rsité
de Nancy 1

Inria-Lorraine

Centre de Recherche en
Informatique de Nancy

Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et réalisations

THÈSE

présentée et soutenue publiquement le **20 Février 1989**

pour l'obtention du

Doctorat de l'Université de Nancy 1

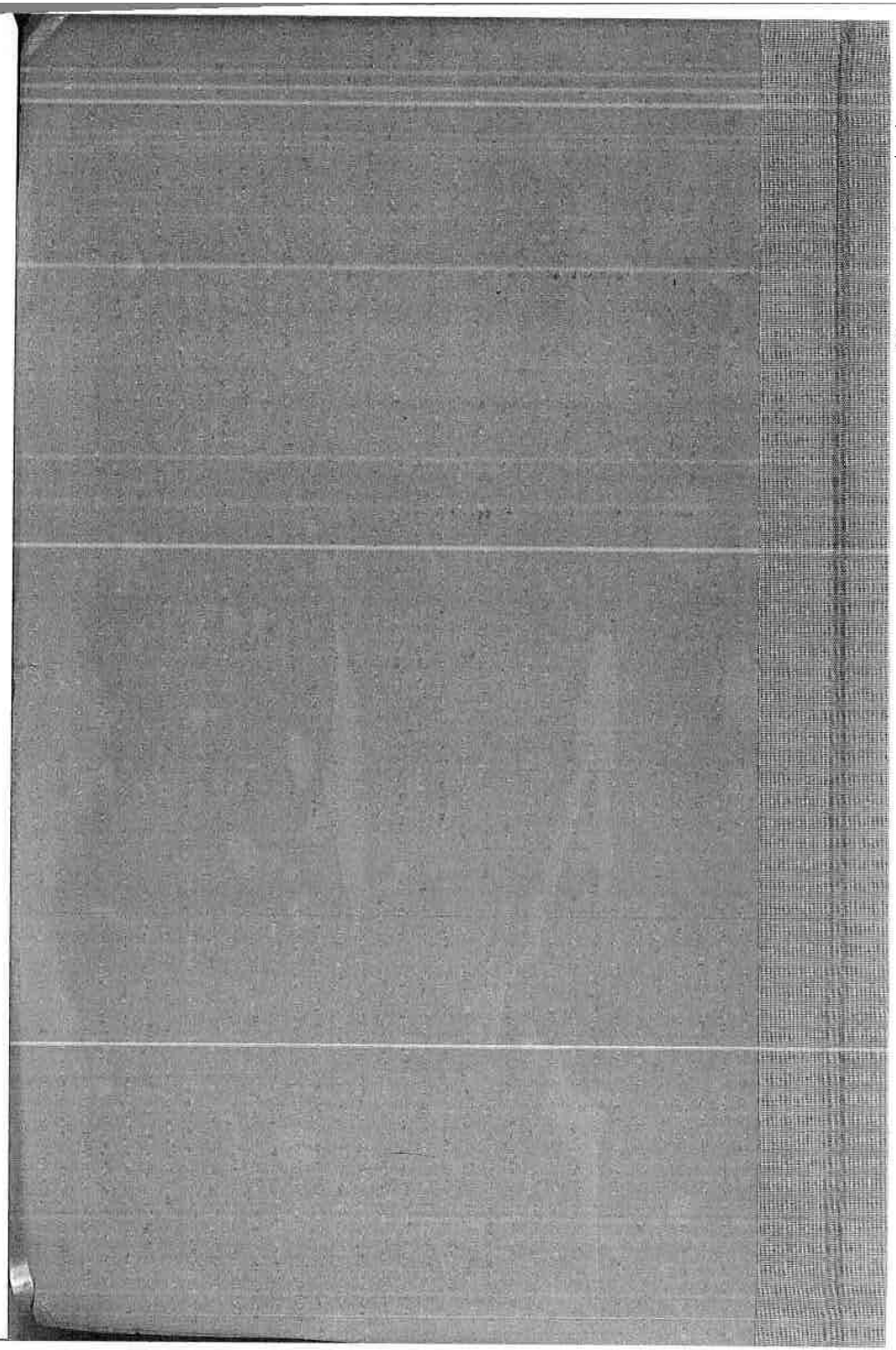
(Spécialité Informatique)

par

Hassan LÂASRI et Brigitte MAÎTRE

Composition du jury :

Président :	Roger MOHR
Rapporteurs :	Malik GHALLAB Marion CRÉHANGE
Examinateurs :	Jean-Paul HATON Marie-Christine HATON
Invités :	Michel CLERGET Philippe LE BLÉ



Université de Nancy 1 — Inria-Lorraine — Centre de Recherche en Informatique de Nancy

Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et réalisations

THÈSE

présentée et soutenue publiquement le **20 Février 1989**

pour l'obtention du

Doctorat de l'Université de Nancy 1

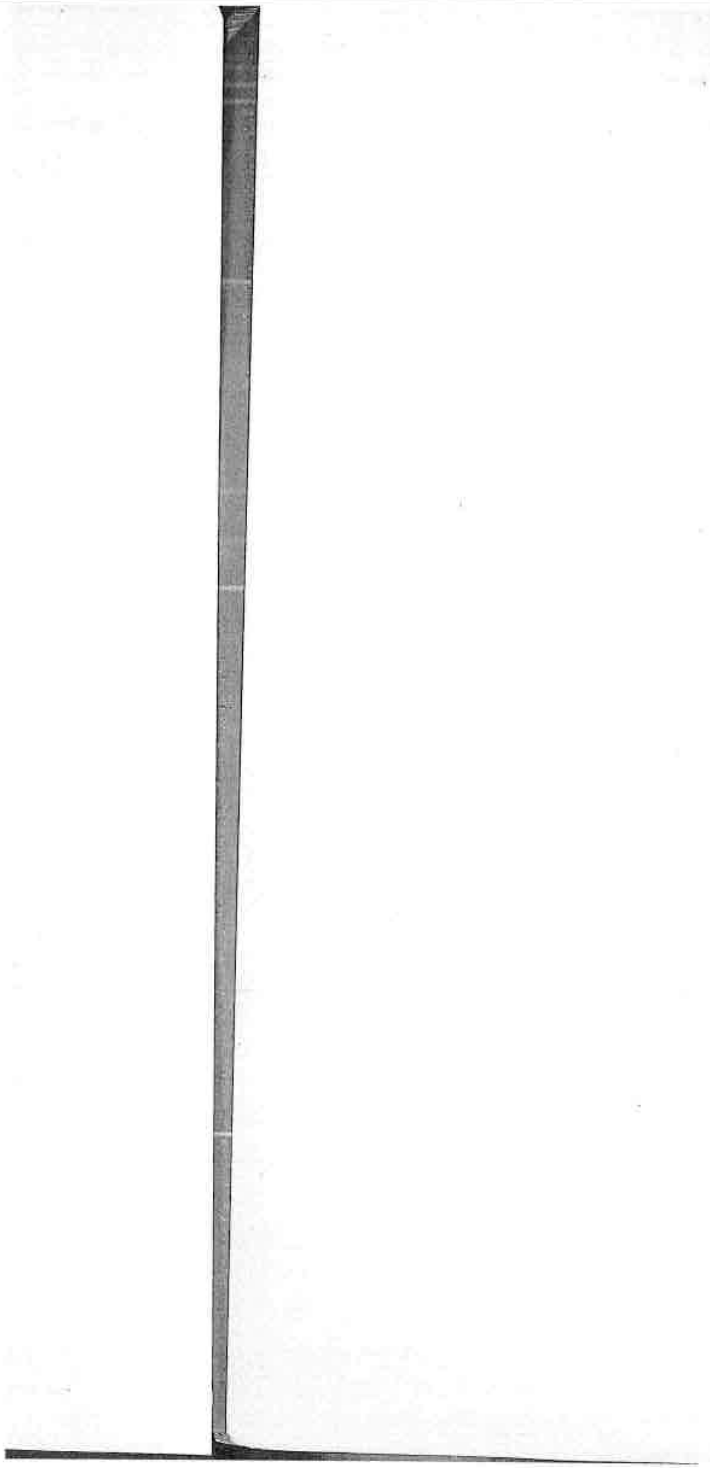
(Spécialité Informatique)

par

Hassan LÂASRI et Brigitte MAÎTRE

Composition du jury :

Président :	Roger MOHR
Rapporteurs :	Malik GHALLAB Marion CRÉHANGE
Examinateurs :	Jean-Paul HATON Marie-Christine HATON
Invités :	Michel CLERGET Philippe LE BLÉ



A nos familles

NOM DE L'ETUDIANT : Hassan LAASRI et Brigitte MAITRE

NATURE DE LA THESE : Doctorat de l'Université de NANCY I en Informatique

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le - 9 FEV. 1989 n° 305

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

Avis aux lecteurs

Le projet présenté dans ce document résulte des travaux que nous avons réalisés ensemble lors de nos deux thèses respectives. Aussi, afin de fournir au lecteur un dossier scientifique complet et concis, intégrant nos reflexions fondamentales ainsi que nos réalisations pratiques, et afin d'éviter une présentation décousue des résultats de nos travaux, nous avons choisi de rédiger un mémoire unique, commun, volumineux et très dense.

Le lecteur désirant connaître la contribution personnelle de chacun pourra se reporter aux deux documents complémentaires intitulés "Contribution personnelle" [160,185].

Brigitte & Hassan

Remerciements

Nous tenons particulièrement à remercier :

- M. Jean-Paul Haton, Professeur à l'Université de Nancy 1, qui nous a accueillis au sein de l'équipe "Reconnaissances des Formes et Intelligence Artificielle" du CRIN. Il nous a proposé un sujet de recherche passionnant et nous a encadrés et orientés tout au long de ce travail. Nous le remercions de la confiance qu'il nous a toujours accordée, de ses conseils, de sa patience et de sa disponibilité.

- M. Roger Mohr, Professeur à l'Institut Polytechnique de Grenoble, de nous avoir fait l'honneur de présider ce jury. Nous lui sommes particulièrement reconnaissants pour l'intérêt qu'il a toujours porté à nos travaux et qui s'est manifesté notamment à travers de fructueuses discussions.

- M. Malik Ghallab, Chargé de Recherches CNRS Première Classe au LAAS à Toulouse, qui a accepté d'être rapporteur de ce mémoire et de siéger à ce jury. Qu'il trouve ici l'expression de notre gratitude pour l'intérêt dont il fait preuve à l'égard de ce travail.

- Mme Marion Créhange, Professeur de l'Université de Nancy 2, qui a accepté d'être rapporteur de ce document. Nous la remercions pour sa gentillesse et son intérêt pour notre travail.

- Mme Marie-Christine Haton, Maître de Conférences à l'Université de Nancy 1, qui nous a fait découvrir le monde de la recherche lors de nos études au second cycle. Nous la remercions également pour ses encouragements tout au long de notre thèse.

- M. Michel Clerget, Directeur des Etudes et de la Production de COGNITECH à Paris, et M. Philippe Le Blé, ingénieur au GERDSM à Toulon, pour l'honneur qu'ils nous font en acceptant de juger notre travail malgré les nombreuses charges et responsabilités qu'ils assument. Nous leur sommes reconnaissants pour le soutien qu'ils nous ont manifesté durant l'élaboration d'ATOME.

Merci à M. Jean-Marie Proth, directeur de l'INRIA-Lorraine, pour sa sympathie et son soutien.

Merci aussi à M. Jean-Marie Pierrel, Professeur à l'Université de Nancy 1, pour ses conseils et ses encouragements.

Nous voulons également remercier tous nos collègues du CRIN pour l'atmosphère de travail agréable dont nous avons bénéficié. Nous tenons également à remercier ceux avec qui nous avons collaboré à un moment ou un autre, notamment tous les membres du Groupe "ATOME". Merci aussi à Karl Tombre pour la lecture de ce document. Ses conseils et ses critiques ont été très utiles pour son amélioration. Un grand merci également à Stephan Brunessaux pour la lecture de ce document et surtout pour son importante contribution dans ATOME. Il a porté ATOME sur Machine-Lisp Symbolics en Common Lisp. Ses idées et ses remarques ont permis de perfectionner ATOME.

Enfin, nous remercions toutes les personnes qui s'intéressent de près ou de loin à ATOME.

Résumé

Notre travail s'inscrit dans le cadre du projet SYCO du CRIN/INRIA-Lorraine au sein de l'équipe RFIA (Reconnaissance des Formes et Intelligence Artificielle). Ce projet concerne la réalisation de systèmes à base de connaissances et leur application à la compréhension.

Cette thèse présente le résultat de notre travail sur la résolution de problèmes nécessitant la coopération de plusieurs agents "intelligents". Nous traitons plus particulièrement le cas où cette coopération est réalisée par partage d'informations à travers une zone de données commune aux différents agents (modèle du blackboard). Pour être mené à bien, ce type de coopération nécessite un mécanisme de contrôle sophistiqué capable de gérer de manière efficace et souple l'interaction entre ces agents. Ceci a constitué l'essentiel de nos travaux de recherche durant notre thèse.

Dans ce document, nous présentons d'abord les différentes approches ou réalisations déjà existantes dans ce domaine en les classant par *type de contrôle* et en évaluant leurs avantages et limites respectives. Le *contrôle procédural*, par exemple, est efficace mais rigide et très dépendant de l'application. Il est illustré dans la thèse par les systèmes HEARSAY-II et DVMT. Le *contrôle hiérarchique* caractérisé par un ensemble de sources de connaissances de contrôle hiérarchisées est quant à lui illustré par les systèmes HASP/SIAP et CRYSALIS. Le *contrôle à base de blackboard* représente le contrôle dans une architecture de blackboard par une autre architecture de blackboard. Il est illustré par les systèmes BB-1 et GBB-1.

A partir de cet état de l'art, nous mettons en évidence quatre facteurs discriminants des architectures de blackboard (efficacité, uniformité, souplesse et facilité d'expression). Par référence à ces facteurs, nous présentons l'approche *contrôle hybride à multi-phases* que nous avons définie en réalisant ATOME, outil d'aide au développement de systèmes multi-agents. Cette approche été mise en œuvre en deux étapes décrites dans cette thèse.

A l'issue de ces deux étapes, l'architecture d'ATOME est organisée en quatre plans hiérarchiques :

1. les *blackboards* qui, à chaque instant de la résolution du problème, contiennent les résultats fournis par les différents agents. Ces blackboards contiennent également les données initiales du problème;
2. les *spécialistes* qui renferment les connaissances du domaine et constituent les agents du système. Une spécialiste est une source de connaissances du domaine.
3. les *tâches* qui fournissent un *contrôle local* au système en gérant un sous-ensemble de spécialistes. Une tâche est une source de connaissances de contrôle disposant d'une structure de contrôle locale (liste d'événements) et d'un mode de raisonnement (dirigé par les événements, dirigé par les règles ou opportuniste). Chaque tâche applique donc un contrôle particulier adapté au sous-problème qu'elle doit traiter, d'où l'appellation *contrôle hybride*.
4. une *stratégie* qui fournit un *contrôle global* au système en gérant l'ensemble des tâches. La stratégie est une méta-source de connaissances de contrôle disposant d'un résumé des blackboards pour évaluer l'état d'avancement de la résolution du problème et faire évoluer le système de phase en phase en activant des tâches, d'où l'appellation "contrôle hybride *à multi-phases*".

Nous présentons ensuite diverses extensions effectuées sur ATOME afin de le rendre convivial, souple et paramétrable dans son utilisation.

Nous terminons cette thèse par un bilan complet sur ATOME, son utilisation et son avenir et par nos réflexions sur le modèle du blackboard et les axes de recherche intéressants à poursuivre dans ce domaine.

Table des matières

Avis aux lecteurs III

Remerciements V

Résumé VII

Introduction générale 1

I Les systèmes multi-agents en IA 5

1 Les univers multi-agents en IA 9
- 1.1 Principe des systèmes multi-agents 9
  - 1.1.1 La communication dans un univers multi-agents 10
  - 1.1.2 Le contrôle dans un univers multi-agents 11
- 1.2 Apport du concept multi-agents en IA 13

2 Le modèle du blackboard 15
- 2.1 Présentation du modèle 15
  - 2.1.1 Historique 16
  - 2.1.2 Le Modèle 19
- 2.2 Notion d'événement 23
- 2.3 Problèmes de type blackboard 24
- 2.4 Conclusion 25

II Architectures de blackboard 27

1 Le contrôle dans une architecture de blackboard 31

1.1 Rôle 32
1.2 Focalisation de contrôle 35
1.3 Représentation du contrôle 37

2 Contrôle procédural **39**
2.1 Principe 39
2.2 HEARSAY-II 41
2.2.1 Le blackboard 41
2.2.2 Les sources de connaissances 41
2.2.3 Le contrôle 45
2.2.4 Conclusion 48
2.3 DVMT 48
2.3.1 Système IA distribué 50
2.3.2 Architecture d'un nœud de DVMT 51
2.3.3 Fonctionnement d'un nœud de DVMT 53
2.4 Conclusion 53

3 Contrôle hiérarchique **57**
3.1 Principe 57
3.2 HASP/SIAP 57
3.2.1 Le blackboard 59
3.2.2 Les sources de connaissances 59
3.2.3 Le contrôle 59
3.3 CRYSALIS 62
3.3.1 Le blackboard 62
3.3.2 Les sources de connaissances 63
3.3.3 Le contrôle 63
3.4 Conclusion 65

4 Contrôle à base de blackboard **69**
4.1 Principe 69
4.2 BB-1 71
4.2.1 Le blackboard 71
4.2.2 Les sources de connaissances 72
4.2.3 Le contrôle 75
4.3 GBB 78

4.3.1 Spécification et implantation du blackboard 79
4.3.2 Environnement de contrôle 80
4.4 Avantages du contrôle à base de blackboard 81
4.5 Limites du contrôle à base de blackboard 82
4.6 Conclusion 83

5 Conclusion **85**

III Le projet ATOME **87**

1 Introduction **91**

2 Efficacité et facilité d'expression dans ATOME **93**
2.1 Approche générale 93
2.2 Structure du blackboard 95
2.3 Evénements 98
2.4 Les sources de connaissances 99
2.5 Le mécanisme de contrôle 102
2.5.1 Structures de contrôle 102
2.5.2 Les tâches 104
2.5.3 La stratégie 110
2.5.4 Architecture d'ATOME 112
2.5.5 Boucle de contrôle de base 113
2.6 Evaluation de la première étape 115
2.6.1 Les avantages 116
2.6.2 Les limites 117
2.7 Conclusion 119

3 Vers une architecture souple et paramétrable **121**
3.1 Approche générale 121
3.2 Les blackboards 122
3.3 Les événements 124
3.4 Les sources de connaissances 126
3.5 Structures de contrôle 130
3.5.1 Les listes d'événements 130
3.5.2 Résumés des blackboards 135

3.6 Les tâches 136
3.6.1 Tâche dirigée par les événements 136
3.6.2 Tâche dirigée par les règles 138
3.6.3 Tâche opportuniste 142
3.6.4 Comparaison des tâches 152
3.7 La stratégie 152
3.8 Mémoire à long terme 153
3.9 Architecture résultante 157
3.10 Le modèle hybride à multi-phases 161
3.11 Conclusion 163

4 Entités manipulées par ATOME **165**
4.1 Entités et opérations 165
4.2 Schéma de traduction 170

5 Le problème du voyageur de commerce avec ATOME **171**
5.1 Résolution du problème 171
5.2 Définition des blackboards 172
5.2.1 Tour-info 172
5.2.2 Solution 173
5.3 Les spécialistes 174
5.3.1 Spécialistes d'initialisation 174
5.3.2 Spécialistes de résolution 179
5.4 Les tâches 184
5.4.1 Tâche d'initialisation 184
5.4.2 Tâche de résolution 184
5.5 La stratégie 186
5.6 Exécution et première amélioration 189
5.6.1 Exécution 189
5.6.2 Apport des conditions de rejet 196
5.7 Extensions 196
5.7.1 Les blackboards 197
5.7.2 Spécialiste Init-résolution 197
5.7.3 Tâche Construire-chemin 198
5.7.4 La stratégie 198
5.7.5 Le comportement 198

5.8 Résultats 199
5.8.1 Mode opportuniste 199
5.8.2 Mode dirigé par les événements 201
5.8.3 Comparaison 202
5.9 Conclusion 202

6 Le mécanisme explicatif dans ATOME **205**
6.1 Entités de base d'une architecture de blackboard 205
6.2 Le mécanisme explicatif dans ATOME 208
6.2.1 Le blackboard explicatif 208
6.2.2 Le module explicatif 217
6.2.3 Architecture résultante d'ATOME 219
6.3 Conclusion 221

7 Représentation hétérogène des spécialistes dans ATOME **223**
7.1 Principe 224
7.2 Réalisations 227
7.3 Conclusion 232

8 Bilan d'ATOME **235**
8.1 Généricité 235
8.2 Efficacité 239
8.3 Souplesse 240
8.4 Modularité 242
8.5 Comparaison entre ATOME et BB-1 243
8.5.1 Similitudes 243
8.5.2 Différences 244
8.6 Extensions à court et moyen terme 246
8.6.1 Raisonnement dirigé par les buts 246
8.6.2 Extensions des résumés des blackboards 247
8.6.3 Stratégie opportuniste 247
8.6.4 Extensions du mécanisme explicatif 248
8.6.5 Compilation de règles 248
8.7 Extensions à long terme 248
8.8 ATOME au CRIN 250
8.8.1 Applications 250

8.8.2 Raisonnements approximatif, hypothétique et temporel dans ATOME 253
8.9 ATOME à l'extérieur du CRIN 254

Conclusion générale **255**

A Conférences spécialisées sur les architectures de blackboard **261**

B Glossaire des termes ATOME **265**

Bibliographie **271**

Index **294**

Liste des Figures

Partie I

1.1 Structure d'un système à base de blackboard. 11
1.2 Structure d'une société d'experts. 12

2.1 Exemple de deux sources d'informations coopératives. 16
2.2 Les architectures de blackboard et leurs origines. 20
2.3 Les trois composantes de base du modèle du blackboard. 22
2.4 Emission d'un événement. 23

Partie II

1.1 Le mécanisme de contrôle dans une architecture de blackboard. 33
1.2 Les SCs SC1 et SC2 sont en conflit car elles sont intéressées par la même donnée D située dans leur niveau d'entrée commun N. . 34
1.3 Les SCs SC1 et SC2 sont en conflit car elles ont deux données compétitives D1 et D2 dans leurs niveaux d'entrée respectifs N1 et N2. 35
1.4 Les SCs SC1 SC2 et SC3 sont toutes les trois en conflit à cause des données compétitives D1 et D2 situées dans leur niveau d'entrée respectifs N1 et N2. 36

2.1 Contrôle procédural. 40
2.2 Les sept niveaux du blackboard dans HEARSAY-II. 42
2.3 Les SCs dans HEARSAY-II. 43
2.4 Rôles des SCs dans HEARSAY-II. 44
2.5 Architecture du système HEARSAY-II. 46
2.6 Stratégie implicite de HEARSAY-II. 49
2.7 Domaine d'application de DVMT. 50
2.8 Le blackboard et les SCs de DVMT. 52
2.9 L'architecture de HEARSAY-II adaptée au système DVMT. 54

2.10 L'architecture d'un nœud de DVMT. 55

3.1 Contrôle hiérarchique. 58
3.2 Le blackboard et les SCs dans HASP/SIAP. 60
3.3 Organisation du système HASP/SIAP. 61
3.4 Représentation du blackboard dans CRYSALIS. 64
3.5 Les sources de connaissances de CRYSALIS. 67
3.6 Le cycle de base dans CRYSALIS. 68

4.1 Contrôle à base de blackboard. 70
4.2 Les SCs du domaine et de contrôle opérant respectivement sur les blackboards du domaine et de contrôle. 74
4.3 Les mouvements des ISCs à travers les agendas. 76
4.4 Architecture du système BB-1. 78
4.5 Structure d'un blackboard dans GBB. 79

Partie III

2.1 Approche globale d'ATOME. 94
2.2 Définition des niveaux du blackboard. 96
2.3 Hypothèses du blackboard. 97
2.4 Structure du blackboard. 98
2.5 Structure d'un événement. 99
2.6 E/S d'une spécialiste. 100
2.7 Caractéristiques d'une spécialiste. 101
2.8 Activation d'une spécialiste. 101
2.9 Liste des événements. 103
2.10 Règle type d'une tâche. 104
2.11 E/S d'une tâche. 105
2.12 Cycle de base du mode dirigé par les événements. 106
2.13 Cycle de base du mode dirigé par les règles. 107
2.14 Caractéristiques d'une tâche. 108
2.15 Activation d'une tâche dirigée par les événements. 109
2.16 Activation d'une tâche dirigée par les règles ou de la stratégie. 110
2.17 Règle type de la stratégie. 111
2.18 E/S de la stratégie. 111
2.19 Caractéristiques de la stratégie. 112

