

Pour tout agent, cette liste est initialisée avec un modèle de son créateur (nom, classe, adresse et rôle), et un modèle de lui-même plus complet. Il peut également disposer de modèles d'accointances prédéfinies. Cette liste peut être modifiée de manière dynamique, en ajoutant ou en ôtant de nouvelles accointances et les connaissances sur ces accointances. Finalement, l'agent peut à tout moment augmenter sa connaissance sur les autres en demandant des informations sur un agent (adresse, rôle,...). Dans le système MAGES **[Bouron 91]**, ces listes sont constituées en termes de buts, c'est-à-dire associées à l'agent dans le cadre d'un problème à résoudre.

Dans le système FELINE **[Woolridge 91]**, les agents maintiennent également une structure

de données représentant leurs croyances sur eux-mêmes et leur environnement. Cette structure de données est appelée modèle de l'environnement. Elle contient une entrée pour l'agent qui la détient et une entrée pour chaque agent avec lequel il peut communiquer (ses accointances). Chaque entrée contient 2 attributs : les "compétences" et l'"intérêt". L'attribut "compétences" représente les hypothèses que l'agent peut émettre ou nier. Elles correspondent approximativement aux noeuds racines du réseau d'inférence représentant l'expertise du domaine de l'agent. L'attribut "intérêt" contient les identificateurs des hypothèses pour lesquelles l'agent peut vouloir la valeur de vérité. Ils correspondent plutôt aux feuilles du réseau d'inférences.

Une question importante est de savoir comment manipuler et acquérir dynamiquement une telle connaissance. Dans les systèmes MAGES et MACE, des fonctions pour manipuler les listes d'accointances sont prédéfinies, telles les fonctions de recherche associative d'agents selon certains critères (attributs du modèle) du système MACE. La connaissance des autres est soit prédéfinie, soit acquise par échange de messages. Certains auteurs se sont également attachés à définir des méthodes d'observation **[Sueyoshi 91]** permettant aux agents la modélisation dynamique des autres.

Pour cela, les actions entreprises par un agent sont modélisées par un diagramme de transition d'état, une tâche étant représentée par un 6-tuplet (Entrée, Etats Internes, Sorties, Fonctions de Transition, Fonctions de Sortie, Etat Initial). Un agent construit alors un modèle des tâches entreprises par les autres, en observant les paramètres d'entrée et de sortie des tâches, considérant ainsi l'agent comme une boite noire. Une telle observation est réalisée par l'envoi d'une requête à un agent "observateur" chargé de superviser l'environnement (dit d'observation) au sein duquel s'observent les agents.

L'intérêt de ces facultés d'observation est d'aboutir à la modélisation dynamique des tâches entreprises par les autres agents et de permettre la prédiction, la simulation ou la reproduction des actions qu'ils vont entreprendre. L'agent doit ensuite savoir établir sa propre stratégie en fonction de ces connaissances, c'est-à-dire modifier son propre diagramme de transition d'état, adaptant ainsi son comportement de manière dynamique et coordonnant ses actions avec celles des autres agents. Les capacités de reproduction d'actions peuvent être vues comme des capacités d'apprentissage par l'observation.

Un codage direct des connaissances sur les autres (dans des règles par exemple) permet de simplifier les raisonnements que doit effectuer l'agent. Par exemple, introduire directement le destinataire d'un message limitera les raisonnements à effectuer pour le déduire à partir des connaissances disponibles sur les autres. Par contre, ce codage direct se fait au détriment de l'autonomie de l'agent. Il faut en effet qu'un agent soit suffisamment autonome, c'est-à-dire

non figé dans une organisation pré-définie, et donc dispose d'une représentation intermédiaire des rôles, des capacités ou des intentions des autres agents. Dans ce cas, l'agent doit bâtir cette représentation et donc disposer de raisonnements et de protocoles de communication plus sophistiqués. La manière dont doivent être introduites les connaissances sur les autres semble donc être un compromis entre d'une part la simplification des raisonnements et des communications, et d'autre part le besoin d'autonomie des agents.

1.3. COMMUNICATION ET PROTOCOLES

Un agent doit maintenir une représentation du monde et des autres pour raisonner pleinement ; le résultat sera d'autant meilleur si la représentation est complète. Pour construire cette représentation, l'agent doit acquérir des connaissances sur les autres et sur l'environnement en communicant et en percevant **[Berthet 92]**. Cette acquisition peut être directe : une requête explicite est alors effectuée par un agent afin de maintenir sa propre représentation des connaissances, ou indirecte : des connaissances sur les autres agents sont dans ce cas inférées à partir des actes de communication qui ont été mis en œuvre pour arriver à un but commun.

La communication entre agents s'effectue en général par l'envoi de messages asynchrones, objets passifs transférés d'un expéditeur à un ou plusieurs destinataires (un agent, un groupe ou tous les agents) **[Maruichi 90]**, **[Sueyoshi 91]**, **[Gasser 87]**. Ces messages sont stockés dans une file d'attente, avec un système de priorités attaché. Deux types de communications sont possibles comme chez **[Hämmäinen 90]** : la communication directe (inter-processus) et la communication indirecte (par un tiers).

Les messages diffèrent selon leur structure et leur contenu. En ce qui concerne leur structure, elle est en général simple, et contrainte à une liste structurée **[Sueyoshi 91]**, **[Woolridge 91]** comportant les adresses de l'expéditeur et des destinataires, ainsi qu'un champ contenu dont la forme est plus ou moins contrainte (non contrainte chez **[Gasser 87]**). Elle est plus complexe dans le système PAGES **[Hämmäinen 90]**, où les agents communiquent par l'échange de "frames" complexes appelées formes.

En ce qui concerne leur contenu, différents types de messages peuvent être distingués selon :

- le type de destinataire (messages d'observation par exemple chez **[Sueyoshi 91]**) ;
- le type de message (message de requête, de réponse ou d'information chez **[Woolridge 91]**) ;
- la forme de l'information transmise.

Trois types de messages sont distingués chez **[Hämmäinen 90]** : les messages en langue naturelle, les messages structurés (par champs attributs valeurs), et les messages programmés.

Ces messages sont alors des objets actifs programmables ou agents mobiles, comprenant des variables privées, des règles internes et des fonctions d'interfaces, capables de transporter de l'information et de prendre des décisions sur leur destination et leur apparence ; leur migration est contrôlée par un moniteur particulier.

Pour permettre la communication entre agents hétérogènes, il peut être nécessaire d'utiliser des messages de complexités différentes. Dans le système MAGES **[Bouron 91]**, la communication implique ainsi des messages simples ou des messages plus sophistiqués de type "frame", qui font partie de la base de connaissances de l'agent. Des capacités de communication sophistiquées sont conférées aux agents les plus complexes, sous la forme d'un langage de haut niveau, incluant protocole et actes de langages.

Une bonne description de quelques protocoles de communication peut être trouvée dans **[Berthet 92]**. Les protocoles introduits peuvent être très simples comme dans le système de Gaspar **[Gaspar 91]**, qui introduit seulement quatre messages qui expriment des modes de communication différents (présentation, information, requête et réponse). Des protocoles plus sophistiqués seront utilisés pour diminuer les échanges en chaînant les messages émis lors du protocole **[Sian 91]**, ou pour supprimer la redondance dans les échanges ou les boucles qui peuvent survenir dans la protocole **[Campbell 90]**. Dans le système de Campbell, une taxonomie des différentes tonalités des messages couvrant par exemple la demande d'information, l'instruction, la promesse ou le marchandage, a été bâtie. Un protocole particulier est alors employé par le récepteur selon la tonalité du message. Des modèles issus de l'étude linguistique des situations de lutte dans des sociétés humaines sont également utilisés chez Chang **[Chang 91]**.

Les différentes approches présentées posent le problème de l'intention de la communication. Les auteurs font remarquer que plus le protocole est simple, plus le langage est compliqué et plus l'intention de communication est codée dans l'agent. Cette intention peut être entièrement codée dans l'agent (dans ce cas, il n'y pas de protocoles sophistiqués), partiellement codée dans le protocole ou entièrement codée dans le protocole (dans ce cas, les protocoles sont sophistiqués).

Cet accent mis sur les communications et sur les actes de langages traduit l'effort d'augmenter l'aspect social de l'agent. Il parait dans ce cadre important de disposer au moins des deux types de communication de base (synchrone et asynchrone), qui induisent les protocoles les plus simples. A partir de là, deux solutions sont possibles, soit disposer d'un langage de programmation sophistiqué qui permette de bâtir des protocoles plus complexes à partir de ces protocoles minimaux, soit disposer de protocoles de plus haut niveau. Le fait de disposer de protocoles de plus haut niveau est intéressant car l'efficacité de l'agent est augmentée par une

diminution des échanges et des redondances par rapport à une programmation explicite. Il faut cependant que le concepteur de l'agent dispose d'une boîte à outils de protocoles suffisamment riche. La solution idéale serait de disposer à la fois de protocoles de haut niveau mais également d'un langage sophistiqué qui permette d'implanter un protocole particulier.

1.4. CONTROLE

Deux niveaux de contrôle sont introduits au sein de l'agent. Un premier niveau concerne la planification et le suivi des tâches internes de l'agent telles l'analyse d'un message ou la mise à jour de sa base de connaissances. Il définit la façon dont l'agent va exploiter ses connaissances, son expertise, lors de la réception d'un message et recoupe les compétences, les stratégies et les mécanismes d'inférences, de raisonnement et de décisions d'un agent. Un second niveau concerne le suivi des tâches externes telles la demande d'une information à un autre agent. Ce niveau est ainsi lié à la communication avec autres agents et aux protocoles de communication.

Dans le système PAGES **[Hämmäinen 90]**, le comportement des agents est globalement dirigé par les événements. Toutes les opérations sont fondées sur le paradigme "événement-condition-action". Les événements correspondent à des entrées de l'utilisateur, à des arrivées de formes, ou à des messages internes du "timer". Ces événements déclenchent la facette règle correspondante des champs et des formes de l'agent. Ces facettes peuvent être prédéfinies ou programmables, l'utilisateur pouvant également en ajouter d'autres. L'intérêt des facettes standard est de contrôler certaines actions de base : empêcher la destruction d'une forme si certaines conditions ne sont pas remplies, empêcher l'envoi d'une forme à un autre agent, ou restaurer l'état antérieur de certains champs. Ces mécanismes permettent de garantir la préservation de contraintes explicites au niveau système. Les messages eux-mêmes peuvent posséder des facettes de comportement, dans le cas des messages programmés qui sont définis comme des "agents mobiles".

Dans d'autres approches, un contrôle supplémentaire au niveau des messages est introduit avec la définition d'interpréteurs de messages. Dans l'approche de **[Maruichi 90]**, l'interpréteur de message s'exécute dès qu'un agent est créé. Lorsqu'un message est reçu, il est ajouté dans une file d'attente. Le rôle de l'interpréteur est alors de rechercher un message important dans cette file, et de l'exécuter. Il dispose pour cela de méthodes expertes de gestion de file d'attente. L'interpréteur dispose de règles expertes lui permettant de raisonner sur l'état de l'agent et la nature de la requête, et de décider des actions à entreprendre en fonction de cela. Le comportement de base est défini par le moteur qui régit l'activité de l'agent. Dans le système MACE **[Gasser 87]** un agent est initialement dans un état passif, il passe à l'état actif dès la réception d'un message. C'est alors la responsabilité du moteur de faire passer l'agent à l'état inactif dès qu'il a fini sa tâche.

D'autres approches font intervenir des types de comportements différents dans une même architecture. Dans le système MAGES **[Bouron 91]**, les agents sont associés en une hiérarchie. L'agent "Kernel" définit un comportement générique hérité par tous les agents (comportement réactif). La classe "Expert" l'augmente par l'adjonction d'un moteur d'inférence. Un niveau de complexité supplémentaire intervient dans les agents "Message" qui peuvent intégrer des messages transmis sous forme de "frame" dans leur base de connaissances. Enfin, les agents "Blackboard" intègrent l'activation des sources de connaissances (KS) et des sources de contrôle et permettent l'intégration de blackboard de contrôle à plusieurs niveaux.

L'étude effectuée montre la diversification des rôles impartis aux agents, de leurs capacités de raisonnement, de comportement et de communication. Il parait important d'obtenir cette diversification par la spécification de modèles généraux, susceptibles de spécialisations mais aussi par la conception d'interpréteurs de messages de plus en plus sophistiqués. Dans ce cadre, la définition de cycles de contrôle génériques pour un type d'agent donné, qui utilisent les diverses connaissances de l'agent et qui définissent ainsi des comportements distincts selon celles-ci parait nécessaire.

2. SOCIÉTÉS COGNITIVES

L'objectif est ici d'analyser les interactions entre les agents sous l'angle social. Dans l'approche considérée, l'organisation sociale n'est pas représentée de manière explicite, mais plutôt représentée indirectement par les engagements et les attentes de ses membres **[Bouron 91]**. Trois paramètres régissent l'organisation et le fonctionnement des sociétés d'agents : la distribution des tâches et des compétences, les modes et les protocoles de communication, et les modes d'organisation et de coopération. Ces paramètres influent alors sur le raisonnement de la société.

2.1. DISTRIBUTION

Le problème de la distribution peut être posé en termes de distribution des connaissances, de distribution des tâches et de distribution du contrôle. Deux solutions à la résolution distribuée de problèmes peuvent en effet être distinguées **[Chaib-Draa 90]**, **[Woolridge 91]** : le partage de tâches et le partage de connaissances. Le partage de tâche ou partage de rôles est une forme de coopération dans laquelle les agents s'assistent mutuellement en partageant les rôles et les tâches nécessaires à la résolution d'un même problème. Cette approche conduit naturellement à une distribution des connaissances en termes de connaissances opératoires, c'est-à-dire les connaissances qui permettent de manipuler des éléments figuratifs d'un problème telle une procédure ou une action d'inférence. Le partage des connaissances est une

forme de coopération dans laquelle les agents s'assistent mutuellement par des échanges de connaissances, de façon à permettre à chacun d'eux d'affiner sa propre connaissance du monde et de construire progressivement sa propre solution. Cette approche conduit naturellement à une distribution des connaissances en termes de connaissances figuratives, c'est-à-dire les connaissances qui permettent de nommer et de décrire les éléments d'un problème tels des données, des faits, des hypothèses ou des résultats **[Ovalle 91]**.

Ces deux approches de la distribution impliquent alors des notions de contrôle différentes **[Chaib-Draa 90]**. Dans le cas du partage de tâches, deux types de contrôle peuvent intervenir, qui tout deux définissent les modes d'attribution des tâches : une approche empirique fondée sur des alliances établies au sein d'une organisation et une approche fondée sur la négociation. Dans le cas du partage de connaissances, deux types de contrôle peuvent également intervenir : un contrôle fondé sur le partage de résultats intermédiaires et un contrôle de nature plus sociale impliquant des connaissances sur les autres et l'environnement.

Une distribution du contrôle lui-même est enfin proposée par **[Regazzoni 91]**, qui introduit des modules spécialisés dans des tâches de contrôle : ces modules de contrôle ou régulateurs sont chargés de contrôler les résultats de certains modules de traitement, et éventuellement d'en réguler les paramètres.

Il semble particulièrement intéressant de distinguer connaissances figuratives et connaissances opératoires et des les distribuer au sein d'agents différents. En effet, cette solution permet d'une part de lier à chaque type de connaissance le niveau de contrôle qui lui est propre, et d'autre part de définir des ensembles d'agents où apparaissent les quatre types de contrôle intervenant dans le cas du partage de connaissances et du partage de tâches.

2.2. COMMUNICATION

Les modes et protocoles de communication établissent les moyens puis les styles de communication entre agents. Ces modes de communication ne se réduisent pas à un niveau d'implantation algorithmique, ils induisent au contraire des styles de coopération particuliers. Deux types de systèmes multi-agents peuvent être distingués selon le mode de communication développé par les agents : les approches à base de tableau noir et les approches fondées sur la communication par envoi de messages. Ces deux approches peuvent également être combinées et ont donné naissance à des approches hybrides.

Approche par Partage de ressources (tableau noir)

La communication par partage de ressources (ou approche de type tableau noir) a été introduite très tôt dans des applications de reconnaissance de la parole et de vision par ordinateur. Dans

cette approche, la communication entre les différents agents du système s'effectue par l'intermédiaire d'une structure de données globale partagée par tous les agents. Cette structure, organisée en niveaux correspondant aux niveaux d'abstraction d'un problème, renferme les données initiales ainsi que les résultats partiels fournis par les agents lors du processus de résolution de problème. Parmi les systèmes les plus connus utilisant cette approche, citons ATOME **[Maître 90]**, Hearsay III **[Erman 81]** et BB-1 **[Hayes-Roth 85]**.

Approche par Envoi de messages

Une seconde approche est fondée sur l'envoi de messages entre agents. Les systèmes utilisant ce type de communication relèvent alors d'une distribution totale des connaissances, des résultats partiels et des traitements à effectuer pour parvenir à un résultat. Les langages d'acteurs donnent une excellente représentation de cette approche **[Hewitt 77]**.

Comparaison des deux approches

Chacune de ces deux approches possède des avantages et des inconvénients. Chaib-Draa **[Chaib-Draa 90]** rappelle les conclusions des comparaisons effectuées entre ces deux modes de communication. Une comparaison a été effectuée par **[Leblanc 86]** qui note que l'échange de messages est plus efficace dans certains cas que le partage de ressources, et par Hewitt **[Hewitt 83]** et **[Hewitt 84]** qui pense que les approches à base de tableau noir sont irréalisables dans un contexte distribué. Dans le cas où il n'y a qu'un seul tableau noir, un goulot d'étranglement se forme en effet quand le nombre de communication devient important, et dans le cas où le tableau noir est distribué, l'approche revient alors à une approche par envoi de messages entre tableau noir. L'approche par envoi de messages implique néanmoins l'élaboration d'un langage de communication plus complexe, les agents pouvant posséder des représentations du monde différentes et des formes de communication spécifiques.

Approches hybrides

D'autres travaux de recherche tendent à combiner les deux approches, en introduisant par exemple des agents conçus comme des systèmes à base de tableau noir et communicant avec les autres agents par envoi de messages. Ces approches ont par exemple été introduites dans des systèmes de vision par ordinateur **[Lane 89]** et dans des systèmes dédiés au raisonnement temporel **[Zanconato 88]**. Ces travaux sont des tentatives pour combiner l'efficacité du partage d'information et de la supervision des approches de type tableau noir avec la souplesse des approches de type envoi de messages.

L'intérêt de la communication par tableau noir est qu'elle s'applique bien dans le cas d'une distinction entre connaissances figuratives et opératoires, dont nous avons souligné

l'importance. Par contre, l'accès au tableau noir est souvent lourd et rigide. La solution consistant à distribuer le tableau noir, c'est-à-dire créant des agents figuratifs semble la plus intéressante. La communication alors introduite sera une communication hybride.

2.3. COOPÉRATION

Nous présentons dans ce qui suit, quatre approches pour la coopération entre agents cognitifs : la négociation de contrats, l'échange de résultats intermédiaires, l'approche organisationnelle et la planification locale. Il convient de remarquer que le contrôle est représenté de différentes façons selon le style de communication adopté (communication par tableau noir ou par envoi de messages). Il est plutôt de nature hiérarchique dans le cas des approches par tableau noir **[Laasri 89]** et plutôt de nature distribuée dans le cas des approches par envoi de messages.

Coopération par négociation de contrats

Dans les approches fondées sur le principe de la négociation, des alliances temporelles sont définies entre des agents dans un réseau. Deux formes d'allocation de sous-tâches fondées sur la négociation sont le "Contract Net Protocol" **[Smith 80]** **[Davis 83]** et la négociation multi-niveaux **[Conry 88]** qui implique plusieurs itérations du processus de négociation. Dans ces approches, les rôles et les tâches sont alloués de manière dynamique en fonction de la disponibilité des ressources et des capacités effectives des agents. Cependant le procédé de négociation obéit à une stratégie de développement descendante qui présente un caractère assez rigide. Du fait du manque de vue globale de l'état du réseau à un instant donné, des tâches similaires risquent de plus d'être exécutées dans deux contrats différents. Enfin, un risque de récursivité peut apparaître, du fait que les agents peuvent assurer des rôles différents pour différents contrats.

Coopération par échange de résultats intermédiaires

L'objectif est ici de traiter l'existence de vues inconsistantes. Dans des systèmes à base de connaissances importants, il est en effet improbable qu'une base de données totalement consistante puisse être maintenue. L'existence d'inconsistances est ainsi inévitable et peut même permettre de considérer des interprétations différentes. Les inconsistances sont alors résolues en échangeant les résultats de haut niveau issus de l'interprétation de données de bas niveau. Cette résolution des ambiguités forme une partie intégrale de la tâche de résolution de problème. Permettre un libre échange d'information pour résoudre les ambiguités, conduit rapidement à des coûts de communication excessifs **[Lesser 80]**. Dans une étude de la métaphore de la communauté scientifique, Kornfeld **[Kornfeld 81]** a montré comment réduire la communication de solutions partielles à un petit nombre d'agents.

Par opposition au protocole de la négociation, cette approche est de nature ascendante, ce qui implique une plus grande autonomie des prises de décision des agents. Les solutions sont générées au niveau de l'agent sans être influencées par les autres agents. Les agents pouvant échanger librement des informations sur l'hypothèse de travail courante, acquièrent une vue plus globale de la résolution. Par contre, aucune allocation de tâche ni de planification n'est effectuée.

Coopération par approche organisationnelle

L'approche organisationnelle constitue un compromis entre la négociation de contrat et le partage de résultats intermédiaires en définissant des alliances parmi les agents appartenant à la même organisation : les agents possèdent ainsi une vue de haut niveau de leur rôle dans le processus de résolution et de leurs relations avec les autres. Ces rôles et relations sont pré-assignés d'une façon plutôt permanente, chaque fois qu'une nouvelle organisation est instanciée pour exécuter une tâche.

Des stratégies de contrôle hiérarchiques peuvent être développées comme dans le système IBIS **[Dellepiane 89]**, qui est une adaptation du concept de tableau noir. Dans ce système, les sources de connaissances peuvent communiquer à travers le tableau noir en utilisant un schéma hiérarchique des connexions existant entre elles. Seuls des messages de contrôle peuvent être échangés entre les sources de connaissances, les données doivent toujours être partagées à travers la base de données.

Le problème principal des agents travaillant en coopération est que des situations conflictuelles surviennent fréquemment, comme résultant d'environnements changeant constamment et imprévisibles. Un concept pour résoudre de tels problèmes à travers un dialogue entre les agents est présenté dans **[Galliers 88]**.

La capacité à reformuler les buts et à réallouer les tâches pour résoudre les inconsistances qui peuvent survenir dans une interprétation est essentielle, c'est pourquoi certaines approches permettent de modifier dynamiquement la structure organisationnelle. Corkill et Lesser par exemple utilisent des structures organisationnelles pour réduire les échanges d'informations entre les agents **[Corkill 83]**. Ces structures sont élaborées de manière dynamique et concurrente par les agents. Durfee et Lesser **[Durfee 87]** suggèrent qu'un méta-niveau de contrôle est nécessaire pour guider la génération de nouvelles structures et pour identifier les échecs des structures existantes.

Coopération par planification locale

La coordination est la notion centrale de la coopération par planification locale : c'est une partie

de l'activité de contrôle des résolveurs de problème conçus selon ce principe. La génération de plans partiels globaux fournit à chaque agent une vue globale sur la façon dont est résolue une tâche à un instant donné, ce qui est une manière de pallier le paradigme de la rationalité limitée. Une meilleure distribution des efforts est ainsi obtenue parmi les agents et la redondance est diminuée. Trois types d'informations sont manipulées dans cette approche, dont la principale caractéristique est la planification : les plans locaux, les plans agent et les plans partiels globaux.

Dans une approche multi-agents centralisée, Georgeff **[Georgeff 83]** se fonde sur un seul agent central pour détecter les conflits potentiels dans un plan initial. De nombreuses alternatives pour la représentation des événements survenant au cours de la planification ont été utilisées **[Georgeff 84]** **[Georgeff 86]** et **[Lansky 87]**. Dans une autre approche **[Camarata 83]**, une négociation dynamique limitée est introduite pour sélectionner un agent qui assure alors la responsabilité du développement d'un plan consistant.

Des approches plus distribuées de la planification sont décrites par **[Corkill 79]** **[Corkill 83]** **[Konolige 83]** et **[Durfee 87]**. Le système décrit par Corkill est fondé sur NOAH **[Sacerdoti 77]** et utilise une approche hiérarchique pour le développement de plans dans lesquels les conflits à chaque niveau sont résolus avant de traiter le niveau suivant. Dans cette approche, chaque agent à un niveau particulier construit un modèle des ressources nécessaires pour les autres agents de ce niveau, afin qu'ils puissent chacun développer un plan synchronisé. Dans le système décrit par Durfee et Lesser, des plans développés localement sont échangés entre les agents pour planifier la synchronisation des actions. Cette approche permet à la tâche de résolution de problème de débuter avant qu'un plan global n'ait été développé, contrairement à la définition stricte de la planification multi-agents. Le système de Durfee et Lesser utilise également l'approche organisationnelle et la négociation.

DePlan est un planificateur hiérarchique qui permet aux agents de modéliser les actions des autres agents, d'utiliser ces modèles à l'intérieur de leur propre plan et d'anticiper ainsi les activités futures. Le résultat est un système de résolution de problème coopératif décrit par **[Hopkins 88]**. Des recherches supplémentaires dans le comportement des agents coopérants sont en cours. **[Connah 88]** présente une plate-forme pour la recherche sur de tels systèmes multi-agents.

La coopération est étroitement liée aux problèmes de communication entre agents. Dans le cas du partage de résultats intermédiaires, il s'agit d'échanger des solutions, ou des hypothèses. Chaque agent dispose alors d'une vue globale de l'ensemble de la résolution mais il y a des problèmes de coûts de communication et les organisations sont assez statiques. Dans les approches où le contrôle est plus distribué, de type planification locale, les coûts de communication sont moindres et la communication concerne les connaissances sur les autres

pour définir des plans locaux. Les organisations sont dans ce cas plus dynamiques.

IV CONCLUSION

Dans ce chapitre, nous avons présenté les principaux langages à objets et l'approche cognitive des systèmes multi-agents, du point de vue de la structure interne des agents et des différentes interactions. Ces deux domaines sont assez proches, les systèmes multi-agents empruntant beaucoup aux objets et surtout aux acteurs. La notion importante introduite par les systèmes multi-agents est l'autonomie de décision par rapport aux messages que ne possèdent pas les objets. Une autre caractéristique importante des systèmes multi-agents est la diversité de la structure des agents et la diversité des types de coopération. Notre but par rapport à cet état de l'art est ainsi la conception d'un langage de programmation, inspiré des langages à objets, qui permette de répondre à ces besoins d'autonomie et de diversité.

CHAPITRE 2

MOTIVATIONS

I INTRODUCTION

Notre objectif est la réalisation d'applications dans les domaines de la Vision par Ordinateur et du Diagnostic Biomédical. Ces applications complexes nécessitent l'intégration de connaissances hétérogènes par ailleurs souvent issues de domaines d'expertise différents. Ces connaissances concernent aussi bien des faits, des données ou des hypothèses que des procédures ou des règles d'inférence. Cette distinction entre des connaissances que nous appelerons figuratives et des connaissances que nous appelerons opératoires, distinction qui nous parait essentielle, se retrouve dans les approches orientées objets. Les langages à objets ont en effet le souci de séparer le déclaratif du procédural, c'est le cas des langages de classes qui possèdent une partie déclarative sous forme d'attributs et une partie procédurale sous forme de méthodes. C'est également le cas des langages de frames qui attachent à chaque attribut des facettes procédurales (facettes réflexes). Afin de pouvoir introduire des raisonnements sur ces objets, notre premier souci a donc été de coupler une représentation à base d'objets avec des règles de production.

Notre second souci a été de distribuer les objets et de les doter d'autonomie (glissement vers le concept d'acteur). Dans ce cadre, il nous a paru intéressant d'employer les techniques de l'Intelligence Artificielle Distribuée, et notamment d'utiliser l'approche multi-agents. Par ailleurs, nous avons voulu pousser le paradigme connaissances figuratives / connaissances opératoires que l'on trouve dans les langages à objets mais également dans les systèmes à base de tableau noir et concevoir à cet effet des agents dédiés.

Nous avons eu finalement le souci de disposer d'un outil de conception souple, regroupant un langage de programmation d'agent et un environnement de programmation, afin de disposer d'un outil non pas dédié à un seul domaine d'application mais réutilisable. Dans ce chapitre, nous présentons tout d'abord nos choix par rapport au modèle d'agent, puis nos choix par rapport à la société d'agents et enfin nos choix pour le langage de programmation.

II MODELE D'AGENT

Le modèle d'agent reprend dans sa structure les éléments présentés au chapitre précédent : l'expertise, les connaissances sur les autres, la communication et le contrôle. Notre base de départ a été de définir des agents de type système expert communicant et dédiés selon le type des connaissances manipulées, afin de conserver la distinction déclaratif / procédural. Dans ce cadre, il nous a paru utile de bâtir des types d'agents génériques spécialisables par un langage de programmation.

Nous avons ainsi retenu une représentation des connaissances expertes sous forme de hiérarchies d'objets pour les connaissances déclaratives et une représentation sous forme de règles d'inférence (pouvant faire appel à des procédures) pour les connaissances opératoires. Ce mode de représentation nous parait suffisamment général et indépendant d'un domaine d'application, et se rapproche du mode de représentation utilisé dans les langages hybrides tes ART **[Clayton 84]** ou KEE **[Fikes 85]**. Les règles vont jouer un rôle important au niveau du contrôle interne par l'introduction de raisonnements sur les objets, c'est-à-dire de raisonnements sur l'expertise de l'agent. Du fait de la distinction entre agents figuratifs et agents opératoires, les échanges entre ces agents concerneront principalement des résultats. Ces échanges seront effectués à partir des règles, qui vont ainsi également jouer un rôle au niveau du contrôle externe.

Ce type de représentation et le rôle alloué aux règles va induire une programmation bas niveau. De fait, les connaissances sur les autres vont être codées explicitement dans l'agent, à l'aide du langage de programmation, limitant ainsi les raisonnements sur les autres.

En ce qui concerne les communications, du fait de l'approche utilisée pour le codage des connaissances sur les autres, les échanges vont essentiellement concerner la transmission de données et de résultats. Il n'y aura pas de protocoles complexes dus à la nécessité d'acquérir des connaissances sur les autres. Nous nous sommes donc limités à une communication de bas-niveau (messages synchrones) provenant également de la programmation orientée objet, et du protocole de plus haut niveau constitué par la technique de diffusion (broadcast). Il sera cependant possible par une programmation explicite à l'aide des règles des agents de définir des protocoles plus complexes.

En ce qui concerne le contrôle, l'introduction de règles dans les agents nous a amené à définir des moteurs d'inférence. En outre, nous avons choisi de doter chaque type d'agent de cycles de contrôle génériques correspondant aux comportements de l'agent face à la réception d'un message. Ces cycles de contrôle génériques utiliseront les connaissances expertes pour effectuer les raisonnements et communiquer avec les autres selon les protocoles codés dans

l'agent. Rappelons que le contrôle va s'exercer à un niveau interne par le choix des règles à appliquer et des objets à sélectionner, et à un niveau externe par le choix des agents à activer.

III SOCIÉTÉ

La société d'agents doit être abordée en terme de distribution des connaissances et du contrôle, en terme de communication et en terme de coopération.

En terme de distribution des connaissances, la distinction d'agents chargés des connaissances figuratives et d'agents chargés des connaissances opératoires nous rapproche du partage des tâches et des connaissances introduit par Chaib-Draa **[Chai-Draa 90]**. Cette distinction, que l'on retrouve également dans des approches de type tableau noir, ne se limite pas à cette seule distribution, elle induit également une distribution du contrôle au sein de chaque agent, et différencie notre approche des approches à contrôle centralisé. Un contrôle figuratif (focalisation sur des données) sera ainsi introduit dans les agents figuratifs et une contrôle opératoire (adaptation d'un traitement à des données) dans les agents opératoires.

En terme de communication, nous sommes en présence d'une communication hybride. En effet, la disctinction figuratif / opératoire est similaire à la distinction introduite dans les approches à base de tableau noir. Le tableau lui-même s'apparente aux agents figuratifs et les sources de conniassance s'apparentent aux agents opératoires. La définition d'agents figuratifs correspond à une distribution du tableau noir, et ces agents vont communiquer avec des agents opératoires. Notre approche ressemble ainsi à une approche fondée sur les tableaux noirs communicants. Une dimension supplémentaire est cependant introduite dans ces agents figuratifs, avec l'introduction d'un contrôle figuratif.

En terme de coopération, notre approche conduit naturellement à une coopération par partage de résultats intermédiaires, les résultats des traitements effectués par les agents chargés des connaissances opératoires étant stockés dans les agents chargés des connaissances figuratives. Les organisations bâties avec nos agents vont également introduire une structuration en terme de tâches. Un agent opératoire peut en effet être assimilé à une tâche et différentes tâches vont ainsi se succéder en échangeant leurs résultats par l'intermédiaire des agents figuratifs. Chaque agent va en outre posséder un plan partiel de la résolution d'un problème par le biais de son expertise locale. En fait, notre approche introduit une programmation de bas niveau qui peut permettre de reprendre de nombreux aspects des différents types de coopération. Cependant, par rapport aux organisations construites avec un langage de plus haut niveau, introduisant une plus grande abstraction, nos organisations seront plutôt statiques.

IV LANGAGE

Du fait de nos applications, le langage que nous proposons de développer doit être issu de plusieurs formalismes de programmation et de représentation des connaissances : procédures, langages à objets et règles de production. Le langage doit donc bénéficier des vocations des trois styles de programmation : programmation fonctionnelle, programmation objet et programmation logique, et va donc s'apparenter à un langage hybride. Nous avons par ailleurs déjà signalé que le mode de représentation des connaissances choisi (objets et règles d'inférence) se rapprochait du mode de représentation des langages hybrides KEE et ART.

Ce langage doit également servir à définir les agents par spécialisation de classes génériques. Rappellons par ailleurs que nos agents seront des agents autonomes dotés d'un comportment générique. Il est similaire en cela aux langages d'acteurs. Notre approche en différera cependant par la communication : nous avons en effet retenu une messagerie synchrone, plus proche d'une programmation de type langage de classes (classes et méthodes).

La spécialisation de ces classes génériques va s'obtenir par la définition de règles et d'objets. En fait, le rôle important alloué aux règles (raisonnement interne sur les objets, et raisonnement externe lié aux communications) va impliquer une programmation de bas niveau certes limitante par le manque d'abstraction mais assez puissante pour définir des applications complexes.

Nous avons enfin décidé de créer un langage efficace, et donc de bâtir ce langage comme une couche logicielle au dessus d'un langage performant, en l'occurence le langage C++. Nous avons retenu le langage C++ plutôt qu'un langage déclaratif comme Lisp, car ce langage est au départ un langage à objets, car il est d'assez bas niveau et doit nous permettre d'accéder assez facilement aux routines systèmes Unix, et pour des raisons d'efficacité.

V CONCLUSION

Deux classes d'agents génériques, de type système expert, vont être définies permettant de représenter les connaissances figuratives et les connaissances opératoires. Cette distinction de deux types d'agent nous parait importante car elle va nous permettre de distribuer également le contrôle propre à chaque type de connaissance.

La définition des agents se fera par spécialisation de ces classes génériques à l'aide du langage de programmation. Il faut noter ici le rôle central des règles dans nos agents. Elles vont en effet servir à définir un contrôle interne, en introduisant un raisonnement sur des objets, mais

également un contrôle externe puisque chargées des échanges avec les autres agents. Le langage de programmation va permettre cela avec un style d'écriture unique.

Ces différents choix effectués nous ont finalement amené à créer un environnement et un langage de programmation d'assez bas niveau, MAPS, qui sera décrit en détail au chapitre suivant. Ce langage va disposer de nombreuses primitives de bas niveau qui devraient permettre de programmer des applications complexes. Certes, pour pouvoir entreprendre de réels problèmes de coopération ou de planification, il vaudrait mieux disposer d'un langage doté de primitives de plus haut niveau comme par exemple des protocoles de communication plus complexes. Cependant, la solution retenue, autorisant la programmation explicite des raisonnements et des protocoles va permettre d'en programmer et d'en étudier plusieurs et de définir ainsi un cahier des charges, utilisable pour de futures extensions du langage.

CHAPITRE 3

MAPS

I INTRODUCTION

L'environnement de programmation MAPS (Multi-Agents Problem Solver) a été conçu comme une architecture générique pour le développement d'applications **[Pesty 89] [Baujard 90a]**. MAPS est à la fois un langage de programmation, permettant de définir les agents intervenant dans un système multi-agents, et un environnement de programmation permettant la conception et la mise au point d'applications multi-agents complexes.

Dans les paragraphes suivants, nous étudions l'approche multi-agents MAPS. Nous présentons tout d'abord l'approche du point de vue agent à travers sa structure interne et son comportement, puis du point de vue société d'agents à travers les modes d'organisation et de coopération impliqués dans des groupes d'agents. Nous présentons ensuite, en l'illustrant par un exemple simple, le langage de programmation MAPS et le style de programmation. Finalement, nous présentons les choix d'implantation et les principales caractéristiques techniques de l'environnement de programmation MAPS.

II ARCHITECTURE LOGICIELLE

Le système MAPS est fondé sur la distinction entre deux types de connaissance : les connaissances figuratives et les connaissances opératoires. Les connaissances figuratives regroupent les connaissances sur les propriétés, les relations décrivant les éléments d'un problème et les stratégies de focalisation sur ces éléments, alors que les connaissances opératoires se réfèrent aux outils (procéduraux ou inférenciels) et aux stratégies d'exploitation de ces éléments. Modéliser et exploiter ces deux types de connaissances est essentiel pour résoudre un problème; ceci nécessite un savoir-faire dédié, définissant la manière et l'opportunité de se focaliser sur un élément d'un problème ou de sélectionner un opérateur de traitement. Le contrôle est ainsi également distribué au sein de ces deux types d'agents, et s'exerce en terme de focalisation. Cette approche conduit à la conception de réseaux d'agents où les agents KS dépositaires des hypothèses et des résultats alternent avec les agents KP qui peuvent alors être considérés comme des tâches. Les agents KS sont ainsi connectés à des agents KP et vice-versa. Dans ce paragraphe, nous allons tout d'abord décrire ces deux types génériques d'agents, puis nous étudierons leur distribution et leur organisation dans des

architectures complexes.

1. LES AGENTS MAPS

Deux classes d'agents génériques ont été introduites : les serveurs de connaissance, ou agents KS (Knowledge Server) ont été conçus pour modéliser et exploiter les connaissances figuratives, tandis que les processeurs de connaissance, ou agents KP (Knowledge Processor) ont été conçus pour modéliser et exploiter les connaissances opératoires. Tout agent MAPS est de plus défini comme un système à base de connaissances à part entière, communicant avec les autres agents par envoi de messages, en mode "commande" **[Hautin 86]**. Chaque agent est ainsi conçu comme une entité autonome capable de réagir à la réception d'événements extérieurs selon un comportement dédié. Selon la classification donnée dans **[Erceau 91]**, les agents MAPS sont plutôt des agents cognitifs communicants.

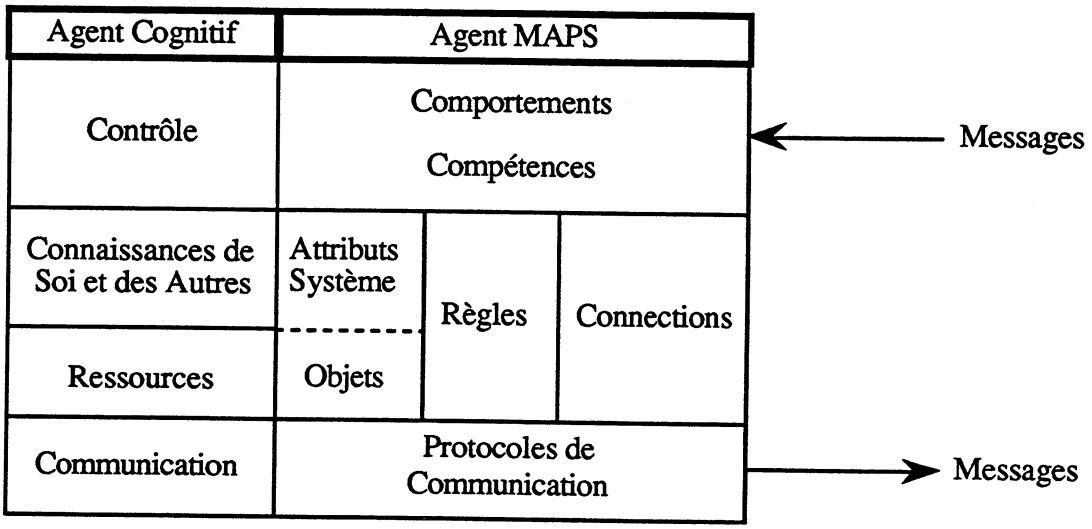


Fig. B.1. Structure d'un agent cognitif.

La figure **B.1** résume les quatres parties de notre modèle d'agent, fondé sur le modèle de Ferber **[Ferber 91b]**. La partie gauche de la figure reprend les aspects généraux : ils seront utilisés par la suite pour présenter les agents KS et KP. La partie droite de la figure présente les caractéristiques correspondantes des agents MAPS.

1.1. LES RESSOURCES

Chaque agent est défini par un ensemble de ressources, qui peut être considéré comme une base de connaissances à part entière. Des objets peuvent être définis : ils sont alors dotés de mécanismes classiques d'accès tels que la lecture ou l'écriture de valeurs d'attributs. Des règles peuvent également être définies : toutes les opérations de bas niveau telles que la consultation, la restriction et l'application des règles seront effectuées par les moteurs d'inférence à ce niveau.

1.1.1. AGENT KS

Un agent KS est chargé de modéliser et d'exploiter des connaissances figuratives, ces connaissances étant représentées sous forme d'objets **(Fig. B.2)**. Dans MAPS, les objets définissent un modèle pour la structure de leur représentants physiques (instances), regroupent des attributs (ou variables d'instance) et peuvent être organisés en hiérarchie, introduisant ainsi une structuration hiérarchique de la connaissance. Un mécanisme d'héritage simple est introduit dans ce cas. Ces objets peuvent également être considérés du point de vue conceptuel comme des unités de connaissances, représentant le prototype d'un concept **[Masini 90]**, voire d'un cadre de pensée ; les instances peuvent alors être considérées comme des mondes de pensée. La notion d'instance rejoint ici la notion d'environnement de travail. Ce dernier est construit comme un modèle dynamique du monde au fur et à mesure des traitements effectués par les agents. Cet environnement constitue le contexte effectif de travail des agents.

Aux attributs d'information définis par le concepteur des agents (attributs "utilisateur"), le système MAPS adjoint les attributs de contrôle (attributs "système") PROPOSE_FLAG, INTERROGATE_FLAG et RESULT_FLAG. Ces attributs sont ajoutés à la liste des attributs de chaque objet et hérités par leur instances. Une liste d'instances et une instance courante (ou instance de travail) sont également maintenues pour chaque objet.

Dans le système MAPS, tous les accès à des attributs s'effectuent sur une instance particulière, l'instance courante ou instance de travail. Les méthodes "Obtenir" et "Modifier" contrôlent ainsi l'accès en lecture et en écriture des attributs d'information de cette instance. Des méthodes dédiées, LAST_PROPOSITION, LAST_INTERROGATION et LAST_RESULT, permettent quant à elles d'accéder en lecture aux attributs de contrôle de cette instance.

Un ensemble de méthodes dédiées est également fourni pour manipuler les instances d'objets. Elles permettent de créer ou de détruire des instances et également de changer l'instance courante. Ces méthodes permettent ainsi de créer ou de détruire des mondes de pensée, ou de se focaliser sur un monde de pensée particulier. Ces méthodes qui visent à modifier ou mettre à jour l'instance de travail constituent des moyens de focalisation dans l'environnement de travail.

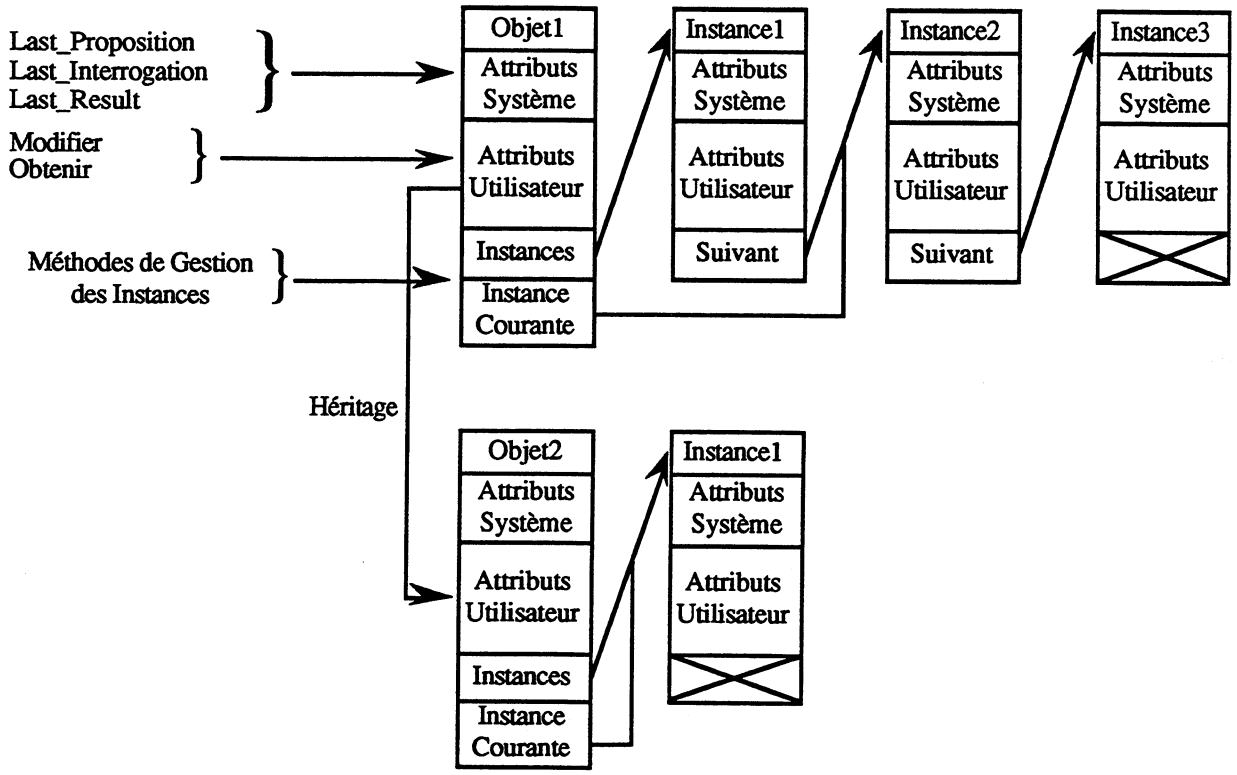


Fig. B.2. MAPS : objets, instances et méthodes associées.

Chaque agent dispose en outre d'un jeu de règles dites primaires, dont le rôle est de propager localement une information ou d'effectuer des opérations de vérification de la cohérence (règles de propagation), et de deux jeux de règles exploratoires (règles de proposition et règles d'interrogation) dont la fonction consiste à sélectionner une information susceptible d'être transmise ou requise vis-à-vis de l'extérieur, c'est-à-dire un agent KP. Les règles exploratoires ont donc un rôle mixte de focalisation interne et de focalisation externe (liée à la communication entre agent).

Les mécanismes d'accès à ces règles sont alors les méthodes "Propager", "Proposer" et "Interroger", qui conditionnent la propagation interne (Propager) et la sélection des informations devant être transmises à l'extérieur (Proposer) ou requises (Interroger). Ces méthodes activent le moteur d'inférence de l'agent en le paramétrant avec le jeu de règles requis.

1.1.2. AGENT KP

Un agent KP est chargé de manipuler des connaissances opératoires. Pour représenter ces connaissances, il dispose d'un ensemble de règles, dites règles d'action, qui peuvent être des règles d'inférence classiques ou des règles plus complexes, mettant en œuvre des procédures d'analyse et faisant appel à des valeurs d'attributs d'instances stockées dans des agents KS. Des méta-règles, dites règles de stratégie, sont également introduites, qui contrôlent la sélection des règles d'actions par un mécanisme de restriction. Des méthodes dédiées conditionnent la

sélection (règles de stratégie) puis l'application des éléments opératoires (règles d'action). Ces méthodes activent le moteur d'inférence, en le paramétrant avec le jeu de règles requis.

1.2. Les Connaissances sur Soi et sur les Autres

Dans MAPS, un agent possède des connaissances sur lui même et sur les autres agents. Ces connaissances peuvent concerner aussi bien une trace des actions effectuées par l'agent que ses compétences et celles des autres. Une partie de ces connaissances, que nous appelerons connaissances statiques, est fournie avec les ressources de l'agent (son expertise). L'autre partie, que nous appelerons connaissances dynamiques, est acquise pendant le déroulement d'une exécution. Grâce à ces connaissances, les agents MAPS sont capables de raisonnements complexes comme le montre la figure **B.3**.

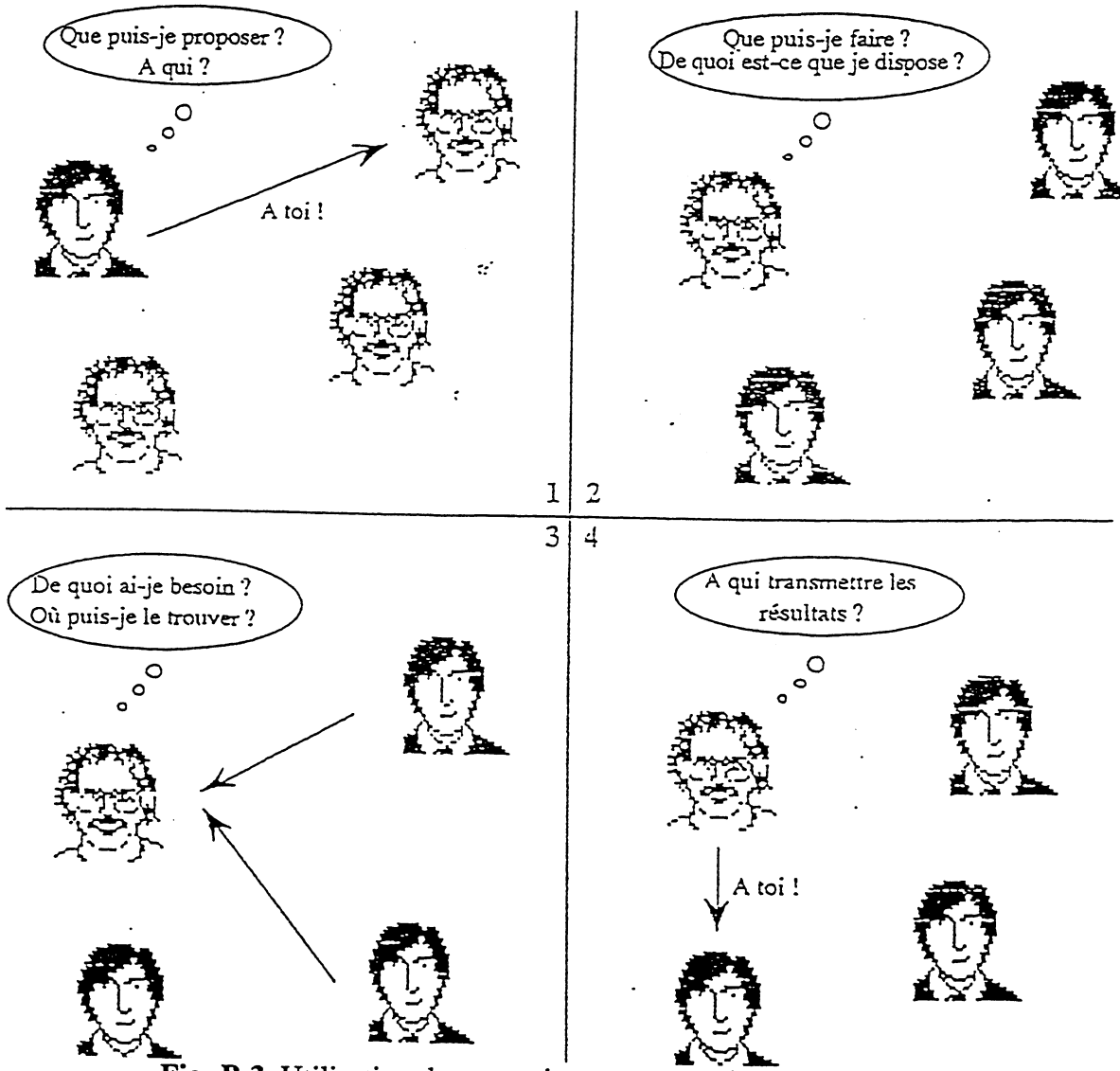


Fig. B.3. Utilisation des connaissances sur soi et sur les autres.
1 - Connaissances sur soi et sur les autres (règles de proposition)
2 - Connaissances sur soi (règles de stratégie)
3, 4 - Connaissances sur les autres (règles d'action)

1.2.1. AGENT KS

Connaissances statiques

Un agent KS possède des connaissances statiques sur les autres agents. Il connait ses accointances, c'est-à-dire les agents KP auxquels il est connecté et avec lesquels il peut communiquer. Cette connaissance est codée sous la forme d'une liste et fait partie des ressources de l'agent.

Les règles de proposition et d'interrogation contiennent également des connaissances sur les agents KP. Elles contiennent en effet l'identification de l'agent KP auquel envoyer (proposition) ou demander (interrogation) une information. Un agent KS possède ainsi la connaissance implicite des compétences des autres agents.

Connaissances dynamiques

Un agent KS possède des connaissances dynamiques sur lui-même et sur les autres agents.

Ces connaissances constituent une trace des actions effectuées par l'agent KS. Elles sont fournies par les attributs système des instances d'objets. Ces connaissances sont donc attachées à chacune des instances d'un objet, maintenues dynamiquement par l'agent, ce qui lui permet de raisonner sur des mondes de pensée. L'attribut PROPOSE_FLAG contient l'identification du dernier agent auquel l'instance a été transmise (Proposition). L'attribut INTERROGATE_FLAG contient l'identification du dernier agent auquel une information sur l'instance a été demandée (Interrogation). L'attribut RESULT_FLAG permet quant à lui de connaître le résultat (échec ou succès) d'un transfert ou d'une demande d'information.

Ces connaissances peuvent être accédées par leurs méthodes dédiées et utilisées dans les règles de l'agent KS, qui peut ainsi raisonner sur ce qu'il a déjà entrepris et sur les résultats obtenus. L'attribut RESULT_FLAG fournit dans une certaine mesure des connaissances dynamiques sur les autres. L'agent KS peut en effet déduire de sa valeur, associée à la valeur de l'attribut PROPOSE_FLAG ou INTERROGATE_FLAG, qu'un agent KP n'est pas capable de répondre à un transfert ou une demande d'information.

1.2.2. AGENT KP

Connaissances statiques

Un agent KS possède des connaissances statiques sur lui-même et sur les autres agents. Grâce à ses règles de stratégie, un agent KP possède une connaissance sur l'applicabilité d'une action à une situation donnée. Un agent KP connait également ses accointances, c'est-à-dire les agents KS auxquels il est connecté et avec lesquels il peut communiquer. Cette connaissance est codée

sous la forme d'une liste et fait partie des ressources de l'agent.

Finalement, depuis une règle d'action ou de stratégie, un agent KP peut accéder (demande ou transfert) à une information contenue dans un agent KS. Il possède ainsi la connaissance implicite de la localisation d'une information.

Connaissances dynamiques

Un agent KP possède des connaissances dynamiques sur les autres agents. Il peut en effet accéder depuis ses règles aux attributs système d'une instance d'objet stockée dans un agent KS. Il peut ainsi acquérir une trace des actions entreprises par l'agent KS et des résultats obtenus. Il peut ainsi raisonner sur ces connaissances et effectuer ses propres actions en connaissance de cause.

1.3. LES COMMUNICATIONS

La partie communication définit le comportement social des agents, entités autonomes qui réagissent aux événements extérieurs. Nous avons déjà pu voir dans les paragraphes précédents qu'un certain nombre d'échange d'information était possible entre les agents KS et KP. La communication s'effectue par envoi de messages en mode commande **[Hautin 86]**, et non par partage d'informations comme dans les approches de type tableau noir. Deux classes de messages peuvent être échangés par les agents KS et KP : des messages d'information et des messages de contrôle.

1.3.1. AGENT KS

Un agent KS peut recevoir des messages d'information et des messages de contrôle, transmis par un agent KP. Ces messages permettent à un agent KP d'accéder aux ressources d'un agent KS : ils sont en effet interprétés comme des appels aux méthodes d'accès à ces ressources. Des protocoles de communication dédiés sont alors mis en œuvre entre les agents en réponse à ces requêtes.

Messages

Deux messages d'information peuvent être adressés à un agent KS : la requête "Mettre à jour" (Set) et la requête "Collecter" (Get). A travers la requête "Mettre à jour", un agent KS reçoit de nouvelles informations depuis un agent KP, qui pourra ainsi modifier sa base de connaissances. A travers la requête "Obtenir", un agent KS reçoit une demande d'information depuis un agent KP. Ces requêtes permettent de communiquer sur les propriétés d'un objet (instance courante). La connaissance figurative maintenue par un KS peut ainsi être partagée par des agents KP.

Trois messages de contrôle concernant les attributs système PROPOSE_FLAG,

INTERROGATE_FLAG et RESULT_FLAG peuvent également être adressés à un agent KS. Ces messages permettent l'activation des méthodes d'accès à ces attributs, présentées au paragraphe concernant les ressources d'un agent KS. Ces messages permettent ainsi à un agent KP d'obtenir des informations sur les actions entreprises par un agent KS et les résultats obtenus pour ces actions, c'est-à-dire d'avoir un accès à l'historique de cet agent.

Enfin, trois messages de focalisation concernant les instances d'objets peuvent être adressés à un agent KS. Ces messages permettent l'activation des méthodes de manipulation d'instances, présentées au paragraphe concernant les ressources d'un agent KS. Ces messages permettent ainsi à un agent KP de manipuler les instances d'objet dans un agent KS, c'est-à-dire de changer d'objet de discussion (instance courante) ou d'observer une nouvelle instance d'un phénomène (création d'instance).

A travers ces messages, le rôle d'un agent KP est d'enrichir la connaissance sur l'environnement de travail d'un agent KS par des actions de description (calcul d'attributs d'instances), par des actions de perception (découverte de nouveaux objets de l'environnement) et par des actions de compréhension ou d'identification (transformations progressives des attributs attachés aux instances).

Protocoles de communication

Toutes les requêtes décrites précédemment s'effectuent en mode synchrone **(Fig. B.4)**, l'agent émetteur attendant une réponse de l'agent récepteur. Dans le cas d'une requête "Mettre à jour", l'agent KP émetteur attend confirmation que l'information transmise a bien été utile à l'agent KS. Dans le cas d'une requête "Collecter", l'agent KP émetteur attend la réponse de l'agent KS (information demandée ou échec). Dans le cas des messages de contrôle, l'agent KS signifie par sa réponse, l'échec ou le succès de la requête.

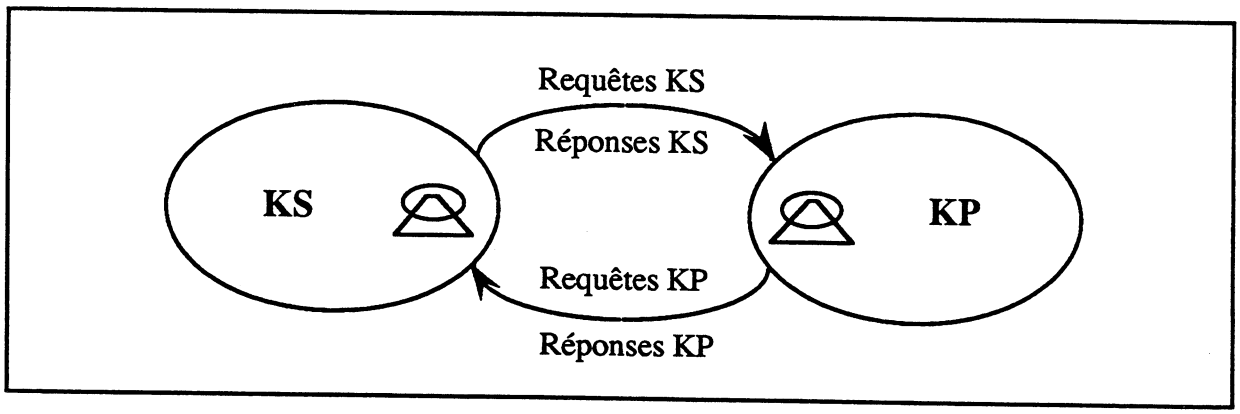


Fig B.4. Modes de transmission des messages KP <--> KS

Notons que les règles exploratoires (proposition et interrogation) qui permettent de sélectionner

un agent KP avec lequel communiquer, permettent de programmer des protocoles d'échanges plus complexes.

1.3.2. AGENT K P

Un agent KP reçoit uniquement des messages d'information depuis un agent KS. Des protocoles de communication dédiés sont alors mis en œuvre entre les agents en réponse à ces requêtes.

Messages

Deux requêtes portant sur des informations peuvent ainsi être adressée à un agent KP : la requête "Envoyer" (Send) et la requête "Demander" (Ask). A travers la requête "Envoyer", un agent KP reçoit la demande explicite de traiter l'information transmise depuis un agent KS. Cette information est alors une instance d'objet, ou un attribut d'instance. A travers la requête "Demander", un agent KP reçoit la demande d'une information (spécification d'un but à atteindre, d'un attribut d'instance à calculer) depuis un agent KS.

A travers ces messages, le rôle d'un agent KS est de veiller à la cohérence et à la complétude des connaissances sur l'environnement de travail (message "Demander") et de veiller à la progression du système dans la compréhension de cet environnement (message "Envoyer").

Protocoles de communication

Ces deux requêtes s'effectuent en mode synchrone **(Fig. B.4)**, dans le cas de la requête "Envoyer", l'agent émetteur attend confirmation par l'agent KP de sa capacité à traiter l'information, ou bien le signal que le traitement a échoué. Dans le cas de la requête "Demander", l'agent émetteur attend la réponse de l'agent KP au problème posé (échec ou résultat). Ces résultats sont stockés dans l'attribut RESULT_FLAG des instances considérées. Il faut noter ici que la notion d'échec est globale : il n'y a pas de différenciation entre un échec dû à l'incapacité de l'agent KP et un échec dû à l'incapacité des agents KS auprès desquels l'agent KP a cherché des informations.

Notons que les communications impliqués par les règles de l'agent KP, en prémisse (recherche de données) ou en conclusion (transmission de résultats), permettent comme pour les agents KS de programmer des protocoles plus complexes.

1.4. LE CONTROLE

La partie contrôle définit comment un message va être interprété et comment l'agent va réagir à sa réception. Cette réaction s'effectue en utilisant l'expertise donnée à l'agent ainsi que ses connaissances sur lui et sur les autres agents. Cette partie contrôle peut être décomposée en trois

niveaux. Le premier niveau correspond aux comportements, qui peuvent être considérés comme des plans d'action. Le deuxième niveau correspond aux actions proprement dites, qui constituent les compétences de l'agent. Ces compétences font appel à l'expertise de l'agent : manipulation d'objets ou de règles. Le troisième niveau concerne finalement le fonctionnement des moteurs d'inférence.

On peut noter que l'expertise donnée à l'agent sous forme de règles définit à nouveau mais à un niveau d'abstraction différent, des plans ou des stratégies ainsi que des actions à entreprendre.

1.4.1. AGENT KS

Le rôle des agents KS est de recevoir ou de fournir des informations sur les éléments d'un problème. Deux comportements, nommés "Recevoir" et "Fournir", sont ainsi attachés à tout agent KS ; ces comportements définissent la manière de réagir face à la réception ou à la demande d'une information impliquées par les requêtes "Mettre à jour" et "Collecter". Ils mettent en œuvre la capacité à maintenir, rechercher ou sélectionner des éléments de connaissance. Cette partie définit l'intentionnalité de l'agent KS, c'est-à-dire "la motivation qui le porte à faire en sorte qu'une situation advienne. Il s'agit alors de relier ses buts, ses possibilités d'actions, ce qu'il sait des autres et les engagements qu'il a contracté dans une perspective globale qui lui permette d'aboutir dans ce qu'il entreprend" **[Ferber 91b]**.

Niveau comportement

Les comportements de l'agent KS sont résumés ci-dessous sous une forme algorithmique.

Recevoir

```
(Et     Modifier
        (Ou (Et Proposer
                (Envoyer KP Proposition))
            (Et Interroger
                (Envoyer KP Interrogation)
                (Envoyer Self Recevoir))))
```

Fournir

```
(Ou     Obtenir
        (Et Proposer
            (Envoyer KP Proposition))
        (Et Interroger
            (Envoyer KP Interrogation)
            (Envoyer Self Recevoir)))
```

Dans le cas du comportement "Recevoir", la base de connaissances est tout d'abord mise à jour (Modifier), ce qui peut mettre en œuvre des vérifications ou propagations internes, avec le déclenchement des règles de propagation (Propager). Les règles exploratoires sont alors activées pour sélectionner et proposer une information valide à un agent KP à des fins de traitement (Proposition suivi de l'envoi du message "Propose") . En cas d'échec, d'autres règles exploratoires peuvent sélectionner une information manquante à requérir auprès d'un agent KP (Interrogation suivi de l'envoi du message "Interroge"); le cycle Recevoir en entier est alors répété.

Dans le cas du comportement "Fournir", l'élément est tout d'abord recherché dans la base de

connaissances de l'agent KS et retourné en cas de succès. Un agent KP externe, faisant partie des accointances de l'agent KS est invoqué sinon : une stratégie de broadcast est appliquée dans ce cas pour rechercher l'agent capable de résoudre le problème. En cas d'échec total et afin de ne pas se bloquer, l'agent KS transmettra une nouvelle information vers l'extérieur (application des cycles Proposition / Interrogation) : il y a dans ce cas rupture de la stratégie de résolution en cours.

Ces comportements traduisent l'intentionnalité d'un agent KS. Le comportement "Recevoir" dénote l'intention de communiquer de l'agent KS. L'agent a la volonté de diffuser des connaissances, de focaliser les traitements des autres agents sur un monde de pensée (Proposition) et la volonté de compléter ses connaissances sur l'environnement de travail (Interrogation). Le comportement "Fournir" dénote la volonté de ne pas bloquer le système à cause d'un échec. En cas d'échec local, des agents KP sont invoqués et en cas d'échec de ces agents, une stratégie de reprise est développée.

Niveau compétence

Un agent KS regroupe tout d'abord des compétences "dédiées", dépendantes du domaine d'application et exprimées par l'expertise de l'agent sous la forme de ses règles de propagation et de sélection. Il regroupe également des compétences indépendantes du domaine qui conditionnent l'application et l'activation des règles : capacités de propagation, de proposition et d'interrogation.

Capacité de propagation

La capacité de propagation permet la mise en œuvre de mécanismes de propagation interne ou d'activités de maintien de la cohérence.

Capacité de proposition

La capacité de proposition permet à l'agent de sélectionner des éléments figuratifs à transmettre à des agents KP à des fins de traitements (sélection figurative locale). Cette sélection change l'instance courante d'un objet et constitue ainsi une focalisation sur un monde de pensée dans l'environnement.

Une sélection d'un agent parmi les agents KP est effectuée simultanément, qui spécifie l'agent KP auquel transmettre l'information sélectionnée (sélection opératoire locale). Par le biais de cette sélection, une planification des traitements à appliquer peut être mise en œuvre. Cette capacité traduit la volonté de l'agent KS que le système progresse dans sa connaissance de l'environnement.

Capacité d'interrogation

La capacité d'interrogation permet à l'agent KS de sélectionner une donnée à demander ou un problème à soumettre à un agent KP. Une sélection d'un agent KP est également effectuée. Nous retrouvons ici la sélection figurative locale et la sélection opératoire locale présentée pour les capacités de proposition. Cette capacité traduit la volonté de l'agent KS de veiller à la complétude de ses connaissances sur l'environnement.

Niveau moteur d'inférence

L'agent KS possède un moteur d'inférence général : ce moteur fonctionne en chaînage avant. Un ensemble de paramètres permet de le spécialiser pour fonctionner sur les différents ensembles de règles d'un agent KS.

Règles de propagation

La capacité de propagation active le moteur d'inférence sur l'ensemble des règles de propagation. Le moteur effectue alors le cycle suivant :

1- Recherche d'un ensemble de conflit parmi l'ensemble des règles de propagation.
2- Activation des règles de l'ensemble de conflit.
3- Reprise du cycle jusqu'à l'obtention d'un ensemble de conflit vide.

Au cours des cycles, le moteur garantit qu'une règle n'est exécutée qu'une fois, évitant ainsi les bouclages possibles. Le moteur fonctionne ici par saturation, il s'arrête lorsque toutes les règles applicables ont été exécutées.

Règles de proposition

La capacité de proposition active le moteur d'inférence sur l'ensemble des règles de proposition. Le moteur effectue alors le cycle suivant :

1- Recherche d'un ensemble de conflit parmi les règles de proposition
2- Activation de la première règle de l'ensemble de conflit
3- Arrêt

Une seule règle est appliquée dans ce cas, qui peut posséder une partie action (conclusion) complexe. Le moteur collecte ainsi un ensemble de couple (élément figuratif, agent KP) qui sera utilisé par la suite, chaque élément figuratif étant transmis à l'agent KP.

Règles d'interrogation

La capacité d'interrogation active le moteur d'inférence sur l'ensemble des règles d'interrogation. Le moteur effectue alors le cycle suivant :

1- Recherche d'un ensemble de conflit parmi les règles d'interrogation
2- Activation de la première règle de l'ensemble de conflit
3- Arrêt

Une seule règle est appliquée, qui peut posséder une partie action (conclusion) complexe. Le moteur collecte ainsi un ensemble de couple (élément figuratif, agent KP) qui sera utilisé par la suite, chaque élément figuratif étant requis auprès de l'agent KP.

1.4.2. AGENT KP

Le rôle de tout agent KP est de traiter des informations ou de résoudre des problèmes. Deux comportements, nommés "Traiter" et "Résoudre" ont ainsi été définis, qui mettent en œuvre l'activation des règles d'action à la suite de la requête "Envoyer" ou "Demander". Une focalisation conceptuelle est tout d'abord effectuée (sélection d'un sous ensemble de règles d'action par des stratégies) suivie d'une focalisation perceptuelle (sélection des règles applicables).

Niveau comportement

Les comportements de l'agent KP obéissent à un schéma assez linéaire. Le contrôle est transmis à la compétence Traiter-Donnée dans le cas du comportement "Traiter" et à la compétence Résoudre-Problème dans le cas du comportement "Résoudre". L'agent ne peut conclure qu'au succès ou à l'échec de ses activités. L'intentionnalité d'un agent KP est faible, elle est plus proche du schéma stimulus / réponse. L'intentionnalité de l'agent a en fait été reportée dans ses règles.

Niveau compétence

Un agent KP regroupe tout d'abord des compétences "dédiées", dépendantes du domaine d'application et exprimées par l'expertise de l'agent sous la forme de ses règles d'action et de stratégie. Il regroupe également des compétences indépendantes du domaine qui conditionnent l'application et l'activation des règles : capacités à traiter une donnée et à résoudre un problème.

Capacité à traiter une donnée

La capacité à traiter une donnée met en œuvre l'application des règles d'action après une sélection par les règles de stratégie. La restriction et le filtrage des règles d'action s'effectuera

dans ce cas sur leur prémisse.

Capacité à résoudre un problème

La capacité à résoudre un problème met en œuvre l'application des règles d'action après une sélection par les règles de stratégie. La restriction et le filtrage des règles d'action s'effectuera dans ce cas sur leur conclusion.

Les règles de stratégie définissent dans les deux cas une sélection opératoire locale. Par le biais de cette sélection, une planification des actions à entreprendre peut être mise en œuvre. Les règles d'action quand à elles mettent en œuvre des sélections figuratives (demande ou transfert d'information auprès d'un agent KS). Ces sélections constituent un mécanisme de focalisation sur des informations partagées au sein d'agents KS.