2.20 Architecture d'ATOME. 114
2.21 Boucle de contrôle de base. 115

3.1 Utilisation de blackboards locaux et globaux. 123
3.2 Création des événements systématique pour un attribut, un niveau ou un blackboard donné. 125
3.3 Création des événements à la demande des spécialistes. 125
3.4 Phases de fonctionnement des spécialistes. 128
3.5 Caractéristiques d'une spécialiste. 129
3.6 Phase de vérification de la précondition d'une spécialiste. . . . 129
3.7 Phase d'activation d'une spécialiste. 130
3.8 Liste d'événements commune à toutes les tâches. 132
3.9 Listes des événements locales. 133
3.10 Synthèse de deux événements dans la liste d'événements de la tâche T. 135
3.11 Règle type d'une tâche. 137
3.12 Caractéristiques d'une tâche. 137
3.13 Cycle élémentaire d'une tâche dirigée par les événements. . . . 139
3.14 Algorithme d'activation d'une tâche dirigée par les événements. 140
3.15 Cycle élémentaire d'une tâche dirigée par les règles. 141
3.16 Algorithme d'activation d'une tâche dirigée par les regles. . . . 142
3.17 Agenda dont les éléments sont rangés par priorité décroissante. 143
3.18 Cycle local d'une tâche opportuniste. 145
3.19 Activation d'une tâche opportuniste. 146
3.20 Phase de test des conditions de rejet d'une spécialiste. 147
3.21 Paramètres associés à la spécialiste SC1. 148
3.22 Heuristique H1. 149
3.23 Règle d'intégration R1. 150
3.24 Calcul de la priorité d'une ISP. 151
3.25 Exemple d'une règle de la stratégie. 153
3.26 Nouvelle architecture d'ATOME. 154
3.27 Blackboards à long terme. 156
3.28 Ménage du blackboard *solution*. 157
3.29 Flux de contrôle dans ATOME 159
3.30 Organisation des sources de connaissances dans ATOME. 160
3.31 Tableau de correspondance. 163

5.1 Villes à visiter. 176
5.2 Architecture du système-ATOME résolvant le problème du voyageur de commerce. 188
5.3 Etat initial. 191
5.4 Arc New York - Boston. 191
5.5 Arc Seattle - San Francisco. 192
5.6 Arc Los Angeles - San Francisco. 192
5.7 Arc New York - Miami. 193
5.8 Arc Chicago - Boston. 193
5.9 Arc Los Angeles - Las Vegas. 194
5.10 Arc Miami - Dallas. 194
5.11 Arc Seattle - Kansas City. 195
5.12 Arc Dallas - Las Vegas. 195
5.13 Arc Chicago - Kansas City. 196

6.1 Relations entre les entités fondamentales d'une architecture de blackboard. 207
6.2 Mécanisme explicatif dans ATOME. 209
6.3 Relations entre la stratégie, les résumés des blackboards et les tâches. 210
6.4 Relations entre une tâche dirigée par les événements ou par les règles, sa liste d'événements et les spécialistes. 211
6.5 Relations entre une tâche opportuniste, sa liste d'événements et les agendas des instances de spécialistes. 212
6.6 Relations entre une spécialiste, les blackboards et les événements. 213
6.7 Blackboard explicatif à cinq niveaux. 214
6.8 Le blackboard explicatif dans ATOME. 216
6.9 Création d'un événement. 218
6.10 Utilisation d'un événement. 219
6.11 Architecture d'ATOME. 220

7.1 Accès au blackboard dans une architecture homogène. 225
7.2 Accès au blackboard dans une architecture hétérogène. 226
7.3 Récupération et transformation d'objets du blackboard. 228
7.4 Transformation et stockage de paramètres SAGANE dans le blackboard. 229
7.5 Structure d'une spécialiste hétérogène dans ATOME. 230

7.6 Architecture hétérogène d'ATOME. 231

8.1 Communication par partage d'informations dans ATOME. . . . 236
8.2 Communication par envoi d'informations dans ATOME. 238
8.3 Similitudes entre le mécanisme de contrôle de BB-1 et ATOME. . 244

Introduction générale

Le but des recherches en intelligence artificielle (IA) est de tenter de construire des systèmes et des machines qui effectuent des tâches relevant de l'intelligence humaine [209]. Parmi les premiers problèmes qui ont été abordés en IA, figurent les problèmes de jeux et de démonstration de théorèmes [171]. Les progrès de la recherche en IA ont permis d'aborder des problèmes liés à la perception (vision et parole), à la compréhension des langages naturels, à l'acquisition, à la représentation et à l'apprentissage des connaissances, aux techniques de raisonnement, et à la résolution de problèmes dans des domaines spécialisés [221,34].

Pour résoudre un problème, un système d'IA[1] dispose de connaissances sur le domaine qu'il traite et éventuellement de données qui traduisent la situation initiale du problème. Ensuite il construit la solution en appliquant certaines de ces connaissances sur les données initiales ou sur des résultats partiels. La résolution d'un problème par un système d'IA peut être effectuée soit par un seul agent soit par plusieurs agents. Le premier cas suppose que l'agent ait une vue globale du problème et possède donc toutes les capacités et les connaissances pour résoudre ce problème. En revanche, dans le second cas, chaque agent est supposé ne disposer que d'une vue partielle du problème et n'est capable de résoudre qu'une partie de celui-ci. Chaque agent est donc amené (ou contraint) à coopérer avec les autres agents en échangeant avec eux des informations pouvant contribuer à la résolution du problème. Ce second cas fait référence aux systèmes d'IA appelés communément *systèmes multi-agents "intelligents"* et constitue le cadre de travail de notre thèse.

A propos

Nos travaux de recherche ont porté sur l'élaboration d'un mécanisme de contrôle sophistiqué pour gérer la coopération de plusieurs agents intelligents dans un système multi-agents. Nous nous sommes plus particulièrement in-

1. Système utilisant des techniques d'intelligence artificielle.

téressés à une coopération par partage d'informations à travers une zone de communication de type *blackboard*[2].

Les approches ou réalisations déjà existantes dans ce domaine ont montré le besoin de poursuivre les recherches dans ce sens. Nous avons en effet recensé et évalué les différents types de contrôle existants en les classant en trois catégories et en énumérant leurs avantages et limites respectifs.

Nous avons alors défini un autre type de contrôle, *hybride à multi-phases*, corrigeant les limites rencontrées dans les approches précédentes tout en intégrant leurs avantages. Ce type de contrôle a été mis en œuvre dans un outil d'aide au développement de systèmes multi-agents appelé ATOME.

Organisation de la thèse

Notre thèse est composée de trois parties :

1. La première partie fournit une présentation générale des systèmes multi-agents en IA. Elle introduit ensuite les systèmes à base de blackboard.

   D'une part, dans cette partie nous présentons les différents moyens mis en œuvre pour permettre le transfert d'informations entre les agents d'un système multi-agent. D'autre part, nous citons les approches généralement adoptées pour gérer les conflits d'intervention liés à la présence de plusieurs agents pour résoudre un même problème.

   Nous nous intéressons ensuite plus particulièrement aux systèmes multi-agents respectant le modèle du blackboard pour gérer la communication et la coordination entre leurs agents. Nous présentons d'abord ce modèle, puis nous donnons un bref historique des différents systèmes et domaines d'application où ce modèle a été utilisé avec succès depuis 1971[3] jusqu'à ce jour (1988).

   Nous terminons cette partie en caractérisant les domaines d'application pour lesquels une architecture de blackboard est appropriée.

2. La seconde partie de cette thèse aborde le problème de contrôle dans les architectures de blackboard.

   Nous commençons cette partie par une présentation générale du mécanisme de contrôle et de son rôle dans les systèmes d'IA et plus particulièrement dans une architecture de blackboard.

2. Bien que le terme *blackboard* puisse être traduit par *tableau noir*, nous avons opté ici pour le mot anglais couramment utilisé dans la communauté scientifique francophone.
3. Année de naissance des systèmes à base de blackboard.

   Nous présentons ensuite les trois types de contrôle que nous avons rencontrés dans les systèmes basés sur le modèle du blackboard : le *contrôle procédural* et son implantation dans les systèmes HEARSAY-II et DVMT, le *contrôle hiérarchique* et sa mise en œuvre dans les systèmes HASP/SIAP et CRYSALIS, et le *contrôle à base de blackboard* et son implantation dans les systèmes BB-1 et GBB. Chacun de ces types de contrôle est évalué et présente des avantages et limites décrits également dans cette partie.

3. La troisième partie décrit de façon détaillée notre contribution dans le domaine des architectures de blackboard. Elle présente en particulier la démarche que nous avons suivie pour réaliser l'outil ATOME. Nous décrivons d'abord la première réalisation que nous avons effectuée ainsi que ses avantages et ses limites. Nous présentons ensuite la seconde réalisation mise en œuvre en mettant en évidence les limites corrigées.

   Nous donnons ensuite un exemple complet d'utilisation d'ATOME afin de permettre au lecteur de bien comprendre les mécanismes de base d'ATOME et de donner un aperçu de sa syntaxe.

   Nous terminons cette partie, par une description des extensions effectuées sur ATOME afin de le rendre convivial, souple et paramétrable dans son utilisation.

   Nous terminons cette partie par un bilan complet sur ATOME, son utilisation et son avenir.

Suite à l'expérience que nous avons acquise lors des études bibliographiques, des discussions que nous avons eues durant des congrès ou des manifestations spécialisées[4] et des réalisations effectuées sur ATOME, nous présentons en conclusion les axes de recherche qui nous semblent les plus prometteurs dans le domaine des architectures de blackboard.

A la fin du document, nous avons ajouté deux annexes. La première résume les différents thèmes abordés au cours des trois conférences spécialisées qui ont eu lieu sur les architectures de blackboard. La seconde fournit un glossaire des termes ATOME.

Comment parcourir cette thèse

Cette thèse présente, d'une part, l'état de l'art dans le domaine des architectures de blackboard en citant les principales réalisations dans ce domaine, leurs avantages et leurs faiblesses. Elle décrit d'autre part nos travaux dans ce

4. Conférences spécialisées sur les architectures de blackboard [3,2] ou visites de laboratoires de recherche.

domaine en précisant les objectifs que nous avons visés et en montrant comment nous les avons atteints.

Le lecteur intéressé uniquement par des généralités concernant les approches multi-experts, leur apport en IA, les problèmes qu'elles soulèvent, et les modèles utilisés pour résoudre ces problèmes, est invité à consulter les chapitres 1 et 2 de la première partie.

Celui intéressé par l'architecture de blackboard comme modèle d'organisation et d'exploitation des connaissances, et par son implantation dans des systèmes d'IA pourra consulter le chapitre 1 et 2 de la partie I, les chapitres 2, 3 et 4 de la partie II, les chapitres 2, 3 et 4 de la partie III ainsi que l'annexe A.

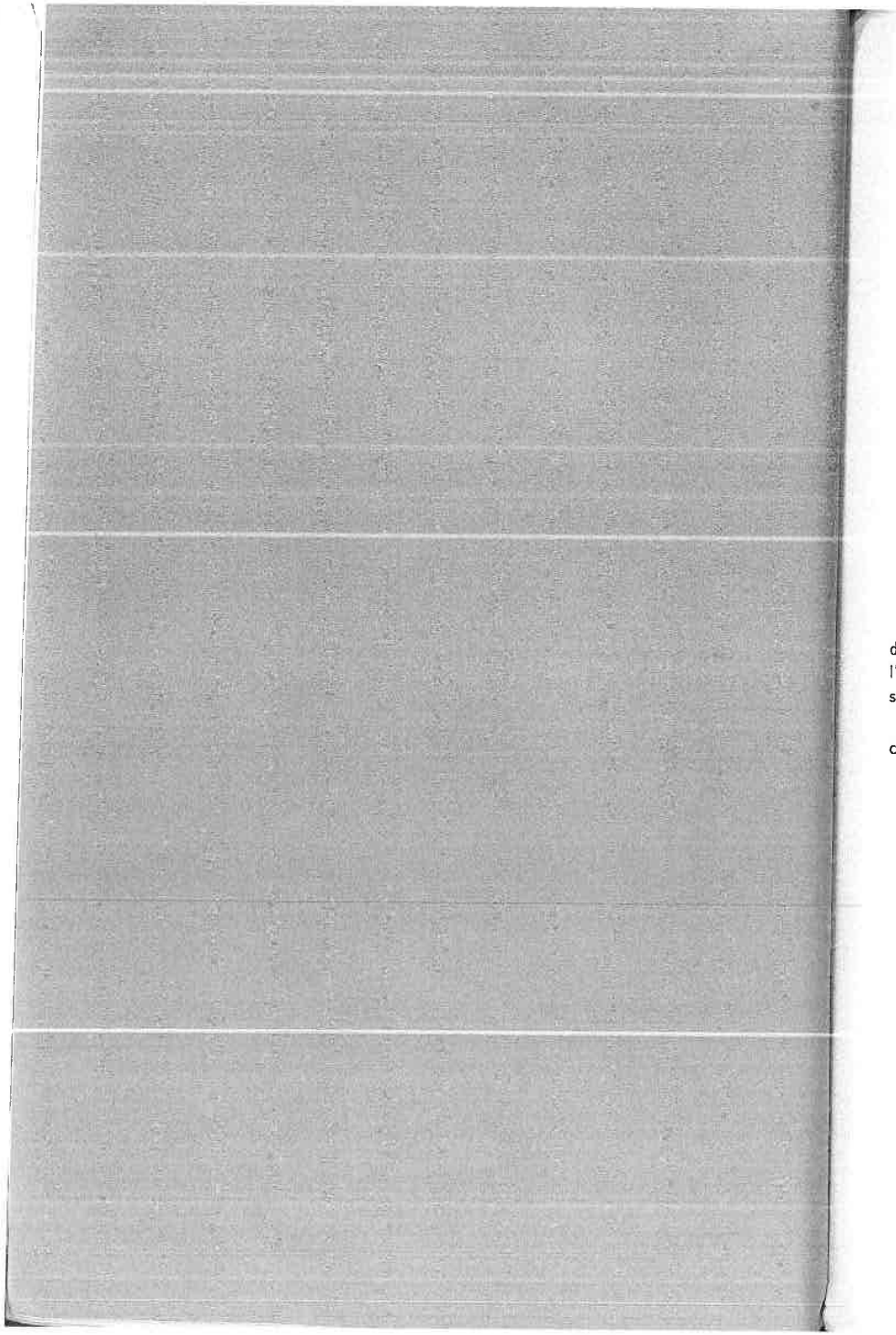
Le lecteur intéressé plus particulièrement par l'architecture d' ATOME, son type de contrôle et ses facilités de mise au point de systèmes d'IA à base de blackboard pourra consulter tous les chapitres de la partie III et l'annexe B. Pour un utilisateur potentiel d'ATOME, ces chapitres représentent un complément d'informations par rapport à son manuel d'utilisation.

Celui intéressé par nos remarques vis à vis du modèle du blackboard est invité à lire les chapitres 1 et 2 de la première partie, le chapitre 5 de la partie II, la conclusion générale de cette thèse et l'annexe A.

Nous conseillons la lecture totale du document pour tout chercheur dans le domaine des architectures de blackboard, ou tout concepteur de systèmes d'IA à base de blackboard désirant, soit approfondir ses connaissances dans le domaine, soit trouver des approches et des implantations du modèle du blackboard pouvant l'aider à choisir une architecture appropriée aux besoins de son application.

PARTIE I

Les systèmes multi-agents en IA



Le premier chapitre de cette partie a pour but de mettre en évidence le besoin de coopération de plusieurs agents dans certaines classes de problèmes relevant de l'intelligence artificielle. Ce chapitre décrit le concept des systèmes multi-agents et son apport en intelligence artificielle.

Le second chapitre de cette partie introduit et caractérise le cas particulier d'une coopération entre plusieurs agents basée sur le modèle du blackboard.

1
Les univers multi-agents en IA

L'IA a atteint un stade où elle aborde des domaines d'application très complexes, pour lesquels l'organisation d'un système d'IA sous forme d'*une* base de connaissances, d'*une* mémoire de travail et d'*un* mécanisme d'inférences aussi puissant soit-il devient insuffisante [21,22,133,172,173]. Parmi ces domaines d'application, on peut citer les problèmes faisant intervenir des techniques de reconnaissance et de compréhension des formes (e.g. interprétation de signaux complexes, ...) [113,114,115,116,198], le contrôle de processus industriels ou l'aide à la décision.

De telles applications font souvent intervenir une expertise hétérogène provenant de plusieurs sources d'informations, qui sont parfois peu fiables et amènent par conséquent à des hypothèses incohérentes ou incertaines. Il est nécessaire pour ces types d'applications de prendre en compte la pluralité des sources de connaissances ou d'information, et les différentes manières dont ces sources de connaissances sont structurées et exploitées, ainsi que les modèles de raisonnement pouvant d'une part maintenir la cohérence de l'ensemble des connaissances du système et d'autre part prendre en compte l'aspect incertain des hypothèses engendrées par le système durant la résolution d'un problème.

Dans ce chapitre, nous ne nous intéressons qu'à l'aspect *multi-expertise* des systèmes d'IA, en laissant de côté les aspects *maintien de la cohérence* et *gestion de l'incertain*.

1.1 Principe des systèmes multi-agents

Le principe des univers multi-agents en IA [91] consiste à partager, classer et distribuer à plusieurs *agents* "intelligents" l'ensemble des connaissances que possède un système d'IA. Chacun de ces agents devient alors spécialisé dans un sous-domaine du système. Il est à même, de par l'expertise qu'il renferme, de résoudre une partie du problème initial si les données dont il dispose le

lui permettent. Chaque agent est amené à coopérer et à collaborer avec les autres agents du système, d'une part, afin d'améliorer sa propre participation à la résolution du problème global et de compléter les informations dont pourraient avoir besoin les autres agents, d'autre part, afin de n'agir qu'au moment opportun. Ce transfert d'informations (transfert direct ou indirect) entre les différents agents est donc incontournable et nécessaire à un bon fonctionnement du système.

Les systèmes multi-agents peuvent être classés selon de nombreux critères dont les principaux sont cités ci-après. La *taille et la complexité des agents* conduisent à une participation plus ou moins complexe et complète au processus de résolution du problème. Les difficultés qui résident dans la coopération entre les agents en interaction dépendent également *du nombre de ces agents*, qui constituent alors un groupe, une communauté ou une société. De plus, il existe différents *mécanismes de communication* entre les agents pour lesquels il est nécessaire de définir des méthodologies de contrôle adaptées.

Dans cette partie, nous nous intéressons plus particulièrement à ce dernier critère. Nous décrivons les types de communication possibles entre agents ainsi que les mécanismes de contrôle nécessaires pour gérer cette communication.

1.1.1 La communication dans un univers multi-agents

On peut distinguer deux types de communication largement utilisés pour réaliser l'interaction entre les agents d'un système multi-agents intelligents :

1. *Communication par partage d'informations.*

   La communication entre les différents agents du système est réalisée par partage d'informations lorsque la solution courante du problème est centralisée dans une structure de données globale et commune à tous les agents.

   Cette structure renferme les données initiales ainsi que les différents résultats partiels fournis par les agents à chaque instant du processus de résolution du problème. Elle constitue ainsi une zone de communication entre les agents et représente leur seul moyen d'échange d'informations: ils n'ont alors pas besoin de se connaître mutuellement.

   Ce type de communication est souvent désigné dans la littérature d'IA par le modèle du *blackboard* ou tableau noir [59,76,122,161,202,200,201] (cf. figure 1.1), qui fait l'objet de notre thèse.

2. *Communication par envoi d'informations.*

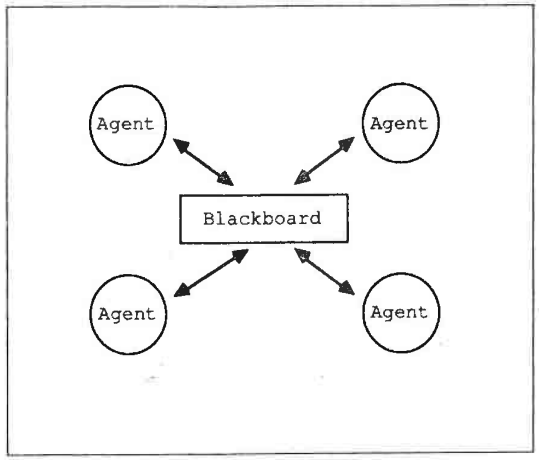


Figure 1.1. Structure d'un système à base de blackboard.

La communication entre les différents agents du système, lorsqu'elle est réalisée par envoi d'informations (messages), nécessite un protocole de communication utilisé par les agents pour échanger leurs résultats partiels. Chaque agent possède sa base de faits locale où il renferme ses propres résultats qui représentent une partie de la solution courante.

Ce type de communication est ce que l'on désigne dans la littérature d'IA, par le modèle des *acteurs* [136,137,159,182], ou *data flow* [97] ou *sociétés d'experts* [109] (cf. figure 1.2).

1.1.2 Le contrôle dans un univers multi-agents

La partie la plus complexe à mettre en œuvre dans un système multi agents est l'organisation et le contrôle du bon fonctionnement des agents. Le contrôle doit respecter les aspects de *coopération*, *conflit* et *concurrence* résultant du partage des données et du problème à résoudre en commun.

Il peut être réalisé entre autres par :

- Un système d'IA monolithique qui gère l'ensemble de tous les agents.

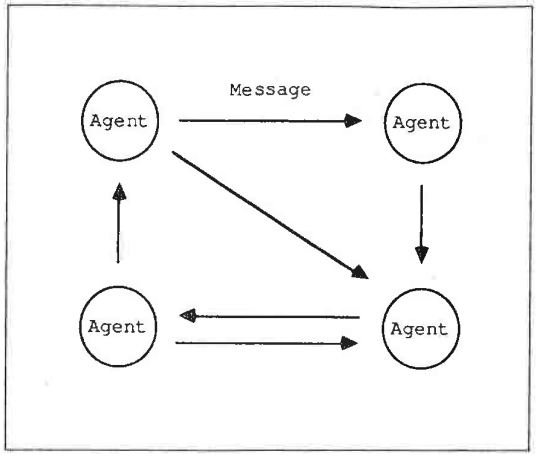


Figure 1.2. Structure d'une société d'experts.

Dans ce cas ce système requiert des connaissances sur tous les travaux potentiels de chaque agent, et doit être capable d'évaluer ces travaux afin de ne sélectionner que les meilleurs [86].

- Plusieurs systèmes d'IA, pouvant eux-mêmes être des agents. Le rôle de chacun d'eux peut être par exemple de gérer un sous-ensemble d'agents (spécialistes du domaine) ou d'orienter la résolution vers une sous-partie du problème de départ [118,166,208,215,224,236].
- Un programme "classique" de type système de gestion de ressources [52]. Les agents ont chacun une priorité non dépendante de l'état du processus de résolution du problème. De plus, la résolution des conflits entre ces agents n'est pas basée sur la connaissance.
- Négociations. Le contrôle dans un système multi-agents peut éventuellement être distribué entre les agents qui négocient alors leur activation [57,102,103,136,142,211,219,40].

Les différentes méthodes de contrôle définies précédemment ont permis de mettre en œuvre divers systèmes et modèles d'IA multi-agents [3,2.1]. Ces

derniers ont montré l'apport du principe multi-agents dans le domaine de l'intelligence artificielle. Ce principe constitue un axe de recherche important en IA comme le prouvent les conférences spécialisées [56,55,88,1,3,2].

1.2 Apport du concept multi-agents en IA

L'approche multi-experts pour aborder un problème relevant de l'IA est justifiée par les points suivants :

1. L'approche multi-agents s'adapte particulièrement bien à la réalité dans la mesure où de nombreux problèmes nécessitent diverses expertises pour être résolus. Des cas typiques de problèmes qui entrent dans cette catégorie sont les problèmes d'interprétation et de compréhension des formes (parole, vision, langage naturel...). Une telle approche peut conduire à un renforcement ou à un affaiblissement mutuel des propositions faites par les différents agents.

2. La coopération entre plusieurs systèmes experts dont les expertises se chevauchent mais dont les points de vue sont différents permet une résolution de problèmes complexes plus efficace et plus complète qu'avec un seul système expert.

3. L'approche multi-experts permet de résoudre des problèmes de taille et de complexité telles qu'il n'est pas réaliste d'essayer de les résoudre à l'aide d'un seul système expert.

4. L'intégration de plusieurs expertises incomplètes ou peu fiables peut mener à une expertise plus sûre et plus robuste.

5. L'évolution des moyens matériels tels que les machines parallèles semble être un atout favorable à ce type d'approche.