Niveau moteur d'inférence

Le moteur d'un agent KP fonctionne en chaînage mixte. Dans le cas des deux capacités, le moteur d'inférence sera appliqué sur les règles de stratégie puis sur les règles d'action. Son fonctionnement résumé par la figure **B.5** sera alors le suivant :

Règles de stratégie

1- Définition de l'ensemble de conflit parmi les règles de stratégie.
2- Application des règles de l'ensemble de conflit et obtention des restrictions.
3- Définition du sous ensemble de règles d'action.
4- Arrêt et paramétrage du moteur d'inférence avec le sous ensemble de règles d'action.

Règles d'action

1- Définition de l'ensemble de conflit parmi les règles d'action retenues.
2- Application des règles de l'ensemble de conflit.
3- Arrêt.

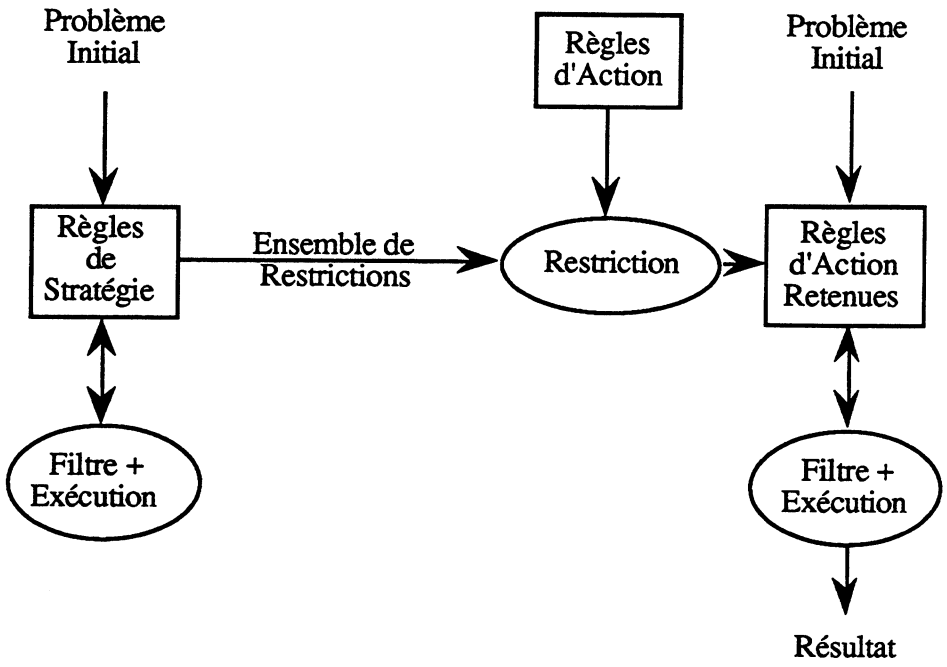


Fig. B.5. Fonctionnement du moteurs d'inférence des agents KP

2. LA SOCIÉTÉ D'AGENTS

Dans le système MAPS, un groupe d'agents n'est pas une entité physique mais apparait plutôt comme la résultante des activités de communication des agents à un moment donné. Un agent forme ainsi un groupe avec ses accointances.

Cette partie reprend mais au niveau du groupe la structuration introduite au niveau de l'agent. Une première partie étudie la distribution des tâches et des compétences ; elle correspond à la partie expertise de l'agent. La deuxième partie étudie les modes et protocoles de communication. Enfin, la troisième partie étudie les modes d'organisation et de coopération ; elle correspond à la partie contrôle de l'agent.

2.1. DISTRIBUTION DES TACHES ET DES COMPÉTENCES

La figure **B.6** représente un réseau minimal d'agents. Dans un tel réseau, les agents KS alternent avec les agents KP. Ce réseau comprend un KS amont, un KP central et un KS aval. Le KS amont fournit des données au KP qui les traite et qui transmet les résultats au KS aval.

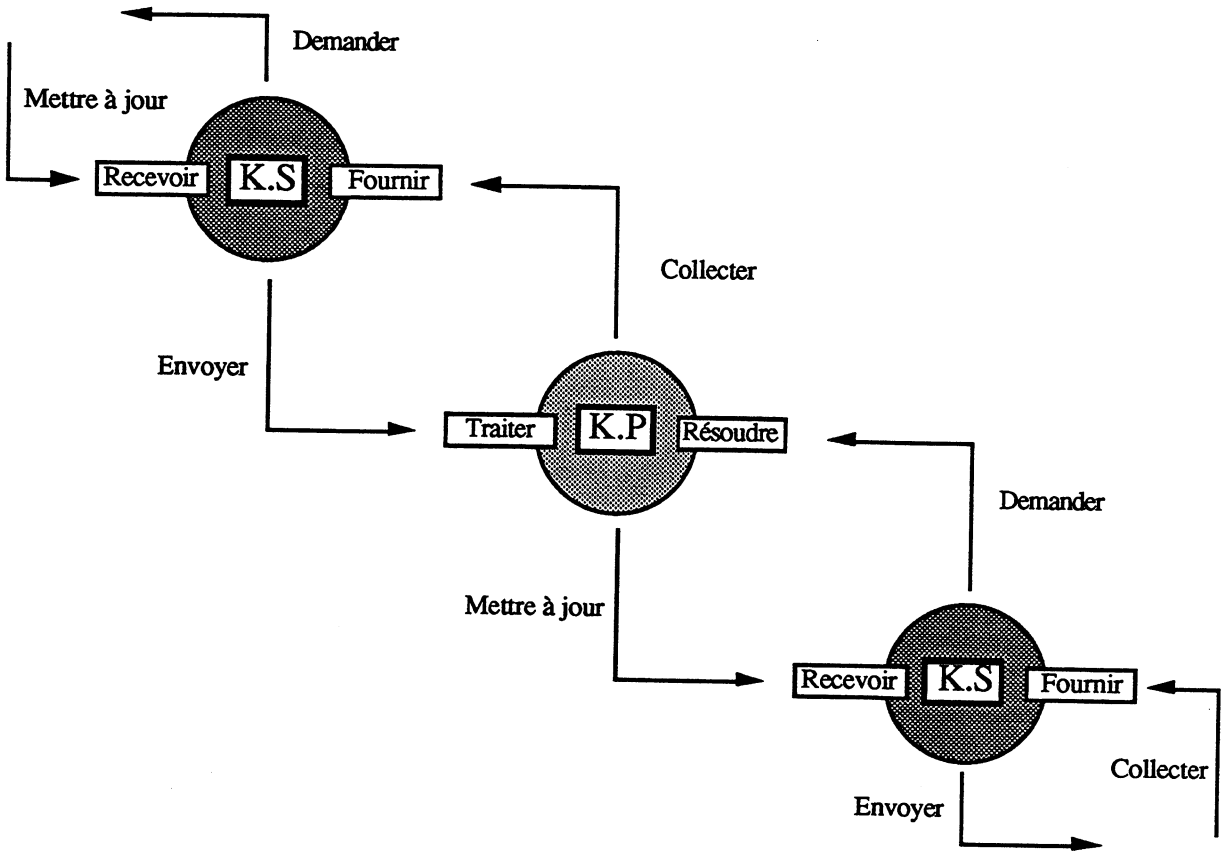


Fig. B.6. Réseau minimal d'agents.

Deux types de groupes peuvent être dégagés dans de telles architectures : des groupes "centrés

connaissance" et des groupes "centrés tâche". La figure **B.7** représente une architecture où de tels groupes sont mis en évidence. Dans cette architecture, un groupe "centré connaissance" met en œuvre un agent KS et plusieurs agents KP. L'agent KS correspond alors à un niveau d'abstraction et les agents KP mettent en œuvre des traitements locaux. Chaque groupe "centré connaissance" communique avec d'autres groupes par l'envoi de messages : cet échange de messages constitue le passage d'un niveau d'abstraction à un autre. Chaque niveau d'abstraction étant lié à un style de représentation, un mécanisme de traduction est mis en œuvre pour que les messages soient compréhensibles d'un niveau à l'autre. Ce mécanisme est effectué par un agent KP reliant deux niveaux. Un groupe "centré connaissance" peut par exemple représenter une image et un autre groupe des primitives régions. Le passage d'un niveau d'abstraction à l'autre s'effectuera par l'intermédiaire d'un agent KP qui effectuera une segmentation région. Ce type de distribution introduit alors une coopération orientée vers la progression et la complétion de la connaissance.

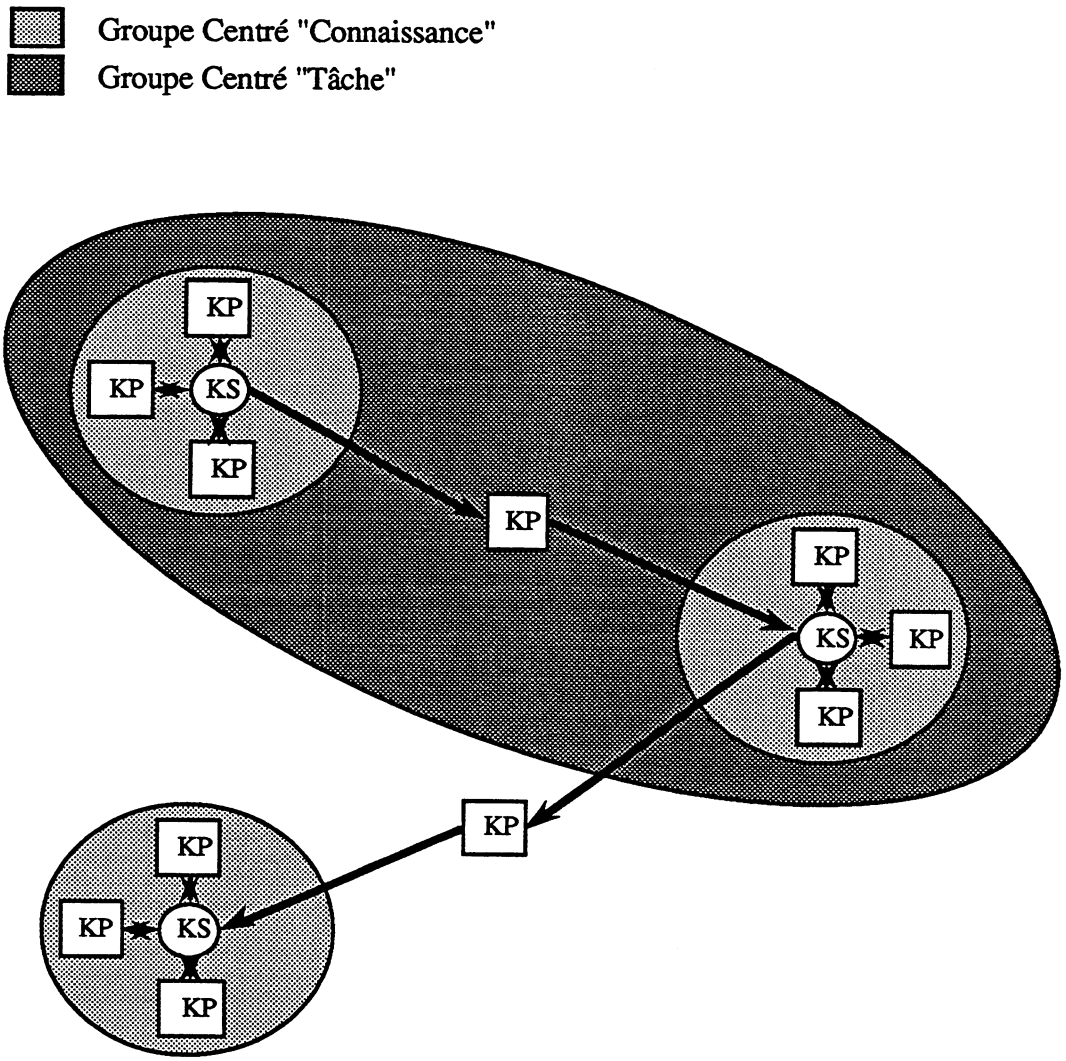


Fig. B.7. Architecture centrée "tâche" et centrée "connaissance".

Cette architecture met également en évidence des groupes "centré tâche" impliquant un agent KP

et plusieurs agents KS. L'agent KP est lié à une tâche particulière et les agents KS maintiennent les données, les hypothèses et les résultats. Chaque groupe "centré tâche" communique avec d'autres groupes par partage de résultats intermédiaires. Ce type de distribution introduit alors une coopération orientée partage de tâches.

L'intérêt de l'approche présentée apparait ici. Une architecture multi-agents MAPS peut en effet être vue de façon duale comme "centrée connaissance" ou "centrée tâche". Cette dualité est présente dans les systèmes à base de tableau noir, la différence vient ici de la distribution de la structure de tableau noir au sein d'agents (nos KS) : chaque agent KS correspond alors à l'un des niveaux d'abstraction représenté dans cette structure. Une autre différence vient de la distribution du contrôle dans les agents : il s'exerce ainsi au niveau connaissance et au niveau tâche.

2.2. Modes et Protocoles de Communication

Au niveau communication, nous avons vu dans le paragraphe sur les communications des agents (paragraphe B.II.1.3), que deux types de messages pouvaient être échangés entre deux agents : des messages d'information et des messages de contrôle. A l'aide des messages d'information, un agent KP partage une information (instance) avec un agent KS, ils conversent ainsi sur un objet de discussion commun : l'agent KP pourra étoffer cette information en transmettant de nouvelles valeurs.

C'est ce niveau d'échange de messages qui rapproche notre système des approches de type tableau noir. En effet, dans des approches classiques, où des agents hétérogènes communiquent entre eux, un mécanisme de traduction est souvent employé afin de passer d'un mode de représentation à un autre. Dans notre approche, les connaissances échangées sont stockées dans les agents KS et sont partagés par des agents KP sans passer par un tel mécanisme de traduction.

Par le biais des messages de focalisation, un agent KP modifie les environnement de travail, de nouvelles instances peuvent être créées, d'autres détruites et l'instance courante ou objet de discussion peut être modifiée. Enfin, par le biais des messages de contrôle, un agent KP connait les actions de ses accointances, c'est-à-dire les actions du groupe dont il fait partie.

2.3. Modes d'Organisation et de Coopération

La version actuelle de MAPS offre des possibilités variées de programmation en ce qui concerne la manipulation des connaissances et le contrôle. Des éléments de connaissances hétérogènes peuvent être distribués parmi plusieurs agents et alors manipulés d'une manière spécifique dans un contexte bien modélisé.

Le contrôle est distribué au sein des agents KS et KP et s'exerce au niveau connaissance et tâche. Dans un agent KS, il s'agit de contrôler les flux d'information et leur circulation par les différents points de vue ou niveau d'abstraction. Il s'agit également de sélectionner les tâches à effectuer : le contrôle s'exerce ici en terme de planification. Dans un agent KP, il s'agit de générer, d'exciter ou d'inhiber des mondes de pensées. Des stratégies de contrôle orientées données ou orientées but peuvent ainsi être dégagées. Le basculement entre ces stratégies de contrôle s'effectue de manière très opportuniste. Du point de vue KP, il peut intervenir durant l'activation d'une règle, lors de la requête d'une valeur d'attribut nécessaire. Du point de vue KS, au contraire, il est plutôt contrôlé à un haut niveau "comportemental" (méthode Recevoir), au moyen d'un basculement entre l'activation des règles de Proposition et d'Interrogation.

L'approche de la résolution de problème apparait finalement comme émergeant de l'application de stratégies de résolution diverses et locales, dirigées individuellement par les agents eux-mêmes. Les agents se comportent alors comme des superviseurs locaux. Chaque agent possède à un instant donné un rôle central dans l'organisation, qui se traduit par son intentionnalité de communication. Les agents KS ont le souhait de placer une instance dans un monde de pensée différent, et les agents KP ont pour mission d'en répercuter les effets, par le biais de nouvelles perceptions. La puissance potentielle d'une telle approche sera démontrée par la suite avec une application en vision par ordinateur.

III LANGAGE DE PROGRAMMATION

Le langage de programmation MAPS est un langage hybride, issu de plusieurs formalismes standards de programmation et de représentation des connaissances : procédures, classes et règles de production (**Fig. B.8**). Le système MAPS bénéficie donc des vocations des trois styles de programmation : programmation fonctionnelle, programmation objet et programmation logique et permet ainsi de concevoir des systèmes à base de connaissances hétérogènes, difficilement représentables sous un formalisme unique **[Ovalle 91]**.

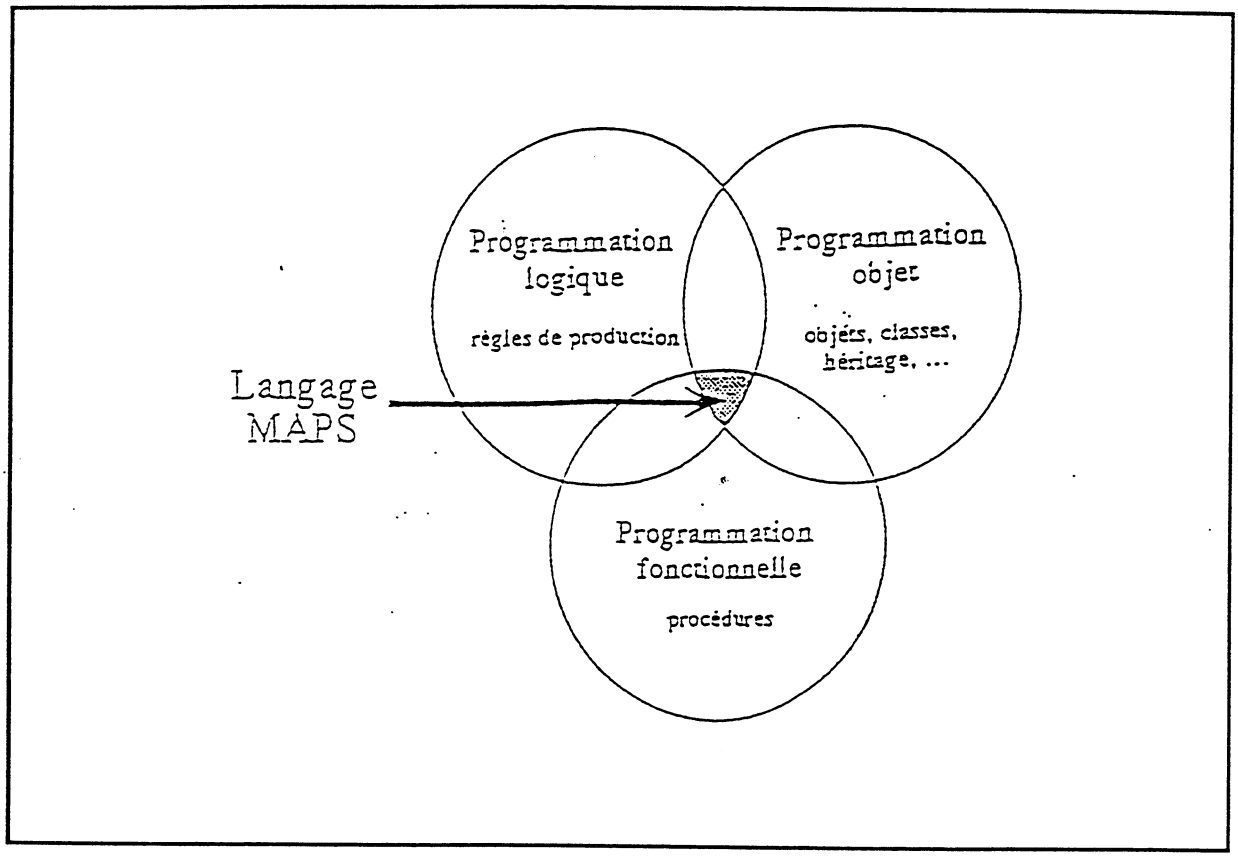


Fig. B.8. Situation du langage de programmation MAPS **[Ovalle 91]**

1. SYNTAXE DU LANGAGE

Une description détaillée de la syntaxe du langage de programmation MAPS peut être trouvée dans **[Baujard 90b]**. Nous nous contentons ici d'en rappeler les principales caractéristiques.

1.1. LES AGENTS

Un agent KS ou KP est défini de la manière suivante, les renseignements en italique étant facultatifs :

DEFINE_AGENT Nom1 : Nom2 : X : Y;
Définition des objets, règles et connexions.
END_AGENT;

Un agent possède une identification interne (Nom1), utilisée notamment dans les mécanismes de communication. Des renseignements d'ordre graphique, utilisés lors d'un représentation de l'agent dans un environnement graphique sont également fournis. Il s'agit du nom (Nom2) et du positionnement (coordonnées X et Y) à l'écran de l'agent.

1.2. Les Objets

Un objet est défini de la manière suivante :

DEFINE_CLASS Nom : *Père*;
Définition des attributs.
END_CLASS;

Un objet possède une identification interne (Nom) et peut éventuellement hériter des attributs d'un autre objet (Père). Des attributs propres à cet objet peuvent en outre être définis.

1.3. Les Attributs

Les attributs seront typés selon les trois types de données (entier, réel et chaîne de caractères) définis dans le langage MAPS. Un attribut sera ainsi défini de la manière suivante :

STRING Nom : *Valeur*;
INTEGER Nom : *Valeur*;
REAL Nom : *Valeur*;

On peut noter ici la richesse des types de données qui permet de représenter un grande variété de concepts. Le type "liste" peut en outre être simulé à l'aide des chaînes de caractères et de procédures de manipulation de chaînes adéquates : ces procédures sont des procédures externes applicables depuis une règle et permettent de simuler les opérateurs CAR, CONS, CDR, APPEND du Lisp.

1.4. Les Connexions

Une connexion définit un agent accointance de la manière suivante :

DEFINE_CONNEXION Nom;

1.5. Les Regles

Une règle sera définie de la manière suivante :

DEFINE_RULE Nom : Type;
Définition de la prémisse et de la conclusion.
END_RULE;

Une règle possède une identification interne (Nom) et un type (Propagation, Proposition, Interrogation, Action et Strategy). La syntaxe des règles du langage MAPS est présentée ci-

dessous sous forme BNF (Backus Nohr Form).

```
< Règle >            ::=  < Prémisse > < Conclusion >
< Prémisse >         ::=  < EXPP >
< Conclusion >       ::=  < EXPP >

< EXPP >             ::=  (OR < EXPP > < EXPP >)
                          | (AND < EXPP > < EXPP >)
                          | (CASE ( < SYMB > ) < EXPP > )
                          | (DEFINE_PARAMETER < EXP > ( < SYMB > ))
                          | (PROCEDURE ( < Nom-Procédure > ) < Liste-Paramètres >)
                          | (METHOD ( < Nom-Méthode > ) < Liste-Paramètres >)
                          | (REPEAT ( < SYMB > < SYMB > < SYMB > ) < EXPP > )
                          | (SEARCH ( < SYMB > ) < EXPP > )
                          | (SEARCH ( < SYMB > < SYMB > ) < EXPP > )
                          | (QUOTE < SYMB > )
                          | (QUOTE < SYMB > < SYMB > )
                          | (QUOTE < SYMB > < SYMB > < SYMB > )
                          | < EXP >
                          | < EXPBOOL >

< EXP >              ::=  (STRING < Valeur > )
                          | (INTEGER < Valeur > )
                          | (REAL < Valeur > )
                          | ( < SYMB > )
                          | ( < SYMB > < SYMB > )
                          | ( < SYMB > < SYMB > < SYMB > )
                          | < EXPARIT >

< EXPARIT >          ::=  ( < OPE > < EXP > < EXP > ) |
                          (FUNCTION ( < Nom-Function > ) Liste-Paramètres)

< EXPBOOL >          ::=  (< COMPAR > < EXPP > < EXPP >)

< COMPAR >           ::=  < | ≥ | ≤ | > | <> | = | ELEMENT | NELEMENT

< OPE >              ::=  + | - | * | /

< Liste-Paramètres >  ::=  < SYMB >
                          | <Liste-Paramètres > < SYMB >
< SYMB >             ::=  Chaîne de caractères
< Nom-Procédure >    ::=  Chaîne de caractères
< Nom-Function >     ::=  Chaîne de caractères
< Nom-Méthode >      ::=  Chaîne de caractères
< Valeur >           ::=  entier | réel | chaîne de caractères
```

Les parties prémisse et conclusion d'une règle sont des disjonctions et / ou des conjonctions d'expressions. Ces expressions peuvent impliquer des définitions de variables locales à la règle (DEFINE_PARAMETER), des appels de procédure (PROCEDURE) ou de fonction (FUNCTION) externes, des recherches d'instances vérifiant un certain critère (SEARCH), des opérations logiques ou arithmétiques, des boucles (REPEAT) et des appels de méthodes (METHOD). Une valeur d'attribut est référencée dans un agent KS par le doublet (Objet Attribut) et dans un agent KP par le triplet (Agent Objet Attribut). Chaque expression retourne le

résultat de son évaluation (NIL ou une valeur différente). Les parties prémisse et conclusion d'une règle peuvent ainsi être assez complexes, introduisant des possibilités supplémentaires de contrôle.

Une prémisse sera vérifiée si le résultat de son évaluation est différent de NIL. Dans le cas d'une conclusion, l'évaluation sera stoppée si l'évaluation d'une sous-partie renvoit NIL, à moins d'utiliser des disjonctions **(Exemple 1)**.

```
(AND  Exp1                                    (OR   Exp1
(AND  Exp2  -->  Si NIL, l'évaluation         (OR   Exp2  -->  Si NIL, l'évaluation
(AND  Exp3       de la conclusion est         (OR   Exp3       continue.
      Exp4)))    stoppée.                           EXP4)))
```

Exemple 1. Mode d'évaluation d'une conclusion de règle.

2. EXEMPLE D'UTILISATION

Ecrire une application revient à écrire un certain nombre d'agents, c'est-à-dire définir des objets, des connexions et des règles, dans des fichiers de définition (fichiers de ressources). Nous allons illustrer dans ce qui suit la conception des agents KS et KP concernant une application simple de généalogie. Nous donnerons pour chaque agent quelques règles en exemple : l'application complète pourra être trouvée en annexe.

2.1. PRÉSENTATION DE L'APPLICATION

Afin de construire l'arbre généalogique d'une famille, un généalogiste consulte en particulier les états-civil des villes où ont vécu des membres de cette famille, afin d'y collecter les noms des personnes ayant un lien de parenté. Ces états-civil se trouvent dans les mairies et les archives régionales, et ces organismes peuvent aider le généalogiste dans ses recherches en gardant une trace des personnes recherchées et en l'avertissant en cas de découverte de renseignements à leur sujet (suite à des mises à jour de l'état-civil par exemple).

Nous allons définir deux agents KS et un agent KP **(Fig. B.9)**. Les agents KS joueront le rôle d'un Etat-Civil (KS Etat-Civil) et d'un Arbre Généalogique (KS Arbre). L'agent KP connecté à ces deux agents jouera le rôle du généalogiste (KP Généalogiste) chargé d'obtenir un arbre à partir d'un état-civil. Nous obtenons ainsi le réseau suivant:

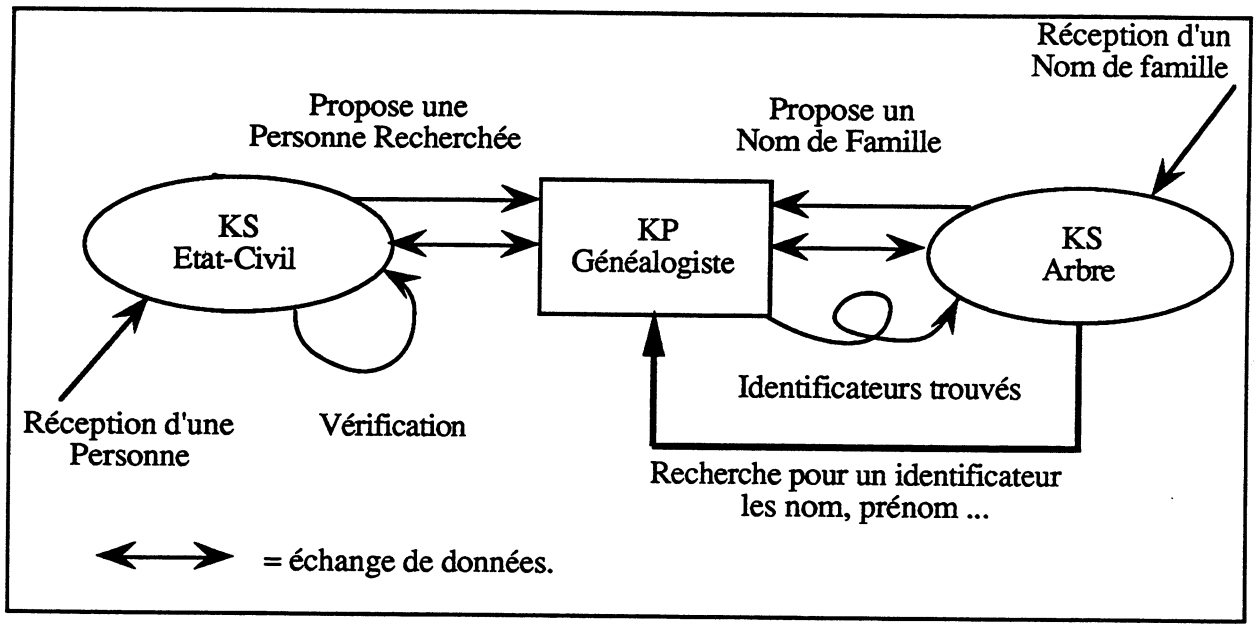


Fig. B.9. L'application de généalogie: agents et communications.

L'agent Etat-Civil contient l'objet "Personne". Il doit maintenir des instances de cet objet et donc posséder des mécanismes de mise à jour et de vérification de la cohérence. Il doit aussi se souvenir des renseignements que l'agent KP Généalogiste a pu lui demander; c'est pourquoi il contient aussi l'objet "Personne-Recherchée", afin de pouvoir proposer à l'agent KP Généalogiste toute instance de "Personne" ajoutée à la base, et qui correspond à une personne recherchée.

L'agent Arbre possède comme ressource l'objet "Membre-Famille". Il doit maintenir des instances de cet objet et possède donc des mécanismes de mise à jour et de complétion. Il est chargé de proposer à l'agent KP Généalogiste un nom à partir duquel l'arbre sera construit. C'est pourquoi il possède également comme ressource l'objet "Nom-Famille".

Enfin, l'agent KP Généalogiste est chargé de faire le lien entre les deux agents KS et doit construire l'arbre recherché, répondre à des demandes de la part de ces agents et leur transmettre les informations qu'il a obtenues.

2.2. Les Agents KS

Nos agents KS seront définis par les fichiers de ressources suivants :

```
DEFINE_AGENT    Etat-Civil;
DEFINE_CLASS Personne : ;
STRING      Nom : Vide;
STRING      Prénom : Vide;
STRING      Naissance : Vide;
STRING      Décès : Vide;
INTEGER     Identificateur : 0;
INTEGER     Père : 0;
INTEGER     Mère : 0;
INTEGER     Déjà-Proposé : 0;
END_CLASS;

DEFINE_CLASS    Personne-Recherchée : ;
STRING      Liste-Noms : ();
END_CLASS;

DEFINE_CONNEXION Généalogiste;

            Introduire ici la définition des règles.
END_AGENT;
```

```
DEFINE_AGENT    Arbre;
DEFINE_CLASS    Membre-Famille : ;
STRING      Nom : Vide;
STRING      Prénom : Vide;
INTEGER     Identificateur : 0;
INTEGER     Père : 0;
INTEGER     Mère : 0;
STRING      Grand-Père : ();
STRING      Grand-Mère : ();
END_CLASS;

DEFINE_CLASS    Nom-Famille : ;
STRING      Nom : ;
END_CLASS;

DEFINE_CONNEXION Généalogiste;

END_AGENT;
```

Nous allons à présent définir quelques unes des règles des agents KS, afin d'illustrer le style de programmation impliqué par le langage MAPS.

Règles de Propagation

Une règle de propagation est chargée de déduire d'une valeur d'attribut réceptionnée, les valeurs d'autres attributs qui peuvent en dépendre. Une règle typique sera la déduction dans l'agent Arbre, des attributs "Grand-Père" et "Grand-Mère" d'une instance de "Membre-Famille". Cette règle illustre la focalisation figurative locale effectuée par un agent KS à l'aide de l'instruction SEARCH, qui lui permet de sélectionner une instance vérifiant un certain critère.

```
1     DEFINE_RULE Trouve-Grand-Père : PROPAGATION;
2     (AND (<> (Membre-Famille Identificateur) (INTEGER 0))
3     (AND (DEFINE_PARAMETER (Membre-Famille Identificateur) (Fils))
4     (AND (DEFINE_PARAMETER (Membre-Famille Père) (Père))
5     (AND (DEFINE_PARAMETER (Membre-Famille Mère) (Mère))
6     (AND (DEFINE_PARAMETER (STRING ()) (Résultat))
7     (AND (OR    (SEARCH (Membre-Famille)
8                 (AND (= (Membre-Famille Identificateur) (Père))
9                 (AND (DEFINE_PARAMETER (Membre-Famille Père) (GP))
10                      (PROCEDURE (Ajoute-Liste) (GP Résultat)))))
11                (SEARCH (Membre-Famille)
12                (AND        (= (Membre-Famille Identificateur) (Mère))
13                (AND (DEFINE_PARAMETER (Membre-Famille Père) (GP))
14                      (PROCEDURE (Ajoute-Liste) (GP Résultat)))))
15                (SEARCH (Membre-Famille)
16                (= (Membre-Famille Identificateur) (Fils)))))))))));
17    (Membre-Famille Grand-Père Résultat);
18    END_RULE;
```

But : Pour un membre de la famille donné, rechercher parmi les autres membres de la famille son père et sa mère et collecter leur père, c'est-à-dire ses grand-pères.

En prémisse de cette règle, on trouve les valeurs des attributs "Identificateur", "Père" et "Mère" de l'instance courante de l'objet "Membre-Famille". Une recherche est alors effectuée (7) dans la branche paternelle, afin de trouver une instance dont la valeur de l'attribut "Identificateur" correspond à la valeur Père (8), cette instance correspond alors à l'instance Père de l'instance de départ. L'instance courante est alors remplacée par cette instance et la valeur de l'attribut "Père" de cette instance, c'est-à-dire l'identificateur du grand-père de l'instance de départ est alors collectée (9) et ajoutée à la liste des grand-pères (10). Une seconde recherche dans la branche maternelle est ensuite effectuée selon le même principe (7-15). En fin de prémisse, une dernière recherche (16) permet de se repositionner sur l'instance de départ. Le paramètre "Résultat" contient alors les grand-pères connus et l'attribut "Grand-Père" est mis à jour en conclusion (17).

L'agent KS Etat-Civil possèdera en outre la règle de propagation suivante:

```
1       DEFINE_RULE Obtient-Ident : PROPAGATION;
2       (AND  (<> (Personne Père) (INTEGER 0))
3       (AND  (<> (Personne Mère) (INTEGER 0))
4       (AND  (= (Personne Identificateur) (INTEGER 0))
5       (AND  (DEFINE_PARAMETER (Personne Identificateur) (Id))
6             (PROCEDURE (Obtient-Ident) (Id))))));
7       (Personne Identificateur Id);
8       END_RULE;
```

But : Pour une personne ne possédant pas d'identificateur, en obtenir un par un appel à la procédure Obtient-Ident

Cette règle illustre l'appel d'une procédure depuis une prémisse. Elle est chargée d'obtenir un numéro d'identification unique (appel de la procédure Obtient-Ident en 6) et de l'affecter à l'attribut "Identificateur" de l'instance courante de l'objet "Personne" (7). Cette affectation ne peut se faire que si tous les autres attributs de l'instance sont connus (nous dirons que l'instance est valide). Nous supposerons pour cela, que les attributs Père et Mère ont forcément une valeur non nulle dans ce cas : tests (2) et (3). La ligne (5) initialise un paramètre (Id), devant contenir le numéro d'identificateur.

Une règle de propagation peut aussi servir à maintenir la cohérence dans une base d'objets d'un agent KS. Dans notre application, il est ainsi nécessaire d'éviter des redondances au niveau des instances. A la réception d'une instance de "Personne", il faudra par exemple vérifier qu'elle n'est pas déjà présente au sein de la base, auquel cas il faudra détruire l'instance créée. La règle suivante effectue ce travail et illustre le mécanisme de manipulation des instance d'objet dans un agent KS (méthode DELETE_CURRENT_INSTANCE).

```
1     DEFINE_RULE Cohérence : PROPAGATION;
2     (AND  (<> (Personne Identificateur) (INTEGER 0))
3     (AND  (DEFINE_PARAMETER (Personne Identificateur) (Id))
4     (AND  (DEFINE_PARAMETER (Personne Nom) (Nom))
5     (AND  (DEFINE_PARAMETER (Personne Prénom) (Prénom))
6     (AND  (DEFINE_PARAMETER (Personne Père) (Père)
7     (AND  (DEFINE_PARAMETER (Personne Mère) (Mère))
8     (AND  (SEARCH (Personne)
9           (AND  (<> (Personne Identificateur) (Id))
10          (AND  (= (Personne Nom) (Nom))
11          (AND  (= (Personne Prénom) (Prénom))
12          (AND  (= (Personne Père) (Père))
13                (= (Personne Mère) (Mère)))))))
14          (SEARCH (Personne) (= (Personne Identificateur) (Id)))))))));
15    (METHOD (DELETE_CURRENT_INSTANCE) (Etat-Civil Personne));
16    END_RULE;
```

But : si l'instance courante de "Personne" correspond à une personne ayant les mêmes caractéristiques et déjà stockée dans la base de l'agent, alors détruire cette instance.

L'information la plus ancienne est ici privilégiée, mais la gestion aurait pu être différente. En prémisse, les valeurs des attributs de l'instance courante sont collectées (3 à 7) puis une recherche (8) est effectuée afin de trouver, si elle existe, une autre instance ayant les mêmes attributs (9 à 13). Si la recherche réussit, l'instance courante sera remplacée par cette seconde instance. En fin de prémisse (14), une seconde recherche permet de se repositionner sur l'instance de départ qui est alors détruite en conclusion (15).

Règles de Proposition

L'agent Etat-Civil doit proposer à l'agent Généalogiste des Personnes qui sont recherchées (leur nom fait alors partie de la liste maintenue dans l'attribut "Liste-Noms" de l'objet "Personne-Recherchée"). Il faut toutefois qu'elles n'aient pas déjà été proposées (elles sont alors déjà instanciées dans l'agent Arbre). La règle suivante effectue cette opération et illustre la focalisation figurative locale (sélection d'une instance particulière) et la focalisation opératoire locale (sélection de l'agent KP auquel la transmettre). Un appel à l'historique de l'agent est également explicité (méthode LAST_PROPOSITION).

```
1       DEFINE_RULE Propose-Personne-Recherchée : PROPOSITION;
2       (SEARCH (Personne)
3       (AND  (= (METHOD (LAST_PROPOSITION) (Etat-Civil Personne)) (STRING NO))
4       (AND  (= (Personne Déjà-Proposé) (INTEGER 0))
5       (AND  (<> (Personne Identificateur) (INTEGER 0))
6       (AND  (DEFINE_PARAMETER (Personne Nom) (Nom))
7             (SEARCH (Personne-Recherchée)
8             (ELEMENT (Nom) (Personne-Recherchée Liste-Noms))))))));

9       (QUOTE Généalogiste Personne Nom);
10      END_RULE;
```

But : Chercher une personne qui n'a pas encore été proposée à l'agent Généalogiste et qui fait partie des personnes recherchées. Si elle existe, la proposer à l'agent Généalogiste.

En partie prémisse, une instance qui n'a pas déjà été proposée est recherchée : la méthode LAST_PROPOSITION doit alors renvoyer NO (3). Si de plus, tous les attributs de cette instance sont corrects (5), et si l'attribut "Nom" fait partie de la liste Liste-Noms (8), une proposition pourra être effectuée en conclusion, à l'agent KP Généalogiste (9).

Pour l'agent Arbre, une règle de proposition servira à poser le problème initial à l'agent KP Généalogiste. L'attribut "Nom" de l'objet "Nom-Famille" contiendra le nom de départ de notre arbre généalogique. C'est la mise à jour de cet attribut par le biais d'une réception (voir schéma) qui déclenchera toute la construction de l'arbre (en effet, comme nous l'avons décrit, une réception entraîne une mise à jour, puis l'obtention d'une information à proposer (ici l'attribut "Nom" de l'objet "Nom-Famille"). Cette règle sera:

```
1       DEFINE_RULE Famille-a-Trouver : PROPOSITION;
2       (= (METHOD (LAST_PROPOSITION) (Arbre Nom-Famille)) (STRING NO));

3       (QUOTE  Généalogiste Nom-Famille Nom);
4       END_RULE;
```

But : Si l'instance courante de "Personne" n'a pas encore été proposée, la proposer à l'agent Généalogiste.

La prémisse de cette règle vérifie que le système n'est pas déjà activé, c'est-à-dire que l'information n'a pas déjà été proposée à l'agent KP Généalogiste et donc que la méthode LAST_PROPOSITION renvoit bien NO (2).

Règles d'Interrogation

Des règles d'interrogation seront employées, notamment dans l'agent KS Arbre. L'agent KP Généalogiste comme nous allons le voir plus loin va transmettre uniquement l'identificateur d'une personne à l'agent KS Arbre. Lors de la réception de la valeur de l'attribut, tous les autres attributs seront donc inconnus. Les règles de recherche seront donc chargées d'obtenir auprès de l'agent KP Généalogiste les valeurs de ces attributs. Une règle de ce type sera:

```
1    DEFINE_RULE Demande-Nom : INTERROGATION;
2    (= (Membre-Famille Nom) (STRING Vide)));
3    (QUOTE Généalogiste Membre-Famille Nom);
4    END_RULE;
```

But : Si le nom de l'instance courante est inconnu, le demander à l'agent Généalogiste.

La prémisse vérifie ici que les renseignements n'ont pas déjà été demandés, auquel cas la méthode LAST_INTERROGATION doit renvoyer YES (2) et que l'attribut "Nom" est inconnu (3). Cet attribut est alors demandé à l'agent Généalogiste en conclusion (4).

2.3. L'AGENT KP

Comme les agents KS, l'agent KP Généalogiste est défini par un fichier de ressources. Nous allons définir quelques unes de ses règles.

Règles d'Action

L'agent KP Généalogiste possède plusieurs règles d'actions. Nous allons détailler la principale, les autres étant seulement présentées : leur définition pourra être trouvée en annexe. La règle présentée illustre le mécanisme de bouclage, et la recherche d'instance dans un agent KS, c'est-à-dire la focalisation sur une connaissance particulière.

```
1     DEFINE_RULE Trouve-Famille : ACTION;
2     (AND  (DEFINE_PARAMETER (Arbre Nom-Famille Nom) (Nom-Recherché))
3     (AND  (DEFINE_PARAMETER (INTEGER 0) (Nbre-Personne))
4     (AND  (PROCEDURE (Get-Nbre-Index) (Nbre-Personne))
5     (AND  (DEFINE_PARAMETER (STRING  ()) (Liste-Index))
6     (AND  (DEFINE_PARAMETER (STRING  ()) (Liste-Nom))
7     (AND  (DEFINE_PARAMETER (INTEGER 1) (Début))
8     (AND  (PROCEDURE (Ajoute-Liste) (Nom-Recherché Liste-Nom))
9           (REPEAT (I Début Nbre-Personne)
10          (OR    (SEARCH (Etat-Civil Personne)
11                 (AND  (OR (= (Etat-Civil Personne Nom) (Nom-Recherché))
12                       (SEARCH (Etat-Civil Personne)
13                       (AND (ELEMENT (Etat-Civil Personne Identificateur)
                                       (Liste-Index))
14                       (AND (DEFINE_PARAMETER (Etat-Civil Personne Nom)
                                                (Nom))
15                       (AND (NELEMENT (Nom) (Liste-Nom))
16                       (AND (DEFINE_PARAMETER (Nom) (Nom-Recherché))
17                            (PROCEDURE (Ajoute-Liste) (Nom Liste-Nom))))))))
18                 (AND (DEFINE_PARAMETER (Etat-Civil Personne Identificateur)
19                                        (Ident1))
20                 (AND  (DEFINE_PARAMETER (Etat-Civil Personne Père) (Ident2))
21                 (AND  (DEFINE_PARAMETER (Etat-Civil Personne Mère) (Ident3))
22                 (AND  (NELEMENT (Ident1) (Liste-Index))
23                 (AND  (PROCEDURE (Ajoute-Liste) (Ident1 Liste-Index))
24                 (AND  (PROCEDURE (Ajoute-Liste) (Ident2 Liste-Index))
25                       (PROCEDURE (Ajoute-Liste) (Ident3Liste-Index))))))))))))))));
```

But : Rechercher parmi toutes les instances de "Personne" celles qui possède le nom recherché (branche paternelle) et celles qui porte le nom des branches maternelles.

Nous avons vu que l'agent KS Arbre allait proposer à l'agent KP Généalogiste d'exploiter l'attribut "Nom" de l'objet "Nom-Famille" afin d'obtenir l'arbre généalogique correspondant. L'agent Généalogiste possède donc une règle chargée de ce travail. Cette règle doit rechercher (10) parmi les instances de l'objet "Personne" dans l'agent Etat-Civil, celle dont l'attribut "Nom" correspond au nom recherché, stocké dans un paramètre (1) (on suppose ici pour simplifier que toutes les personnes ayant le même nom appartiennent effectivement à la même famille). Un paramètre est aussi maintenu, qui s'occupe d'une liste de noms initialisés avec le nom recherché (6 et 8).

La valeur de l'attribut "Identificateur" des instances qui vérifient cette condition est alors rangée dans une liste (17). Cette recherche doit s'effectuer sur toutes les instances, c'est pourquoi la règle mettra en oeuvre une répétition sur le nombre d'instance de Personne (9) (nous supposons qu'une procédure nous donne ce nombre). Les identificateurs de Père et Mère sont aussi ajoutés dans la liste (18 et 19). Chaque répétition fournit ainsi l'identificateur d'un membre de la famille et ceux de ses parents.

Si la recherche échoue, cela signifie que toutes les instances dont le Nom correspond au Nom-Recherché ont été trouvées. Mais le travail n'est pas terminé, il faut aussi tenir compte des ascendants maternels dont le nom est différent. C'est pourquoi on va mettre à jour le paramètre Nom-Recherché avec un nouveau nom. Ce nom est obtenu en effectuant une seconde recherche d'instance dont l'identificateur appartient à la liste des identificateurs (21) (c'est un membre de la famille) et dont le nom n'a pas déjà été étudié (24) (il n'est pas présent dans la liste des noms).

Si une telle instance est trouvée, son Nom est placé dans le paramètre Nom-Recherché (25) et ajouté à la liste des noms (26). La recherche peut alors continuer pour ce Nom. Le fonctionnement est illustré sur les données suivantes :

Instances de Personne

Personne1	Personne2	Personne3	Personne 6
Nom : DUPOND	Nom : DUPOND	Nom : DURAND	Nom : DURAND
Identificateur : 1	Identificateur : 2	Identificateur : 3	Identificateur : 6
Père : 2	Père : 4	Père : 6	Père : 8
Mère : 3	Mère : 5	Mère : 7	Mère : 9

Si le nom de famille est DUPOND, au premier passage de boucle les paramètres auront les valeurs suivantes:

Liste-Nom contient DUPOND.
Liste-Index contient 1 2 et 3. Passage dans le premier chercher.

Au second passage, les paramètres auront les valeurs :

Liste-Nom contient DUPOND DURAND. Passage dans le second chercher.
Liste-Index contient 1 2 et 3.

Au troisième passage, les paramètres auront les valeurs :

Liste-Nom contient DUPOND DURAND.
Liste-Index contient 1 2 3 6 8 et 9.

L'agent Généalogiste doit alors transmettre les résultats de sa recherche (les identificateurs de la liste) à l'agent Arbre en mettant à jour l'attribut Identificateur de Membre-Famille (33). Pour cela en conclusion, nous trouvons une répétition sur la liste des identificateurs (27). Cette mise à jour doit créer des instances, c'est pourquoi la procédure système RESET_CURRENT_INSTANCE sera utilisée (31). La conclusion suivante illustre également l'envoi répétitif d'identificateur et la création d'instances :

```
27    (AND  REPEAT (I ELEMENT Liste-Index)
28          (AND  (SEARCH (Etat-Civil Personne)
29                      (= (Etat-Civil Personne Identificateur) (I)))
30          (AND  (Etat-Civil Personne Déjà-Proposé I)
31          (AND  (METHOD     (RESET_CURRENT_INSTANCE)
32                            (Arbre Membre-Famille))
33                      (Arbre Membre-Famille Identificateur I)))
34          (Etat-Civil Personne-Recherchée Liste-Noms Liste-Nom)))));
35    END_RULE;
```

But : Complétion de l'environnement attaché à l'agent KS Arbre. Pour toutes les personnes trouvées, indiquer qu'elle a déjà été examinée et créer dans l'arbre généalogique l'entrée correspondante. Mettre également à jour la liste des personnes recherchées.

Notons aussi dans cette conclusion la mise à jour de l'attribut Liste-Noms de l'objet Personne-Recherchée avec le paramètre Liste-Nom (34). L'agent Etat-Civil est alors en mesure d'appliquer sa règle de proposition si à un moment donné, il connait une instance de "Personne" dont le Nom correspond à un nom recherché. L'attribut Déja-Proposé de chaque instance maintenue dans la liste des identificateurs est lui aussi mis à jour (30). Ainsi, l'agent Etat-Civil ne pourra pas proposer cette instance, son identificateur aura en effet déjà été transmis à l'agent Arbre, puisqu'il faut que l'attribut Déjà-Proposé soit nul (voir la règle de proposition).

Une autre règle d'action sera mise en oeuvre suite à la proposition par l'agent KS Etat-Civil d'une personne recherchée. Dans ce cas, les valeurs des attributs de la Personne seront collectés en Prémisse et transmis en conclusion à l'agent KS Arbre.

Enfin, une dernière règle sera chargée de trouver la valeur d'un attribut d'une Personne et de la transmettre à l'agent KS Arbre qui l'avait demandé par le biais d'une règle de recherche.

Règles de Stratégie

Notre agent KP Généalogiste possède trois règles d'action. Nous pouvons définir en conclusion de chacune d'entre elles un mot-clé, par exemple pour la règle d'action Trouve-Famille, nous aurons la nouvelle conclusion:

```
27    (AND  (QUOTE TROUVER)
28    (AND  (REPEAT (I ELEMENT Liste-Index)
29          (AND  (SEARCH (Etat-Civil Personne)
30                (= (Etat-Civil Personne Identificateur) (I)))
31          (AND  (Etat-Civil Personne Déjà-Proposé I)
32          (AND  (METHOD (RESET_CURRENT_INSTANCE)
33                      (Arbre Membre-Famille))
34                (Arbre Membre-Famille Identificateur I)))
35          (Etat-Civil Personne-Recherchée Liste-Noms Liste-Nom)))));
36    END_RULE;
```

Reste alors à définir une règle de stratégie pouvant sélectionner cette règle à partir du mot-clé TROUVER. La règle de stratégie suivante illustre ce mécanisme avec la notion de contexte dans les stratégies : seule la partie prémisse dépendante du contexte est utilisée pour vérifier la règle de stratégie.

```
1     DEFINE_RULE Selectionne-Trouve-Famille : STRATEGIE;
2     (CASE (Nom-Famille)
3           (=    (METHOD (LAST_PROPOSITION) (Arbre Nom-Famille))
4                 (STRING Généalogiste)));
5     (QUOTE      TROUVER);
6     END_RULE;
```

But : Dans le cas d'instances de "Nom-Famille" et si ces instances ont été proposées à l'agent Généalogiste, alors utiliser la règle d'action contenant le mot clé TROUVER.

Dans cette règle on ne trouve que le contexte "Nom-Famille" (2). Ce contexte correspond à l'information proposée, qui est alors l'attribut "Nom" de l'objet "Nom-Famille" (règle de proposition de l'agent KS Arbre). La partie prémisse correspondante est alors vérifiée : l'information a bien été proposée à l'agent KP Généalogiste (3 et 4). Cette règle de stratégie sera alors vérifiée lors d'une proposition de l'objet Membre-Famille et permettra ainsi de sélectionner la règle d'action ayant en conclusion le mot clé TROUVER. Dans notre cas ce sera la règle Trouve-Famille.

Dans notre exemple, les règles de stratégie seraient particulièrement intéressantes si l'application était étendue avec plusieurs agents KS Etat-Civil (chaque agent correspondant à l'état-civil d'une ville). L'agent KP Généalogiste possèderait plusieurs règles d'actions exploitant les données de chacun de ces agents KS. Il serait alors possible de construire l'arbre généalogique à partir des états-civil maintenus dans chaque ville : les règles de stratégie sélectionneraient alors les règles d'action adaptées (les règles d'actions exploitant l'état-civil adéquate).

3. TECHNIQUES AVANCÉES DE PROGRAMMATION

De part la richesse du langage, des styles de programmation différents peuvent être développés. Ces styles de programmation ont des conséquences au niveau conception (distribution, nombre d'agents) et au niveau social (communication). Nous allons illustrer ces possibilités sur trois exemple : la synchronisation de la communication entre agents, l'ajustement de paramètre et la complétion de la connaissance.

Synchronisation

Il est possible de moduler le comportement des agent KS en introduisant des variables d'états (actif ou passif par exemple), qui seront testées en prémisse des règles de proposition et d'interrogation. Ces variables pourront ainsi en inhiber l'exécution. Selon son "état", l'agent développera alors des comportements différents : actif, l'agent aura pour rôle le simple stockage d'information, actif l'agent en plus de ce rôle de stockage, aura un rôle de propagation des informations vers l'extérieur.

Ces situations de blocage peuvent être modifiées à un niveau local par l'agent KS lui-même, par le biais de ses règles de propagation. L'agent a ainsi la connaissance de la pertinence de son rôle dans un contexte particulier. Il peut ainsi bloquer la communication entre certains agents en faisant de la rétention d'information.

Ces situations de blocage peuvent être également modifiées à un niveau plus global par un agent KP. Dans ce cas, l'agent KP à une vue globale de la résolution du problème, il sait que certains agents KS doivent être désactivés ou réactivés à la suite de la complétion d'un traitement particulier. Un aspect contrôle au niveau des groupes d'agents est alors démontré, l'agent KP ayant un rôle de superviseur.

Ajustement de Paramètre

Des astuces de programmation diverses peuvent être aussi utilisées, comme par exemple le contrôle externe de l'ajustement itératif d'un paramètre de procédure. Une règle d'un agent KP peut boucler en prémisse comme en conclusion sur l'application d'une procédure, jusqu'à ce qu'un paramètre de cette procédure vérifie un certain critère (**exemple 2**).

```
Répéter
        Procédure1(A)
Jusqu'à ce que A vérifie un certain critère.
```

Exemple 2. Boucle d'ajustement de paramètre.

Supposons à présent que deux procédures dépendantes l'une de l'autre (par exemple un filtre

suivi d'une procédure de segmentation) soient appliquées :

Procédure1(A,B) suivie de Procédure2(B,C).

Le paramètre B de la procédure 1 est utilisé par la procédure 2, et la valeur du paramètre C de cette procédure dépend de celle du paramètre B. Le problème est de modifier B jusqu'à l'obtention d'une valeur "optimale" de C. Deux solutions sont possibles.

Solution 1 :

Répéter
Mettre à jour A
Procédure1(A,B)
Procédure2(B,C)
Jusqu'à ce que C vérifie un certain critère.

Solution 2 :

Répéter
Effacer B dans un agent KS
Demander B à cet agent KS
Procédure2(B,C)
Jusqu'à ce que C vérifie un certain critère.

La solution 1 privilégie le caractère local de l'opération. Les deux procédures seront appliquées dans le même agent KP. La seconde solution privilégie le caractère distribué. Seule la deuxième procédure est appliquée dans l'agent KP initial. C'est la demande de la valeur de B à un agent KS qui entrainera son calcul : l'agent KS ne connaissant pas cette valeur (elle est détruite à chaque fois) en demandera le calcul à un agent KP en mode broadcast. La deuxième solution est plus souple, car elle permet de mieux distribuer les tâches au sein d'agent KP distincts. Dans notre cas, le deuxième agent KP peut posséder plusieurs procédures équivalentes (par exemple plusieurs filtres) et les stratégies nécessaires pour en choisir une, ce qui n'est pas possible avec la première solution.

Complétion de la Connaissance

L'échange entre un agent KS et un agent KP se fait par l'exploitation d'un environnement commun, autour de la notion d'instance courante. Les intentions de ces agents à cet égard sont les suivantes. Pour l'agent KS, il s'agit de compléter sa connaissance en collectant un maximum d'information sur l'instance courante (par le biais de ses règles d'interrogation). Pour l'agent KP, il s'agit d'exploiter l'environnement, la connaissance nécessaire et suffisante à l'activation des règles mises en jeu à un instant donné. Le conflit d'intention évident entre ces agents conduit à deux formes de programmation différentes.

La première solution se déroule à un niveau local. L'agent KS attend de tout connaître sur une

instance avant de la transmettre. L'agent KS suppose ainsi que l'agent KP aura besoin de toutes ces informations. Les mécanismes de complétion sont ici statiques, car codés dans des règles d'interrogation.

La seconde solution se déroule au niveau du groupe. L'agent KS transmet l'instance incomplète et laisse l'agent KP lui demander les informations nécessaires. Les mécanismes de complétion sont ici dynamiques, car déclenchés en mode "broadcast" suite à l'envoi d'un message "Collecter" par l'agent KP (se rapporter au comportement "Fournir" défini au paragraphe B.II.1.4).

Ces deux solutions ont ainsi des répercussions au niveau de la distribution des connaissances et au niveau de la communication entre les agents **(Fig. B.10)**

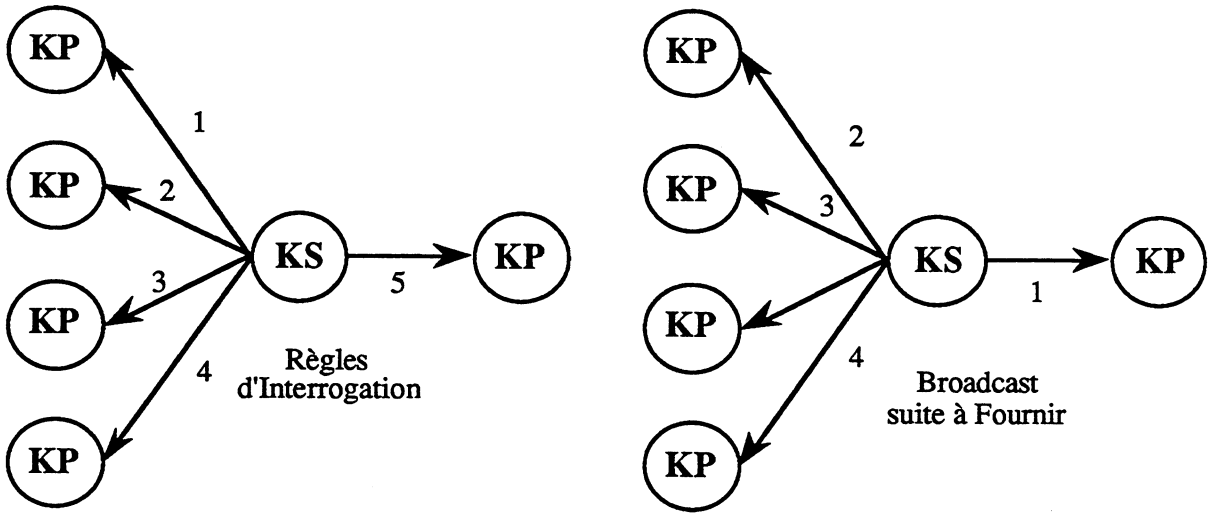


Fig. B.10. Différents échanges d'informations entre KS et KP, la communication 3 introduite dans la première solution n'est pas reprise dans la seconde (demande d'information superflue).

Le langage de programmation apparait finalement comme assez souple, plusieurs styles de programmation pouvant être utilisés. Cette souplesse permet de trouver le meilleur compromis entre la distribution des tâches et des connaissances, et la communication entre les agents. La distribution des tâches et des connaissances entraine en effet plus de souplesse, plus de modularité et permet de bâtir des architectures réellement coopératives mais entraine également un nombre de communications accru. La conception d'agents plus complexes permet de diminuer ce nombre de communications mais au détriment de la souplesse de la distribution.

IV IMPLANTATION

Une maquette du système MAPS a tout d'abord été réalisée en COMMON LISP sur station de travail Apollo **[Baujard 89a]**. Pour des raisons d'efficacité un portage a été décidé. Après une étude de différent logiciel du commerce comme KEE, ART ou HyperCard, nous avons décidé de construire notre propre environnement de programmation. Pour des raisons de standardisation, le langage retenu est C++ **[Stroustrup 87]** , et le système de fenêtrage X Window. La machine cible est une station de travail Apollo DN3550, mais du fait des standards retenus, l'environnement MAPS est portable sur d'autres stations de travail. Nous allons dans les paragraphes suivants décrire les choix d'implantation et les fonctionnalités de l'environnement de programmation MAPS.

1. LES AGENTS

Un ensemble de classes C++, organisées en hiérarchies **(Fig. B.11)** implante les caractéristiques des agents KS et KP.

Des classes sont définies, définissant le modèle C++ des ressources des agents ainsi que les mécanismes d'accès à ces ressources : Common (méthodes communes à toutes les classes), Attribute (attributs d'un objet), Object (un objet d'un agent KS), Element (les objets d'un agent KS), Rule (les règles) et Activity (les interpréteurs de règles). Des classes implantent les moteurs d'inférence (Engine, KS-Engine et KP-Engine) et les protocoles de communication entre les agents (KS-Interface et KP-Interface). Finalement, deux classes génériques sont définies à partir de cette hiérarchie de classes, qui définissent les agent KS et KP.

Le compilateur du langage de programmation est défini au niveau de ces classes. Comme nous l'avons déjà mentionné, les ressources de chaque agent sont définies dans un fichier externe privé. A chaque création d'agent, les classes génériques correspondantes (KS ou KP) sont instanciées et le fichier de ressource est alors compilé pour créer dynamiquement les représentations internes des objets et des règles : une instanciation des classes représentant les objets et les règles est alors effectuée. Pour les règles, le compilateur bâtit une représentation arborescente qui sera parcourue par l'interpréteur lors de l'exécution.

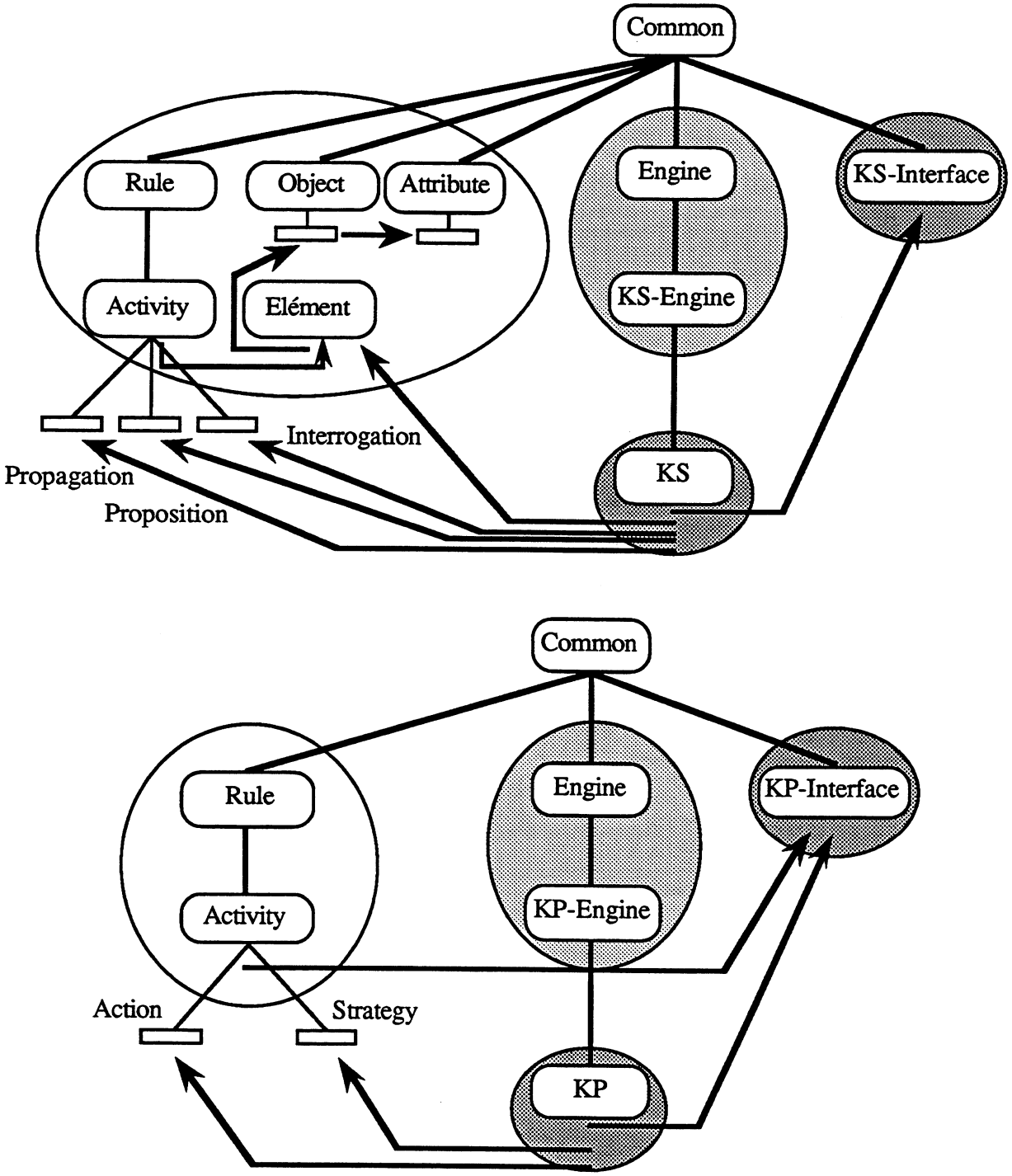


Fig. B.11. Hiérarchie des classes C++ définissant les classes génériques KS et KP

2. LE CONTROLEUR D'APPLICATION

L'environnement MAPS est constitué de deux processus UNIX distincts communicant à travers un protocole d'envoi de messages (**Fig. B.12**). Le premier processus ou contrôleur d'application est écrit en C++, il regroupe actuellement l'implantation des principales caractéristiques des agents présentées au paragraphe précédent : protocole d'envoi de message, comportement, moteurs d'inférence, interpréteur de règle, modèle de base des objets et procédures de traitement.

Ce processus est chargé de compiler les fichiers de ressources des agents et d'en bâtir une représentation interne. Il permet en outre à l'utilisateur d'accéder à cette représentation interne, d'accéder aux mécanismes de contrôle (trace, point d'arrêt) et aux fichiers de ressources. Le processus possède pour cela d'une interface graphique dédiée.

Le second processus est chargé des procédures externes et sera décrit au paragraphe 4.

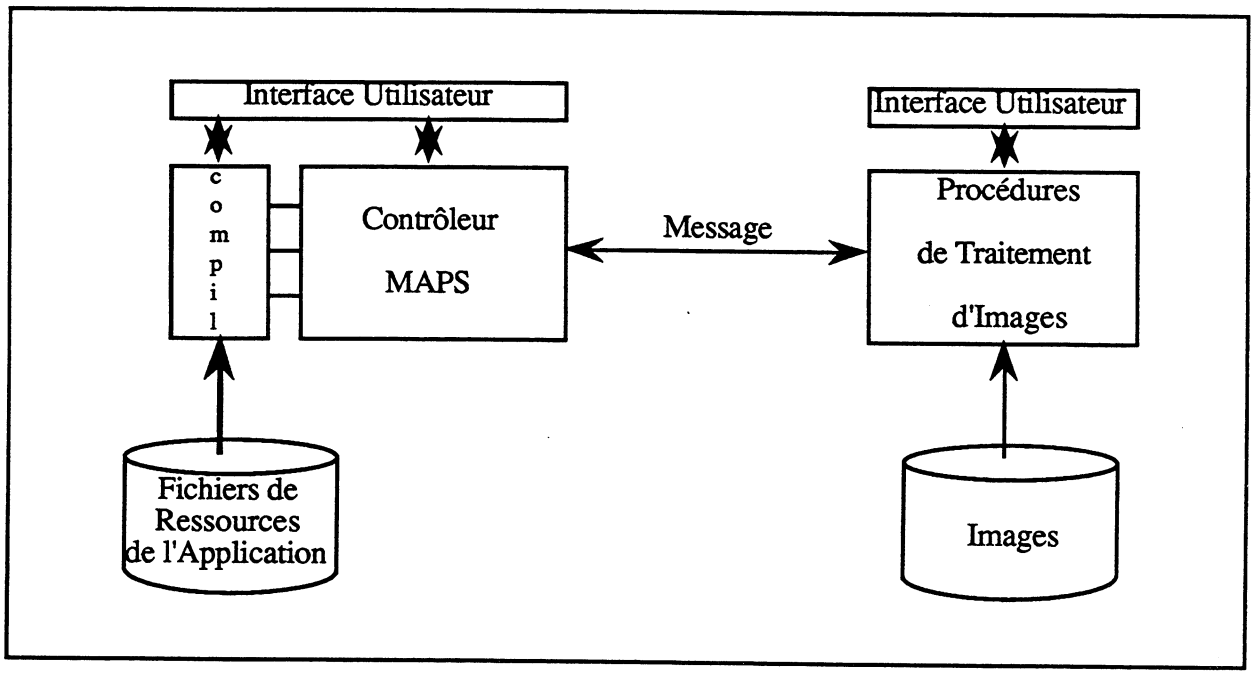


Fig. B.12. Architecture générale de l'environnement de programmation MAPS.

3. L'INTERFACE

Une interface X Window a été conçue pour développer des applications complètes. Cette interface est chargée de représenter une application, de tracer l'exécution d'un application et de visualiser les ressources des agents.

En terme de représentation, un réseau d'agent est affiché à l'écran (**Fig. B.13**). Il correspond à l'application chargée. Il représente les agents KS et KP et les connexions entre ces agents.

Ces connexions représentent les échanges possibles entre les agents. Ces échanges peuvent être visualisés en cours d'exécution grâce à un curseur graphique se déplaçant sur le réseau.

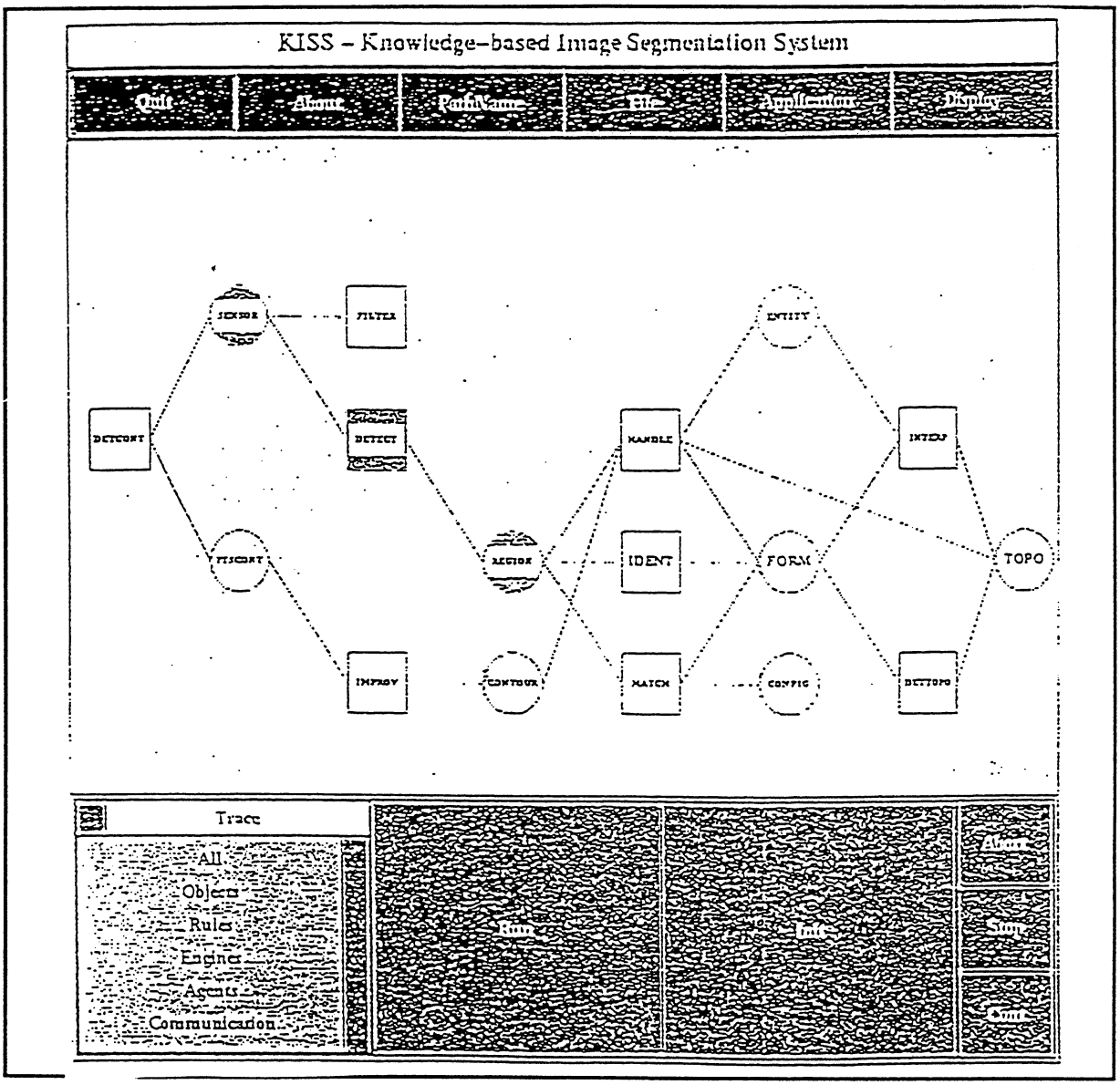


Fig. B.13. MAPS : vue générale de l'interface.