D'autre part, une approche multi-agents pour aborder un problème complexe permet de respecter au maximum les normes prescrites par le génie logiciel [228] dans l'élaboration d'un système, à savoir :

1. *Modularité*

   La complexité d'un système expert croît aussi rapidement que la taille de sa base de connaissances. Partitionner ce système en N sous-systèmes réduit sa complexité par un facteur plus grand que N et la configuration résultante se trouve plus facile à développer, tester et maintenir.

2. *Efficacité*

   Les sous-systèmes, lorsque les machines supports et le type d'application le permettent, peuvent fonctionner en parallèle. La rapidité de la résolution du problème peut ainsi être améliorée.

3. *Fiabilité*

   Le fait qu'un sous-système devient non opérationnel ou peu fiable n'entraîne pas automatiquement l'effondrement total du système.

4. *Réutilisabilité*

   Un sous-système peut être réutilisé pour implanter une partie d'un autre système. L'expertise qu'il englobe ne demande pas à être de nouveau acquise et réimplantée.

L'apport multi-agents dans les systèmes d'IA n'est donc pas négligeable et doit être considéré comme une nécessité dans les années à venir. Ce domaine de recherche est très vaste et fait l'objet de nombreux travaux à travers le monde [91]. Dans la suite de ce document, nous décrirons plus particulièrement le modèle du blackboard, modèle de coopération entre agents dans le cas où ils communiquent par partage d'informations.

Une première approche de ce modèle est proposée dans le chapitre suivant, une description plus approfondie des différentes architectures issues de ce modèle et des travaux que nous avons effectués dans ce cadre faisant l'objet des parties suivantes.

2
Le modèle du blackboard

Le but de ce chapitre est de présenter l'architecture de blackboard comme modèle de représentation, organisation et traitement de la connaissance dans un univers multi-agents. Nous définirons les différents composants de base de ce modèle. Nous caractériserons les problèmes relevant des techniques d'intelligence artificielle (IA) pour lesquels ce modèle est bien adapté.

2.1 Présentation du modèle

Le modèle du blackboard traite la résolution d'un problème comme un processus *incrémental* et *opportuniste* d'assemblage des éléments de la solution finale. Il permet de voir un problème à plusieurs niveaux de détails. Par exemple, si le problème est d'identifier des bâtiments navigant sur ou sous la surface de l'eau en s'appuyant essentiellement sur la lecture de diagrammes d'évolution temporelle des spectres des signaux, une architecture de blackboard permet la représentation et l'utilisation de *sources d'informations* permettant soit d'identifier des bâtiments à partir de sources de bruit (hélice, moteur-diesel...), soit d'identifier la signature d'une source de bruit à partir du signal reçu par le sonar (cf. figure 2.1).
Un tel système commence par extraire les caractéristiques du signal pour ensuite regrouper les signaux en harmoniques. A partir de ces harmoniques, il essaye de reconnaître des sources de bruit pour ensuite identifier le bâtiment émetteur. Cette étape de résolution du problème correspond à la phase ascendante du système; la phase descendante du système consiste en des vérifications, confirmations ou prédictions d'informations.

Ainsi l'interprétation de ces signaux apparaît comme un processus de transformation progressive de données venant de capteurs en une description symbolique en passant par plusieurs *niveaux d'abstraction* (signal, harmonique, source de bruit et bâtiment émetteur).

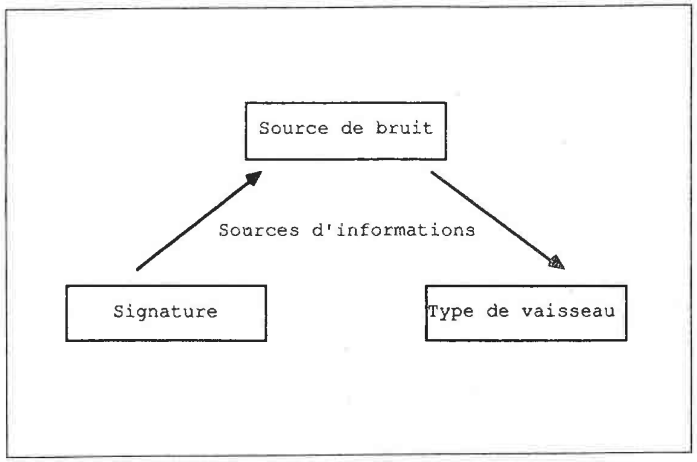


Figure 2.1. Exemple de deux sources d'informations coopératives.

2.1.1 Historique

Le modèle du blackboard a été implanté pour la première fois dans HEARSAY-II [179,85,82,197], système de reconnaissance et de compréhension de la parole continue. Il a été ensuite utilisé dans des domaines d'application très variés tels que l'interprétation des signaux [208,194,247], la modélisation de la structure 3D de molécules [237,236,124,123,26], la compréhension d'images [111,65,108, 10], la médecine [33] et la planification [126,127,215,216].

Dès les premières réalisations des systèmes à base de blackboard dans des applications comme celles citées ci-dessus, plusieurs chercheurs en IA ont senti la généricité du modèle et ont mis au point des systèmes génériques tels que AGE [6,5,7,8.199.205]. HEARSAY-III [13.87] et BB-1 [128].

Victor Lesser, l'un des concepteurs de HEARSAY-II et ses collègues à l'université de Massachusetts ont adopté le modèle du blackboard comme principe de base pour la résolution de problèmes d'IA dans un univers distribué [175]. Ils ont ainsi développé un réseau de systèmes à base de blackboard identiques qui partagent leurs résultats intermédiaires afin d'arriver à une solution globale consistante. Ce réseau de systèmes appelé DVMT (Distributed Vehicle Monitoring Testbed) sert à tester et évaluer différentes stratégies de coordination entre plusieurs systèmes d'IA coopératifs [176,42,67,138,213,214,31].

Depuis 1980, l'architecture du blackboard est reconnue par une large communauté de chercheurs en IA comme l'un des modèles les plus prometteurs pour les prochaines générations de systèmes experts ou à base de connaissances. En Europe, ce sont des chercheurs en Grande Bretagne qui ont été les premiers à manifester leur intérêt pour les architectures de blackboard. Une société appelée SPL a développé un système de compréhension du signal (SUS) pour les besoins d'*Admiralty Research Establishment* [169]. Ce système à architecture de blackboard a donné ensuite naissance à un système générique appelé MXA [234]. Les concepteurs de SUS et MXA se sont largement inspirés des systèmes Américains HASP [208] et AGE. D'autres systèmes ont vu également le jour en Grande Bretagne : BLOBS [249] qui intègre quelques aspects du raisonnement temporel, MUSE [218] qui prend en compte des contraintes de temps réel et un troisième système générique développé en PROLOG à l'université d'Edinburgh [148]. Ces trois systèmes ont de nombreuses similitudes avec AGE.

Aux Etats Unis, Victor Lesser, Frederic Hayes-Roth et Lee Erman, trois des concepteurs de HEARSAY-II ont quitté l'université de Carnegie Mellon pour d'autres organismes de recherche.

1. Victor Lesser avec Daniel Corkill a mis à profit l'expérience acquise sur les systèmes HEARSAY-II et DVMT pour développer un nouveau système générique appelé GBB [46].

2. Frederic Hayes-Roth développa avec Barbara Hayes-Roth, un système de planification appelé OPM [126]. En 1982, Barbara Hayes-Roth a rejoint *Heuristic Programming Project* (HPP)[1] à l'université de Stanford et a commencé le développement du système générique BB-1. Le mécanisme de contrôle dans BB-1 est une généralisation d'idées développées initialement pour OPM. Le système BB-1 ou son architecture a été utilisé dans plusieurs domaines d'application tels que la détermination de la structure des protéines, la planification et le génie civil [129]. Plus tard, un environnement a été développé autour de BB-1 et porte le nom de BB* [130].

3. Lee Erman a développé à *Information Science Institute* (ISI) de l'université de Californie du sud, une généralisation du système HEARSAY-II qui s'appelle HEARSAY-III.

A l'université de Stanford, actuellement Penny Nii et ses co-équipiers travaillent sur des implantations parallèles du modèle du blackboard : les systèmes CAGE

1. Plus tard HPP a été intégré dans *Knowledge Systems Laboratory* (KSL) de l'université de Stanford [4].

et POLIGON [220,203,206], tandis que Barbara Hayes-Roth étend BB-1 vers une architecture prenant en compte des événements externes : le système MI-1 [119, 120] dont le rôle est de surveiller en permanence les pulsations cardiaques d'un malade placé dans un service de soins intensifs.

Depuis 1986, le modèle du blackboard connaît un succès auprès des industriels d'IA. Ainsi, le centre des techniques avancées de Boeing a développé le système générique appelé BBB [18] qui s'inspire énormément du système BB-1. Pour des raisons d'efficacité et d'utilisation spécifique aux problèmes traités par ce centre [233,151], BBB a donné suite à un autre système générique appelé ERASMUS [16].

Les sociétés d'IA américaines Teknowledge et IntelliCorp, quant à elles, ont développé des environnements de l'ingénierie de la connaissance[2] appelé respectivement ABE [83,96,131,170] et OPUS [93] où l'architecture du blackboard est l'un des moyens de communication interprocessus entre les différents systèmes de l'environnement.

En plus de DVMT, d'autres systèmes faisant appel aux techniques d'intelligence artificielle distribuée (IAD) [140,1] ont adopté cette architecture comme modèle de base de coopération et de coordination entre plusieurs systèmes d'IA. Parmi eux, nous pouvons citer MINDS [141] pour le domaine de recherche d'archives, AF [110] et AGORA [20] pour le domaine de la parole.

En dehors de la Grande Bretagne, *à notre connaissance*, l'INRIA/CRIN-Lorraine à travers le projet SYCO est le premier laboratoire de recherche en Europe qui s'est intéressé au modèle du blackboard pour la résolution de problèmes d'IA. Nous avons ainsi développé le système générique appelé ATOME (cf. partie II de cette thèse).

Ne disposant pas préalablement d'une application réelle pour développer ATOME, nous nous sommes inspirés des travaux antérieurs effectués sur les architectures du blackboard et plus précisement les réalisations concernant les systèmes CRYSALIS, HASP/SIAP, AGE et BB-1.

En plus, des systèmes génériques cités ci-dessus, d'autres systèmes continuent à voir le jour dont les principaux sont BBC [181], ARIADNE-1 [51], BLONDIE [241,242,184,94] et autres [89,157,223].

De plus en plus, on constate l'apparition de systèmes d'IA dont l'architecture se situe entre les approches classiques et celle des systèmes à base de blackboard tels que CODGER de l'université de Carnegie-Mellon : système de vision utilisé en robotique [226,238] , GEST de l'institut des technologies de Géorgie : système générique pour développer des applications d'IA [222,239], SCHEMA [66] de l'université de Massachusetts : système d'interprétation d'images naturelles, et en France, les systèmes génériques SMECI [41] et KIRK [35].

2. Equivalent du terme anglo-saxon *Knowledge Engineering*.

De nouveaux systèmes à base de blackboard voient le jour en France depuis 1987, tels que TRAM [158] au CEA pour la robotique, SONIA [39] pour les problèmes d'ordonnancement à l'université de Paris VI, IKAROS [30] pour la reconnaissance de la parole, qui utilisent le modèle de BB-1 et GRAPHEIN [36] au CRIN pour la reconnaissance des caractères, qui utilise ATOME comme outil de mise au point.

Actuellement, les recherches sur les architectures de blackboard se focalisent de plus en plus sur les points suivants [3,2] :

- L'organisation du contrôle [25,39,99,164,178,118,69,70,195,235,50,49].
- Le parallélisme des sources de connaissances [43,144,206,227,81,80,183, 243].
- Le temps réel [180,217,218,134].

Nous nous sommes intéressés à ces problèmes et plus particulièrement à l'aspect *organisation du contrôle* dans une architecture de blackboard.

La figure 2.2 représente l'ensemble de ces systèmes, sachant que cette liste n'est pas exhaustive.

Les systèmes HEARSAY-II, DVMT, HASP/SIAP, CRYSALIS, BB-1 et GBB seront décrits plus en détail dans les chapitres 2, 3 et 4 de cette partie.

2.1.2 Le Modèle

Malgré la diversité des implantations dans des systèmes tels que ceux cités précédemment, tout système à base de blackboard comporte les trois caractéristiques de base suivantes :

1. *Tous les éléments de la solution d'un problème (appelés aussi hypothèses, entrées ou nœuds) engendrés durant la résolution du système sont stockés dans une base de données structurée appelée le blackboard.*

   Le blackboard est une structure de données qui représente l'état de la solution courante du problème posé. Cette structure de données est divisée en plusieurs niveaux d'abstraction et permet donc de considérer la solution sous plusieurs niveaux de détails. Les hypothèses émises dans le blackboard (les éléments du blackboard) peuvent être reliées entre elles de telle sorte que le blackboard peut être vu comme un réseau d'hypothèses. Chaque hypothèse est caractérisée par un certain nombre d'attributs liés au niveau du blackboard qu'elle occupe ainsi qu'au domaine d'application. Elle est souvent implantée sous forme d'objet [152] ou de structure [149].

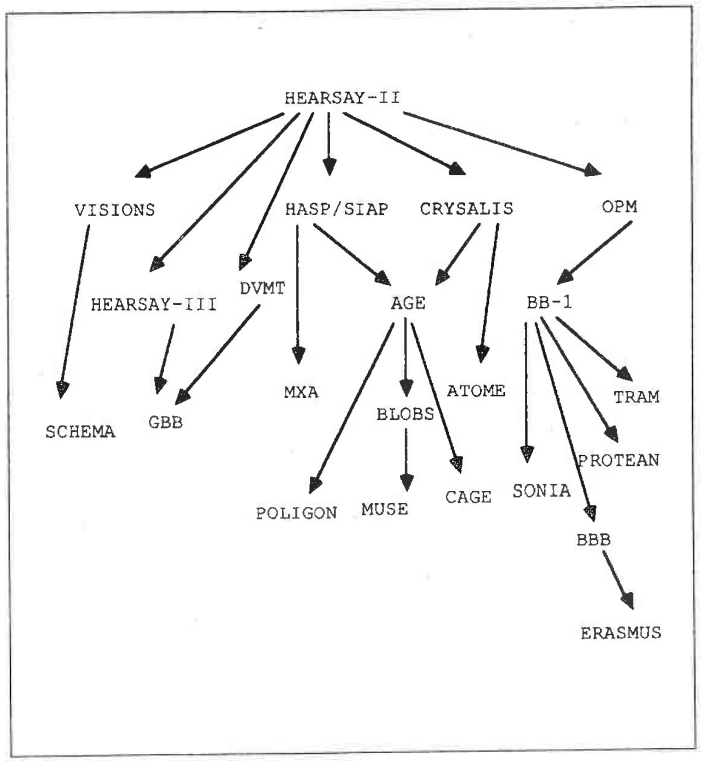


Figure 2.2. Les architectures de blackboard et leurs origines.

2. *Les agents qui engendrent et stockent leurs hypothèses dans le blackboard sont des modules indépendants appelés sources de connaissances (SCs) ou spécialistes.*

   Les sources de connaissances représentent des processus indépendants dont le rôle est de coopérer pour résoudre un problème posé. Chaque SC est experte dans un sous-domaine du domaine de départ. Ces SCs sont amenées à se communiquer des informations et à négocier leur participation à la résolution du problème. Une telle interaction ne peut se faire qu'à travers le blackboard, seul moyen de communication entre les SCs.

   Les SCs ont un format *condition-action*. La partie *condition* décrit les situations dans lesquelles la SC peut contribuer à l'avancement de la résolution du problème. Généralement, cette partie requiert une configuration particulière des hypothèses dans le blackboard. La partie *action* spécifie la contribution de la SC une fois que cette dernière est activée. Généralement, elle se résume à des créations de nouvelles hypothèses, des modifications ou des suppressions d'hypothèses déjà existantes dans le blackboard. Elle peut être exprimée sous forme procédurale (programme) ou déclarative (système à base de connaissances). Seules les SCs dont la partie condition est satisfaite peuvent être activées à un instant donné.

   Chaque changement dans le blackboard constitue un *événement* qui, en présence d'autres informations spécifiques dans le blackboard, peut satisfaire la partie condition d'une SC et ainsi la rendre activable. Les SCs sont *indépendantes* puisqu'elles s'ignorent mutuellement et ne réagissent qu'aux changements du blackboard. Elles sont *coopératives* étant donné que leur rôle est de construire une solution satisfaisante à un problème commun à toutes ces SCs. Elles sont *concurrentes* puisqu'elles ont en commun le blackboard en lecture et écriture.

3. *Un mécanisme de contrôle, assurant le fonctionnement du système en fonction d'une certaine stratégie.*

   En général, les informations présentes dans le blackboard intéressent plusieurs SCs qui déclarent simultanément vouloir contribuer à la résolution du problème et deviennent alors compétitives. Le rôle du contrôle est entre autres de résoudre le conflit entre ces SCs en fonction de l'état de la solution courante et des travaux potentiels de chaque SC. C'est le point fondamental d'une architecture de blackboard. C'est aussi la partie la plus difficile à mettre en œuvre du fait de la complexité des stratégies nécessaires à la bonne résolution du système. La diversité des systèmes basés sur le modèle du blackboard a fait apparaître quatre types de contrôle :

procédural fondé sur la notion d'agendas, *hiérarchique* fondé sur la notion de méta-sources de connaissances, *opportuniste* à base de blackboard et d'agendas et *hybride* fondé sur la notion de phases. Nous reviendrons en détail sur ces types de contrôle dans les chapitres 2, 3, 4 de la partie II et dans la partie III.

La figure 2.3 illustre ces trois composantes de base.

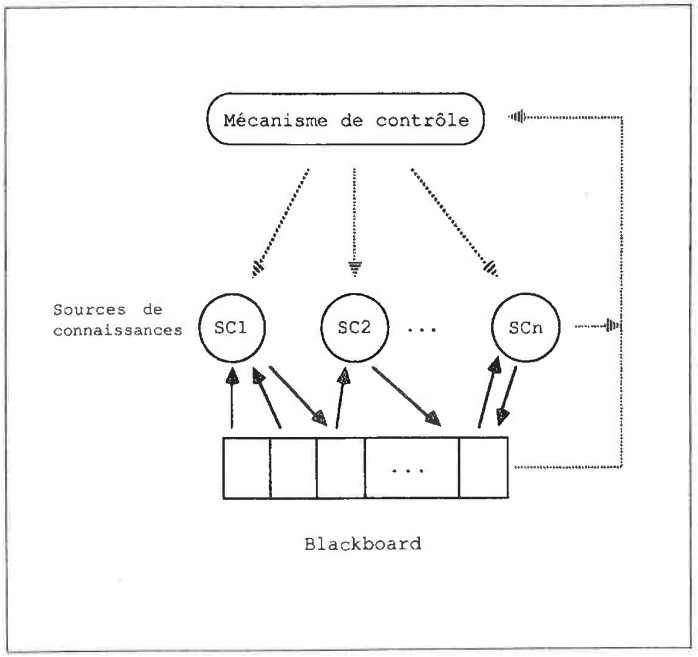


Figure 2.3. Les trois composantes de base du modèle du blackboard.

2.2 Notion d'événement

Dans le modèle du blackboard, la notion d'*événement* est inhérente. Un événement peut être vu comme un message ou plutôt comme un signal émis par les sources de connaissances pour prévenir le mécanisme de contrôle des actions qu'elles ont effectuées dans le blackboard. Un événement constitue alors une structure de contrôle qui renferme des informations générales sur les différents types d'actions (création, modification ou suppression d'une hypothèse) qui ont eu lieu dans le blackboard. Il mémorise ce qui s'est passé dans le blackboard mais ne contient pas d'informations détaillées sur le contenu de ce dernier.

Les événements sont utilisés par le mécanisme de contrôle pour détecter et gérer l'activation des sources de connaissances intéressées par les dernières mises à jour de la solution courante (cf. figure 2.4). Ils permettent ainsi au contrôleur d'éviter un parcours combinatoire du blackboard et d'analyser uniquement ce qui s'est passé dans celui-ci.

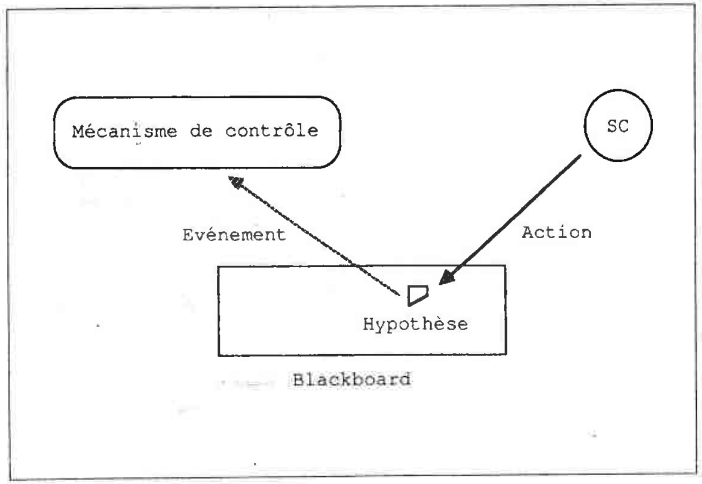


Figure 2.4. Emission d'un événement.

Nous montrerons comment cette notion d'événements est exploitée dans les systèmes basés sur le modèle du blackboard décrits dans les parties II et III.

2.3 Problèmes de type blackboard

Dans le cas général, l'utilisation de l'architecture de blackboard pour développer un système d'IA est intéressante lorsqu'il s'agit de faire coexister dans la même structure des sous-systèmes dont le fonctionnement peut être très divers [75]. Le modèle du blackboard est notamment adapté à des problèmes nécessitant la coopération des techniques de reconnaissance des formes et de l'intelligence artificielle, dans la mesure où ces problèmes combinent les traitements numériques à un raisonnement fondé sur des connaissances [246,231].

Il est conseillé d'utiliser une architecture de blackboard pour développer une application d'IA quand [207] :

1. *On a besoin d'analyser une très grande quantité d'informations.*

   Par exemple en interprétation et compréhension de signaux complexes [194,196], en vision par ordinateur [10,108,154,112,245], pour le contrôle de processus [12,61,53], en planification [215,216], en aide à la décision [230] ou en fusion de données [240]. Dans tous ces cas, les systèmes manipulent une très grande quantité de données souvent hétérogènes, incertaines, incomplètes et même parfois inconsistantes. Le modèle du blackboard permet de représenter, d'organiser et de gérer efficacement l'exploitation de ces données.

2. *L'analyse du domaine d'application peut être décomposée en différents niveaux d'abstraction.*

   Dans des applications telles que celles citées ci-dessus, le domaine peut être décomposé de façon naturelle en plusieurs niveaux de détails, de telle sorte que l'interprétation se réduit à une succession de transformations d'informations d'un niveau à un autre.

   A chaque étape de la résolution du problème, des SCs enregistrent des résultats partiels, c'est-à-dire intermédiaires, à un certain niveau du blackboard. Ces résultats seront évalués, corrigés et harmonisés par d'autres SCs qui en déduiront alors une nouvelle solution partielle à un niveau du blackboard (éventuellement le même). Ce processus est répété jusqu'à arriver à une solution globale satisfaisante du problème.

3. *Le problème nécessite la collaboration de plusieurs sources d'informations. Certaines de ces sources d'informations relèvent de l'expertise humaine, un expert pouvant être à l'origine de plusieurs d'entre elles.*

   En compréhension de la parole, par exemple, on peut faire coopérer des SCs acoustiques, phonétiques, phonologiques, prosodiques, lexicales, syntaxiques, sémantiques et pragmatiques.
   Beaucoup d'applications d'IA utilisent des techniques algorithmiques ou heuristiques qui, si elles sont manipulées seules, ne donnent pas de résultats satisfaisants et s'avèrent parfois inefficaces. Cependant, une fois ces techniques regroupées, la qualité des résultats est sensiblement améliorée. En effet chaque SC peut modifier et ainsi corriger les informations erronées émises par une autre SC.

4. *Des stratégies opportunistes et complexes doivent être utilisées.*

   L'ordre et l'instant d'intervention des SCs sont déterminés de façon opportuniste en fonction de l'avancement de la résolution du problème.

   Une architecture de blackboard offre un mécanisme de contrôle très robuste et sophistiqué pour guider la conduite de systèmes d'IA comportant une grande incertitude sur le flux de contrôle.

Par la suite, nous désignerons par "problème de type blackboard" tout problème relevant des points cités précédemment.

2.4 Conclusion

Dans un système d'IA dont l'architecture est fondée sur le modèle du blackboard, le codage des connaissances sous forme de SCs indépendantes et modulaires, et la séparation entre les concepts liés au domaine et ceux liés au contrôle, conduit à un système évolutif, facilement maintenable, lisible, concis, structuré et fiable. Les SCs peuvent ainsi être testées et mises à jour séparément avant d'être intégrées au système. Néanmoins, le développement et la mise au point de systèmes d'IA fondés sur le modèle de blackboard comme HEARSAY-II, CRYSALIS, HASP/SIAP, INTERSENSOR, DVMT, PROTEAN, ABC,...est encore un art difficile à comprendre et à maîtriser, et demande un effort considérable de la part des concepteurs, étant donné la complexité du contrôle dans une telle architecture. C'est pourquoi l'élaboration d'outils d'IA tels que HEARSAY-III, AGE, BB-1, et GBB occupe une bonne part de la recherche en IA. Ces outils, bien qu'au stade de la recherche, facilitent la compréhension de la résolution des problèmes d'IA très complexes ainsi que le processus de coopération entre plusieurs agents chargés de résoudre ensemble ces problèmes. Ces outils offrent en plus un gain de temps considérable dans la mise en œuvre d'applications d'IA.

Bien que destinés à fournir des outils généraux et indépendants du domaine d'application, ces générateurs de systèmes d'IA se sont avérés orientés vers des types d'applications assez restreints, étant donné que chacun d'eux impose un contrôle uniforme pour toute la résolution du problème.

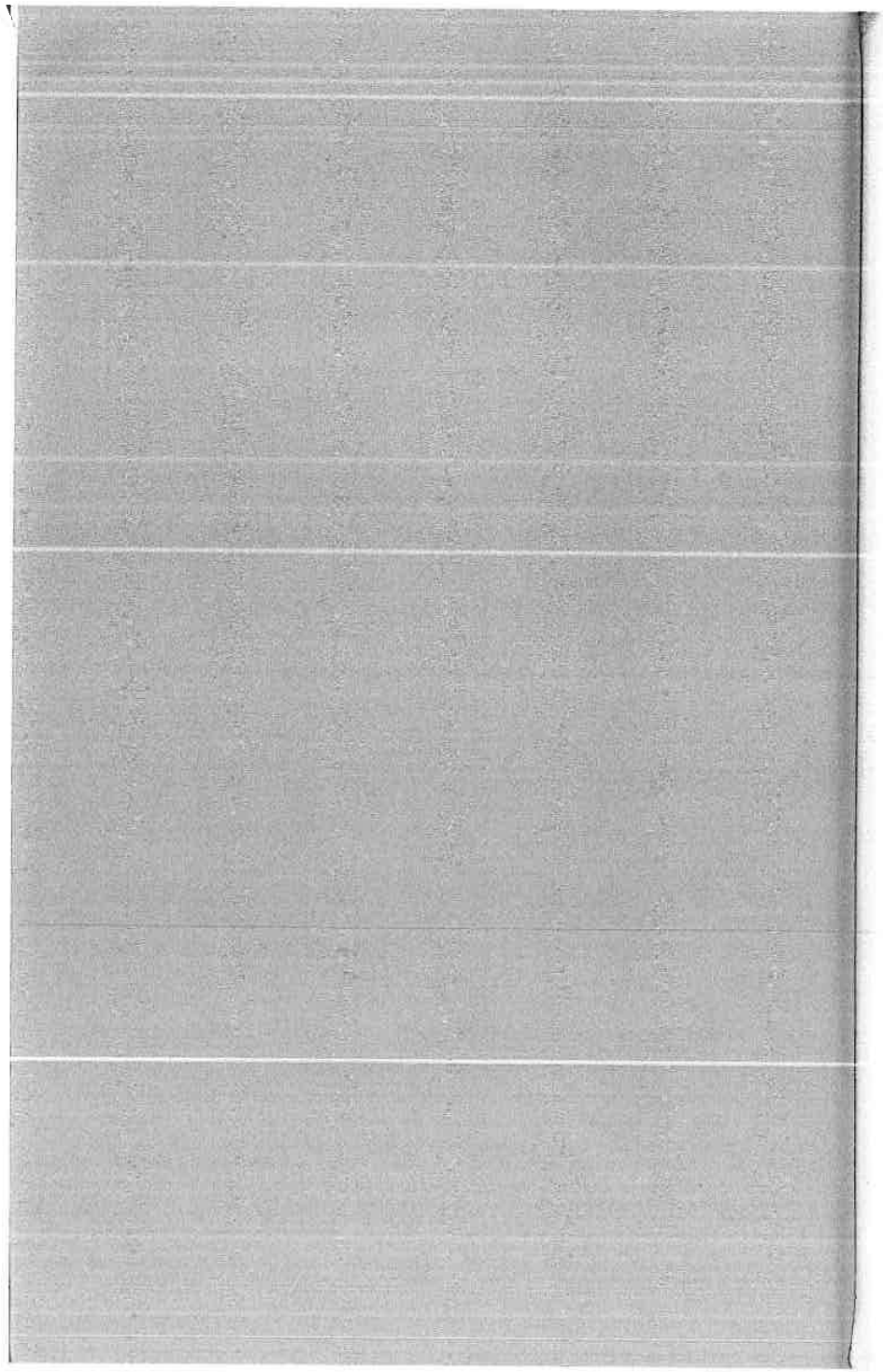
Le modèle *hybride à multi-phases* que nous avons défini et implanté dans l'outil ATOME corrige ces limites en proposant un contrôle approprié à chaque phase de la résolution d'un problème. Cet outil se veut être le plus générique possible et vise un plus grand domaine d'applications tout en sauvegardant la puissance de ces dernières.

Le but de la partie suivante n'est pas de décrire le plus grand nombre possible de systèmes basés sur le modèle du blackboard mais de présenter les différentes architectures, les différents types de contrôle (composante la plus importante du modèle du blackboard) qui sont nés des diverses implantations de ce modèle.

Nous définirons chacun de ces types de contrôle en illustrant nos propos par une description du ou des systèmes à son origine et de quelques systèmes ayant corrigé certaines des faiblesses de ces derniers.

PARTIE II

Architectures de blackboard



Cette partie définit les principaux types de contrôle utilisés jusqu'à présent dans les systèmes à base de blackboard. Le premier chapitre de cette partie définit le "contrôle procédural" et son implantation dans les systèmes HEARSAY-II et DVMT. Le second chapitre introduit le "contrôle hiérarchique" et sa mise en œuvre dans les systèmes HASP/SIAP et CRYSALIS. Le "contrôle à base de blackboard" ainsi que son utilisation dans les générateurs BB-1 et GBB sont décrits dans le troisième chapitre.

Nous terminons cette partie par une conclusion permettant de situer nos travaux dans ce domaine.

1
Le contrôle dans une architecture de blackboard

Afin de résoudre un problème particulier, un système d'IA exécute une série d'actions. Chaque action est rendue possible grâce à la configuration des données initiales ou à celle des éléments de la solution engendrés auparavant. En appliquant certaines connaissances liées au domaine, il engendre de nouveaux éléments de la solution ou en modifie des anciens. A chaque point du processus de résolution du problème, plusieurs actions peuvent être possibles et ainsi entrer en conflit. Le problème du contrôle qui est primordial pour la conduite des activités d'un système d'IA [14,54,174,105] peut donc être formulé comme suit : *à chaque instant du processus de résolution du problème, quelle action le système doit il exécuter ?*

Pendant le traitement d'un problème donné, le système choisit de manière implicite ou explicite quelle partie de la connaissance appliquer, quelle région des données traiter, et quelles méthodes et stratégies utiliser pour résoudre ce problème. Pour ce faire, il doit être capable de juger la qualité de la solution courante, d'évaluer diverses solutions alternatives, de savoir reconnaître quand un problème est résolu, et d'interrompre le processus de résolution d'un sous-problème pour en considérer un autre.

Malgré le succès des méthodes qu'utilisent les systèmes d'IA pour résoudre un problème particulier, la plupart de ces méthodes restent simplistes et figées. Face à des problèmes complexes, un mécanisme de contrôle dans un système d'IA, doit adopter plusieurs niveaux de raisonnement lors de la résolution du problème, séparer les stratégies locales des stratégies globales, adopter et combiner des heuristiques de plusieurs granularités, et planifier des séquences d'actions en vue d'atteindre des buts à moyen et long terme.

Les architectures de blackboard avec leur mécanisme de contrôle sophistiqué permettent d'intégrer de telles capacités. Nous présentons dans ce chapitre le rôle que doit jouer un mécanisme de contrôle dans de telles architectures. Nous décrivons également différentes approches pour représenter et exploiter

les connaissances liées à ce type de contrôle.

1.1 Rôle

Le rôle du mécanisme de contrôle dans un système à architecture de blackboard est de gérer le fonctionnement global du système : il coordonne la coopération entre les différentes SCs. Pour ce faire, il contrôle l'évolution de la *solution courante* qui est mémorisée dans le blackboard et oriente les activités du système (cf. figure 1.1).

En général, les informations présentes dans le blackboard (représentant ainsi la solution courante) intéressent plusieurs SCs qui déclarent simultanément, de façon implicite ou explicite, vouloir contribuer à la résolution du problème et deviennent alors compétitives.

Dans la suite, nous entendons par *niveaux d'entrée* et *niveaux de sortie* d'une SC les niveaux du blackboard où respectivement elle lit ses données et place ses résultats.

Plusieurs SCs peuvent entrer en conflit pour les raisons suivantes :

- Elles ont des niveaux d'entrée dans le blackboard en commun. Cas par exemple où plusieurs SCs essaient de travailler sur les mêmes données (cf. figure 1.2).
- L'état actuel du blackboard présente une situation telle que plusieurs régions de données différentes constituent des îlots de données compétitifs à traiter (cf. figure 1.3). Traiter en premier un îlot est susceptible de perturber le traitement du second et vice-versa. En effet, le traitement d'un îlot peut conduire à annuler celui de l'autre, à en réduire les effets ou au contraire à améliorer sa participation. La qualité de la solution finale (état final du blackboard) ainsi que le temps de résolution du problème dépendent également du nombre d'îlots à traiter et de l'ordre dans lequel ils sont traités.
- L'organisation du système et l'état actuel du blackboard présentent une situation qui combine les deux cas précédents (cf. figure 1.4).

Afin de résoudre le conflit entre plusieurs SCs compétitives, deux approches sont possibles :

1. *Utilisation d'agendas :*

   Les systèmes HEARSAY-II et BB-1 (cf. chapitres 2 et 4) associent à chaque SC des paramètres tels que sa *fiabilité*, son *efficacité*, son *coût*... Ces

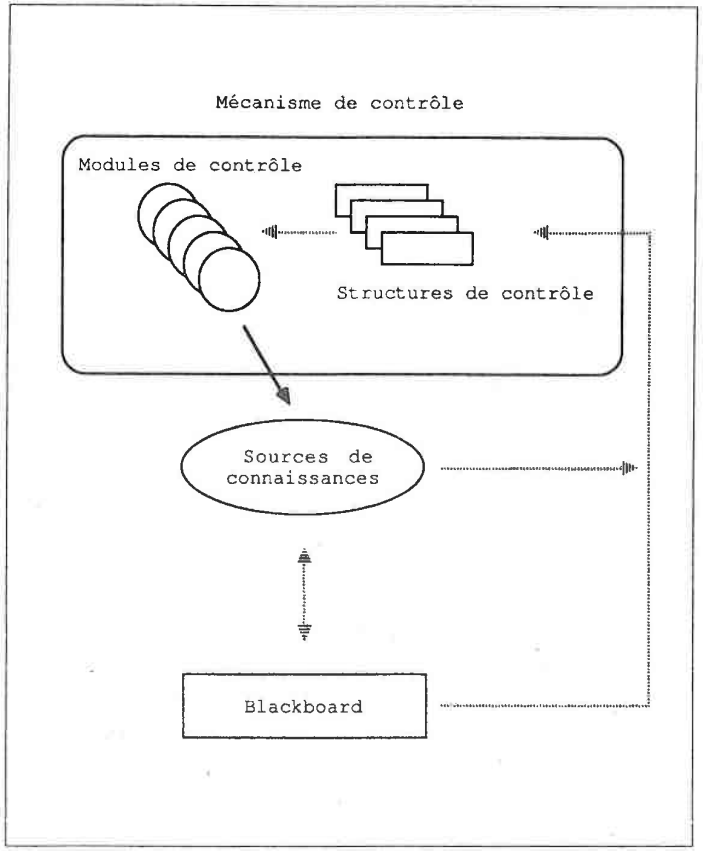


Figure 1.1. Le mécanisme de contrôle dans une architecture de blackboard.

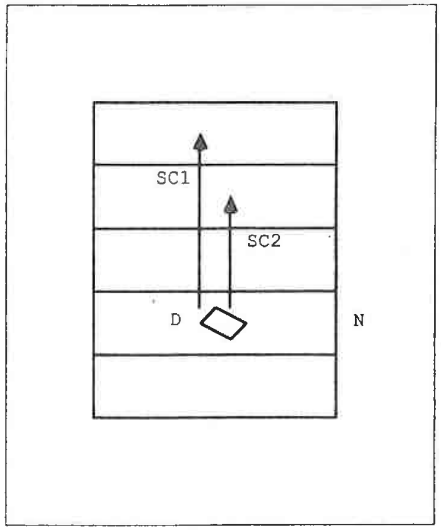


Figure 1.2. Les SCs SC1 et SC2 sont en conflit car elles sont intéressées par la même donnée D située dans leur niveau d'entrée commun N.

paramètres sont généralement dépendants de la configuration du blackboard et sont par conséquent susceptibles d'évoluer en fonction de l'état d'avancement de la résolution du problème. A chaque cycle de la résolution du problème, le mécanisme de contrôle détecte les SCs activables par les derniers changements du blackboard. Il crée alors des *instanciations de sources de connaissances* (ISCs) qui associent à chaque SC les changements du blackboard qui l'ont rendue activable et constituent ainsi les travaux potentiels de cette SC. Le mécanisme de contrôle place ces ISCs dans une liste d'attente appelée *agenda* et leur affecte ensuite une priorité en appliquant un ensemble d'heuristiques pour en sélectionner la meilleure.

2. *Utilisation de méta-sources de connaissances (méta-SC) :*

   Les systèmes HASP/SIAP et CRYSALIS (cf. chapitre 3) utilisent la notion

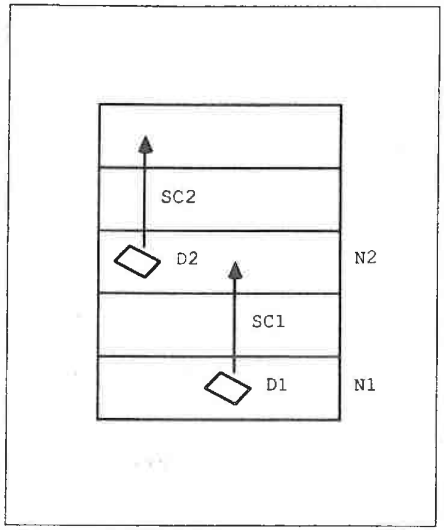


Figure 1.3. Les SCs SC1 et SC2 sont en conflit car elles ont deux données compétitives D1 et D2 dans leurs niveaux d'entrée respectifs N1 et N2.

de méta-source de connaissances pour gérer le processus de résolution du problème. En effet, une méta-SC se charge de coordonner les activités d'un sous-ensemble des SCs du système, et fournit ainsi un contrôle local à ce dernier. Pour gérer l'ensemble de toutes les méta-SCs, les deux systèmes utilisent une seule méta-méta-SC qui leur fournit alors un contrôle global.

1.2 Focalisation de contrôle

Nous rappelons qu'à chaque instant du processus de résolution d'un problème, un système d'IA dispose généralement de plusieurs actions compétitives. Dans une architecture de blackboard, le résultat du choix d'une action parmi toutes les actions possibles est appelé *focalisation de contrôle* ou *focalisation d'attention.*

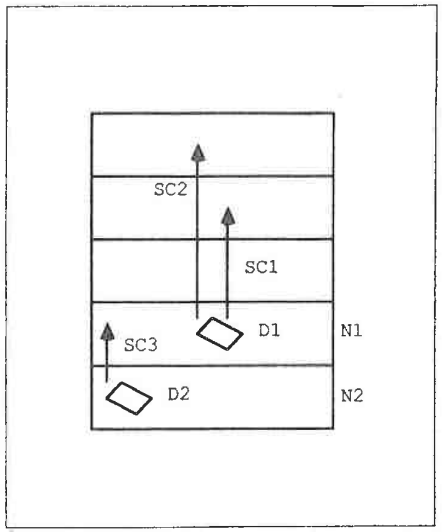


Figure 1.4. Les SCs SC1 SC2 et SC3 sont toutes les trois en conflit à cause des données compétitives D1 et D2 situées dans leur niveau d'entrée respectifs N1 et N2.

Si la prochaine action que doit effectuer le mécanisme de contrôle consiste à sélectionner la prochaine région (ou plus généralement les prochaines régions) du blackboard à traiter, on dira que la focalisation de contrôle dans ce cas est une région (ou plus généralement des régions) du blackboard. En revanche, si cette action consiste à sélectionner la prochaine SC (ou plus généralement les prochaines SCs) à exécuter, on dira que la *focalisation de contrôle* dans ce cas est une SC (ou plus généralement des SCs). La focalisation de contrôle peut être également la prochaine région du blackboard à traiter et la prochaine SC à exécuter avec cette région comme contexte d'activation (ou plus généralement les prochaines régions du blackboard à traiter et les prochaines SCs à exécuter avec ces régions comme contexte de travail).

Dans HEARSAY-II et BB-1, la focalisation de contrôle est la prochaine ISC qui

doit être activée par le système. En revanche, dans HASP/SIAP la focalisation d'attention est la prochaine région du blackboard vers laquelle le système doit orienter ses activités, alors que dans CRYSALIS et ATOME, c'est une combinaison des deux cas précédents.

1.3 Représentation du contrôle

Nous désignons par représentation du contrôle, l'ensemble des *structures de contrôle* ainsi que les *modules de contrôle* qui opèrent sur ces structures pour gérer la conduite du système.

Dans les architectures de blackboard, deux approches sont possibles pour représenter le mécanisme de contrôle généralement constitué de modules et de structures de contrôle :

- *Représentation implicite du contrôle :*

  Dans une architecture de blackboard, un contrôle est dit *implicite* lorsque les stratégies de raisonnement qu'il utilise apparaissent implicitement au cours du fonctionnement du système et n'avaient pas été spécifiées auparavant lors de la mise en œuvre du système.

  Le système HEARSAY-II utilise un programme monolithique englobant toutes les heuristiques du système. Ce dernier est chargé de sélectionner une ISC à activer dans un agenda d'ISCs activables. Pour ce faire, il applique l'ensemble des heuristiques qu'il renferme. Ces heuristiques prennent en compte l'état actuel du blackboard, les niveaux d'entrée et de sortie de chaque ISC, ainsi que les résultats attendus pour chacune d'elles.

  Une représentation implicite du contrôle présente de nombreuses limites. En effet, les systèmes à base de blackboard adoptant ce type de contrôle sont difficilement évolutifs et maintenables du fait que toutes les composantes du système sont étroitement liées et qu'une mise à jour de l'une d'elles implique des mises à jour d'autres composantes. De plus, ces systèmes se sont avérés incapables d'expliquer aisément leur raisonnement.

  Le seul avantage qu'ils présentent vient du fait que la sélection de la prochaine ISC à exécuter se fait d'une manière relativement plus rapide que les systèmes adoptant une représentation explicite des connaissances liées au contrôle [101].

- *Représentation explicite du contrôle :*

  Dans une architecture de blackboard, un contrôle est dit *explicite*, lorsque les stratégies de raisonnement qu'il utilise sont spécifiées explicitement

lors de la mise en œuvre du système. Par exemple, les systèmes HASP/SIAP, CRYSALIS, BB-1 et ATOME définissent des SCs *de contrôle* qui gèrent d'une façon explicite les SCs du domaine d'application.

Dans HASP/SIAP, CRYSALIS et ATOME, chaque SC de contrôle est chargée de coordonner *directement* les activités d'un sous-ensemble des SCs du domaine, tandis que dans BB-1, les SCs de contrôle opèrent sur un second blackboard. appelé *blackboard de contrôle*, afin de gérer d'une façon *indirecte* les activités de tout l'ensemble des SCs du domaine.

Une représentation explicite du contrôle présente plus d'avantages qu'une représentation implicite [98]. En effet, les systèmes à base de blackboard représentant explicitement leur problème de contrôle sont évolutifs, faciles à maintenir et modulaires. Les connaissances liées au contrôle peuvent être mises à jour sans remettre en cause systématiquement les connaissances liées au domaine et *vice-versa*.

Les chapitres 2, 3 et 4 de cette partie définissent de façon plus détaillée les principaux types de contrôle qui ont été adoptés dans les architectures de blackboard.

2
Contrôle procédural

Le contrôle procédural a été implanté pour la première fois dans le système HEARSAY-II [84,82,179], système de reconnaissance et de compréhension de la parole continue. Il a été ensuite étendu dans le système DVMT [176] appliqué à l'identification de véhicules dans une région placée sous surveillance à l'aide de capteurs. Dans ce chapitre, nous définissons d'abord le contrôle procédural. Nous présentons ensuite son implantation dans les systèmes HEARSAY-II et DVMT.

2.1 Principe

Une architecture à base de blackboard dont le contrôle est procédural résoud le problème du contrôle par l'intermédiaire d'un programme très complexe qui rassemble toutes les connaissances de contrôle. Ce programme est un *ordonnanceur* sophistiqué dont le rôle est de sélectionner la prochaine action à exécuter en fonction de l'état global du blackboard et des heuristiques de contrôle (cf. figure 2.1). Pour ce faire, il dispose d'un agenda des actions exécutables et de procédures lui permettant d'évaluer les heuristiques de contrôle à partir de l'état de la solution courante mémorisée dans le blackboard. Ces heuristiques sont utilisées pour calculer les priorités associées à chaque action potentielle et ainsi déterminer l'action la plus prioritaire qui sera exécutée. Les stratégies de contrôle sont donc déterminées de façon *implicite* par l'activation de procédures dans la mesure où elles ne sont pas spécifiées *a priori*. Elles sont de plus *opportunistes* car elles évoluent en fonction de l'état du blackboard.

Dans ce type d'architecture, le cycle de base du mécanisme de contrôle est constitué de deux phases : la première phase consiste à exécuter une source de connaissances qui effectue des changements dans le blackboard susceptibles d'intéresser un certain nombre d'autres sources de connaissances. Ces dernières sont alors instanciées et sont placées dans un agenda si leur précondition est

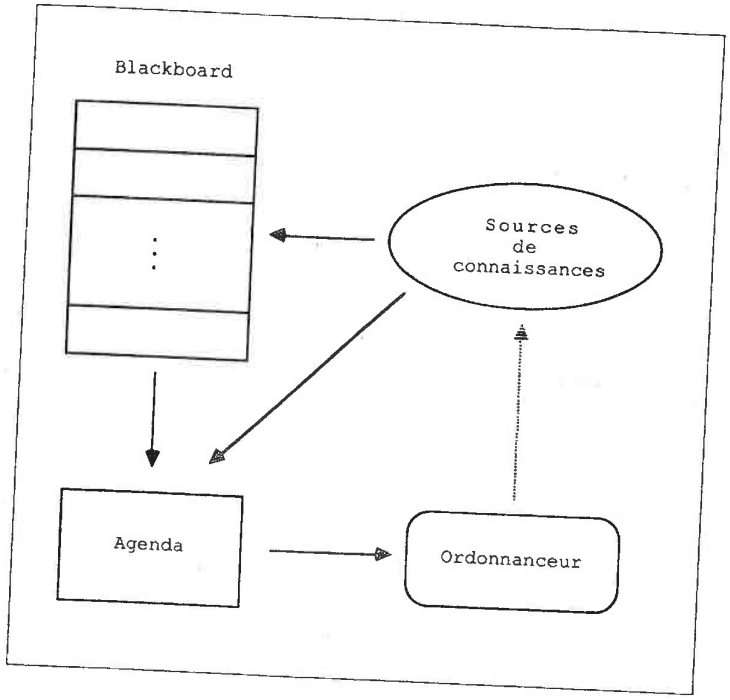


Figure 2.1. Contrôle procédural.

satisfaite. L'ordonnanceur trie ensuite l'agenda en calculant la priorité de chacune de ces sources de connaissances. L'ordonnanceur exécute ensuite la source de connaissances la plus prioritaire et le cycle recommence (cf. figure 2.1).

2.2 HEARSAY-II

HEARSAY-II a été développé à l'université de Carnegie Mellon dans le cadre d'un grand projet de recherche en reconnaissance et compréhension de la parole financé par l'agence DARPA (Defense Advanced Research Projects Agency) entre les années 1971 et 1976 [79].

En utilisant les techniques d'IA en parole, HEARSAY-II a donné naissance au modèle du blackboard et a inspiré de nombreuses implantations de ce modèle.