L'exécution d'une application peut être déclenchée à partir d'un "script" prédéfini (Run) et le système entier peut être réinitialisé (Init). Des points d'arrêt peuvent être introduits (Stop, Abort et Cont) et des niveaux de trace définis.

Au cours d'une interruption, le contenu des agents (instances crées et valeur d'attributs) peut être visualisé. Pour ce faire, une simple sélection, à l'aide de la souris, de l'agent souhaité sur le réseau d'agents, permet d'avoir accès à une boite de dialogue.

Cette boite de dialogue (**Fig. B.14**) est un véritable panneau de contrôle. Elle permet d'une

part de consulter les ressources de l'agent, de lire ou de modifier des valeurs d'attributs ou de changer l'instance de travail d'un objet. Elle permet d'autre part d'activer l'agent en déclenchant ses comportements sur des informations particulières. Elles permet enfin de définir des niveaux de trace et des points d'arrêt.

Un agent spécifique, l'agent USER, est introduit dans le cadre de l'environnement de programmation MAPS. Il est perçu comme un agent graphique, c'est-à-dire un agent possédant une apparence graphique et un comportement particulier. Son rôle s'apparente à celui d'un agent KP ; il n'est accessible qu'en résolution et son unique comportement est de poser le problème à l'utilisateur. En cours d'exécution, l'agent USER peut être activé par un agent KS, depuis une règle d'interrogation ou en mode broadcast, conduisant alors en réaction, à l'affichage de sa boite de dialogue dédiée présentée en figure **B.15**.

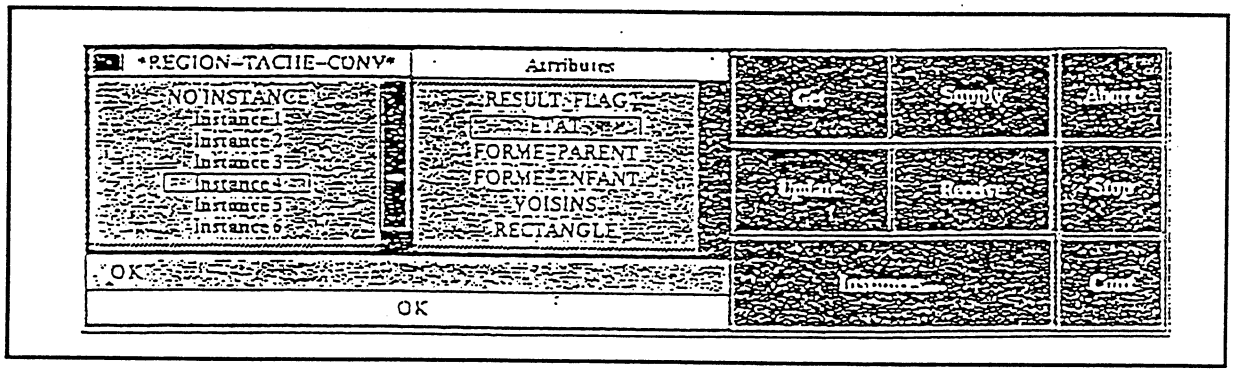


Fig. B.14. MAPS : vue du panneau de consultation des agents.

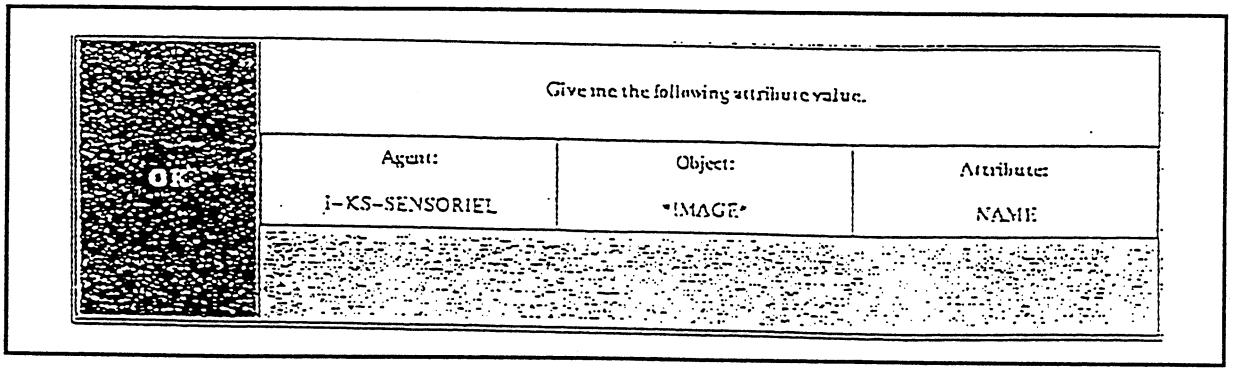


Fig. B.15. Agent USER : vue de la boite de dialogue.

4. Le Serveur de Procédures

Le second processus (**Fig. B.12**) manipule les procédures qui peuvent être appelées depuis les règles d'une application multi-agents et appliquées sur des paramètres donnés ; elles peuvent être écrites avec n'importe quel langage de haut niveau tel le Fortran, le Pascal ou le C. Ce processus est ainsi personnalisé selon le domaine du problème à étudier. Dans le cas de la

Vision par Ordinateur par exemple, il regroupe les procédures de traitement d'image faisant partie du logiciel IPS **[Chassery 86]**. Pour un autre domaine, comme le traitement du signal ou l'analyse de données par exemple, un processus différent regroupant des procédures dépendantes du domaine devra être construit. Les fichiers de ressources des agents seront alors utilisés comme un moyen pour spécialiser le système entier pour une application donnée comme l'analyse de fibres musculaires ou l'analyse de signaux électro-cardiographiques. La communication entre les deux processus s'effectue de manière synchrone : la requête contient le nom de la procédure à appliquer et les paramètres de cette procédure, la réponse contient le nom de la procédure et les paramètres résultats de cette procédure.

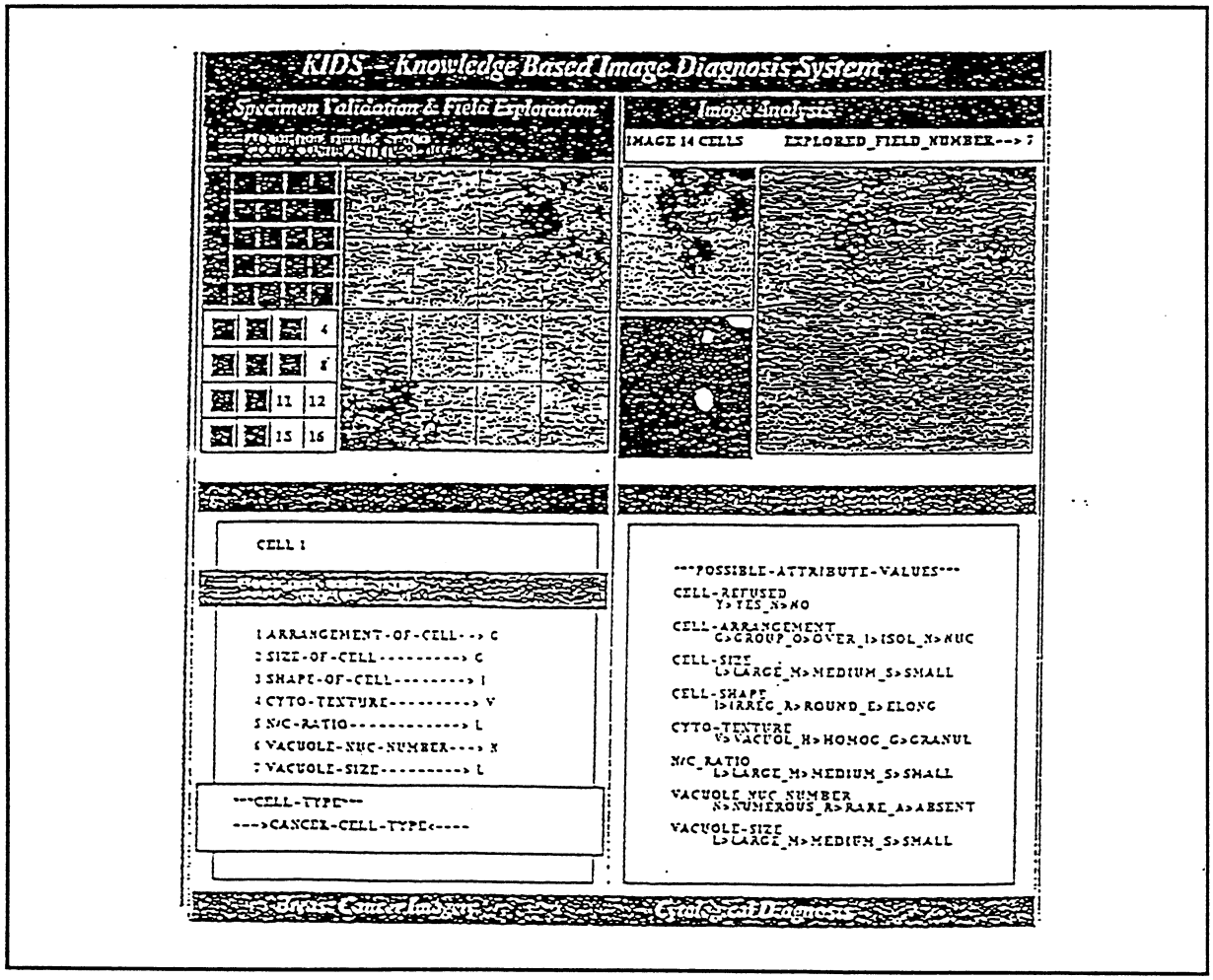


Fig. B.16. MAPS : Exemple d'interface liée au processus de gestion des routines externes (KIDS)

Une interface X Window dédiée est de plus attachée à ce processus pour afficher les résultats des traitements. Dans le cas de la Vision par Ordinateur, les images et les résultats de l'interprétation seront visualisés sur cette interface. Des procédures graphiques, définies dans le serveur de procédures externes sont définies à cette fin et peuvent être appelées depuis une règle de KP. La figure **B.16** représente l'interface de ce processus pour l'application de diagnostic biomédical KIDS **[Ovalle 91]**.

V CONCLUSION

L'approche multi-agents MAPS s'avère particulièrement intéressante de part sa souplesse au niveau distribution des connaissances, au niveau modularité et au niveau coopération. En ce qui concerne la distribution, l'intérêt de MAPS vient de la dualité tâches / connaissances. Un réseau d'agents MAPS peut en effet être approché selon deux points de vue : un point de vue "centré information" et un point de vue "centré tâche". Deux sortes de groupes d'agents peuvent ainsi coexister : des groupes "centré information" et des groupes "centré tâche". Cette double approche permet alors d'introduire des protocoles de coopération puissants et flexibles.

En ce qui concerne l'environnement de programmation, il se montre particulièrement intéressant de part sa souplesse au niveau configuration, au niveau langage de programmation et au niveau interface utilisateur. L'environnement peut être personnalisé pour un domaine d'application donné par la définition de processus de gestion des procédures externes, et pour une application donnée par la définition de ressources d'agents. Le langage MAPS de part sa syntaxe particulièrement riche, permet la création d'agents complexes possédant des comportements variés. Enfin, l'interface rend aisée la mise au point d'applications complexes et la visualisation de l'exécution d'une application.

Plusieurs applications ont ainsi été développées avec cet environnement, en Vision par Ordinateur avec le système KISS (voir Partie C), en Diagnostic Biomédical avec le système KIDS **[Ovalle 91]** et en Compréhension de la Parole **[Caillaud 91]**. Des travaux sont également développés autour de cet environnement, notamment en Acquisition des Connaissances **[Hugonnard 91]**, qui nous ont amené à faire la critique de cet environnement.

CONCLUSION PARTIE A

L'étude des principaux langages à objets et de l'approche cognitive des systèmes multi-agents a montré que ces deux domaines sont assez proches, les systèmes multi-agents empruntant beaucoup aux objets et surtout aux acteurs. La notion supplémentaire introduite par les systèmes multi-agents est l'autonomie de décision par rapport aux messages que ne possèdent pas les objets. Les systèmes multi-agents définissent en outre des agents et des types de coopération extrêmement divers.

Nous avons voulu ainsi bâtir un environnement de programmation qui intègre des caractéristiques des langages à objets et de l'Intelligence Artificielle Distribuée (agents cognitifs). Nous avons alors créé un environnement de développement hybride qui introduit deux classes d'agents génériques, de type système expert, qui permettent de représenter de de manipuler les connaissances figuratives et les connaissances opératoires. Cette distribution de deux types d'agent nous paraît importante car elle va permettre également une distribution du contrôle propre à chaque type de connaissance. Par ailleurs, tant les besoins des applications que la nature même des langage de programmation nous ont poussé à concevoir deux classes d'agents génériques de type système expert, permettant de représenter les connaissances figuratives et les connaissances opératoires. Cette vue duale de la distribution et de la coopération fait toute la force de cet environnement, permettant de concevoir des architectures selon deux points de vue : un point de vue centré connaissance et un point de vue centrée tâche.

Un environnement de développement performant, construit autour d'un langage de programmation spécialisant les classes d'agents a également été défini pour permettre la mise au point d'applications complexes et la visualisation des résultats propres à ces applications. Le langage possède une syntaxe d'assez bas niveau ce qui est à la fois un atout et une gêne. Il peut en effet permettre d'implanter de nombreux mécanismes mais également limiter le développement d'applications complexes qui nécessiteraient des primitives de plus haut niveau, implantant par exemple des protocoles de communication sophistiqués.

REFERENCES

[Baujard 89a] O. Baujard (1989). Un Système de Vision par Ordinateur. *Rapport de DEA*. ENSIMAG / INPG. Grenoble.

[Baujard 90a] O. Baujard & C. Garbay (1990). A programming environment for distributed expert system design. *Expert System Applications, ExperSys.*, pp 27-32.

[Baujard 90b] O. Baujard (1990). MAPS : Guide de Programmation. Rapport Technique 89G/MRT/DT002. Alcatel TITN Answare.

[Berthet 92] S. Berthet, Y. Demazeau & O. Boissier (1992). Knowing Each Other Better, *11th DAI Workshop*, pp 1-20, Glen Arbor.

[Bessières 83] P. Bessières (1983). Etude de la génération de plans en univers multi-agents. Thèse de 3° cycle, INPG, Grenoble.

[Birtwistle 73] G. Birtwistle, O.J. Dahl, B. Myhrhaug & K. Nygaard (1973). SIMULA begin, Petrocelli Charter, New York.

[Bobrow 77] D.G. Bobrow & T. Winograd (1977). An overview of KRL, a Knowledge Representation Language. In Proceedings of the 5th IJCAI, pp 213-222, Cambridge, MA.

[Bobrow 83] D.G. Bobrow & M. Stefik (1983). The LOOPS Manual : a Data and Object Oriented Programming System for Interlisp. *Knowledge-Based VLSI Design Group Memo KB-VLSI-81-13*, Xerox PARC, Palo Alto, California.

[Bouron 91] T. Bouron, J. Ferber & F. Samuel (1991). MAGES : a Multi-Agent Testbed for Heterogeneous Agents. *Decentralized A.I 2*. pp 195-214. Demazeau & Müller Eds. Elsevier Science Publishers.

[Briot 88] J.P. Briot (1988). From Objects to Actors : Study of a Limited Symbiosis in Smalltalk-80. Rapport de recherche 88-58 RXF, LITP, Université de Paris 6.

[Caillaud 91] B. Caillaud (1991). Architectures Multi-Agents pour la Reconnaissance de la Parole. Rapport de DEA Signal Image Parole. Grenoble.

[Camarata 83] S. Camarata, D. McArthur & R. Steeb (1983). Strategies of cooperation in distributed problem solving. *Proc 8th Joint Conf Artificial Intel*, pp 767-770. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988.

[Campbell 90] J. Campbell & M. D'Inverno (1991). Knowledge Interchange Protocol, *Decentralized AI 2*, pp 63-80, Demazeau & Müller eds, North-Holland/Elsevier.

[Chaib-Draa 90] B. Chaib-Draa (1990). Contribution à la Résolution Distribuée de Problème : une Approche basée sur les Etats Intentionnels. Thèse de l'Université de Valanciennes et du Hainaut-Cambrésis.

[Chang 91] M. Chang & C. Woo (1991). SANP : a communication level protocol for negociations, *Decentralized AI 3,* Demazeau & Müller eds, North-Holland/Elsevier (à paraître).

[Chassery 86] J. M. Chassery and G. Bourrel (1986). An image package software: IPS design and abilities, *8th Int. Conf. Pattern Recog,* vol. 2, pp. 913-915, Paris.

[Clayton 84] B.D. Clayton (1984). ART Programming Primer. Inference Corporation, Los Angeles, California.

[Connah 88] D. Connah, M. Shiels & P. Wavish (1988). A Testbed for Research on Cooperating Agents, *in Proc European Conference on Artificial Intelligence.*

[Conry 88] S. Conry, R.A. Meyer & J. Searlmen (1988). A shared knowledge base for independent problem solving agents. *Proc IEEE Expert Systems in Government Symposium*, McLean VA.

[Corkill 79] D.D. Corkill (1979). Hierarchical planning in a distributed environment. *in Proc 6th Joint Int Conf Artificial Intel*, pp 168-179.

[Corkill 83] D.D. Corkill & V.R. Lesser (1983). The use of meta-level control for coordination in a distributed problem solving network. *in Proc 8th Joint Conf Artificial Intelligence* , pp 748-756.

[Davis 83] R. Davis & R.G. Smith (1983). Negociation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence 20.*

[Dellepiane 89] S. Dellepiane, C. Regazzoni, S.B. Serpico & G. Vernazza (1989). Application independent knowledge-based framework for complex image recognition. *in Proc 5th Int Conf Analysis and Processing.*

[Ducournau 88] R. Ducournau (1988). YAFOOL. Version 3.22. Manuel de référence. SEMA.METRA, Montrouge.

[Duguerdil 87] P. Dugerdil (1987). Les mécanismes d'héritage d'OBJLOG : vertical et sélectif multiple avec point de vue. Actes du 6ème CARFIA, pp 259-273, Antibes.

[Durfee 87] E.H. Durfee & V.R. Lesser (1987). Using partial global plans to coordinate distributed problem solvers. *in Proc 10th Int Joint Conf Artificial Intel,* pp 875-883. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988.

[Erceau 91] J. Erceau & J. Ferber (1991). L'Intelligence Artificielle Distribuée. "La Recherche". Numéro 233, Vol 22, pp. 750-, juin 1991.

[Erman 81] L.D. Erman, P.E. London & S.F. Fickas (1981). The Design and an example of use of Hearsay III. *Proc 7th IJCAI,* pp 409-415, 1981.

[Ferber 85] J. Ferber (1985). Définition récursive et évaluation distribuée : un modèle rigoureux de langage objet. Actes du 5ème CARFIA, pp 715-724.

[Ferber 91a] J. Ferber & E. Jacopin (1991). The Framework of Eco-Problem Solving. *Decentralized A.I 2*. pp 181-193. Demazeau & Müller Eds. Elsevier Science Publishers.

[Ferber 91b] J. Ferber (1991). Introduction à l'Intelligence Artificielle Distribuée. Dossier : Intelligence Artificielle Distribuée. *Bulletin de l'AFIA*, n°6, pp 16-19, juillet 1991.

[Fikes 85] R. Fikes & T. Kehler (1985). The Role of Frame-Based Representation in Reasoning. Communications of the ACM, 28(9):904-920.

[Galliers 88] J.R.Galliers (1988). A strategic framework for multi-agent cooperative dialogue. *in Proc Europ Conf on Artif Intell, ECAI*, pp 415-420.

[Gaspar 91] G. Gaspar (1991). Communication and Belief Changes in a Society of Agents : Towards a Formal Model of Autonomous Agent, *Decentralized AI 2*, pp 245-256, Demazeau & Müller eds, North-Holland/Elsevier.

[Gasser 87] L. Gasser, C. Braganza & N. Herman (1987). MACE : A Flexible Testbed for Distributed AI Research. Distributed Artificial Intelligence. Vol 1, pp 119-154. Morgan Kaufmann Publishers.

[Georgeff 83] M. Georgeff (1983). Communication and interaction in multi-agent planning. *in Proc National Conf Artificial Intel*. Washington DC, pp 125-129. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 200-204.

[Georgeff 84] M. Georgeff (1984). A theory of action for multiagent planning. in Proc National Conf Artificial Intel, Austin TX, pp 121-125. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 205-209.

[Georgeff 86] M. Georgeff (1986). A representation of events in multiagent planning. *in Proc National Conf Artificial Intel*, Philadelphia, PA, pp 70-75.

[Goldberg 83] A. Goldberg & D. Robson (1983). Smalltalk-80, the language and its implementation. Addison Wesley, reading, Massachusetts.

[Goldstein 77] I.P. Goldstein & R.B. Roberts (1977). NUDGE, a Knowledge-based Scheduling Program. In Proceedings of the 5th IJCAI, pp 257-263, Cambridge, MA.

[Hämmäinen 90] H. Hämmäinen, J. Alasuvanto & R. Mäntylä (1990). Experiences on Semi-Autonomous User Agents. *Decentralized A.I.* pp 253-249. Demazeau & Müller Eds. Elsevier Science Publishers.

[Harvey 88] G. Harvey (1988). Understanding HyperCard for Version 1.1 ; Sybex Books Publishers.

[Hautin 86] F. Hautin & A. Vailly (1986). La coopération entre systèmes experts. *Actes des Journées Nationales du PRC-GRECO "Intelligence Artificielle"*, Cepadues Editions.

[Hayes-Roth 85] B. Hayes-Roth (1985). A blackboard architecture for control. *Artificial Intelligence,* 26:251-321, 1985.

[Hewitt 71] C.E. Hewitt (1971). Procedural Embedding of Knowledge in PLANNER. In Proceedings of the 2nd IJCAI, pp 167-182, London.

[Hewitt 73] C.E. Hewitt, P. Bishop & R. Steiger (1973). A Universal Modular ACTOR Formalism for Artificial Intelligence. In Proceedings of the 3rd IJCAI, pp 235-245, Stanford, California.

[Hewitt 77] C.E. Hewitt (1977). Viewing Control Structure as Patterns of Passing Messages. *Artificial Intelligence,* 8:323-364, 1977.

[Hewitt 83] C. Hewitt & H. Lieberman (1983). Design Issues in Parallel Architectures for Artificial Intelligence. MIT Artificial Intelligence Lab, Cambridge, MA, Memo 750.

[Hewitt 84] C. Hewitt & H. Lieberman (1984). Design Issues in Parallel Architectures for Artificial Intelligence, *Proc 28th IEEE Computer Soc Int Conf, San Fransisco, CA,* pp 418-423.

[Hopkins 88] C. Hopkins (1988). DePlan : Enabling Agents to Produce Plans that Achieve Cooperative Problem Solving, *in Proc Europ Conf on Artif Intell (ECAI),* pp 421-425.

[Hugonnard 91] E. Hugonnard & C. Garbay (1991). Knowledge Elicitation in Biomedicine : a Multi-Agent Approach. *Proc of Annual Int Conf of the IEEE Engineering in Medicine and Biology Society.* Vol 13:1991, pp 1317-1318. IEEE.

[Konolige 83] K. Konolige (1983). A deductive model of belief, *in Proc 8th Int Joint Conf Artificial Intell,* pp 377-381.

[Kornfeld 81] W.A. Kornfeld & C. Hewitt (1981). The scientific community metaphor, IEEE Trans Syst Man Cyber, SCM-11, pp 24-33. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 311-320.

[Lâasri 89] H. Lâasri & B. Maître (1989). Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et Réalisations. Thèse de l'Université de Nancy 1. CRIN / INRIA-LORRAINE.

[Lane 89] D.M. Lane, M. J. Chantler, E.W. Robertson & A.G. McFadzean (1989). A Distributed Problem Solving Architecture for Knowledge Based Vision. *Distributed Artificial Intelligence.*

[Lansky 87] A.L. Lansky & D. Fogelsong (1987). Localised representation and planning methods of parallel domains, *in Proc National Conf Artificial Intel,* Seattle WA, pp 240-245.

[Leblanc 86] T.J. Leblanc (1986). Shared memory versus message-passing in a tightly-coupled multiprocessor : a case study. Univ of Rochester Computer Science Department, Rochester, NY, Tech Rep. Buterfly Project Report 3.

[Lesser 80] V.R. Lesser & L.D. Erman (1980). Distributed interpretation : a model and experiment. IEEE Trans Computer. C-29. pp 1144-1163. also in

Readings in Distributed Artificial Intelligence (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 120-139.

[Lieberman 81] H. Lieberman (1981). A preview of Act 1. AI Memo 625, AI Lab, MIT, Cambridge, MA.

[Maître 90] B. Maître & H. Laâsri (1990). Cooperating Expert Problem-Solving in Blackboard Systems : Atome Case Study. *Decentralized A.I.* pp 251-263. Demazeau & Müller Eds. Elsevier Science Publishers.

[Martin 77] W.A. Martin (1977). OWL. In Proceedings of the 5th IJCAI, pp 985-987, Cambridge, MA.

[Maruichi 90] T. Maruichi, M. Ichikawa & M. Tokoro (1990). Modeling Autonomous Agents and their Groups. *Decentralized A.I.* pp 215-249. Demazeau & Müller Eds. Elsevier Science Publishers.

[Masini 90] G. Masini, A. Napoli, D. Colnet, D. Léonard & K. Tombre (1990). Les langages à objets. IIA. InterEditions.

[McDermott 72] D.V. McDermott & G.J. Sussman (1972). The CONNIVER Reference Manual. AI Memo 259a, AI Lab, MIT, Cambridge, MA.

[Meyer 86] B. Meyer (1986). Eiffel : programming for reusability and extendibility. ACM SIGPLAN Notices, 22(2):85-94.

[Minsky 75] M. Minsky (1975). A framework for representing knowledge. In P. Winston, editor, The Psychology of Computer Vision, pp 211-281, McGraw-Hill, New-York.

[Moon 86] D. Moon (1986). Object-oriented programming with flavors. In Proceedings of the 1st OOPSLA, pp 1-8, Portland, Oregon.

[Ovalle 91] D.A. Ovalle (1991). Contribution à l'étude du raisonnement en univers Multi-Agent : KIDS, une application pour l'Interprétation d'Images Biomédicales. Thèse de l'Université Josehp Fourier - Grenoble I.

[Pesty 89] S. Pesty & C. Garbay (1989). MAPS: A Multi-Agent Problem Solving Environment. *Proc. IASTED Int. Symp. on Expert System Theory and Application*, pp 110-113, Acta Press.

[Pleiad 92] PLEIAD : Pôle et Lieu d'Echange en Intelligence Artificielle Distribuée (1992). Définition Taxonomique du Vocabulaire utilisé en Intelligence Artificielle Distribuée. Groupe de Travail Grenoblois sur l'IAD. Rapport de Recherche Interne.

[Rechenmann 85] F. Rechenmann (1985). SHIRKA : mécanismes d'inférences sur une base de connaissances centrée objets. Actes du 5ème CARFIA, pp 1243-1254, Grenoble.

[Regazzoni 91] V. Murino & C.S. Regazzoni (1991). A Distributed Algorithm for Adaptative Regulation of Image Processing Parameters. *1991 IEEE / SMC Int. Conf. on Systems, Man, and Cybernetics*. Vol 1, pp 259-264. IEEE.

[Roche 89] C. Roche & J.P. Laurent (1989). Les approches objets et le langage LRO2 (KEOPS). Techniques et sciences informatiques, 8(1):21-39.

[Sacerdoti 77] E.D. Sacerdoti (1977). A structure for plans and behavior, Elsevier, New York.

[Sian 91] S. Sian (1991). Adaptation based on Cooperative Learning in Multi-Agent Systems, *Decentralized AI 2*, pp 257-272, Demazeau & Müller eds, North-Holland/Elsevier.

[Smith 80] R.G.Smith & R. Davis (1980). Framework for co-operation in distributed problem solving, *IEEE Trans Syst Man Cyber*. SMC-11, 1, pp 61-70.

[Stefik 79] M.J. Stefik (1979). An examination of a frame-structured representation system. In Proceedings of the 6th IJCAI, pp 845-852, Tokyo.

[Stroustrup 86] B. Stroustrup (1986). The C++ Programming Language. Addison-Wesley Series in Computer Science, Reading, Massachussets.

[Stroustrup 87] B. Stroustrup (1987). The C++ Programming Language. Addison-Wesley Publishing Company, Inc.

[Sueyoshi 91] T. Sueyoshi & M. Tokoro (1991). Dynamic Modeling of Agents for Coordination. *Decentralized A.I 2*. pp 161-176. Demazeau & Müller Eds. Elsevier Science Publishers.

[Theriault 83] D.G. Theriault (1983). Issues in the design and implementation of Act 2. Technical Report 728, IA Lab, MIT, Cambridge, MA.

[Woolridge 91] M. Woolridge, G. O'Hare & R. Elks (1991). FELINE : A Case Study in the Design and Implementation of a Co-operating Expert System. *Proc 11th Int Workshop on Expert Systems and their Applications*. Avignon May 1991.

[Wright 84] J.M. Wright, M.S. Fox & D.L. Adam (1984). SRL2 User's Manual. Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

[Yonezawa 86] A. Yonezawa, J.P. Briot & E. Shibayama (1986). Object-oriented concurrent programming in ABCL/1. *In Proceedings of the 1st OOPSLA*, pp 258-268, Portland, Oregon.

[Zanconato 88] R. Zanconato (1988). BLOBS - An object-oriented blackboard system framework for reasoning in time. *Blackboard Systems*, Addison-Wesley.

PARTIE B

LE SYSTEME DE VISION PAR ORDINATEUR KISS

Chapitre 1 : Etat de l'Art

Chapitre 2 : Motivations

Chapitre 3 : KISS

INTRODUCTION PARTIE B

La vision a été un domaine d'application privilégié pour démontrer la validité des concepts MAPS. Conçu comme une application de MAPS, la vision a contribué au renouvellement de la problématique multi-agents en posant des problèmes spécifiques. Ces problèmes, présentés dans un premier chapitre, concernent la distribution des niveaux de représentation et des niveaux de traitements impliqués en vision. Ils concernent également le besoin de considérer des informations selon plusieurs points de vue, par exemple les points de vue région et contour, à l'intérieur de ces niveaux. Le problème du contrôle se pose alors, en terme de focalisation et d'adaptation afin de faire coopérer plusieurs tâches et plusieurs styles de représentation à des niveaux d'abstraction différents et souvent selon des points de vue différents.

Après une présentation de nos motivations, effectuée dans le second chapitre, nous présentons KISS (Knowledge-based Image Segementation System) le système de vision réalisé. Le système KISS a été conçu en considérant une application privilégiée : l'analyse d'images biomédicales. L'architecture de ce système est issue d'une décomposition en tâches du problème de la vision. Elle propose en effet une décomposition originale des connaissances figuratives et opératoires ainsi qu'une distribution du contrôle associé à ces connaissances à différents niveaux : niveau action et tâche, niveau information et niveau point de vue.

Outre l'introduction de niveaux de contrôle différents, notre modèle illustre l'introduction de points de vue avec l'introduction d'une coopération région / contour. Il illustre également la coopération entre des niveaux d'abstraction différents. L'information image sera ainsi analysée selon des points de vue géométriques, relationnels et sémantiques au long de trois phases principales d'analyse (phases d'analyse de bas niveau, de niveau intermédiaire et de haut niveau).

Le système ainsi créé, appliqué à l'analyse d'images de microscopie (fibres musculaires), bien qu'intégrant des outils et des techniques de traitement simples, permettra de tester et de valider la robustesse de l'architecture globale et l'intérêt de l'environnement MAPS pour la vision.

CHAPITRE 1

ETAT DE L'ART

I INTRODUCTION

Selon le paradigme "segmentation-interprétation", deux tâches essentielles peuvent être clairement identifiées en Vision par Ordinateur : la tâche de segmentation et la tâche d'interprétation d'image **[Lux 85]**. La segmentation consiste à identifier les parties significatives de l'image, qui peuvent être utilisées de manière adéquate par la tâche d'interprétation. Alors que la tâche de segmentation s'appuie sur les correspondances spatiales entre des objets et des primitives de l'image (des régions par exemple), l'interprétation d'image s'appuie sur leurs correspondances sémantiques. La recherche des meilleures correspondances est le défi majeur de ces tâches. L'amélioration du découpage en niveaux de représentation a contribué à la réactualisation de ce paradigme, comme étant le mode de traitement de type prédiction / vérification, permettant de passer d'un niveau à l'autre **[Boissier 92]**.

Les différentes tâches de la vision reposent sur l'emploi d'outils variés, qui sont décrits dans le premier paragraphe. Le problème majeur est alors la mise en œuvre de ces outils. Cette mise en œuvre nécessite des techniques de focalisation, de sélection et d'adaptation appropriées aux informations et aux tâches manipulées, ainsi que des mécanismes de contrôle sophistiqués. Ces mécanismes sont décrits dans un second paragraphe. Des approches à base de connaissances ont alors été introduites, dans les tâches d'interprétation notamment. Les systèmes de vision se caractérisent également par le besoin de faire coopérer plusieurs approches (segmentation région et contour par exemple) et plusieurs styles de représentation (souvent à différents niveaux d'abstraction). Nous insistons ainsi dans ce second paragraphe sur les problèmes d'intégration et de coopération, c'est-à-dire le point de vue algorithmique, et sur les problèmes de conception associés aux systèmes à base de connaissances, c'est-à-dire le point de vue logiciel.

II OUTILS DE LA VISION

Le problème de la vision consiste à donner un sens à des données brutes de départ, constituées d'images contenant une représentation planaire et imparfaite du monde. Il y a ainsi une distance importante qui peut être franchie en décomposant le problème en plusieurs étapes entre l'information brute et le sens final de l'image : une étape de perception, une étape de description, une étape d'identification et une étape de compréhension. A chacune de ces étapes

correspond un type d'information particulier ou indice. Le travail a été ainsi parcellé et une expertise parcellaire obtenue, en détection de contours, en décomposition géométrique, ou en inférence de formes par exemple. Dans ce paragraphe, nous présentons tout d'abord les informations manipulées en Vision par Ordinateur, puis les outils d'extraction de primitives, les techniques de manipulation de primitives, et finalement les techniques d'interprétation.

1. INFORMATIONS MANIPULÉES

Une primitive de l'image est définie comme un ensemble de points image, pouvant être restreint au pixel élémentaire. Elle est de plus caractérisée par un ensemble d'attributs qui la divisent en deux catégories, selon qu'ils décrivent le support spatial ou la distribution des niveaux de gris ou la couleur caractérisant la primitive. Un type de primitive est défini par un ensemble de contraintes opérant sur les attributs de la primitive : des primitives contraintes spatialement et/ou contraintes par l'intensité peuvent ainsi être définies. Relâcher ces contraintes permet alors de fournir une description multi-résolution de la primitive d'image **[Garbay 86a]**.

Un grand nombre d'approches pour la description des primitives de l'image peuvent être dérivées de ces définitions **[Levialdi 83]**, incluant :

- une description intrinsèque des couleurs au niveau pixel ;
- une description relative des éléments de frontière au niveau pixel ;
- une description des primitives locales spatialement contraintes (calcul de la moyenne en niveau de gris et de la variance parmi quelques voisins, aboutissant à une modélisation pyramidale par exemple);
- l'utilisation des contraintes de surface pour définir des primitives, incluant le célèbre modèle de facette de Haralick **[Haralick 81]**.

La notion de primitive est un concept vital pour la compréhension d'images, elles peuvent être manipulées par des procédures visant à les étiqueter ("phase d'interprétation"), ou par des procédures de décomposition/fusion destinées à la construction de primitives de plus haut niveau, appelées entités ("phase de segmentation").

Une entité est décrite par un ensemble maximal de primitives selon un ensemble de contraintes. Deux classes d'entités peuvent être distinguées :

- les entités cumulatives : ce sont les entités rassemblant un ensemble de pixels, ayant été étiquetés sur un critère individuel, au moyen d'un seuillage des niveaux de gris par exemple;
- les entités associatives : ce sont les entités associant les pixels selon un critère de regroupement local (critère de similarité) ou global (degré de convexité).

Deux types de descripteurs d'entités peuvent être distingués : ceux du premier ordre tels que les histogrammes et ceux du second ordre tels que les matrices de cooccurrence **[Haralick 75]**. Des descripteurs de plus haut niveau incluent la caractérisation topologique, le calcul des graphes d'adjacences ou la représentation arborescente des entités.

2. OUTILS ET TECHNIQUES

Une description des outils généraux de détection, inspirée de la revue établie par Haralick **[Haralick 85]** et le Greco "Traitement du Signal et Images" **[GDR 91]** est effectuée tout d'abord. Ces outils conduisent souvent à des primitives bruitées ou incomplètes. Il est alors nécessaire de construire des entités plus robustes en vue de la tâche d'interprétation. A cette fin, plusieurs techniques de manipulation ont été développées, dont nous donnerons une description des principales caractéristiques dans un second paragraphe. Les techniques d'interprétation seront finalement présentés dans un troisième paragraphe.

2.1. OUTILS D'EXTRACTION DE PRIMITIVES

Un grand nombre d'approches pour la segmentation d'images peut être trouvé dans la littérature. Ces approches diffèrent selon le type et le niveau de la primitive qu'elles manipulent et selon le type d'entité qu'elles sont censées produire. C'est ainsi que les techniques de détection de région et d'extraction de contours s'opposent : elles différent principalement par le type de primitives qu'elles manipulent (primitives intrinsèques ou obtenues par différenciation relative) et par le type d'entité (région ou contour) finalement obtenu. Ce paragraphe présente ainsi les principaux outils d'extraction de régions et de contours.

Les schémas de relaxation ne sont pas traités ici, bien que beaucoup d'efforts de recherche leur ait été consacrés : en effet, ils sont reconnus par la communauté scientifique, comme des procédures améliorant la tâche de segmentation. Cependant, il ne faut pas négliger leur contribution au renouvellement du paradigme de la segmentation et à l'ouverture vers les approches connexionistes.

2.1.1. OUTILS D'EXTRACTION DES PRIMITIVES RÉGION

Deux techniques de segmentation région peuvent être distinguées selon que les entités sont obtenues par classification ou par agrégation de primitives. Les premières conduisent en effet à des entités cumulatives alors que les seconde conduisent à des entités associatives. Une revue complète de ces différentes techniques est présenté dans **[Haralick 85]**.

Techniques de Classification

Les techniques de classification sont employées pour définir une partition d'après un ensemble

de descripteurs de primitives. Une étude assez complète des techniques de segmentation par classification peut être trouvée dans **[Kittler 85]**, **[Celeux 89]**, **[Lee 90]** et **[Maitre 91]**. Une étiquette est affectée à chacune des primitives en fonction de la classe d'appartenance. Les entités de l'image finalement obtenues correspondent aux composantes connexes des primitives possédant les mêmes étiquettes. La performance de ces procédés peut être mesurée par leur capacité à séparer les entités dans l'espace des classes.

Le point faible de l'approche par classification est évident : puisque l'analyse est faite dans l'espace des descripteurs, sans aucune contrainte spatiale, les frontières des régions résultantes sont très bruitées et mal définies. C'est la raison pour laquelle les approches relationnelles ont été introduites, de façon à trouver un compromis entre descripteurs et contraintes spatiales.

Techniques d'Aggrégation

La technique de base des approches relationnelles est la procédure de "croissance de régions". Dans cette approche, les pixels représentent les noeuds d'un graphe; les pixels voisins dont les caractéristiques sont suffisamment proches sont rejoints par un lien. Les entités de l'image correspondent alors à des ensembles maximaux dans le graphe.

Un contrôle géométrique peut être introduit pour contrôler le processus de segmentation. Ce type d'approche a été utilisé en segmentation d'images cytologique **[Chassery 84]**, **[Garbay 86b]** et **[Chassery 91]**. Dans ces images, chaque cellule possède une forme convexe, le processus de segmentation est alors un processus de croissance de région introduisant un contrôle de la convexité. D'autres approches utilisent les contours actifs **[Kass 87]**, **[Durbin 87]**, **[Berger 91]**. Une courbe de contour est placée au voisinage d'une forme à détecter, cette courbe évolue en fonction de l'image, sa déformation étant effectuée sous le contrôle d'une fonction d'énergie.

2.1.2. Outils d'Extraction des Primitives Contours

Dans une image numérique, les contours se situent entre des régions ayant des intensités moyennes différentes ; il s'agit alors de contours de type "saut d'amplitude". Un contour peut également correspondre à une variation locale d'intensité présentant un maximum ou un minimum ; il s'agit alors d'un contour dit "en toit". Une description des méthodes détectant et localisant les variations locales d'intensité, ainsi qu'une comparaison des résultats obtenus peut être trouvée dans une revue comparative effectuée par le groupe national "Greco Traitement du Signal et Images" **[GDR 91]**. Ce paragraphe rappelle le principe de ces différentes méthodes.

Méthodes Dérivatives

Les approches dérivatives peuvent être séparées en trois grandes classes : les approches utilisant

le gradient, les approches utilisant la dérivée seconde directionnelle et les approches utilisant le laplacien. Dans le cas d'une utilisation du gradient, après un calcul du gradient en chaque point de l'image, une image des normes du gradient est créée suivie d'une extraction des maxima locaux dans un voisinage de chaque point de l'image de la norme, dans la direction du gradient. Dans le cas d'une utilisation de la dérivée seconde, après un calcul de la dérivée seconde dans la direction du gradient, une recherche des passages à zéro de la dérivée est effectuée, suivie de la création de l'image des passages par zéro affectés de la norme du gradient. Dans le cas d'une utilisation du laplacien, après un calcul du laplacien, une recherche des passages par zéro est effectuée, suivie de la création de l'image des passages par zéro affectés de la norme du gradient.

Plus récemment, des approches par filtrage optimum ont été introduites. Dans ces approches, le contour est modélisé par un échelon d'amplitude noyé dans un bruit blanc, le filtre optimum est alors chargé de détecter et localiser la transition. L'opérateur de Canny a tout d'abord été proposé **[Canny 86]**, repris par Deriche pour la définition d'un filtre optimal **[Deriche 87]**. Ces opérateurs calculent le gradient d'une image, Deriche **[Deriche 90]** a également montré comment calculer le laplacien en proposant une approche voisine de celle de Shen **[Shen 86]**.

Ces différentes approches sont les plus immédiates pour détecter et localiser les variations d'un signal, cependant elle sont assez sensibles au bruit : un prétraitement est souvent employé pour y remédier, s'il n'est pas déjà inclus dans l'opérateur comme dans l'opérateur de Deriche. Parmi les prétraitements, citons les modifications d'histogramme, la réduction de bruit par l'application de filtres (comme le filtre médian) ou bien le rehaussement de contraste.

Modèle Surfacique

Ces approches utilisent un modèle surfacique de transition (par exemple un échelon bidimensionnel sur un voisinage donné). Un contour est détecté en un point, s'il y a une correspondance entre le modèle et la fenêtre de l'image centrée en ce point. C'est le cas de la méthode de Hueckel **[Hueckel 71]** qui se donne un modèle idéal de contour. Une autre technique consiste à approcher la surface représentant une transition par un polynôme. L'équation est alors utilisée pour obtenir une meilleure détermination de l'orientation du contour **[Haralick 84]**.

D'autres approches utilisent la morphologie mathématique, ou font appel aux champs de Markov, une présentation détaillée de ces techniques peut être trouvée dans **[GDR 91]**.

2.2. TECHNIQUES DE MANIPULATION

Les outils présentés précédemment fournissent souvent des primitives bruitées et non utilisables telles quelles par la phase d'interprétation. Dans le cas des contours par exemple, les approches

fournissent des contours qui ne sont pas forcément minces (épaisseur de un pixel) et qui présentent des déconnexions. Dans le cas des régions, des sur-segmentations ou des sous-segmentations sont souvent remarquées. Des techniques de manipulation doivent être alors introduites : ce paragraphe décrit tout d'abord diverses techniques de reconnexion de contours puis des techniques de décomposition de forme fondées sur des critères géométriques.

2.2.1. FERMETURE DES CONTOURS

La fermeture des contours est un problème difficile à résoudre. Dans certains cas, une simple procédure morphologique telle une fermeture binaire peut suffire, dans d'autres cas, il est nécessaire de faire appel à des techniques plus sophistiquées. Nous présentons ici deux techniques de fermeture l'une fondée sur des critères de distance et l'autre sur des informations contour.

Fabre **[Fabre 88]** propose une méthode de reconnexion fondée sur l'optimisation d'un critère de distance. Chaque élément de contour devient le sommet d'un graphe, un arc sur ce graphe représentant alors une connexion possible entre deux éléments de contour. Chaque arc est valué par une fonction de coût, qui correspond au coût de la reconnexion (nombre de pixels pour reconnecter les deux éléments de contour par exemple). Un algorithme de recherche de coût optimal (Disjkstra ou A*) suffit alors pour trouver le graphe du contour optimal.

Ces techniques sont fondées uniquement sur des critères de distance. D'autres techniques introduisant des informations contours telles que le gradient, paraissent plus aptes à fournir une reconnexion correcte de deux contours. Deriche et Coquerez **[Deriche 87]** présentent ainsi un processus de fermeture de contour fondé sur l'étude de contours d'épaisseur un pixel : il est alors aisé de détecter les extrémités des contours. Le procédé élabore à partir de chaque extrémité de contour, tous les chemins possibles d'une longueur donnée, et ne retient que le meilleur (la fonction de coût peut être la somme des normes du gradient des points qui composent les chemins). Cette technique de fermeture est assez performante, car elle se fonde sur l'image des gradients et non pas comme dans la technique précédente sur le seul critère de distance.

2.2.2. DÉCOMPOSITION GÉOMÉTRIQUE

La décomposition géométrique recouvre deux classes de méthodes. Une première classe de méthodes introduit une décomposition de forme à l'aide de composantes plus simples (des triangles ou des polygones convexes) ; l'aspect structurel de l'image n'est pas considéré dans ces approches. Une seconde classe de méthodes s'intéresse à résoudre le problème de l'organisation structurelle de l'image. Ces méthodes effectuent alors une décomposition faisant apparaître les composantes de l'image **[Chassery 91]**.

Les méthodes de la première classe travaillent à partir d'un approximant polygonal de la forme à décomposer. Parmi ces méthodes, citons la décomposition en éléments convexes. Cette décomposition peut être effectuée en étudiant les sommets successifs de la ligne polygonale associée au polygone, et en reliant les sommets consécutifs concaves **[Chazelle 85]**. D'autres méthodes utilisent des rectangles **[Ferrari 80]** ou des polygones de Voronoï **[Schachter 78]** **[Preparata 88]**.

Les méthodes de la seconde classe utilisent un environnement de graphe qui permet de spécifier les relations entre les composantes qui interviennent dans la décomposition. Une première approche consiste à utiliser un diagramme de Voronoï généralisé **[Hu 89]**. Cette approche généralise la représentation par régions de Voronoï en passant de germes ponctuels à des germes d'une complexité plus grande tels que des segments de droite. Il est alors possible de construire un diagramme de Voronoï pour l'intérieur d'un forme polygonale et de l'utiliser à des fins de décomposition. Une seconde approche est fondée sur la morphologie mathématique. Une déconnexion peut être réalisée en générant une succession d'érosions et en contrôlant la conservation de la connexité à chaque étape. Dès que la déconnexion se réalise, on effectue sur chaque composante l'opération inverse de dilatation.

Enfin, une troisième approche exploite la ligne médiane. Le traitement de la décomposition repose sur le partitionnement de la ligne médiane et peut être facilité par la structure du graphe de la ligne médiane. Une étude du graphe de la ligne médiane est présenté dans **[Montanvert 87]**. Ce graphe est obtenu à partir de la ligne médiane d'un objet. La ligne médiane est formée de l'axe médian d'un objet complété des chemins de connexion discrets, orientés perpendiculairement aux courbes de niveau. Les sommets du graphe seront les composantes 8 connexes de l'axe médian de même poids. Les arcs seront les chemins de connexion discrets, rajoutés à l'axe pour obtenir la ligne médiane. Les arcs sont orientés des sommets de poids i aux sommets de poids j (i<j). Les sommets sont classés selon les arcs qui les relient : sommet principaux, sommets de proéminence large et stricte et sommets de rétrécissement. Les sommets principaux sont les sommets sans aucun arc sortant, les sommets de proéminence large sont les sommets avec un seul arc sortant et au moins un arc rentrant, les sommets de proéminence stricte sont les sommets avec un seul arc sortant et aucun arc rentrant, et les sommets de rétrécissement sont les sommets avec au moins deux arcs sortants. Il est alors possible d'interpréter ce graphe et des procédés de décomposition, de nettoyage ont été mis en évidence : il est par exemple possible d'utiliser un point de rétrécissement du graphe pour décomposer une forme en deux parties.

2.3. TECHNIQUES D'INTERPRÉTATION D'IMAGES

L'étape d'interprétation représente une tâche problématique pour les systèmes de Vision par Ordinateur actuels : il est nécessaire de construire les modèles représentant les objets

susceptibles d'être rencontrés (forme d'expression pratique) dans l'image mais aussi de définir les stratégies d'identification des régions qui ont effectivement été obtenues. Ces stratégies doivent être suffisamment sophistiquées **[Nagao 84]** pour faire face aux difficultés rencontrées, telles des scènes complexes incluant des objets partiellement visibles ou des objets d'apparence variable, un bruit important ou des ombres présentes sur les images.

En ce qui concerne les techniques d'interprétation d'images, une distinction essentielle peut être faite entre mise en correspondance et inférence. Nous abordons également les techniques adaptatives, fondées sur les modèles markoviens et les réseaux de neurones.

2.3.1. Techniques de Mise en Correspondance

Une revue approfondie des techniques de mise en correspondance est présentée dans **[Mohr 88]** et **[Belaïd 91]** d'où nous tirons la synthèse présentée ci-dessous. Nous abordons tout d'abord le problème central de la modélisation, puis décrivons successivement les techniques de prédiction / vérification, la recherche de cliques maximales, la relaxation continue et discrète, et enfin les techniques de mise en correspondance statistique.

Modélisation

La modélisation est l'aspect central de la mise en correspondance. Il existe divers types de modèles dont :

- le modèle paramètré : cette approche simple met en oeuvre la modélisation d'un objet par une fonction analytique définie par un ensemble de paramètres **[Ayache 86]**, **[Horaud 87]**;
- le modèle rigide : cette approche considère la forme comme un assemblage de primitives de base, groupées selon une disposition fixe, modulo une transformation de l'espace de travail;
- le modèle à contrainte déformable : ces modèle sont une extension naturelle des modèles rigides. Ils sont définis comme des groupes de modèles rigides mutuellement contraints;
- le modèle générique : une forme "générique" est définie comme une forme localement continuement déformable; les modèles "génériques" sont définis pour représenter les formes et leur variations possibles, contrôlées par un ensemble de contraintes **[Bajcsy 74]**, **[Brooks 78]**;

Contrôle

Les systèmes travaillant selon le paradigme de la prédiction / vérification conduisent à une intense coopération entre les tâches de segmentation et d'interprétation : la phase d'interprétation

n'apparait plus en fait après la phase de segmentation, ces deux étapes alternant rapidement. L'approche de prédiction / vérification constitue actuellement l'une des techniques les plus employées parmi les stratégies de mise en correspondance **[Souvigné 83] [Lux 84]**. Un contrôle efficace est nécessaire pour éviter d'explorer inutilement un nombre de chemins exponentiel.

Recherche de cliques maximales

Les méthodes d'interprétation d'images fondées sur la recherche de cliques maximales exploitent les informations contenues dans un graphe de compatibilités entre hypothèses ; les noeuds du graphe représentent les hypothèses et les arcs sont utilisés pour relier deux hypothèses compatibles. Les cliques ou sous-graphes complets sont alors des ensembles d'hypothèses mutuellement compatibles, assistant ainsi la procédure de confirmation d'hypothèse **[Bolles 82]**.

Relaxation continue & discrète

Les méthodes fondées sur la relaxation font elles aussi appel à un graphe modélisant les contraintes de compatibilité entre deux hypothèses. L'étiquetage est modifié itérativement d'après l'examen des étiquettes voisines, de manière à diminuer la confiance d'un étiquetage peu consistant, et au contraire augmenter la confiance des étiquetages les plus consistants **[Faugeras 82]**.

Mise en correspondance statistique

Pendant les deux dernières décennies, la plupart des développements en reconnaissance de formes ont fait appel à la mise en correspondance statistique **[Fu 80]**. Cette mise en correspondance (ou classification de formes inconnues en classes de caractéristiques différentes) implique le calcul et la sélection de mesures caractéristiques. Ces mesures sont en général sélectionnées selon leur "importance", ou plus précisément leur contribution à l'amélioration de la reconnaissance. Pour des problèmes complexes de reconnaissance, les techniques de mise en correspondance statistique deviennent extrêmement difficiles à exploiter; c'est le cas quand les formes à étudier deviennent complexes et quand le nombre de descriptions possibles est très important. C'est pourquoi d'autres approches telles que les approches structurelles ou syntaxiques ont été proposées.

Selon Mohr **[Mohr 88]**, la meilleure solution pour les systèmes utilisant les techniques de mise en correspondance est de faire coopérer étroitement plusieurs de ces méthodes. Par exemple, il serait intéressant de combiner la méthode de prédiction / vérification avec la recherche de cliques maximales et obtenir ainsi un ensemble initial d'hypothèses. La méthode de relaxation pourrait alors éliminer les hypothèses indésirables. Enfin, le parallélisme pourrait

éventuellement être utilisé puisque la plupart des opérations impliquées dans ces méthodes peuvent s'exécuter simultanément.

2.3.2. TECHNIQUES D'INFÉRENCE

Approches à base de règles de production

Les techniques d'interprétation fondées sur des règles de production assistent les processus d'interprétation déductifs, elles sont fondées sur l'utilisation des connaissances disponibles dans la base de connaissance et leur exploitation se fait par un mécanisme à base de règles de production **[Mckeown 85] [Mitiche 88]**. L'intérêt de ces approches est lié à la très grande combinatoire des problèmes et à la nécessité de combiner les données perceptuelles aux informations sémantiques sur la scène traitée **[Haton 87]**.

Utilisation d'une modélisation orientée objet

Une modélisation orientée objet permet de représenter des formes complexes selon un schéma structuré et hiérarchique, capable de décrire aussi bien les propriétés d'objets individuels que leur relations mutuelles (incluant les relations "est-un" et dans certains systèmes "partie-de"). Des "classes" ou "types" de formes sont introduites pour représenter des familles de formes partageant des attributs similaires, tandis que les "instances" représentent la réalisation instantanée d'une classe. Un tel schéma de modélisation est souvent utilisé en association avec des règles de production pour enrichir le mécanisme d'inférence, limité autrement aux mécanismes d'héritage et d'instanciation. De nombreux systèmes ont utilisé ce formalisme tels VISIONS **[Hanson 78]**, ACRONYM **[Brooks 83]** et MESSIE **[Garnesson 89]**.

2.3.3. TECHNIQUES ADAPTATIVES

Modèles markoviens

Les modèles d'image par champs aléatoires de Markov sont particulièrement utilisés pour la segmentation d'images texturées, certains d'entre eux permettant la modélisation des textures et d'autres la modélisation des régions qui contiennent ces textures. L'un des principaux avantages des champs markoviens vient de la possibilité de travailler simultanément à plusieurs niveaux. Dans **[Geman 86]** un système a été construit pour la segmentation d'images qui exploite deux niveaux simultanés : le premier niveau ou "niveau d'observation" correspond aux pixels et le second niveau ou "niveau étiquette" correspond aux régions extraites.

Le recuit simulé est une méthode d'optimisation utilisée pour minimiser la fonction d'énergie : il est utilisé dans ce cas pour déterminer une distribution optimale des étiquettes. Une étude complète des modèles de champs aléatoires de Markov et de leur application à la vision par

ordinateur peut être trouvée dans **[Geman 87]**.

Réseaux neuronaux

Les réseaux de neurones artificiels ont été étudiés il y a plusieurs années pour atteindre des performances proches de celles de l'homme, en particulier dans le domaine de la compréhension de la parole et de la reconnaissance d'image. Ces structures sont constituées de plusieurs éléments calculatoires opérant de manière non linéaire et parallèle. Ces éléments sont arrangés en modèles rappellant les réseaux de neurones biologiques.

L'adaptation ou l'apprentissage constitue l'un des thèmes majeurs de la recherche actuelle sur les réseaux de neurones. La possibilité d'adapter et de continuer l'apprentissage est essentielle en présence de nouveaux environnements, quand les données apprises sont insuffisantes **[Lippmann 87]**; c'est le cas des systèmes de reconnaissance de cibles où plusieurs séquences d'images doivent être traitées simultanément.

3. DISCUSSION

Dans ce paragraphe, nous avons vu que de nombreuses techniques, associées avec de multiples représentations possibles avaient été définies dans les domaines de la segmentation, de la manipulation des primitives et de l'interprétation. Aucun de ces outils n'est cependant suffisamment général pour fournir un résultat correct dans tous les cas. De nombreux auteurs pensent qu'il est alors nécessaire de faire coopérer plusieurs méthodes. C'est par exemple le cas de la fermeture de contour qui pourrait bénéficier de la coopération des approches géométriques et des approches fondées sur le gradient. C'est également le cas pour la mise en correspondance où l'on pourrait faire coopérer plusieurs méthodes **[Mohr 88]**. Une représentation plus fine du problème semble cependant manquer, pour savoir comment combiner, sélectionner et faire coopérer les différentes techniques avec les différents styles de représentation. Il convient alors de trouver les algorithmes et les styles de conception les mieux adaptés à ces besoins de représentation et de contrôle. Ces nouvelles problématiques seront discutées dans le paragraphe suivant.

III DES OUTILS AUX SYSTEMES DE VISION

Ce paragraphe aborde la problématique des systèmes de vision. Nous décrivons tout d'abord les problématiques propres à la complexification algorithmique des systèmes (souvent les systèmes de segmentation). Il s'agit de faire coopérer plusieurs techniques de segmentation et d'utiliser pour cela des représentations adaptées de l'image : dans ce cadre, il semble intéressant d'employer des représentations multi-niveaux. Des problèmes de focalisation, d'adaptation et de contrôle se posent alors pour exploiter au mieux ces représentations. Dans ce cadre, le style de

conception du système révèle toute son importance et nous assistons donc à une complexification des technologies logicielles employées. Nous présentons tout d'abord le point de vue algorithmique puis le point de vue conception.

1. POINT DE VUE ALGORITHMIQUE

La segmentation est une tâche complexe, qui ne peut être appréhendée comme la simple application d'un opérateur unique à une forme donnée de représentation de l'image ; il apparait au contraire que chaque méthode apporte une contribution partielle au problème, et que sa résolution ne peut venir que de l'intégration de différentes méthodes de segmentation, et de leur coopération.

Il ne s'agit pas néanmoins de réduire le problème à sa seule dimension opératoire, l'accumulation irraisonnée d'opérateurs conduisant à la conception de systèmes complexes, dont la maîtrise et la réutilisabilité sont difficiles. Tout opérateur en effet ne vaut que s'il est appliqué à bon escient, eu égard aux situations rencontrées, et aux buts recherchés. Ces questions sont abordées dans un second paragraphe.

1.1. COOPÉRATION

Plusieurs auteurs ont cherché à élaborer des processus avancés de segmentation, fondés sur la prise en compte de formes différentes de représentation de l'image, mais aussi sur l'application de formes différentes de traitement. Deux classes d'approches peuvent être distinguées : les premières oeuvrent à des niveaux de "résolution" différents, alors que les secondes impliquent des principes de détection différents.

1.1.1. REPRÉSENTATIONS MULTI-NIVEAUX

Outre les techniques de filtrage **[Chehikian 91]**, sur lesquelles nous ne reviendrons pas ici, de nombreuses méthodes permettent l'obtention de représentations multiples d'une image, parmi lesquelles la représentation par "quadtree", la décomposition par diagramme de Voronoï, ou les représentations pyramidales.

Représentation par Quadtree

La décomposition en "quadtree" est un processus descendant qui procède par activation successive de cycles de décomposition, de manière à fournir progressivement une description plus précise des primitives. La décomposition "quadtree" a tout d'abord été proposée par **[Klinger 73]** : l'image est décomposée en quadrants, décomposés à leur tour si et seulement si un certain critère d'homogénéité n'est pas vérifié. L'efficacité de la décomposition peut être augmentée en découpant arbitrairement l'image en régions carrées d'une taille choisie par

l'utilisateur et en les décomposant alors si elles ne sont pas homogènes.

Cette procédure tend cependant à produire des frontières d'aspect carré un peu artificielles: une stratégie de décomposition et de fusion a été proposée par **[Horowitz 76]** pour faire face à ce problème.

Décomposition par Diagramme de Voronoï

D'autres auteurs ont proposé d'utiliser une décomposition en diagrammes de Voronoï plutôt que la décomposition en "quadtree" **[Melkemi 91]**. L'intérêt d'une telle approche réside dans la plus grande diversité des formes (polygones) par rapport aux carrés des "quadtree" et dans la conservation de la convexité.

Représentations Pyramidales

Une représentation pyramidale de l'image peut être obtenue de manière "ascendante" par l'activation itérative d'un processus de fusion **[Rosenfeld 83]**. Dans cette approche, l'image est initialement représentée par un graphe d'adjacence des pixels. Un nouveau graphe est calculé à chaque cycle, résultant de la contraction du graphe précédent **[Rosenfeld 82]**. De nouveaux descripteurs de primitives sont alors calculés et le graphe est réévalué. Pour éviter des redondances dans leur évaluation, il est préférable que les descripteurs partagent un même formalisme récursif **[Kropatsch 83]**.

1.1.2. COOPÉRATION MULTI-NIVEAUX

Le principe

Le principe de la coopération multi-niveaux est d'opérer de manière progressive, en se fondant sur des représentations intermédiaires progressivement plus précises, plus raffinées de l'image : les détails de l'image sont en effet inutiles à la perception globale des structures présentes dans l'image, ils sont par contre nécessaires à leur appréhension précise .

On peut ainsi procéder simplement par décomposition récursive de l'histogramme des niveaux de gris : des regroupements d'abord grossiers, puis plus fins sont ainsi opérés dans l'image **[Parvin 84]**. On peut inversement procéder par fusions progressives de primitives **[Gagalowitz 85]**, en relachant progressivement les contraintes utilisées. Deux classes d'attributs sont particulièrement employées dans ce but, qui regroupent les attributs de niveau de gris et de texture **[Funakubo 84]**, **[Xu 84]**.

Il est plus généralement possible d'articuler fusion et décomposition, d'associer différents critères d'analyse au sein de processus complexes de segmentation d'images.

Processus coopératifs

Les systèmes de segmentation que nous décrivons ici impliquent tout d'abord la construction d'une représentation complexe de l'image, puis son exploitation par diverses techniques d'analyse. Le principe est ici d'utiliser les représentations obtenues pour guider les traitements ultérieurs : une représentation grossière de l'image est tout d'abord analysée, qui permet de guider et de contrôler les traitements opérant à haute résolution. Des primitives de haut niveau sont en effet accessibles d'emblée : elles peuvent être fournies aux traitements dès l'étape initiale. Plusieurs procédures peuvent être utilisées à cette fin : la relaxation **[Narayanan 83]**, la croissance de région **[Tilton 84]** et le seuillage **[Parvin 84]**.

Dans les approches pyramidales, il s'agit également de profiter de la faible résolution des niveaux supérieurs de la pyramide **[Tanimoto 75]** pour progresser de manière efficace dans la segmentation. Par ailleurs, chaque étape de contraction représentant une étape du processus de fusion, une segmentation de l'image peut être effectuée à chaque niveau de la pyramide, multipliant ainsi les possibilités de raisonnement **[Montanvert 91]**.

D'autres approches, plus complexes, consistent à élaborer puis modifier progressivement la représentation multi-niveaux d'une image, de type "quadtree" par exemple. La représentation initiale du "quadtree" est tout d'abord modifiée par des décompositions et fusions successives de blocs adjacents, de façon à identifier les quadrants significatifs. Le graphe d'adjacence des régions est alors utilisé, en association avec des critères sophistiqués de regroupement, pour aboutir à la segmentation.

1.1.3. COOPÉRATION RÉGION / CONTOUR

Dans les systèmes de segmentation opérant par coopération entre détection des régions et détection des contours, les contours sont en général utilisés comme des contraintes, ou des références, permettant de guider et de contrôler les processus d'extraction des régions : plusieurs de ces approches sont décrites dans **[Haralick 85]**.

La technique de segmentation proposée par Wrobel **[Wrobel 87]** utilise également une segmentation par croissance de région guidée par une carte des contours. Une autre approche **[Pavlidis 90]** consiste à procéder à une sur-segmentation des régions de l'image, en appliquant un processus de type décomposition / fusion. Les erreurs de segmentation sont ensuite corrigées en combinant des mesures de contraste le long des frontières des régions et une fonction de lissage.

[Melkemi 91] propose au contraire de procéder d'abord à une segmentation contour optimale, puis de l'utiliser dans le cadre d'une segmentation région. L'avantage de la solution réside dans l'initialisation : les points de forte variation de contraste font partie des frontière des régions

générées au départ, on exploite ainsi dès le départ des contours pertinents. Un calcul du diagramme de Voronoï généralisé est effectué ensuite pour chaque ligne polygonale correspondant à un contour. Des régions sont ainsi obtenues, qui sont alors divisées jusqu'à l'obtention de région homogènes. Un processus de fusion est ensuite mis en œuvre.

Ces différentes approches ont conduit à la conception d'algorithmes de segmentation dédiés et puissants : ils souffrent cependant de faibles performances dues à l'augmentation de leur complexité. La diversité des niveaux hiérarchiques introduits en vue de rendre l'analyse plus flexible a en fait abouti à une plus grande complexité du contrôle des déplacements dans les niveaux, ce qui dégrade les performances du système.

Une des améliorations apportée a alors été d'introduire des comportements adaptatifs ainsi que des facultés de focalisation, en vue d'adapter la complexité du traitement à la complexité de la région choisie et de focaliser l'analyse sur des informations valables.

1.2. FOCALISATION, ADAPTATION ET CONTROLE

La principale caractéristique des systèmes de traitement d'images est le mécanisme de raisonnement consistant à sélectionner d'une part les meilleures primitives et d'autre part les meilleures actions d'après l'étude des situations courantes **[Shapiro 83]**. Dans cet ordre d'idées, la construction de règles d'interprétation particulières permettant de juger le degré d'information d'une entité ou bien le degré de robustesse d'un processus de segmentation est d'une importance vitale ; de ce fait, les systèmes de vision sont dotés d'un mécanisme de focalisation sur des zones d'intérêt et de la capacité à sélectionner les outils de façon adaptative. Nous présentons tout d'abord les mécanismes de focalisation puis les mécanismes d'adaptation.

1.2.1. FOCALISATION

Les mécanismes de "sélection de zones d'intérêt" permettent d'éviter un traitement systématique et "parallèle" de toutes les entités de l'image, ce qui conduit souvent à bon nombre d'erreurs de segmentation et d'interprétation. Ces erreurs sont alors tellement dépendantes les unes des autres qu'elles compliquent le développement d'un mécanisme robuste de contrôle.

De tels mécanismes ont pour but la sélection des zones qui engendrent les hypothèses les plus robustes; les entités facilement interprétables sont tout d'abord sélectionnées, telles que les entités qui concernent les régions uniformes de grande taille. La sélection de régions complexes permet en outre de restreindre l'application de traitements complexes à certaines régions **[Reinhardt 82]**, **[Tucker 84]**. Les approches énoncées auparavant ont été particulièrement mises en œuvre par les deux systèmes suivants.

La "focalisation sur des caractéristiques locales" **[Bolles 82]** procède à la recherche d'une

caractéristique clé, ou caractéristique cible, sur une image. Cette caractéristique est alors utilisée pour prédire des caractéristiques voisines. Pour cela, un graphe de correspondance est employé afin de trouver la plus grande classe parmi les caractéristiques de l'image qui corresponde aux caractéristiques prédites.

Le système à base de règles de production développé par Nazif et Levine **[Nazif 84]** fait un emploi intensif des mécanismes de "sélection de zones d'intérêt". En plus du critère de sélection mentionné auparavant, ce système rajoute la possibilité de sélectionner des zones d'intérêt dans le cas où certains événements se présentent, tels que la rencontre d'une entité amplement informative (la traversée d'une région par une large ligne de contraste, par exemple). Des critères spatiaux sont également introduits, guidant l'analyse vers la région présentant le plus grand degré d'adjacence, ou la plus grande région voisine.

1.2.2. ADAPTATION

La conception de stratégies d'analyse d'images robustes et flexibles implique une adaptation dynamique de l'analyse d'image pour faire face à des situations particulières rencontrées à un moment donné. Cette notion d'adaptation dynamique doit être considérée en tenant compte d'une part du type de traitement à appliquer, et d'autre part du système de représentation à utiliser, c'est à dire du type de primitives, de la nature des attributs et de la forme des contraintes, employés pour représenter l'image.

En ce qui concerne le problème de la sélection dynamique du système de représentation optimal, par rapport à la situation à traiter et du but de l'analyse, il s'agit d'un aspect certainement critique et qui conduit malheureusement à des paradigmes compliqués et imbriqués qui ne sont en réalité pas bien maîtrisés et peu étudiés par la communauté scientifique.

En ce qui concerne la sélection dynamique du traitement d'analyse le mieux adapté pour faire face à une situation donnée, des aspects plus simples et immédiats sont concernés et ont été adressés plus largement au sein de la communauté. Les systèmes à base de règles de production conviennent particulièrement à cette adaptation dynamique. Le système de Nazif **[Nazif 84]** possède ainsi des règles qui adaptent une action à un événement particulier. Par exemple, en présence d'une région dont l'histogramme est bimodal, une règle mettant en œuvre un procédé de décomposition sera déclenchée. Dans le cadre de systèmes plus complexes, de telles techniques sont également mises en œuvre. Parmi ces systèmes, citons le système OCAPI **[Clément 89]**, système d'intégration sémantique de programmes où certaines tâches sont automatisées telles la sélection des meilleurs programmes pour atteindre un but ou l'initialisation des paramètres.

1.2.3. CONTROLE

Finalement, le développement d'un mécanisme robuste de contrôle est essentiel pour la conception des systèmes de vision. Deux facteurs doivent être pris en compte pour un tel mécanisme : premièrement les critères d'arrêt et d'évaluation et deuxièmement les stratégies de reprise.

En ce qui concerne les critères d'arrêt, il est essentiel de donner au système la faculté d'évaluer le degré de qualité de la segmentation des entités obtenues. Un tel degré doit être calculé en utilisant de simples critères gestaltistes, tels que l'homogénéité, la continuité ou la fermeture, ou bien des caractéristiques plus sophistiquées, produit des quelques connaissances à priori sur l'image sous étude. Ces questions sont aussi adressées plus loin dans ce chapitre.

En ce qui concerne le développement de stratégies de reprise, elles doivent être appliquées en cas d'échec lors de l'application de l'un des mécanismes d'évaluation précédentes. Ces questions se sont rapidement révélées vitales pour le développement d'algorithmes de suivi de contours **[Montanari 71]**, **[Martelli 76]**. Ce paradigme a été entièrement renouvelé, plus récemment, par l'attribution de la théorie du contrôle au sein des systèmes experts et par le développement de schémas d'inférences **[Matsuyama 85]**. A part cela, des mécanismes "ad-hoc" ont généralement été proposés.

2. POINT DE VUE LOGICIEL

Des problèmes accrus se posent dans la conception d'un système de vision, dus à la nécessité de manipuler de manière simultanée des informations et des traitements de nature extrêmement diverses : si dans un système de segmentation, on peut se contenter d'une approche algorithmique mêlant procédures et représentations numériques, on ne peut ici faire l'impasse d'une représentation explicite et symbolique des données, des modèles, voire des stratégies de contrôle. C'est ce double problème qui est évoqué ici : d'une part comment aboutir à des représentations explicites, déclaratives, et comment les exploiter, d'autre part comment marier les exigences de représentation issues de la segmentation (une pyramide, par exemple) et de l'interprétation.

Une question importante est alors de sélectionner un style de représentation des connaissances approprié et une structure de contrôle, et de les intégrer dans une architecture cohérente. De fait, implanter quelques-uns des mécanismes de raisonnement habituellement utilisés, comme par exemple la prédiction / vérification ou la focalisation, dépend fortement de la manière dont les connaissances et le contrôle sont implantés **[Rao 88]**.

Notre but n'est pas ici de décrire les tâches et les connaissances et le contrôle définis par les différents systèmes mais plutôt de voir comment elles sont décrites et utilisées.

2.1. Connaissances et Représentation des Connaissances

Nous décrivons tout d'abord les connaissances manipulées, puis les styles de représentation utilisés.

2.1.1. Connaissances Manipulées

Suivant la typologie proposée par **[Ovalle 91]**, nous abordons d'abord l'étude des connaissances figuratives, puis celles des connaissances opératoires nécessaires à la conception d'un système de vision. Cette distinction est fondamentale, car elle supporte en effet deux axes orthogonaux de conception des systèmes de vision. Le premier (l'axe centré-figuratif) décrit les différents niveaux de représentation impliqués en vision **[Marr 82]**, il décrit de manière indirecte les opérations ou les traitements assumant les transformations progressives de l'information, depuis ses formes concrètes (perception), jusqu'à ses formes plus abstraites (compréhension). Le second (l'axe centré-opératoire) décrit au contraire, de manière duale, les différents niveaux de traitement impliqués, depuis la notion de procédure jusqu'à la notion de tâche **[Ballard 82]**. Il peut conduire à la construction d'arbres de tâches et à une vision distribuée et robuste des questions de coopération et de contrôle. Il conduit de manière indirecte à la généralisation (respectivement. spécialisation) progressive des concepts manipulés en vision (concept d'image, de forme, ou d'objet par exemple). Le premier axe peut être vu comme un axe d'abstraction de l'information, alors que le second peut être perçu comme un axe d'abstraction des tâches.

Connaissances Figuratives

Les systèmes de vision ne travaillent pas directement sur une image, mais en manipulent des descriptions de plus en plus abstraites (pixel, région, contour, forme, objet...). Il est ainsi possible de représenter l'information visuelle à plusieurs niveaux, selon les différents niveaux de représentation manipulés **[Marr 82]**. Une étude des niveaux de représentation en vision par ordinateur et une discussion sur l'existence de niveaux pour la vision humaine peut être trouvée dans **[Demazeau 86]**. Cet auteur cite notamment les travaux de Marr, repris par de nombreux systèmes, qui distingue les niveaux suivants : le niveau "croquis élémentaire" qui correspond aux primitives extraites de l'image telles des régions ou des contours, le niveau "croquis 2,5-D" qui correspond à une description géométrique des primitives et le niveau description 3D qui correspond aux objets et aux relations spatiales entre ces objets. A ces trois niveaux sont également ajoutés le niveau image et le niveau scène.

Le système VISIONS **[Hanson 78]** **[Hanson 87]** maintient ainsi dans une structure de tableau noir une description à plusieurs niveaux d'une image. Les niveaux inférieurs représentent des régions, des segments de ligne et des sommets, et forment un graphe. Ce

graphe est construit par un sous-système de segmentation. Les deux niveaux supérieurs du tableau noir décrivent des surfaces et des volumes, et servent donc à reconstituer la description 3D de la scène. Les deux niveaux du sommet de la hiérarchie représentent des objets et des schémas. C'est également le cas du système de Nagao **[Nagao 80]** qui maintient une structure de tableau noir contenant la description et l'interprétation de l'image en plusieurs niveaux : niveau région élémentaire, niveau région caractéristique, niveau objet et niveau catégorie d'objet. Ce type de représentation a été repris dans de nombreux systèmes tels le système ACRONYM **[Brooks 83]**, le système SATURNE **[Demazeau 86]** et le système de Lane **[Lane 89]**.

Connaissances Opératoires

De nombreux outils ont été définis, que ce soient des outils de segmentation, des outils de manipulation ou des outils d'interprétation. A ces outils élémentaires, que nous avons décrit au paragraphe II, peut se superposer la notion de tâche. Il est alors également possible d'effectuer une abstraction en terme de tâche et de construire une structuration hiérarchique des tâches les plus élémentaires vers les tâches les plus abstraites.

La définition et l'exploitation d'un arbre de tâche ont par exemple été introduites dans le système IBIS **[Dellepiane 89]**. Citons dans ce cadre la tâche de sélection de scène, qui est elle même composée d'une tâche de recherche des objets significatifs, elle même composée d'une tâche de sélection de région, composée des tâches de comparaison d'attributs, de comparaison de relations, et de génération d'objets et d'hypothèses.

2.1.2. Représentation des Connaissances

Le style de représentation des connaissances adopté dépend comme indiqué précédemment du niveau d'abstraction et du type des connaissances considérées.

Approches orientées objet

La modélisation fondée sur les objets ou les frames **[Minsky 75]** offre une approche puissante aux questions de structuration : elle permet d'organiser les connaissances figuratives de manière robuste, et offre la possibilité de lier certains éléments de connaissance opératoire.

Des représentations hiérarchiques sont souvent utilisées, qui modélisent des relations complexes entre des entités de natures diverses **[Hanson 78]**, **[Matsuyama 85]**, **[Woods 89]**, et qui supportent l'introduction de caractéristiques spécifiques telles que la fonction ou le contexte d'apparition des éléments **[Garnesson 89]**.

C'est également le cas du système CLASSIC **[Thonnat 89]**, système dédié à des tâches

d'interprétation. Ce système maintient un ensemble d'objets structurés (prototypes) contenant la connaissance descriptive des objets à classer, et leur attache des règles de production décrivant une connaissance heuristique comme dans **[Granger 85]**.

La connaissance opératoire est introduite dans un frame comme un démon ou une activité réflexe, déclenché lors d'un accès à un attribut. Le rôle de telle connaissance opératoire n'est ni comportemental ni déductif : il est plutôt induit comme un effet de bord pour manipuler les valeurs d'attribut et maintenir la cohérence dans une hiérarchie de frames. En ce qui concerne l'approche orientée objet, la connaissance opératoire est au contraire introduite comme des méthodes activées explicitement par l'envoi de messages et capable d'effectuer n'importe quelle activité procédurale ou inférentielle.

De telles techniques ont été employées assez tôt dans le système VISIONS **[Hanson 78]** et se sont depuis largement répandues dans la communauté de la vision par ordinateur. Cette approche a été largement appliquée à la conception des couches de haut niveau des systèmes de vision, même si des acceptations différentes des notions d'objets ou de frames sont considérées.

Systèmes à base de règles

Les systèmes à base de règles de production promeuvent réciproquement un style de conception orienté action, où les règles peuvent modéliser n'importe quel type d'activité heuristique, déductive ou procédurale **[Nazif 84]**, **[McKeown 85]**. La connaissance figurative dans ce cas est répartie entre les différentes règles et introduite dans le contexte d'une connaissance opératoire donnée, de manière imbriquée. De telles approches sont souvent utilisées pour la conception des couches de bas niveau des systèmes de vision, où le besoin d'activités procédurales est souvent considéré comme plus important que le besoin de modèles de haut niveau. Pour vaincre le manque inhérent de structuration, qui produit une surcharge de gestion, des solutions variées peuvent être envisagées, comme regrouper les règles en modules et / ou introduire des méta-règles **[Nazif 84]**.

Systèmes issus de l'Intelligence Artificielle Distribuée

Des architectures plus complexes ont été proposées, qui permettent un meilleur compromis entre conceptions "centrées objet" et conceptions "centrées tâche". Un effort de structuration dual est alors souvent engagé, se portant également sur les niveaux d'analyse (axe d'abstraction de l'information) et sur les niveaux de traitement à modéliser (axe d'abstraction des tâches).

Le paradigme du tableau noir, initialement introduit par le système VISION **[Hanson 78]** a été largement repris dans ce but, comme le prouvent les nombreuses publications utilisant cette approche **[Tan 86]**, **[Andress 87]**, **[Mader 88]**, **[Garnesson 89]**, **[Tan 89]**,

[Nugues 89], **[Tehrani 89]** et **[Brown 90]**. Plus récemment, des approches de type multi-agents, introduisant la communication par envoi de messages entre plusieurs modules, ont également fait leur apparition **[Lane 89]** et **[Regazzoni 91]**. Une étude de quelques systèmes de vision généraux, incluant les systèmes SATURNE **[Demazeau 86]**, VAP **[Crowley 89]** et MEDUSA **[Aloymonos 90]** fondés sur les paradigmes de l'Intelligence Artificielle Distribuée peut également être trouvée dans **[Boissier 92]**.

Dans les approches de type tableau noir, les tâches sont habituellement modélisées comme des sources de connaissances, ou KS **[Nagao 80]**. Le système MESSIE **[Garnesson 89]** possède ainsi des sources de connaissances spécialisées dans la reconnaissance de types de formes telles que des routes, des bâtiments ou des ombres.

Une structuration supplémentaire de ces KS peut être introduite dans le cadre d'une modélisation par arbre de tâches, où l'interdépendance entre tâches et sous-tâches est modélisée à l'intérieur d'une structure hiérarchique **[Dellepiane 89]**. Le tableau noir contient par ailleurs une représentation structurée des éléments de connaissances, qui sont distribués dans plusieurs niveaux d'analyse et souvent représentés à l'aide de frames **[Hanson 78]**.

Au delà de la notion de tâche ou de spécialiste opératoire, la notion de module expert peut être introduite, comme groupant aussi bien des connaissances figuratives que des connaissances opératoires, correspondant à un niveau d'analyse commun. Ces modules experts sont alors des systèmes experts ou des agents mixtes (agents possédant un tableau noir interne mais communiquant par envoi de messages avec d'autres agents) **[Lane 89]**, **[Regazzoni 91]**. Trois principaux niveaux d'analyse sont souvent distingués, le bas niveau, le niveau intermédiaire et le haut niveau. Le système SIGMA **[Matsuyama 85]**, est ainsi composé de trois modules experts dédiés au raisonnement géométrique, à la sélection de modèles et à la vision de bas niveau. C'est également le cas du projet VISIONS **[Hanson 87]** où le sous système GOLDIE **[Kohl 87]** a été introduit afin d'améliorer les tâches de segmentation.

Le système introduit par LANE **[Lane 89]** introduit finalement une notion de distribution supplémentaire. Les connaissances et les tâches sont en effet réparties dans des agents, eux mêmes distribués sur un réseau de stations de travail.

2.2. CONTROLE

Le contrôle définit la manière de sélectionner et appliquer les éléments de connaissance figurative et opératoire face à un problème donné. Il doit être suffisamment flexible pour manipuler aussi bien les modèles statiques que les éléments créés dynamiquement, comme les résultats intermédiaires ou les hypothèses incertaines, générés pendant la résolution du problème.

La manière dont le contrôle est appliqué dans les systèmes de vision dépend de la manière dont les connaissances sont représentées, mais aussi de la conception de l'architecture. Il est particulièrement important de distinguer le contrôle global du contrôle local : le contrôle global définit des stratégies pour approcher la résolution du problème dans son ensemble alors que le contrôle local est plutôt chargé de décider comment et quand sélectionner tel objet ou telle action.

2.2.1. CONTROLE GLOBAL

Dans les systèmes à base de règles de production, un contrôle global est introduit sous la forme de méta-règles définissant une stratégie globale d'exploration. Ces méta-règles sont par exemple chargées de la sélection des sous-ensembles de règles à appliquer et des critères d'arrêt **[Nazif 84]**.

D'autres systèmes font l'emploi de structures hiérarchiques pour représenter les différents niveaux de traitement des systèmes de vision. Le contrôle global est alors généralement exercé en utilisant une exploration hiérarchique de schémas ou de tâches, selon le style de représentation de ces structures. Une exploration fondée sur les schémas permet par exemple de dériver une succession de buts à atteindre par le système **[Kohl 87]** : des mécanismes de prédiction / vérification sont développés à chaque étape **[Hanson 78]**. Un arbre de tâches **[Dellepiane 89]** ou un agenda hiérarchique **[Woods 89]** sont utilisés au contraire pour développer une planification des tâches. Un superviseur est finalement chargé du contrôle de l'exploration de la hiérarchie utilisée. La résolution d'un problème peut ainsi être vue de manière dualitaire comme l'exploration de sous-buts successifs (parcours de l'axe d'abstraction de l'information) ou comme l'exécution de sous-tâches successives (parcours de l'axe d'abstraction des tâches).

Des modules spécialisés dans des tâches de contrôle sont enfin introduits dans le système proposé par **[Regazzoni 91]**, où des modules de contrôle ou régulateurs sont chargés de contrôler les résultats de modules de traitement, tels le module de calibrage de caméra ou de prétraitement, et éventuellement d'en réguler les paramètres. De tels modules peuvent communiquer explicitement par envoi de messages ou de requêtes **[Regazzoni 91]**, ou implicitement via un tableau noir **[Kohl 87]**. Le contrôle étant ici distribué au sein des régulateurs, il s'exerce de manière plutôt locale.

2.2.2. CONTROLE LOCAL

D'un point de vue local, la sélection d'objets ou d'actions peut impliquer des critères intrinsèques ou des critères contextuels. Les critères intrinsèques impliquent les propriétés des éléments étudiés, tels que la taille d'un objet ou la complexité d'une action. Les critères

contextuels définissent des contraintes sur les situations pour lesquelles sélectionner tel objet ou telle action, qualifiant par exemple le domaine d'application d'une action.

Les deux critères sont utilisés simultanément dans **[Nazif 84]**, où les mécanismes de focalisation peuvent combiner des facteurs d'adjacence (critère contextuel) et de taille (critère intrinsèque). Une approche similaire est proposée par **[Woods 89]**, où des indices images sont tout d'abord localisés en utilisant des contraintes spatiales, puis sélectionnés selon leur facteur de confiance. L'information structurelle est finalement utilisée dans **[Matsuyama 85]** où des hypothèses sont propagées parmi des objets en relation de composition.

La sélection d'action est fondée en général sur des critères contextuels tels que les connaissances sur le domaine d'application de l'action **[Nazif 84]**, ou les connaissances sur le but que chaque action permet d'atteindre **[Kohl 87]**.

3. DISCUSSION

Selon le point de vue algorithmique, il semble important de disposer d'une représentation multi-niveaux des informations et des traitements. Cette représentation multi-niveaux pose alors des problèmes de contrôle en termes de focalisation et d'adaptation. De façon analogue, selon le point de vue logiciel, nous avons montré le besoin d'une distribution des connaissances et des traitements selon deux axes de structuration : l'axe d'abstraction de l'information et l'axe d'abstraction des tâches. Il convient également de représenter le contrôle de manière adaptée. Deux niveaux de contrôle sont utilisés : un contrôle local et un contrôle global.

Nous avons également vu le souci de déclarativité des systèmes de vision qui utilisent les objets et les règles.

Enfin, nous avons vu l'évolution des systèmes vers l'IAD, qui permet la distribution des tâches et des connaissances au sein d'agents, de même qu'une distribution du contrôle. Des stratégies de contrôle, local à l'agent ou plus global peuvent alors être mises en évidence. Notons également l'intérêt de bénéficier dans ce cadre de modèle d'agents génériques qui permet un représentation homogène des différents niveaux de connaissances et de tâches.

IV CONCLUSION

La vision apparait finalement comme un problème complexe mettant en œuvre la modélisation et la manipulation d'objets et d'actions variés, à plusieurs niveaux d'analyse, rendant ainsi difficile la conception de systèmes de vision. Deux notions se dégagent de cet état de l'art : la notion de coopération et la notion de multi-niveaux.

La coopération parait en effet être le maître mot de toute la vision par ordinateur. Au niveau segmentation, mis à part les outils "primitifs", la coopération entre plusieurs techniques est largement employée, cette coopération servant à raffiner, contrôler le processus de segmentation. Au niveau interprétation, là encore la coopération semble être un besoin **[Mohr 88]**. Cet état de fait peut s'expliquer par l'incapacité à trouver un outil suffisamment général pour arriver à un résultat correct dans tous les cas. Il est donc nécessaire d'utiliser plusieurs techniques et de les faire coopérer, de manipuler et d'intégrer plusieurs informations et résultats.

La seconde notion concerne l'introduction de niveaux dans les traitements. En segmentation, ce sont par exemple les approches de type multi-résolution, qui permettent de considérer une image selon plusieurs point de vue et permettent d'améliorer l'efficacité des traitement, du contrôle. Dans les systèmes de vision généraux, il s'agit d'introduire plusieurs niveaux d'abstraction, à nouveau pour des raisons de contrôle, de localisation des traitements. C'est par exemple le cas des systèmes qui manipulent une représentation hiérarchique des différents niveaux de traitement.

Les différentes techniques à mettre en œuvre ont montré l'importance des représentations déclaratives pour modéliser les connaissances, les traitements et le contrôle. Ces besoins de déclarativité ont naturellement débouché sur la conception de systèmes experts, tant au niveau de la segmentation qu'au niveau de l'interprétation. Parmi ceux-ci, nous pouvons également noter une évolution : des premiers systèmes à base de règles, les chercheurs se sont rapidement orientés vers des systèmes permettant une meilleure structuration, une meilleure introduction du contrôle, conséquence de la diversité des connaissances et des outils employés, et une plus grande généricité.

Dans ce cadre, les techniques issues de l'Intelligence Artificielle Distribuée sont de plus en plus employées, introduites assez tôt avec le concept du tableau noir **[Hanson 78]**, elle se répandent largement, l'évolution actuelle outre une distribution en terme de tâche et de connaissances allant vers une distribution physique, due à l'évolution du matériel (introduction de ressources physiques dédiées, parallélisme, vitesse des traitements...). Les approches issues des techniques d'Intelligence Artificielle Distribuée sont effet appropriées, particulièrement en ce qui concerne la question de l'organisation et de la flexibilité : de telles approches sont en effet basées sur la distribution des connaissances et des tâches, et une résolution coopérative du problème.

L'Intelligence Artificielle Distribuée semble également apporter des solutions au besoin de représentation multi-niveaux, avec la possibilité de définir des architectures hiérarchiques d'agents **[Lane 89]**. Dans ce cadre, des architectures en couches introduisant des couches réactives correspondant aux niveaux les plus bas de la hiérarchie de traitement, et des couches

plus cognitives correspondant aux niveaux les plus élevés et contrôlant les couches inférieures pourraient être prometteuses.

CHAPITRE 2

MOTIVATIONS

I INTRODUCTION

Notre objectif est la conception d'un système de vision dédié à l'analyse d'images biomédicales. Les systèmes de vision introduisent une distribution des connaissances en niveaux selon deux axes orthogonaux, un axe figuratif et un axe opératoire. Dans ce cadre, il nous parait important de disposer d'un mécanisme de représentation, d'une structure générique qui permette d'appréhender les différents niveaux de représentation de manière assez homogène. Nous avons donc décidé de définir une architecture multi-agents représentant différents niveaux de représentation à partir de l'environnement de programmation MAPS. MAPS semble en effet particulièrement bien adapté à ce problème de représentation grâce à la définition de deux types d'agents génériques chargés de représenter et de manipuler des connaissances figuratives (KS) et opératoires (KP).

La question du contrôle s'avère également de première importance. Il s'agit en effet de pouvoir représenter les mécanismes de focalisation agissant à travers ces niveaux de représentation des connaissances **[Boissier 92]**. Ici encore l'environnement MAPS apporte des solutions, en permettant d'introduire des stratégies de contrôle dédiées aux connaissances figuratives et opératoires de manière distincte. A un niveau figuratif, le contrôle concernera la sélection d'informations (focalisation sur des données ou hypothèses) et à un niveau opératoire la sélection d'un traitement adapté à ces informations.

Nous souhaitons également introduire une coopération région / contour parmi les différents niveaux retenus. Il s'agira ici d'utiliser les résultats des détections des régions et des contours pour bâtir une représentation des relations (topographiques) entre ces primitives afin d'obtenir à la fois une interprétation sémantique des images et des stratégies de manipulation pour raffiner la segmentation initiale.

Notons enfin que les outils introduits pour la détection des primitives, la mise en correspondance de ces primitives et l'interprétation sémantiques seront très simples. Il s'agit en effet de tester plutôt la robustesse de l'architecture globale et l'intérêt de l'environnement MAPS pour la vision, que d'obtenir des résultats excellents. Nous pensons en outre que l'efficacité du système viendra plus de la complexité de l'architecture que de la complexité des algorithmes mis en œuvre.

II CONNAISSANCES ET REPRÉSENTATION DES CONNAISSANCES

Notre domaine d'application est l'imagerie biomédicale et nous nous intéressons plus précisement à des images issues de la microscopie, donc bidimensionnelles. Les choix quant au découpage en niveaux de représentation en découlent naturellement. Nous avons choisi de distribuer les connaissances figuratives selon quatre niveaux d'abstraction : un niveau image, un niveau indice, un niveau représentation et un niveau objet. Le niveau image correspondra à l'image discrète, le niveau indice correspondra aux primitives région et contour, le niveau représentation correspondra à une description géométrique et relationnelle (relation topographique entre régions et contours), le niveau objet correspondra à l'interprétation sémantique des primitives. Ces différents niveaux seront distribués au sein d'agents KS.

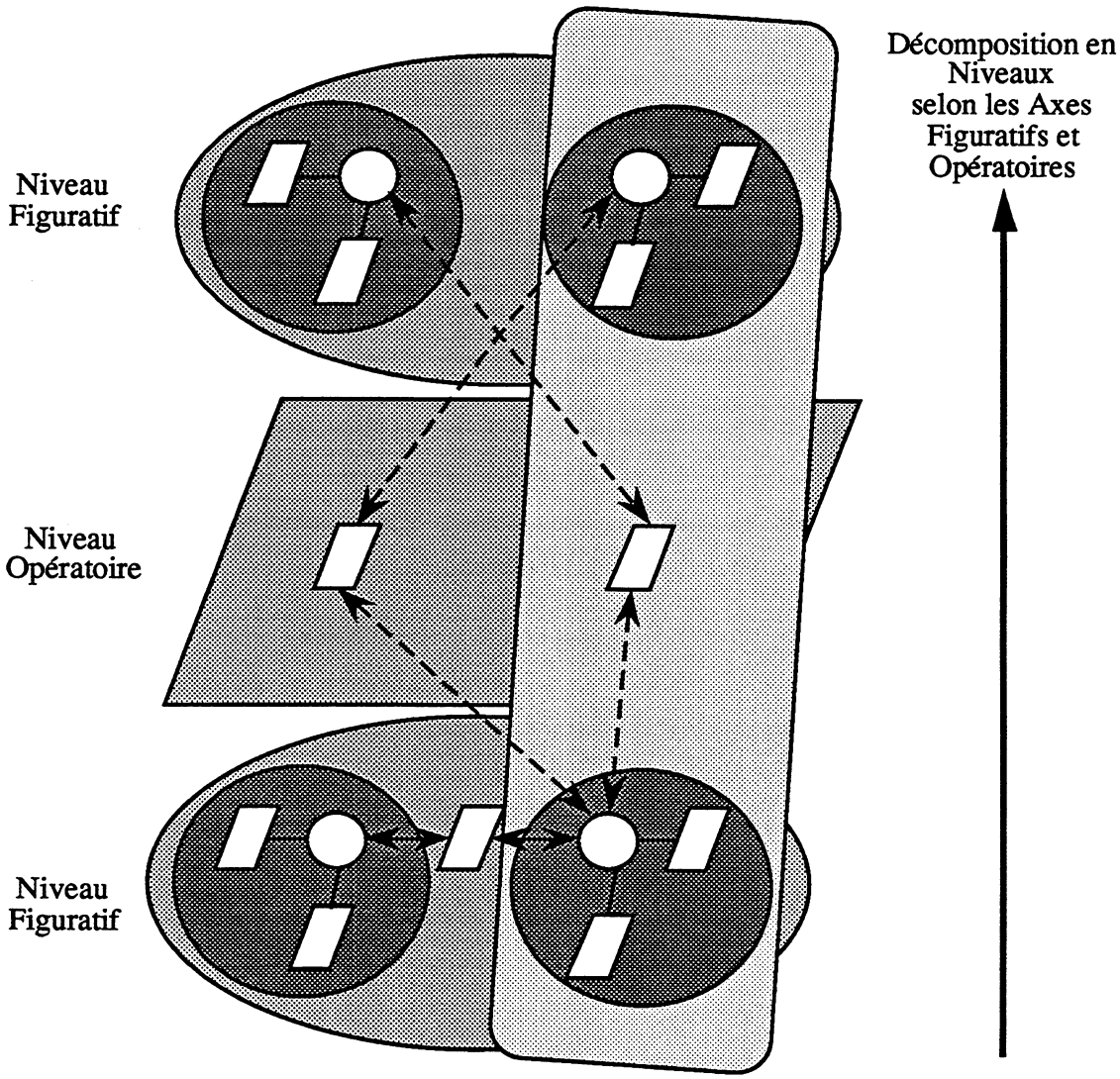
Trois niveaux opératoires vont être également définis, permettant de se déplacer parmi ces niveaux figuratifs, qui correspondent à une phase d'analyse de bas niveau, une phase d'analyse de niveau intermédiaire et une phase d'analyse de haut niveau. La phase d'analyse de bas niveau permettra de passer du niveau image au niveau indice en effectuant une détection des primitives région et contour. La phase d'analyse de niveau intermédiaire va ensuite construire le niveau représentation à partir de ce niveau indice. La phase d'analyse de haut niveau permettra finalement de passer du niveau représentation au niveau objet (interprétation sémantique) et du niveau représentation au niveau indice (manipulation de primitives). Ces différents niveaux seront distribuées au sein d'agents KP.

Enfin, les images seront analysées selon deux points de vue différents : on s'attachera en effet à détecter des régions et des contours. Nous introduisons ainsi la notion de point de vue figuratif (type d'élément manipulé) et de point de vue opératoire (filière d'analyse correspondante). Notons enfin que plusieurs points de vue peuvent définir un même niveau de représentation.

Nos images étant bidimensionnelles, nous nous sommes limités à ces niveaux de représentation, n'ayant pas besoin des niveaux distingués dans des systèmes traitant des images tridimensionnelles. Nous pensons en outre que si MAPS permet de créer une architecture valide, celle-ci pourra être aisément étendue par modification des agents définis, par ajout de nouveaux agents et de nouveaux niveaux de représentation pour atteindre une architecture 3D.

III CONTROLE

Nous avons vu dans le chapitre précédent (partie B, chapitre 1) toute l'importance du contrôle et de sa représentation dans les systèmes de vision. Ce contrôle doit également être distribué afin d'obtenir une représentation des stratégies de focalisation entre les différents niveaux de représentation des connaissances.



Point de vue opératoire ou groupe centré tâche

Point de vue figuratif ou groupe centré connaissance

Niveau d'abstraction

Agent KS

Agent KP

Fig. B.1. Représentation d'une architecture multi-agents MAPS selon les niveaux figuratifs et opératoires

La distribution du contrôle va venir tout naturellement de la distribution de ces niveaux de représentation au sein des agents KS et KP. La figure **B.1** présente une telle distribution. Quatre types de contrôle peuvent alors être définis : un contrôle intra-agent correspondant au contrôle interne de l'agent, un contrôle inter-agent correspondant au contrôle externe de l'agent, un contrôle intra point de vue correspondant à des stratégies locale de raffinement et un contrôle inter point de vue correspondant au passage d'un niveau à l'autre ou au basculement d'un point

de vue à l'autre. Notons que la notion de point de vue et de niveau ne sera jamais présente de manière explicite par la suite. Des stratégies de focalisation et d'adaptation sophistiquées, que nous présentons dans un premier paragraphe, vont ainsi pouvoir être dégagées à l'intérieur et entre les différents niveaux de représentation. Ces possibilités de focalisation entre les différents niveaux de représentation vont finalement nous permettre d'introduire plusieurs sortes de coopération que nous présentons dans un second paragraphe.

1. Focalisation et Adaptation

Nous allons dans ce paragraphe présenter la manière dont la focalisation et l'adaptation vont s'exercer pour chacun des types de contrôle. La focalisation et l'adaptation peuvent intervenir à deux niveaux : à un niveau local (local à un agent, local à un niveau de représentation) et à un niveau global (entre deux agents ou niveaux de représentation).

Contrôle intra-agent

Le contrôle intra-agent s'exerce pour les agents KS comme pour les agents KP. En ce qui concerne les agents KS, il s'agit de se focaliser sur une hypothèse particulière (règle de proposition et d'interrogation). En ce qui concerne les agents KP, il s'agit d'adapter un traitement à une hypothèse (règles de stratégie). Ce contrôle local (local à l'agent) va par exemple permettre la focalisation sur des hypothèses de segmentation ou d'interprétation et l'adaptation des procédures de traitement en fonction de ces hypothèses.

Contrôle inter-agent

Le contrôle inter-agent est d'un niveau plus global et s'exerce pour les agents KS comme pour les agents KP. En ce qui concerne les agents KS, il s'agit de définir des stratégies de traitement en fonction des connaissances locales (règles de proposition et d'interrogation). En ce qui concerne les agents KP, il s'agit de définir des stratégies de collection d'informations et de diffusion des résultats en fonction des traitements effectués (règles d'action).

Contrôle intra point de vue

Le contrôle intra point de vue concerne un niveau de contrôle local à un niveau figuratif. Ce contrôle agit sur les points de vue figuratifs, c'est-à-dire les groupes centrés connaissance. Il va définir des stratégies de raffinement de données locales à ces points de vue (raffinement de l'image initiale par filtrage au sein du niveau image par exemple).

Contrôle inter point de vue

Le contrôle inter point de vue est plus global. Il intervient dans des points de vue opératoires, c'est-à-dire les groupes centrés tâche qui permettent de passer d'un niveau d'abstraction des

connaissances à un autre, pour un point de vue donné. Il s'exerce également entre deux points de vue d'un même niveau figuratif et concerne le basculement d'un point de vue à l'autre (passage du point de vue région au point de vue contour dans le niveau indice par exemple). Le contrôle est dans ce cas régi par les agents KP.

Il s'exerce également entre deux points de vue opératoires et concerne les problèmes de synchronisation, de planification entre deux filières d'analyse. Le contrôle est dans ce cas régi par les agents KS (règles de proposition).

2. Coopération

Le problème de la coopération se pose de plusieurs façons simultanées. Nous souhaitons tout d'abord introduire une coopération entre les deux approches fondées sur les régions et sur les contours. Cette coopération passe par la création de deux filières d'analyse (une filière région et une filière contour) au sein des différents niveaux de représentation, c'est-à-dire par la création de différents points de vue, tant figuratifs qu'opératoires. Comme nous l'avons vu au paragraphe précédent, plusieurs types de contrôle interviennent et vont permettre alors plusieurs types de coopération.

En particulier, le contrôle inter point de vue va permettre de définir une coopération entre ces deux filières, il va en effet permettre de définir des stratégies de synchronisation. Il va également permettre de définir une coopération entre les différents niveaux figuratifs. Nous souhaitons par exemple utiliser des informations relationnelles du niveau représentation (relation topographique comme une intersection ou une inclusion d'une région par un contour) pour définir des stratégies de manipulation permettant de repasser au niveau indice (par décomposition ou fusion). Par ailleurs, une relation comme la correspondance entre une région et un contour doit être utilisée pour bâtir une représentation au niveau objet (étape d'interprétation sémantique). Finalement, des informations du niveau objet doivent permettre de revenir sur les niveaux inférieurs.

IV Conclusion

Le système de vision que nous souhaitons construire va donc introduire les principales caractéristiques des systèmes de vision, et en particulier une distribution originale des connaissances et du contrôle (en terme de focalisation et d'adaptation) à différents niveaux. L'apport de notre approche vient de l'architecture logicielle utilisée : l'environnement de programmation MAPS qui convient particulièrement bien à la création d'un système de vision. La définition de deux types d'agents représentant et manipulant les connaissances figuratives et opératoires (KS et KP), l'introduction de groupes centrés connaissances et centrés tâches

permet en effet d'appréhender les niveaux de connaissances et le contrôle selon les axes figuratifs et opératoires d'une manière homogène (les agents définis seront fondés sur des modèles génériques spécialisables) qui nous semble-t-il n'a pas encore été introduite dans les systèmes existants. Le système ainsi conçu nous permettra en retour de valider les concepts de base de l'environnement MAPS, d'en cerner les faiblesses et de définir ainsi des objectifs d'extensions.

CHAPITRE 3

KISS

I INTRODUCTION

KISS (Knowledge-based Image Segmentation System) a été conçu pour fournir une modélisation haut niveau du problème de la Vision par Ordinateur **[Baujard 89a], [Baujard 89b], [Baujard 91b]**. Comme le montre la figure **B.2**, KISS est conçu comme une application de l'environnement de programmation MAPS, et constitue un réseau complexe d'agents KS / KP.

Les agents KS sont chargés de maintenir les données et les résultats des différents traitements effectués lors des tâches de vision. Chaque agent est lié à un niveau de représentation de l'image : niveau image (KS-Sensory), niveau indice (KS-Region et KS-Contour), niveau de description et de mise en relation des primitives (KS-Form et KS-Relation) et niveau objet (KS-Semantic). A chaque agent KS est également attaché un agent KP qui lui est propre. Il fournit plutôt un travail de type procédural et utilitaire, local à ce niveau de représentation. Cet agent non représenté, appelé agent KP-Descriptif, est ainsi chargé du calcul de nombreux descripteurs. De tels groupes d'agents correspondent alors à des groupes centrés connaissance.

Rappelons que le système MAPS implique une distinction fondamentale entre environnement de perception et environnement de compréhension. Les premiers impliquent des instances référant directement des événements effectivement perçus alors que les seconds réfèrent des instances qui correspondent à l'identification (compréhension) d'événements perçus par ailleurs. Deux sortes d'agents KP peuvent ainsi être distingués, selon qu'ils manipulent des règles de traitement faisant appel à des procédures d'analyse d'images (KP-Segm, KP-Detect, KP-Close et KP-Handle par exemple) ou des règles d'inférence pures (KP-Geometrical, KP-Relational et KP-Semantical). Ils peuvent puiser des informations ou transmettre des résultats dans plusieurs agents KS. De tels groupes d'agents correspondent alors à des groupes centrés tâche.

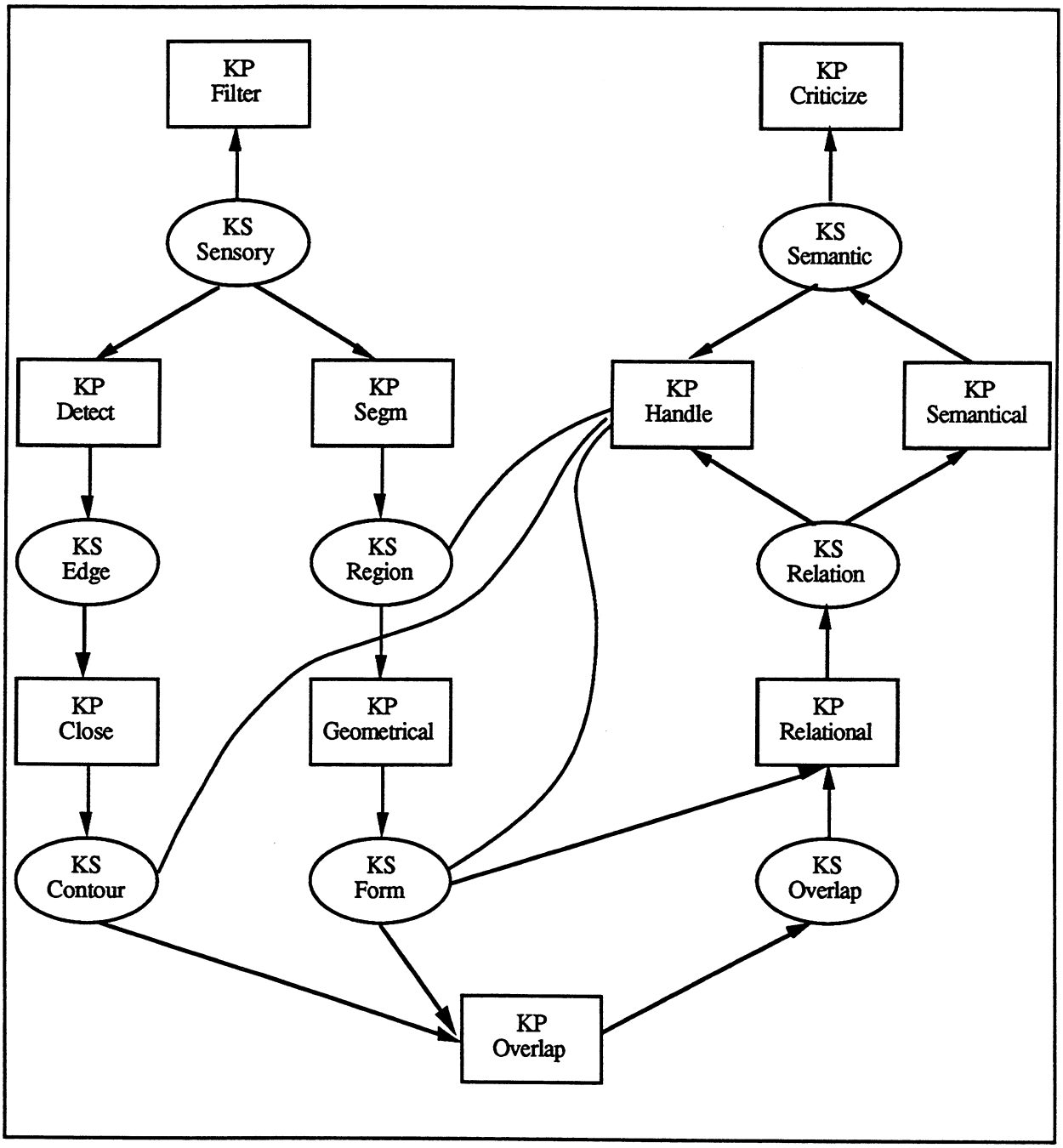


Fig. B.2. L'architecture de KISS : vue globale.

Ce modèle est issu d'une décomposition en tâches propres au problème de la vision. Un aperçu des principales tâches d'analyse est fourni en figure **B.3.** Comme on peut le voir, deux filières d'analyse séparées sont envisagées (correspondant à des points de vue opératoires), l'une fondée sur l'analyse des régions et l'autre fondée sur l'analyse des contours. L'information image est en outre analysée selon des points de vue géométriques, relationnels et sémantiques. L'étape relationnelle est conçue pour assister la segmentation région en utilisant les résultats de la segmentation contour : l'interprétation sémantique est effectuée en cas de correspondance entre l'information région et l'information contour disponible, alors qu'une manipulation des

primitives (fusion ou décomposition) est effectuée dans le cas contraire. Une telle manipulation peut fournir des informations structurelles additionnelles, qui peuvent être utilisées pour confirmer ou infirmer l'interprétation sémantique courante. Un retour peut finalement intervenir, de l'interprétation sémantique vers la manipulation des primitives, dans le cas d'entités complexes comme des agrégats de plusieurs entités similaires : une procédure de décomposition sera appliquée dans un tel cas.

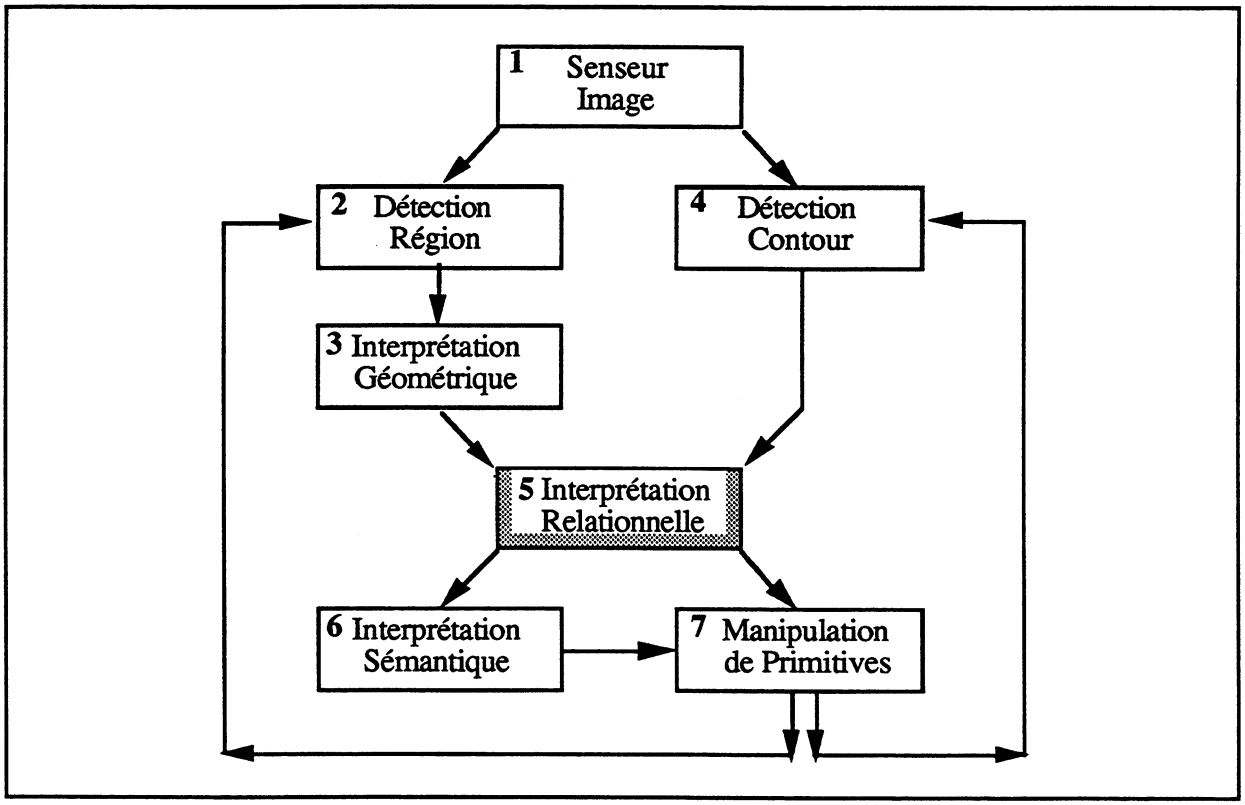


Fig. B.3. Modélisation orientée tâche du problème de vision : les nombres indiquent l'ordre des tâches.

Trois phases principales d'analyse sont distinguées (phases d'analyse de bas niveau, de niveau intermédiaire et de haut niveau); elles seront décrites successivement par la suite. Nous donnerons pour chaque phase une figure globale décrivant ses étapes principales et pour chaque agent une description de ses ressources et comportements.

II ANALYSE BAS NIVEAU

Le rôle de la phase d'analyse de bas niveau est de fournir des primitives pour la phase d'interprétation. Comme nous souhaitons formaliser la coopération région / contour, nous avons introduit deux filières d'analyse bas niveau, une filière de détection des régions et une filière de détection des contours.

Description Globale

Une figure globale décrivant la structure de la phase d'analyse bas niveau est donnée en figure **B.4.** Une caractéristique principale, comme on peut le voir, est l'initiation de deux filières d'analyse indépendantes à partir de l'agent KS Sensory, l'une fondée sur les régions (agent KP Segm) et l'autre sur les contours (agent KP Detect), qui sont suivies tout au long de cette phase. Il est à noter que KISS est censé agir comme un analyseur de type région, l'information contour étant utilisée comme référence, pour assister, guider et contrôler les traitements mis en œuvre pendant les étapes de niveau intermédiaire et de haut niveau. Un ajustement coopératif de paramètre de traitement d'image sera illustré dans cette phase : l'agent KP-Filter est activé de manière itérative, afin de filtrer l'image selon des degrés de filtrage progressivement plus élevés jusqu'à l'obtention d'un certain résultat de segmentation dans l'agent KP-Segm.

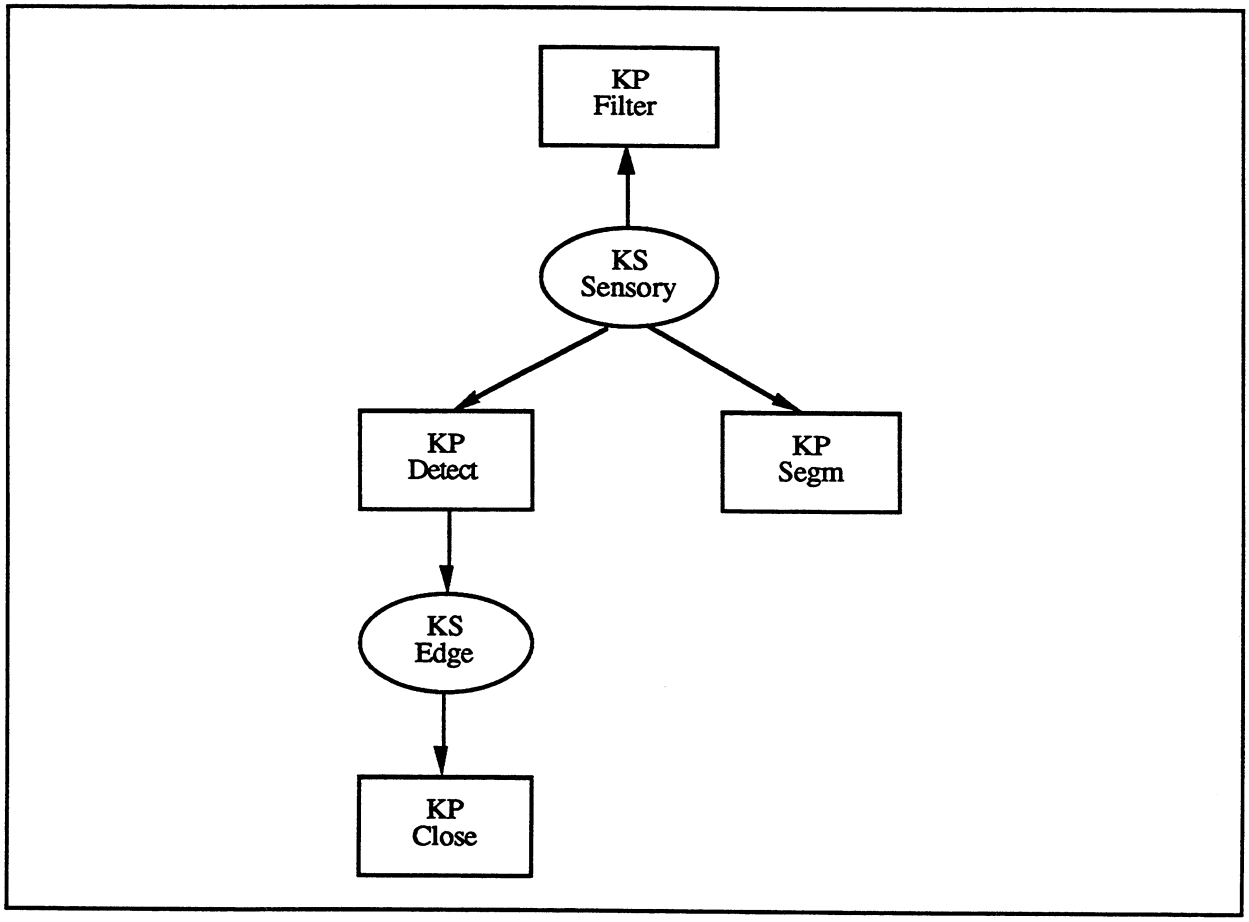


Fig. B.4. L'architecture de KISS : agents intervenant dans la phase d'analyse bas niveau.

Agent Initiateur

Le rôle et les ressources de l'agent KS-Sensory sont brièvement présentés dans la **Table 1.** L'activation de cet agent débute l'exécution du système : les règles d'interrogation sont tout d'abord déclenchées pour compléter la connaissance de l'agent sur les caractéristiques de

l'image. Une règle de propagation peut alors être activée pour charger une image dans la base de connaissance de l'agent. Cette image est ensuite proposée aux étapes de détection par le biais des règles de proposition. Une telle proposition est effectuée selon une stratégie prédéfinie : l'agent KP-Segm est tout d'abord activé, pour procéder à une segmentation région : l'agent KP-Detect est activé ensuite, pour procéder à la détection des frontières.

Agent	**KS-Sensory**
Rôle	Représenter et Gérer l'Image Initiale à Traiter
Ressources	--> Objets : *Image* : Caractéristiques de l'image & Taille de la fenêtre du Filtre Médian *Filtered-Image* : Caractéristiques de l'image, Nombre de classes dans l'histogramme des niveaux de gris & Seuils de segmentation
	--> Règle de Propagation : SI Les caractéristiques image de l'objet *Image* ont été obtenues ALORS Charger l'image à étudier et la stocker dans une instance de l'objet *Image*
	--> Règle de Proposition : SI Une instance de l'objet *Image* est chargée ALORS (1) Proposer cette instance à l'agent KP-Segm (2) Proposer cette instance à l'agent KP-Detect
	--> Règles d'Interrogation : Demander à l'utilisateur les attributs de l'objet *Image*

Table 1 : Ressources de l'Agent KS-Sensory

1. FILIERE DE DÉTECTION DES RÉGIONS

La filière de détection des régions met en œuvre deux agents, nommés KP-Filter et KP-Segm. Les ressources de ces agents sont respectivement présentées en **Table 2** et **Table 3**.

1.1. OBJECTIFS

L'objectif de cette filière est d'effectuer une segmentation région. La technique retenue est la classification, l'histogramme des niveaux de gris est découpé en classes, l'image étant ensuite segmentée selon les classes obtenues. Une autre caractéristique de la filière est l'application itérative d'un filtre, afin d'optimiser un critère de qualité de la segmentation.

1.2. DESCRIPTION DE LA FILIERE

Le rôle de l'agent KP-Filter est d'appliquer un filtre médian sur l'image initiale. La taille de la fenêtre du filtre est incrémentée à chaque activation de la règle : le degré du filtrage peut ainsi être modifié itérativement, par des appels successifs à cet agent (voir **Table 3**). La taille de la fenêtre et l'image filtrée résultante sont finalement renvoyées à l'agent KS-Sensory.

Agent	**KP-Filter**
Rôle	Filtrer l'Image sous Etude
Ressources	--> Règle d'Action : SI (1) Collecter les caractéristiques de l'image dans l'agent KS-Sensory (2) Collecter la taille de la fenêtre du filtre médian dans l'agent KS-Sensory (3) Augmenter la taille de la fenêtre du filtre médian ALORS (1) Appliquer le filtre médian sur l'instance de l'objet *Image* (2) Transmettre le résultat (instance de l'objet *Filtered-Image*) à l'agent KS-Sensory (3) Transmettre la taille de la fenêtre du filtre médian à l'agent KS-Sensory

Table 2 : Ressources de l'Agent KP-Filter

Le rôle de l'agent KP-Segm est d'effectuer la segmentation de l'image, requise par l'agent KS Sensory. Un filtrage est appliqué itérativement jusqu'à l'obtention d'un critère de qualité, c'est à dire un nombre de petites régions suffisamment faible. Le principe utilisé en celui décrit dans le paragraphe sur les techniques de programmation avancé (partie A, chapitre 3). L'agent KP Filter est réactivé à chaque itération avec une taille de fenêtre augmentée. Une image plus lisse

est obtenue, et le nombre de petites régions détectées dans l'image segmentée diminue en conséquence. Les seuils de segmentation sont calculés selon l'algorithme de Fisher **[Fisher 58]** par un agent KP-Descriptif additionnel, à la requête de l'agent KS-Sensory. Le nombre de classes peut être fixé manuellement ou calculé par l'agent KP-Descriptif selon l'algorithme de Bhattacharya **[Bhattacharya 67]**.

Les résultats de la segmentation, c'est à dire l'image étiquetée et le nombre de régions sont alors transmis à l'agent KS-Region ; une boucle est finalement introduite pour contrôler l'envoi itératif des étiquettes de région à cet agent.

Agent	**KP-Segm**
Rôle	Effectuer une Segmentation Région
Ressources	--> Règle d'Action : SI (1) Collecter les caractéristiques de l'image dans l'agent KS-Sensory (2) Répéter jusqu'à "Nombre de petites régions < Seuil" ou "Nombre maximum d'itérations atteint" (a) Détruire l'instance de l'objet *Filtered-Image* dans l'agent KS-Sensory (b) Collecter une instance de l'objet *Filtered-Image* dans l'agent KS-Sensory (c) Segmenter et étiqueter l'image filtrée (d) Calculer le "Nombre de petites régions" ALORS (1) Transmettre l'image des régions à l'agent KS-Region *cc : une instance de l'objet *Region-Image* sera créée* (2) Transmettre itérativement l'étiquette d'une région à l'agent KS-Region

Table 3 : Ressources de l'Agent KP-Segm

1.3. DISCUSSION

Les ressources de l'agent sont encore rudimentaires, mais peuvent être aisément augmentées en incluant des procédures de segmentation région variées. Des règles de stratégie, introduites dans chaque agent, permettront de sélectionner la règle optimale dans une situation donnée.

L'ajustement itératif de paramètre se révèle être une caractéristique puissante, qui pourrait être utilisée plus souvent. Des techniques de programmation de plus haut niveau devront cependant être introduites dans ce sens, pour améliorer leur implantation. L'intérêt de distribuer les procédures de filtrage et de segmentation dans des agents différents, et donc d'introduire une approche coopérative pour ce filtrage itératif, réside dans une plus grande modularité et une plus grande autonomie de décision quant au choix de la procédure. Le choix est en effet dicté par des règles de stratégie propres à l'agent.

Il faut également noter que le filtrage progressif avec une taille de fenêtre de plus en plus grande, introduit une représentation multi-niveaux de l'image : à chaque niveau correspond une image de plus en plus filtrée, où des détails ont été gommés. Actuellement, seule l'image finale est conservée. Il serait intéressant de conserver toutes les images correspondant aux différents niveaux de filtrage, afin de pouvoir revenir sur une image moins filtrée en cours d'exécution.

2. FILIERE DE DÉTECTION DES CONTOURS

La filière de détection des contours met en œuvre trois agents, nommés KP-Detect, KS-Edge et KP-Close.

2.1. OBJECTIFS

L'objectif de la filière de détection des contours est d'effectuer une segmentation contour afin de fournir des références pour l'amélioration de la segmentation région. Il s'agit donc ici d'obtenir des contours fermés afin de pouvoir les comparer aux régions. Une étape d'amélioration est ainsi introduite, chargée de nettoyer l'image des points de frontière et de reconnecter les contours ouverts.

2.2. DESCRIPTION DE LA FILIERE

L'agent KS-Sensory envoit une requête de traitement à l'agent KP-Detect dès que la détection des régions et l'interprétation géométrique ont été effectuées (filière de détection des régions et filière d'analyse intermédiaire). Ses ressources sont décrites en **Table 4** : une règle d'action est définie, qui effectue un rehaussement de contraste suivie d'une détection utilisant l'opérateur PREWITT. L'image résultante est envoyée à l'agent KS-Edge, qui la transmet ensuite simplement à l'agent KP-Close, agissant ainsi comme un buffer d'entrée/sortie synchronisant les actions successives mises en œuvre par deux agents différents.

Agent	**KP-Detect**
Rôle	Effectuer la Détection des Points de Frontière
Ressources	--> Règle d'action : SI (1) Collecter les caractéristiques de l'image dans l'agent KS-Sensory (2) Initialiser les paramètres des procédures de traitement bas niveau ALORS (1) Appliquer un rehaussement de contraste (2) Appliquer une détection de gradient suivant l'algorithme de PREWITT (3) Segmenter l'image gradient résultante (4) Transmettre le résultat à l'agent KS-Edge *cc : une nouvelle instance de l'objet *Edge-Point-Image* sera créée dans l'agent*

Table 4 : Ressources de l'Agent KP-Detect

Les ressources de l'agent KP-Close sont décrites en **Table 5**. Son rôle est de manipuler l'image brute *Edge-Point-Image* afin d'obtenir des contours fermés, réalisant ainsi une partition du domaine de l'image. Cette fermeture de contour, fondée sur des manipulations morphologiques, est illustrée par la figure **B.5**.

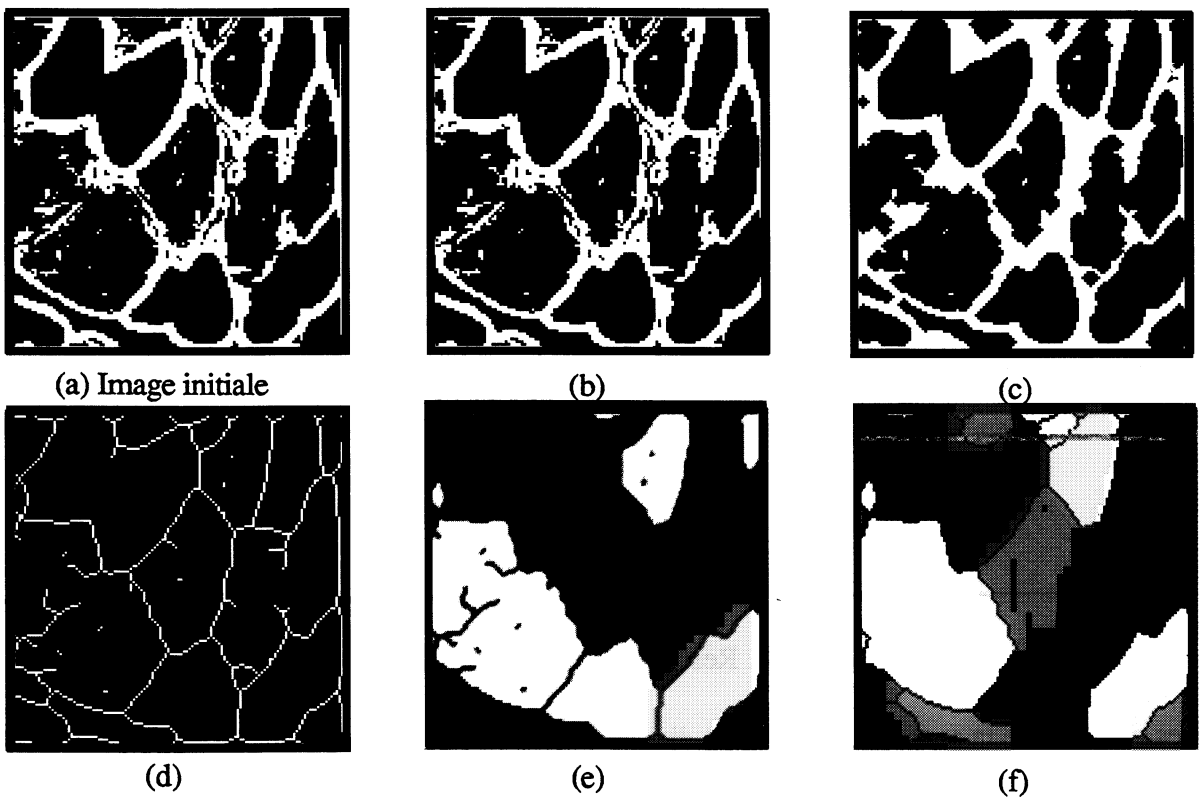


Fig. B.5. Etape de manipulation de l'image des points frontière.

Les points frontière isolés sont supprimés en premier (b), et une fermeture binaire est ensuite appliquée pour supprimer les discontinuités locales (c). Un squelette est ensuite calculé selon l'algorithme de Zhang et Suen **[Zhang 84]** (d); il est suivi d'un étiquetage des composantes connexes des régions du fond (e). Une fermeture binaire est finalement appliquée à chaque région étiquetée pour supprimer les segments de contour pendants (f). Les régions obtenues constituent alors les supports de l'information contour qui sont utilisés pendant les étapes d'analyse suivantes.

Agent	**KP-Close**
Rôle	Partitionner le Domaine d'Image en Contours Fermés
Ressources	--> Règle d'Action : SI (1) Collecter l'instance de l'objet *Edge-Point-Image* dans l'agent KS-Edge (2) Initialiser les paramètres des procédures de bas niveau ALORS (1) Supprimer les points frontière isolés (2) Appliquer une fermeture binaire (3) Calculer le squelette des frontières selon l'algorithme de Zhang & Suen (4) Etiqueter les composantes connexes correspondant aux régions circonscrites (5) Appliquer une fermeture binaire à l'intérieur de chaque région étiquetée (6) Transmettre le résultat à l'agent KS-Contour *cc : une instance de l'objet *Contour-Image* sera créée* (7) Boucler sur : Transmettre l'étiquette d'un contour fermé (instance de l'objet *Current-Contour*) à l'agent KS-Contour (8) Transmettre un message d'activation à l'agent KS-Form

Table 5 : Ressources de l'Agent KP-Close

Les résultats de la segmentation, c'est à dire l'image étiquetée et le nombre de contours détectés, sont alors transmis à l'agent KS-Contour; une boucle est ensuite introduite pour contrôler l'envoi itératif des contours étiquetés à cet agent. On doit noter ici que l'analyse fondée sur les

contours est supposée fournir des informations contour robustes mais approximatives, devant être utilisées comme références pour contrôler les résultats de l'analyse région.

2.3. DISCUSSION

Les mêmes remarques que pour la détection de région peuvent être faite en ce qui concerne les algorithmes de détection des contours : les techniques de détection de Canny **[Canny 86]** ou Deriche **[Deriche 87]** pourraient par exemple être introduites. Le traitement des contours pourrait aussi être amélioré : il faudrait pour cela utiliser des procédures de reconnexion sophistiquées qui permettraient d'obtenir des contours fermés qui préserveraient plus les informations frontière à l'intérieur de l'image.

Globalement, les différents opérateurs introduits dans la règle de l'agent KP-Close pourraient être redistribués au sein de plusieurs agents KP. Cette solution permettrait ainsi de bénéficier du caractère coopératif de l'approche multi-agents et d'introduire des mécanismes d'ajustement de paramètres et de sélection d'opérateurs tels que ceux décrits pour la filière de détection des régions.

3. DISCUSSION

Cette phase d'analyse bas niveau est chargée de la détection des primitives région et des primitives contour. Les deux filières introduites sont entièrement indépendantes et pourraient donc être effectuées en parallèle. Dans la version actuelle de MAPS, ce parallélisme n'est pas mis à profit, la filière de détection des régions est considérée avant la filière de détection des contours. Une version parallèle de l'environnement MAPS pallierait ce besoin intrinsèque. Plusieurs algorithmes pourraient être ainsi mis en œuvre en parallèle pour chacune des deux filières, suivi alors d'une comparaison entre les différents résultats, qui sélectionnerait le meilleur. Déclencher plusieurs algorithmes en parallèle permettrait également de maintenir plusieurs hypothèses de segmentation en parallèle : ce mécanisme est celui des mondes multiples. La sélection des meilleures hypothèses se ferait alors non pas de manière immédiate mais plus tard au vu d'informations de plus haut niveau.

Il serait également intéressant de fonder la phase d'analyse de bas niveau sur une représentation multi-échelle. Une représentation pyramidale parait en effet indispensable afin de raisonner à différents niveaux de représentation de l'image, et afin d'introduire des mécanismes de focalisation rapides pour les phases d'analyse suivantes. Cette représentation nécessite cependant l'introduction d'agents de granularité différente et l'introduction d'une structuration hiérarchique des agents.

III ANALYSE INTERMÉDIAIRE

L'objectif de la phase d'analyse intermédiaire est de bâtir une modélisation de haut niveau des primitives région et contour détectées durant la phase d'analyse de bas niveau. KISS est en effet un système fondé sur la coopération région / contour. Une représentation des relations existant entre les primitives région et contour est ainsi nécessaire, afin d'exploiter d'éventuelles correspondances ou des différences marquantes à des fins d'interprétation sémantique ou de manipulation des primitives.

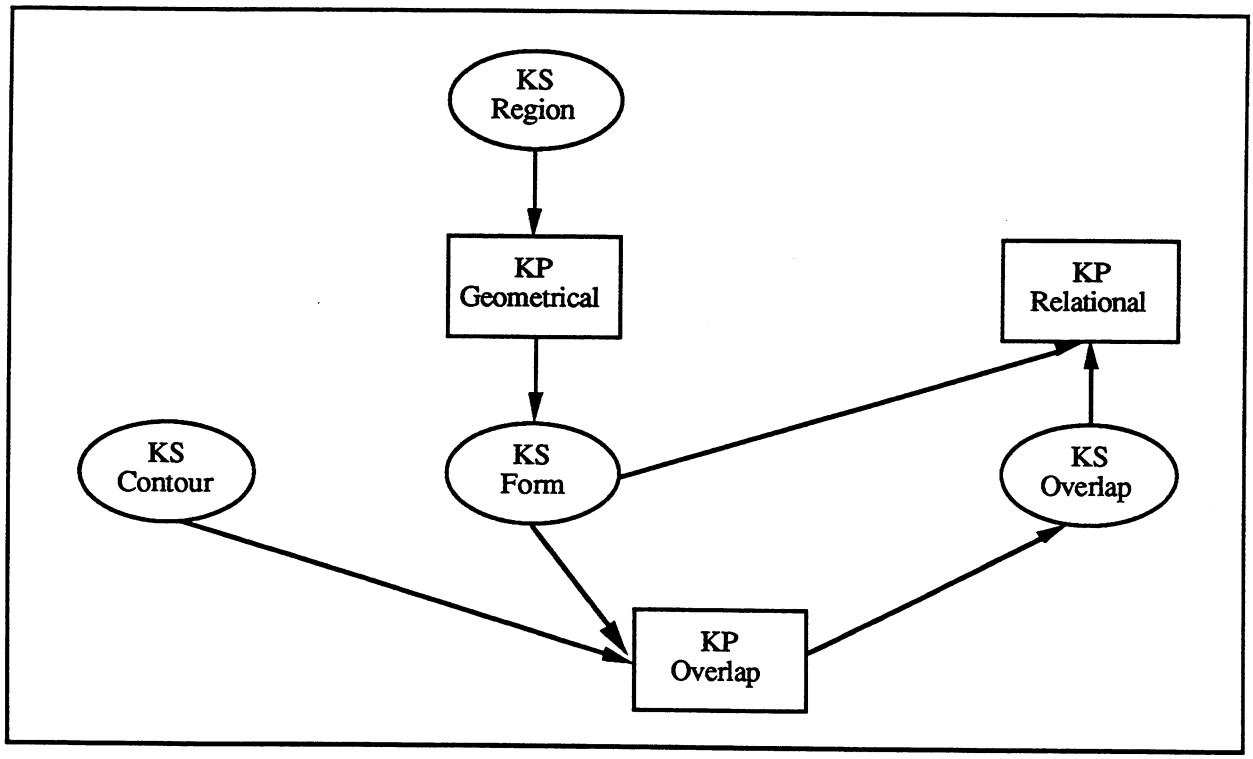


Fig. B.6. L'architecture de KISS : les agents intervenant dans la phase d'analyse intermédiaire.

Une figure globale décrivant les principales étapes mises en œuvre dans l'analyse de niveau intermédiaire est fournie en figure **B.6.** Deux filières séparées d'interprétation sont décrites, impliquant un raisonnement géométrique (KP-Geometrical) et relationnel (KP-Relational). Le raisonnement relationnel est fondé sur l'analyse des relations spatiales apparaissant entre des régions et des contours (KS-Overlap), cette étape est essentielle car la phase d'analyse suivante est fondée sur les résultats de ce raisonnement. Une autre caractéristique essentielle de cette filière est le basculement du comportement d'un agent d'un état passif (simple mise à jour et maintenance de l'information) à un état actif (sélection et transmission d'information vers l'extérieur). Un tel basculement est décidé par l'agent lui-même, à la réception d'une information particulière venant de l'extérieur. Il est utilisé pour synchroniser les activités de l'agent, c'est à dire être sûr qu'un agent attend la complétude des activités des autres agents

avant d'effectuer un traitement. Cette technique de programmation a été décrite dans le paragraphe décrivant les techniques de programmation avancées dans MAPS (partie A, chapitre 3).

1. FILIERE D'INTERPRÉTATION GÉOMÉTRIQUE

La filière d'interprétation géométrique implique deux agents, nommés KS-Region et KP-Geometrical.

1.1. OBJECTIFS

L'objectif de la filière d'interprétation géométrique est de fournir une modélisation des régions en terme de formes. Cette modélisation sera utile par la suite pour des besoins de manipulation et d'interprétation. La technique mise en œuvre est une mise en correspondance entre une primitive région et un modèle de forme, à l'aide de règles d'inférence.

1.2. DESCRIPTION DE LA FILIERE

Le rôle de l'agent KS-Region est simplement de maintenir et de transmettre les résultats de la filière de détection des régions. Il reste passif tant qu'il reçoit et enregistre des informations globales sur l'objet *Region-Image*, c'est à dire les attributs décrivant la référence du buffer image, la taille de l'image ou le nombre de composantes connexes détectées : aucune requête ni message n'est émis vers l'extérieur. Les règles de proposition sont déclenchées quand au contraire il reçoit des étiquettes de régions (instances de l'objet *Current-Region*), et contrôle leur transmission à l'étape suivante de l'interprétation géométrique (agent KP-Geometrical) : un comportement "actif" est démontré par l'agent dans un tel cas.

Les ressources de l'agent KP-Geometrical sont décrites en **Table 6**. Deux règles d'actions sont définies, dont le rôle est d'inférer une nouvelle instance de forme dans l'agent KS-Form, et de décider si son type correspond à l'objet *Convex* ou à l'objet *Non Convex*. Une telle décision est prise en comparant le rapport entre la surface de la région et celle de son enveloppe convexe à un seuil prédéterminé. Le processus d'inférence est limité à des régions suffisamment grandes.

Agent	**KP-Geometrical**
Rôle	Inférer l'Interprétation Géométrique d'une Région
Ressources	--> Règles d'Action : SI (1) Collecter les attributs de la région courante (instance de l'objet *Current-Region*) dans l'agent KS-Region (2) La surface de cette région est supérieure à un seuil minimum (3) Le degré de convexité est élevé ALORS Transmettre le résultat à l'agent KS-Form *cc : une instance de l'objet *Convex-Form* sera créée* SI (1) Collecter les attributs de la région courante (instance de l'objet *Current-Region*) dans l'agent KS-Region (2) La surface de cette région est supérieure à un seuil minimum (3) Le degré de convexité est faible ALORS Transmettre le résultat à l'agent KS-Form *cc : une instance de l'objet *Non-Convex-Form* sera créée*

Table 6 : Ressources de l'Agent KP-Geometrical

1.3. DISCUSSION

Les possibilités d'inférence pourraient être augmentées en considérant une plus grande variété de formes, et donc un ensemble de caractéristiques morphologiques plus vaste. Différentes approches de raisonnement pourraient également être considérées, incluant un raffinement progressif des formes détectées ou des schémas de prédiction / vérification. Dans ce cadre, une hiérarchisation des formes, des formes les plus générales (convexe, non convexe ...) aux formes les plus spécifiques (cercle, carré...) pourrait être introduite. Il serait ici également intéressant de maintenir plusieurs hypothèses en parallèle (mondes multiples).

2. FILIERE D'INTERPRÉTATION RELATIONNELLE

La filière d'interprétation relationnelle implique cinq agents, nommés KS-Contour, KP-Overlap, KS-Overlap, KS-Form et KP-Relational.

2.1. OBJECTIFS

L'objectif de la filière d'interprétation relationnelle est de donner une interprétation de haut niveau des relations entre un contour et une région. Cette filière est décomposée en deux étapes : dans une première étape, les relations entre un contour et des régions sont détectées, puis, dans une seconde étape, une de ces relations (la plus pertinente) est interprétée. Les techniques employées sont des techniques de mise en correspondance entre primitives à partir de règles d'inférence.