Le but de ce système est de reconnaître une phrase prononcée et de l'interpréter en tant que requête à une base de données. Celle-ci contient un ensemble des résumés de documents relevant de l'IA. Le système doit retrouver le ou les documents sollicités par la requête.

Dans la suite, nous allons montrer comment sont implantés les trois composants de base d'une architecture de blackboard dans HEARSAY-II et ainsi mettre en évidence son aspect procédural.

2.2.1 Le blackboard

Le blackboard est divisé en sept niveaux d'abstraction allant du niveau signal-paramétré au niveau interface-base-de-données, en passant par les niveaux segment, syllabe, mot, séquence-de-mots et phrase. Ces niveaux représentent des étapes intermédiaires dans la phase de décodage de l'expression prononcée et sont spécifiques à ce domaine d'application (cf. figure 2.2).

Chaque hypothèse dans le blackboard est caractérisée entre autres par sa valeur, sa validité, son niveau d'abstraction et ses temps de début et de fin dans la phrase prononcée.

Si deux ou plusieurs hypothèses d'un même niveau dans le blackboard se chevauchent dans le temps, elles sont dites *compétitives*, car elles représentent deux interprétations partielles incompatibles d'une même portion de la phrase prononcée. Dans le cas contraire, elles sont dites *coopératives*. Deux sources de connaissances peuvent donc être *compétitives* ou *coopératives* selon les hypothèses sur lesquelles elles travaillent.

2.2.2 Les sources de connaissances

Les parties condition et action d'une source de connaissances dans HEARSAY-II sont des programmes écrits dans un dialecte d'Algol-60. La partie condition évalue si la configuration courante du blackboard justifie l'émission des hypothèses contenues dans la partie action de la source de connaissances. Dans l'affirmative, la partie action devient exécutable.

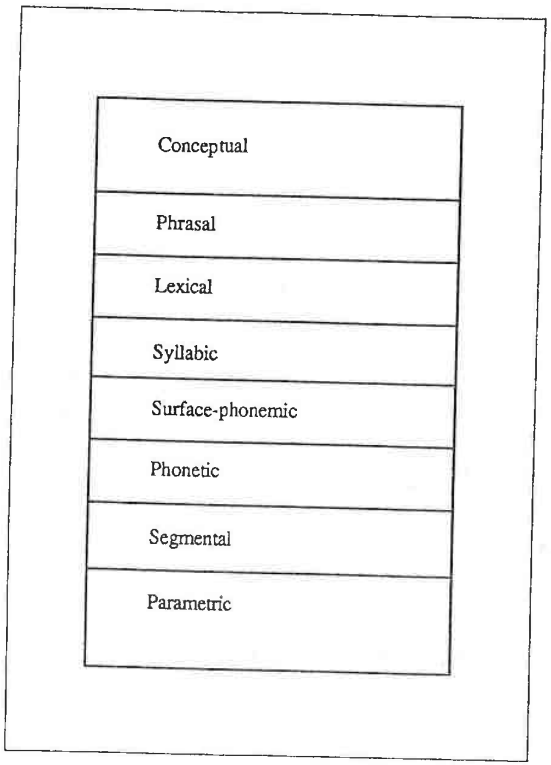


Figure 2.2. Les sept niveaux du blackboard dans HEARSAY-II (d'après [50]).

Les SCs de HEARSAY-II sont illustrées par les figures 2.3 et 2.4.

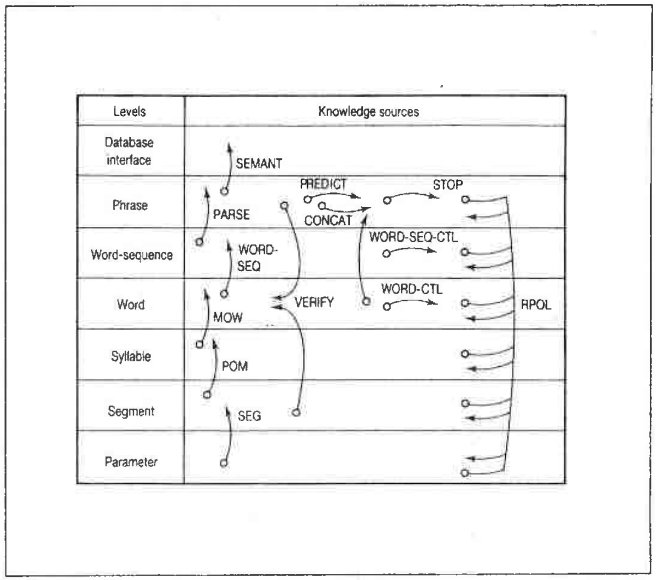
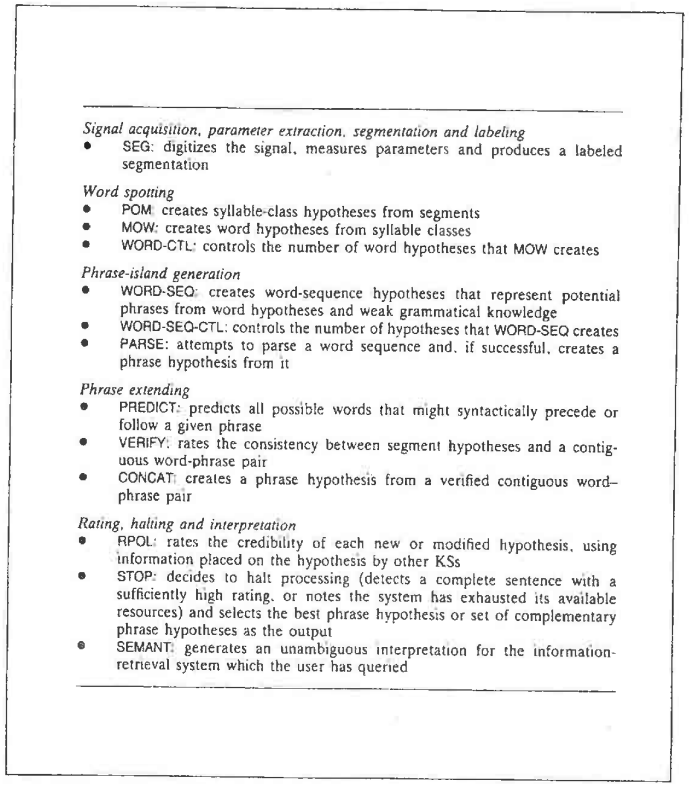


Figure 2.3. Les SCs dans HEARSAY-II (d'après [82]).

Deux champs supplémentaires sont associés à chaque SC : le *stimulus-frame* qui contient les hypothèses du blackboard ayant satisfait la partie condition de la SC; et le *response-frame* qui contient une estimation des travaux potentiels de la SC. La prise en compte de ces champs par le mécanisme de contrôle est décrite dans le paragraphe suivant.



Signal acquisition, parameter extraction, segmentation and labeling
- SEG: digitizes the signal, measures parameters and produces a labeled segmentation

Word spotting
- POM: creates syllable-class hypotheses from segments
- MOW: creates word hypotheses from syllable classes
- WORD-CTL: controls the number of word hypotheses that MOW creates

Phrase-island generation
- WORD-SEQ: creates word-sequence hypotheses that represent potential phrases from word hypotheses and weak grammatical knowledge
- WORD-SEQ-CTL: controls the number of hypotheses that WORD-SEQ creates
- PARSE: attempts to parse a word sequence and, if successful, creates a phrase hypothesis from it

Phrase extending
- PREDICT: predicts all possible words that might syntactically precede or follow a given phrase
- VERIFY: rates the consistency between segment hypotheses and a contiguous word-phrase pair
- CONCAT: creates a phrase hypothesis from a verified contiguous word–phrase pair

Rating, halting and interpretation
- RPOL: rates the credibility of each new or modified hypothesis, using information placed on the hypothesis by other KSs
- STOP: decides to halt processing (detects a complete sentence with a sufficiently high rating, or notes the system has exhausted its available resources) and selects the best phrase hypothesis or set of complementary phrase hypotheses as the output
- SEMANT: generates an unambiguous interpretation for the information-retrieval system which the user has queried

Figure 2.4. Rôles des SCs dans HEARSAY-II (d'après [82]).

2.2.3 Le contrôle

Principe de fonctionnement

Dans HEARSAY-II, le contrôle est de type procédural; il est réalisé à l'aide d'un ordonnanceur et d'un *moniteur du blackboard*. Le rôle du moniteur est de détecter les parties condition des sources de connaissances intéressées par les types de changements effectués dans le blackboard (événements) et de les placer dans un agenda contenant l'ensemble des *activités* à exécuter. A cette fin, il dispose d'une table définie *a priori* qui, à chaque événement possible, associe l'ensemble des SCs intéressées. Le moniteur du blackboard doit également mettre à jour une structure de contrôle qui contient les changements qui ont eu lieu dans le blackboard.

L'ordonnanceur, quant à lui, calcule les priorités des activités à exécuter en fonction de l'état global du système (blackboard, structure de contrôle ...) et choisit l'activité la plus prioritaire. Une activité peut être soit une partie condition d'une SC placée dans l'agenda par le moniteur du blackboard, soit une partie action d'une SC placée dans l'agenda par l'ordonnanceur lui-même. En effet, lorsque l'ordonnanceur sélectionne la partie condition d'une SC et que cette dernière est vérifiée, les deux champs *stimulus-frame* et *response-frame* de cette SC sont mis à jour et sa partie action est placée dans l'agenda. Les informations contenues dans le *stimulus-frame* et le *response-frame* sont prises en compte par l'ordonnanceur dans le calcul des priorités de la partie action concernée.

En résumé, le cycle de base dans HEARSAY-II est le suivant :

- L'ordonnanceur sélectionne une activité à exécuter dans l'agenda.
- Si cette activité est une partie condition d'une source de connaissances et si elle est satisfaite, le *stimulus-frame* et le *response-frame* de la source de connaissances sont évalués.
- Si cette activité est une partie action, des changements sont effectués dans le blackboard, le moniteur du blackboard repère les parties condition intéressées et les place dans l'agenda.

La figure 2.5 représente l'architecture générale de HEARSAY-II.

Calcul des priorités

Le calcul de la priorité d'une source de connaissances est basé sur les principes suivants [132] :

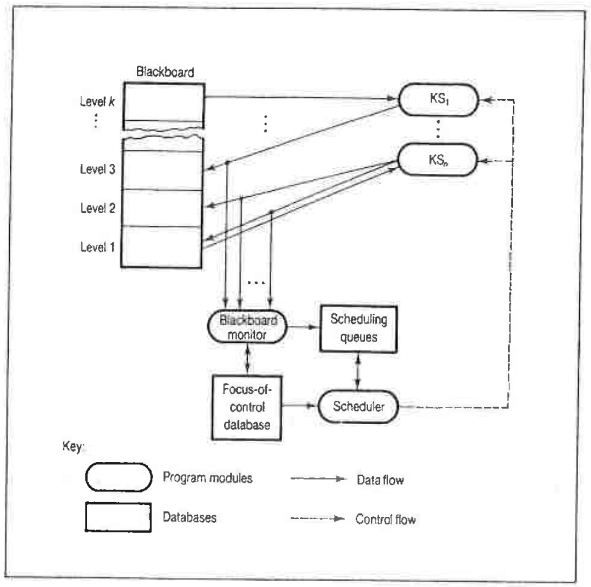


Figure 2.5. Architecture du système HEARSAY-II (d'après [82]).

- *Principe de compétition :*

  Le but de ce principe est de sélectionner la meilleure SC parmi un ensemble de SCs susceptibles de travailler sur des hypothèses compétitives. Pour cela, la faveur est donnée à la SC dont le *response-frame* propose les hypothèses plus crédibles, de plus haut niveau dans la hiérarchie du blackboard, et de durée plus grande (interprétation d'une plus grande portion de la phrase prononcée) que les hypothèses déjà existantes dans le blackboard.

- *Principe de crédibilité :*

  Ce principe fait intervenir la crédibilité des hypothèses qui apparaissent dans le *stimulus-frame* des sources de connaissances exécutables. Cette crédibilité est prise en compte lorsque le système cherche à évaluer les

différentes actions possibles d'une SC. En effet, lorsque deux ou plusieurs actions d'une même SC sont possibles, le système favorise l'action dont le *stimulus-frame* porte sur les hypothèses les plus crédibles.

- *Principe d'importance :*

  Ce principe consiste à choisir les sources de connaissances dont le *response-frame* est le plus important. Un *response-frame* est d'autant plus important qu'il contient des actions possibles importantes. L'importance d'une action est une fonction croissante du niveau d'abstraction sur lequel porte cette action.

- *Principe d'efficacité :*

  Ce principe est fondé sur l'efficacité de l'exécution d'une source de connaissances.

- *Principe de satisfaction de but :*

  Ce principe favorise les sources de connaissances dont le *response-frame* satisfait au mieux les objectifs visés, c'est-à-dire les buts que l'on cherche à résoudre à ce moment de l'interprétation.

Ces principes font donc intervenir de nombreux facteurs contextuels qui ont été ramenés à des valeurs numériques. Les heuristiques qui évaluent, ajustent, combinent et pondèrent ces valeurs en fonction de l'état du blackboard sont intégrées dans des procédures au sein de l'ordonnanceur. Le calcul de la priorité d'une source de connaissances vis à vis de ces principes et de ces heuristiques relève d'opérations très complexes et très difficiles à mettre en œuvre [132]. Cette complexité a rendu l'ordonnanceur difficile à tester et à maintenir [118].

Stratégie implicite

Dans HEARSAY-II, aucune stratégie n'a été définie explicitement pour influencer l'ordonnanceur dans son choix des sources de connaissances à activer. Ce choix dépend uniquement des résultats des calculs des priorités affectées à chaque SC. Il s'avère pourtant que HEARSAY-II utilise une stratégie de contrôle qui résout le problème en deux phases. La première phase est dirigée par les données : quatre sources de connaissances émettant successivement des hypothèses sur les niveaux *signal paramétré*, *segment*, *syllabe* et *mot* sont activées *séquentiellement*. La seconde phase consiste ensuite à sélectionner de façon *opportuniste* les sources de connaissances exécutables [118]. Cette stratégie est dite *implicite* (cf. chapitre 1) car elle n'a pas été définie *a priori* par le contrôle (l'ordonnanceur). En effet, la première phase correspond à l'exécution

des quatre premiers cycles du système où *une seule* SC exécutable était en attente dans l'agenda à chacun de ces cycles. En revanche, durant la seconde phase, *plusieurs* sources de connaissances exécutables étaient en attente dans cet agenda (cf. figure 2.6).

Un exemple complet d'exécution de HEARSAY-II est donné dans [82].

2.2.4 Conclusion

Dans le domaine de la parole, le système HEARSAY-II n'a pas satisfait les objectifs visés par l'agence DARPA. Cependant, il a été d'un grand intérêt en IA : son architecture a servi de base à de nombreux travaux. En effet, elle a été réutilisée et complétée dans de nombreux systèmes touchant divers domaines d'application tels que la vision, l'interprétation de signaux complexes, la robotique, la cristallographie par rayons X...et dans des générateurs de systèmes à multi-sources de connaissances.

De nouveaux modèles de contrôle sont apparus et ont introduit une représentation déclarative des connaissances de contrôle. Nous présentons ces modèles dans les chapitres suivants. Auparavant, nous décrivons comment le système DVMT intègre et étend l'architecture de HEARSAY-II décrit précédemment.

2.3 DVMT

DVMT (Distributed Vehicle Monitoring Testbed) est un outil de recherche très complet développé à l'université de Massachusetts pour évaluer empiriquement les différentes techniques mises en œuvre pour la résolution de problèmes dans un système distribué [177]. DVMT est un système d'identification de véhicules circulant dans une région sous surveillance. Pour ce faire, plusieurs capteurs sont placés dans une zone géographique donnée. Cette zone est découpée en plusieurs régions qui peuvent éventuellement se recouvrir. Les capteurs d'une région donnée détectent la présence de véhicules et envoient les informations sur les signaux reçus à un ou plusieurs systèmes chargés de les interpréter (cf. figure 2.7). La coopération entre tous les systèmes présents doit permettre de retrouver la trace complète de tous les véhicules qui se déplacent dans la zone [42].

Après avoir rappelé brièvement ce que nous entendons par système d'IA distribué, nous nous intéresserons à l'architecture adoptée pour un *nœud* (cf. 2.3.1) de DVMT, basée sur le modèle du blackboard et à contrôle procédural. Le lecteur intéressé par l'organisation et le fonctionnement du système DVMT

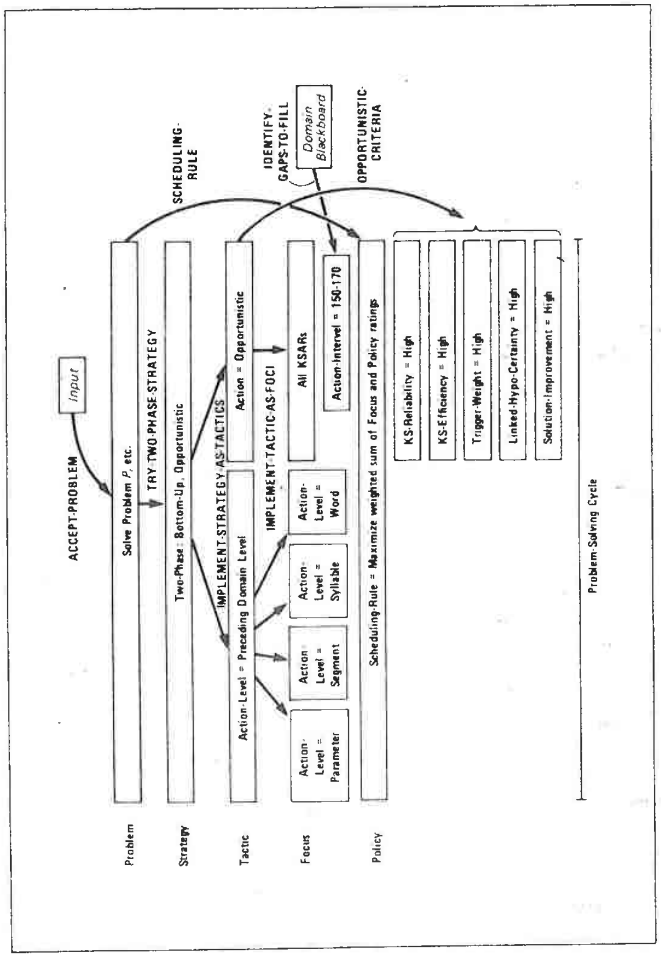


Figure 2.6. Stratégie implicite de HEARSAY-II (d'après [118]).

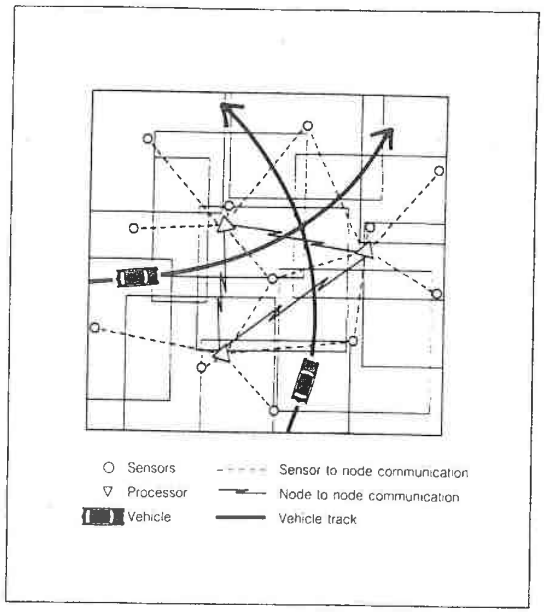


Figure 2.7. Domaine d'application de DVMT (d'après [176]).

complet pourra consulter [175,47,69,68,74,71,73,72,212]

2.3.1 Système IA distribué

Un système d'IA distribué est composé d'un réseau de nœuds, c'est-à-dire de sous-systèmes qui, en communiquant, coopèrent entre eux afin de résoudre un même problème. Cet ensemble de nœuds peut être *logiquement* ou *géographiquement* distribué (dans DVMT, les nœuds sont géographiquement distribués).

Chacun des nœuds n'a qu'une vue partielle du problème initial et doit essayer de résoudre cette partie du problème. A cette fin, il dispose d'un ensemble de connaissances qu'il gère lui-même selon son propre mode de contrôle.

Globalement, chaque nœud est capable de résoudre sa partie du problème

mais le résultat obtenu risque d'être incomplet. Une meilleure résolution est obtenue si chaque nœud coopère et échange des informations avec les autres. Ces informations peuvent porter sur des données, des buts à atteindre ou des tâches à réaliser. L'intégration et l'harmonisation des solutions partielles obtenues par cette coopération sont censés fournir une solution globale au système [58,140].

2.3.2 Architecture d'un nœud de DVMT

Un nœud dans le système DVMT a une architecture similaire mais plus sophistiquée que celle de HEARSAY-II. Celle-ci a été adaptée au problème d'identification de véhicules. Le blackboard et les sources de connaissances d'un nœud de ce système sont illustrés par la figure 2.8.

En essayant d'adopter l'organisation du système HEARSAY-II (cf. figure 2.9) au problème d'identification de véhicules, Corkill, Lesser et Hudlicka [48] ont mis en évidence quelques faiblesses du contrôle de type HEARSAY-II :

- Le choix de la meilleure instance de source de connaissances (ISC) en attente dans la file des ISCs exécutables est local. L'ordonnanceur prend en considération seulement l'état actuel du blackboard pour sélectionner une ISC à activer. Le système ne raisonne que sur les effets immédiats dûs à l'activation d'une ISC : il ne tient pas compte des effets à moyen et long terme de ces actions.

- Quand la précondition d'une SC échoue par manque d'informations nécessaires pour déclencher la SC, l'ordonnanceur ne mémorise pas ces informations manquantes et n'a aucun moyen de recalculer les priorités attribuées aux ISCs en attente afin de déclencher une ou plusieurs ISCs susceptibles d'engendrer ces informations.

Afin de corriger les limites décrites auparavant comme présentes dans les contrôles procéduraux de type HEARSAY-II, l'architecture d'un nœud de DVMT a été étendue [48] :

- Un second blackboard, appelé *le blackboard des buts*, a été défini. Il est composé des mêmes niveaux que le blackboard des données, mais contient des buts. Chaque but représente une requête de création d'hypothèses dans le blackboard des données (cf. figure 2.10). Par exemple, un but peut désigner le désir de créer une hypothèse avec certaines valeurs d'attributs dont la crédibilité dépasse un certain seuil, et dont la position doit être dans une certaine région du blackboard des données.

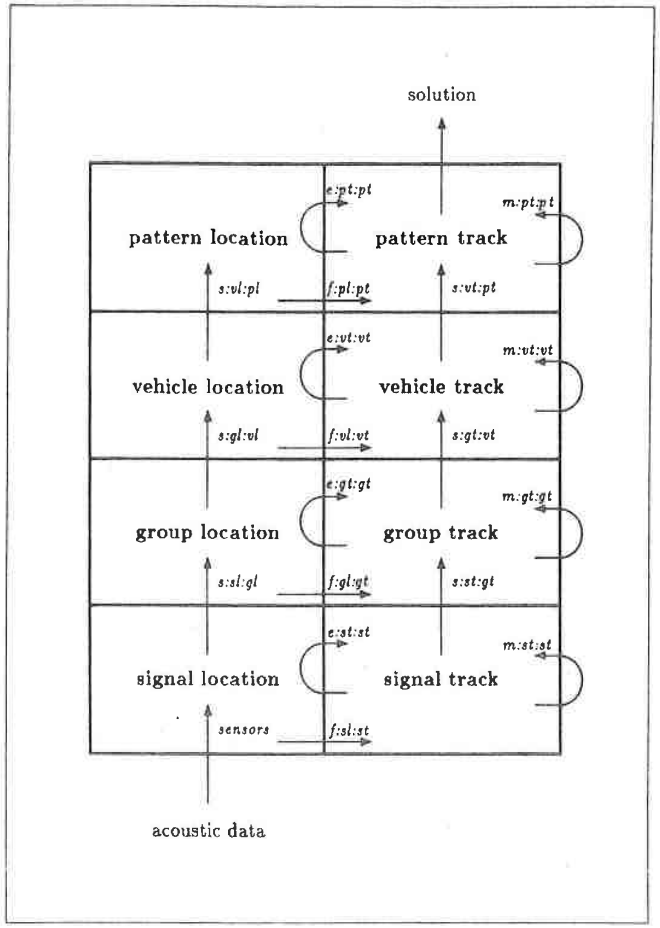


Figure 2.8. Le blackboard et les SCs de DVMT (d'après [67]).

- Un *planificateur* qui répond à la création de buts dans le blackboard des buts par la mise en œuvre de *plans* d'actions, c'est-à-dire des séquences de ISCs qui résoudraient ces buts.