2.2. DESCRIPTION DE LA FILIERE

L'agent KS-Contour se comporte d'une manière similaire à l'agent KS-Region. Il reste passif quand il reçoit les informations globales portées par l'objet *Contour-Image*. Il devient actif chaque fois qu'une nouvelle étiquette de contour (objet *Current-Contour*) est reçue : une règle de proposition est alors déclenchée et contrôle l'envoi de ce contour à la prochaine étape de l'interprétation relationnelle.

Agent	**KP-Overlap**
Rôle	Détecter la Présence de Relations Spatiales entre un Contour et des Régions
Ressources	--> Règle d'Action : SI (1) Collecter les attributs du contour courant (instance de l'objet *Current-Contour*) dans l'agent KS-Contour (2) La surface de ce contour est suffisamment élevée (3) Collecter l'image des contours (instance de l'objet *Contour-Image*) dans l'agent KS-Contour (4) Collecter l'image des régions (instance de l'objet *Region-Image*) dans l'agent KS-Region (5) Calculer la liste des régions recouvrant le contour courant (6) Calculer la taille des recouvrements ALORS Transmettre le résultat à l'agent KS-Overlap *cc : une instance de l'objet *Overlap* sera créée, contenant la liste des régions associées au contour courant et la taille des recouvrements*

Table 7 : Ressources de l'Agent KP-Overlap

Les ressources de l'agent KP-Overlap sont décrites en **Table 7**. Son rôle est de calculer pour chaque contour la liste des régions qu'il recouvre ainsi que la taille des zones de recouvrement. Seuls les contours d'une taille suffisante sont pris en considération. Les résultats, impliquant tous les attributs nécessaires, sont finalement envoyés à l'agent KS-Overlap. Un tel agent réagit d'une manière très passive en enregistrant simplement toutes les informations venant de l'analyse relationnelle; il ne possède aucune règle de proposition ou d'interrogation.

Les ressources de l'agent KS-Form sont décrites en **Table 8**. Son rôle est d'envoyer les régions aux phases d'analyse de plus haut niveau, mais aussi de synchroniser leur envoi de façon à attendre la complétion de la tâche de détection des contours. L'analyse fondée sur les contours, comme nous l'avons indiqué, est en fait mise en œuvre pour fournir des informations de référence qui seront utilisées pour assister et améliorer le traitement des régions. La collecte de ces références doit donc être terminée avant de poursuivre l'analyse.

Agent	**KS-Form**
Rôle	Représenter et Gérer les Résultats de l'Etape d'Interprétation Géométrique
Ressources	--> Objets : *Form* : Etiquette de la région, Caractéristiques morphologiques, densitométriques & relationnelles (Régions voisines, Formes parentes & filles) & Variables d'état *Convex-Form* & *Non-Convex-Form* : objets héritant des attributs de l'objet *Form*
	--> Règles de Proposition : SI (1) L'agent est "actif" (2) Trouver une instance de l'objet *Convex-Form* qui n'a pas déjà été proposée à l'agent KP-Relational ALORS Proposer cette instance à l'agent KP-Relational SI (1) L'agent est "actif" (2) Toutes les instances de formes ont été proposées ALORS Proposer un "message de complétion" à l'agent KP-Relational

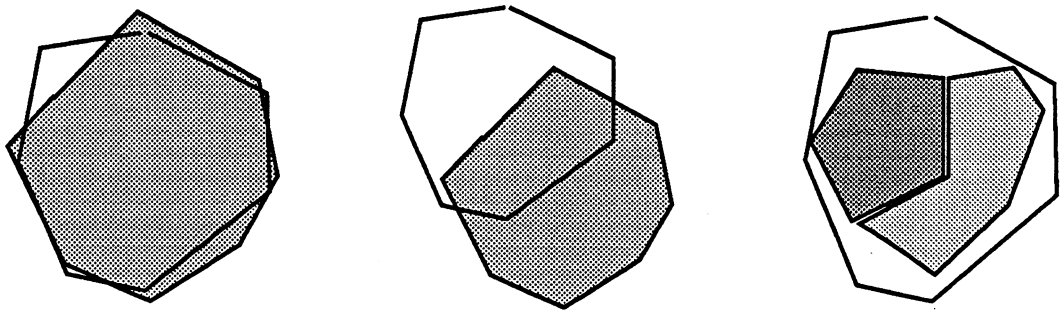
Table 8 : Vue partielle des ressources de l'Agent KS-Form

L'agent KS-Form est initialisé comme un agent passif : les informations concernant l'interprétation géométrique des régions sont simplement enregistrées, jusqu'à la réception d'un message particulier, émis par l'agent KP-Close, qui signale la complétion de la tâche de détection des contours (voir **Table 5**). En devenant alors actif, le rôle de l'agent KS-Form est de transmettre toutes les instances de formes (objets *Convex-Form* ou de *Non-Convex-Form*) à la prochaine étape de l'interprétation relationnelle. Un tel processus de transmission est contrôlé par des règles de proposition. Lorsque toutes les instances ont été transmises à l'extérieur, un message de complétion est envoyé à la prochaine étape d'analyse.

Agent	**KP-Relational**
Rôle	Interpréter une Relation de Recouvrement entre une Région et un Contour
Ressources	--> Règles d'Action : SI (1) Collecter les attributs de la forme courante (instance de l'objet *Convex-Form*) dans l'agent KS-Form (2) Trouver un recouvrement (instance de l'objet *Overlap*) dans l'agent KS-Overlap vérifiant : (a) La région support de l'instance de forme fait partie de la liste des régions du recouvrement (b) Il y a une étroite similarité entre la surface de la région, celle du contour impliqué dans le recouvrement et la surface du recouvrement ALORS Transmettre le résultat à l'agent KS-Relation *cc : une instance de l'objet *Match* sera créée*

Table 9 : Vue partielle des ressources de l'Agent KP-Relational

Le rôle de l'agent KP-Relational est d'interpréter un recouvrement entre une région (collectée dans l'agent KS-Form) et un contour (collecté dans l'agent KS-Overlap) comme définissant soit une correspondance, soit une inclusion (cas d'une région largement recouverte par un contour) soit une intersection (**Fig. B.7**). Plusieurs règles d'action sont définies dans ce sens, l'une d'elles étant fournie en **Table 9**.



Match *Intersection* *Inclusion*

Fig. B.7. Types de recouvrement

Les attributs de la forme courante sont collectés en prémisse. Une recherche est alors effectuée pour trouver et collecter dans l'agent KS-Overlap parmi les instances de recouvrement (objet *Overlap*), celles qui référençent la région support de l'instance de forme courante dans son attribut "Liste des Régions Recouvertes" (c'est à dire celles la recouvrant). Une comparaison est alors effectuée entre les surfaces de la région et du contour considérés et celle de leur recouvrement. Les résultats, impliquant tous les attributs nécessaires, sont finalement envoyés à l'agent KS-Relation. Une règle d'action particulière est de plus introduite, dont le rôle est de transmettre le message de complétion initié par l'agent KS-Form vers l'agent KS-Relation.

2.3. DISCUSSION

Les techniques courantes d'interprétation pourraient être raffinées en ajoutant des caractéristiques telles que le centre de gravité ou le rectangle d'encadrement d'une forme et aussi en qualifiant plus précisément le type du voisinage entre les éléments région et contour. La nécessité d'introduire un mécanisme de synchronisation a aussi été présentée : une telle étude devra être poursuivie et replacée dans le contexte de la programmation parallèle dans MAPS.

3. DISCUSSION

La phase d'analyse intermédiaire est chargée de la modélisation de haut niveau des primitives région et contour. Le problème essentiel de cette phase d'analyse est un problème de synchronisation . En début de phase, il est nécessaire que toutes les primitives région et contour aient été détectées. En milieu de phase, il est nécessaire d'attendre que tous les recouvrements et que toutes les formes aient été identifiés. Finalement, en fin de phase, il est nécessaire d'attendre que toutes les interprétations des recouvrements aient été effectuées avant de débuter la phase d'analyse suivante. Ces points de synchronisation sont présentés en figure **B.8**. Entre chacun de ces points de synchronisation, il est alors possible d'introduire du parallélisme.

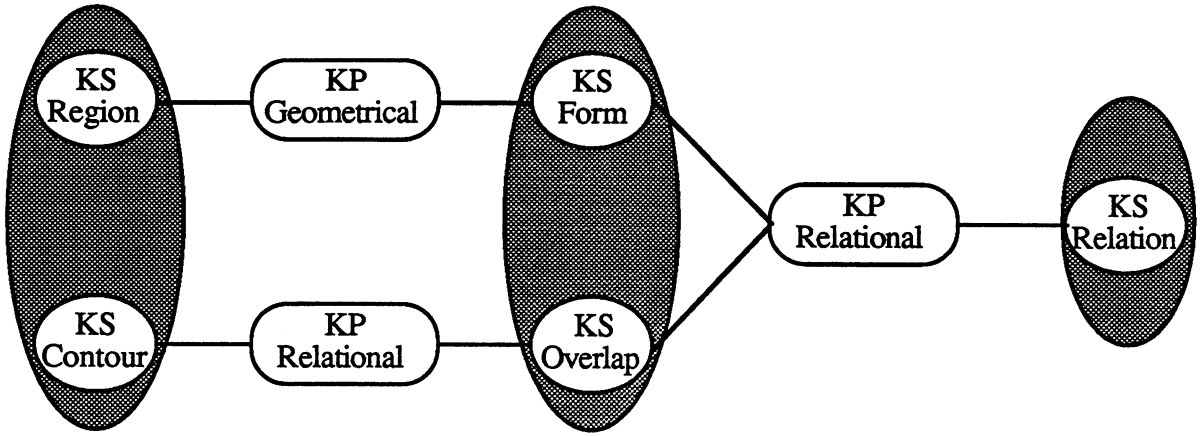


Fig B.8. Points de synchronisation.

Notons que les points de synchronisation correspondent à des niveaux figuratifs, dans lesquels plusieurs points de vue sont introduits (points de vue région et contour pour le niveau indice par exemple). De même, dans un niveau opératoire, plusieurs points de vue opératoires peuvent intervenir (KP Geometrical et KP Relational par exemple). Les problèmes de synchronisation se posent d'une part à l'intérieur d'un niveau entre deux points de vue et d'autre part entre deux niveaux et sont liés aux besoins de contrôle intra point de vue et inter point de vue (voir partie B, chapitre 2). Une structuration des points de vue à l'intérieur de chaque niveau à l'aide d'agents de haut niveau permettrait un meilleur contrôle et donc des mécanismes de synchronisation plus flexibles. Ces agents ou méta-agents constitueraient en effet un niveau d'observation, de contrôle et d'organisation des agents de traitement (point de vue opératoire) et des connaissances (point de vue figuratif).

Cette phase d'analyse étant de plus assez complexe et lourde, un mécanisme de focalisation sur certaines régions et certains contours est nécessaire. Actuellement, seuls les régions et les contours de taille suffisamment importante sont traités. Ce mécanisme pourrait être étendu (notamment avec l'introduction d'une représentation multi-échelle de l'image), et permettrait ainsi une analyse plus rapide (régions et contours évidents) suivie de schémas de prédiction / vérification sophistiqués dans les étapes suivantes.

IV Analyse Haut Niveau

L'objectif de la phase d'analyse de haut niveau est d'utiliser les descriptions de haut niveau fournies par la phase d'analyse intermédiaire pour interpréter sémantiquement les régions détectées et pour les manipuler afin de raffiner la segmentation initiale.

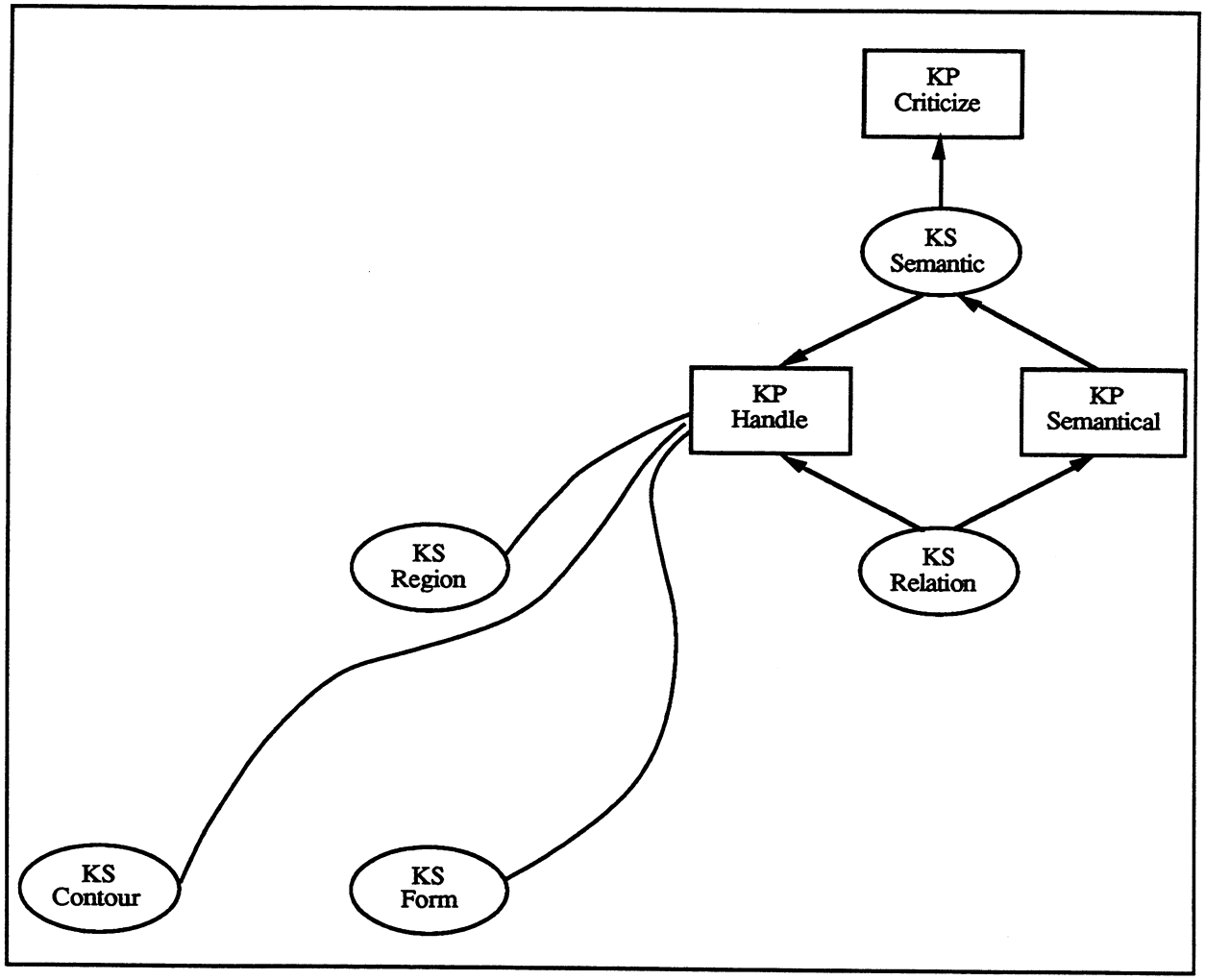


Fig. B.9. L'architecture KISS : agents intervenant dans la phase d'analyse haut niveau.

Description Globale

Une figure globale, décrivant l'approche de l'analyse de haut niveau est fournie en figure **B.9.** Une caractéristique essentielle, comme on peut le voir, est l'initiation de deux filières d'analyse indépendantes à partir de l'agent KS-Relation : une filière d'interprétation sémantique (KP-Semantical) et une filière de manipulation de région (KP-Handle). Ces deux filières sont placées en exclusion mutuelle : les régions en cours d'analyse sont en effet transmises à l'une ou l'autre de ces filières d'analyse, selon les résultats de l'analyse relationnelle. La manipulation des régions s'effectue sur des critères purement relationnels mais peut également s'effectuer sur des critères sémantiques, en considérant les régions support d'entités composites.

Agent	**KS-Relation**
Rôle	Représenter et Gérer les Résultats de l'Etape d'Interprétation Relationnelle
Ressources	--> Objets : *Relation* : Région (Etiquette, Surface & Voisins), Contour (Etiquette, Surface & Voisins) & Taille de la surface recouverte *Intersection*, *Inclusion* & *Match* héritant des attributs de l'objet *Relation*
	--> Règles de Proposition : SI (1) L'agent est "actif" (2) Trouver une instance de l'objet *Match* (région en correspondance avec un contour) qui n'a pas déjà été proposée à l'agent KP-Interpret ALORS Proposer cette instance à l'agent KP-Interpret SI (1) L'agent est "actif" (2) Trouver une instance de l'objet *Inclusion* (région incluse dans un contour) qui n'a pas déjà été proposée à l'agent KP-Handle ALORS Proposer cette instance à l'agent KP-Handle SI (1) L'agent est "actif" (2) Trouver une instance de l'objet *Intersection* (région traversée par un contour) qui n'a pas déjà été proposée à l'agent KP-Handle ALORS Proposer cette instance à l'agent KP-Handle

Table 10 : Ressources de l'Agent KS-Relation

Agent Initiateur

Le rôle de l'agent KS-Relation est de représenter et de gérer les résultats de l'interprétation relationnelle. Ses ressources sont décrites en **Table 10**. Cet agent est initialisé comme un agent passif; il devient actif à la réception d'un message de complétion transmis par l'agent KP-Relational : ce message provient de la retransmission du message de complétion émis par l'agent KS-Form (voir **Table 8**). Il garantit que l'interprétation relationnelle de toutes les

régions est terminée. Lorsqu'il est actif, les instances de l'objet *Relation* sont transmises soit vers l'étape d'interprétation sémantique soit vers l'étape de manipulation des régions. Une stratégie dédiée est développée à cette fin par les règles de proposition.

Les régions qui ont été mises en correspondance avec un contour (instances de l'objet *Match*) sont tout d'abord transmises vers l'étape d'interprétation sémantique. Nous postulons ici qu'une étroite correspondance entre un contour et une région est le signe d'une segmentation correcte : une interprétation sémantique peut être effectuée dans ce cas. Au contraire, les régions dont le support n'a pas pu être mis en correspondance (relations d'inclusion ou d'intersection) sont transmises vers l'étape de manipulation des régions. Les régions identifiées comme instances de l'objet *Inclusion* sont traitées en premier, pour éviter la duplication des efforts.

1. FILIERE D'INTERPRÉTATION SÉMANTIQUE

La filière d'interprétation sémantique implique trois agents, nommés KP-Semantical, KS-Semantic et KP-Criticize.

1.1. OBJECTIFS

L'étape d'interprétation sémantique est chargée de l'étiquetage sémantique des régions qui ont été détectées et analysées lors des niveaux d'analyse précédents. La technique mise en œuvre est une mise en correspondance entre des modèle de formes et de régions et des modèles des entités présentes dans l'image. Les agents de cette filière sont dépendants de l'application : les images analysées dans l'application actuelle (images de fibres musculaires) sont ainsi décrites tout d'abord, suivie d'une présentation des ressources des agents et de leur comportement.

Les images de fibres musculaires représentent des coupes transversales de fibres musculaires qui ont été colorées pour révéler une activité ATP-ase. Ces images sont acquises au moyen d'une caméra couplée à un microscope. Elles révèlent trois sortes de fibres et un espace intercellulaire plus clair, conduisant à un histogramme des niveaux de gris quadrimodal (**Fig. C.9(a)**). Nous avons décidé d'identifier ces sortes de fibres comme des fibres "noires", "grises" et "blanches" et l'espace intercellulaire comme du "fond". Les fibres peuvent être regroupées en agrégats : nous distinguons deux sortes d'agrégats, des agrégats homogènes qui regroupent des fibres de même type et des agrégats hétérogènes qui regroupent des fibres de type différent.

1.2. DESCRIPTION DE LA FILIERE

Le rôle de l'agent KP-Semantical est d'inférer une interprétation sémantique pour une instance de région donnée, identifiée d'un point de vue relationnel comme une région correspondant à un contour (instance de l'objet *Match*). Ses ressources sont décrites en **Table 11.**

Agent	**KP-Semantical**
Rôle	Interpréter Sémantiquement une Instance de *Correspondance*
Ressources	--> Règles d'Action : SI (1) Collecter les attributs de la région mise en correspondance (instance de l'objet *Match*) dans l'agent KS-Relation (2) Collecter les attributs de l'instance de forme correspondante dans l'agent KS-Form (3) Collecter les valeurs de référence des attributs des modèles d'objets (4) Calculer le degré de similarité entre les attributs de l'instance de forme et les valeurs de référence ALORS Transmettre le résultat à l'agent KS-Semantic *cc : une instance d'entité dépendante du degré de similarité sera créée*

Table 11 : Ressources de l'Agent KP-Semantical

Sept règles d'action ont été définies, qui concluent sur une identification unique pour chacune des régions (instances de l'objet *Match*) considérées. Un degré de similarité est calculé, fondé sur la comparaison entre les caractéristiques de l'instance et des intervalles de référence. Les attributs de l'instance impliquent le type de la forme (convexe ou non), la surface de la région, tout comme la moyenne et l'écart type des niveaux de gris : ces caractéristiques sont collectées dans l'agent KS-Form (voir **Table 8**). Les résultats, c'est à dire le type de l'entité et le coefficient de confiance (degré de similarité calculé) sont transmis à l'agent KS-Semantic.

Agent	**KS-Semantic**
Rôle	Représenter et Gérer les Résultats de l'Interprétation Sémantique
Ressources	--> Objets : *Entity* : Etiquette de la région, Confiance & Valeur symbolique, Caractéristiques relationnelles (Entités mères et filles) *Background*, *Heterogeneous*, *Homogeneous*, *Black-Fiber*, *Grey-Fiber*, *White-Fiber* héritant des attributs de l'objet *Entity*
	--> Règle de Propagation : SI (1) L'instance de l'objet *Homogeneous* a déjà été proposée à l'agent KP-Handle (2) Les instances filles d'*Homogeneous* sont des fibres du même type ALORS Augmenter le coefficient de confiance de l'instance d'*Homogeneous*
	--> Règles de Proposition : SI (1) L'instance courante d'*Homogeneous* est complète (2) L'instance courante d'*Homogeneous* n'a pas déjà été proposée à l'agent KP-Criticize ALORS Proposer l'instance courante d'*Homogeneous* à l'agent KP-Criticize SI (1) L'instance courante d'*Homogeneous* est complète (2) L'instance courante d'*Homogeneous* n'a pas déjà été proposée à l'agent KP-Handle ALORS Proposer l'instance courante d'*Homogeneous* à l'agent KP-Handle

Table 12 : Ressources de l'Agent KS-Semantic

Le rôle de l'agent KS-Semantic est de représenter et de gérer les résultats de la filière d'interprétation sémantique. Ses ressources sont décrites en **Table 12**. Plusieurs règles de proposition sont définies, dont le rôle est de transmettre chaque entité reçue à l'agent KP-Criticize. Les régions identifiées comme agrégats homogènes (instances de l'objet *Homogeneous*) sont ensuite transmises à l'agent KP-Handling pour subir une segmentation supplémentaire. De nouvelles entités filles sont obtenues comme résultat et identifiées. Les résultats de cette identification sont utilisés pour décider d'augmenter ou de diminuer le

coefficient de confiance attaché à l'hypothèse d'agrégat étudiée (activation des règles de propagation).

Le rôle de l'agent KP-Criticize est de transformer le coefficient de confiance d'une valeur réelle, comprise entre 0 et 1, en un argument symbolique (CORRECT, INCORRECT ou PEUT-ETRE). Le rôle d'une telle transformation, effectuée au cas par cas est d'ajuster l'évaluation symbolique finale en fonction de la difficulté estimée de la segmentation et de l'interprétation d'une entité. Le résultat est alors retransmis à l'agent KS-Semantic.

1.3. DISCUSSION

D'autres mécanismes d'interprétation pourraient être utilisés à ce niveau, tels que l'analyse par réseaux de neurones ou l'examen de graphes de dépendance. La prédiction / vérification pourrait également être employée. L'utilisation de critères topographiques aussi bien que morphologiques apparait de plus comme une nécessité pour interpréter des scènes complexes. Les résultats de l'interprétation pourraient également être utilisés pour diriger une nouvelle segmentation en se focalisant sur une partie restreinte de l'image initiale. Dans notre application de fibres musculaires, par exemple, trouver un agrégat hétérogène (instance de l'objet *Heterogeneous*) pourrait conduire à une réactivation de la filière d'analyse de bas niveau. Une nouvelle segmentation serait alors effectuée sur la partie de l'image initiale masquée par le support de l'instance. Des stratégies de prédiction / vérification complexes pourraient être effectuées, fondées sur l'étude du développement d'hypothèses d'interprétation en parallèle dans des mondes multiples.

2. FILIERE DE MANIPULATION

La filière de manipulation des régions se réduit à l'agent KP-Handle.

2.1. OBJECTIFS

L'objectif de la filière de manipulation est de raffiner la segmentation en utilisant des critères de haut niveau : critères relationnels et critères sémantiques. Cette filière réalimente la phase d'analyse intermédiaire avec de nouvelles régions issues des manipulations. Les techniques de manipulation retenues impliquent des décompositions de régions fondées sur une étude du graphe de la ligne médiane, des décompositions fondées sur des informations contour, et des fusions contraintes par des contours et un degré de similarité des niveaux de gris. Cette filière utilise ainsi plusieurs niveaux de représentation des connaissances et types de contrôle.

2.2. DESCRIPTION DE LA FILIERE

Le rôle de l'agent KP-Handling est d'améliorer les résultats de la segmentation, en fusionnant ou en décomposant les régions. Trois règles d'action sont définies à cet égard (**Table 13**).

Agent	**KP-Handle**
Rôle	Décomposer ou Fusionner le Support des Régions
Ressources	--> Règles d'Action : SI (1) Collecter les attributs de l'instance d'*Homogeneous* dans l'agent KS-Semantic (2) Collecter les attributs de l'instance de *Region-Image* dans l'agent KS-Region (3) Collecter les attributs de l'instance de *Match* dans l'agent KS-Relation (4) Collecter les attributs de l'instance de *Contour-Image* dans l'agent KS-Contour (5) Décomposer la région correspondante dans l'instance de *Region-Image* (6) Décomposer le contour correspondant dans l'instance de *Contour-Image* ALORS (1) Transmettre un "message de désactivation" à l'agent KS-Relation (2) Transmettre un "message de désactivation" à l'agent KS-Form (3) Transmettre les résultats de la décomposition à l'agent KS-Region (4) Boucler sur : Transmettre l'étiquette des composantes à l'agent KS-Region (5) Transmettre les attributs relationnels résultants à l'agent KS-Form (6) Transmettre les résultats de la décomposition à l'agent KS-Contour (7) Boucler sur : Transmettre l'étiquette des composantes à l'agent KS-Contour (8) Transmettre un "message d'activation" à l'agent KS-Form (9) Transmettre l'attribut "entités filles" de l'instance d'*Homogeneous* à l'agent KS-Semantic

Table 13 : Ressources de l'Agent KP-Handle

La première est chargée de la décomposition d'une instance de région identifiée comme agrégat homogène (instance de l'objet *Homogeneous*) pour obtenir ses composants. Du fait de la distribution de la connaissance, les valeurs de divers attributs nécessaires à cette décomposition sont tout d'abord collectées dans divers agents; la région aussi bien que le contour associé sont alors décomposés, selon l'étude de leur ligne médiane **[Montanvert 87]**. Les nouveaux constituants sont alors transmis comme décrit précédemment (voir **Table 3** pour la transmission des régions et **Table 6** pour la transmission des contours). Les attributs relationnels sont mis à jour dans l'agent KS-Form et KS-Semantic. Il est à noter que les agents KS-Form et KS-Relation sont désactivés, avant de traiter les primitives région et contour obtenues, puis réactivés : ceci pour s'assurer que la réception de nouvelles régions dans l'agent KS-Form sera terminée avant d'entamer l'analyse relationnelle, et que cette analyse est terminée avant d'entamer les étapes suivantes.

La seconde règle met en œuvre un processus de fusion dont le rôle est d'agréger une région incluse dans un contour (instance de l'objet *Inclusion*) d'un point de vue relationnel, avec ses voisins les plus semblables : les seuls voisins sélectionnés sont ceux qui apparaissent comme recouverts par le même contour. La troisième règle met finalement en œuvre un processus de décomposition, dont le rôle est de découper une région traversée par un contour (instance de l'objet *Intersection*). Les recouvrements Région / Contour de même que l'information relationnelle sont remis à jour dans les deux cas, pour préserver la consistance de la base de connaissance.

2.3. DISCUSSION

La notion importante introduite dans cette filière de manipulation est la notion de fusion de données. L'agent KP-Handling va en effet puiser des informations dans plusieurs agents KS : agent KS-Form, agent KS-Region, agent KS-Contour et agent KS-Relation. Ces données représentent des niveaux de représentation différents de l'image et sont structurés par leur distribution au sein d'agents KS. Une procédure de manipulation est ensuite introduite et un résultat produit, qui est alors réinjecté dans le système.

Une autre notion importante est la notion de contrôle des processus de manipulation. Deux critères de contrôle sont invoqués à ce niveau : un critère de convexité et un critère relationnel fondé sur les relations entre un contour et une région. Le processus de manipulation ne sera invoqué que pour des entités non convexes (provenant de l'agent KS-Semantic) et pour des régions intersectées par un contour ou incluses dans un contour (provenant de l'agent KS-Relation).

3. DISCUSSION

La phase d'analyse de haut niveau est responsable de l'arrêt du système : deux critères d'arrêt sont introduits, arrêt des décompositions quand il n'y a plus d'intersection et plus d'inclusion et arrêt des décompositions quand il n'y a plus d'entités non convexes.

Cette phase est un lieu de contrôle important et démontre la philosophie générale du système KISS : KISS est un système qui introduit une représentation multi-niveaux des connaissances sur une image. C'est également un système de type coopération région / contour qui fait intervenir des informations de type géométrique pour contrôler le processus de segmentation. Le système fonctionne tant que des manipulations sont nécessaires (présence d'inclusions, d'intersections ou d'entités composites), il converge donc vers l'obtention de correspondances entre des régions et des contours convexes.

Cette phase d'analyse est en outre en partie dépendante de l'application. Changer d'application nécessitera une modification des agents de ce niveau et / ou l'ajout de nouveaux agents. Une distribution optimale doit être alors trouvée afin de réduire le nombre d'agents à changer ou à modifier pour une chaque nouvelle application.

Des schémas de prédiction / vérification complexes doivent également être introduits à ce niveau. Ces schémas vont avoir une incidence sur le contrôle inter point de vue (partie B, chapitre 2). Dans ce cadre, l'introduction d'agents de contrôle ou d'observation parait indispensable.

V RÉSULTATS ET DISCUSSION

Des expérimentations ont été effectuées sur des images de fibres musculaires présentes dans la banque de données du PRC Greco "Traitement du Signal et Images" **(Fig. B.6(a))**; de telles images présentes trois types de fibres appelées fibres "noires", fibres "blanches" et fibres "grises" et un espace intercellulaire appelé "fond". Les fibres noires tout comme le fond apparaissent comme des constituants homogènes et bien contrastés. Les fibres grises et blanches, au contraire, présentent une texture hétérogène dont les primitives peuvent présenter des nuances de gris variant du gris clair au gris foncé. Les fibres peuvent être regroupées en agrégats. Les agrégats homogènes regroupent des fibres de même type, alors que les agrégats hétérogènes regroupent des fibres de type différent.

En ce qui concerne les résultats de l'analyse d'image, les fibres noires et le fond sont bien segmentés en général, aussi bien d'un point de vue région que d'un point de vue contour, même si des agrégats sont parfois obtenus : une étroite correspondance entre les deux types d'information est toujours obtenue. Les fibres grises et blanches conduisent à une segmentation

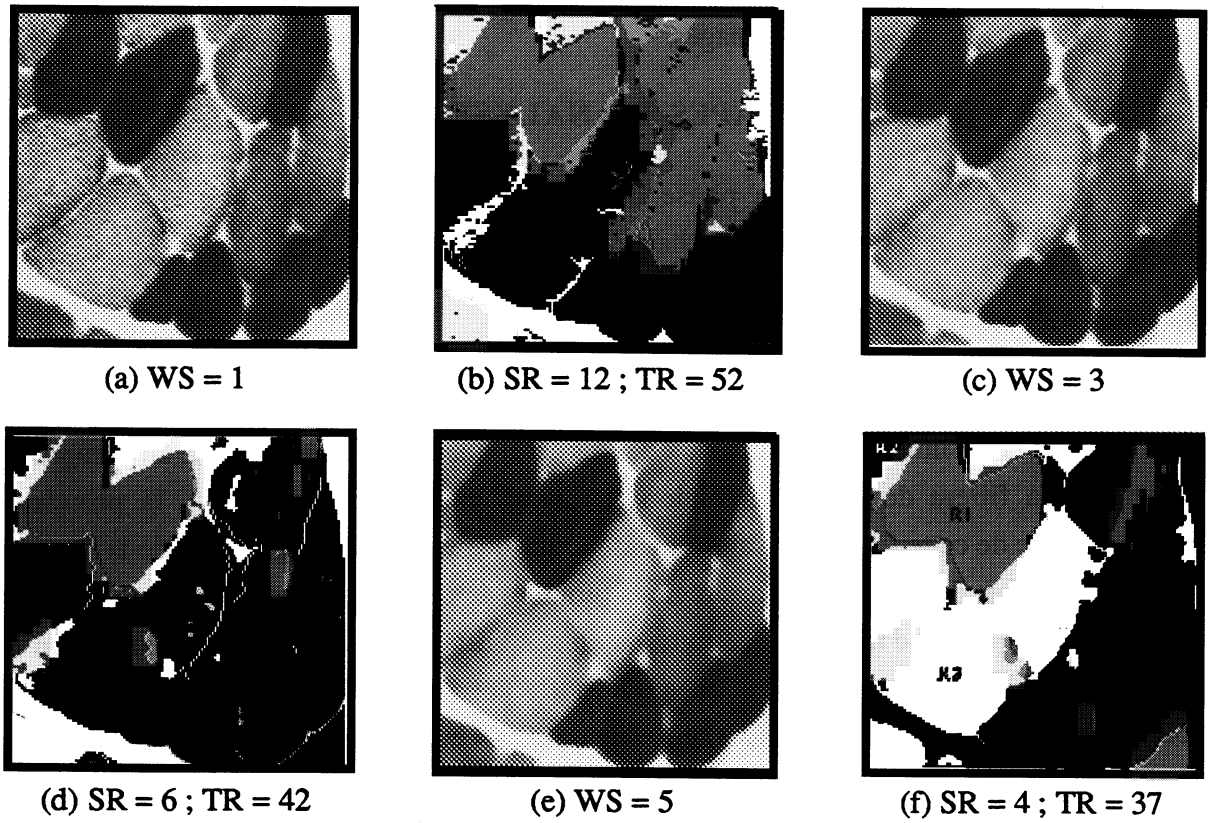
incorrecte, due à la fois à la détection d'éléments de contour traversant les régions et à un surpartitionnement des régions correspondant aux fibres. Nous illustrons tout d'abord dans un premier paragraphe l'analyse de telles images étape par étape, puis présentons et discutons les résultats du système sur plusieurs images.

1. ILLUSTRATION DES ÉTAPES D'ANALYSE

Nous illustrons tout d'abord la phase d'analyse bas niveau, puis la phase d'analyse intermédiaire et enfin la phase d'analyse haut niveau.

1.1. ANALYSE BAS NIVEAU.

L'image initiale des niveaux de gris est présentée en figure **B.10(a)**. La taille de l'image est de 128x128 points et l'intervalle des niveaux de gris est [0..256]. Comme expliqué au paragraphe présentant la filière de détection région (paragraphe II.2 du chapitre 1), la principale caractéristique de cette filière est l'ajustement itératif du degré de filtrage (taille de la fenêtre), selon les résultats de la détection des régions (nombre de petites régions). La figure **B.10(a-f)** montre une séquence d'images illustrant les étapes initiale, intermédiaire et finale d'un tel ajustement, utilisant une taille de fenêtre (WS) de 1, 3 et 5 respectivement. Comme on peut le voir, le nombre de petites régions (SR) diminue de 12 à 4 et le nombre total de régions (TR) de 52 à 37. Plusieurs erreurs de segmentation peuvent être observées, impliquant des surpartitionnements et des sursegmentations de régions. Des agrégats ont également été détectés.

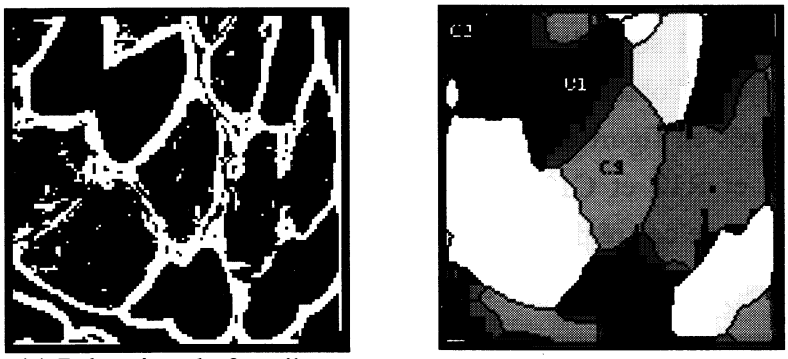


(a) WS = 1 (b) SR = 12 ; TR = 52 (c) WS = 3

(d) SR = 6 ; TR = 42 (e) WS = 5 (f) SR = 4 ; TR = 37

Fig. B.10. Illustration de l'étape de détection des régions : (a) image initiale, (b) segmentation de l'image initiale, (c) application d'un filtrage de taille intermédiaire, (d) segmentation résultante, (e) application d'un filtrage de grande taille, (f) segmentation finale.

La détection des frontières puis des contours fermés est illustrée en figure **B.11(a-b)**. Un grand nombre d'éléments de frontière ont été supprimés en figure **B.11(b)**, et des contours fermés obtenus. Comme déjà mentionné, les traitements fondés sur les contours ont été sélectionnés pour fournir des résultats de détection robustes, pouvant servir de référence pour guider les traitements fondés sur les régions. En comparant les résultats présentés en figure **B.10(f)** et en figure **B.11(b)**, on peut voir qu'une correspondance est obtenue dans le cas des fibres noires et du fond. Un grand nombre d'erreurs de segmentation est observé pour les autres régions, alors que les résultats de l'analyse contour sont globalement satisfaisants.



(a) Détection de frontière fondée sur PREWITT

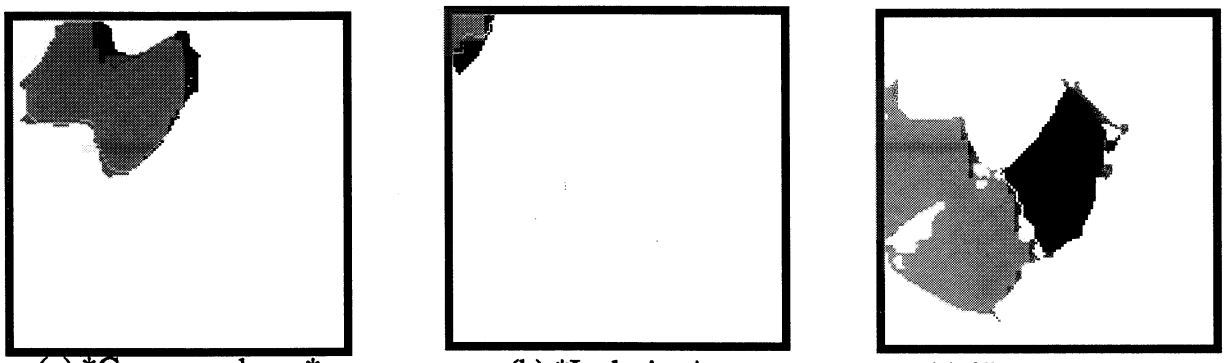
(b) Détection de 34 contours fermés

Fig. B.11. Illustration des étapes de détection de contours: (a) image des points de frontière, (b) image contour.

Trois constituants de l'image sont soulignés en figure **B.10(f)** (régions R1, R2 et R3) et en figure **B.11(b)** (contours C1, C2 et C3) qui sont utilisés pour illustrer la description des étapes de l'analyse de niveau intermédiaire et de haut niveau.

1.2. Analyse Intermédiaire

D'un point de vue géométrique, R1 et R3 ont été identifiées comme des régions non convexes (instances de l'objet *Non-Convex-Form*) alors que R2 a été identifié comme une région convexe (instance de l'objet *Convex-Form*). Les résultats de l'analyse relationnelle sont illustrés en figure **B.12(a-c)**. La région R1 a été mise en correspondance avec le contour C1; la région R2 a été analysée comme incluse dans le contour C2 et la région R3 comme traversée par le contour C3.



(a) *Correspondance*

(b) *Inclusion*

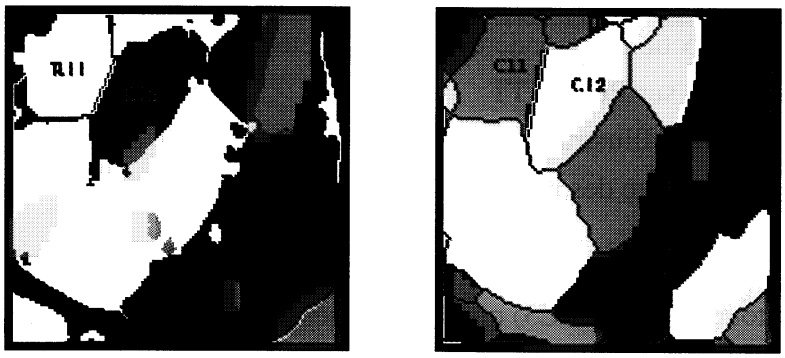
(c) *Intersection*

Fig. B.12. Relations détectées entre(a) R1 and C1, (b) R2 and C2, (c) R3 and C3.

1.3. Analyse Haut Niveau

A cause de la correspondance détectée entre la région R1 et le contour C1, R1 est dirigée vers l'interprétation sémantique. Cette région est alors interprétée comme étant un agrégat homogène

(instance de l'objet *Homogeneous*) et donc transmise à l'agent KP-Handle pour une décomposition. R1 et C1 sont alors décomposés suivant une procédure fondée sur la ligne médiane, donnant lieu à l'obtention des régions R11 et R12 et des contours C11 et C12 : les résultats sont montrés en figure **B.13(a-b)**. Une étroite correspondance peut être observée entre R11 et C11 et R12 et C12. R11 et R12 sont donc dirigées vers l'interprétation sémantique : chacune de ces régions est alors identifiée comme étant une fibre noire, ce qui confirme l'interprétation d'agrégat homogène pour R1.

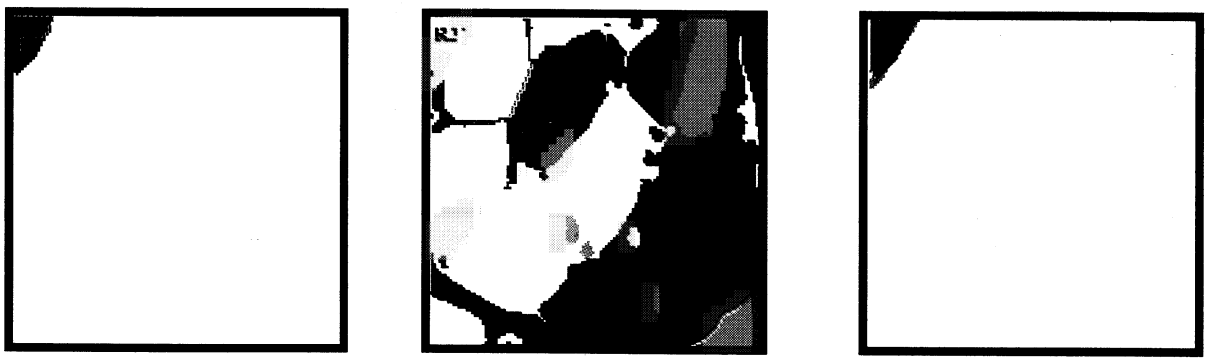


(a) Décomposition de R1 en R11 et R12

(b) Décomposition de C1 en C11 et C12

Fig. B.13. Décomposition fondée sur la ligne médiane de (a) R1, (b) C1.

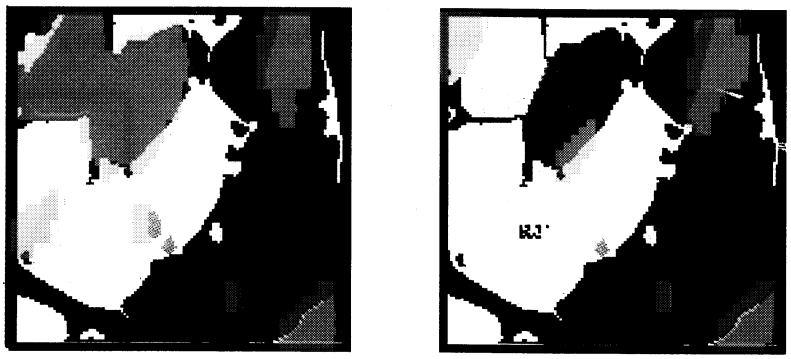
Du fait de l'inclusion détectée pour la région R2, celle-ci est dirigée vers l'agent KP-Handle. Une procédure de fusion est alors activée, qui collecte et fusionne les régions voisines, qui étant incluses dans C2 apparaissent comme les plus similaires à R2. De telles régions sont indiquées en figure **B.14(a)**. Le processus de fusion conduit à une nouvelle région R2' montrée en figure **B.14(b)**. R2' est alors identifiée comme un objet convexe (instance de l'objet *Convex-Form*) et une étroite correspondance est détectée entre R2' et C2 (**Fig. B.14(c)**). A cause de cela, R2' est dirigée vers l'interprétation sémantique qui l'identifie comme étant une fibre grise.



(a) (b) (c) *Correspondance*

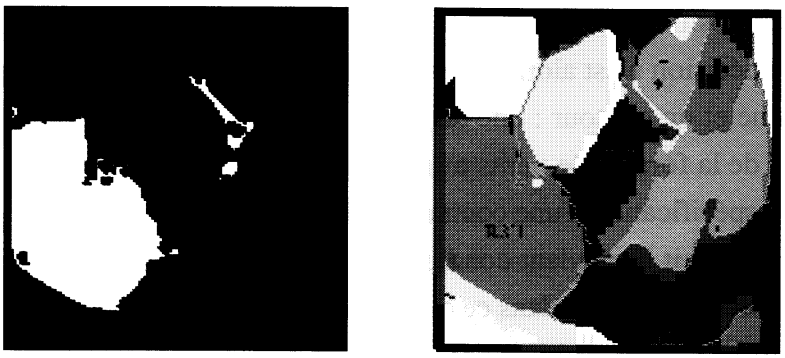
Fig. B.14. Illustration du processus de fusion : (a) une région incluse dans un grand contour,
(b) résultat de la fusion, (c) interprétation relationnelle correspondante.

En ce qui concerne la région R3, elle présente plusieurs trous, qui, considérés comme des régions à manipuler seront fusionnés avec R3, donnant une nouvelle région R3', comme illustré en figure **B.15(a-b)**. R3' est alors identifiée comme une région non convexe (instance de l'objet *Non-Convex-Form*; une intersection est détectée entre R3' et C3. A cause de cette intersection **(Fig. B.16(a))**, R3' est dirigée vers l'agent KP-Handle et décomposée en suivant le contour intersectant C3, comme illustré en figure **B.16(b)**. Deux nouvelles régions sont obtenues, nommées R3'1 et R3'2, qui sont à leur tour interprétées d'un point de vue géométrique, relationnel et sémantique.



(a) (b)

Fig. B.15. Fusion de régions :
(a) une région affichant plusieurs trous entourée par un grand contour,
(b) résultat de la fusion.



(a) (b)

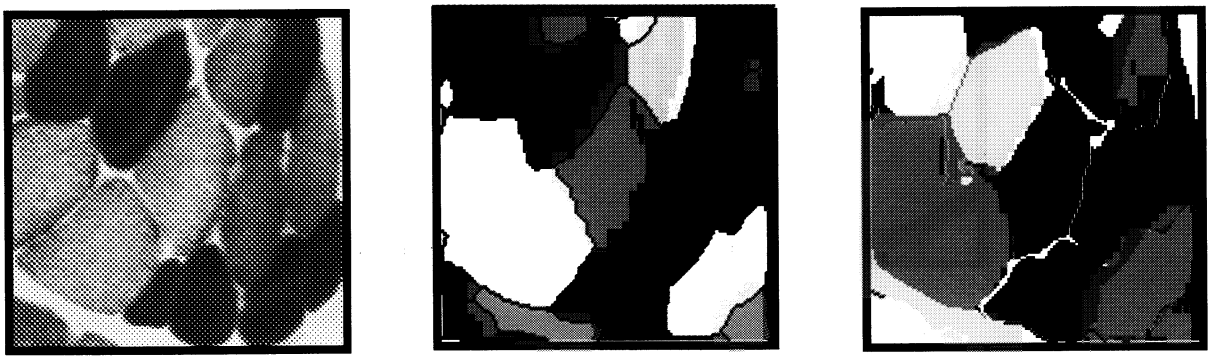
Fig. B.16. Décomposition fondée sur un contour:
(a) région initiale et contour associé, (b) résultat.

2. RÉSULTATS ET DISCUSSION

Nous présentons tout d'abord et discutons les résultats du système sur un ensemble d'images, puis présentons et discutons les résultats du système où l'algorithme de segmentation de Fisher a été remplacé par un algorithme plus performant sur les mêmes images.

2.1. SYSTEME UTILISANT L'ALGORITHME DE FISHER

Les résultats finaux sont résumés en figure **B.17(a-c)**. L'image initiale est présentée en figure **B.17(a)**, les images finales de la segmentation région et contour respectivement en figure **B.17(b)** et en figure **B.17(c)**. Des améliorations de la segmentation sont observées, résultant de la décomposition des grands éléments et de la fusion des petits. La décomposition des grandes régions est due à la fois à des décisions sémantiques et à des décisions relationnelles. Ceci illustre la puissance potentielle de l'approche coopérative, considérée comme coopération entre modes d'analyse (analyse de type région ou de type contour) ou comme coopération entre niveaux d'analyse (de la détection bas niveau à l'interprétation haut niveau).



(a) (b) (c)

Fig. B.17. Illustration des résultats finaux :
(a) image initiale, (b) contours finaux, (c) regions finales.

Des échecs de décomposition sont observés dans le cas de régions hétérogènes correspondant à un contour (aucune décision n'est alors prise) ou quand une correspondance est observée entre un agrégat homogène et un contour : une décomposition est lors effectuée mais peut échouer selon la complexité de la forme. De petits trous subsistent, dus aux échecs des fusions : de tels échecs sont dus à la spécification d'une contrainte de similarité des niveaux de gris trop élevée. Relâcher le critère pourrait cependant conduire à des fusions abbusives : des descripteurs de forme globaux devront être utilisés dans ce cas. L'information contour peut aussi échouer pour la décision d'une fusion. Un problème final est du aux régions artefacts, débordant sur plusieurs régions, et apparaissant comme des parties de régions sous-segmentées, ou des parties de fond sur-segmentées. Une étape d'examen devra être effectuée, pour prendre des décisions consistantes dans ce cas. Nous pouvons tirer la même conclusion en ce qui concerne les excroissances qui peuvent perturber les formes (dues à une sur-segmentation de la forme ou à des erreurs locales).

La détection et la fermeture des contours tout comme la détection initiale des régions pourrait être améliorée, en ce qui concerne les outils de bas-niveau et les stratégies de raisonnement. La figure **B.18** présente ainsi l'influence de la détection initiale des régions sur les résultats

obtenus. Des tests ont été effectués sur d'autres images pour lesquelles nous pouvons remarquer que la détection des contours est globalement satisfaisante et permet d'obtenir des approximations robustes (**Fig. B.18(b)** et **18(f)**). La segmentation région fournit quant à elle des résultats variables selon l'image initiale (**Fig. B.18(c)** et **18(g)**). Globalement, cette segmentation région initiale est assez mauvaise. Dans un premier cas (**Fig. B.18(d)**), nous pouvons remarquer l'effet attracteur des contours, qui a permis de gommer un certain nombre d'erreurs comme dans le cas de la figure **B.17**. L'effet attracteur des contours ne fonctionne que si les informations région sont cohérentes, c'est-à-dire présentent une similarité dans un même contexte (auquel cas des régions seront par exemple fusionnées) et dissimilaires en dehors de ce contexte. On s'aperçoit que pour le second cas (**Fig. B.18(h)**) des régions incohérentes conduisent à un résultat final proche de la segmentation initiale.

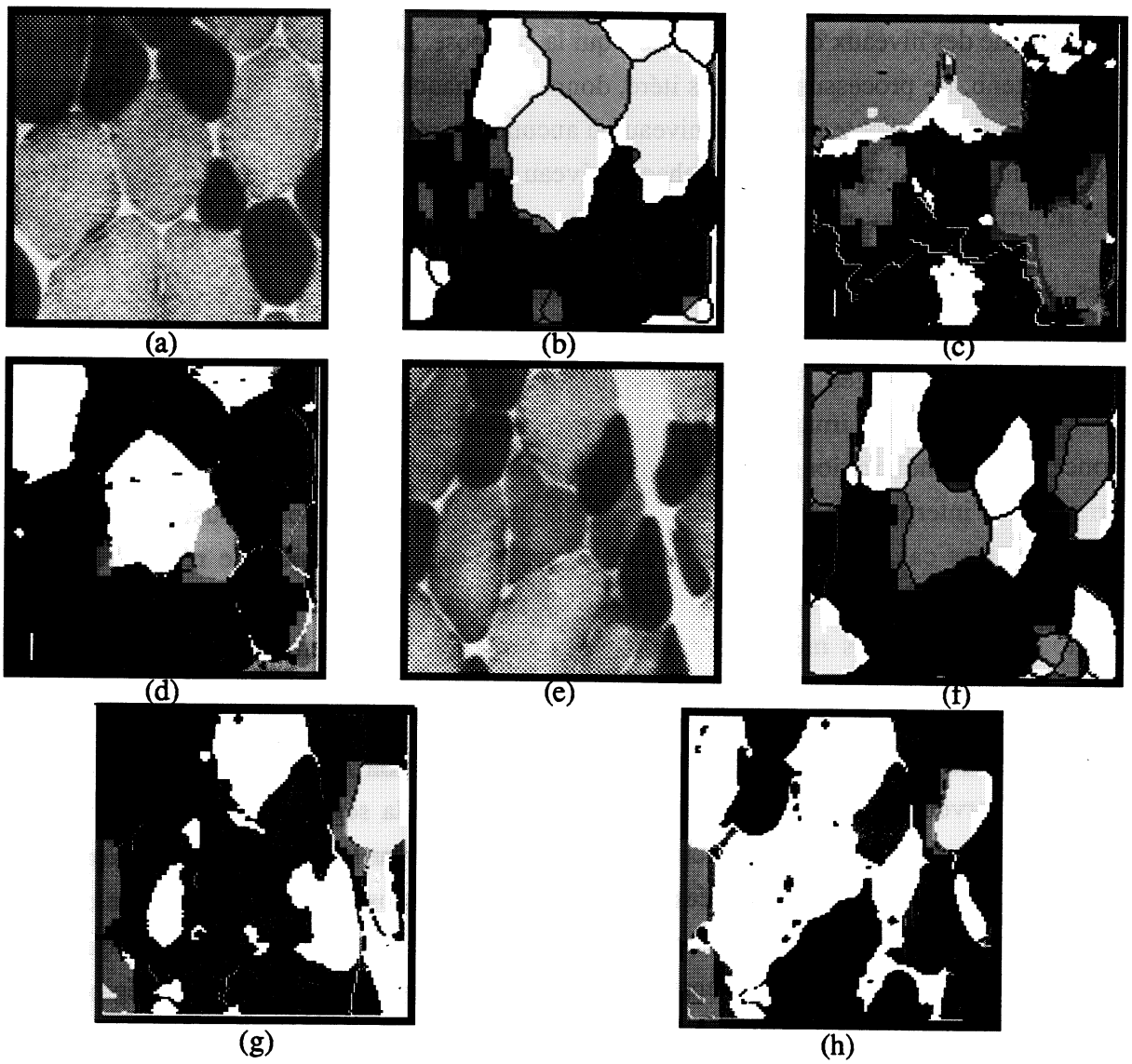


Fig. B.18. Résultats de la segmentation des images initiales (a) (e). Les images (b) (f) représentent les contours, les images (c) (g) les régions initiales et les images (d) (h) les régions finales.

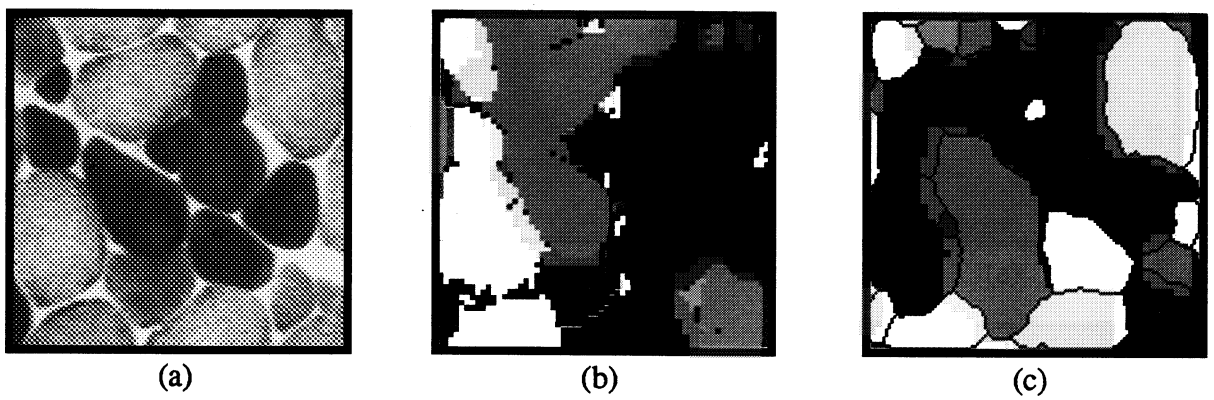
2.2. SEGMENTATION UTILISANT UN ALGORITHME PYRAMIDAL

Le système KISS est donc dépendant des segmentations initiales en région et en contour, et ne peut pas améliorer de façon spectaculaire des images segmentées trop bruitées. Cette conclusion est confirmée par les résultats obtenus en changeant l'opérateur de détection des régions. Les figures **Fig. B.19, Fig. B.20, Fig. B.21** et **Fig. B.22** présentent ainsi les résultats du système KISS où la segmentation selon l'algorithme de Fisher a été remplacée par une segmentation utilisant un algorithme fondé sur une représentation pyramidale **[Montanvert 92]**. Cet algorithme considère chaque pixel de l'image initial, qui constituent le premier niveau de la pyramide et bâtit un graphe des voisinages. Une recherche de cliques maximale est alors effectuée et les pixels voisins les plus similaires dans ces cliques sont fusionnés, fournissant ainsi des régions auxquelles est affectée comme valeur de niveau de gris, la moyenne des niveaux de gris des pixels qui la compose. Le second niveau de la pyramide est ainsi obtenu. Le processus est alors itéré, donnant à chaque itération un nouveau niveau de la pyramide, jusqu'à l'obtention d'un niveau où aucune des régions n'est fusionnable. Les critères de décision de fusion de régions à chaque niveau sont sophistiqués et peuvent faire intervenir des informations contour comme le gradient.

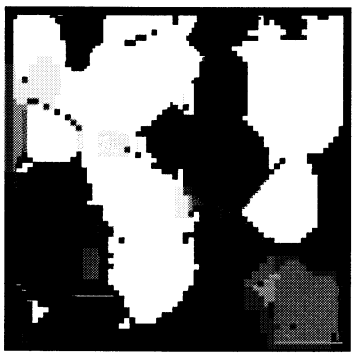
Les segmentations initiales en régions sont globalement plus satisfaisantes que les segmentations obtenues avec l'algorithme de Fisher **(Fig. B.19(a), 20(a), 21(a)** et **22(a))**, et les images finales correctes **(Fig. B.19(d), 20(d), 21(d)** et **22(d))**. Nous pouvons tirer de ces images finales les mêmes conclusions que précédemment en ce qui concerne les améliorations, mais également les remarques suivantes. Dans certains cas, des erreurs de l'interprétation géométrique entrainent des décompositions abusives **(Fig. B.20)**. Dans d'autres cas, l'information contour fausse les résultats (fusion de régions distinctes comme dans la figure **Fig. B.19**). Dans d'autres cas enfin, des erreurs de segmentation contour entrainent les mêmes erreurs de segmentation région en induisant de mauvaises décompositions de régions **(Fig. B.21)**.

Ces dernières erreurs proviennent d'une coopération région / contour à sens unique. Les contours servent en effet de support en vue d'améliorer la segmentation région mais, ces informations contour ne sont en revanche pas améliorées par la segmentation région, ce qui permettrait d'éviter les erreurs citées. A vrai dire, il est impossible pour le système de savoir quelle information (région ou contour) est la plus robuste. Une solution serait d'implanter un système où les deux types de traitement s'effectueraient en parallèle (coopération région fondée sur les contours et coopération contour fondée sur les régions). Ainsi, aucune information ne serait privilégiée à priori. Un contrôle global synchroniserait les deux filières et observerait les traitements et déductions effectués pour prendre des décisions finales. Cette solution nécessite

toutefois une version parallèle de MAPS et l'introduction d'agents superviseur de type méta-agents.

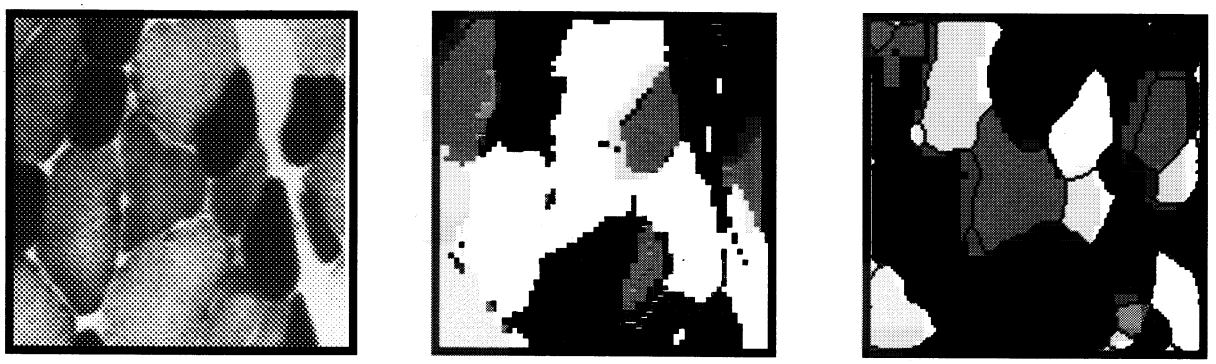


(a) (b) (c)

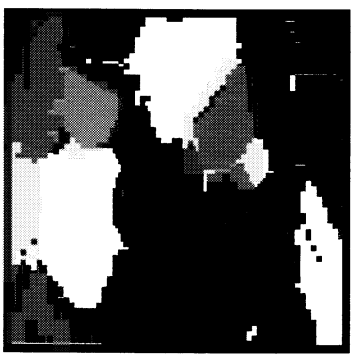


(d)

Fig. B.19. Résultats de la segmentation de l'image initiale (a).
(b) Image des régions initiales.
(c) Image des contours initiaux.
(d) Segmentation finale.

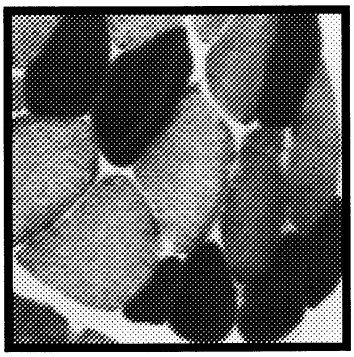


(a) (b) (c)



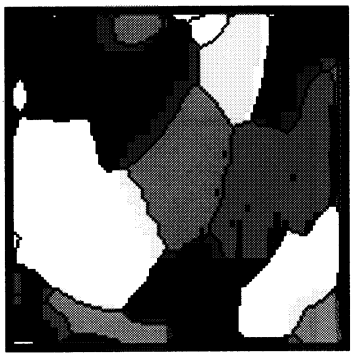
(d)

Fig. B.20. Résultats de la segmentation de l'image initiale (a).
(b) Image des régions initiales.
(c) Image des contours initiaux.
(d) Segmentation finale.

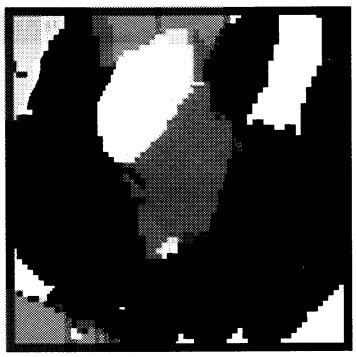


(a)

(b)

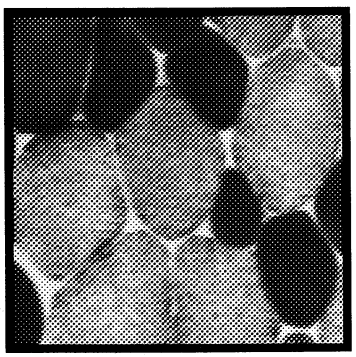


(c)

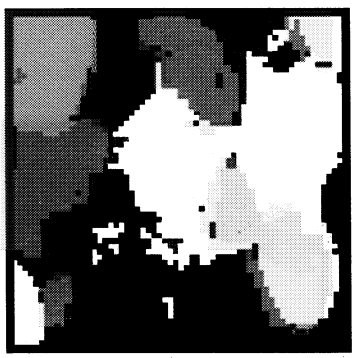


(d)

Fig. B.21. Résultats de la segmentation de l'image initiale (a).
(b) Image des régions initiales.
(c) Image des contours initiaux.
(d) Segmentation finale.

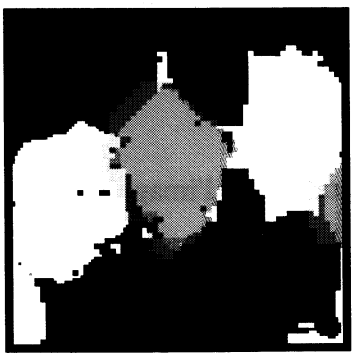


(a)



(b)

(c)



(d)

Fig. B.22. Résultats de la segmentation de l'image initiale (a).
(b) Image des régions initiales.
(c) Image des contours initiaux.
(d) Segmentation finale.

3. IMPLANTATION

KISS est défini comme une application construite avec l'environnement de programmation MAPS et décrit par un ensemble de fichiers de ressource. La figure **B.23** présente l'interface de l'environnement et celle de l'application KISS.

Un fichier de ressource par agent a été introduit, chacun d'eux écrit selon la syntaxe du langage de programmation MAPS. KISS met en œuvre 18 agents principaux, parmi lesquels le tiers sont des agents KS ; 7 agents KP-Descriptif sont en outre introduits et attachés à chaque agent KS. Le système comprend à présent 20 objets et 100 règles. Pendant l'exécution, environ 200 instances sont dynamiquement manipulées pour chaque image, qui correspondent approximativement à 35 régions et autant de contours, qui donnent lieu à autant d'instances dans les agents KS-Form, KS-Overlap et KS-Relation. Environ 15 instances de régions passent finalement par l'étape d'interprétation sémantique tandis que les 20 autres sont dirigées vers l'agent KP-Handle.

KISS est écrit d'une manière très modulaire, et peut ainsi être aisément mis à jour et étendu. Comme application de MAPS, il peut être exécuté sur n'importe quelle station UNIX supportant MAPS. Les expériences ont été effectuées sur station HP-APOLLO DN3550, configurées avec 8Mo de RAM. Le temps d'exécution CPU est de 10-15 mn : il est également distribué entre le processus MAPS et le processus de manipulation d'image. Le processus de manipulation d'image pourrait être exécuté sur une machine dédiée utilisant la puissance des processeurs RISC ou des Transputers, pour augmenter les performances du système.

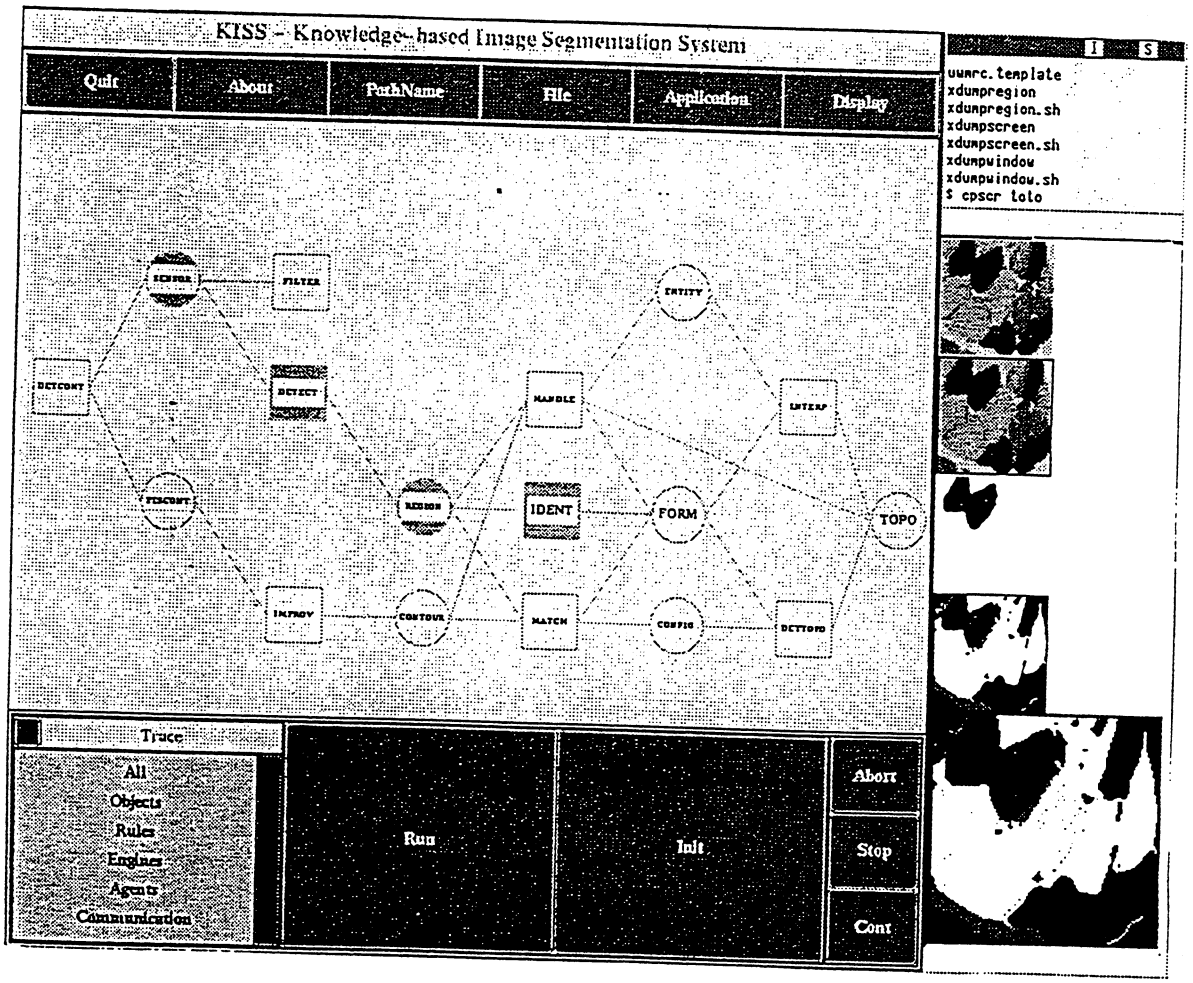


Fig. B.23. Interface de l'environnement de programmation MAPS et de l'application KISS

VI CONCLUSION

Dans cette partie, nous avons présenté le système multi-agents de vision par ordinateur (KISS), conçu à partir de l'environnement de programmation générique (MAPS). KISS a été défini pour combiner diverses filières d'analyse et d'interprétation d'une manière flexible, et illustre le potentiel d'une approche coopérative : coopération entre des primitives différentes à un niveau d'abstraction donné, ou coopération entre des informations issues du bas niveau ou du haut niveau d'analyse.