2.3.3 Fonctionnement d'un nœud de DVMT

Quand une hypothèse est formée dans le *blackboard des données*, elle engendre un événement. En utilisant une table, définie *a priori*, qui associe un ensemble de buts à résoudre à chaque type d'événement engendré, le moniteur du blackboard engendre des buts exprimant le désir d'élargir ou d'abstraire l'hypothèse à l'origine de l'événement considéré (cf. figure 2.10).

Ces buts sont alors insérés dans le blackboard des buts. S'il est complexe, un but peut être éventuellement décomposé en sous-buts. Pour ce faire, le système utilise une seconde table, définie également *a priori*, qui associe à chaque but l'ensemble de ses sous-buts. Ces derniers seront à leur tour insérés dans le blackboard des buts et éventuellement décomposés...

En utilisant une troisième table associant à chaque but l'ensemble des SCs susceptibles de le satisfaire, le planificateur instancie un sous-ensemble de ces SCs et les place dans la file des ISCs activables.

L'ordonnanceur en fonction des relations existantes entre les buts et les SCs attribue alors une priorité à chaque entrée et déclenche l'ISC la plus prioritaire, puis le processus recommence.

La figure 2.10 illustre les différents composants de l'architecture d'un nœud dans le système DVMT.

2.4 Conclusion

Le principal avantage de HEARSAY-II, de DVMT et plus généralement des systèmes à contrôle procédural réside dans l'efficacité de leurs connaissances de contrôle représentées sous forme procédurale [98,101].

Ce type de contrôle est intéressant lorsque les connaissances de contrôle sont uniformes, structurées et bien organisées. Lorsque le contrôle devient incomplet et fragmenté, lorsqu'il faut combiner les différentes connaissances de contrôle et lorsque des stratégies complexes sont nécessaires, cette représentation procédurale des connaissances de contrôle devient difficile à coder et à maintenir. Une description déclarative (explicite) de ces connaissances semble alors plus appropriée.

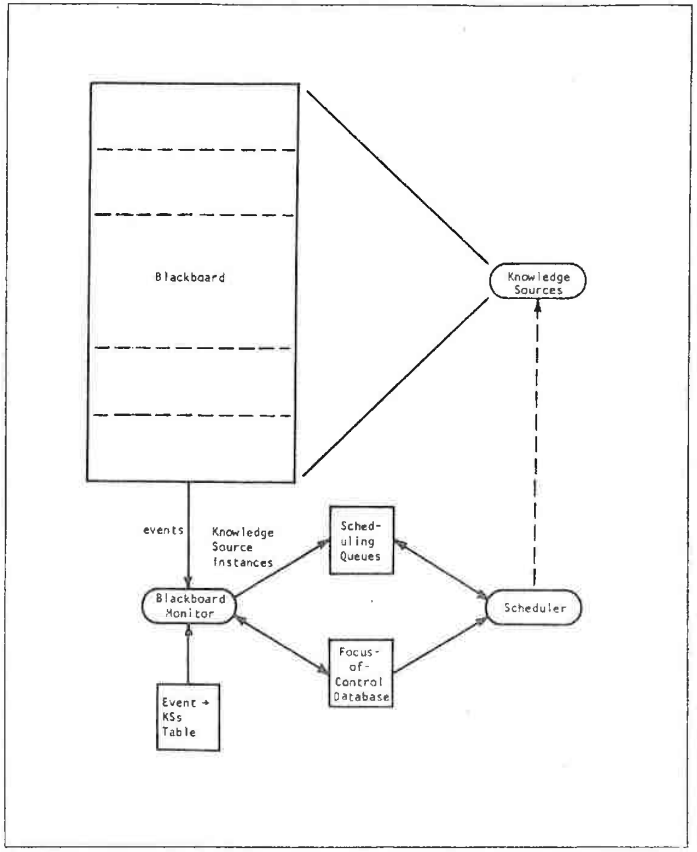


Figure 2.9. L'architecture de HEARSAY-II adaptée au système DVMT (d'après [42]).

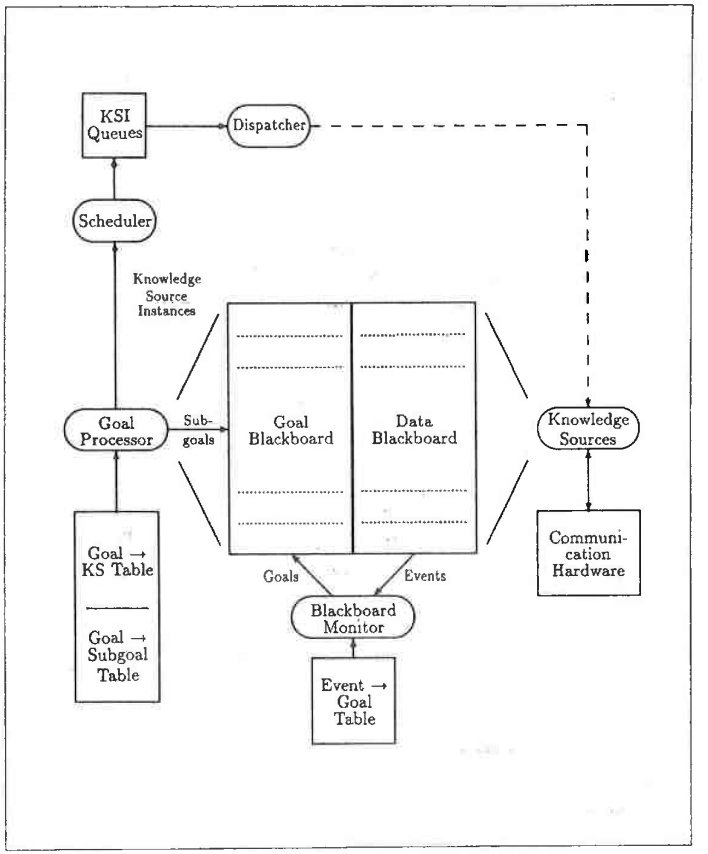


Figure 2.10. L'architecture d'un nœud de DVMT (d'après [73]).

3
Contrôle hiérarchique

Le contrôle hiérarchique dans les architectures de blackboard a été introduit par les systèmes HASP/SIAP [208,204] et CRYSALIS[1] [78,77]. Le premier système était destiné à identifier les navires se trouvant dans une région sous surveillance militaire. Le but du second système était de retrouver la structure tridimensionnelle des protéines en utilisant les techniques de cristallographie. Dans ce chapitre, nous définissons d'abord le contrôle hiérarchique. Nous présentons ensuite son implantation dans les systèmes HASP/SIAP et CRYSALIS.

3.1 Principe

Dans une architecture de blackboard à contrôle hiérarchique [37,104,232], toutes les connaissances du système, qu'elles soient du domaine ou de contrôle, sont représentées sous forme de sources de connaissances organisées en plusieurs niveaux hiérarchiques.

Le niveau le plus bas de la hiérarchie est constitué des SCs *du domaine* tandis que les niveaux supérieurs sont composés des SCs *de contrôle*. Le niveau le plus haut est constitué d'une seule SC de contrôle qui, à partir de la solution courante, active une ou plusieurs SCs du niveau immédiatement inférieur. Ce processus est répété : l'exécution d'une SC à un niveau provoque l'activation de SCs du niveau inférieur, jusqu'aux SCs du plus bas niveau (cf. figure 3.1).

3.2 HASP/SIAP

Le projet HASP soutenu par l'agence DARPA démarra en 1972 et se termina en 1975 à l'université de Stanford. Le projet redémarra à nouveau en 1976 sous le nom de SIAP.

1. Appelés initialement SU/X et SU/P respectivement [207].

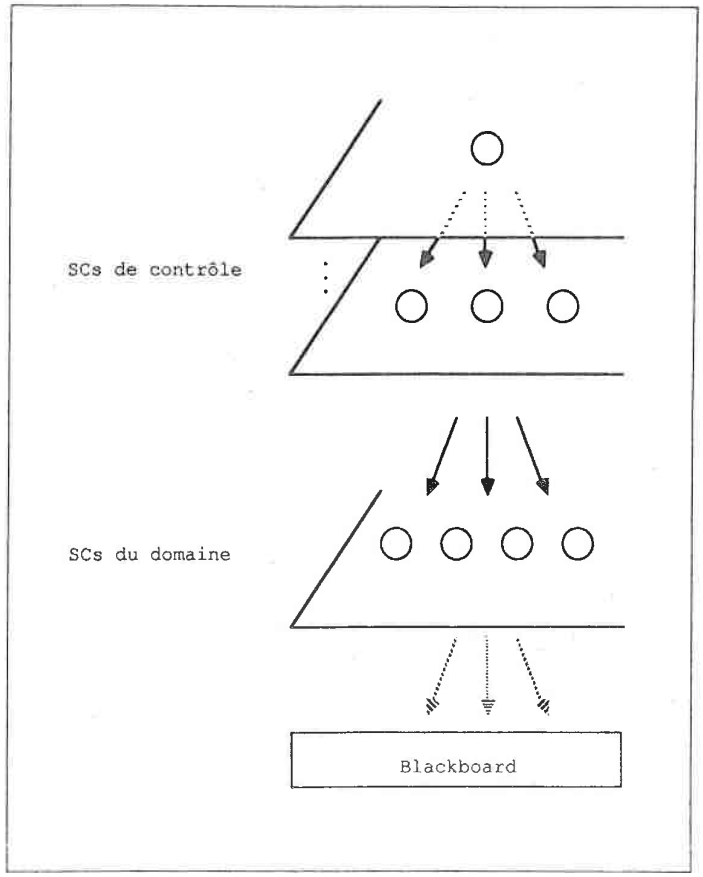


Figure 3.1. Contrôle hiérarchique.

Le but de ce système est d'interpréter des signaux sous-marins afin de reconnaître les sources de bruit les ayant engendrés et d'en déduire les navires présents dans la zone sous surveillance.

3.2.1 Le blackboard

Dans le système HASP/SIAP, le blackboard est composé de six niveaux : *segments*, *raies*, *harmoniques*, *sources émettrices de bruits*, *bâtiments* et *flotte*.

La structure du blackboard est hiérarchique : une raie regroupe des segments, une harmonique regroupe des raies, une source émettrice de bruits (hélice, moteur diesel...) est déduite à partir d'un ensemble d'harmoniques, un bâtiment est caractérisé par plusieurs sources de bruits et une flotte est composée de plusieurs bâtiments.

3.2.2 Les sources de connaissances

La partie condition d'une SC est formée d'une liste de prédicats, tandis que sa partie action est formée d'un paquet de règles de production.

La figure 3.2 illustre les différents niveaux ainsi que les SCs utilisés dans HASP/SIAP.

3.2.3 Le contrôle

Le mécanisme de contrôle est composé de quatre *structures (de contrôle)* manipulées par plusieurs *modules de contrôle* organisés en deux niveaux : le niveau *stratégie* et le niveau des *gestionnaires* (cf. figure 3.3).

Les structures de contrôle

Les quatre structures de contrôle sont des listes dont les éléments sont créés par les actions des SCs dans le blackboard.

- *Liste d'événements du blackboard.*

  Tous les changements du blackboard résultant des actions des SCs sont stockés dans cette structure de données.

- *Liste d'événements attendus.*

  Le système s'attend à rencontrer un certain nombre d'actions dans le blackboard. En effet, il dispose de connaissances *a priori* au sujet des activités prévues dans la région surveillée. Cette liste répertorie l'ensemble des changements prévus dans le blackboard.

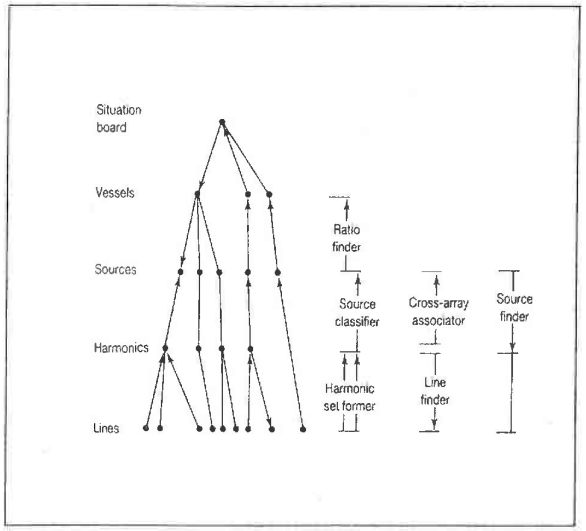


Figure 3.2. Le blackboard et les SCs dans HASP/SIAP (d'après [208]).

- *Liste de problèmes.*

  Elle mémorise les types de problèmes rencontrés par certaines SCs durant leur activation.

- *Liste d'événements temporels.*

  Un événement temporel est composé d'une date, d'un ensemble de SCs et d'un contexte de travail. Ces SCs seront exécutées à la date et dans le contexte spécifiés.

Les modules de contrôle

Le contrôle dans le système HASP/SIAP est organisé en deux niveaux. Au niveau le plus haut figure un seul module de contrôle, la *stratégie* qui en fonction du contenu des quatre listes définies auparavant, sélectionne un *gestionnaire* du second niveau. Le gestionnaire active des SCs en fonction du contenu de la liste

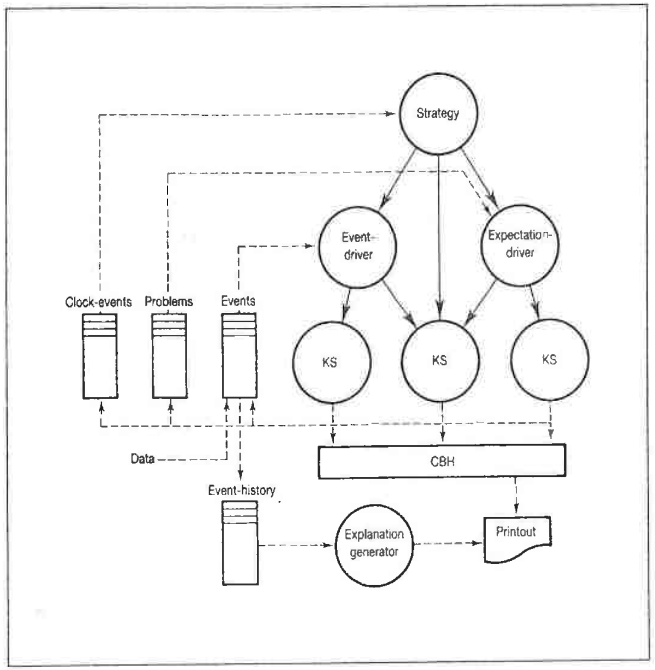


Figure 3.3. Organisation du système HASP/SIAP (d'après [208]).

qui lui est associée.

Le cycle de contrôle de base dans HASP/SIAP est le suivant :

1. La stratégie choisit la catégorie d'événements (événements du blackboard, événements attendus, problèmes ou événements temporels) qu'il faut traiter[2].

2. Les premiers documents décrivant HASP/SIAP [207,208] indiquaient que ce choix était effectué dans un ordre prédéfini, à savoir, événements temporels, problèmes, événements attendus puis événements du blackboard. Les dernières publications concernant ce système [200,201] précisent que cet ordre n'est plus prédéfini mais ne mentionnent pas les critères utilisés par la

2. Le gestionnaire associé à la catégorie choisie est alors activé :

   - Le gestionnaire des événements du blackboard choisit une entrée de la liste des événements du blackboard et exécute dans un ordre prédéfini un ensemble de SCs qui auront comme contexte cet événement (focalisation de contrôle).
   - Le gestionnaire des événements attendus vérifie si les entrées en attente dans la liste des événements attendus apparaissent dans la liste des événements du blackboard.
   - Le gestionnaire des problèmes vérifie si certaines entrées de la liste des événements répondent aux problèmes en attente d'être résolus dans la liste des problèmes. Il active le cas échéant les SCs ayant posé ces problèmes.
   - Le gestionnaire des événements temporels consulte régulièrement la liste des événements temporels afin d'activer aux dates indiquées les SCs associées.

   Le gestionnaire en question choisit donc une entrée dans la liste qui lui est associée et exécute dans un ordre prédéfini des SCs qui auront comme contexte d'activation l'entrée sélectionnée par le gestionnaire.

3. Les règles de production composant la partie action de la SC courante dont la partie gauche est satisfaite sont activées. Ainsi, elles mettront à jour la configuration du blackboard ainsi que les structures de contrôle et le cycle recommence.

3.3 CRYSALIS

Le projet CRYSALIS [237,236] a été développé entre les années 1976 et 1983 dans le cadre d'une collaboration entre des experts en cristallographie de l'université de San Diego et des membres du projet HPP (Heuristic Programming Project) de l'université de Stanford [4].

Le but de ce système est de retrouver la structure tridimensionnelle de protéines à partir de la diffraction par rayons X. L'approche blackboard s'est révélée nécessaire pour la résolution de ce problème et a été adoptée.

3.3.1 Le blackboard

Dans CRYSALIS, le blackboard est divisé en deux plans hiérarchiques :

stratégie pour effectuer son choix.

- Le *plan densité* :

  Ce plan est divisé en quatre niveaux permettant de donner une description de la protéine à analyser (cf. figure 3.4). Il contient les données du problème obtenues après prétraitement des résultats de la diffraction.

- Le *plan hypothèse* :

  Ce plan mémorise l'évolution de la résolution du problème. Il est divisé en trois niveaux d'abstraction mémorisant les éléments de la chaîne constituant la protéine et leur position dans l'espace tridimensionnel à différents niveaux de détails. Les hypothèses émises dans ce plan sont déduites du plan densité et des hypothèses déjà présentes. Le plan hypothèse constitue lui même un niveau d'abstraction du plan densité (cf. figure 3.4).

3.3.2 Les sources de connaissances

Le système CRYSALIS comporte deux types de sources de connaissances : les SCs de contrôle définies en 3.3.3 et les SCs du domaine.

Les SCs du domaine sont les sources de connaissances qui ont accès au blackboard. A la différence des autres systèmes fondés sur le modèle du blackboard, elles n'ont pas le format condition-action. Elles sont en effet réduites à leur partie action (sous forme de base de règles) et ne peuvent donc pas spécifier les conditions sous lesquelles elles peuvent effectivement contribuer à la résolution du problème[3].

3.3.3 Le contrôle

CRYSALIS définit deux structures de contrôle nécessaires au déroulement du système :

- *Une liste des formes :*

  Cette liste visualise au cours de la résolution du problème la chaîne des acides aminés et le résultat de l'analyse de leur répartition spatiale dans la molécule.

- *Une liste d'événements :*

  Cette liste contient l'ensemble des changements dans le blackboard (événements) qui n'ont pas encore participé à la résolution du problème.

3. Pour cette raison, Penny Nii a qualifié CRYSALIS de *blackboard-like system* [201].

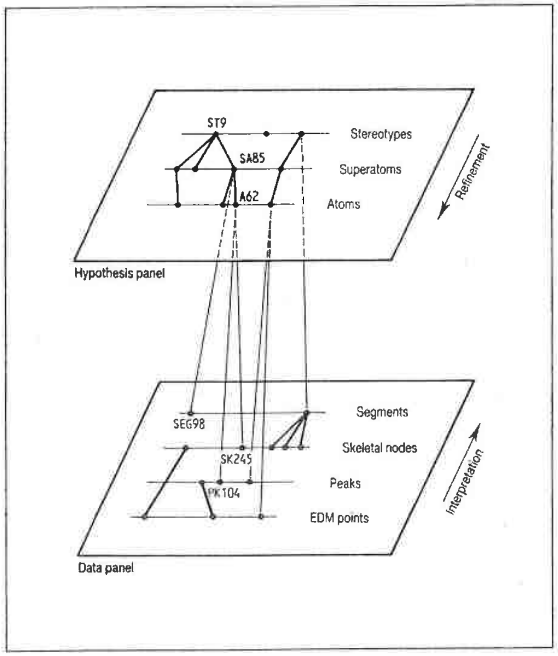


Figure 3.4. Représentation du blackboard dans CRYSALIS (d'après [236]).

Les connaissances de contrôle sont représentées sous forme de SCs de contrôle organisées en deux niveaux hiérarchiques :

- Le niveau le plus haut :

  Ce niveau est constitué d'une seule SC de contrôle appelée *stratégie*, qui consulte la liste des formes afin d'évaluer l'état de la résolution et de déterminer la démarche à suivre. Ainsi, la stratégie peut orienter le système vers une région de données où l'analyse n'a pas encore abouti. Pour ce faire, elle activera séquentiellement une ou plusieurs SCs du niveau inférieur (les tâches) en leur passant cette région de données sous forme de *focalisation de contrôle*.

- Le second niveau :

  Ce niveau est composé de SCs de contrôle appelées *tâches*. Le rôle de chaque tâche est d'activer séquentiellement une ou plusieurs SCs du domaine en fonction des événements apparus dans la liste des événements et de la focalisation de contrôle qui lui est fournie par la stratégie.

  Les tâches peuvent à leur tour orienter le travail des SCs du domaine en leur passant en focalisation de contrôle les événements qui ont permis leur activation. Chaque tâche possède la même méthodologie de contrôle des SCs du domaine placées sous sa responsabilité.

Ce fonctionnement des SCs de contrôle constitue le cycle de base de CRYSALIS et est illustré par la figure 3.6. La figure 3.5 représente la hiérarchie entre les SCs de contrôle et celles du domaine.

3.4 Conclusion

Un contrôle hiérarchique dans une architecture de blackboard permet :

- *Une représentation explicite et modulaire des connaissances liées au contrôle.*

  En effet, cette approche permet de formaliser les connaissances de contrôle de façon structurée et modulaire sous forme de sources de connaissances organisées hiérarchiquement. Cette représentation offre au système une clarté, une souplesse et une évolutivité qui facilitent considérablement le maintien du système.

- *Une formalisation structurée du problème.*

  Une telle approche permet d'aborder un problème en le décomposant en sous-problèmes à résoudre suivant le principe *diviser pour régner*.

- *Une représentation plus simple des stratégies à utiliser pour gérer et contrôler l'ensemble des sources de connaissances.*

  A la différence des contrôles procéduraux (cf. chapitre 2) ou à base de blackboard (cf. chapitre 4), le nombre de SCs compétitives et la complexité des stratégies à mettre en œuvre pour résoudre le conflit entre ces SCs sont considérablement réduits. En effet, à chaque cycle du processus de résolution du problème, seulement un sous-ensemble de sources de connaissances est pris en compte.

- *La mise en œuvre d'un contrôle plus efficace que celui fondé sur des agendas avec calcul de priorité.*

Associer à chaque événement une séquence de SCs à activer évite au concepteur de définir des heuristiques complexes à appliquer sur des agendas pour sélectionner des SCs, comme c'est le cas dans les architectures dont le contrôle est à base de blackboard (cf. chapitre 4). L'activation des sources de connaissances vers une région de données passée en focalisation de contrôle conduit à une meilleure efficacité du système.

Néanmoins, l'utilisation d'un contrôle hiérarchique dans une architecture de blackboard requiert une expertise plus fine du concepteur : en plus des conditions d'activation, qui sont matérialisées par des événements, l'ordre d'exécution de chaque source de connaissances doit être connu, ce qui n'est pas toujours facile à formaliser et conduit à une certaine rigidité du système [118].

Ce type de contrôle est donc conseillé pour des applications d'IA où le flux de contrôle est bien maîtrisé. Malheureusement, de nombreuses applications nécessitent une structure plus souple, où l'ordre d'activation peut être prédéfini pour un sous-ensemble de ses SCs et déterminé dynamiquement pour d'autres SCs en fonction de certaines heuristiques.

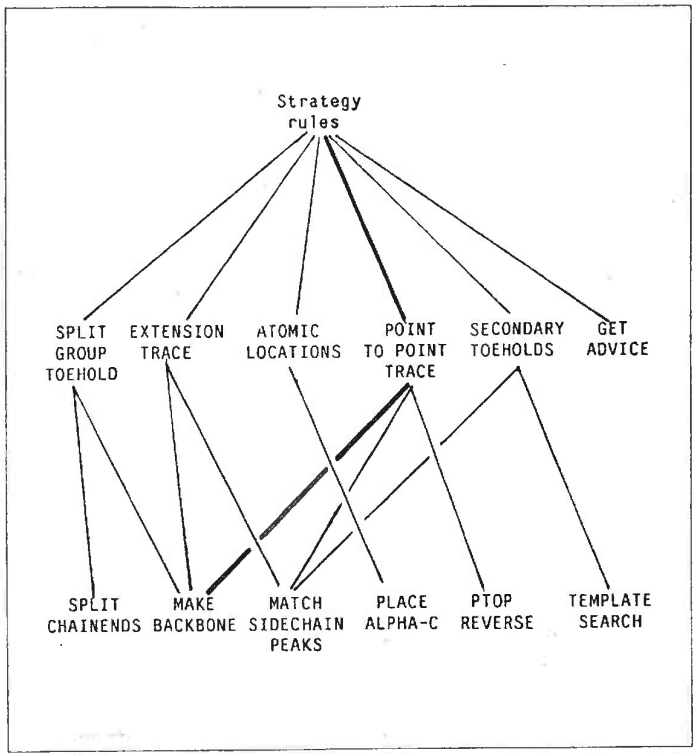


Figure 3.5. Les sources de connaissances de CRYSALIS (d'après [237]).

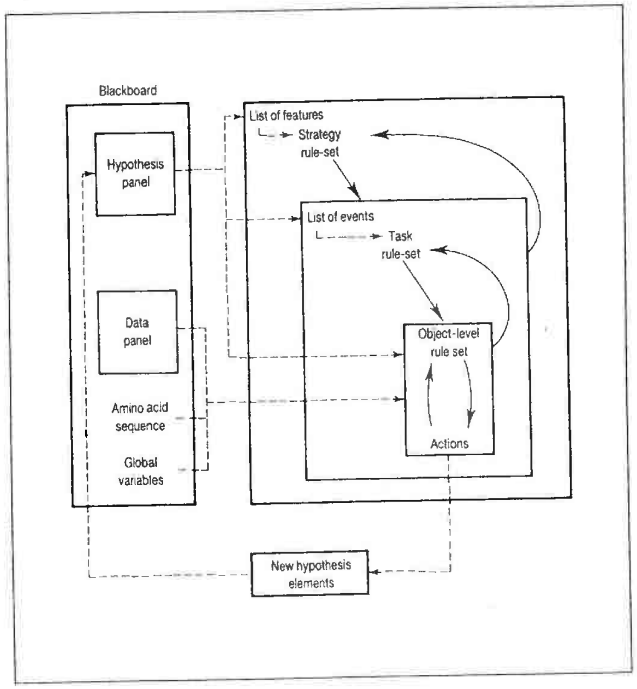


Figure 3.6. Le cycle de base dans CRYSALIS (d'après [236]).

4
Contrôle à base de blackboard

Le contrôle à base de blackboard a été implanté pour la première fois dans le système BB-1 [99,121,125,123], et a été ensuite amélioré dans des systèmes tels que GBB [150], BBB [18] et ERASMUS [16,63,145].

Dans ce chapitre, nous définissons d'abord le contrôle à base de blackboard. Nous présentons ensuite son implantation dans les systèmes BB-1 et GBB, générateurs de systèmes à multi-sources de connaissances.

Nous décrivons également les originalités et les faiblesses que présente une architecture dont le contrôle est à base de blackboard.

4.1 Principe

Dans une architecture dont le contrôle est à base de blackboard, les connaissances de contrôle sont représentées sous la forme d'un ensemble de SCs *de contrôle* qui interagissent et communiquent à travers un second blackboard : le *blackboard de contrôle*. Ainsi, les SCs liées au contrôle manipulent le blackboard de contrôle au même titre que les SCs du domaine opèrent sur le blackboard du domaine. Cette approche représente le contrôle dans une architecture de blackboard par une autre architecture de blackboard [118]. Le rôle des SCs du domaine est de construire la solution dans le blackboard du domaine, tandis que les SCs de contrôle sont chargées de construire et d'adapter le plan de contrôle à suivre pour gérer la conduite du système. Ce plan est mémorisé dans le blackboard de contrôle. Les sources de connaissances, que celles-ci soient du domaine ou de contrôle, sont toutes concurrentes et traitées par un même ordonnanceur (cf. figure 4.1).

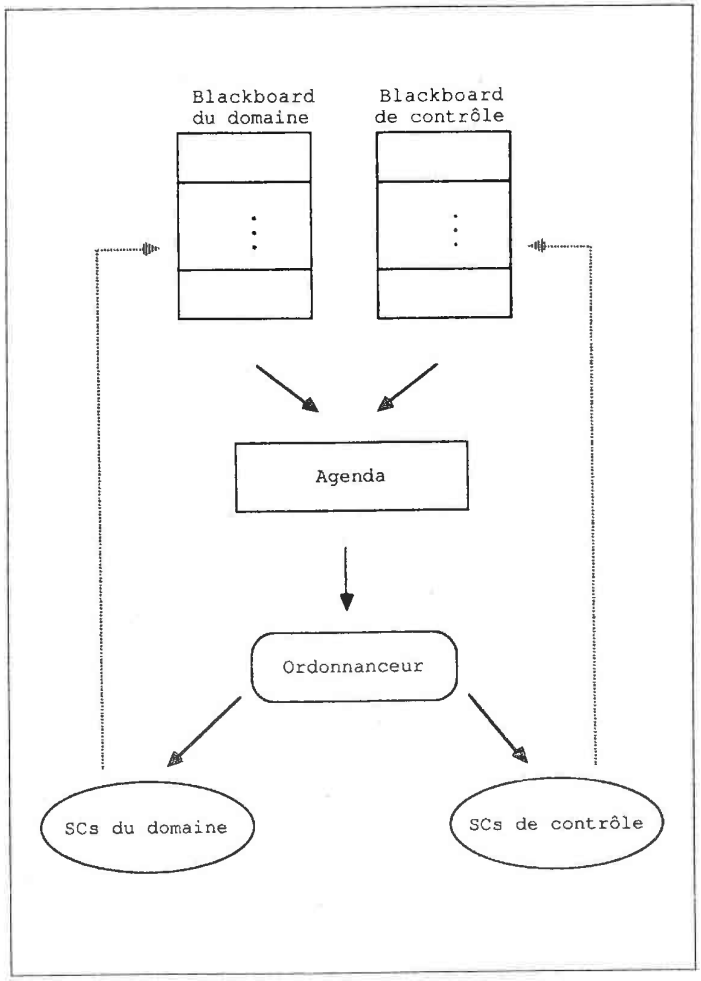


Figure 4.1. Contrôle à base de blackboard.

4.2 BB-1

BB-1 est un outil d'aide au développement de systèmes à multi-sources de connaissances développé au KSL (Knowledge Systems Laboratory) de l'université de Stanford [4]. Il est à compter parmi les premiers systèmes à avoir implanté le modèle du blackboard avec une structure de contrôle elle-même fondée sur ce modèle [100,135]. HEARSAY-III [13,87,62] fut le premier système à avoir introduit l'idée du contrôle à base de blackboard. Néanmoins, son contrôle est beaucoup moins élaboré que celui de BB-1.

4.2.1 Le blackboard

BB-1 introduit deux types de blackboard : le *blackboard du domaine* qui stocke les différents résultats intermédiaires de la solution courante au même titre que le blackboard défini dans les architectures à contrôle procédural ou hiérarchique, et le *blackboard de contrôle* qui contient le plan de contrôle que doit suivre le processus de résolution du système.

Le blackboard du domaine

Le blackboard du domaine est défini par le concepteur de l'application développée avec BB-1. Les différents niveaux de ce blackboard, leurs propriétés et les liens entre ces niveaux sont directement liés à l'application en cours de développement. Ce blackboard contient donc les hypothèses émises par les différentes SCs du domaine.

Le blackboard de contrôle

Les différents niveaux de ce blackboard sont prédéfinis dans BB-1 et sont indépendants de l'application en cours de réalisation. Parmi ces niveaux, nous citerons les trois plus importants :

- *Le niveau stratégie :*

  Ce niveau contient les différentes stratégies qui représentent des descriptions abstraites et générales du comportement que doit suivre le processus de résolution du système. Le temps de validité des stratégies est relativement long : généralement une stratégie reste valide pendant toute la période de résolution de l'application. Ce niveau peut contenir plusieurs stratégies compétitives, complémentaires ou indépendantes.

- *Le niveau objectif :*

Ce niveau mémorise les différents objectifs qui implantent et réalisent une ou plusieurs stratégies. Les décisions de ce niveau établissent des buts locaux opérationnels pour une période de temps relativement courte par rapport à celle des stratégies. De même qu'au niveau stratégie, plusieurs objectifs peuvent être compétitifs, complémentaires ou indépendants.

Généralement, une stratégie est composée d'une séquence d'objectifs; un objectif pouvant faire partie de plusieurs stratégies. Si la complexité de l'application l'exige, BB-1 permet de définir d'autres niveaux, tels que les niveaux sous-stratégie ou tactique, et de les intercaler entre les niveaux stratégie et objectif du blackboard de contrôle.

- *Le niveau heuristique :*

  Ce niveau enregistre les heuristiques qui "implantent" un objectif. Une heuristique décrit la façon dont les différents coefficients associés à une action doivent être évalués et combinés. Ces coefficients caractérisent par exemple le coût ou la fiabilité de l'action.

  Le mécanisme de contrôle applique un ensemble d'heuristiques opérationnelles sur une file d'actions en attente d'être activées. Il sélectionne ensuite la meilleure action face à ces heuristiques, et de ce fait celle qui répond au mieux au plan de contrôle spécifié par l'état courant du blackboard de contrôle.

Si la structure du blackboard de contrôle est prédéterminée et fixe pour toute application développée à l'aide de BB-1, il n'en est pas de même pour son contenu. En effet, les différents éléments du blackboard de contrôle, c'est-à-dire les décisions de contrôle telles que les stratégies, les objectifs et les heuristiques, sont liés au domaine d'application. Ces décisions sont mises à jour par les SCs de contrôle et interviennent dans la boucle de contrôle de base de BB-1 pour la sélection d'une action à exécuter (cf. § 4.2.3).

4.2.2 Les sources de connaissances

Un système développé à l'aide de BB-1 possède deux types de sources de connaissances :

1. *Les sources de connaissances du domaine :*

   Ce sont les SCs qui ont accès au *blackboard du domaine* (cf. figure 4.2). Elles représentent l'équivalent des sources de connaissances ou des spécialistes définies dans les architectures à contrôle procédural ou hiérarchique. Ces SCs sont les spécialistes du domaine d'application et sont à définir par le concepteur de l'application.

2. *Les sources de connaissances de contrôle :*

   Ce sont les SCs qui créent des décisions dans le plan de contrôle (ces décisions sont alors dites *opérationnelles*) ou qui peuvent stopper la validité de ces décisions (dites alors *non-opérationnelles*) dans *le blackboard de contrôle* (cf. figure 4.2). Ces sources de connaissances permettent de construire et de maintenir le plan de contrôle à suivre dans la conduite du système. L'objectif étant d'avoir un comportement incrémental et opportuniste.

   On distingue deux types de SCs de contrôle :

   (a) *Les SCs de contrôle génériques :*

   Ces SCs sont indépendantes du domaine d'application et sont prédéfinies. Ce sont les SCs qui gèrent l'implantation d'une stratégie donnée en une suite d'objectifs.

   (b) *Les SCs de contrôle spécifiques du domaine d'application :*

   Ces SCs de contrôle sont donc à définir par le concepteur de l'application. Elles permettent de prendre des décisions de contrôle adaptées au domaine d'application en fonction de l'évolution de la solution courante.

Toutes les sources de connaissances de BB-1, qu'elles soient du domaine ou de contrôle, respectent la même syntaxe et ont le même comportement. Elles sont caractérisées en particulier par :

1. Leur partie *condition*, elle-même décomposée en deux champs :

   (a) Le champ *trigger* qui spécifie quand le système doit s'intéresser à cette source de connaissances. Ce champ porte sur des tests liés aux derniers changements de l'état du blackboard, c'est-à-dire les événements. Une source de connaissances peut demander à être réveillée à la suite d'une création ou d'une modification dans un blackboard (du domaine ou du contrôle).

   (b) Le champ *précondition*, qui décrit les conditions supplémentaires que doit vérifier cette source de connaissances pour être effectivement activée.

   Avant d'être exécutée, une SC réveillée, c'est-à-dire dont le champ *trigger* est verifié, doit s'assurer qu'elle peut effectivement contribuer à la résolution du problème. Son champ précondition est composé de tests sur l'état des blackboards lui permettant ainsi de faire cette vérification.

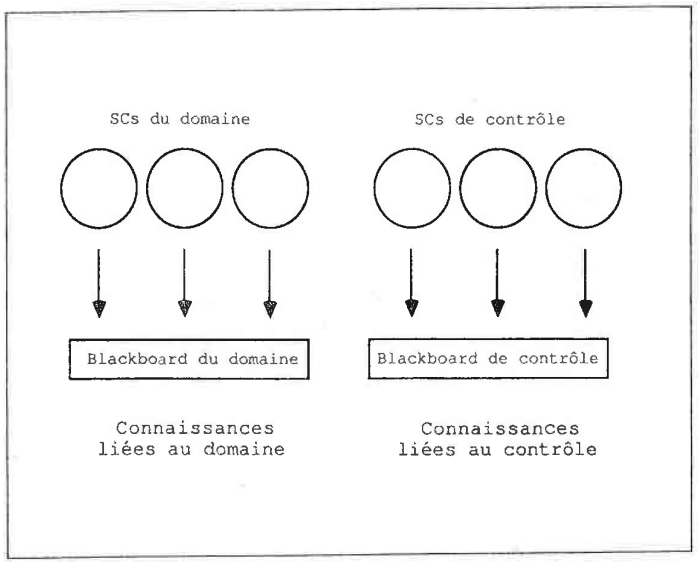


Figure 4.2. Les SCs du domaine et de contrôle opérant respectivement sur les blackboards du domaine et de contrôle.

2. Leur partie *action* est composée d'une base de règles de production. La partie gauche de ces règles est constituée de tests portant sur des hypothèses du blackboard, la partie droite est une suite d'actions à effectuer sur le blackboard : création de nouvelles entrées ou modification d'hypothèses déjà existantes.

3. Leurs *conditions de rejet*. Ces conditions spécifient le contexte dans lequel une action exécutable associée à la SC concernée est rejetée et supprimée de l'agenda des actions exécutables pour être mémorisée dans l'agenda des actions rejetées (cf. § 4.2.3).

4. Leur *contexte de travail*, qui constitue l'ensemble des hypothèses du blackboard sur lesquelles l'activation de la SC peut porter. Ces hypothèses sont mémorisées dans des variables manipulées dans les règles de la partie action de la SC. Ces variables sont liés à chaque activation de la SC.

5. Leur *coût*, valeur numérique qui fournit une estimation du coût de l'activation de cette SC.

6. Leur *fiabilité*, valeur numérique qui mesure la fiabilité attribuée au comportement et aux résultats de cette source de connaissances.

Le coût et la fiabilité d'une SC peuvent être exprimés en fonction de son contexte de travail et plus généralement de l'état des blackboards.

4.2.3 Le contrôle

BB-1 définit quatre agendas qui sont régulièrement manipulés et mis à jour par le contrôle (cf. figure 4.3) :

1. *Agenda des* ISCs[1] *sélectionnées* qui mémorise toutes les ISCs dont le champ *trigger* de la partie condition de la SC associée est satisfait.

2. *Agenda des* ISCs *exécutables* qui contient toutes les ISCs dont la partie condition (les deux champs *trigger* et précondition) de la SC associée est satisfaite.

3. *Agenda des* ISCs *rejetées* qui stocke toutes les ISCs exécutables ou sélectionnées dont les conditions de rejet de la SC associée sont vérifiées.

4. *Agenda des* ISCs *exécutées* qui est composé de toutes les ISCs exécutées depuis le démarrage du système.

Seuls les deux premiers agendas sont pris en compte par le processus de résolution du plan de contrôle, les deux derniers ne servent que pour le mécanisme explicatif.

La boucle de contrôle de base

Le cycle de contrôle de base dans BB-1 se déroule en trois phases :

1. *Phase d'interprétation.*

   L'interpréteur exécute la partie action de la SC associée à une ISC. Au premier cycle, BB-1 demande à l'utilisateur d'indiquer la source de connaissances dont la partie action doit être activée comme source de connaissances d'initialisation. Durant les autres cycles dans la résolution du

1. Nous rappelons qu'une instance de source de connaissances (ISC) représente une activation potentielle d'une source de connaissances dans le contexte des événements qui l'ont réveillée.

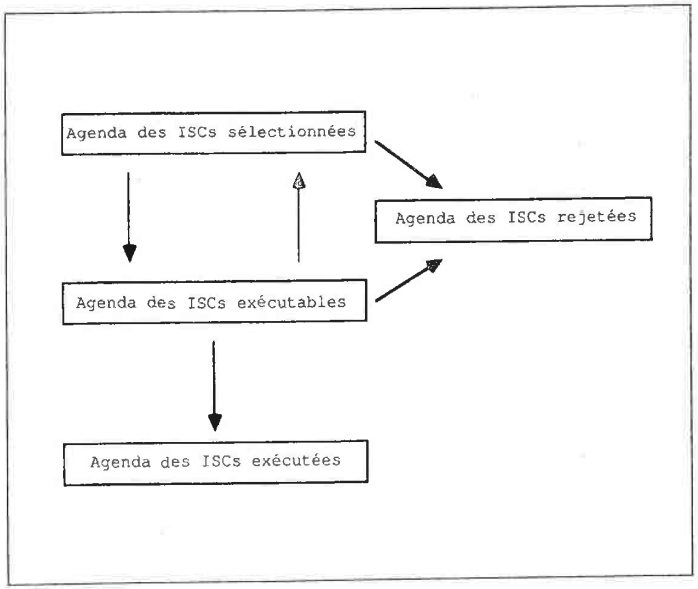


Figure 4.3. Les mouvements des ISCs à travers les agendas.

système, c'est un ordonnanceur qui choisit l'ISC à exécuter dans la phase de sélection (phase 3).

La phase d'interprétation est réalisée en plusieurs étapes :

(a) Vérifier que le champ précondition de la partie condition de la source de connaissances associée à l'ISC est toujours valide;

(b) Vérifier que les conditions de rejet ne sont pas rencontrées;

(c) Créer le contexte de travail de la source de connaissances associée à l'ISC en liant les variables internes à cette dernière;

(d) Exécuter toutes les règles activables de la partie action de la source de connaissances;

(e) Créer les événements engendrés par les actions effectuées sur le blackboard.

2. *Phase de mise à jour des agendas.*

   Cette phase décrit les mouvements des ISCs à travers les quatre agendas définis auparavant :

   (a) Si le cycle courant correspond à un cycle de vérification des préconditions, alors déplacer toutes les ISCs exécutables ayant une précondition non valide vers l'agenda des ISCs sélectionnées;

   (b) Déplacer toutes les ISCs sélectionnées ayant une précondition valide vers l'agenda des ISCs exécutables;

   (c) Pour chaque source de connaissances réveillée par les événements du cycle précédent, créer les ISCs correspondantes et les placer dans l'agenda des ISCs sélectionnées ou dans l'agenda des ISCs exécutables selon la validité de leur précondition;

   (d) Si le cycle courant correspond à un cycle de vérification des conditions de rejet, alors déplacer toutes les ISCs sélectionnées ou exécutables dont les conditions de rejet sont vérifiées vers l'agenda des ISCs rejetées.

3. *Phase de sélection.*

   L'ordonnanceur choisit la prochaine ISC à exécuter parmi toutes celles qui sont présentes dans l'agenda des ISCs exécutables. Pour cela, il doit :

   (a) Evaluer le résultat de l'application de chaque heuristique opérationnelle sur toutes les ISCs exécutables;

   (b) Evaluer le résultat de l'application de chaque objectif opérationnel sur toutes les ISCs exécutables. Pour ce faire, les résultats des heuristiques "implantant" un objectif sont combinés et permettent ainsi de déterminer la valeur attribuée par cet objectif à chaque ISC;

   (c) Déterminer la priorité de chaque ISC en combinant les résultats précédents (les valeurs attribuées à l'ISC par les objectifs opérationnels);

   (d) Choisir l'ISC la plus prioritaire.

Le cycle de contrôle continue jusqu'à ce que les conditions d'arrêt soient rencontrées ou que l'agenda des ISCs exécutables soit vide.

La figure 4.4 représente l'architecture globale de BB-1 utilisé dans PROTEAN [123].

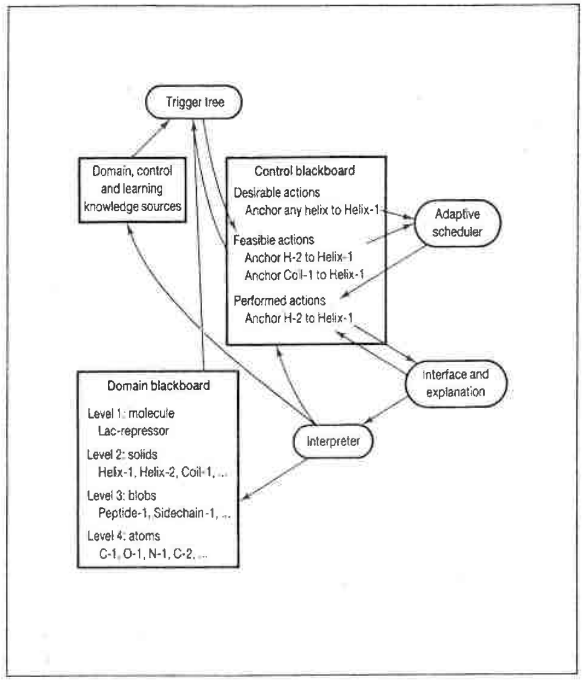


Figure 4.4. Architecture du système BB-1.

4.3 GBB

GBB [45,46,44] est un outil générique de développement de systèmes basés sur le modèle du blackboard. Il a été développé à l'université du Massachusetts (à Amherst). Il est composé de deux sous-systèmes : un outil sophistiqué de spécification du blackboard et de son implantation et un environnement de contrôle.

4.3.1 Spécification et implantation du blackboard

Le but de GBB est de permettre une représentation structurée du blackboard pour laquelle les temps d'accès en lecture ou écriture sont optimisés. A cet effet, il propose un outil de spécification de la structure du blackboard et des objets qui le composent et qui sont nécessaires à une application donnée. Il permet également de définir les stratégies d'implantation à suivre.

Le blackboard dans GBB est composé d'*espaces*[2] ou d'autres blackboards eux-mêmes éventuellement composés d'espaces (cf. figure 4.5). Un espace représente la composante de base du blackboard qui contiendra les objets, appelés *unités*, du système. Chaque espace est structuré en *dimensions* permettant de référencer les objets qu'il contient dans un vocabulaire lié au domaine d'application.

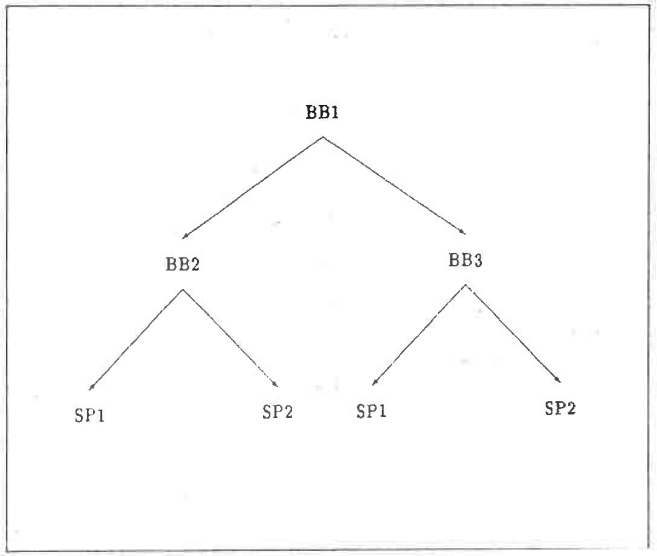


Figure 4.5. Structure d'un blackboard dans GBB [147].

Ainsi, l'accès à une unité dans un espace du blackboard est réalisé à partir

2. Appelés niveaux dans la terminologie habituelle.

des valeurs de ses dimensions. Ces dimensions constituent ainsi un filtre sur l'espace considéré. La spécification des dimensions dans un espace et de leur organisation pour une certaine classe d'unités permet de fournir différents filtres possibles sur l'espace et conduit à différentes implantations des unités. Citons l'exemple du niveau *véhicule* du système DVMT (cf. chapitre 2) caractérisé par les dimensions *temps*, *position-x* et *position-y*. L'accès à un véhicule peut être effectué :

1. en considérant l'instant t où il a été repéré dans le plan, sa position x et sa position y; ou

2. en considérant l'instant t où il a été repéré et sa position (x,y) (les deux dimensions position-x et position-y sont regroupées).

Dans le premier cas, les dimensions du véhicule seront implantées sous forme d'un tableau à trois dimensions, tandis que dans le second cas, elles seront implantées sous forme d'un vecteur pour représenter le temps et d'un tableau à deux dimensions pour la position.