KISS, bien qu'utilisant des outils parfois rudimentaires et n'implantant pas toutes les techniques avancées des systèmes de vision, a ainsi démontré la faisabilité d'un système de vision fondé sur le paradigme multi-agents, en répondant en partie aux besoins dégagés dans l'état de l'art. La distinction entre connaissances figuratives et connaissances opératoires supporte en effet les deux axes orthogonaux de conception des systèmes de vision : un axe centré-figuratif décrivant les différents niveaux de représentation impliqués en vision et un axe centré-opératoire décrivant de manière duale les différents niveaux de traitements impliqués. L'environnement de programmation MAPS fondé sur cette distinction figuratif / opératoire supporte ainsi parfaitement cette représentation duale : les niveaux de représentation ont été distribués au sein d'agents KS et les niveaux de traitement distribués au sein d'agents KP.

Le besoin d'un contrôle adapté en termes de focalisation et d'adaptation a également été abordé. MAPS permet d'introduire des mécanismes de contrôle sophistiqués tant au niveau local qu'au niveau global, qui ont été utilisés pour concevoir KISS. Un exemple de contrôle local est illustré par le mécanisme d'ajustement du paramètre du filtre médian, et un exemple de contrôle global est illustré par les mécanismes de synchronisation entre les différentes filières de traitement et par les mécanismes de planification entre les traitements. Cette souplesse au niveau du contrôle est en grande partie due à sa distribution au sein des agents.

Plusieurs problèmes demeurent cependant en suspens, ces problèmes étant dus pour une partie à l'environnement MAPS et pour une autre partie au système KISS. En ce qui concerne les problème liés à l'environnement MAPS, ils viennent d'une part de la séquentialité de cet environnement. Nous avons en effet vu que le parallélisme permettrait d'améliorer le système KISS pour de nombreux aspects : pour la manipulation d'hypothèses en parallèle ou l'exécution de plusieurs algorithmes ou filières de détection. Ils viennent également du manque d'abstraction du langage et du manque d'agents de granularité différente. Nous avons en effet vu le besoin d'agents de faible granularité pour les représentations multi-échelles et le besoin d'agents superviseur ou observateur (méta-agents) pour résoudre les problèmes de synchronisation qui ne manqueront pas de se poser avec une version parallèle.

En ce qui concerne les problèmes propres au système KISS, notons le manque d'un niveau de contrôle plus global, le manque de stratégies de reprise et le fonctionnement globalement dirigé par les données du système. L'application considérée (détection de fibres musculaires) est en outre essentiellement un problème de bas niveau. Il semble donc nécessaire d'aborder d'autres applications, présentant un caractère plus sémantique, et posant des problèmes d'interprétation. Ces nouvelles applications permettraient ainsi d'améliorer les phases d'analyse de haut niveau du système KISS.

CONCLUSION PARTIE B

Dans cette partie, nous avons voulu montrer l'applicabilité de l'environnement MAPS. L'application choisie a montré sa spécificité en posant plusieurs problèmes particulièrement intéressants tels que la modélisation des niveaux de représentation et des niveaux de traitements impliqués par la vision, la visualisation d'une information selon plusieurs points de vue, ou la synchronisation entre deux filières d'analyse. Deux notions importantes se dégagent ainsi, la notion de niveaux et la notion de coopération, essentielles en vision par ordinateur.

L'introduction de niveaux que ce soit en segmentation avec les approches de type multi-résolution ou dans les systèmes de vision avec l'introduction de niveaux d'abstraction dans les représentations et les traitements a pour but d'améliorer le contrôle en permettant de localiser les traitements, par des mécanismes de focalisation et d'adaptation. Un contrôle adapté va en effet permettre la coopération de plusieurs techniques, manipulant et intégrant des informations à différents niveaux et selon des points de vue différents.

Notons ici l'importance des représentations déclaratives des connaissances, des traitements et du contrôle. L'environnement MAPS apporte dans ce cadre une solution originale, distinguant les connaissances figuratives et les onnaissances opératoires et supportant ainsi les deux axes orthogonaux de conception des systèmes de vision : l'axe centré-figuratif et l'axe centré-opératoire. La définition de deux types d'agents représentant et manipulant ces connaissances, l'introduction de groupes centrés connaissance et centrés tâche permet en outre d'appréhender les niveaux de connaissances et de contrôle selon ces deux axes d'une manière parfaitement homogène. Le besoin d'un contrôle adapté en termes de focalisation et d'adaptation a également été abordé. MAPS permet en effet d'introduire des mécanismes de contrôle sophistiqués tant au niveau local qu'au niveau global, en distribuant le contrôle "figuratif" au sein des agents KS et le contrôle "opératoire" au sein des agents KP.

KISS a ainsi été défini pour combiner diverses filières d'analyse et d'interprétation d'une manière flexible, et pour introduire une coopération entre des primitives à un niveau des niveaux d'abstraction différents et selon des points de vue différents. Notons ici que les techniques et outils de traitements employés dans KISS ont volontairement été simplifiés, ce qui

peut conduire dans certains cas à une dégradation des performances du système. Cependant, globalement les résultats obtenus tendent à valider nos choix architecturaux.

Plusieurs problèmes demeurent cependant en suspens. Le parallélisme permettrait d'améliorer le système KISS pour de nombreux aspects : pour la manipulation d'hypothèses en parallèle ou l'exécution de plusieurs algorithmes ou filières de détection. Nous avons également vu le besoin d'agents de faible granularité pour les représentations multi-échelles et le besoin d'agents superviseur ou observateur (méta-agents) pour résoudre les problèmes de synchronisation. L'évolution du système MAPS vers une architecture hétérogène introduisant des couches de contrôle semble dans ce cadre nécessaire. Il serait alors possible d'introduire des couches réactives correspondant aux niveaux les plus bas de la hiérarchie de traitement, et des couches plus cognitives correspondant aux niveaux les plus élevés et contrôlant les couches inférieures.

References

[Aloymonos 90] J.Y. Aloymonos (1990). Purposive and Qualitative Active Vision, *Workshop on Active Vision European Conference on Computer* Vision, April 90.

[Andress 87] K.M. Andress & A.C. Kak (1987). A production system environment for integrating knowledge with images. Technical report TR-EE-87-36, School of Electrical Engineering, Purdue University, October 1987.

[Ayache 86] N. Ayache & O.D. Faugeras (1986). Hyper: a new approach for the recognition and positionning of 2D objects. *IEEE Trans. on PAMI*, 8(1):44-54.

[Bajcsy 74] R. Bajcsy & L.I. Liberman (1974). Computer description of real outdoor scenes. *Proc. of the 2d ICPR*, IEEE Computer Society Press.

[Ballard 82] D.H. Ballard & C.M. Brown (1982). Computer Vision, Prentice Hall, 1982.

[Baujard 89a] O. Baujard (1989). Un Système de Vision par Ordinateur. *Rapport de DEA*. ENSIMAG / INPG. Grenoble.

[Baujard 89b] O. Baujard and C. Garbay (1989). KISS: Un Système de Vision Multi-Agents. *Actes du VIIème Congrès "RFIA"*., pp 89-98, AFCET / INRIA.

[Baujard 91b] O. Baujard, S. Pesty and C. Garbay (1991). A Programming Environment for Distributed Vision System Design, *in Proc of 6th Int Conf on Image Analysis and Processing*, pp 380-384, World Scientific.

[Belaïd 91] A. Belaïd & Y. Belaïd (1991). Reconnaissance des Formes : Methodes et Applications. InterEditions.

[Berger 91] M. Berger (1991). Towards Dynamic Adaptation of Snake Contours. 6th Int. Conf. on Image Analysis and Processing. (à paraître).

[Bhattacharya 67] C. G. Bhattacharya, "A simple method of resolution of a distribution into Gaussian components," *Biometrics*, vol. 23, pp. 115-135, 1967.

[Boissier 92] O. Boissier & Y. Demazeau (1992). A distributed artificial intelligence view on general purpose vision systems. *in Decentralized AI 3*, Werner & Demazeau Eds. North Holland.

[Bolles 82] R.C. Bolles & R.A. Cain (1982). Recognizing and Locating Partially Visible Objects: the Local-Feature-Focus Method. *Int. J. of Robotics Research*, 1(3):57-82.

[Brooks 83] R.A. Brooks (1983). Model-based Three-dimensional Interpretation of Two-dimensional Images. *IEEE Trans. PAMI*, 5(2):140-150.

[Brown 90] M.D. Brown & R.B. Fisher (1990). A Distributed Blackboard System for Vision Applications. *Proc of the British Machine Vision Conference*. pp 163-168. BMVC 90.

[Canny 86] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 6, pp. 679-698, 1986.

[Canny 86] J.F. Canny (1986). A Computational Approach to Edge Detection, *IEEE Transactions PAMI 8*, N° 6, pp 679-698, IEEE.

[Celeux 89] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier & H. Ralambondrainy. Classification automatique des données, DUNOD, Paris.

[Chassery 84] J.M. Chassery & C. Garbay (1984). An Iterative Segmentation Method based on a Contextual Color and Shape Criterion, *IEEE Trans PAMI 6*, pp 794-800, IEEE.

[Chassery 86] J. M. Chassery and G. Bourrel, "An image package software: IPS design and abilities," *8th Int. Conf. Pattern Recog*, vol. 2, pp. 913-915, Paris 1986.

[Chassery 91a] J.M. Chassery & A. Montanvert (1991). Géométrie discrète en analyse d'images, Série-Image, Ed. HERMES 1991.

[Chazelle 85] B. Chazelle & D.P. Dobkin (1985). Optimal Convex Decomposition. *in Computational Geometry*, Toussaint G.T. Eds, Elsevier Science Publ.

[Chehikian 91] A. Chehikian & J.L. Crowley (1991). Fast Computation of Optimal Semi-Octave Pyramids. *Proc of 7th SCIA*, pp 18-39.

[Clément 89] V. Clément & M. Thonnat (1989). Handling knowledge in image processing libraries to build automatique systems. *2nd Int. Work. on Industrial Applications of Machine Intelligence and Vision*, pp 187-192, Tokyo.

[Crowley 89] J.L. Crowley, A. Chehikian, J.O. Eklundh, J. Kittler, J. Illingworth; G. Granlund, J. Wilkund, E. Granum, H.I. Christensen (1989). Technical Annex for ESPRIT Basic Research Action 3038, Vision As Process, Aalborg, March 89.

[Dellepiane 89] S. Dellepiane, C. Regazzoni, S.B. Serpico & G. Vernazza (1989). "An application-independent knowledge-based framework for complex image recognition," *5th Int. Conf. Image Analysis and Processing*, Positano 1989.

[Demazeau 86] Y. Demazeau (1986). Niveaux de représentation pour la vision par ordinateur. Indices d'image et indices de scène. Thèse de l'Institut National Polytechnique de Grenoble.

[Deriche 87] R. Deriche, "Using Canny's criteria to derive an optimal edge detector recursively implemented," *Int. J. Computer Vision.*, Vol. 1, no. 2, 1987.

[Deriche 90] R. Deriche (1990). Fast Algorithms for Low-Level Vision. *IEEE. Trans on PAMI 12*, N° 1, pp 78-97. IEEE.

[Durbin 87] R. Durbin & D. Willshaw (1987). An Analogue Approach to the Travelling Salesman Problem using an Elastic Net Method. *Nature*, 326, pp 689-691.

[Fabre 88] P. Fabre, F. Albert & P. Louw (1988). Reconnexion de contour utilisant le parcours optimal d'un graphe, application aux images de tunnels. *Congrès PIXIM 88*. pp 379-390.

[Faugeras 82] O.D. Faugeras (1982). Relaxation labelling and evidence gathering. *Proc. of the 7th ICPR*, pp. 405-412, IEEE Computer Society Press.

[Ferrari 80] L. Ferrari, P.V. Sankar & J. Sklansky (1980). Minimal Rectangular Partitions of Digitized Blobs, *5th ICPR*, pp 1040-1043, IEEE Com Soc Press, Miami.

[Fisher 58] W. D. Fisher, "On grouping for maximum homogeneity," JASA, vol. 53, pp. 789-798, 1958.

[Fu 80] K.S. Fu (1980). Introduction. *In "Digital Pattern Recognition"*, pp. 1-14, (K.S. Fu, eds), Springer Verlag.

[Funakubo 84] N. Funakubo (1984). Region segmentation of biomedical tissue image using color texture feature. *Proc. of the 7th ICPR*, pp. 30-32, IEEE Computer Society Press.

[Gagalowitz 85] A. Gagalowitz & O. Monga (1985). Un algorithme de segmentation hiérarchique. *Actes du 5ième Congrès RFIA*, pp. 163-177, AFCET / ADI / INRIA.

[Garbay 86a] C. Garbay (1986). Images, stratégies perceptives et stratégies cognitives d'analyse. Thèse d'Etat. Grenoble.

[Garbay 86b] C. Garbay (1986). Image Structure Representation and Processing : Decision of some Segmentation Methods in Cytology, *IEEE Trans PAMI 8(12)*, pp 140-146, IEEE.

[Garnesson 89] P. Garnesson, G. Giraudon & P. Montesinos (1989). "MESSIE: a multi-expert system in computer vision application for aerial interpretation," *Actes du VIIème Congrès RFIA*, vol. 2, pp. 817-828, AFCET / INRIA.

[GDR 91] GDR 134. (1991). Traitement du Signal et Images : Prétraitement et approche frontière. *Rapport Segmentation*. Décembre 1991.

[Geman 86] D. Geman, S. Geman & C. Graffigne (1986). Locating texture and object boundaries. In "*Pattern Recognition Theory and Applications* ", (P. Devijver, eds), NATO ASI Series, Springer Verlag, Hedelberg.

[Geman 87] S. Geman & C. Graffigne (1987). Markov randon fields image models and their applications to computer vision. *Proc. of the Int. Congress of Mathematicians*, (A.M. Gleason, eds), American Mathematical Society, Providence.

[Granger 85] C. Granger (1985). Reconnaissance d'Objets par Mise en Correspondance en Vision par Ordinateur. Thèse de Doctorat, Université de Nice.

[Hanson 78] A.R. Hanson & E.M. Riseman (1978). VISIONS: a computer system for interpreting scenes. *Computer Vision Systems, (A.R. Hanson & E.M. Riseman, Eds)*, pp. 303-333, Academic Press.

[Hanson 87] A.R. Hanson & .M. Riseman (1987). The VISIONS Image Understanding System, *in Advances in Computer Vision,* C. Brown Eds. L. Erlbaum.

[Haralick 75] R.M. Haralick & I. Dinstein (1975). A spatial clustering procedure for multi-image data. *IEEE Trans. on Circuits & Systems*, CAS-22: 440-450.

[Haralick 81] R.M. Haralick (1981). A facet model for image data: regions, edges and textures. In "Digital Image processing", pp. 337-356, (J.C. Simon and R.M. Haralick, eds), D. Reidel Publishing Company.

[Haralick 84] R.M. Haralick (1984). Digital Step Edges from Zero-Crossings of Second Directional Derivative. *IEEE Transactions PAMI 6*, pp 58-68. IEEE.

[Haralick 85] R.M. Haralick & L.G. Shapiro (1985). Image Segmentation Techniques. *Computer, Graphics, and Image Processing*, 29:100-132.

[Haton 87] J.P. Haton (1987). Les systèmes à base de connaissances en reconnaissance et en interprétation de formes. pp 74-80, *Cognitiva'87.*

[Horaud 87] R. Horaud (1987). New methods for matching 3D objects with single perspective views. *IEEE Trans. on PAMI*, 9(3):401-412.

[Horowitz 76] S.L. Horowitz & T. Pavlidis (1976). Picture segmentation by a tree-traversal algorithm. *J. Assoc. Comp. Mach.* 23:368-388.

[Hu 89] H.T. Hu & J.M. Chassery (1989). Diagrammes de Voronoî généralisés. *Rapport de Recherche 799-IMAG*, Grenoble 1989.

[Hueckel 71] M.F. Hueckel (1971). An operator with locates edges in digitized pictures. *J.Ass. Comput. Match,* Vol 18, N° 1, pp 113-125.

[Kass 87] M. Kass, A. Witkin & D. Terzopoulos (1987). Snakes : active contour models, *Proc of the 1st Int Conf on Computer Vision*, pp 259-268.

[Kittler 85] J. Kittler, J. Illingworth & J. Foglein (1985). Threshold selection based on a simple image statistic, *CVGIP 30,* pp 125-147.

[Klinger 73] A. Klinger (1973). Data structures and pattern recognition. *Proc. of the 1st ICPR*, pp. 497-498, IEEE Computer Society Press.

[Kohl 87] C.A. Kohl, A.R. Hanson & E.M. Riseman (1987). A Goal-Directed Intermediate Level Executive for Image Interpretation. *Proceedings of the tenth International Joint Conference on Artificial Intelligence*, vol. 2 pp 811-814. IEEE Computer Society Press.

[Kropatsch 83] W. Kropatsch (1983). Segmentation of digital images using a priori information about the expected image content. In "Pictorial Data Analysis", pp. 107-119 (R.M. Haralick, eds), Springer Verlag.

[Lane 89] D.M. Lane, M. J. Chantler, E.W. Robertson & A.G. McFadzean (1989). A Distributed Problem Solving Architecture for Knowledge Based Vision. *Distributed Artificial Intelligence.*

[Lee 90] S.U. Lee, S.Y. Chung & R.H. Park (1990). Comparative Performance Study of Several Global Thresholding Techniques for Segmentation. *CVGIP 52,* 171-190.

[Levialdi 83] S. Levialdi (1983). Neighbourhood operators: an outlook. In "*Pictorial Data Analysis*", pp. 1-14, (R.M. Haralick, eds), Springer Verlag.

[Levine 76] M.D. Levine & J. Leemet (1976). A method for non-purposive picture segmentation. *Proc. of the 3d ICPR.*

[Lippmann 87] R.P. Lippmann (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine,* 4(2):4-22.

[Lux 84] A. Lux & V. Souvignier (1984). PVV: un système de vision appliquant une stratégie de prédicion-vérification. *Actes du 4ième Congrès RFIA*, pp. 223-224, AFCET / INRIA.

[Lux 85] A. Lux (1985). Algorithmique et Contrôle en Vision par Ordinateur. Thèse d'état, Université Joseph Fourier et Institut National Polytechnique de Grenoble.

[Mader 88] S. Mader, W. Au & M. Bazakos (1988). A Scene Interpretation System. *SPIE Vol. 937 Applications of Artificial Intelligence VI.* pp 117-123.

[Maitre 91] H. Maitre (1991). Segmentation d'images, Notes de cours ENST Paris.

[Marr 82] D. Marr (1982). VISION : a computational investigation into the human representation and processing of visual information, W.H. Freeman and Company, San Fransisco.

[Martelli 76] A. Martelli (1976). An application of heuristic search methods to edge and contour detection. *Com. ACM*, 19:73-83.

[Matsuyama 85] T. Matsuyama & V. Hwang (1985). "SIGMA: a framework for image understanding - integration of bottom-up anf top-down analyses," *Proc. IJCAI*, vol. 2, pp. 908-915.

[McKeown 85] D. M. McKeown, J. R. Wilson & J. McDermott, "Rule-based interpretation of aerial imagery," *IEEE Trans. Pattern Anal. Machine Intell.*, vol PAMI-7, no. 5, pp 570-585, 1985.

[Melkemi 91] M. Melkemi (1991). Approches géométriques par modèles de voronoï en segmentation d'images. Thèse de Doctorat, Université Joseph Fourier, Grenoble.

[Minsky 75] M. Minsky, "A Framework for Representing Knowledge," *in The Psychology of Computer Vision*, P.H. Winston, Ed., New York: McGraw-Hill, 1975.

[Mitiche 88] A. Mitiche, A. Mansouri & C. Meubus (1988). A knowledge-based image interpretation system. *9th Int Conf on Pattern Recognition.* pp 992-994. IEEE.

[Mohr 88] R. Mohr (1988). Sur l'appariement modèle-perception. Actes du 2nd Atelier Scientifique TIPI, Chap. XXIX, (CNRS, LIFIA, IOTA, LSS).

[Montanari 71] U. Montanari (1971). On the optimal detection of curves in noisy pictures. *Com. ACM*, 14:335-345.

[Montanvert 87] A. Montanvert (1987). Contribution au traitement de formes discrètes. Squelettes et codage par graphe de la ligne médiane. Thèse de l'université Joseph Fourier, Grenoble.

[Montanvert 91] A. Montanvert, P. Meer & A. Rosenfeld (1991). Hierarchical image analysis using irregular tessellations. *IEEE Trans PAMI 13(4)*, pp 307-316, IEEE.

[Nagao 80] M. Nagao & T. Matsuyama (1980). "A structural analysis of complex aerial photographs," New-York: Plenum Press.

[Nagao 84] M. Nagao (1984). Control Strategies in Pattern Analysis. *Pattern Recognition,* 17 (1):45-56.

[Narayanan 83] K.A. Narayanan, D.P. O'Leary & A. Rosenfeld (1983). Multi-resolution relaxation. *Pattern Recog.*, 16:223-230.

[Nazif 84] A. M. Nazif & M. D. Levine, "Low level image segmentation: an expert system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol PAMI-6, no. 5, pp 555-577, 1984.

[Nugues 89] P. Nugues & J.P. Haton (1989). Un Système multi-expert pour l'interprétation d'images d'électrophorèses 2-D. AFCET.

[Ovalle 91] D.A. Ovalle (1991). Contribution à l'étude du raisonnement en univers Multi-Agent : KIDS, une application pour l'Interprétation d'Images Biomédicales. Thèse de l'Université Josehp Fourier - Grenoble I.

[Parvin 84] B.A. Parvin (1984). A split and merge algorithm for segmentation of natural scenes. *Proc. of the 7th ICPR*, pp. 294-295, IEEE Computer Society Press.

[Pavlidis 90] T. Pavlidis & Y.T. Liow (1990). Integrating Region Growing and Edge Detection, *IEEE Trans PAMI 12(3)*, pp 226-233. IEEE.

[Preparata 88] F.P. Preparata & M.I. Shamos (1988). Computational Geometry, an introduction, *Texts and Monographs in Computer Science,* Springer Verlag Ed.

[Prewitt 70] J.M.S Prewitt (1970). Object Enhancement and Extraction. *Picture Processing and Psychopictorics*, B.S. Likin and A. Rosenfeld, Academic Press, pp 75-149.

[Rao 88] A. R. Rao & R. Jain, "Knowledge representation and control in computer vision system," *IEEE Expert.*, pp. 64-79, Spring 1988.

[Regazzoni 91] V. Murino & C.S. Regazzoni (1991). A Distributed Algorithm for Adaptative Regulation of Image Processing Parameters. *1991 IEEE / SMC Int. Conf. on Systems, Man, and Cybernetics.* Vol 1, pp 259-264. IEEE.

[Reinhardt 82] E.R. Reinhardt , W.E. Blanz & al. (1982). Automated classification of cytological specimen based on multi-stage pattern recognition. *Proc. of the 7th ICPR*, pp. 153-159, IEEE Computer Society Press.

[Rosenfeld 82] A. Rosenfeld & A.C. Kak (1982): *"Digital Picture Processing"*. Academic Press.

[Rosenfeld 83] A. Rosenfeld (1983). Quadtrees and pyramids: hierarchical representations of images. In "Pictorial Data Analysis", pp. 29-42, (R.M. Haralick, eds), Springer Verlag.

[Schachter 78] B. Schachter (1978). Decomposition of Polygons into Convex Sets, *IEEE Trans on Computers* 27(11), pp 1078-1082.

[Shapiro 83] L.G. Shapiro (1983). Computer Vision Systems: past, prsent and future. *In "Pictorial Data Analysis"*, pp. 199-237, (R.M. Haralick, eds), Springer Verlag.

[Shen 86] J. Shen & S. Castan (1986). An Optimal Linear Operator for Edge Detection. *Proc of CVPR'86.* pp 109-114.

[Souvigné 83] Souvigné (1983). PVV - Un système d'interprétation d'images par prédiction / vérification. Thèse de 3° cycle. INP Grenoble.

[Tan 86] C.L. Tan & W.N. Martin (1986). A Distributed System for Analysing Time-Varying Multiresolution Imagery. *Computer Vision, Graphics and Image Processing 36.* pp 162-174.

[Tan 89] C.L. Tan & W.N. Martin (1989). An Analysis of a Distributed Multiresolution Vision System. *Pattern Recognition. Vol 22, No. 3.* pp 257-265.

[Tanimoto 75] S. Tanimoto & T. Pavlidis (1975). A hierarchical data structure for picture processing, *CGIP 4*, pp 104-119.

[Tehrani 89] S. Tehrani, T.E. Weymouth & G.B.J. Mancini (1989). An Application of the Blackboard Architecture to Left Ventricular Boundary Detection. *SPIE Vol. 1095 Applications of Artificial Intelligence VII*. pp 503-514.

[Thonnat 89] M. Thonnat & Bijaoui (1989). Knowledge-based classification of galaxies. *Knowledge-Based Systems in Astronomy*, F. Murgath & A. Heck Eds, pp 121-159, Springer-Verlag.

[Tilton 84] J.C. Tilton (1984). Multi-resolution spatially constrained clustering of remotely sensed data on the massively parallel processor. *Proc. of the 7th ICPR*, pp. 1013-1015, IEEE Computer Society Press.

[Tucker 84] L.W. Tucker (1984). Model-guided segmentation using quadtrees. *Proc. of the 7th ICPR*, pp. 216-219, IEEE Computer Society Press.

[Woods 89] P. W. Woods, D. Pycock & C. Taylor (1989). "A frame-based System for Modelling and Executing Visual Tasks," *Image and Vision Computing*, pp. 102-108.

[Wrobel 87] B. Wroebel & O. Monga (1987). Segmentation d'images naturelles. Coopération entre un détecteur de contours et un détecteur de régions. *11ème Colloque GRETSI*, Nice.

[Xu 84] G.Y. Xu & K.S. Fu (1984). Natural scene segmentation based on multiple threshold and textural measurements. *Proc. of the 7th ICPR*, pp. 1111-1113, IEEE Computer Society Press.

[Zhang 84] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Comm of the ACM.*, vol. 3, no. 27, pp. 236-239, 1984.

PARTIE C

VERS UNE PLATE-FORME EN INTELLIGENCE ARTIFICIELLE DISTRIBUÉE

Chapitre 1 : Applications

Chapitre 2 : Etat de l'Art

Chapitre 3 : MAPS Réparti

Chapitre 4 : Perspectives

INTRODUCTION PARTIE C

Outre le domaine de la Vision par Ordinateur (système KISS), l'environnement MAPS a été appliqué aux domaines du Diagnostic Biomédical (système KIDS) et de la Compréhension de la Parole. Les systèmes ainsi conçus, que nous présentons dans le premier chapitre, nous ont permis de dégager les points forts de notre approche mais également d'en montrer les limites.

D'autre part, l'environnement MAPS, bien qu'intégrant des notion issues de l'Intelligence Artificielle Distribuée (agent autonomes, connaissances et contrôle distribués) semble encore proche des langage à objets (langages hybrides) en introduisant une programmation de bas niveau. Il nous a donc paru nécessaire de comparer cet environnement avec les principales plate-formes de développement en Intelligence Artificielle Distribuée et d'en dégager des besoins. L'étude de ces plate-formes est ainsi effectuée dans le second chapitre.

Ces deux études nous ont permis de définir des objectifs à atteindre, qui sont d'ordre fonctionnel et d'ordre structurel. Les améliorations d'ordre fonctionnel concernent l'introduction du parallélisme, à un niveau interne (traitement de plusieurs messages) et à un niveau externe à l'agent (fonctionnement en parallèle des agents de la société), en autorisant l'envoi de messages asynchrones. Elles concernent également l'introduction et l'utilisation de connaissances sur les autres. L'objectif de ces améliorations est de confèrer une plus grande autonomie à l'agent. Les améliorations d'ordre structurel concernent l'introduction d'agents de granularités différents (de l'agent réactif à l'agent cognitif) et une structuration en couches en terme de contrôle. L'objectif de ces modifications est d'obtenir une plus grande hétérogénéité des modèles d'agents et du contrôle.

Les améliorations fonctionnelles sont tout d'abord présentées dans le troisième chapitre. Elles ont conduit à la conception d'une nouvelle version de l'environnement MAPS (MAPS Réparti) et sont implantées en grande partie. A ces modifications fonctionnelles ont également été associées des modifications du langage de programmation et de l'environnement de développement. Les améliorations structurelles, présentées dans le quatrième chapitre, sont encore d'ordre prospectif bien que l'introduction d'une structuration en couche connaisse de premiers développements. Elles s'appuient sur un état de l'art des modèles d'agents réactifs et des architectures hétérogènes multi-couches.

CHAPITRE 1

APPLICATIONS

I INTRODUCTION

Outre le domaine de la Vision par Ordinateur abordé avec le système KISS présenté au chapitre 3 de la partie B, l'environnement de programmation MAPS a été utilisé dans plusieurs domaines d'application. Des systèmes ont été ainsi définis en Diagnostic Biomédical (système KIDS) et en Compréhension de la Parole. Ces systèmes, que nous allons présenter dans ce chapitre, nous ont permis de valider notre modèle initial mais également de dégager les points forts et les faiblesses de l'environnement MAPS. Pour chacun des systèmes, nous donnons un bref aperçu de l'architecture globale et des agents mis en œuvre, puis concluons par une discussion critique.

II LE SYSTEME KIDS

Le système KIDS (Knowledge-based Image Diagnosis System) a donc été conçu à partir de l'environnement de programmation MAPS comme un système multi-agents **[Ovalle 91]** capable de manipuler des connaissances hétérogènes dans le domaine de l'investigation de spécimens en cytologie mammaire, dans le cadre d'un collaboration avec le service de cytopathologie du CHU de Grenoble dirigé par le Professeur Seigneurin. Les connaissances figuratives et opératoires traduisant l'expertise médicale ont été distribuées au sein de plusieurs agents KS et KP connectés en réseau **(Fig. C.1)**.

1. ARCHITECTURE

Les connaissances figuratives décrivent l'ensemble des objets manipulés aux différents stades d'analyse des spécimens cytologiques. Ces connaissances ont été réparties au sein d'agents KS différents, selon leur niveau d'abstraction, mais aussi selon leur relation aux grandes étapes de résolution du problème. On distingue ainsi l'image du spécimen, celle du champ à basse résolution, puis celle du champ à haute résolution, la description des morphologies cellulaires, la représentation des types cellulaires observés, et enfin celle de l'hypothèse diagnostique proposée. Ces connaissances définissent 4 niveaux fondamentaux de représentation : le niveau des images, celui des descripteurs, celui des identifications et enfin celui des compréhensions. Ces niveaux sont les niveaux fondamentaux d'appréhension d'une information visuelle.

En ce qui concerne les connaissances opératoires, six tâches principales ont été modélisées dans le système KIDS et représentées par le biais d'agents KP. Ces tâches sont rappelées ci-dessous, elles adressent divers niveaux d'abstraction de la connaissance :

- validation du spécimen ;
- sélection des champs ;
- changement de résolution ;
- analyse morphologique;
- identification cellulaire ;
- interprétation diagnostique ;

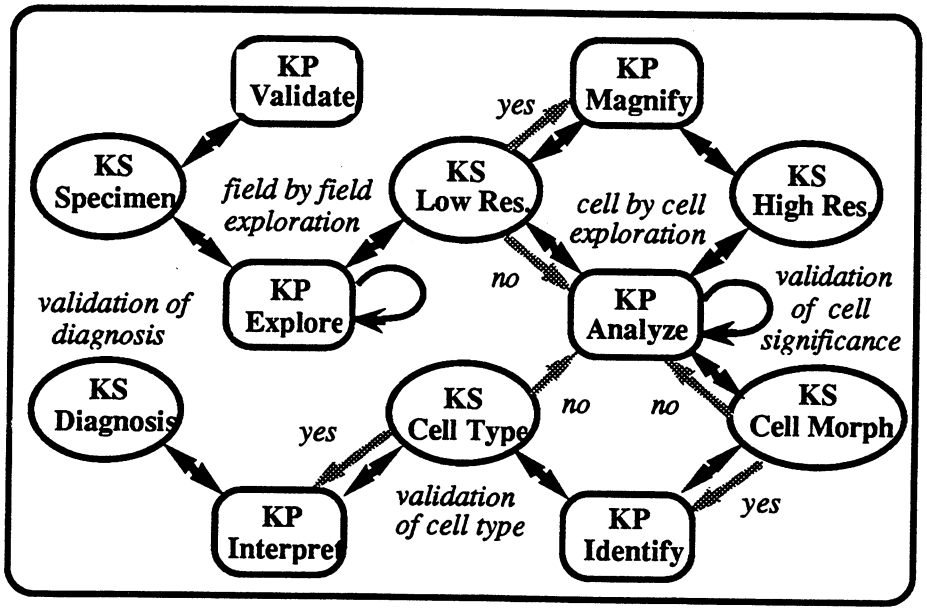


Fig. C.1. Architecture et comportement fonctionnel du système KIDS. Deux boucles d'analyse (flèches circulaires) peuvent être observées, décrivant les stratégies d'analyse "champ par champ" et "cellule par cellule". Les flèches noires reflètent le flot d'information entre agents tandis que les flèches grises représentent l'influence des décisions prises par le système et supportées par l'utilisateur. Notons la présence de deux points de vue différents dans la façon d'analyser une image : basse et haute résolution.

Trois phases d'analyse ou niveaux opératoires, qui définissent trois groupes d'agents différents, sont distinguées dans KIDS : phase exploratoire, phase d'identification cellulaire et phase de formulation diagnostique.

Les rôles à assumer dans la phase exploratoire sont l'acquisition d'images du spécimen (conduite par l'agent KS-Specimen), l'exploration du spécimen (conduite par l'agent KP-Explore) et la sélection des champs significatifs (conduite par l'agent KS-Low-Resolution).

Les rôles à assumer dans la phase d'identification cellulaire sont l'analyse de la morphologie cellulaire (conduite par l'agent KP-Analyze) et le calcul de descripteurs significatifs (conduit par l'agent KS-Morphology), l'identification des types cellulaires selon les caractéristiques morphologiques des cellules (conduite par l'agent KP-Identify) et la mémorisation des cellules identifiées et de leur fréquence d'apparition (conduite par l'agent KS-Cell-Type).

Les rôles à assumer enfin dans la phase de formulation diagnostique sont la formulation diagnostique (conduite par l'agent KP-Interpret) et la mémorisation des hypothèses probables de diagnostic (conduite par l'agent KS-Diagnosis).

2. DISCUSSION

En ce qui concerne ce domaine, l'environnement MAPS a révélé son intérêt en phase de modélisation : modélisation des phases de résolution d'un problème, des connaissances et des tâches. Par contre, outre l'absence de parallélisme déjà révélé par le système KISS, un manque d'abstraction s'est également fait ressentir. Il manque en effet un niveau d'abstraction suffisamment élevé pour faciliter la conception. Des agents de type "méta" serait d'une grande utilité dans ce cadre.

Le système KISS a mis en avant trois notions essentielles pour la conception d'un système multi-agents, la notion de point de vue, la notion de niveau et la notion de phase d'analyse. Une étude approfondie permettrait de mieux cerner les problèmes d'organisation d'un système multi-agents en fonction de la nature des connaissances manipulés. Le système KIDS a ainsi amené son concepteur à étudier une typologie formelle pour la classification des connaissances et du raisonnement, en vue de la de conception de système multi-agents **[Ovalle 91]**. Selon cette typologie, quatre axes principaux d'organisation de la connaissance ont été dégagés : un axe fonctionnel, un axe structurel, un axe "niveau d'abstraction" et un axe "domaine d'application". L'axe fonctionnel regroupe les connaissances figuratives, opératoires, réflexives et heuristiques, et peut être aisément appréhendé par MAPS. L'axe structurel regroupe les connaissances intrinsèques, contextuelles, compositionnelles, taxinomiques et causales, plus difficilement modélisables avec MAPS. Les axes "niveau d'abstraction" et "domaine d'application" regroupent les connaissances profondes (connaissances générales) et de surface (connaissances opératives et routinières) qui peuvent être modélisées avec MAPS. Cependant, la définition de niveaux d'abstraction supplémentaires avec l'introduction de méta-agent par exemple permettrait de mieux appréhender ce dernier axe. En ce qui concerne les raisonnements, quatre niveaux ont été définis fondés sur les quatres axes d'organisation de la connaissance, et qui démontrent également le besoin d'une plus grande abstraction, surtout en ce qui concerne les raisonnements sur les connaissances profondes et de surface.

III LE SYSTEME DE COMPRÉHENSION DE LA PAROLE

Le système de compréhension de la parole **[Caillaud 91]**, a été défini dans le cadre d'une coopération avec l'équipe de Jean Caelen, de l'Institut de la Communication Parlée de Grenoble (ICP). Le but de cette collaboration est de définir une architecture logicielle commune devant supporter deux systèmes, dédiés à la vision et à la compréhension de la parole.

1. ARCHITECTURE

L'architecture proposée est fondée sur les travaux de Sperry et des ses collaborateurs **[Sperry 62]**, concernant la latéralisation du cerveau. Ces travaux ont montré que les deux hémisphères cérébraux sont capables de fonctionner indépendamment et normalement, le corps caleux étant une voie de communication entre les deux hémisphères. Dans le cas de la compréhension de la parole, les deux hémisphères ont chacun leur propre domaine de spécialisation. En simplifiant, l'hémisphère droit traite l'information de façon concrète et non-verbale et l'hémisphère gauche de façon abstraite et verbale. Le réseau d'agent obtenu est illustré par la figure **C.2**. Cette architecture définit une double analyse ; un processus global effectué par l'hémisphère droit, reliant des éléments épars le long de la phrase pour constituer une chaîne de points d'ancrage et un raisonnement détaillé effectué par l'hémisphère gauche pour compléter et affiner les informations locales laissées en attente par le premier processus.

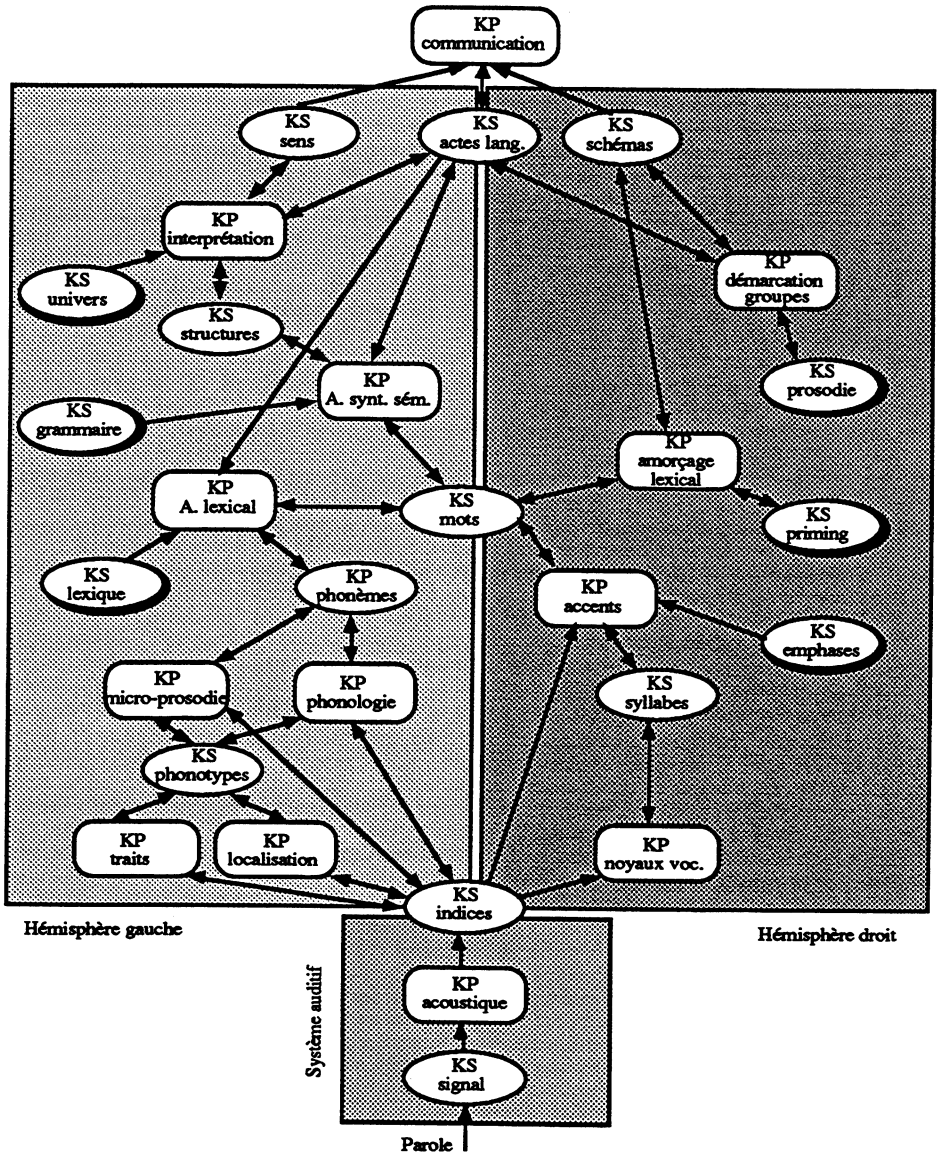


Fig. C.2. Architecture globale du système de compréhension de la parole. Deux filières d'analyse parallèles sont dégagées, une filière correspondant à l'hémisphère gauche et une filière correspondant à l'hémisphère droit.

Comme pour les systèmes KISS et KIDS, les niveaux fondamentaux de représentation et les principales tâches de l'analyse du signal sont définis par une distribution des connaissances au sein d'agents KS et KP. Les niveaux figuratifs concernent en particulier le signal (KS Signal), les indices (KS Indices), les phonotypes et phonèmes (KS Phonotypes et KS Phonèmes), les mots et les structures sémantiques (KS Mots et KS Structures) et enfin le sens (KS Sens). Quatre phases d'analyse (ou niveaux opératoires), qui définissent quatres groupes d'agents différents, sont également distinguées :

- une phase d'analyse acoustique ;
- une phase d'analyse phonétique ;

- une phase d'analyse linguistique ;
- une phase d'ancrage.

Les rôles à assumer dans la phase d'analyse acoustique sont la distribution du signal à analyser et des variables d'environnement (conduite par l'agent KS-Signal), l'analyse du signal (conduite par l'agent KP-Acoustique) et la focalisation sur les informations obtenues (conduite par l'agent KS-Indices). Cette phase est commune aux deux filières d'analyse cerveau gauche / cerveau droit.

Les rôles à assumer dans la phase d'analyse phonétique sont l'obtention d'informations accoustico-phonétiques (conduite par les agents KP-Localisation et KP-Traits) qui sont ensuite distribuées par l'agent KS-Phonotypes, et la conversion des phonotypes en phonèmes (conduite par les agents KP-Micro-Prosodie et KP-Phonologie) stockés dans l'agent KS-Phonèmes.

Les rôles à assumer dans la phase d'analyse linguistique sont l'analyse lexicale (conduite par l'agent KP-Analyseur-Lexical) qui fournit des informations lexicales communes aux deux filières d'analyse (KS-Mots) et l'analyse syntaxico-sémantique de ces informations lexicales (conduite par l'agent KP-Analyseur-Syntaxico-Sémantique), les structures obtenues étant gérées par l'agent KS-Structure. L'analyse pragmatique de ces structure est ensuite effectuée (agent KP-Interprétation) pour obtenir le sens dérivé du contexte de la communication (stocké dans l'agent KS-Sens). Ces deux phases font partie de la filière d'analyse cerveau gauche.

Les rôles à assumer dans la phase d'ancrage sont la définition de frontières de mots sur des considération d'ordre prosodique (conduite par l'agent KP-Démarcation-Groupes) et lexical (conduite par les agents KP-Accent et KP-Amorçage). Cette dernière phase constitue la filière d'analyse cerveau droit.

Un groupe d'agent est finalement définit pour assurer l'interface entre l'utilisateur et le système de compréhension (agent KP-Communication et KS-Actes-de-Langages).

2. DISCUSSION

Comme dans les applications de vision et de diagnostic, le parallélisme intrinsèque de l'architecture n'est pas exploité du fait de la séquentialité de l'environnement MAPS. Une version parallèle de cet environnement permettrait de mieux appréhender les problèmes de contrôle et de synchronisation posés par la double analyse induite par l'architecture cerveau gauche / cerveau droit. Ce besoin de contrôle et de synchronisation pourrait également être étudié à travers la définition d'agents de haut niveau, constituant une couche de supervision et d'observation placée au dessus des agents. Une architecture introduisant ce type d'agents, présentée en figure **C.3**, a été définie. Dans cette architecture, chaque méta-agent KP

correspond à une phase d'analyse. Les méta-agents KS correspondent à un niveau de connaissances figuratives commun aux deux filières cerveau gauche / cerveau droit et regroupent donc les points de vue figuratifs correspondant à un niveau figuratif donné.

Enfin, cette application nécessite le maintient d'un nombre très important d'hypothèses. La possibilité d'étudier en parallèle plusieurs hypothèses (mondes multiples) permettrait au système de gagner en puissance et en rapidité. Des stratégies de prédiction / vérification joueraient également un rôle important dans ce cadre.

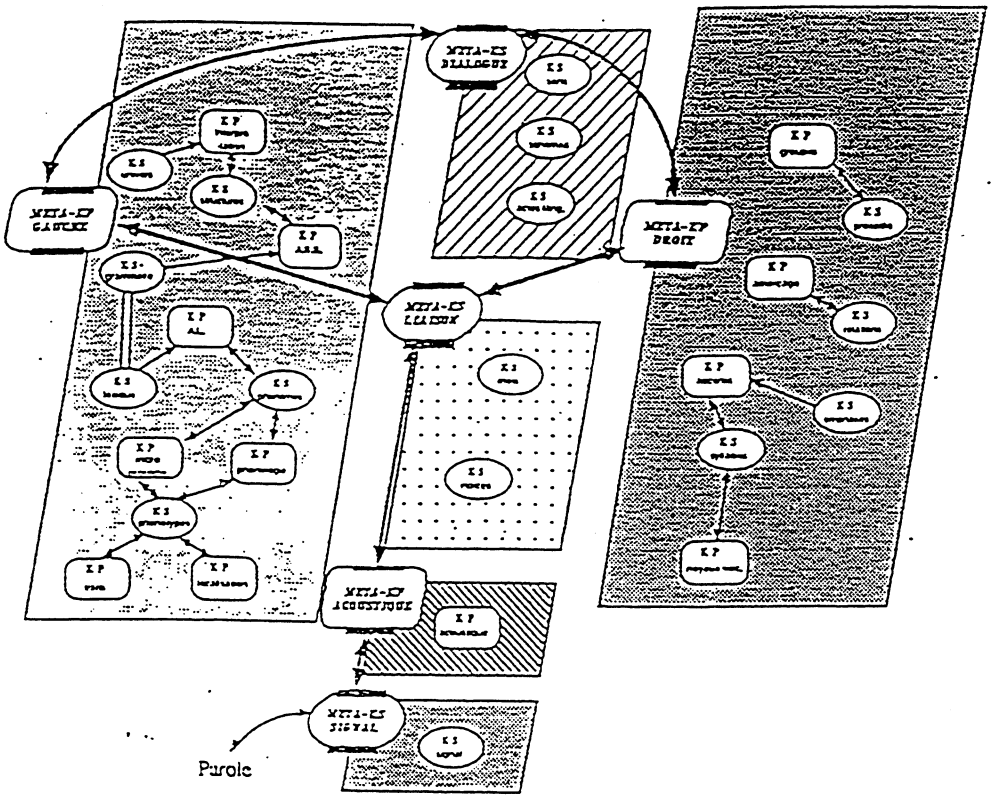


Fig. C.3. Architecture d'un système de compréhension de la parole introduisant un niveau de supervision et d'observation par l'intermédiaire de méta-agents.

IV CONCLUSION

Comme le démontrent les nombreuses applications réalisées, l'environnement de programmation MAPS permet de concevoir des applications dans des domaines variés, ce qui prouve la validité de notre modèle. Il est en effet facile de décomposer un problème à l'aide d'agents KS et KP qui appréhendent parfaitement la décomposition tâches / connaissances. La souplesse du langage de programmation y est également pour beaucoup. Par contre, des faiblesses apparaissent qui relèvent de niveaux différents. Des faiblesses au niveau conception

apparaissent tout d'abord ; la conception des agents dépend d'une programmation parfois trop bas niveau et manque de niveaux d'abstraction supplémentaires. Les faiblesses sont également d'ordre fonctionnel ; le mode d'exécution est en effet séquentiel et l'autonomie des agents de même que le comportement global des applications créées peut en pâtir. Enfin, les faiblesses sont aussi d'ordre structurel, et rejoignent les faiblesses du niveau conception ; il manque en effet des niveaux de connaissances et de raisonnements supplémentaires qui pourraient être introduits par une structuration à l'aide de méta-agents. Les réponses à ces faiblesses constituent les objectifs que nous allons essayer d'atteindre dans les futures versions de MAPS.

CHAPITRE 2

ETAT DE L'ART

I INTRODUCTION

Notre premier souci est de rapprocher MAPS d'une vraie plate-forme multi-agents, en donnant plus d'autonomie aux agents, ceci pour répondre aux besoins fonctionnels des applications. Nous proposons ainsi dans ce chapitre, une revue des principales plate-formes en Intelligence Artificielle Distribuée. Cet état de l'art est suivi d'une discussion permettant de situer l'environnement MAPS dans sa forme actuelle et dans ses extensions futures.

II ENVIRONNEMENTS DE PROGRAMMATION

L'intérêt des plate-formes de développement en Intelligence Artificielle Distribuée est de permettre de s'abstraire de problèmes spécifiques et d'autoriser le développement et la mise au point de différents types d'agents, de différentes organisations et la conception d'applications dans des domaines variés. Selon Bouron **[Bouron 91]**, deux types de plate-forme peuvent être définis, les premières portant plutôt sur le test d'un type agent spécifique, ou d'une technique de coordination spécifique, les secondes étant générales, au sens où elles peuvent être utilisées pour tester et comparer plusieurs techniques de coordination. Nous nous intéressons plutôt dans ce qui suit à ce second type d'approche. Nous étudions tout d'abord les langages de programmation associés à ces plate-formes, puis leurs fonctionnalités et styles d'implantation.

1. LANGAGES DE PROGRAMMATION

En ce qui concerne les langages de programmation, deux approches peuvent être utilisées : la première consiste en une simple extension d'un langage existant, la seconde, par contre, implique la conception de langages spécifiques.

1.1. EXTENSIONS DE LANGAGES EXISTANTS

MAGES **[Bouron 91]** est une plate-forme écrite en Smalltalk comme une implantation du langage Actalk, un langage d'acteur développé par J.B. Briot, et qui permet à différents modèles d'acteurs d'interagir. Smalltalk a été étendu pour incorporer des slots dans les objets du langage, le transformant en langage de représentation des connaissances à base de frames. Ce langage et ses extensions ont été largement utilisés par la communauté de l'Intelligence

Artificielle Distribuée : ils offrent en effet la modularité, le parallélisme et des outils graphiques, ainsi que l'implantation d'agents hétérogènes. MAGES implante une hiérarchie d'agents prédéfinis que l'utilisateur peut utiliser pour concevoir ses applications. Un protocole d'extension permet en outre à l'utilisateur de concevoir de nouveaux agents et de nouveaux interpréteurs de messages.

Maruichi et Suyeoshi **[Maruichi 90]**, **[Sueyoshi 91]**, utilisent le système de programmation orienté-objet PANDORA-II, qui est implanté sur le langage KCL. Le système conçu par Ferguson **[Ferguson 92]** est implanté en SICStus Prolog et la plate-forme MICE **[Montgomery 92]** est implantée en Common Lisp.

1.2. LANGAGES SPECIFIQUES

D'autres plate-formes possèdent un langage de programmation qui leur est propre. Le système MACE **[Gasser 87]** utilise un langage de description de base de données conçu spécialement pour permettre la description d'agents, le langage ADL. ADL est un langage procédural permettant de décrire les opérations nécessaires à la construction d'une nouvelle description d'agent. Il est possible de créer des hiérarchies d'agents permettant l'héritage de valeurs par défaut. Un agent en ADL est ainsi défini par rapport à une classe existante, ou comme en définissant une nouvelle. Une nouvelle classe peut être créée par importation de caractéristiques d'autres classes, telles les moteurs d'inférence ou les attributs. De nouvelles caractéristiques peuvent être créées, et la procédure d'initialisation d'un agent (sorte de constructeur) peut être spécifiée.

Il existe dans ce langage des agents prédéfinis dont le rôle est l'interprétation des commandes, la gestion de l'interface, la définition de nouvelles descriptions d'agents, la gestion des erreurs, le tracé et le suivi de l'exécution. D'autres agents spécifiques servent d'outils interactifs pour construire les programmes MACE. Ces outils incluent des agents pour construire et éditer les comportements et la structure des autres agents, aussi bien que pour changer les paramètres de l'environnement d'exécution, ou de la simulation. Parmi les facilités dont disposent les agents, citons un "pattern matcher", un simulateur, plusieurs moteurs standards, des gestionnaires d'erreurs, et des messages standards.

Bien que l'extension de langage existant soit plus simple, la définition d'un langage spécifique nous parait indispensable, permettant une plus grande déclarativité et facilitant la conception des applications. Cependant, il nous parait également nécessaire de ne pas se fermer sur ce langage mais au contraire de rester ouvert sur l'extérieur, c'est-à-dire de permettre l'intégration de code dans des langages différents.

2. INTERACTION HOMME-MACHINE

Des interfaces sophistiquées sont nécessaires pour guider la conception mais aussi pour visualiser l'exécution des systèmes. Certaines plate-formes vont jusqu'à la définition d'un langage de programmation graphique pour la conception de systèmes.

Dans la plate-forme MAGES **[Bouron 91]**, des interfaces utilisateur sont définies pour faciliter les tâches de mise au point et de suivi d'exécution. Ces interfaces sont fondées sur le modèle "Modèle-Présentation-Contrôleur" de Smalltalk. Des "browsers" spécifiques ont été introduits pour visualiser les classes d'agents, de messages et de sources de connaissances. Des classes de "browsers" génériques permettent de plus la création de "browsers" adaptés à chaque type d'agents. Les activités des agents (déplacement, communication) peuvent être visualisées de plusieurs façons (par exemple les messages sont visualisés à l'aide d'icônes se déplaçant d'un agent à l'autre). Des fenêtres spécifiques peuvent être utilisées pour visualiser l'état de la boîte aux lettres d'un agent, l'envoi ou la réception de messages. L'utilisateur peut interrompre l'exécution, stopper et inspecter un message.

A tout agent est associé une icône et une position. Les groupes d'agents sont représentés par deux types de fenêtres : la vue du réseau d'accointances, et la vue des communications. Les canaux de communications sont visualisés sur le réseau d'accointances.

Dans une philosophie proche de la plate-forme ARCHON **[Jennings 91]**, la plate-forme ABE **[Hayes-Roth 88]**, **[Hayes-Roth 91]** permet la conception d'un système par composition de modules selon des topologies et des styles de communication arbitraires. Des modules différents permettent des compositions de types particuliers, par exemple de type flot de données, tableau noir, traitement de transaction, ou contrôle procédural. Chaque module dispose d'outils de développement graphiques adaptés. L'approche est récursive, dans la mesure où tout système composé de modules ABE devient à son tour un module, et tout module peut impliquer d'autres modules de manière hiérarchique, ou sous la forme de réseaux. Ceci permet à certains modules d'être partagés par plusieurs modules composites. Les modules simples sont dits primitifs et sont considérés comme des boites noires. Ces modules primitifs sont conçus dans un environnement spécifique, appelé "Boite Noire". Leur comportement peut être codé dans n'importe quel langage. Le style d'interaction retenu pour effectuer ces opérations est la programmation graphique.

La plate-forme MICE est un outil pour étudier la coopération entre des systèmes intelligents, où l'accent est mis ici sur les relations de l'agent avec son environnement plus que sur les raisonnements de l'agent **[Montgomery 92]**. Cette plate-forme dispose ainsi d'une interface utilisateur qui permet de représenter les caractéristiques des agents et de l'environnement et de visualiser les déplacements des agents dans l'environnement.

3. FONCTIONNALITES ET STYLE D'IMPLANTATION DES PLATE-FORMES

Dans ce qui suit, nous décrivons le style d'implantation des plate-formes, les outils d'aide à la conception qu'elles mettent en œuvre, les outils d'évaluation des comportements des agents et les outils d'aide à la simulation.

Style d'Implantation

La plate-forme MACE **[Gasser 87]** est écrite elle même sous la forme d'un système multi-agents. Elle est implantée sur un hypercube 16 noeuds, dans un environnement de type machine LISP (TI Explorer). C'est à la fois un langage, un environnement de programmation et une plate-forme de test. Son but est de pouvoir expérimenter différents styles de systèmes distribués, à des niveaux de granularités différents.

Dans la plate-forme MADE **[Woolridge 91]**, les agents peuvent être implantées dans chacun des quatre langages suivants : Common Lisp, Prolog, POP-1 et C. Des groupes d'agents peuvent être instanciées sur une machine unique ou distribués sur un réseau. Le programme MASC (Multi-Agent system Description Compiler) permet de traduire une définition de haut niveau du système MADE en une séquence d'instructions nécessaires à l'instanciation du système. Le programme MP (Model Parser) est un compilateur qui traduit un modèle abstrait d'agent en une structure de donnée du langage de destination requis.

La plate-forme MICE **[Montgomery 92]** est implantée en Common Lisp et peut donc être portée sur de nombreuses machines. Des routines graphiques indépendante d'une quelconque architecture matérielle lui permettent de fonctionner sur station de travail (sous X Window) sur MacIntosh et sur TI Explorer.

Aide à la conception

La plate-forme MACE **[Gasser 87]** fournit un langage de description des agents, et permet de procéder par raffinement progressif dans la construction de systèmes complexes. Des facilités de mise au point sont disponibles. Un ensemble d'agents systèmes est utilisé pour construire les agents, de la description au suivi d'exécution, de la manipulation d'erreurs aux interfaces utilisateurs.

La plate-forme ABE **[Hayes-Roth 88]** offre l'accès à plusieurs environnements de développement et de conception graphique de haut niveau, appelés modules. Elle facilite la réorganisation de composants logiciels et leur attachement aux composants matériels disponibles. Par essence, ABE est donc un environnement et un système d'exploitation coopératif pour la conception de systèmes intelligents.

La plate-forme MAGE **[Bouron 91]** est assez flexible pour permettre la définition d'organisations centrées comportements ou centrées connaissances : dans les premières, les agents réagissent aux changements de l'environnement, dans les secondes, les actions sont la conséquence d'une interaction entre buts explicites et intentionnels. De plus des agents de granularité variable peuvent être définis par extension d'un modèle de base très simple, ce qui facilite l'accès au système. Une librairie d'agents pré-définis peut être également utilisée. Les agents peuvent être créés et détruits de manière dynamique. Notons que la destruction dynamique des agents peut poser des problèmes de communication : ces problèmes sont résolus par l'introduction de "trous noirs" retournant un message signalant l'absence de récepteur si l'émetteur est un agent, ou l'absorbant dans le cas contraire.

RATMAN **[Bürckert 91]** est une plate-forme pour la définition et le test d'agents rationnels dans un environnement multi-agents. Un agent est défini comme une base de données hiérarchique comprenant tous les niveaux, depuis la connaissance sensorielle jusqu'aux capacités d'apprentissage. A chaque niveau de connaissance, le concepteur peut décider si l'agent doit posséder ou non telle capacité et en définir la granularité. La particularité de RATMAN est qu'elle est fondée purement sur la logique : il est possible, à l'intérieur du paradigme logique, de spécifier toutes les caractéristiques individuelles des agents, des plus simples aux plus compliquées.

L'architecture de RATMAN implique quatre modules : la boite à outils agent, le kit de spécification, le scénario courant du monde et la boite d'état séquence / statistiques. Le kit de spécification d'interface utilisateur pour définir les agents, leur relation dans le monde et leur état dans la société. La boite à outil agent fournit une base de connaissances structurée hiérarchiquement, avec des facilités de raisonnement pour modéliser toutes les caractéristiques des agents. La connaissance prédéfinie peut être utilisée ou partiellement non utilisée, et de nouvelles connaissances peuvent être introduites par l'utilisateur. Les agents comme les sociétés d'agents peuvent être placés par l'utilisateur dans le scénario courant du monde. De plus, un tableau noir est introduit, qui sert de plate-forme de communication pour les agents et l'utilisateur.

Le scénario courant du monde remplit plusieurs tâches différentes : il rend tout d'abord compte de l'état du monde, par l'intermédiaire d'un tableau noir. Sur ce tableau noir sont en effet inscrites des informations sur les objets du monde, sur les positions et les relations des agents, et sur le contenu de leurs communications. Un autre module permet de visualiser l'état des agents, les actions à entreprendre prochainement, les actes de communications ou les plans. L'utilisateur peut également intervenir comme un agent, en déposant des informations sur le tableau noir et en possédant son propre représentant dans la zone dédiée aux agents.

Le Kit de spécification comporte trois modules, les facilités de conception des agents, le module du paradigme général, et le vérificateur de consistance. Le module du paradigme général maintient les informations générales sur la société d'agents, telles le nombre d'agents dans le système, si l'organisation est anarchique ou hiérarchique, s'il y a des agents avec des responsabilités spéciales, et plus généralement si la société est homogène, hétérogène, ou en colonie.

MICE **[Montgomery 92]** permet un certain nombre d'opération de mise au point et maintient des mesures statistiques correspondant au nombre de déplacement tentés par un agent au nombre qui ont réussi. MICE tirant partie des possibilités de sauvegarde et restauration de contexte de Common Lisp, il est possible de sauver des exécutions et de les restaurer ultérieurement.

Evaluation de comportements

La plate-forme MACE **[Gasser 87]** permet la mesure des caractéristiques des résolveurs de problèmes pendant les expériences. Le trafic des messages, la taille des bases de données et des files d'attente, le travail effectué par un agent, ou la charge d'un processeur sont des exemples de mesures.

Une architecture commune et unique a été implantée dans le système proposé par Fergusson **[Ferguson 92]**, qui permet l'étude fonctionnelle et comportementale de diverses configurations d'agents. Le but principal de la recherche est de comprendre les contraintes exercées sur le comportement de l'agent par les conditions environnementales et les capacités fonctionnelles de l'agent. A cette fin une plate-forme expérimentale comprenant un monde multi-agents simulé a été construite. La plate-forme offre une facilité de définition permettant d'agir sur les paramètres de l'agent (distribution des ressources dans les couches, capacité de planification, sensibilité des règles réactives ou fréquence d'accès au monde) ou de l'environnement (taille et type de la population, présence d'agents très rapides ou temps cpu alloué).

Aide à la simulation

Pour permettre des tests répétitifs, la plate-forme MACE **[Gasser 87]** inclut un simulateur paramétrable : des communautés d'agents peuvent être développées sur le simulateur puis portées sur une architecture parallèle. Le simulateur permet de simuler différentes vitesses de processeurs, différents styles d'interconnexions, aussi bien que les erreurs de communication et de traitement dans le réseau, ou les interruptions d'agents, en les modulant par un taux de probabilité.

Dans la plate-forme RATMAN **[Bürckert 91]**, la boite de facilité de conception d'agents est utilisée pour activer ou désactiver des parties de la base de connaissance de l'agent de façon à simuler des décisions de conception et à réaliser des agents particuliers en leur ajoutant des connaissances spécifiques ou en spécifiant des canaux de communication. Selon le niveau des connaissances sélectionnés, des agents de granularité différente sont obtenus.

La tâche du vérificateur de consistance est de rapporter des conflits entre les définitions du module de paradigme général et des spécifications d'agents simples. Par exemple, il y a une inconsistance évidente s'il est défini que les agents communiquent entre eux alors que tout le niveau communication est désactivé dans la spécification des agents.

III CONCLUSION

Deux types de plate-formes sont considérés : les plate-formes possédant un langage de programmation, et les plate-formes constituées d'agents pré-définis. Dans ce second cas, l'approche s'apparente à la gestion et à l'exploitation de base de données. Cette base de donnée, ou base d'agents pré-définis peut alors servir à la conception d'applications, à la conception d'agents (à partir des briques constituant les agents) et peuvent être étendues avec des agents entièrement nouveaux. Certains auteurs définissent ainsi des langages pour gérer et manipuler ces bases d'agents **[Gasser 87]**. Ces approches paraissent particulièrement intéressantes. Elles permettent en effet de s'abstraire d'un langage de programmation classique et lourd à manipuler tout en gardant sa souplesse pour l'extension et la création de modèles. Cette approche est similaire à l'approche MAPS, qui définit des types d'agents prédéfinis spécialisables par la suite à l'aide du langage de programmation.

Il faut noter également les besoins distincts d'outils pour des plate-formes qui visent à faciliter la conception de modèles d'agents et l'étude des styles de coopération ou de communication, et d'outils pour des plate-formes qui visent à faciliter la conception d'application concrètes.

Nous présentons dans le tableau suivant **(Table 1)** une comparaison entre l'environnement MAPS et les deux plates-forme qui s'y apparentent le plus, les plates-forme MACE **[Gasser 87]** et MAGES **[Bouron 91]**. Cette comparaison est effectuée au niveau de l'agent (autonomie, création, granularité, hétérogénéité des connaissances, représentation des connaissances et raisonnement), des organisations d'agents (structuration des agents, représentation des autres), des communications (types et protocoles) et des fonctionnalités de la plate-forme.

Caractéristiques	MACE	MAGES	MAPS
Système			
Contrôle	distribué	distribué	distribué
Environnement	accointances	accointances et structures	codage explicite
Adaptativité	oui	oui	non
Agent			
Autonomie	***	***	**
Création	recopie et modification de classes	hiérarchie de classes	classes et langage de description
Granularité	variable	variable	moyenne
Hétérogénéité des connaissances	moyenne	bonne	bonne
Représentation des connaissances	règles	frames, règles, ...	règles / objets
Raisonnement	sur les accointances à partir de règles	selon le type d'agent (tableau noir, agent communicant...)	règles + moteurs
Communication			
Type	asynchrone	asynchrone	synchrone
Protocoles	non	variable (actes de langages)	non
Organisations			
Niveaux d'abstraction	organisation / groupe	groupe	non
Fonctionnalités			
Utilisation	tests et mesures caractéristiques de systèmes	test de systèmes, but éducatif et de recherche	développement d'applications
Outils	***	**	**
Interface	*	***	**
Portabilité	*	**	**

Table 1. Tableau récapitulatif des caractéristiques des plate-formes MACE, MAGES et MAPS

Comme le montre ce tableau, les trois plate-formes possèdent de nombreux points communs, les points faibles de MAPS concernent essentiellement la communication, la représentation des connaissances sur les autres et l'autonomie des agents. Ce sont ces aspects, plutôt fonctionnels, que nous avons choisi d'améliorer tout d'abord et qui vont être décrits dans le chapitre suivant. Les points faibles concernent également la granularité des agents et le manque de niveaux d'abstraction. Ces aspects, plutôt structurels, seront abordés de façon plus prospective et concernent les améliorations à apporter dans les futures versions.

CHAPITRE 3

MAPS REPARTI

I INTRODUCTION

Ce chapitre décrit la nouvelle version de l'environnement de programmation MAPS, qui introduit un mode d'exécution parallèle **[Baujard 91a] [Baujard 92]**, devant accroître le potentiel de décision et l'autonomie intrinsèque des agents. Les agents doivent en effet parfois opérer de manière indépendante, parfois se coordonner et parfois travailler en concurrence. Rappelons également l'importance de la notion de point de vue et de filière d'analyse, déjà évoqués, qui doivent être traités en parallèle. De tels comportements impliquent l'introduction du parallélisme à un premier niveau : celui de l'activation de l'agent, qui implique une concentration des efforts sur la coopération et la synchronisation. Le parallélisme est également nécessaire à un second niveau : celui du traitement des messages. Il est en effet nécessaire de donner aux agents la possibilité de traiter plusieurs requêtes simultanément.

Ce chapitre présente ainsi les modifications fonctionnelles qui ont été effectuées, issues de nos conclusions du chapitre précédent, et concernant les agents et la société d'agents. A ces modifications fonctionnelles sont associées des modifications du langage de programmation et de la plate-forme (implantation, style d'interaction). Le langage de programmation a ainsi été rapproché du langage cible (C++) ce qui permet une meilleure lisibilité, mais surtout une plus grande ouverture, et qui laisse envisager, la possibilité d'étendre les caractéristiques des agents par une programmation directe avec le langage C++ ou avec un autre langage.

II ARCHITECTURE LOGICIELLE

Dans cette nouvelle version, les agents MAPS sont conçus comme des processus dont le rôle majeur est d'attendre des requêtes : ces requêtes sont alors interprétées et le contrôle est transmis à un processus fils dynamiquement créé qui prend en charge leur traitement. Un tel mécanisme fournit les bases pour la génération et la manipulation des mondes hypothétiques. Dans ce paragraphe, nous allons décrire tout d'abord les modifications apportées au niveau de la définition des agents, puis l'apport du parallélisme au niveau de la société d'agents.

1. LES AGENTS MAPS

L'introduction d'un mode d'exécution parallèle nécessite des modifications au niveau de la

structure des agents MAPS afin d'augmenter leur autonomie. Ces modifications portent d'une part sur les protocoles de communication et d'autre part sur le contrôle. Certains aspects, telles les ressources, voient également leur fonctions étendues par l'introduction du parallélisme. Dans ce paragraphe, nous nous contentons d'une présentation des modifications et des nouveaux apports pour chacune des rubriques introduites au chapitre 3 de la partie A.

1.1. RESSOURCES

Une nouvelle dimension est donnée aux instances d'objets par le traitement parallèle des messages par un agent. Nous avons vu au chapitre précédent qu'une instance d'objet pouvait être assimilée à un monde de pensée, et que l'ensemble des instances constituait l'environnement de travail de l'agent. L'introduction du parallélisme au niveau du traitement des requêtes permet à un agent de maintenir plusieurs mondes de pensée en parallèle : les instances d'un objet peuvent ainsi appartenir à des mondes différents **(Fig. C.4)**. A ce niveau, tout se passe comme s'il y avait multiplication virtuelle et dynamique des ressources de l'agent. De cette façon, tout agent explore l'environnement de manière parallèle et simultanée au lieu d'être condamné à l'explorer de manière séquentielle.

Chaque instance correspond à une perception ou une inférence qui correspond à la détection ou la compréhension, par un agent KP, d'un événement de l'environnement. Le parallélisme permet d'émettre plusieurs hypothèses sur ces événements en termes de détection (hypothèses de partitionnement) et d'interprétation. Ce mécanisme correspond au mécanisme des mondes multiples. Les méthodes de manipulation des instances permettent ainsi de créer, de modifier ou de détruire des mondes de pensée en parallèle.

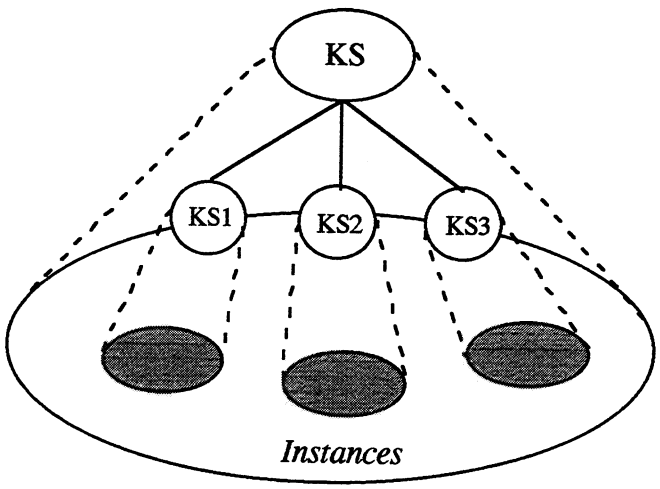


Fig. C.4. Mondes de pensée en parallèle dans un environnement commun.

1.2. LES CONNAISSANCES SUR SOI ET LES AUTRES

Un agent KS ou KP connait à présent l'adresse physique de ses agents accointances (nom de la

machine hôte et port de communication) en plus de leur nom. Ces connaissances sont stockées dans la liste des connexions et sont utilisées pour les opérations de communication de bas niveau. Un agent connait également l'adresse et le port de communication du contrôleur d'application (voir paragraphe IV.2), ce qui lui permet de communiquer avec celui-ci.

Une des faiblesses de l'environnement MAPS concernait la représentation des connaissances sur les autres et l'utilisation de ces connaissances. Nous avons choisi d'introduire la possibilité de les modéliser sous la forme d'attributs d'objets. Il s'agira pour le concepteur de définir dans un agent KS des objets dont les attributs permettront par exemple de modéliser le rôle, la compétence d'un agent KP, les instances de cet objet permettront alors de modéliser dynamiquement le rôle et la compétence des agents KP connectés à l'agent KS. Les raisonnements sur ces connaissances se feront par l'intermédiaire des règles.

Pour les agents KS, cette modélisation ne pose pas de problème, les connaissances seront stockées au sein de l'agent qui pourra utiliser ses règles de propagation, de proposition et d'interrogation pour raisonner sur ces connaissances. Par contre, pour les agents KP, il n'est pas possible de stocker ces connaissances dans l'agent même (l'agent KP ne maintient pas de connaissances figuratives). Un agent KP devra donc utiliser les connaissances stockées au sein d'un agent KS attaché et utiliser ses règles de stratégie ou d'action pour raisonner sur ces connaissances.

Ces mécanismes doivent finalement être replacés dans le contexte du langage de programmation et seront détaillés dans le paragraphe traitant des techniques de programmation avancées (paragraphe III.3).

1.3. LES COMMUNICATIONS

Les messages reçus par les agents sont les mêmes que ceux décrits au chapitre précédent. Cependant, l'introduction du parallélisme confère un potentiel accru à certains d'entre eux. C'est le cas par exemple des messages de manipulation d'instances.

Il convient également de fixer les règles de synchronisation des agents : nous proposons une alternance entre deux modes fondamentaux : le mode synchrone et le mode asynchrone **(Fig. C.5)**. Le mode synchrone définit une relation de type client / serveur entre deux agents, il convient ainsi plus particulièrement aux déclenchements effectués dans le cadre d'une résolution guidée par les buts : requête "Collecter" émise par un agent KP vers un agent KS ou requête "Demander" émise par un agent KS vers un agent KP. Le mode asynchrone, au contraire, définit une communication par boite aux lettres, plus adaptée au traitement des requêtes régissant la transmission de données : requête "Mettre à jour" émise par un agent KP vers un agent KS, ou requête "Envoyer" émise par un agent KS vers un agent KP.

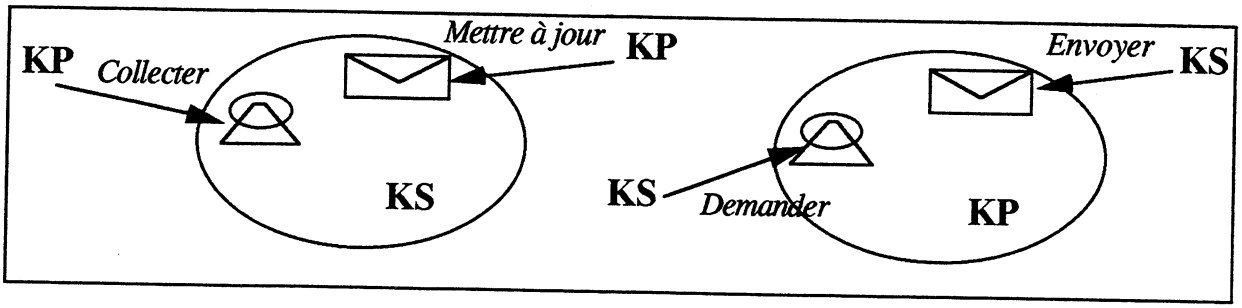


Fig. C.5. Modes de synchronisation entre agents

1.3.1. AGENT KS

L'agent KS fait intervenir des protocoles synchrones (Collecter) et asynchrones (Mettre à jour). Dans le cas d'une requête "Collecter", l'agent KP émetteur attend la réponse de l'agent KS (information demandée ou échec). Dans le cas d'une requête "Mettre à jour", l'agent KP émetteur n'attend pas de confirmation de l'agent KS et peut ainsi poursuivre ses propres opérations internes. Les messages de contrôle qui nécessitent une réponse de l'agent KS sont également traités de manière synchrone. Un rôle particulier sera accordé aux messages de manipulation des instances : celui de la gestion des mondes d'hypothèses. Ces messages permettront en effet de créer de nouveaux mondes d'hypothèse, d'en détruire d'autres et de se positionner sur un monde particulier.

Notons ici que la possibilité pour un agent KS de modéliser des connaissances sur les autres, vont lui permettre de raisonner sur ces connaissances par l'intermédiaire des règles de proposition et d'interrogation. Des protocoles de communication plus complexes pourront être ainsi programmés à l'aide de ces règles.

1.3.2. AGENT KP

L'agent KP fait également intervenir des protocoles synchrones (Demander) et asynchrones (Envoyer). Dans le cas d'une requête "Demander", l'agent KS émetteur attend la réponse de l'agent KP au problème posé (échec ou résultat). Dans le cas d'une requête "Envoyer", l'agent KS émetteur n'attend pas confirmation de l'agent KP et peut donc poursuivre ses propres opérations internes. Les messages de contrôle qui nécessitent une réponse de l'agent KP sont aussi traités de manière synchrone.

De même que pour l'agent KS, des protocoles de communication plus complexes vont pouvoir être programmés avec les règles de stratégie et d'action utilisant la modélisation des connaissances sur les autres.

1.4. LE CONTROLE

Le rôle conféré aux agents KS et KP n'est pas modifié, les modifications portent sur la façon

dont les requêtes vont être traitées. Un message va être interprété et traité en parallèle avec d'autres messages. Les modifications apportées concernent ainsi essentiellement les comportements et les moteurs d'inférence des agents.

1.4.1. AGENT KS

Un agent KS est chargé de recevoir ou de fournir des informations sur les éléments d'un problème en activant les comportements "Recevoir" et "Fournir" suite à la réception des requêtes "Mettre à jour" et "Collecter". Ces deux comportements ont été légèrement modifiés et sont résumés ci-dessous sous une forme algorithmique.

Recevoir

```
(Et     Modifier
        (Ou     Proposer
                (Et     Interroger
                        (Envoyer Self Recevoir)))
```

Fournir

```
Obtenir
```

Dans le cas du comportement "Recevoir", la base de connaissance est tout d'abord mise à jour (Modifier), ce qui comme dans la version précédente, peut mettre en œuvre des vérifications ou propagations internes, avec le déclenchement des règles de propagation. Les règles exploratoires sont alors activées pour sélectionner ou proposer une information valide à un agent KP ou bien, en cas d'échec, pour sélectionner une information manquante à requérir auprès d'un agent KP. Contrairement à la version précédente, l'envoi à l'agent KP n'est pas effectué séparément de la règle (plus de "Envoyer KP Proposition" ou "Envoyer KP Interrogation"), l'interpréteur de règles a ainsi été modifié pour effectuer l'envoi des requêtes lors de l'évaluation de la partie conclusion de la règle. Il n'y a ainsi plus de rupture entre l'évaluation de la règle et l'envoi du message.

Dans le cas du comportement "Fournir", le fonctionnement est le même, sauf que la stratégie de reprise en cas d'échec de l'obtention (Proposition ou Interrogation) n'est pas effectuée. Il n'y a ainsi plus de rupture de la stratégie de résolution en cours.

L'agent KS possède les trois capacités de "Propagation", "Proposition" et d'"Interrogation". Leur rôle est le même que dans la précédente version. Seul le moteur d'inférence a été modifié dans le cas de capacités de "Proposition" et d'"Interrogation". Les nouveaux cycles sont les suivants.

Règles de proposition et d'interrogation

1- Recherche d'un ensemble de conflit parmi l'ensemble des règles.
2- Activation des règles de l'ensemble de conflit.

3- Reprise du cycle jusqu'à l'obtention d'un ensemble de conflit vide.

Au cours des cycles, le moteur garantit qu'une règle n'est exécutée qu'une fois, évitant ainsi les bouclages possibles. Le moteur fonctionne à présent par saturation, il s'arrête lorsque toutes les règles applicables ont été exécutées. Grâce à ce nouveau cycle et à l'introduction du parallélisme, il est donc à présent possible d'envoyer plusieurs requêtes à des agents KP, et donc d'envisager la transmission en parallèle de plusieurs hypothèses.

1.4.2. AGENT KP

Les agents KP ont le même fonctionnement que dans la version précédente. La seule modification introduite concerne la syntaxe et l'interprétation des règles de stratégie et sera donc présentée dans le paragraphe sur le langage de programmation.

2. LA SOCIETE D'AGENTS

La société d'agents va voir son caractère essentiellement modifié par l'introduction du parallélisme. En effet, on retrouve les mêmes possibilités au niveau des groupes "centrés connaissances" et des groupes "centrés tâches". La seule notion nouvelle est celle des mondes multiples due au traitement en parallèle des messages et à l'introduction d'une vraie répartition (distribution des agents sur un réseau de machines).

2.1. DISTRIBUTION DES TACHES ET DES COMPETENCES

Les groupes d'agents peuvent à présent être répartis sur des machines dédiées sur un réseau **(Fig. C.6)**. La distribution en terme de tâches rejoint ici la distribution en terme de ressource opératoire physique. Un groupe "centré tâches" peut en effet être exécuté sur une machine dédiée à des traitements spécifiques, par exemple une machine à base de transputers dédiée au traitement d'image. De même la distribution en terme de connaissances rejoint la distribution physique en terme de base de donnée. Un groupe "centré connaissances" peut en effet être exécuté sur un machine dédiée, par exemple un serveur de données. Cette distribution peut être hétérogène, des éléments d'un groupe pouvant être distribués sur plusieurs machines.

L'introduction des mondes d'hypothèses introduit un autre type de distribution : une distribution dynamique qui correspond à la création dynamique d'"agents" (processus fils créés pour la manipulation de messages en parallèle). Ces processus (ou mondes) partagent alors un ensemble de ressources, qui correspondent au monde commun et possèdent leur propres ressources (instances du monde) **(Fig. C.4)**.

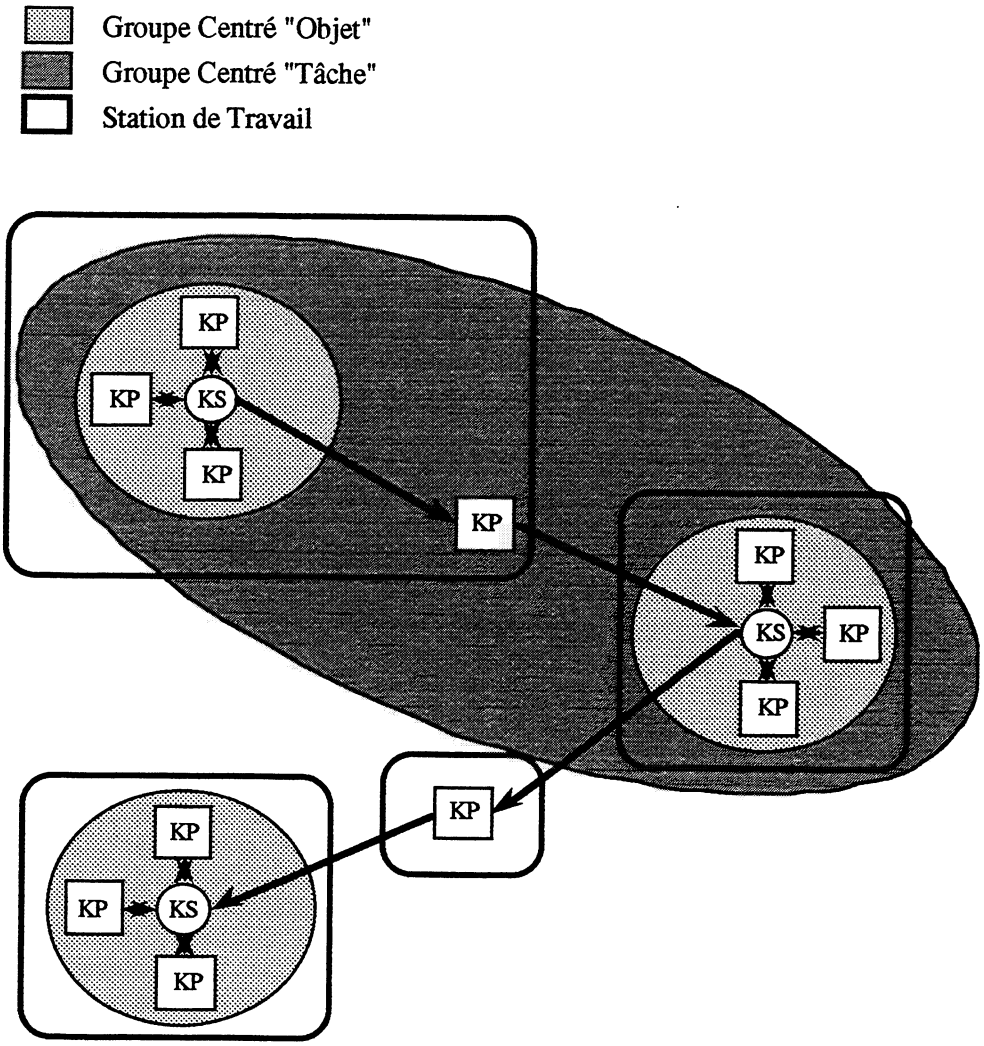


Fig. C.6. Distribution logique et physique des agents

2.2. MODES ET PROTOCOLES DE COMMUNICATION

Le parallélisme va ici introduire une multiplexion des communications : les agents vont en effet s'échanger des messages en parallèle. La création de mondes d'hypothèses multiples va ensuite introduire un autre niveau d'échanges de messages : les agents vont en effet s'échanger des messages concernant des mondes d'hypothèses différents. Les messages ne sont plus liés à un monde de pensée unique, mais à des mondes de pensée multiples, maintenus en parallèle.

2.3. MODES D'ORGANISATION ET DE COOPERATION

Il faut noter ici que si un agent est toujours relié à des agents de façon prédéfinie, la possibilité de modéliser des connaissances sur les autres et de raisonner sur ces connaissances va permettre d'introduire un certain dynamisme dans les relations établies avec ces agents. Un agent va en effet posséder un certain nombre d'accointances, constituées par ces agents prédéfinis qui vont définir l'environnement social de l'agent. Un agent ne pourra établir des relations qu'avec des agents de son environnement social, dont il pourra par exemple modéliser les rôles ou les

compétences. Suite à des raisonnements sur ces rôles et compétences, il pourra alors choisir d'entrer en relation avec tel ou tel agent. Les communications et donc les groupes définis par ces échanges ne sont plus statiques comme dans la version précédente mais surviennent dynamiquement en fonction des connaissances des différents agents.

D'autre part, des groupes d'agents vont pouvoir travailler et coopérer en parallèle. Le problème majeur va être ici la synchronisation de ces groupes. Un groupe peut en effet dépendre des traitements et résultats d'autres groupes. Comme nous l'avons vu dans le chapitre présentant MAPS (chapitre 3, partie A), les mécanismes de synchronisation peuvent être implantés de manière "ad-hoc" par le maintien de variables d'état (attributs d'objets) et l'envoi de messages entre les groupes d'agents concernés. Un approche plus souple serait cependant préférable : dans ce cadre les améliorations structurelles devraient apporter des solutions.

Les mondes multiples vont introduire un aspect nouveau pour la coopération. Le rôle d'un agent, du point de vue social est en effet éclaté. Un agent peut développer un certain rôle dans un monde de pensée (rôle de leader par exemple) et développer un rôle totalement différent dans un monde de pensée parallèle (contraint par un autre groupe par exemple). Un raisonnement plus sophistiqué devra alors être utilisé pour gérer ces mondes.

III LANGAGE DE PROGRAMMATION

Profitant de la refonte de l'environnement de programmation MAPS, le langage de programmation MAPS a été modifié. Ces modifications concernent d'une part la syntaxe du langage, celle-ci est à présent plus proche de celle du langage C++, qui supporte l'implantation des caractéristiques des agents. Les modifications concernent d'autre part l'ajout de nouvelles fonctionnalités, telles que le paramétrage des noms d'agents, d'objets et d'attributs dans les règles. Nous présentons tout d'abord la nouvelle syntaxe du langage, puis un exemple d'utilisation et finalement, les techniques de programmation qui peuvent être introduites grâce aux nouvelles fonctionnalités.

1. SYNTAXE DU LANGAGE

Nous donnons ici les principales caractéristiques du nouveau langage de programmation MAPS. Les extensions concernent la syntaxe, mais surtout de nouvelles fonctionnalités pour la définition des prémisses et conclusions de règles.

1.1. LES AGENTS

Un agent KS ou KP est défini de la manière suivante, les renseignements en italiques étant facultatifs :

AGENT Nom : Type {
Définition des objets, règles et connexions.
}

Un agent possède une identification interne (Nom), utilisée notamment dans les mécanismes de communication. Un agent possède également un type qui est dans notre cas KS ou KP.

1.2. LES OBJETS

Un objet est défini de la manière suivante :

OBJECT Nom *: Père* {
Définition des attributs.
} ;

Un objet possède une identification interne (Nom) et peut éventuellement hériter des attributs d'un autre objet (Père). Des attributs propres à cet objet peuvent en outre être définis.

1.3. LES ATTRIBUTS

Les attributs seront typés selon les quatre types de données (entier, réel, chaînes de caractères et liste) définis dans le langage MAPS. Un attribut sera ainsi défini de la manière suivante :

STRING Nom=*Valeur* ;
INT Nom=*Valeur* ;
FLOAT Nom=*Valeur* ;
LIST Nom=*Valeur* ;

On peut noter ici l'extension apportée par rapport à la précédente version. Le type LIST est un type à part entière, il n'est plus besoin de le simuler par des chaînes de caractères. Il est doté de ses propres opérateurs tels que CAR, CDR, APPEND ou CONS.

1.4. LES CONNEXIONS

Une connexion définit un agent accointance de la manière suivante :

CONNECTION Nom ;

1.5. LES REGLES

Une règle sera définie de la manière suivante :

```
RULE Nom : Type {
IF Expression
THEN Expression
} ;
```

Une règle possède une identification interne (Nom) et un type (Propagation, Proposition, Interrogation, Action et Strategy). La syntaxe des règles du langage MAPS est présentée ci-dessous sous forme BNF (Backus Nohr Form).

```
<Expression>        ::=  <Expression> && <Expression>
                       | <Expression> || <Expression>
                       | ! <Expression>
                       | { <Block> }
                       | <Instruction>
                       | ( <Expression> )
<Block>             ::=  <Instruction> ;
                       | <Block> <Instruction> ;
<Instruction>       ::=  <Case>
                       | <Define-Parameter>
                       | <Affect-Parameter>
                       | <Methode>
                       | <Loop>
                       | <Boolean-Expression>
                       | <Quote>
                       | <Procedure>
                       | <Set>
                       | <Send>
                       | <Ask>
                       | <Search>
<Case>              ::=  CASE ( <Reference> ) <Expression>
                       | CASE ( <Reference> , <Reference> ) <Expression>
<Define-Parameter>  ::=  INT <Name>
                       | FLOAT <Name>
                       | STRING <Name>
                       | LIST <Name>
                       | INT <Name> = <Integer>
                       | FLOAT <Name> = <Float>
                       | LIST <Name> = <List>
                       | STRING <Name> = <String>
                       | INT <Name> = <Get>
                       | FLOAT <Name> = <Get>
                       | LIST <Name> = <Get>
                       | STRING <Name> = <Get>
                       | INT <Name> = <Nom2>
                       | FLOAT <Name> = <Nom2>
                       | LIST <Name> = <Nom2>
                       | STRING <Name> = <Nom2>
                       | INT <Name> = <Agent>
                       | INT <Name> = <Object>
                       | INT <Name> = <Attribute>
                       | INT <Name> = <Search>
```

<Affect-Parameter> ::= <Name> = <Integer>
| <Name> = <Float>
| <Name> = <String>
| <Name> = <List>
| <Name> = <Arithmetical-Expression>
| <Name> = <Get>
| CONS <Name> (<Value-List>)
| APPEND <Name> (<Value-List>)
| <Name> = <Search>
| <Name> = <Name>
| <Name> = <Agent>
| <Name> = <Object>
| <Name> = <Attribute>

<Value-List> ::= <Term-List> <Value-List>
| <Litteral> <Value-List>

<Get> ::= GET (<Reference> , <Reference>)
| GET (<Reference> , <Reference>) IN <Reference>

<Arithmetical-Expression> ::= (<TermA> <Operator> <TermA>)
| (- <TermA>)
| (CAR <Name>)
| (CDR <Name>)

<TermA> ::= <Arithmetical-Expression>
| <Name>
| <Integer>
| <Float>
| <Get>

<Term-List> ::= (CAR <Term-List>)
| (CDR <Term-List>)
| <List>
| <Name>
| <Get>

<Operator> ::= + | - | * | / | %

<Method> ::= MAKE INSTANCE (<Reference>)
| MAKE INSTANCE (<Reference>) IN <Reference>
| LAST PROPOSITION (<Reference>)
| LAST PROPOSITION (<Reference>) IN <Reference>
| LAST INTERROGATION (<Reference>)
| LAST INTERROGATION (<Reference>) IN <Reference>
| LAST RESULT (<Reference>)
| LAST RESULT (<Reference>) IN <Reference>
| RESET CURRENT INSTANCE (<Object-List>)
| RESET CURRENT INSTANCE (<Object-List>)
IN <Reference>
| DELETE ALL INSTANCE (<Object-List>)
| DELETE ALL INSTANCE (<Object-List>) IN <Reference>
| DELETE CURRENT INSTANCE (<Object-List>)
| DELETE CURRENT INSTANCE (<Object-List>)
IN <Reference>

<Object-List> ::= <Reference>
| <Object-List> , <Reference>

<Loop> ::= DO <Expression> WHILE <Expression>
| WHILE <Expression> DO <Expression>
| FOR (<Iteration>) DO <Expression>

```
<Iteration>              ::=  <Iteration-List>
                              | <Iteration-A> ; <Boolean-Expression>
<Iteration-List>         ::=  <Name> ELEMENT <Name>
                              | <Name> ELEMENT <List>
                              | LIST <Name> ELEMENT <Name>
                              | LIST <Name> ELEMENT <List>
<Iteration-A>            ::=  <Name>
                              | <Name> = <Name>
                              | <Name> = <Integer>
                              | INT <Name>
                              | INT <Name> = <Name>
                              | INT <Name> = <Integer>
<Boolean-Expression> ::=  ( <Term> <Comparator> <Term> )
<Term>                   ::=  <Get>
                              | <Litteral>
                              | <Name>
                              | <List>
                              | <Arithmetical-Expression>
                              | <Methode>
<Comparator>             ::=  < | > | <= | >= | == | != | ELEMENT | NELEMENT
<Search>                 ::=  GET INSTANCE OF <Reference> WHERE <Expression>
                              | GET INSTANCE OF <Reference> FROM <Reference>
                                                    WHERE <Expression>
                              | GET INSTANCE OF <Reference> WHOSE ID = <Integer>
                              | GET INSTANCE OF <Reference> FROM <Reference>
                                                    WHOSE ID = <Integer>
                              | GET INSTANCE OF <Reference> WHOSE ID = <Name>
                              | GET INSTANCE OF <Reference> FROM <Reference>
                                                    WHOSE ID = <Integer>
<Quote>                  ::=  ' ( Quote-List )
<Quote-List>             ::=  <Name>
                              | <Quote-List> , <Name>
<Procedure>              ::=  <Name> ( <Parameter-List> )
<Parameter-List>         ::=  <Parameter>
                              <Parameter-List> , <Parameter>
<Parameter>              ::=  <String>
                              | <Integer>
                              | <Float>
                              | <List>
                              | <Name>
                              | <Get>
<Set>                    ::=  SET ( <Reference> , <Reference> ) WITH <Name>
                              | SET ( <Reference> , <Reference> ) WITH <Name>
                                                    IN <Reference>
                              | SET ( <Reference> , <Reference> ) WITH <Litteral>
                              | SET ( <Reference> , <Reference> ) WITH <Litteral>
                                                    IN <Reference>
                              | SET ( <Reference> , <Reference> ) WITH <List>
                              | SET ( <Reference> , <Reference> ) WITH <List>
                                                    IN <Reference>
<Send>                   ::=  SEND ( <Reference> ) TO <Reference>
```

		| SEND (<Reference> , <Reference>) TO <Reference>
<Ask>	::=	ASK (<Reference>) TO <Reference> | ASK (<Reference> , <Reference>) TO <Reference>
<Litteral>	::=	<Integer> | <Float> | <String> | TRUE | NIL
<List>	::=	(<Body-List>)
<Body-List>	::=	<Integer> <Body-List> | <List> <Body-List> | <Foat> <Body-List> | <String> <Body-List> | <List> <Body-List>
<Reference>	::=	USER | <Name> | <Agent> | <Object> | <Attribute>
<Agent>	::=	AGENT : <Name>
<Object>	::=	OBJECT : <Name> : <Name>
<Attribute>	::=	ATTRIBUTE : <Name> : <Name> : <Name>

Plusieurs notions nouvelles sont introduites, qui sont brièvement décrites ci-dessous.

Notion de bloc

La notion de bloc {...} a été introduite pour regrouper des instructions n'ayant aucun lien logique ou temporel entre elles, c'est-à-dire pouvant s'exécuter en parallèle.

Paramétrage

Il est à présent possible de paramétrer les noms d'agents, d'objets ou d'attributs dans les règles des agents. L'instruction de mise à jour d'un attribut d'objet pourra s'écrire :

SET(Object:Image,Attribute:Nom) WITH "muscle" IN Agent:A1

ou bien en utilisant des paramètres:

SET(Im,Nm) WITH "muscle" IN A

où Im, Nm et A sont des paramètres définis comme suit :

INT Im=Object:A1:Image INT Nm=Attribute:A1:Image:Nom et INT A=Agent:A1

Cette notion de paramétrage doit être replacée dans le contexte de la modélisation des connaissances sur les autres et sera illustrée dans le paragraphe traitant des techniques de programmation avancées.

Stratégies

Les règles de stratégie des agents KP ne concluent plus sur un mot clé devant être présent dans les règles d'actions comme dans la version précédente mais directement sur un nom de règle d'action. Une règle de stratégie pourra par exemple s'écrire :

```
RULE S1 : Strategy {
IF CASE (Agent:A1,Object:Image)
        Get(Object:Image,Attribute:Loaded)==TRUE
THEN '(R1) };
```

où R1 est une règle d'action définie comme suit :

```
RULE R1 : Action {
IF ...
THEN ...
};
```

Proposition et Interrogation

Les règles de proposition et d'interrogation d'un agent KS contiennent à présent une instruction d'envoi direct aux agents KP, respectivement Send et Ask. Il n'y a ainsi plus de rupture entre la règle de proposition ou d'interrogation et l'envoi du message à l'agent KP comme dans la version précédente. Un règle de proposition s'écrira ainsi :

```
RULE P1 : Proposition {
IF LAST PROPOSITION (Object:Image)=="NO"
THEN Send(Object:Image) To Agent:KP2
};
```

2. EXEMPLE D'UTILISATION

La nouvelle syntaxe est plus lisible et plus proche du code cible (C++). L'exemple 1 permet de comparer les deux syntaxes pour la définition d'un agent KS.

Nouvelle syntaxe :

```
AGENT Example : KS {

OBJECT Image {
STRING Name="file1.ima";
INT Dim=128;
INT Num=1;
INT Loaded=0;
};

RULE Load_File : PROPAGATION {
IF (GET(object:Image,attribute:Loaded)==0) &&
        {
        STRING N=GET(object:Image,attribute:Name);
        INT D=GET(object:Image,attribute:Dim);
        INT Nb=GET(object:Image,attribute:Num)
        }
```

Ancienne syntaxe :

```
DEFINE_AGENT Exemple:Ex:10:20;

DEFINE_CLASS Image : ;
STRING Name : "file1.ima";
INTEGER Dim : 128;
INTEGER Num : 1;
INTEGER Loaded : 0;
END_CLASS;

DEFINE_RULE Load_File : PROPAGATION;
(AND ((Image Loaded) (INTEGER 0))
(AND (DEFINE_PARAMETER
                (Image Name) (N))
(AND (DEFINE_PARAMETER
                (Image Dim) (D))
    (DEFINE_PARAMETER
```

```
THEN Load_File(N,D,Nb) &&
        SET(object:Image,attribute:Loaded) with 1
};
        ......
}
```

```
                                                    (Image Num) (Nb)))));
(AND (PROCEDURE (Load_File) (N D Nb))
    (Image Loaded Num));
END_RULE;
END_AGENT;
```

Exemple 1 : Comparaison des deux syntaxes du langage MAPS

3. TECHNIQUES AVANCEES DE PROGRAMMATION.

Parmi les nouvelles caractéristiques et fonctionnalités du langage, le paramétrage et l'envoi de messages directs aux agents KP dans les règles d'interrogation et de proposition, introduisent de nouvelles facilités de programmation. Il est ainsi possible de définir des modèles des autres agents, des modèles de groupes et d'introduire des stratégies de prédiction / vérification.

Paramétrage

La notion de paramètre va introduire un niveau d'abstraction supplémentaire dans le langage, en autorisant des raisonnements sur ces paramètres. Une plus grande généricité des règles est alors également possible.

Modélisation du savoir sur les autres.

Comme nous l'avons déjà signalé, les connaissances sur les autres peuvent être modélisées sous la forme d'attributs d'objets, ces attributs étant utilisés par la suite dans les règles des agents. Par exemple, la notion de groupe d'agents peut être modélisée sous la forme d'une liste stockée dans un attribut d'objet. Cette liste peut être initialisée avec certains agents et peut également être mise à jour en cours d'exécution. Pour envoyer une information à un groupe, il suffira d'utiliser cet attribut. Pour une règle de proposition d'un agent KS, nous aurons la syntaxe suivante :

```
OBJECT Groupe {
LIST Liste_Agent = (Agent:KP1 Agent:KP2 Agent:KP3);
};
OBJECT Image {
...
};
RULE P1 : Proposition {
IF      LAST PROPOSITION (Object:Image)=="NO" &&
        {LIST Liste=Get (Object:Groupe,Attribute:Liste_Agent);}
THEN For (I Element Liste) Send (Object:Image) To I
};
```

Ce mécanisme peut également servir à modéliser le rôle d'agents :

```
OBJECT RoleOfKP {
STRING Role;
INT AgentKP;
```

```
};
RULE Proposition {
IF      Get Instance Of (Object:RoleOfKP) Where
              Get(Object:RoleOfKP,Attribute:Role)=="ROLE1" &&
        {INT Agent=Get(Object:RoleOfKP,Attribute:AgentKP);}
THEN Send (Object:Image) To Agent
};
```

Les instances de l'objet RoleOfKP modélisent le rôle des agents KP accointances. Une règle de proposition pourra donc rechercher l'agent ayant un rôle spécifique (Get Instance), récupérer l'identificateur de l'agent (Get) et lui transmettre une information (Send).

Prédiction / Vérification

La prédiction / vérification peut être définie localement à un agent KS ou à un agent KP. Dans le premier cas, elle est modélisée sous une forme plutôt figurative alors que dans le second cas, elle est modélisée sous une forme plutôt opératoire.

Prédiction / Vérification contrôlée par un agent KS

L'exemple suivant illustre la programmation d'une stratégie de prédiction / vérification dans un agent KS.

```
AGENT KS1 : KS {
OBJECT Cellule {              // Un objet cellule
INT Confiance=0;              // Le coefficient de confiance en cette cellule
INT Noyau=0;                  // L'indice de l'instance de noyau contenu dans la cellule
};
OBJECT Noyau {                // Un objet noyau
INT Confiance=0;              // Le coefficient de confiance en ce noyau
INT Cellule=0;                // L'indice de l'instance de cellule contenant ce noyau
};
RULE P1 : Propagation {
IF      Get(Object:Noyau,Attribute:Confiance)<0.4 &&
        {INT Cell=Get(Object:Noyau,Attribute:Cellule);} &&
        Get Instance Of (Object:Cellule) Whose Id = Cell
THEN Delete Current Instance(Object:Cellule,Object:Noyau)
};
RULE I1 : Interrogation {
IF      {INT Cell=Get Instance Of (Object:Cellule)
              Where {Get(Object:Cellule,Attribute:Confiance)<0.4;} &&
        {INT Noy=Make Instance (Object:Noyau);}          //Création d'une instance de noyau
THEN Set(Object:Cellule,Attribute:Noyau) With Noy &&     // Mise à jour de l'indice de noyau
        Set(Object:Noyau,Attribute:Cellule) With Cell &&  // Mise à jour de l'indice de cellule
        Ask(Object:Noyau,Attribute:Confiance) To KP1      // On demande la confiance
};
}
AGENT KP1 : KP {
```

```
RULE A1 : Action {
IF      Get(Object:Noyau,Attribute:Confiance) In Agent:KS1==0 &&
        {INT Conf= "Appel d'une fonction de calcul de confiance";}
THEN Set(Object:Noyau,Attribute:Confiance) With Conf In Agent:KS1
};
}
```

L'agent KS1 maintient des instances des objets Cellule et Noyau : les objets Cellule contiennent un objet Noyau. Cet agent est par exemple impliqué dans une chaîne d'interprétation d'images biomédicales. A chaque objet est attaché un coefficient de confiance. La règle d'interrogation introduit une stratégie de prédiction / vérification. S'il existe une instance de cellule dont le coefficient de confiance est faible (Get Instance), alors une instance de noyau est créée (Prédiction) et divers attributs des instances sont mis à jour (Make Instance et Set). En conclusion, le coefficient de confiance du noyau est demandé à un agent KP (Ask). Cet agent va calculer ce coefficient d'une certaine manière (par exemple par la recherche d'une région possédant les caractéristiques d'un noyau à une position donnée), et le mettre à jour dans l'agent KS (Set). Cet agent peut alors par le biais d'une règle de propagation, détruire les instances de noyau et de cellule si le coefficient de confiance du noyau est trop faible, ou au contraire augmenter la confiance en la cellule.

Prédiction / Vérification contrôlée par un agent KP

Un agent KP peut également introduire un mécanisme de prédiction / vérification local par le biais de ses règles de stratégie. Une règle de stratégie peut en effet prédire que certaines règles d'action sont activables dans un certain contexte et les sélectionner, la vérification s'effectue alors si l'une des règles sélectionnée est réellement activable :

```
RULE S1 : Strategy {
IF Contexte
THEN '(R1)
};
RULE R1 : Action {
IF A && B && C
THEN H1
};
```

Dans cet exemple, si le contexte de la règle de stratégie est vrai, alors il y a prédiction de l'hypothèse H1 par l'activation simultanée de la règle d'action R1 (la règle R1 est une règle d'inférence qui conclue sur l'hypothèse H1). La règle R1 implique le test de A, B et C en prémisse, qui permettra effectivement de conclure sur l'hypothèse H1 (vérification). Le test de A, B et C nécessite de connaître leur valeur, ce qui peut entraîner des stratégies de recherche sophistiquées au sein d'agents (envoi du message Collecter à un agent KS, suivi d'une stratégie de broadcast au sein de cet agent par exemple).

IV IMPLANTATION

Le passage à une nouvelle version de l'environnement de programmation MAPS s'est doublé d'un portage sur une nouvelle machine cible : une station de travail SUN 4 (SPARC II). Le langage retenu pour l'implantation de l'environnement est toujours le langage C++. Les fonctions système utilisées proviennent du système d'exploitation SUN OS 4.1 et sont largement compatibles UNIX Système V. L'interface utilisateur repose sur les standards X Window et Open Look pour l'apparence graphique. Une version fondée sur Motif est également en cours d'implantation. Compte tenu des standards retenus en matière d'outils de programmation, l'environnement MAPS est donc portable sur la plupart des stations de travail actuelles et peut donc fonctionner dans des réseaux hétérogènes. Nous allons dans les paragraphes suivants, décrire plus en détail les choix d'implantation et les fonctionnalités de l'environnement de programmation.

1. LES AGENTS

La hiérarchie de classes C++ implantant les caractéristiques des agents KS et KP, présentée dans le chapitre 1 (paragraphe B.IV.1) a été légèrement remaniée **(Fig. C.7)**.

Un agent est instancié à partir des classes génériques KS et KP, et d'un fichier contenant les ressources compilées (voir le paragraphe IV.2 sur le contrôleur d'application). Chaque agent est un processus UNIX autonome en attente de messages. Ces messages peuvent provenir des autres agents ou du contrôleur : le protocole de communication inter-agent est fondé sur les "sockets". Actuellement, les messages sont traités un par un (dans une extension en cours de réalisation, les messages seront effectivement traités en parallèle) et les messages en attente sont stockés dans une file.

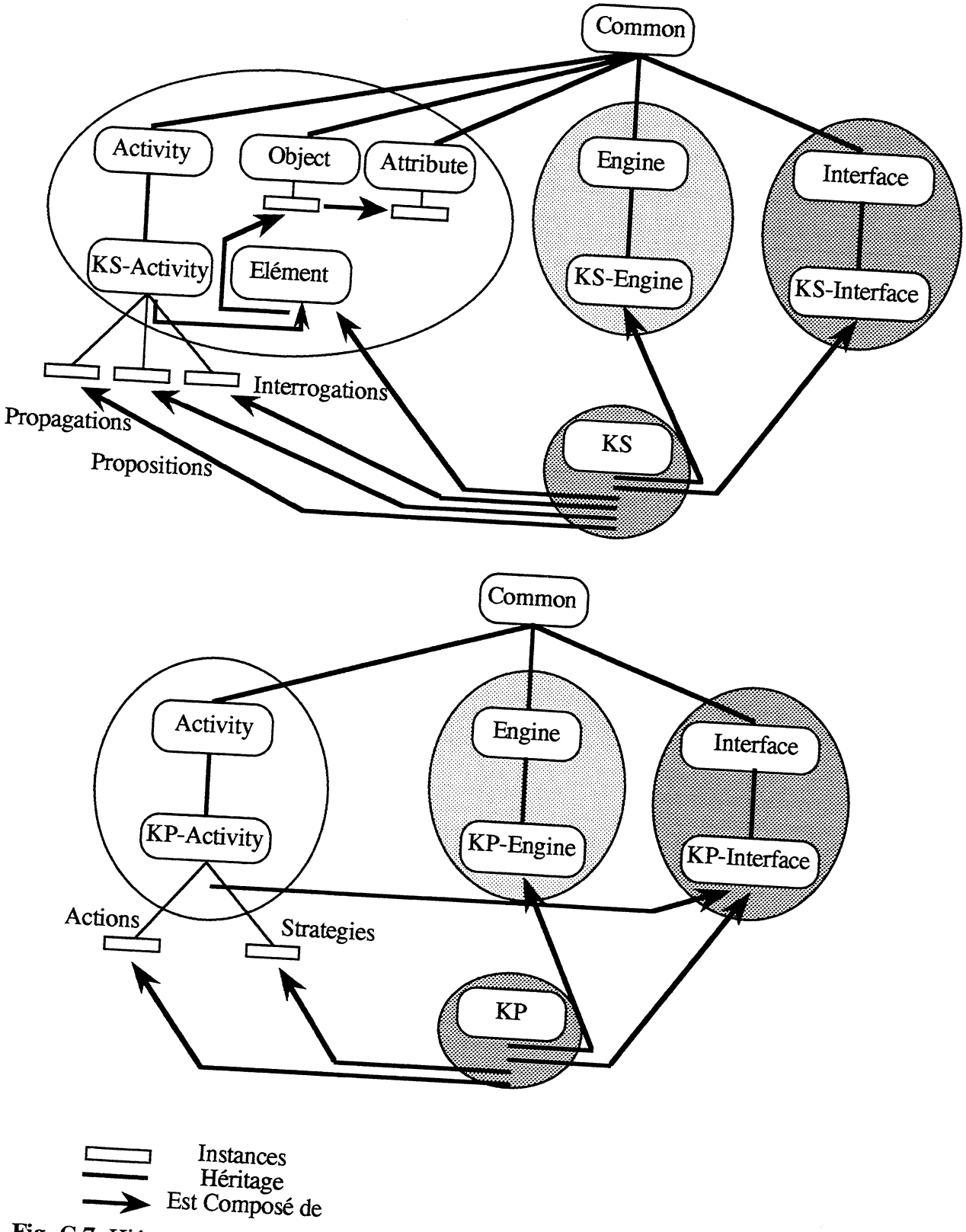


Fig. C.7. Hiérarchie des classes C++ définissant les classes génériques KS et KP.

Les classes chargées des protocoles de communication (Interface, KS-Interface et KP-Interface) ont été redéfinies pour introduire le parallélisme. La construction des classes génériques KS et KP se fait à présent uniquement par un mécanisme de composition (plus d'héritage comme dans

la version précédente). Les classes KS et KP sont donc construites à partir de "briques de base", qu'il est facile de modifier ou de changer sans altérer leur structure.

2. LE CONTROLEUR D'APPLICATION

L'environnement de programmation MAPS est constitué d'un processus principal, nommé Contrôleur d'Application, de plusieurs processus correspondants aux agents de l'application et de serveurs de procédures externes **(Fig. C.8)**. Le contrôleur d'application est chargé de compiler les fichier de ressources des agents d'une application, d'exécuter ces agents sur un réseau de stations de travail, et de faire le lien entre l'utilisateur et ces agents.

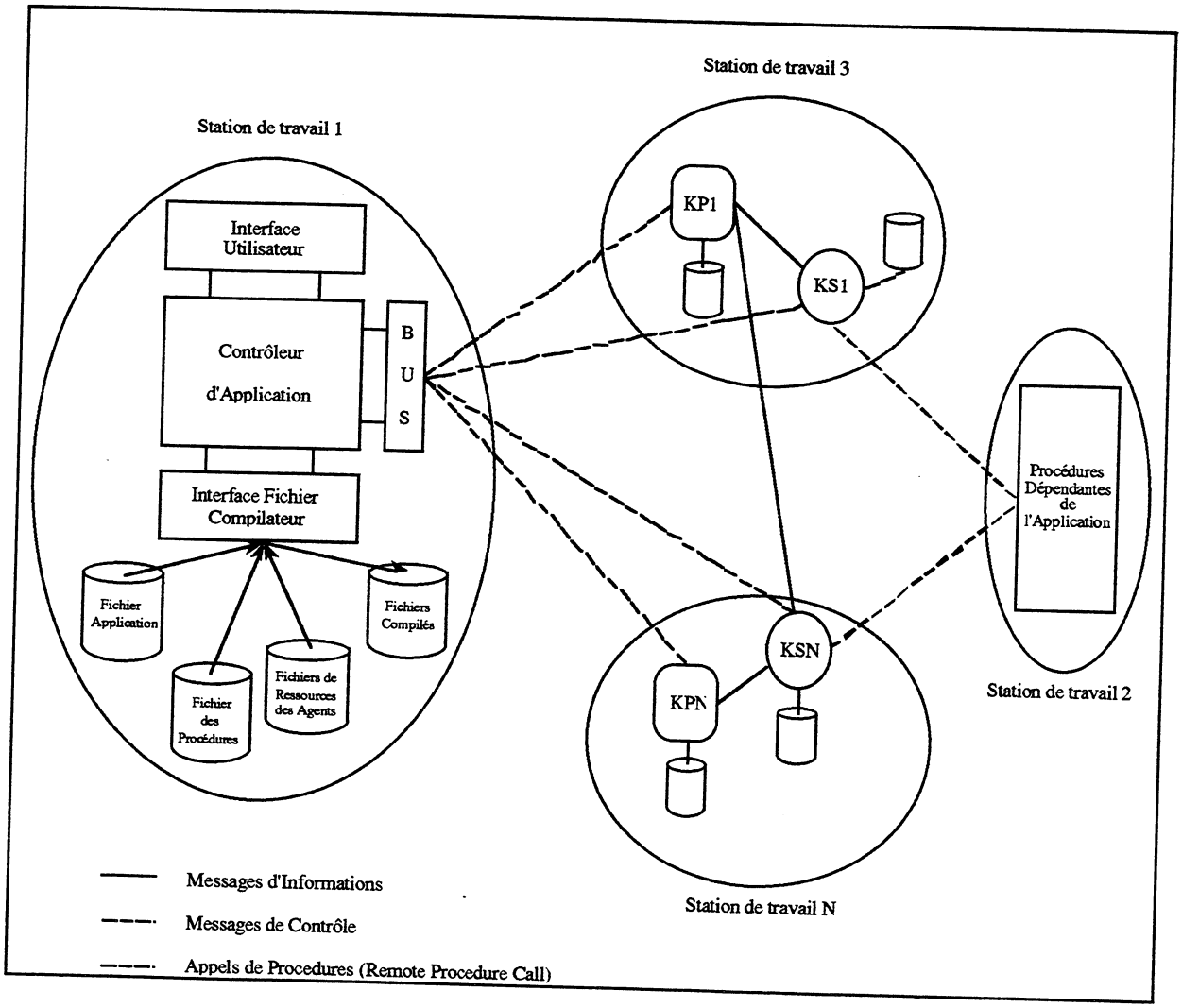


Fig. C.8. MAPS : Vue globale de l'architecture.

Comme dans la version précédente, chaque agent est défini par un fichier de ressources à l'aide du langage de programmation MAPS. Le contrôleur est chargé de compiler ces fichiers de ressources afin d'obtenir une version exécutable des agents. A cette fin, un compilateur a été implanté à ce niveau, à l'aide des outils d'aide à la génération de compilateur Lex++ et Yacc++ (version étendue pour la programmation C++ des outils Lex et Yacc) **[Bruno 91]**. Cette étape

de compilation utilise donc les fichiers de ressources de agents et ainsi qu'un fichier de ressource où sont spécifiées les procédures externes applicables depuis une règle et l'adresse machine du serveur de procédure les maintenant. A la fin de cette phase de compilation, des fichiers de ressources compilés sont obtenus et sont directement utilisables pour créer les processus "agent" de l'application.

Le contrôleur est également chargé d'exécuter une application. Pour cela, il va créer les processus correspondant aux agents de l'application sur un réseau. Un fichier de ressource est utilisé à cette fin, où sont spécifiées pour chaque agent la station de travail sur laquelle l'exécuter et les adresses des stations de travail de ses accointances. Un lien de communication entre le contrôleur et chaque agent est également défini par le biais du mécanisme des sockets, qui permet l'échange de messages de contrôle.

Le contrôleur est finalement chargé de faire le lien entre les agents d'une application et l'utilisateur. A cette fin, une interface utilisateur a été introduite.

3. L'INTERFACE

Une nouvelle interface utilisateur a été définie, qui reprend les principales fonctionnalités de l'interface définie dans la précédente version. Un ensemble de menus permet d'accéder à des éditeurs de texte (création des fichiers de ressource), aux fonctions de compilation d'une application et aux fonctions d'exécution d'une application (création des processus "agent" sur le réseau). Une fenêtre principale permet de visualiser le réseau d'agents d'une application **(Fig. C.9)**.

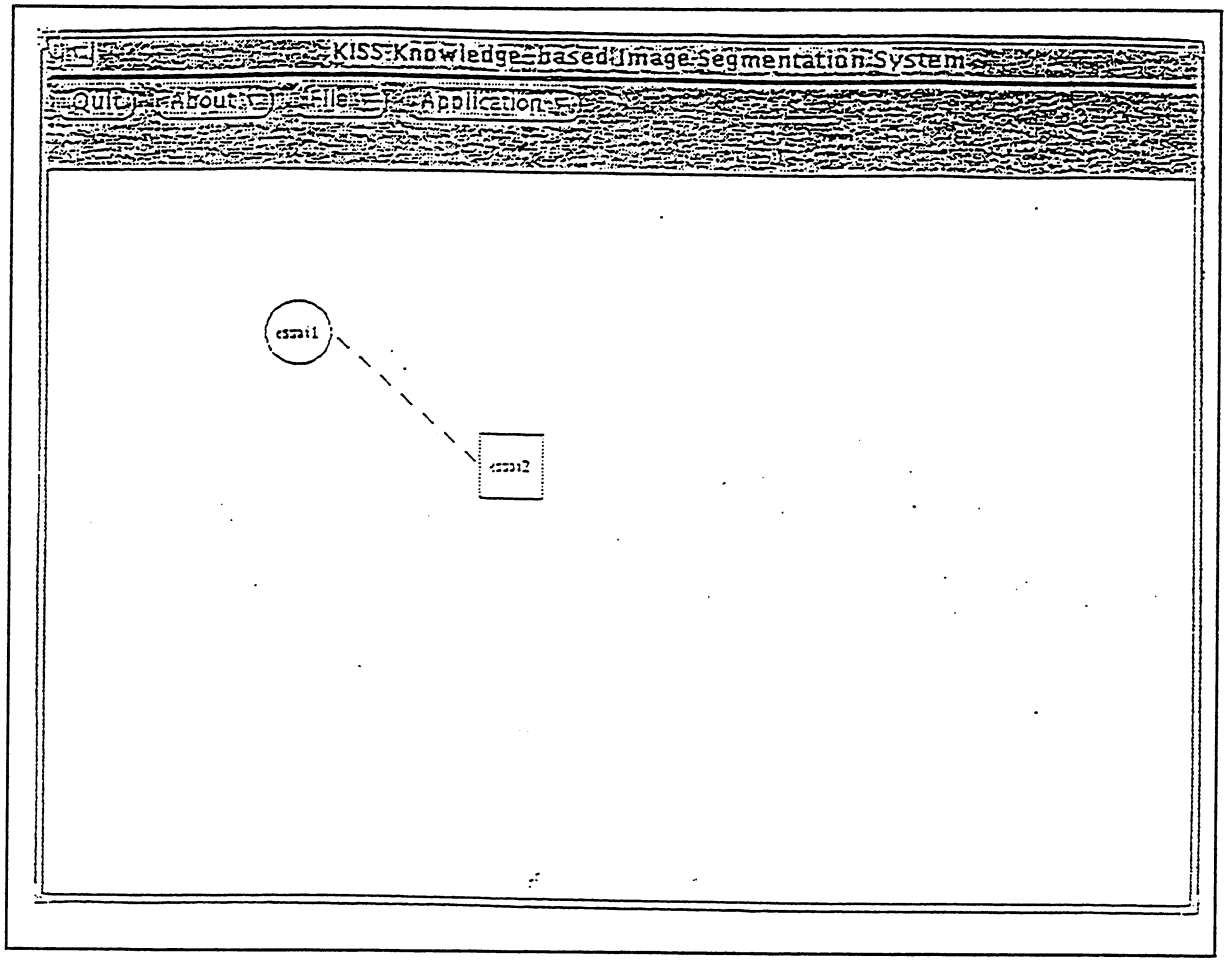


Fig. C.9. MAPS : Vue générale de l'interface.

Sélectionner un agent sur ce réseau à l'aide de la souris permet d'accéder à un panneau de contrôle **(Fig. C.10)** à partir duquel le contenu de l'agent peut être visualisé (connexions, objets, attributs et instances). Des instances peuvent être créées et les valeurs des attributs peuvent être mises à jour. L'agent peut également être activé par l'envoi d'un message d'information (Collecter, Mettre à jour, Envoyer ou Demander). Toutes ces opérations s'effectuent par le biais d'envois de messages d'information ou de contrôle aux agents depuis le contrôleur d'application. Contrairement à la version précédente où une seul panneau de contrôle était affiché, cette version autorise l'affichage d'un panneau de contrôle par agent, permettant ainsi de visualiser en même temps le contenu de plusieurs agents.

```
AGENT Resultat : KS_Graphique {

OBJECT Im1 : Image {
STRING Name="res.ima";
INT Dim=128;
INT Num=1;
};

RULE Affiche : PROPAGATION {
IF GET (Object:Im1,Attribute:Show)==NIL
THEN SET(Object:Im1,Attribute:Show) WITH TRUE
};
...
}
```

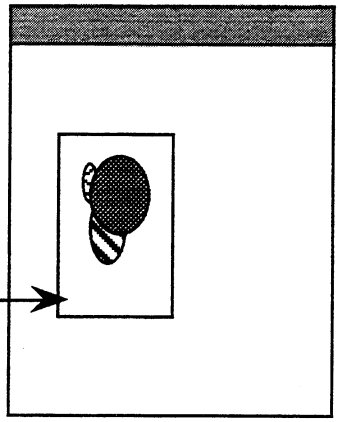


Fig. C.11. Exemple d'agents graphiques. L'attribut Show impliqué dans la règle de propagation est un attribut système propre aux objets de type Image. La mise à jour de cet attribut avec la valeur Show entraine l'affichage de l'image (comportement propre à l'objet).

Ces agents graphiques peuvent être impliqués dans les architectures habituelles, et donc comporter des objets et des règles n'ayant aucun rôle graphique. Il est également possible de définir des agents purement graphiques, ces agents sont alors des agents de présentation de résultats. Il est alors possible de définir des architectures proches du concept Présentation / Abstraction / Contrôle (PAC) **[Coutaz 88]** ou du concept Modèle / Présentation / Contrôleur de Smalltalk. Dans ces architectures, à chaque agent KS, lié à une abstraction, correspond un agent KS de présentation : un agent KP joue alors le rôle du contrôleur et permet passer de l'abstraction à la présentation et inversement. De telles architectures sont actuellement à l'étude dans une application d'acquisition des connaissances **[Hugonnard 91]**.

4. Le Serveur de Procedures

Des procédures externes peuvent être appelées depuis une règle d'un agent (**Fig. C.8**). Ces procédures sont maintenues par un ou plusieurs serveurs de procédures, et peuvent être écrites avec n'importe quel langage de haut niveau tel que Fortran, Pascal ou C. Ces processus comme décrit dans le chapitre précédent (paragraphe B.IV.4) peuvent être personnalisés selon le domaine du problème à étudier (vision par ordinateur, traitement du signal ...). Ces serveurs peuvent être exécutés sur n'importe qu'elle station de travail du réseau, la communication entre les agents et ces serveurs utilise alors le mécanisme d'appel de procédure à distance (Remote Procedure Call) du système d'exploitation SUN OS 4.1. Un fichier de ressource utilisé par le contrôleur d'application lors de la phase de compilation, contient le nom de toutes les procédures maintenues dans ces serveurs et pouvant être appelées depuis une règle, ainsi que l'adresse machine du serveur dont elles dépendent.

Contrairement au serveur de procédure de la version précédente, ces serveurs ne disposent pas

d'interfaces graphiques. La définition d'agents graphiques rend de telles interfaces obsolètes et est en outre beaucoup plus en harmonie avec la programmation multi-agents. Ils permettent d'afficher les résultats selon une forme adaptée au contexte dans lequel l'appel s'est fait.

V CONCLUSION

Dans ce chapitre, nous avons présenté les modifications apportées à l'environnement de programmation MAPS. Ces améliorations fonctionnelles concernent l'introduction d'un mode d'exécution parallèle des agents. Un nouveau langage de programmation plus performant a été défini et constitue une ouverture vers le langage C++. Enfin, au niveau de l'environnement de programmation, de nouvelles fonctionnalités globales telles l'introduction d'agents et d'objets graphiques pour la visualisation des résultats, permettent de faciliter la conception des applications. Les définitions de base se prêtant facilement aux transformations introduites, le concept initial de MAPS (distinction entre connaissance et tâche) semble ainsi confirmé. Des modifications minimes ont permis l'implantation d'un mode d'exécution parallèle des agents : des modifications plus complexes devront cependant être introduites pour permettre un raisonnement sophistiqué au niveau des mondes multiples. Le tableau **Table 1** résume les principales améliorations envisagées et l'état d'implantation.

Améliorations	Etat d'avancement
parallélisme inter-agent	implanté
langage	implanté
plate-forme	implantée
parallélisme intra-agent	non implanté
mondes multiples	non implanté
agents graphiques	en cours d'implantation

Table 1. Etat d'avancement de l'implantation des améliorations.

Une application en Compréhension de la Parole est en cours de réalisation avec cette nouvelle plate-forme. Le portage des applications existantes demandera peu d'effort (réécriture), la nouvelle syntaxe du langage étant dans l'esprit assez proche de la précédente. Il sera par contre peut-être plus délicat d'introduire les nouvelles possibilités de programmation telles que le paramétrage et la prédiction / vérification.

CHAPITRE 4

PERSPECTIVES

I INTRODUCTION

Nos applications ont défini un certain nombre de besoins d'ordre fonctionnel et structurel, ces besoins ayant également une incidence au niveau du langage de programmation (partie C, chapitre 1). La version répartie de MAPS a permis de répondre aux besoins d'ordre fonctionnel. Nous avons ainsi fait évoluer MAPS vers une architecture parallèle et répartie où les agents sont à présent plus autonomes (asynchronisme des communications, intégration de connaissances sur les autres et de raisonnements sur ces connaissances). Il est nécessaire à présent d'aborder des besoins d'ordre plus structurel. Ils concernent d'une part l'introduction d'agents de granularités différentes (du réactif au cognitif) et d'autre part la structuration de ces agents en terme de contrôle, par la définition d'architectures hétérogènes et multi-couches. Nous présentons tout d'abord un état de l'art sur les systèmes réactifs et les architectures hétérogènes et multi-couches puis nous présentons quelques propositions pour une évolution de MAPS vers de telles architectures. Ces réflexions sont d'ordre prospectif mais connaissent cependant de premiers développements, notamment en ce qui concerne l'introduction d'une structuration en couches **[Chiron 92]**.

II ETAT DE L'ART

Nous présentons tout d'abord les systèmes réactifs du point de vue de leur structure interne puis les architectures hétérogènes et multi-couches.

1. MODELES REACTIFS

L'étude des systèmes réactifs doit être replacée dans le contexte de la vie artificielle : le concept clé de ce domaine est l'émergence de comportements **[Langton 88]**. Dans les systèmes naturels, un comportement global et intelligent émerge en effet d'un grand nombre d'interactions entre des entités autonomes, sans intervention d'un contrôleur global. Cette approche est ainsi ascendante, distribuée et locale. Les principales caractéristiques de la vie artificielle sont alors les suivantes : un système est conçu comme une population de programmes simples, aucun programme ne dirige toutes les autres, chaque programme décrit la façon selon laquelle une entité réagit à des situations locales dans son environnement, il n'existe

aucune règle qui édicte un comportement global et tout comportement plus élevé que ceux des programmes individuels est émergent.

Par analogie, les organisations fondées sur des agents réactifs sont plutôt centrées comportement que centrées connaissance **[Bouron 91]**. L'approche réactive s'oppose en effet au postulat que l'intelligence d'un agent lui est conférée par la vertu d'une architecture représentationaliste, c'est-à-dire par la représentation explicite de connaissances **[Seel 91]**. Elle explore au contraire des architectures de type stimulus-réponse, dans lesquelles l'intentionnalité de l'agent n'est pas représentée de manière explicite, ni présente de manière structurelle, mais plutôt dépendante des situations dans lesquelles se trouve placé l'agent. Chaque agent est une entité autonome dotée d'un comportement très simple et placée dans un environnement donné, sans représentation explicite de cet environnement et sans historique de ses actions.

Les approches réactives sont utilisées dans les domaines de la vie artificielle, de l'éthologie **[Drogoul 92]** et de la robotique **[Brooks 86]**, **[Steels 90]**. Parmi les métaphores utilisées pour caractériser ces approches, citons l'analogie entre agents et circuits logiques asynchrones, systèmes et éco-systèmes, organisations et auto-organisations.

1.1. Representation de l'Agent

L'agent réactif pour Seel **[Seel 91]** est un objet, décrit par un nom, un état interne privé, un état externe visible des autres agents et une fonction de transition d'état. Cette fonction calcule un nouvel état pour l'agent en fonction de l'état courant, des noms et des états externes des autres agents. Un agent réactif peut être ainsi modélisé à l'aide d'un simple objet doté d'un moyen de communication et d'un comportement très simple. C'est par exemple le cas des agents définis par **[Ferber 91a]**, **[Drogoul 92]** et **[Bouron 91]** qui sont fondés sur des extensions du langage d'acteur Actalk. Ils n'ont donc pas la capacité de raisonner sur les messages qu'ils reçoivent, ou de développer des stratégies de contrôle. Le comportement est alors binaire et l'agent s'apparente à un automate d'état fini.

1.2. L'Agent et son Environnement

Un agent réactif coexiste avec d'autres agents dans un environnement, environnement social (les autres agents) ou physique (le milieu dans lequel il est plongé) et son comportement est étroitement lié à ce qu'il perçoit de cet environnement. Contrairement aux approches cognitives, les connaissances sur les autres s'expriment plus par des relations de dépendance entre agents, que par une représentation explicite des compétences et des rôles des autres agents. Pour Drogoul, **[Drogoul 92]** ou Ferber **[Ferber 91a]** un agent possède ainsi un autre agent comme but et accointance. Ce mécanisme définit alors une relation de type maître / esclave entre

un agent et ses dépendances ; les dépendances voient leur satisfaction seulement après celle de leur but : celui-ci les en informera. L'agent possède ainsi une connaissance locale de son état de satisfaction, de ses dépendances et de ses "geôliers" (agents qui l'empêchent d'agir). L'agent ne dispose donc que d'une connaissance locale, et pas d'une connaissance de l'état global du monde.

En ce qui concerne les représentations de l'environnement physique, elles ne sont pas logiques mais plutôt analogiques, parce qu'elles conservent une partie de ce qu'elles représentent sous une forme implicite **[Steels 90]** ; il n'y a pas de bases de faits centralisées, ni de réseaux sémantiques ou de frames. Au contraire, les représentations utilisées sont plus proches du monde lui-même. Par exemple pour représenter des relations dans l'espace, une approche symbolique utiliserait des prédicats tels que "à gauche de" ou "position de". Ici au contraire, on utilise une grille où les objets occupent une position qui reflète les positions qu'ils occupent dans le monde. Ces représentations sont alors plus proches de ce que les senseurs peuvent capter et requièrent peu ou pas de catégorisation du monde. Il y a ainsi une connexion directe entre le monde et sa représentation, entre la représentation et l'action.

1.3. COMMUNICATION ET PROTOCOLES

Deux types de communication peuvent intervenir : une communication directe par envoi de messages, et une communication indirecte par perception d'événements survenant dans l'environnement.

En ce qui concerne la communication directe, Drogoul **[Drogoul 92]** utilise le modèle de continuation d'Agha (dans ce modèle, l'agent à qui envoyer une réponse est fourni dans le message). Les agents communiquent alors par message de fuite ou d'agression. Ils peuvent aussi être avertis des modifications de l'environnement par l'envoi de messages. Signalons que l'environnement constitue une ressource partagée, dont l'accès doit s'effectuer en exclusion mutuelle.

Pour Steels **[Steels 90]**, au contraire, les agents communiquent de manière indirecte en laissant des traces dans l'environnement. Ces traces sont perçues comme des événements qui déclenchent des comportements particuliers, par exemple celui de se diriger vers elles (effet d'attracteur vers les zones intéressantes au sens de la résolution du problème). Par la notion de trace, les agents ont indirectement conscience des besoins des autres, et donc une connaissance indirecte des autres.

Cette notion d'attraction a été formalisée par Kephart **[Kephart 89]**, qui propose un modèle selon lequel l'agent est libre de choisir parmi plusieurs ressources selon leur intérêt perçu. Il

considère que l'intérêt d'une ressource dépend du nombre d'agents qui l'utilisent déjà, et par ailleurs, que l'intérêt perçu peut être différent de l'intérêt effectif.

1.4. COMPORTEMENT

Le comportement de base de l'agent réactif est régi par un cycle simple de type stimulus / réponse et l'analogie est souvent fait avec les automates d'état fini. Dans un système réactif, un comportement global naît des interactions de ces entités au comportement simple. Les approches de type éco-résolution ou auto-organisation étudient dans ce cadre la résolution d'un problème ou l'obtention d'un comportement global particulier.

Comportement de type stimulus / réponse

Dans le cas des approches de type stimulus / réponse, le comportement de l'agent est régi par une fonction de type transition d'état qui en fonction des entrées (ce que l'agent perçoit) et de l'état courant de l'agent calcule un nouvel état. Cette approche est par exemple utilisée par **[Bouron 91]** dans le système MAGES. Les agents réagissent aux messages en utilisant un modèle de type stimulus-réponse, en déclenchant les actions selon leur état interne, leurs croyances et leurs règles. Un comportement réactif de base est décrit par un cycle de comportement, exécuté à chaque activation du processus associé. Ce cycle est ici entièrement défini en termes d'acceptation de messages, ce qui se traduit par un simple comportement réactif aux entrées.

Eco-résolution et Auto-organisation

Pour Drogoul **[Drogoul 92]** ou Ferber **[Ferber 91a]** la résolution de problème est vue comme la production d'états stables dans un système dynamique, dont l'évolution dépend des comportements des agents. Le comportement de l'agent est très simple, de type satisfaction ou fuite ; il est régit par trois règles de base : le désir d'être satisfait, le désir d'être libre et l'obligation de fuir. Le paradigme sous-jacent est que la résolution du problème émerge d'interactions locales (paradigme de l'éco-résolution).

Steels **[Steels 90]** explore spécifiquement comment une fonctionnalité ou un comportement global émerge de l'auto-organisation d'agents. Il se place dans le contexte d'un système dynamique, capable d'atteindre un état d'équilibre, et d'en sortir en cas de perturbation extérieure (le système est donc ouvert). La structure se forme elle-même en réponse à la perturbation.

1.5. DISCUSSION

Si l'on examine un agent réactif sous l'angle de sa structure interne, on peut le considérer comme un agent autonome dégradé, c'est-à-dire réduit à son statut d'objet. En effet, nous avons vu qu'un agent réactif pouvait être vu comme un agent sans capacité de raisonnement sur ses perceptions du monde extérieur. Il définit ainsi un comportement élémentaire et "générique" susceptible d'être étendu par rajout de compétences vers un comportement de type cognitif.

Néanmoins, les différences entre ces deux approches ne se réduisent pas à de simples différences structurelles, car elles sous-tendent des problématiques qui leur sont propres. En effet, si l'agent cognitif possède une autonomie de décision réelle, il développe toutefois une dépendance sociale forte, ses décisions étant fortement dépendantes des connaissances qu'il possède sur les autres. L'agent réactif au contraire se caractérise par une dépendance à l'environnement, les sociétés réactives se développant en effet par recherche d'un équilibre dans l'exploitation de cet environnement.

Ce type d'agent semble indispensable pour nos applications de vision. Il pourrait être introduit dans le bas niveau des systèmes, de par son aspect dégradé et pour l'aspect recherche d'équilibre, que l'on retrouve par exemple dans la segmentation pyramidale où le processus de fusion de régions dans les niveaux cesse quand la similarité entre les régions devient trop faible (atteinte d'un équilibre).

2. ARCHITECTURES MULTI-COUCHES HETEROGENES

L'objectif de ce paragraphe est plutôt d'évoquer quelques pistes de recherche dans le domaine de la conception d'architectures multi-couches hétérogènes et ne constitue donc pas une revue exhaustive du domaine. Nous introduisons tout d'abord la notion d'architectures hétérogènes puis la notion d'architectures multi-couches hétérogènes.

2.1. ARCHITECTURES HETEROGENES

Les architectures hétérogènes font intervenir des agents de granularité différente, allant de l'agent de faible granularité (réactif) à l'agent de forte granularité (cognitif). Le problème est alors de trouver un mode de représentation suffisamment homogène pour permettre d'appréhender des agents de différentes granularités. La plate-forme MACE **[Gasser 87]** est un exemple d'architecture hétérogène. Un langage est défini qui permet de décrire des architectures allant de la négociation de contrats (architecture de forte granularité) et des systèmes à base de tableau noir (architecture de granularité moyenne), jusqu'aux règles de production (architecture de faible granularité). La plate-forme MAGES **[Bouron 91]** permet

également la réalisation de telles architectures. Une hiérarchie d'agents est définie où les agents complexes sont bâtis sur des agents plus simples étendus par des mécanismes propres.

Outre ce problème d'homogénéité des représentations, le problème des systèmes hétérogènes est essentiellement un problème de communication. Il faut en effet faire cohabiter et coopérer des agents ayant des modes de représentation des connaissances différents et des styles de comportement différents. Les agents simples utilisent une communication de bas niveau, échangeant des messages simples, alors que les agents plus complexes échangent des messages complexes tels des frames. La plate-forme MAGES introduit un mécanisme de communication spécial qui exploite un processus de traduction. Les messages de type frame envoyés à des agents simples sont convertis en commandes simples, compréhensibles par ces agents. Inversement, les messages de bas niveau transmis à des agents complexes sont transformés en messages de type frame. La traduction est opérée par le biais d'une méthode dédiée (NeComprendPas), attachée aux agents.

Dans le système RATMAN **[Bürckert 91]**, des capacités de communication hétérogènes permettent d'échanger aussi bien des vecteurs que des mots clés ou des phrases complexes. L'intention des auteurs est d'utiliser la grammaire et la syntaxe de Montague qui peut être compilée en logique du premier ordre.

2.2. ARCHITECTURES MULTI-COUCHES

La "subsumption machine" de Brooks **[Brooks 91]** constitue un premier type d'architecture multi-couches issue de la décomposition par activité. La décomposition par activité est une alternative à la classique décomposition fonctionnelle. Cette décomposition ne fait pas de distinction entre les systèmes périphériques, tels la vision et les systèmes centraux. Le découpage fondamental d'un système intelligent s'effectue plutôt dans la direction orthogonale, le divisant en activités constituant des sous systèmes. Chaque activité constitue alors une couche connectant la perception à l'action. L'avantage de cette approche est qu'elle fournit un chemin incrémental, des systèmes simples aux systèmes autonomes complexes. Chaque couche ou activité de la "subsumption machine" est composée d'un réseau fixe d'automates d'état fini. Les couches sont combinées en ajoutant des connexions entre deux réseaux d'automates selon deux mécanismes, la suppression et l'inhibition. Dans le cas de la suppression, la connexion s'effectue en dérivant une connexion du réseau inférieur avant l'entrée dans un des automates de ce réseau, dans le cas de l'inhibition, cette dérivation s'effectue après la sortie d'un des automates. Ce type d'architecture est utilisé en robotique. Une première couche peut implanter par exemple un comportement permettant à un robot d'éviter des objets, une couche intermédiaire implanter un comportement de "promenade" et une couche de plus haut niveau implanter un comportement d'exploration. Chacune de ces couches est indépendante mais les

couches les plus élevées peuvent agir sur les couches inférieures en les inhibant ou en les supprimant.

Brooks définit donc une architecture multi-couches où chaque couche indépendante est constituée d'entités de granularité à peu près similaire, de type automate d'état fini. Fergusson **[Ferguson 92]** introduit également la notion de multi-couches. Cet auteur définit un système hybride pour la modélisation d'agents adaptatifs, rationnels et mobiles. Un tel agent, un robot par exemple, doit se montrer adaptatif, c'est-à-dire capable de développer ses objectifs dans un environnement non prédictible. Il doit donc être capable de développer des comportements différents : il doit se montrer réactif mais aussi capable de raisonner sur les événements qui surviennent, de déterminer leur effet sur ses objectifs, et éventuellement de les prédire. Ce besoin de flexibilité des comportements ne peut être atteint ni par des techniques de contrôle purement réactives, ni par des techniques purement délibératives. Une architecture multi-couches se prête bien à l'implantation de cette disparité d'aptitudes.

Dans ce système, chaque agent est composé de trois couches opérant de manière concurrente, avec des motivations indépendantes : une couche réactive, une couche de planification et une couche de modélisation (réflective / prédictive). Chacune de ces couches modélise le monde à des niveaux d'abstractions différents et possède des capacités orientées tâches. Elles créent de plus un lien direct entre la perception et l'action et décident de manière indépendante si elles doivent s'activer ou non dans une situation particulière. Un contrôleur global est nécessaire pour éviter les conflits entre décisions et actions des couches.

Un mécanisme de communication entre couches leur permet de s'examiner, voire se contrôler mutuellement. Un agent peut en effet recevoir un message émis par l'une ou l'autre des couches, cette communication horizontale peut être médiée par l'influence des autres couches dont le rôle peut être la suppression ou l'inhibition (communication verticale).

La couche réactive ne possède pas de mémoire et ne considère que des informations de nature sensorielle, elle peut alors produire des effets non rationnels. Pour y remédier, un message est envoyé à la couche de modélisation chaque fois que la couche réactive s'anime. Cette couche peut évaluer ainsi les effets de la décision réactive.

La couche de planification manipule et instancie des schémas de plans, extraits d'une librairie de schémas. Elle construit les plans d'un agent donné en utilisant une stratégie de type profondeur d'abord / meilleur d'abord. Elle ne perçoit pas l'environnement, sauf pour situer certains éléments en cas de besoin. En particulier, elle n'est pas influencée par les autres agents, ce rôle étant réservé à la couche de modélisation.

La couche de modélisation est la seule couche qui possède une vue raisonnée sur les autres et sur les événements du monde. Les structures utilisées par cette couche pour modéliser le comportement d'une entité sont des 4-uplets de la forme (C, B, D, I) où C est la configuration de l'entité (position, vitesse, ...), B est l'ensemble des croyances de cette entité, D est une liste de buts partiellement ordonnés, et I est un plan, ou structure d'intention. Ces modèles permettent de raisonner sur les différences entre des comportements effectifs et des comportements prédits par le modèle, ou bien, s'il s'agit d'un modèle de soi, entre un comportement effectif et le comportement désiré par l'agent. Une différence entre un comportement effectif et un comportement souhaité ne conduit pas nécessairement à une révision, il est en effet possible de programmer des seuils de similarité, autorisant certains écarts.

2.3. DISCUSSION

Brooks introduit une structuration en couche où chaque couche indépendante, est constituée d'entités similaires. L'approche de Fergusson introduit au contraire une structuration en couches interne au modèle d'agent, et où chaque couche possède un degré cognitif plus élevé que à la précédente. Une approche hybride, qui constitue notre définition du multi-couches, reviendrait à introduire une structuration comportant plusieurs groupes d'agents disposés en couche. Les agents impliqués dans les couches les plus basses seraient alors des agents réactifs et les agents impliqués dans les couches les plus élevées des agents cognitifs.

Les développements proposés par Fergusson montrent tout l'intérêt de concevoir des architectures multi-couches hétérogènes. Comme nous l'avons déjà souligné, il peut être particulièrement intéressant d'intégrer au sein d'un même système des agents capables de développer des comportements différents, tels des comportements opportunistes (réactifs) et des comportements raisonnés ou planifiés. Il apparait également que les informations perçues et manipulées par les agents diffèrent de manière significative d'une couche à l'autre. Il s'agit pour les couches réactives d'informations de nature plutôt perceptuelle et pour les couches cognitives d'informations de haut niveau concernant les configurations, les croyances, les buts et les intentions des autres entités. Ces adjonctions accroissent les comportements dont disposent le système et la gamme d'information qu'il est capable de manipuler, et donc ses possibilités de raisonnement et de contrôle. Dans ce cadre, la définition d'un modèle d'agent qui permette facilement de représenter des agents réactifs et des agents cognitifs et d'appréhender des liens de structuration verticaux entre agents nous semble indispensable.

III AMELIORATIONS STRUCTURELLES

Les améliorations structurelles concernent d'une part l'introduction d'agents de granularités différentes et d'autre part l'introduction d'une structuration hiérarchique des groupes d'agents par la définition d'architectures multi-couches.

En ce qui concerne l'introduction d'agents réactifs, il nous parait important de disposer d'un modèle d'agent qui permette, par simple glissement des définitions, d'appréhender des agents de granularités différentes.

En ce qui concerne les architectures multi-couches, deux solutions sont envisageables. Une première solution procéderait par composition, un agent pouvant faire appel à d'autres agents d'une couche différente, cette solution ayant l'avantage de pouvoir être implantée par simple extension du langage de programmation. La seconde solution procéderait par reflexivité et concernerait simplement le glissement du concept d'objet au concept d'agent, selon lequel tout élément référencé dans le cadre d'une quelconque activité exploitant des connaissances peut désigner indifféremment des objets ou des agents. Il ne s'agirait pas alors d'introduire un niveau "méta" défini de façon formelle par des spécifications précises, mais plutôt de parvenir à un glissement des définitions et des comportements, induit par une simple extension des spécifications de base : une telle extension deviendrait ainsi "transparente", et des définitions similaires conduiraient à des effets différents selon les types de ressources (objets ou agents) qu'elles exploitent. Pour des raisons de simplification, la première solution a été retenue dans un premier temps.

1. AGENTS REACTIFS

Nous proposons de dégrader un agent KS vers la notion de "frame". Un frame est un objet structuré dont les attributs sont dotés de facettes. Le frame s'active dès que certains de ses attributs sont modifiés, cette réaction d'exprime par des propagations internes ou externes (facettes si-ajout). Lorsqu'on lui demande une information, la réaction s'exprime alors par l'activation des facettes si-besoin : il y a alors appel de procédure (ou tout autre action simple) ou plus généralement l'activation d'un agent KP réactif. Une forme ultime de dégradation est alors le tableau noir, toute faculté de communication avec des agents KP est alors ôté à l'agent KS.

De même qu'un agent KS peut se voir doté de plus de réactivité, un agent KP peut être dégradé vers un agent KP réactif. Nous proposons de définir un tel agent comme une extension des sources de connaissance des architectures de type tableau noir. Un agent KP réactif est alors doté de plus d'opportunisme, il peut se déclencher de façon autonome dans certaines

conditions, et non plus attendre qu'un agent KS l'active par l'envoi d'un message. Dans une dégradation ultime, l'agent KP peut se résumer à une règle ou une procédure.

En terme de société d'agents, ces nouvelles définitions permettent la conception d'architectures réactives centrées KS ou centrées KP. Dans les architectures centrées KS, les KS sont des frames qui dirigent les KP par leurs facettes si-ajout et si-besoin (comportement de type démon) **(Fig. C.12)**.

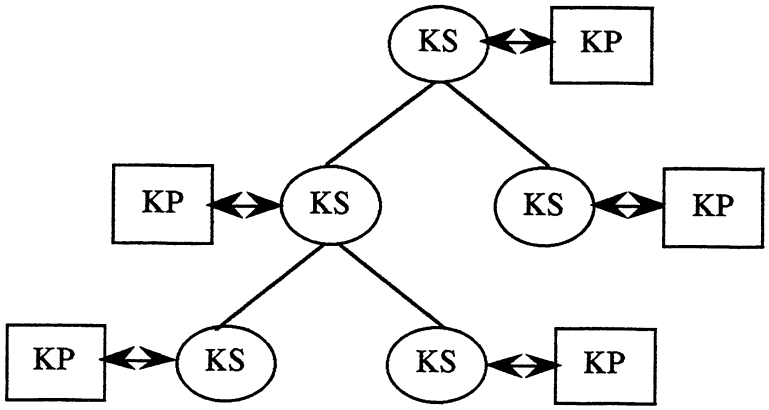


Fig. C.12. Architecture Réactive Centrée KS

Dans les architectures centrées KP, les KS sont au contraire des tableaux noirs passifs et les KP alors autonomes construisent la solution de façon incrémentale **(Fig. C.13)**. Les deux types d'architectures peuvent aisément être mixées.

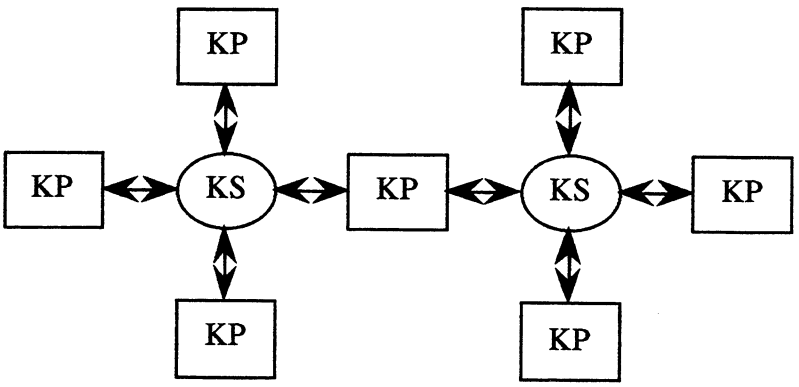


Fig. C.13. Architecture Réactive Centrée KP

Notre approche se prête finalement bien à une telle dégradation. Il est en effet possible de voir un agent KS comme un "super-frame", c'est à dire une entité capable de modification et de propagation interne, mais également capable d'accéder (en requête lecture ou écriture) à d'autres frames / agents. On aurait donc une approche réflexive, du "super-frame" (cognitif) au frame (réactif) avec un glissement de la facette, vers la méthode et le comportement, et de la valeur d'attribut vers l'instance, en termes d'information véhiculée lors des communications. En

suivant la même démarche, il est facile de montrer les liens entre la notion d'opérateur et la notion de KP.

2. HIERARCHIE MULTI-COUCHES

Nous avons vu précédemment (partie C, chapitre 3, paragraphe II.2.3), le besoin de mécanismes de synchronisation sophistiqués. Il est nécessaire de démultiplier les traitements et le contrôle qui s'appliquent à une information quelque soit son niveau d'abstraction. Ceci permet la création de hiérarchies de tâches et donc la modélisation de la résolution d'un problème à différents niveaux de détail (effet de loupe). Dans les couches les plus élevées l'information est alors perçue avec plus de recul, dans un contexte plus large et le discours qu'on a sur elle a plus de distance. Les agents ont ainsi des raisonnements de plus haut niveau, notamment des capacités plus sophistiquées de raisonnement sur les autres et sur l'environnement.

Comme nous l'avons vu (partie C, chapitre 1), la définition d'architectures multi-couches hétérogènes est particulièrement importante pour nos applications. La conception de systèmes performants en Compréhension de la Parole ou en Vision par Ordinateur nécessite en effet de manipuler des agents réactifs (étapes de bas niveau) et cognitifs (étapes de haut niveau) à des niveaux d'abstraction différents. Dans de telles architecture, les couches les plus basses pourront introduire des agents réactifs ou cognitifs et les couches les plus hautes des agents d'un degré cognitif plus élevé. Les agents d'une couche pourront être organisés en groupe, et s'échangeront essentiellement des messages d'information. Les messages échangés entre deux couches seront au contraire des messages de contrôle. Les agents d'une couche donnée pourront alors superviser ou organiser des agents ou des groupes d'agents de la couche inférieure, et ainsi effectuer des opérations de synchronisation, ou de planification.

Il y a ainsi deux axes de distribution : un axe vertical, qui concerne le degré de granularité des agents du réactif au cognitif et un axe horizontal, qui concerne les différentes étapes de transformation des informations. Le premier axe concerne ainsi l'abstraction en terme du contrôle des traitements et les second axe concerne l'abstraction de l'information.

Une première implantation d'architectures multi-couches a été effectuée **[Chiron 92]**. Ce travail constitue une tentative de structuration des actions (introduite au niveau des agents KP) au sein d'une architecture d'agents MAPS. Il est fondé sur la notion d'actinomie **[Vogel 88]**, structure qui sous-tend la représentation et le contrôle de l'action. Une actinomie peut être vue comme une combinaison de séquences telle que la succession, la juxtaposition ou l'enchâssement, une séquence étant une suite de trois actions : l'ouverture de la séquence, la réalisation de la séquence et la fermeture de la séquence. L'actinomie retenue est l'enchâssement **[Fig. C.14]**.

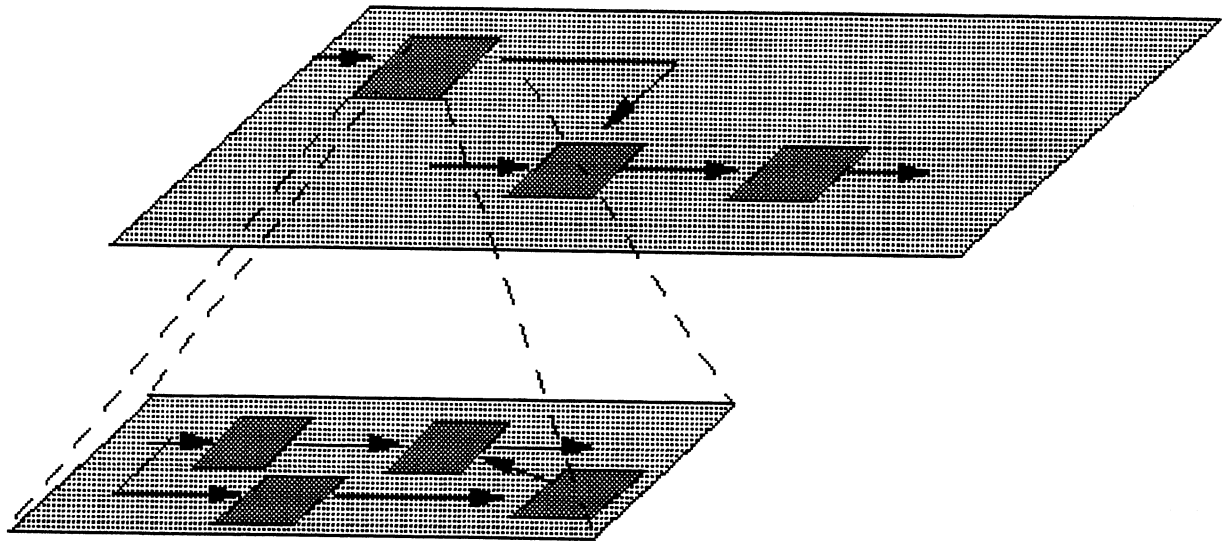


Fig. C.14. Architecture multi-couches implantant l'enchâssement.

Dans les architectures ainsi créées, une couche correspond à un groupe d'agent MAPS classique (ou réseau) et est responsable de la résolution d'un sous-problème, définissant ainsi un niveau de contrôle. Chaque agent (KS ou KP) appartient à un réseau. Un agent (KS ou KP) peut également appartenir à plusieurs réseaux du même niveau de contrôle et un agent KS peut appartenir à des réseaux de niveaux différents. Un agent KP selon l'actinomie retenue, contrôle et peut faire appel à des réseaux d'une couche inférieure (résolution d'un sous-problème). Un agent KP peut activer plusieurs réseaux d'une couche inférieure en parallèle (actuellement deux) et selon des stratégies prédéfinies (actuellement la concurrence et le vote). Une simple extension du langage de programmation **(exemple 1)** et l'ajout de classes au modèle générique des agents KP ont permis d'implanter de tels mécanismes.

```
AGENT Agt1 : KP {

CONNECTION Agt2;
CONNECTION Agt3;

NETWORKMEMBER : n3;

CONNECTIONNETWORK : n1;
CONNECTIONNETWORK : n2;

APPLIEDSTRATEGY : CONCURRENCE;

RULE r1 : ACTION {
IF   {int i; int j; int k; int l;
     SUB (i,j,k) in l WITH (OBJECT:c1,ATTRIBUTE:c2) IN AGENT:Agt2;
     }
THEN SET (i,j) WITH k IN l;
};

}
```

Exemple 1. L'agent KP Agt1 qui appartient au réseau n3 (NETWORKMEMBER) est relié aux deux agents KS Agt2 et Agt3 qui font partie du même réseau que lui. Il contrôle et peut faire appel aux réseau n1 et n2 (CONNECTIONNETWORK). La stratégie retenue est la concurrence (APPLIEDSTRATEGY). Les sous réseau sont activés par l'intermédiaire du mot-clé SUB. Ici, l'information constitué par l'attribut c2 de l'objet c1 de l'agent Agt2 est déléguée aux sous réseaux et le résultat du traitement de cette information est stocké dans les paramètres de retour qui correspondent à la valeur (k) d'un attribut (j) de l'objet (i) de l'agent (l). Une mise à jour peut alors être effectuée (SET).

Bien qu'étant encore rudimentaire, cette première implantation du multi-couches va nous permettre d'implanter des mécanismes de planification de tâches dans nos applications de vision et de compréhension de la parole et ainsi introduire un début de distribution vertical du contrôle (découpage en niveau de contrôle).

IV CONCLUSION

Dans ce chapitre, nous avons présenté les perspectives quant à l'évolution de MAPS vers une architecture hétérogène multi-couches. Cette évolution passe par des améliorations structurelles telles l'introduction d'agents réactifs et d'une structuration en couche des groupes d'agents. Nous avons montré que ces améliorations pouvait facilement être implantées, l'agent KS pouvant être facilement dégradé vers le concept de frame et l'agent KP vers la notion de source de connaissance ou de procédure. Ces développements sont encore embryonnaires, une première implantation du multi-couches a en particulier été effectuée, mais les perspectives au niveau de l'Intelligence Artificielle Distribuée, avec la conception d'une plate-forme de développement hétérogène et multi-couches et au niveau des applications, avec une plate-forme répondant aux besoins démontrés par l'état de l'art, semblent prometteuses.

CONCLUSION PARTIE C

L'intérêt de l'environnement MAPS pour le développement d'applications complexes dans des domaines aussi variés que la Vision par Ordinateur, le Diagnostic Biomédical et la Compréhension de la Parole a été confirmé par la création de plusieurs systèmes (KISS, KIDS, Parole). Ces systèmes ont toutefois montré les faiblesses de l'environnement MAPS et défini un certain nombre de besoins. Ces faiblesses relèvent de deux niveaux différents : un niveau fonctionnel et un niveau structurel. Les faiblesses du niveau fonctionnel s'expriment par un manque d'autonomie qui provient d'une part des dépendances temporelles liées au mode de fonctionnement séquentiel et d'autre part du manque d'intentionalité des agents. L'agent intentionel en effet raisonne sur les autres en fonction des connaissances dont il dispose, pour décider de manière autonome des ses actions. L'agent MAPS actuel au contraire ne peut décider de manière autonome des relations à établir avec les autres, il ne peut qu'appliquer ses règles dans lesquelles les liens de communication sont prédéfinis et codés de manière explicite. Enfin les faiblesses du niveau structurel proviennent à la fois du manque d'agents de haut niveau permettant la définition de couches de contrôle, et du manque d'agents de bas niveau permettant la définition de couches réactives.

Dans le même ordre d'idée, une comparaison avec les plate-formes de développement en Intelligence Artificielle Distribuée a certes montré que l'environnement MAPS possédait de nombreux points communs avec ces plate-formes mais que certaines caractéristiques (communication asynchrone, représentation et utilisation des connaissances sur les autres, autonomie des agents) lui faisaient défaut pour pouvoir véritablement le qualifier de plate-forme de développement en Intelligence Artificielle Distribuée.

Face à ces considérations, l'environnement MAPS a été étendu. Le nouvel environnement pallie à présent la plupart des faiblesses fonctionnelles. Un mode d'exécution parallèle des agents a été introduit grâce à l'introduction d'une communication asynchrone. De plus, il est à présent possible pour un agent de modéliser des connaissances sur les autres et de raisonner sur ces connaissances. Un nouveau langage plus proche du langage cible (le langage C++) a également été défini qui constitue une ouverture sur ce langage. Enfin, de nouvelles fonctionnalités de l'environnement de développement telles que la possibilité de définir des agents graphiques sont en cours d'implantation. D'autres améliorations telles que le parallélisme

intra-agent et le raisonnement en mondes multiples restent toutefois à implanter. Ce nouvel environnement est en cours de validation par la définition d'une nouvelle application en Compréhension de la Parole.

Les améliorations concernant les besoins du niveau structurel sont encore des perspectives. Nous avons toutefois montré la possibilité de dégrader les agents KS vers des agents plus réactifs de type "Frame" et les agents KP vers de simples règles ou procédures. Nous avons également présenté l'ébauche d'une conception multi-couches. Nos concepts initiaux se prêtant assez bien à l'introduction de ces améliorations, les perspectives semblent prometteuses au niveau de l'Intelligence Artificielle Distribuée, avec la conception d'une plate-forme de développement hétérogène et multi-couches répondant aux besoins des applications. Notons néanmoins que ces améliorations doivent être replacées dans le contexte d'un langage de programmation de plus haut niveau, impliquant par exemple des facilités d'expression de protocoles de communication et de contrôle sophistiqués, les rendant ainsi plus facilement exploitables.

REFERENCES

[Baujard 91a] O. Baujard (1991). Un Environnement pour le Développement d'Applications Réparties en Intelligence Artificielle. *Journées Unix de Grenoble, Actes des Conférences*, AFUU.

[Baujard 92] O. Baujard, S. Pesty and C. Garbay (1992). A Programming Environment for Distributed Applications Design in Artificial Intelligence. *Applications of Artificial Intelligence X : Knowledge-Based Systems,* pp 110-116. SPIE The International Society for Optical Engineering, Orlando 92.

[Bouron 91] T. Bouron, J. Ferber & F. Samuel (1991). MAGES : a Multi-Agent Testbed for Heterogeneous Agents. *Decentralized A.I 2.* pp 195-214. Demazeau & Müller Eds. Elsevier Science Publishers.

[Brooks 86] R.A. Brooks (1986). A Robust Layered Control System for a Mobile Robot, *IEEE Journ. Robotics and Automation*, pp 14-23, RA-2, April.

[Brooks 91] R.A. Brooks (1991). Intelligence Without Representation. Artificial Intelligence 91, numéro spécial IAD.

[Bruno 91] G. Bruno (1991). MAPS : un nouveau compilateur. Rapport de stage. IUT II Grenoble.

[Burckert 91] H.J. Bürckert & J. Müller (1991). RATMAN : Rational Agents Testbed for Multi-Agent Networks. *Decentralized A.I 2.* pp 217-230. Demazeau & Müller Eds. Elsevier Science Publishers.

[Caillaud 91] B. Caillaud (1991). Architectures Multi-Agents pour la Reconnaissance de la Parole. Rapport de DEA Signal Image Parole. Grenoble.

[Chiron 92] B. Chiron (1992). Une approche multi-couches dans une plate-forme de construction d'applications multi-agents. Rapport de DEA, ENSIMAG-INPG.

[Coutaz 88] J. Coutaz (1988). Interface Homme-Ordinateur : Conception et Réalisation. Thèse de l'Université Joseph Fourier. Grenoble 1988.

[Drogoul 92] A. Drogoul & C. Dubreuil (1992). Eco-Problem-Solving Model : Results of the N-Puzzle. *Decentralized A.I 3.* Werner & Demazeau Eds. Elsevier Science Publishers. (à paraitre)

[Ferguson 92] I.A. Fergusson (1992). Toward an Architecture for Adaptative, Rational, Mobile Agents. *Decentralized A.I 3.* Werner & Demazeau Eds. Elsevier Science Publishers. (à paraitre).

[Gasser 87] L. Gasser, C. Braganza & N. Herman (1987). MACE : A Flexible Testbed for Distributed AI Research. Distributed Artificial Intelligence. Vol 1, pp 119-154. Morgan Kaufmann Publishers.

[Hayes-Roth 88] F. Hayes-Roth, L.D. Erman, S. Fouse, J.S. Lark & J. Davidson (1988). ABE : a Cooperative Operating System and Development Environment. *Readings in Distributed Artificial Intelligence.* pp 457-488. A. Bond and L. Gasser Eds.

[Hayes-Roth 91] F. Hayes-Roth, J.E. Davidson, L.D. Erman & J.S. Lark (1991). Frameworks for Developing Intelligent Systems : the ABE Systems Engineering Environment. *IEEE Expert.* pp 30-40. IEEE.

[Hugonnard 91] E. Hugonnard & C. Garbay (1991). Knowledge Elicitation in Biomedicine : a Multi-Agent Approach. *Proc of Annual Int Conf of the IEEE Engineering in Medicine and Biology Society.* Vol 13:1991, pp 1317-1318. IEEE.

[Jennings 91] N.R. Jennings (1991). Cooperation in Industrial Systems, ESPRIT Conference, Brussels, 25-29 November.

[Kephart 89] J.O. Kephart, T. Hogg & B.A. Huberman (1989). Dynamics of Computational Ecosystems : Implications for DAI. *Distributed Artificial Intelligence.* Vol 2, pp 79-95. Morgan Kaufmann Publishers.

[Langton 88] C.G. Langton (1988). Artificial Life, SFI Studies in the Sciences of Complexity Artificial Life, pp 1-47, C. Langton Ed, Addison-Wesley Publishing Company.

[Montgomery 92] T.A. Montgomery, J. Lee, D. Musliner, E.H. Durfee, D. Damouth, Y. So & UM-DIAG Group (1992). MICE Users Guide, January 92.

[Ovalle 91] D.A. Ovalle (1991). Contribution à l'étude du raisonnement en univers Multi-Agent : KIDS, une application pour l'Interprétation d'Images Biomédicales. Thèse de l'Université Josehp Fourier - Grenoble I.

[Seel 91] N. Seel (1991). Intentional Description of Reactive Systems. *Decentralized A.I 2.* pp 15-33. Demazeau & Müller Eds. Elsevier Science Publishers.

[Sperry 61] R.W. Sperry (1961). Cerebral Organization and Behavior. *Science,* pp. 1749-1757. N° 133.

[Steels 90] L. Steels (1990). Cooperation between Distributed Agents through Self-Organisation. *Decentralized A.I.* pp 175-196. Demazeau & Müller Eds. Elsevier Science Publishers.

[Stroustrup 87] B. Stroustrup (1987). The C++ Programming Language. Addison-Wesley Publishing Company, Inc.

[Vogel 88] C. Vogel (1988). Génie Cognitif, Collection Sciences Cognitives, MASSON.

CONCLUSION THESE

L'objectif de cette thèse a été la conception d'un environnement de développement pour la résolution de problème. Une caractéristique importante de cette thèse concerne la technique de développement de cet environnement. Il est en effet issu de la synergie des besoins définis par plusieurs applications et des notions dégagées de l'étude de plusieurs formalismes de conception : langages à objets, systèmes experts et systèmes multi-agents. Ainsi, à partir de premiers besoins issus d'applications en Vision par Ordinateur et en Diagnostic Biomédical, ainsi que de l'étude des langages à objets et des systèmes multi-agents nous avons fait un certain nombre de choix qui nous ont amené à créer l'environnement de développement MAPS. Nous avons alors décidé de confronter cet environnement à de nombreux domaines d'application tels que la Vision et le Diagnostic mais également la Compréhension de la Parole, l'application privilégiée étant l'applcation de Vision.

Il faut noter ici une autre caractéristique importante, qui est l'apport de cet environnement pour ces applications. Le système KISS par exemple, n'est en effet pas un simple système de validation de l'environnement MAPS intégrant des notions classiques de la vision, mais surtout un système démontrant une architecure originale issues des caractéristiques de cet environnement. Le développement de ces applications a alors en retour illustré de nouveaux besoins, qui confrontés à l'état de l'art nous ont fourni les améliorations à apporter à l'environnement MAPS, améliorations qui ont alors été en partie implantées.

L'Environnement MAPS

MAPS a été conçu pour intégrer des formalismes issus des langages à objets, de l'Intelligence Artificielle Distribuée et dans le souci de conserver la distinction de connaissances figuratives et de connaissances opératoires mise en évidence par nos applications et confirmée par les langages de programmation. Dans ce cadre, deux classes génériques d'agents ont été définies : des agents dédiés à la modélisation et à l'exploitation des connaissances figuratives (agents KS), et des agents dédiés à la modélisation et à l'exploitation des connaissances opératoires (agents KP). Cette disinction de deux types d'agents nous a de plus permis de distribuer au sein de chaque agent le contrôle propre à chaque type de connaissance : un contrôle "figuratif" pour les agents KS, en terme de focalisation sur des informations et un contrôle "opératoire" pour les agents KP, en terme d'adaptation des traitements aux données. Notons que ce contrôle va

s'exercer ainsi à un niveau interne et local, mais également à un niveau externe et plus global par le choix des agents à activer.

Ces concepts architecturaux ont été implantés avec la définition d'un environnement de développement performant comprenant un langage de programmation permettant de spécialiser les classes d'agents KS et KP, et doté d'outils de mise au point et de visualisation (interfaces graphiques). Le langage de programmation s'apparente à un langage hybride (langage à objets) permettant d'intégrer plusieurs formalismes tels la programmation orientée objets, la programmation logique et procédurale. Il possède ainsi une syntaxe assez souple mais de bas niveau ce qui peut parfois limiter le développement des applications.

KISS

Les applications ont permis de montrer la faisabilité d'un système à partir de cet environnement. En ce qui concerne le système de vision KISS, les différents niveaux de représentation et de traitements de la vision ont pu être facilement introduits grâce à leur distribution au sein d'agents KS et KP. La distribution du contrôle propre à chaque type de connaissance (figuratif et opératoire) a permis d'implanter des mécanismes de focalisation et d'adaptation d'une façon locale et décentralisée. L'originalité du système est d'avoir alors pu définir des filières d'analyse séparées, introduisant des points de vue différents au sein des différents niveaux et d'avoir utilisé les possibilités de contrôle pour faire coopérer ces deux filières d'analyse et pour intégrer les informations de différents points de vue, pour bâtir une segmentation et une interprétation des images. Ce système bien que démontrant des caractéristiques intéressantes, pourrait toutefois être amélioré par l'introduction d'outils plus performants, par l'introduction de stratégies de prédiction / vérification plus sophistiquées, notamment dans les niveaux les plus abstraits et l'introduction d'un mode de fonctionnement parallèle. L'architecture du système KISS pourrait également être mieux structurée, à l'aide de niveaux de contrôle supplémentaires et à l'aide d'agents de différentes granularités.

Evolutions

En retour des applications, notamment de l'application de vision, deux types de besoins ont été dégagés d'ordres différents : des besoins d'ordre fonctionnel et des besoins d'ordre struturel. Des améliorations ont ainsi été introduites dans l'environnement MAPS, concernant une partie des besoins fonctionnels. Un mode d'exécution parallèle des agents a été introduit, ainsi que la possibilité de modéliser des connaissances sur les autres, dont nous avons souligné l'intérêt en cve qui concerne l'accroissement de l'autonomie des agents. Le langage de programmation a également été modifié, dont la syntaxe est à présent plus proche du langage cible C++. Enfin, l'environnement de programmation a été modifié afin de prendre en compte l'exécution parallèle et répartie des agents. Notons cependant que le langage de programmation souffre toujours du

manque de primitives de haut niveau ce qui nécessite une programmation parfois fastidieuse des agents. Les besoins d'ordre structurel, concernant la définition d'agents de différentes granularités et une structuration du contrôle à l'aide de couches, sont encore d'ordre prospectif.

Perspectives

En ce qui concerne l'environnement de programmation MAPS, les perspectives concernent la conception d'une plate-forme autorisant le développement de modèles d'agents pour la conception d'architectures hétérogènes multi-couches. Pour cela, l'ouverture du langage de programmation, permettant l'intégration de code C++ au niveau de la définition des agents s'avère de première importance. Les principales caractéristiques des agents telles les comportements, les capacités pourront être surchargées (mécanisme propre au langage C++, permettant de redéfinir des méthodes de classes) par de nouvelles caractéristiques définies par l'utilisateur.

En ce qui concerne la Vision par Ordinateur, l'introduction de l'hétérogénéité permettra de définir des groupes réactifs dans le bas niveau par exemple, où une représentation pyramidale de l'image pourra être bâtie sur de tels agents et permettra d'introduire des mécanismes de focalisation et de prédiction / vérification complexes. La possibilité de construire des architectures multi-couches introduira une structuration au niveau du contrôle, qui permettra de mettre en œuvre des mécanismes de planification, de synchronisation et de focalisation. Ces différentes caractéristiques feront de KISS un système de vision, intégrant une représentation multi-niveaux des connaissances, des tâches et du contrôle ainsi que des données à analyser, et ce de façon parfaitement homogène.

Enfin, nous avons vu que cet environnement dépassait largement le cadre des applications de vision. Toutes les applications vont se voir renforcées par l'évolution de MAPS. En particulier, nous avons vu l'intérêt d'une structuration du contrôle pour l'application de Compréhension de la Parole. D'autres applications sont également à l'étude telle la simulation de populations cellulaires visant à simuler des phénomènes de prolifération cellulaire dans des populations saines ou caractéristiques d'états pathologiques, rendue possible par l'introduction d'agents réactifs. Nous sommes en effet persuadés que MAPS ne peut que bénéficier d'une synergie des applications et des techniques logicielles, qui a constitué notre méthode de travail.

REFERENCES

[Aloymonos 90] J.Y. Aloymonos (1990). Purposive and Qualitative Active Vision, *Workshop on Active Vision European Conference on Computer* Vision, April 90.

[Andress 87] K.M. Andress & A.C. Kak (1987). A production system environment for integrating knowledge with images. Technical report TR-EE-87-36, School of Electrical Engineering, Purdue University, October 1987.

[Ayache 86] N. Ayache & O.D. Faugeras (1986). Hyper: a new approach for the recognition and positionning of 2D objects. *IEEE Trans. on PAMI*, 8(1):44-54.

[Bajcsy 74] R. Bajcsy & L.I. Liberman (1974). Computer description of real outdoor scenes. *Proc. of the 2d ICPR*, IEEE Computer Society Press.

[Ballard 82] D.H. Ballard & C.M. Brown (1982). Computer Vision, Prentice Hall, 1982.

[Baujard 89a] O. Baujard (1989). Un Système de Vision par Ordinateur. *Rapport de DEA*. ENSIMAG / INPG. Grenoble.

[Baujard 89b] O. Baujard and C. Garbay (1989). KISS: Un Système de Vision Multi-Agents. *Actes du VIIème Congrès "RFIA"*., pp 89-98, AFCET / INRIA.

[Baujard 90a] O. Baujard & C. Garbay (1990). A programming environment for distributed expert system design. *Expert System Applications, ExperSys.*, pp 27-32.

[Baujard 90b] O. Baujard (1990). MAPS : Guide de Programmation. Rapport Technique 89G/MRT/DT002. Alcatel TITN Answare.

[Baujard 91a] O. Baujard (1991). Un Environnement pour le Développement d'Applications Réparties en Intelligence Artificielle. *Journées Unix de Grenoble, Actes des Conférences*, AFUU.

[Baujard 91b] O. Baujard, S. Pesty and C. Garbay (1991). A Programming Environment for Distributed Vision System Design, *in Proc of 6th Int Conf on Image Analysis and Processing*, pp 380-384, World Scientific.

[Baujard 92] O. Baujard, S. Pesty and C. Garbay (1992). A Programming Environment for Distributed Applications Design in Artificial Intelligence. *Applications of Artificial Intelligence X : Knowledge-Based Systems,* pp 110-116. SPIE The International Society for Optical Engineering, Orlando 92.

[Belaïd 91] A. Belaïd & Y. Belaïd (1991). Reconnaissance des Formes : Methodes et Applications. InterEditions.

[Berger 91] M. Berger (1991). Towards Dynamic Adaptation of Snake Contours. 6th Int. Conf. on Image Analysis and Processing. (à paraître).

[Berthet 92] S. Berthet, Y. Demazeau & O. Boissier (1992). Knowing Each Other Better, *11th DAI Workshop*, pp 1-20, Glen Arbor.

[Bessières 83] P. Bessières (1983). Etude de la génération de plans en univers multi-agents. Thèse de 3° cycle, INPG, Grenoble.

[Bhattacharya 67] C. G. Bhattacharya, "A simple method of resolution of a distribution into Gaussian components," *Biometrics*, vol. 23, pp. 115-135, 1967.

[Birtwistle 73] G. Birtwistle, O.J. Dahl, B. Myhrhaug & K. Nygaard (1973). SIMULA begin, Petrocelli Charter, New York.

[Bobrow 77] D.G. Bobrow & T. Winograd (1977). An overview of KRL, a Knowledge Representation Language. In Proceedings of the 5th IJCAI, pp 213-222, Cambridge, MA.

[Bobrow 83] D.G. Bobrow & M. Stefik (1983). The LOOPS Manual : a Data and Object Oriented Programming System for Interlisp. *Knowledge-Based VLSI Design Group Memo KB-VLSI-81-13,* Xerox PARC, Palo Alto, California.

[Boissier 92] O. Boissier & Y. Demazeau (1992). A distributed artificial intelligence view on general purpose vision systems. *in Decentralized AI 3,* Werner & Demazeau Eds. North Holland.

[Bolles 82] R.C. Bolles & R.A. Cain (1982). Recognizing and Locating Partially Visible Objects: the Local-Feature-Focus Method. *Int. J. of Robotics Research*, 1(3):57-82.

[Bouron 91] T. Bouron, J. Ferber & F. Samuel (1991). MAGES : a Multi-Agent Testbed for Heterogeneous Agents. *Decentralized A.I 2.* pp 195-214. Demazeau & Müller Eds. Elsevier Science Publishers.

[Briot 88] J.P. Briot (1988). From Objects to Actors : Study of a Limited Symbiosis in Smalltalk-80. Rapport de recherche 88-58 RXF, LITP, Université de Paris 6.

[Brooks 83] R.A. Brooks (1983). Model-based Three-dimensional Interpretation of Two-dimensional Images. *IEEE Trans. PAMI*, 5(2):140-150.

[Brooks 86] R.A. Brooks (1986). A Robust Layered Control System for a Mobile Robot, *IEEE Journ. Robotics and Automation*, pp 14-23, RA-2, April.

[Brooks 91] R.A. Brooks (1991). Intelligence Without Representation. Artificial Intelligence 91, numéro spécial IAD.

[Brown 90] M.D. Brown & R.B. Fisher (1990). A Distributed Blackboard System for Vision Applications. *Proc of the British Machine Vision Conference.* pp 163-168. BMVC 90.

[Bruno 91] G. Bruno (1991). MAPS : un nouveau compilateur. Rapport de stage. IUT II Grenoble.

[Burckert 91] H.J. Bürckert & J. Müller (1991). RATMAN : Rational Agents Testbed for Multi-Agent Networks. *Decentralized A.I 2.* pp 217-230. Demazeau & Müller Eds. Elsevier Science Publishers.

[Caillaud 91] B. Caillaud (1991). Architectures Multi-Agents pour la Reconnaissance de la Parole. Rapport de DEA Signal Image Parole. Grenoble.

[Camarata 83] S. Camarata, D. McArthur & R. Steeb (1983). Strategies of cooperation in distributed problem solving. *Proc 8th Joint Conf Artificial Intel,* pp 767-770. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988.

[Campbell 90] J. Campbell & M. D'Inverno (1991). Knowledge Interchange Protocol, *Decentralized AI 2,* pp 63-80, Demazeau & Müller eds, North-Holland/Elsevier.

[Canny 86] J.F. Canny (1986). A Computational Approach to Edge Detection, *IEEE Transactions PAMI 8,* N° 6, pp 679-698, IEEE.

[Celeux 89] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier & H. Ralambondrainy. Classification automatique des données, DUNOD, Paris.

[Chaib-Draa 90] B. Chaib-Draa (1990). Contribution à la Résolution Distribuée de Problème : une Approche basée sur les Etats Intentionnels. Thèse de l'Université de Valanciennes et du Hainaut-Cambrésis.

[Chang 91] M. Chang & C. Woo (1991). SANP : a communication level protocol for negociations, *Decentralized AI 3,* Demazeau & Müller eds, North-Holland/Elsevier (à paraître).

[Chassery 84] J.M. Chassery & C. Garbay (1984). An Iterative Segmentation Method based on a Contextual Color and Shape Criterion, *IEEE Trans PAMI 6,* pp 794-800, IEEE.

[Chassery 86] J. M. Chassery and G. Bourrel, "An image package software: IPS design and abilities," *8th Int. Conf. Pattern Recog,* vol. 2, pp. 913-915, Paris 1986.

[Chassery 91] J.M. Chassery & A. Montanvert (1991). Géométrie discrète en analyse d'images, Série-Image, Ed. HERMES 1991.

[Chazelle 85] B. Chazelle & D.P. Dobkin (1985). Optimal Convex Decomposition. *in Computational Geometry,* Toussaint G.T. Eds, Elsevier Science Publ.

[Chehikian 91] A. Chehikian & J.L. Crowley (1991). Fast Computation of Optimal Semi-Octave Pyramids. *Proc of 7th SCIA,* pp 18-39.

[Chiron 92] B. Chiron (1992). Une approche multi-couches dans une plate-forme de construction d'applications multi-agents. Rapport de DEA, ENSIMAG-INPG.

[Clayton 84] B.D. Clayton (1984). ART Programming Primer. Inference Corporation, Los Angeles, California.

[Clément 89] V. Clément & M. Thonnat (1989). Handling knowledge in image processing libraries to build automatique systems. *2nd Int. Work. on Industrial Applications of Machine Intelligence and Vision,* pp 187-192, Tokyo.

[Connah 88] D. Connah, M. Shiels & P. Wavish (1988). A Testbed for Research on Cooperating Agents, *in Proc European Conference on Artificial Intelligence.*

[Conry 88] S. Conry, R.A. Meyer & J. Searlmen (1988). A shared knowledge base for independent problem solving agents. *Proc IEEE Expert Systems in Government Symposium,* McLean VA.

[Corkill 79] D.D. Corkill (1979). Hierarchical planning in a distributed environment. *in Proc 6th Joint Int Conf Artificial Intel,* pp 168-179.

[Corkill 83] D.D. Corkill & V.R. Lesser (1983). The use of meta-level control for coordination in a distributed problem solving network. *in Proc 8th Joint Conf Artificial Intelligence* , pp 748-756.

[Coutaz 88] J. Coutaz (1988). Interface Homme-Ordinateur : Conception et Réalisation. Thèse de l'Université Joseph Fourier. Grenoble 1988.

[Crowley 89] J.L. Crowley, A. Chehikian, J.O. Eklundh, J. Kittler, J. Illingworth; G. Granlund, J. Wilkund, E. Granum, H.I. Christensen (1989). Technical Annex for ESPRIT Basic Research Action 3038, Vision As Process, Aalborg, March 89.

[Davis 83] R. Davis & R.G. Smith (1983). Negociation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence 20.*

[Dellepiane 89] S. Dellepiane, C. Regazzoni, S.B. Serpico & G. Vernazza (1989). "An application-independent knowledge-based framework for complex image recognition," *5th Int. Conf. Image Analysis and Processing,* Positano 1989.

[Demazeau 86] Y. Demazeau (1986). Niveaux de représentation pour la vision par ordinateur. Indices d'image et indices de scène. Thèse de l'Institut National Polytechnique de Grenoble.

[Deriche 87] R. Deriche & J.P. Cocquerez (1987). Extraction de composantes connexes basée sur une détection optimale des contours, *MARI 87,* vol 2, pp1-9.

[Deriche 90] R. Deriche (1990). Fast Algorithms for Low-Level Vision. *IEEE. Trans on PAMI 12,* N° 1, pp 78-97. IEEE.

[Drogoul 92] A. Drogoul & C. Dubreuil (1992). Eco-Problem-Solving Model : Results of the N-Puzzle. *Decentralized A.I 3.* Werner & Demazeau Eds. Elsevier Science Publishers. (à paraitre)

[Ducournau 88] R. Ducournau (1988). YAFOOL. Version 3.22. Manuel de référence. SEMA.METRA, Montrouge.

[Duguerdil 87] P. Dugerdil (1987). Les mécanismes d'héritage d'OBJLOG : vertical et sélectif multiple avec point de vue. Actes du 6ème CARFIA, pp 259-273, Antibes.

[Durbin 87] R. Durbin & D. Willshaw (1987). An Analogue Approach to the Travelling Salesman Problem using an Elastic Net Method. *Nature,* 326, pp 689-691.

[Durfee 87] E.H. Durfee & V.R. Lesser (1987). Using partial global plans to coordinate distributed problem solvers. *in Proc 10th Int Joint Conf Artificial Intel,* pp 875-883. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988.

[Erceau 91] J. Erceau & J. Ferber (1991). L'Intelligence Artificielle Distribuée. "La Recherche". Numéro 233, Vol 22, pp. 750-, juin 1991.

[Erman 81] L.D. Erman, P.E. London & S.F. Fickas (1981). The Design and an example of use of Hearsay III. *Proc 7th IJCAI,* pp 409-415, 1981.

[Fabre 88] P. Fabre, F. Albert & P. Louw (1988). Reconnexion de contour utilisant le parcours optimal d'un graphe, application aux images de tunnels. *Congrès PIXIM 88.* pp 379-390.

[Faugeras 82] O.D. Faugeras (1982). Relaxation labelling and evidence gathering. *Proc. of the 7th ICPR,* pp. 405-412, IEEE Computer Society Press.

[Ferber 85] J. Ferber (1985). Définition récursive et évaluation distribuée : un modèle rigoureux de langage objet. Actes du 5ème CARFIA, pp 715-724.

[Ferber 91a] J. Ferber & E. Jacopin (1991). The Framework of Eco-Problem Solving. *Decentralized A.I 2.* pp 181-193. Demazeau & Müller Eds. Elsevier Science Publishers.

[Ferber 91b] J. Ferber (1991). Introduction à l'Intelligence Artificielle Distribuée. Dossier : Intelligence Artificielle Distribuée. *Bulletin de l'AFIA,* n°6, pp 16-19, juillet 1991.

[Ferguson 92] I.A. Fergusson (1992). Toward an Architecture for Adaptative, Rational, Mobile Agents. *Decentralized A.I 3.* Werner & Demazeau Eds. Elsevier Science Publishers. (à paraitre).

[Ferrari 80] L. Ferrari, P.V. Sankar & J. Sklansky (1980). Minimal Rectangular Partitions of Digitized Blobs, *5th ICPR,* pp 1040-1043, IEEE Com Soc Press, Miami.

[Fikes 85] R. Fikes & T. Kehler (1985). The Role of Frame-Based Representation in Reasoning. Communications of the ACM, 28(9):904-920.

[Fisher 58] W. D. Fisher, "On grouping for maximum homogeneity," JASA, vol. 53, pp. 789-798, 1958.

[Fu 80] K.S. Fu (1980). Introduction. *In "Digital Pattern Recognition",* pp. 1-14, (K.S. Fu, eds), Springer Verlag.

[Funakubo 84] N. Funakubo (1984). Region segmentation of biomedical tissue image using color texture feature. *Proc. of the 7th ICPR,* pp. 30-32, IEEE Computer Society Press.

[Gagalowitz 85] A. Gagalowitz & O. Monga (1985). Un algorithme de segmentation hiérarchique. *Actes du 5ième Congrès RFIA*, pp. 163-177, AFCET / ADI / INRIA.

[Galliers 88] J.R.Galliers (1988). A strategic framework for multi-agent cooperative dialogue. *in Proc Europ Conf on Artif Intell, ECAI,* pp 415-420.

[Garbay 86a] C. Garbay (1986). Images, stratégies perceptives et stratégies cognitives d'analyse. Thèse d'Etat. Grenoble.

[Garbay 86b] C. Garbay (1986). Image Structure Representation and Processing : Decision of some Segmentation Methods in Cytology, *IEEE Trans PAMI 8(12),* pp 140-146, IEEE.

[Garnesson 89] P. Garnesson, G. Giraudon & P. Montesinos (1989). "MESSIE: a multi-expert system in computer vision application for aerial interpretation," *Actes du VIIème Congrès RFIA*, vol. 2, pp. 817-828, AFCET / INRIA.

[Gaspar 91] G. Gaspar (1991). Communication and Belief Changes in a Society of Agents : Towards a Formal Model of Autonomous Agent, *Decentralized AI 2,* pp 245-256, Demazeau & Müller eds, North-Holland/Elsevier.

[Gasser 87] L. Gasser, C. Braganza & N. Herman (1987). MACE : A Flexible Testbed for Distributed AI Research. Distributed Artificial Intelligence. Vol 1, pp 119-154. Morgan Kaufmann Publishers.

[GDR 91] GDR 134. (1991). Traitement du Signal et Images : Prétraitement et approche frontière. *Rapport Segmentation.* Décembre 1991.

[Geman 86] D. Geman, S. Geman & C. Graffigne (1986). Locating texture and object boundaries. In "*Pattern Recognition Theory and Applications* ", (P. Devijver, eds), NATO ASI Series, Springer Verlag, Hedelberg.

[Geman 87] S. Geman & C. Graffigne (1987). Markov randon fields image models and their applications to computer vision. *Proc. of the Int. Congress of Mathematicians,* (A.M. Gleason, eds), American Mathematical Society, Providence.

[Georgeff 83] M. Georgeff (1983). Communication and interaction in multi-agent planning. *in Proc National Conf Artificial Intel.* Washington DC, pp 125-129. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 200-204.

[Georgeff 84] M. Georgeff (1984). A theory of action for multiagent planning. in Proc National Conf Artificial Intel, Austin TX, pp 121-125. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 205-209.

[Georgeff 86] M. Georgeff (1986). A representation of events in multiagent planning. *in Proc National Conf Artificial Intel,* Philadelphia, PA, pp 70-75.

[Goldberg 83] A. Goldberg & D. Robson (1983). Smalltalk-80, the language and its implementation. Addison Wesley, reading, Massachusetts.

[Goldstein 77] I.P. Goldstein & R.B. Roberts (1977). NUDGE, a Knowledge-based Scheduling Program. In Proceedings of the 5th IJCAI, pp 257-263, Cambridge, MA.

[Granger 85] C. Granger (1985). Reconnaissance d'Objets par Mise en Correspondance en Vision par Ordinateur. Thèse de Doctorat, Université de Nice.

[Hämmäinen 90] H. Hämmäinen, J. Alasuvanto & R. Mäntylä (1990). Experiences on Semi-Autonomous User Agents. *Decentralized A.I.* pp 253-249. Demazeau & Müller Eds. Elsevier Science Publishers.

[Hanson 78] A.R. Hanson & E.M. Riseman (1978). VISIONS: a computer system for interpreting scenes. *Computer Vision Systems, (A.R. Hanson & E.M. Riseman, Eds)*, pp. 303-333, Academic Press.

[Hanson 87] A.R. Hanson & .M. Riseman (1987). The VISIONS Image Understanding System, *in Advances in Computer Vision,* C. Brown Eds. L. Erlbaum.

[Haralick 75] R.M. Haralick & I. Dinstein (1975). A spatial clustering procedure for multi-image data. *IEEE Trans. on Circuits & Systems*, CAS-22: 440-450.

[Haralick 81] R.M. Haralick (1981). A facet model for image data: regions, edges and textures. In "Digital Image processing", pp. 337-356, (J.C. Simon and R.M. Haralick, eds), D. Reidel Publishing Company.

[Haralick 84] R.M. Haralick (1984). Digital Step Edges from Zero-Crossings of Second Directional Derivative. *IEEE Transactions PAMI 6*, pp 58-68. IEEE.

[Haralick 85] R.M. Haralick & L.G. Shapiro (1985). Image Segmentation Techniques. *Computer, Graphics, and Image Processing*, 29:100-132.

[Harvey 88] G. Harvey (1988). Understanding HyperCard for Version 1.1 ; Sybex Books Publishers.

[Haton 87] J.P. Haton (1987). Les systèmes à base de connaissances en reconnaissance et en interprétation de formes. pp 74-80, *Cognitiva'87.*

[Hautin 86] F. Hautin & A. Vailly (1986). La coopération entre systèmes experts. *Actes des Journées Nationales du PRC-GRECO "Intelligence Artificielle"*, Cepadues Editions.

[Hayes-Roth 85] B. Hayes-Roth (1985). A blackboard architecture for control. *Artificial Intelligence,* 26:251-321, 1985.

[Hayes-Roth 88] F. Hayes-Roth, L.D. Erman, S. Fouse, J.S. Lark & J. Davidson (1988). ABE : a Cooperative Operating System and Development Environment. *Readings in Distributed Artificial Intelligence.* pp 457-488. A. Bond and L. Gasser Eds.

[Hayes-Roth 91] F. Hayes-Roth, J.E. Davidson, L.D. Erman & J.S. Lark (1991). Frameworks for Developing Intelligent Systems : the ABE Systems Engineering Environment. *IEEE Expert.* pp 30-40. IEEE.

[Hewitt 71] C.E. Hewitt (1971). Procedural Embedding of Knowledge in PLANNER. In Proceedings of the 2nd IJCAI, pp 167-182, London.

[Hewitt 73] C.E. Hewitt, P. Bishop & R. Steiger (1973). A Universal Modular ACTOR Formalism for Artificial Intelligence. In Proceedings of the 3rd IJCAI, pp 235-245, Stanford, California.

[Hewitt 77] C.E. Hewitt (1977). Viewing Control Structure as Patterns of Passing Messages. *Artificial Intelligence,* 8:323-364, 1977.

[Hewitt 83] C. Hewitt & H. Lieberman (1983). Design Issues in Parallel Architectures for Artificial Intelligence. MIT Artificial Intelligence Lab, Cambridge, MA, Memo 750.

[Hewitt 84] C. Hewitt & H. Lieberman (1984). Design Issues in Parallel Architectures for Artificial Intelligence, *Proc 28th IEEE Computer Soc Int Conf, San Fransisco, CA,* pp 418-423.

[Hopkins 88] C. Hopkins (1988). DePlan : Enabling Agents to Produce Plans that Achieve Cooperative Problem Solving, *in Proc Europ Conf on Artif Intell (ECAI),* pp 421-425.

[Horaud 87] R. Horaud (1987). New methods for matching 3D objects with single perspective views. *IEEE Trans. on PAMI,* 9(3):401-412.

[Horowitz 76] S.L. Horowitz & T. Pavlidis (1976). Picture segmentation by a tree-traversal algorithm. *J. Assoc. Comp. Mach.* 23:368-388.

[Hu 89] H.T. Hu & J.M. Chassery (1989). Diagrammes de Voronoî généralisés. *Rapport de Recherche 799-IMAG,* Grenoble 1989.

[Hueckel 71] M.F. Hueckel (1971). An operator with locates edges in digitized pictures. *J.Ass. Comput. Match,* Vol 18, N° 1, pp 113-125.

[Hugonnard 91] E. Hugonnard & C. Garbay (1991). Knowledge Elicitation in Biomedicine : a Multi-Agent Approach. *Proc of Annual Int Conf of the IEEE Engineering in Medicine and Biology Society.* Vol 13:1991, pp 1317-1318. IEEE.

[Jennings 91] N.R. Jennings (1991). Cooperation in Industrial Systems, ESPRIT Conference, Brussels, 25-29 November.

[Kass 87] M. Kass, A. Witkin & D. Terzopoulos (1987). Snakes : active contour models, *Proc of the 1st Int Conf on Computer Vision,* pp 259-268.

[Kephart 89] J.O. Kephart, T. Hogg & B.A. Huberman (1989). Dynamics of Computational Ecosystems : Implications for DAI. *Distributed Artificial Intelligence.* Vol 2, pp 79-95. Morgan Kaufmann Publishers.

[Kittler 85] J. Kittler, J. Illingworth & J. Foglein (1985). Threshold selection based on a simple image statistic, *CVGIP 30,* pp 125-147.

[Klinger 73] A. Klinger (1973). Data structures and pattern recognition. *Proc. of the 1st ICPR,* pp. 497-498, IEEE Computer Society Press.

[Kohl 87] C.A. Kohl, A.R. Hanson & E.M. Riseman (1987). A Goal-Directed Intermediate Level Executive for Image Interpretation. *Proceedings of*

the tenth International Joint Conference on Artificial Intelligence, vol. 2 pp 811-814. IEEE Computer Society Press.

[Konolige 83] K. Konolige (1983). A deductive model of belief, *in Proc 8th Int Joint Conf Artificial Intell,* pp 377-381.

[Kornfeld 81] W.A. Kornfeld & C. Hewitt (1981). The scientific community metaphor, IEEE Trans Syst Man Cyber, SCM-11, pp 24-33. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 311-320.

[Kropatsch 83] W. Kropatsch (1983). Segmentation of digital images using a priori information about the expected image content. In "Pictorial Data Analysis", pp. 107-119 (R.M. Haralick, eds), Springer Verlag.

[Lâasri 89] H. Lâasri & B. Maître (1989). Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et Réalisations. Thèse de l'Université de Nancy 1. CRIN / INRIA-LORRAINE.

[Lane 89] D.M. Lane, M. J. Chantler, E.W. Robertson & A.G. McFadzean (1989). A Distributed Problem Solving Architecture for Knowledge Based Vision. *Distributed Artificial Intelligence.*

[Langton 88] C.G. Langton (1988). Artificial Life, SFI Studies in the Sciences of Complexity Artificial Life, pp 1-47, C. Langton Ed, Addison-Wesley Publishing Company.

[Lansky 87] A.L. Lansky & D. Fogelsong (1987). Localised representation and planning methods of parallel domains, *in Proc National Conf Artificial Intel,* Seattle WA, pp 240-245.

[Leblanc 86] T.J. Leblanc (1986). Shared memory versus message-passing in a tightly-coupled multiprocessor : a case study. Univ of Rochester Computer Science Department, Rochester, NY, Tech Rep. Buterfly Project Report 3.

[Lee 90] S.U. Lee, S.Y. Chung & R.H. Park (1990). Comparative Performance Study of Several Global Thresholding Techniques for Segmentation. *CVGIP 52,* 171-190.

[Lesser 80] V.R. Lesser & L.D. Erman (1980). Distributed interpretation : a model and experiment. IEEE Trans Computer. C-29. pp 1144-1163. also in *Readings in Distributed Artificial Intelligence* (Bond AH and Gasser L Eds), San Mateo, CA, Morgan Kaufmann, 1988, pp 120-139.

[Levialdi 83] S. Levialdi (1983). Neighbourhood operators: an outlook. In "*Pictorial Data Analysis*", pp. 1-14, (R.M. Haralick, eds), Springer Verlag.

[Levine 76] M.D. Levine & J. Leemet (1976). A method for non-purposive picture segmentation. *Proc. of the 3d ICPR.*

[Lieberman 81] H. Lieberman (1981). A preview of Act 1. AI Memo 625, AI Lab, MIT, Cambridge, MA.

[Lippmann 87] R.P. Lippmann (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine,* 4(2):4-22.

[Lux 84] A. Lux & V. Souvignier (1984). PVV: un système de vision appliquant une stratégie de prédicion-vérification. *Actes du 4ième Congrès RFIA*, pp. 223-224, AFCET / INRIA.

[Lux 85] A. Lux (1985). Algorithmique et Contrôle en Vision par Ordinateur. Thèse d'état, Université Joseph Fourier et Institut National Polytechnique de Grenoble.

[Mader 88] S. Mader, W. Au & M. Bazakos (1988). A Scene Interpretation System. *SPIE Vol. 937 Applications of Artificial Intelligence VI.* pp 117-123.

[Maitre 91] H. Maitre (1991). Segmentation d'images, Notes de cours ENST Paris.

[Maître 90] B. Maître & H. Laâsri (1990). Cooperating Expert Problem-Solving in Blackboard Systems : Atome Case Study. *Decentralized A.I.* pp 251-263. Demazeau & Müller Eds. Elsevier Science Publishers.

[Marr 82] D. Marr (1982). VISION : a computational investigation into the human representation and processing of visual information, W.H. Freeman and Company, San Fransisco.

[Martelli 76] A. Martelli (1976). An application of heuristic search methods to edge and contour detection. *Com. ACM*, 19:73-83.

[Martin 77] W.A. Martin (1977). OWL. In Proceedings of the 5th IJCAI, pp 985-987, Cambridge, MA.

[Maruichi 90] T. Maruichi, M. Ichikawa & M. Tokoro (1990). Modeling Autonomous Agents and their Groups. *Decentralized A.I.* pp 215-249. Demazeau & Müller Eds. Elsevier Science Publishers.

[Masini 90] G. Masini, A. Napoli, D. Colnet, D. Léonard & K. Tombre (1990). Les langages à objets. IIA. InterEditions.

[Matsuyama 85] T. Matsuyama & V. Hwang (1985). "SIGMA: a framework for image understanding - integration of bottom-up anf top-down analyses," *Proc. IJCAI*, vol. 2, pp. 908-915.

[McDermott 72] D.V. McDermott & G.J. Sussman (1972). The CONNIVER Reference Manual. AI Memo 259a, AI Lab, MIT, Cambridge, MA.

[McKeown 85] D. M. McKeown, J. R. Wilson & J. McDermott, "Rule-based interpretation of aerial imagery," *IEEE Trans. Pattern Anal. Machine Intell.*, vol PAMI-7, no. 5, pp 570-585, 1985.

[Melkemi 91] M. Melkemi (1991). Approches géométriques par modèles de voronoï en segmentation d'images. Thèse de Doctorat, Université Joseph Fourier, Grenoble.

[Meyer 86] B. Meyer (1986). Eiffel : programming for reusability and extendibility. ACM SIGPLAN Notices, 22(2):85-94.

[Minsky 75] M. Minsky (1975). A framework for representing knowledge. In P. Winston, editor, The Psychology of Computer Vision, pp 211-281, McGraw-Hill, New-York.

[Mitiche 88] A. Mitiche, A. Mansouri & C. Meubus (1988). A knowledge-based image interpretation system. *9th Int Conf on Pattern Recognition.* pp 992-994. IEEE.

[Mohr 88] R. Mohr (1988). Sur l'appariement modèle-perception. Actes du 2nd Atelier Scientifique TIPI, Chap. XXIX, (CNRS, LIFIA, IOTA, LSS).

[Montanari 71] U. Montanari (1971). On the optimal detection of curves in noisy pictures. *Com. ACM*, 14:335-345.

[Montanvert 87] A. Montanvert (1987). Contribution au traitement de formes discrètes. Squelettes et codage par graphe de la ligne médiane. Thèse de l'université Joseph Fourier, Grenoble.

[Montanvert 91] A. Montanvert, P. Meer & A. Rosenfeld (1991). Hierarchical image analysis using irregular tessellations. *IEEE Trans PAMI 13(4)*, pp 307-316, IEEE.

[Montgomery 92] T.A. Montgomery, J. Lee, D. Musliner, E.H. Durfee, D. Damouth, Y. So & UM-DIAG Group (1992). MICE Users Guide, January 92.

[Moon 86] D. Moon (1986). Object-oriented programming with flavors. In Proceedings of the 1st OOPSLA, pp 1-8, Portland, Oregon.

[Nagao 80] M. Nagao & T. Matsuyama (1980). "A structural analysis of complex aerial photographs," New-York: Plenum Press.

[Nagao 84] M. Nagao (1984). Control Strategies in Pattern Analysis. *Pattern Recognition,* 17 (1):45-56.

[Narayanan 83] K.A. Narayanan, D.P. O'Leary & A. Rosenfeld (1983). Multi-resolution relaxation. *Pattern Recog.*, 16:223-230.

[Nazif 84] A. M. Nazif & M. D. Levine, "Low level image segmentation: an expert system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol PAMI-6, no. 5, pp 555-577, 1984.

[Nugues 89] P. Nugues & J.P. Haton (1989). Un Système multi-expert pour l'interprétation d'images d'électrophorèses 2-D. AFCET.

[Ovalle 91] D.A. Ovalle (1991). Contribution à l'étude du raisonnement en univers Multi-Agent : KIDS, une application pour l'Interprétation d'Images Biomédicales. Thèse de l'Université Josehp Fourier - Grenoble I.

[Parvin 84] B.A. Parvin (1984). A split and merge algorithm for segmentation of natural scenes. *Proc. of the 7th ICPR*, pp. 294-295, IEEE Computer Society Press.

[Pavlidis 90] T. Pavlidis & Y.T. Liow (1990). Integrating Region Growing and Edge Detection, *IEEE Trans PAMI 12(3)*, pp 226-233. IEEE.

[Pesty 89] S. Pesty & C. Garbay (1989). MAPS: A Multi-Agent Problem Solving Environment. *Proc. IASTED Int. Symp. on Expert System Theory and Application*, pp 110-113, Acta Press.

[Pleiad 92] PLEIAD : Pôle et Lieu d'Echange en Intelligence Artificielle Distribuée (1992). Définition Taxonomique du Vocabulaire utilisé en Intelligence

Artificielle Distribuée. Groupe de Travail Grenoblois sur l'IAD. Rapport de Recherche Interne.

[Preparata 88] F.P. Preparata & M.I. Shamos (1988). Computational Geometry, an introduction, *Texts and Monographs in Computer Science,* Springer Verlag Ed.

[Prewitt 70] J.M.S Prewitt (1970). Object Enhancement and Extraction. *Picture Processing and Psychopictorics*, B.S. Likin and A. Rosenfeld, Academic Press, pp 75-149.

[Rao 88] A. R. Rao & R. Jain, "Knowledge representation and control in computer vision system," *IEEE Expert.*, pp. 64-79, Spring 1988.

[Rechenmann 85] F. Rechenmann (1985). SHIRKA : mécanismes d'inférences sur une base de connaissances centrée objets. Actes du 5ème CARFIA, pp 1243-1254, Grenoble.

[Regazzoni 91] V. Murino & C.S. Regazzoni (1991). A Distributed Algorithm for Adaptative Regulation of Image Processing Parameters. *1991 IEEE / SMC Int. Conf. on Systems, Man, and Cybernetics.* Vol 1, pp 259-264. IEEE.

[Reinhardt 82] E.R. Reinhardt , W.E. Blanz & al. (1982). Automated classification of cytological specimen based on multi-stage pattern recognition. *Proc. of the 7th ICPR*, pp. 153-159, IEEE Computer Society Press.

[Roche 89] C. Roche & J.P. Laurent (1989). Les approches objets et le langage LRO2 (KEOPS). Techniques et sciences informatiques, 8(1):21-39.

[Rosenfeld 82] A. Rosenfeld & A.C. Kak (1982): "*Digital Picture Processing*". Academic Press.

[Rosenfeld 83] A. Rosenfeld (1983). Quadtrees and pyramids: hierarchical representations of images. In "Pictorial Data Analysis", pp. 29-42, (R.M. Haralick, eds), Springer Verlag.

[Sacerdoti 77] E.D. Sacerdoti (1977). A structure for plans and behavior, Elsevier, New York.

[Schachter 78] B. Schachter (1978). Decomposition of Polygons into Convex Sets, *IEEE Trans on Computers* 27(11), pp 1078-1082.

[Seel 91] N. Seel (1991). Intentional Description of Reactive Systems. *Decentralized A.I 2.* pp 15-33. Demazeau & Müller Eds. Elsevier Science Publishers.

[Shapiro 83] L.G. Shapiro (1983). Computer Vision Systems: past, prsent and future. *In "Pictorial Data Analysis"*, pp. 199-237, (R.M. Haralick, eds), Springer Verlag.

[Shen 86] J. Shen & S. Castan (1986). An Optimal Linear Operator for Edge Detection. *Proc of CVPR'86.* pp 109-114.

[Sian 91] S. Sian (1991). Adaptation based on Cooperative Learning in Multi-Agent Systems, *Decentralized AI 2,* pp 257-272, Demazeau & Müller eds, North-Holland/Elsevier.

[Smith 80] R.G.Smith & R. Davis (1980). Framework for co-operation in distributed problem solving, *IEEE Trans Syst Man Cyber.* SMC-11, 1, pp 61-70.

[Souvigné 83] Souvigné (1983). PVV - Un système d'interprétation d'images par prédiction / vérification. Thèse de 3° cycle. INP Grenoble.

[Sperry 61] R.W. Sperry (1961). Cerebral Organization and Behavior. *Science,* pp. 1749-1757. N° 133.

[Steels 90] L. Steels (1990). Cooperation between Distributed Agents through Self-Organisation. *Decentralized A.I.* pp 175-196. Demazeau & Müller Eds. Elsevier Science Publishers.

[Stefik 79] M.J. Stefik (1979). An examination of a frame-structured representation system. In Proceedings of the 6th IJCAI, pp 845-852, Tokyo.

[Stroustrup 86] B. Stroustrup (1986). The C++ Programming Language. Addison-Wesley Series in Computer Science, Reading, Massachussets.

[Stroustrup 87] B. Stroustrup (1987). The C++ Programming Language. Addison-Wesley Publishing Company, Inc.

[Sueyoshi 91] T. Sueyoshi & M. Tokoro (1991). Dynamic Modeling of Agents for Coordination. *Decentralized A.I 2.* pp 161-176. Demazeau & Müller Eds. Elsevier Science Publishers.

[Tan 86] C.L. Tan & W.N. Martin (1986). A Distributed System for Analysing Time-Varying Multiresolution Imagery. *Computer Vision, Graphics and Image Processing 36.* pp 162-174.

[Tan 89] C.L. Tan & W.N. Martin (1989). An Analysis of a Distributed Multiresolution Vision System. *Pattern Recognition. Vol 22, No. 3.* pp 257-265.

[Tanimoto 75] S. Tanimoto & T. Pavlidis (1975). A hierarchical data structure for picture processing, *CGIP 4,* pp 104-119.

[Tehrani 89] S. Tehrani, T.E. Weymouth & G.B.J. Mancini (1989). An Application of the Blackboard Architecture to Left Ventricular Boundary Detection. *SPIE Vol. 1095 Applications of Artificial Intelligence VII.* pp 503-514.

[Theriault 83] D.G. Theriault (1983). Issues in the design and implementation of Act 2. Technical Report 728, IA Lab, MIT, Cambridge, MA.

[Thonnat 89] M. Thonnat & Bijaoui (1989). Knowledge-based classification of galaxies. *Knowledge-Based Systems in Astronomy,* F. Murgath & A. Heck Eds, pp 121-159, Springer-Verlag.

[Tilton 84] J.C. Tilton (1984). Multi-resolution spatially constrained clustering of remotely sensed data on the massively parallel processor. *Proc. of the 7th ICPR,* pp. 1013-1015, IEEE Computer Society Press.

[Tucker 84] L.W. Tucker (1984). Model-guided segmentation using quadtrees. *Proc. of the 7th ICPR,* pp. 216-219, IEEE Computer Society Press.

[Vogel 88] C. Vogel (1988). Génie Cognitif, Collection Sciences Cognitives, MASSON.

[Woods 89] P. W. Woods, D. Pycock & C. Taylor (1989). "A frame-based System for Modelling and Executing Visual Tasks," *Image and Vision Computing*, pp. 102-108.

[Woolridge 91] M. Woolridge, G. O'Hare & R. Elks (1991). FELINE : A Case Study in the Design and Implementation of a Co-operating Expert System. *Proc 11th Int Workshop on Expert Systems and their Applications.* Avignon May 1991.

[Wright 84] J.M. Wright, M.S. Fox & D.L. Adam (1984). SRL2 User's Manual. Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

[Wrobel 87] B. Wroebel & O. Monga (1987). Segmentation d'images naturelles. Coopération entre un détecteur de contours et un détecteur de régions. *11ème Colloque GRETSI,* Nice.

[Xu 84] G.Y. Xu & K.S. Fu (1984). Natural scene segmentation based on multiple threshold and textural measurements. *Proc. of the 7th ICPR*, pp. 1111-1113, IEEE Computer Society Press.

[Yonezawa 86] A. Yonezawa, J.P. Briot & E. Shibayama (1986). Object-oriented concurrent programming in ABCL/1. *In Proceedings of the 1st OOPSLA*, pp 258-268, Portland, Oregon.

[Zanconato 88] R. Zanconato (1988). BLOBS - An object-oriented blackboard system framework for reasoning in time. *Blackboard Systems*, Addison-Wesley.

[Zhang 84] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Comm of the ACM.*, vol. 3, no. 27, pp. 236-239, 1984.

Conception d'un Environnement de Développement pour la Résolution de Problèmes :

Apport de l'Intelligence Artificielle Distribuée et Application à la Vision

Résumé : L'objectif de cette thèse a été de concevoir un environnement de développement pour la résolution de problèmes s'inspirant des langages à objets et de l'Intelligence Artificielle Distribuée. Un environnement de programmation multi-agents a été conçu dans ce cadre. Cet environnement (MAPS) est fondé sur la distinction de deux classes d'agents dédiées à la modélisation et à l'exploitation des connaissances figuratives (agent KS ou Knowledge Server) et des connaissances opératoires (agent KP ou Knowledge Processor). Ces classes peuvent alors être spécialisées à l'aide d'un langage de programmation dédié.

Une première version de cet environnement nous a permis de valider ces concepts de base, par la conception d'applications en Vision par Ordinateur (système KISS), en Diagnostic Biomédical (système KIDS), en Compréhension de la Parole et en Acquisition des Connaissances. Le système KISS, décrit dans cette thèse, a été conçu pour manipuler les connaissances introduites par les phases d'analyse de bas niveau, de niveau intermédiaire et de haut niveau, dans une approche de type coopération région / contour. Dans notre approche multi-agents, ces connaissances sont distribuées au sein d'agents KS et KP, formant un réseau.

Plusieurs améliorations ont été envisagées pour pallier un certain nombre de faiblesses de cet environnement et le rapprocher des plates-formes de conception en Intelligence Artificielle Distribuée. Des améliorations fonctionnelles concernent l'introduction d'un mode d'exécution parallèle. Ces améliorations ont conduit à la définition d'un nouvel environnement. Des améliorations structurelles, encore à l'étude, concernent la définition d'agents hétérogènes et une structuration en couche des agents.

Abstract : The aim of this work was to draw an environment dedicated to problem solving, from object oriented languages and Distributed Artificial Intelligence features. A multi-agent programming environment was conceived in this framework. This environment (MAPS) is based on two main agent classes, dedicated to the modelling and the processing of figurative knowledge (KS agent or Knowledge Server) and operative knowledge (KP agent or Knowledge Processor). These classes can be specialized using a dedicated programming language.

A first version of this environment allowed to validate these basic concepts, with the design of several applications in Computer Vision (KISS system), in Biomedical Diagnosis (KIDS system), in Speech Understanding and in Knowledge Elicitation. The KISS system, described in this thesis, was conceived to handle knowledge introduced at the low level, the intermediate level and the high level analysis step, using a region / contour cooperation way. Using our approach, this knowledge is distributed in KS or KP agents, organized into a network.

Several improvements have been envisaged, to palliate some drawbacks of this environment. Functional improvements concern the insertion of a parallel execution mode. These improvements have led to the design of a new environment. Structural improvements, still under study, concern the design of heterogeneous agents, and a multi-layered structuration of agents.

Mots-clés : Intelligence Artificielle Distribuée, Architecture Multi-Agents, Environnement de Programmation, Vision par Ordinateur, Segmentation d'Images, Coopération Région / Contour, Interprétation d'Images