4.3.2 Environnement de contrôle

GBB [147] propose d'une part un outil très simple permettant de tester une SC mais non prévu pour définir une application complète, d'autre part un environnement de contrôle plus élaboré GBB1 [146] qui reprend le modèle défini dans BB1 à savoir un contrôle à base de blackboard. GBB1 corrige certaines faiblesses de BB1 en essayant de réduire le temps dépensé dans les calculs de priorité des ISCs, dans les vérifications de préconditions et conditions de rejet et dans la recherche de création des ISCs (chaque SC est confrontée à chaque événement engendré pour tester si son *trigger* est satisfait). Il offre les possibilités suivantes :

1. Il a repris la notion de *phases* initialement définie dans BBB [18] et ERASMUS[3] [16,63,145], développés par le centre de technologies avancées de Boeing, qui organisent la résolution du problème en une succession de phases durant lesquelles seul un sous-ensemble des SCs du système sont actives.

2. Chaque SC est caractérisée par un champ spécifiant les niveaux du blackboard qu'elle utilise en entrée. Dans BB1, ce champ ne sert qu'à documenter la SC. Dans GBB1, il est utilisé pour filtrer les événements pour

3. BBB et ERASMUS sont des descendants de BB-1.

lesquels le *trigger* de la SC sera testé[4].

3. GBB1 permet de définir des préconditions ou conditions de rejet dites *stables* qui, une fois vérifiées, ne seront plus testées.

4. Il permet également de définir des heuristiques et objectifs du plan de contrôle *stables* vis à vis desquels il n'est pas nécessaire de recalculer le poids d'une SC si ce dernier a déjà été évalué.

Il est à remarquer que la partie action d'une SC dans GBB1 est réduite à une fonction Lisp.

4.4 Avantages du contrôle à base de blackboard

1. Une architecture dont le contrôle est à base de blackboard représente explicitement le problème du domaine en définissant des sources de connaissances qui interagissent à travers une zone de données commune : le blackboard du domaine. Elle représente également le problème du contrôle par des sources de connaissances de contrôle qui construisent, adaptent et maintiennent un plan de contrôle à travers une zone de décisions commune : le blackboard de contrôle. L'ensemble de ces sources de connaissances est géré par un ordonnanceur beaucoup plus simple que celui décrit dans les architectures à contrôle procédural.

   Une distinction entre les concepts liés au domaine et ceux liés au contrôle améliore la clarté, la souplesse, la fiabilité, la robustesse ainsi que la facilité de maintenance du système. Le concepteur peut ajouter, modifier ou supprimer des sources de connaissances de contrôle indépendamment de celles liées au domaine et vice versa.

2. Une architecture dont le contrôle est à base de blackboard intègre la résolution du problème et le contrôle dans une même boucle de base.

   En effet, les sources de connaissances du domaine et les sources de connaissances de contrôle sont manipulées pendant les différentes phases du cycle de base de façon uniforme : durant les deux premières phases (interprétation et mise à jour des agendas), une ISC est exécutée, d'autres sont

4. Toutes les entités manipulées dans GBB sont représentées sous forme d'unités. La structure *événement* est un espace dont les dimensions correspondent aux niveaux du blackboard et dont les éléments sont les événements engendrés par les SCs.

créées et positionnées dans les agendas indépendamment du type de la SC associée. De même, lors de la dernière phase, l'ordonnanceur cherche à sélectionner la meilleure ISC en fonction du plan de contrôle spécifié dans le blackboard de contrôle. Seul, ce plan peut l'influencer dans sa décision. L'ordonnanceur ne peut en effet favoriser *a priori* les actions liées au contrôle par rapport à celles liées au domaine ou *vice-versa*.

Par conséquent, les sources de connaissances de contrôle se trouvent toujours en compétition avec celles du domaine, même si *a priori* et par définition, le rôle du mécanisme de contrôle est avant tout de guider et de gérer la résolution du problème.

3. Une architecture dont le contrôle est fondé sur le blackboard permet une représentation uniforme des connaissances, que celles-ci soient liées au domaine ou au contrôle. En effet, les sources de connaissances respectent la même syntaxe indépendamment de leur type. Ceci fournit au système souplesse, clarté et facilité de programmation.

4.5 Limites du contrôle à base de blackboard

Malgré toutes les originalités que peut amener une telle architecture, certaines limites apparaissent :

1. Complexité de représentation du mécanisme de contrôle devant des situations simples, bien connues ou prédéfinies :

   Quand le flux de contrôle, l'ordre et l'instant d'intervention des sources de connaissances sont bien maîtrisés, par exemple, un groupe de sources de connaissances est toujours exécuté en séquence ou sous forme d'arbre[5], il est assez difficile pour le concepteur du système d'ajuster les champs coût et fiabilité de ces SCs pour représenter un tel fonctionnement du système.

2. Une architecture dont le contrôle est à base de blackboard est coûteuse en temps d'exécution et stockage d'informations. En effet :

   (a) Durant la phase de mise à jour des différents agendas, le système parcourt toutes les sources de connaissances du domaine et de contrôle à la recherche de celles qui ont leur partie condition satisfaite. Il

5. L'ordre d'exécution des sources de connaissances peut changer en fonction de l'état du blackboard par exemple, mais à chaque fois qu'une source de connaissances est activée, les autres le sont aussi.

serait plus intéressant que le système ne parcoure que les SCs susceptibles d'être intéressées par les nouveaux changements apparus dans le blackboard pendant ce cycle. Ainsi, le processus de résolution du système pourrait être vu comme une succession de phases durant lesquelles seul un sous ensemble des SCs est actif.

   (b) De nombreux traitements numériques sont effectués lors de la phase de sélection d'une ISC : un ensemble d'heuristiques est appliqué aux ISCs exécutables, par combinaison des résultats obtenus, le système évalue l'action d'un ensemble d'objectifs sur ces ISCs. Ces résultats sont de nouveau combinés pour calculer la priorité des ISCs, pour enfin sélectionner une seule ISC à exécuter.

   Certaines séquences d'actions à déclencher sont parfois prévisibles et pourraient être exécutées en un seul cycle, ce qui réduirait le nombre de calculs effectués ainsi que la taille des agendas.

4.6 Conclusion

Bien que certaines limites de BB-1 aient été corrigées dans GBB, le contrôle à base de blackboard reste coûteux en temps d'exécution. Ce type de contrôle en contre-partie offre beaucoup de souplesse, de clarté et d'opportunisme pour développer des applications d'IA fondées sur le modèle du blackboard. Ce contrôle est donc conseillé pendant la phase de développement et de conception d'un système, lorsque les stratégies de contrôle ne sont pas figées, sont fragmentées ou en cours de test. En revanche, il est à déconseiller dans des applications où le temps de réponse est un problème primordial.

Une telle architecture sera inefficace dans des applications nécessitant des évolutions trop fréquentes des stratégies de contrôle car de nombreux calculs de priorité et parcours d'agendas sont nécessaires pour maintenir la cohérence du plan de contrôle.

5
Conclusion

La présentation des différents types de contrôle qui sont apparus dans les architectures de blackboard et les remarques que nous avons faites dans chacun des cas (cf. chapitres 2, 3 et 4) mettent en évidence quatre facteurs fondamentaux de discrimination des architectures de blackboard :

1. *Efficacité du système :*

   Un système fondé sur le modèle du blackboard est efficace lorsqu'il est capable de sélectionner rapidement la ou les prochaines actions à exécuter. Par exemple, le contrôle hiérarchique ne nécessite aucun calcul de priorité et peut être considéré comme efficace (les séquences d'actions sont prédéfinies).

2. *Uniformité du système :*

   Un système fondé sur le modèle du blackboard est uniforme lorsqu'il traite le problème de contrôle au même niveau et de la même manière que celui du domaine. Le contrôle à base de blackboard en est un exemple : les SCs de contrôle et du domaine respectent le même formalisme et sont manipulées par le système indépendamment de leur type.

3. *Souplesse du système aussi bien dans sa conception que dans son comportement :*

   Un système fondé sur le modèle du blackboard est souple si, d'une part, c'est un système *ouvert* — de nouvelles composantes peuvent lui être facilement ajoutées — et si, d'autre part, il a un comportement *opportuniste* — les stratégies de contrôle évoluent en fonction de la solution courante. Le contrôle à base de blackboard est souple puisque le plan de contrôle à suivre évolue au fur et à mesure du déroulement du système.

4. *Facilité d'expression :*

   Ce facteur est lié à la clarté de l'architecture globale du système — représentation simple des stratégies et organisation des connaissances. Par

exemple, une architecture basée sur un contrôle hiérarchique fournit une représentation structurée des connaissances et représente un bon modèle de spécification d'un problème.

Définir un compromis entre ces différents facteurs représente le point essentiel, le plus sensible et le plus fastidieux à traiter dans de telles architectures. En effet, un gain en efficacité dans un système se traduit souvent par une perte en souplesse. Ce compromis est d'autant plus complexe à définir que le type de contrôle choisi dans ces architectures est fixé pour toute la résolution du problème (cf. § 2.4 de la partie I).

Il apparaît également que toutes les SCs liées au domaine d'application sont décrites dans un même formalisme. En effet, elles sont soit toutes procédurales, soit toutes à base de règles. Or, dans des domaines d'applications faisant appel à des techniques de reconnaissance des formes par exemple, les connaissances du domaine sont souvent hétérogènes. Il est alors important de pouvoir mêler connaissances procédurales et connaissances déclaratives.

Le but de nos travaux a été de corriger ces limites. Pour ce faire, nous avons défini une architecture appelée *hybride à multi-phases* proposant un contrôle approprié à chaque phase de la résolution du problème et intégrant différents formalismes de représentation des connaissances du domaine (programme C, base de règles respectant la logique des propositions, prédicats du premier ordre ou d'ordre 0+).

Dans la partie suivante, nous présentons cette approche et son implantation dans ATOME[1], environnement de développement de systèmes multi-experts basé sur le modèle du blackboard, que nous avons développé sur stations de travail SUN en Le_Lisp [32] et sur SYMBOLICS en Common Lisp [149].

1. Another TOol for developing Multi-Expert systems.

PARTIE III

Le projet ATOME

Dans cette partie, nous présentons notre apport dans le domaine des architectures de blackboard. Le chapitre 1 décrit les objectifs que nous nous sommes fixés. Dans le chapitre 2, nous présentons les premières réalisations que nous avons effectuées, puis nous donnons une évaluation de l'outil ATOME qui en résulte. Dans le chapitre 3, nous décrivons les extensions et améliorations effectuées sur cette version d'ATOME. Le chapitre 4 fait une synthèse des différents types d'objets qui interviennent dans ATOME ainsi que les opérations qui les manipulent. Nous donnons un exemple complet d'utilisation d'ATOME dans le chapitre 5. Les chapitres 6 et 7 décrivent des extensions concernant la convivialité d'utilisation d'ATOME et l'enrichissement de son formalisme de représentation des connaissances. Le chapitre 8 dresse un bilan complet sur ATOME, ses extensions futures et son utilisation.

1
Introduction

Dans les parties précédentes, nous avons mis en évidence que le mécanisme de contrôle constitue le point clé du modèle du blackboard. Nous avons également montré que la puissance d'une architecture face aux différents facteurs cités au chapitre 5 de la partie II dépend du choix du type de contrôle adopté. Dans le cadre de la mise en œuvre d'un outil d'aide au développement de systèmes à multi-sources de connaissances, nos travaux de recherche ont porté sur l'élaboration d'un mécanisme de contrôle puissant offrant au système une architecture à la fois souple, paramétrable, efficace et d'expression facile. L'outil réalisé doit ainsi constituer un environnement de développement visant différentes classes d'application (interprétation et compréhension des formes, planification, interprétation de signaux complexes, contrôle de processus, aide à la décision...) et doit permettre l'intégration de connaissances de natures diverses.

La démarche que nous avons suivie est composée de quatre étapes :

1. La première étape a consisté à développer un noyau ATOME[1] en favorisant d'abord les deux facteurs *efficacité* et *facilité d'expression*. Il nous a semblé important en effet que notre outil permette une formalisation structurée du problème à résoudre et offre un niveau d'efficacité intéressant. Pour ce faire, nous avons adopté une structure de contrôle dérivée du modèle hiérarchique à deux niveaux (cf. chapitres 3 de la partie II et 2 de cette partie).

2. La seconde étape a consisté à étendre l'architecture précédente vers une architecture *souple* et *paramétrable* en ajoutant un mode *opportuniste* dans le comportement du système tout en rendant plus indépendantes

1. Nous avions dans un premier temps baptisé cet outil OMSC (Outil d'aide au développement de systèmes à Multi-Sources de Connaissances [162]). Mais ce nom était peu agréable à prononcer et à entendre, aussi nous avons choisi de le renommer dès les premières réalisations. Nous avons trouvé le nom ATOME (Another TOol for developing Multi-Expert systems) qui nous a *charmés* et que nous avons adopté.

les différentes composantes du contrôle. Nous avons désigné par contrôle *hybride à multi-phases* le type de contrôle résultant de cette architecture.

3. La troisième étape a été d'introduire un *mécanisme explicatif* offrant à tout utilisateur d'ATOME des facilités de mise au point d'une application.

4. La dernière étape a consisté à intégrer dans ATOME *différents formalismes de représentation des connaissances*, permettant ainsi de développer les SCs liées au domaine d'application sous forme :

   (a) *procédurale :* programmes C ou Lisp;

   (b) *déclarative :*

      i. bases de règles respectant le formalisme ATOME (ordre 0+),

      ii. systèmes experts ou à base de connaissances développés à l'aide des systèmes génériques SAGANE (ordre 0) [11] et IROISE (ordre 1) [107]. Ces outils ont été sélectionnés d'une part, parce qu'ils complétaient le formalisme proposé par ATOME et d'autre part, parce qu'une expérience acquise sur ces systèmes a facilité considérablement cette intégration [186].

Nous décrivons dans cette partie les différentes étapes citées précédemment. Pour chaque étape, nous précisons les objectifs visés et les choix effectués pour les atteindre. Nous montrons également comment une étape corrige les limites rencontrées lors des étapes précédentes. Nous présentons le modèle *hybride à multi-phases* qui en résulte et que nous illustrons par l'exemple simple du voyageur de commerce.

Un bilan complet d'ATOME est donné à la fin de cette partie. Dans ce bilan, nous montrons comment ce dernier intègre efficacité, souplesse, modularité et généricité; nous comparons le modèle du contrôle à base de blackboard implanté dans BB-1 et le modèle hybride à multi-phases implanté dans ATOME; nous citons les extensions envisagées sur ATOME à court, moyen et long terme.

Nous présentons également les différents travaux au CRIN concernant l'intégration des *raisonnements approximatif, temporel et hypothétique* dans ATOME; et nous terminons ce bilan par une description des applications en cours de développement avec ATOME.

Une conclusion d'ordre général sur les architectures de blackboard est donnée à la fin de cette thèse.

2
Efficacité et facilité d'expression dans ATOME

L'architecture adoptée pour développer un noyau ATOME intégrant à la fois efficacité et facilité d'expression est fondée sur un modèle de *contrôle hiérarchique.* Nous avons choisi un modèle à deux niveaux de contrôle intégrant de plus deux modes de raisonnement (dirigé par les événements et dirigé par les règles). Dans ce chapitre, nous présentons les trois composantes de base d'une architecture de blackboard dans le cadre de cette approche. Nous mettons ensuite en évidence les avantages et les limites de cette architecture, qui nous conduiront vers la seconde étape de nos travaux décrite dans le chapitre suivant.

Dans toute la suite de cette thèse, nous désignons par *système*-ATOME tout système d'IA développé avec ATOME.

2.1 Approche générale

ATOME définit trois types de sources de connaissances : les *spécialistes* (SPs) qui regroupent les connaissances liées au sujet traité, les *tâches* (TAs) qui fournissent un contrôle *local* en gérant les spécialistes et la *stratégie* (ST) qui propose un contrôle *global* du système en gérant les tâches.

Les spécialistes sont les seules SCs qui accèdent au *blackboard.* Les tâches disposent d'une structure de contrôle globale appelée *liste d'événements.* La stratégie, quant à elle, accède à une seconde structure de contrôle appelée *résumé du blackboard* (cf. figure 2.1).

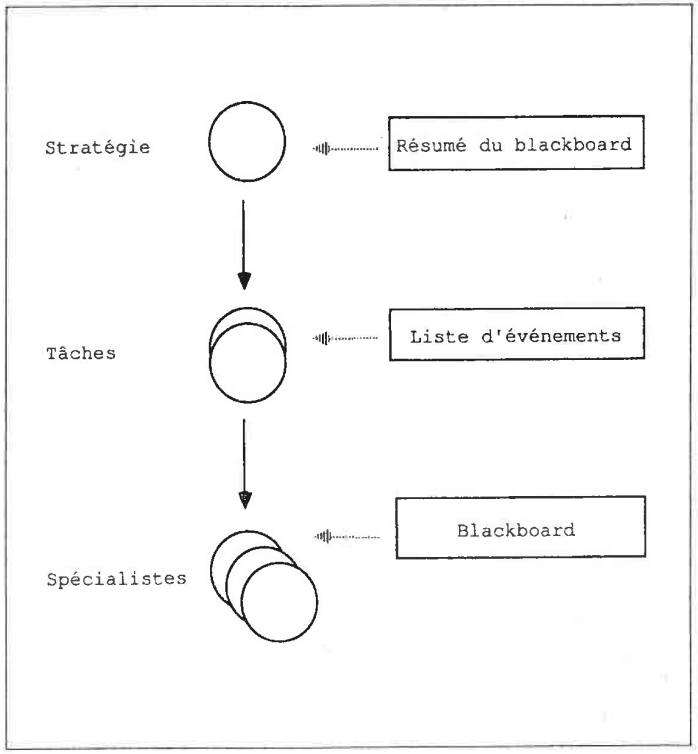


Figure 2.1. Approche globale d'ATOME.

La suite de ce chapitre est consacrée à une description détaillée de ces trois types de SCs et des structures qu'elles manipulent.

2.2 Structure du blackboard

Un niveau (d'abstraction) dans le blackboard est représenté sous forme d'objet structuré, caractérisée par un certain nombre d'attributs. Une hypothèse dans le blackboard correspond à une instance d'un de ces objets. Chaque attribut d'une instance peut avoir plusieurs valeurs alternatives, provenant d'une ou plusieurs sources d'informations et affectées d'un coefficient de vraisemblance numérique ou égal à *indéfini*[1]. Les relations entre les hypothèses d'un même niveau ou de niveaux différents du blackboard sont également représentées par des attributs : un attribut de type *lien* d'une hypothèse est constitué de la liste des hypothèses qui sont en relation avec elle. Tout lien doit obligatoirement posséder un lien inverse de manière à maintenir la symétrie des relations entre les hypothèses.

On appellera *action* dans le blackboard toute mise à jour du contenu du blackboard :

- *création* d'une hypothèse dans un niveau du blackboard;
- *modification* d'attributs ou de liens d'une hypothèse du blackboard avec conservation de leurs anciennes valeurs;
- modification d'attributs ou de liens d'une hypothèse du blackboard en écrasant leurs anciennes valeurs. Ce type d'action est appelé *remplacement*;
- *suppression* d'une hypothèse du blackboard.

De par la possibilité de supprimer des hypothèses ou des valeurs d'attributs (ou de liens) d'hypothèses dans le blackboard, le système risque de se retrouver dans un état antérieur et par conséquent de boucler. De plus, ces suppressions peuvent nécessiter la remise en cause d'hypothèses engendrées à partir des éléments détruits. Aussi il est préférable d'utiliser cette possibilité avec prudence.

1. Actuellement ces coefficients ne sont pas combinés dans ATOME, mais ils peuvent être manipulés explicitement par l'utilisateur. Les difficultés liées à l'aspect raisonnement approximatif dans un univers multi-bases de connaissances et les études en cours au CRIN sur ce sujet sont introduites au chapitre 8.8.2 de cette partie.

Prenons comme exemple la représentation de véhicules stationnés dans des parkings. Supposons qu'un véhicule est caractérisé par son type (camion, bus, car, voiture, moto ou bicyclette), sa marque (Renault dans le cas d'une voiture...), sa couleur et son immatriculation. Supposons également qu'un parking est défini par son nom, son lieu et sa capacité. On créera alors les deux niveaux *véhicule* et *parking* (cf. figure 2.2) composés des attributs correspondant aux caractéristiques citées précédemment.

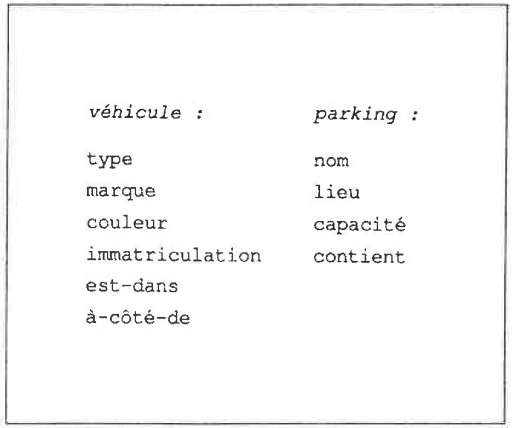


Figure 2.2. Définition des niveaux du blackboard.

Certains attributs supplémentaires représentent les liens entre les instances (hypothèses) de ces niveaux. L'attribut *contient* du niveau *parking*, par exemple, représente un lien (une relation) entre les hypothèses du niveau *parking* et celles du niveau *véhicule*. L'attribut *est-dans* du niveau *véhicule* est le lien inverse du précédent. Ces deux relations traduisent le fait que des véhicules peuvent être stationnés dans un parking. L'attribut *à-côté-de* du niveau *véhicule* définit un lien entre deux hypothèses de ce niveau permettant ainsi de représenter des véhicules voisins.

Parking-1, *véhicule-1* et *véhicule-2* représentés dans la figure 2.3 sont des hypothèses des niveaux *parking* et *véhicule*. Pour faciliter la compréhension de cette figure, nous avons simplifié l'écriture des valeurs des attributs et des liens. Le blackboard aura alors la structure donnée dans la figure 2.4.

Cet exemple indique que les véhicules stationnés dans le parking S^{t} Sébastien de Nancy sont une Renault Supercinq et une Visa Citroën. La première a une couleur tendant vers le rouge et est immatriculée 3423 TW 54 tandis que la seconde est bleue et est immatriculée 1342 SH 54. De plus, ces deux voitures sont voisines dans le parking.

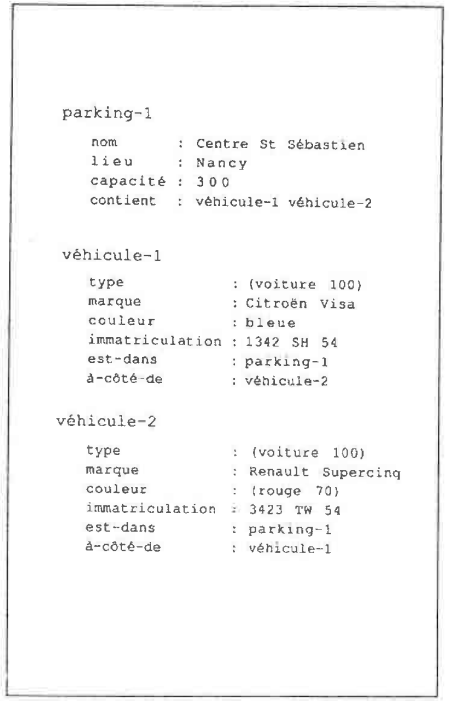


Figure 2.3. Hypothèses du blackboard.

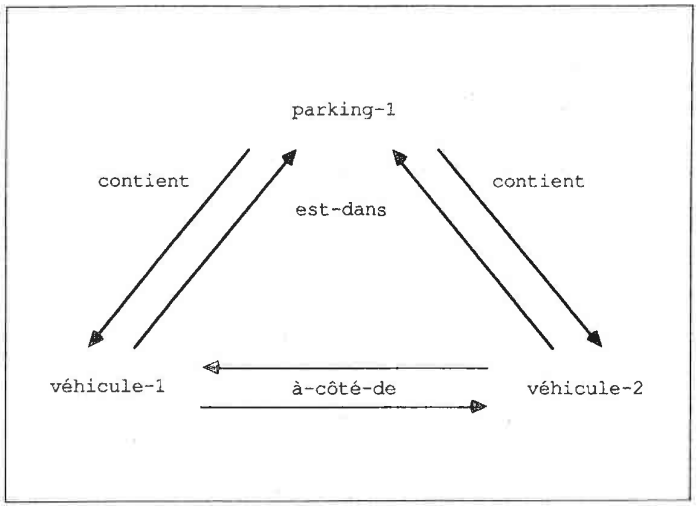


Figure 2.4. Structure du blackboard.

2.3 Evénements

Durant cette étape, seules les actions relatives à la création, à la modification ou au remplacement d'une hypothèse dans le blackboard engendrent des événements[2]. Ces derniers sont créés systématiquement dès que l'action est effectuée. Ils sont immédiatement placés dans la liste des événements à une position fonction de leur importance (cf. § 2.5). L'importance d'un événement est fournie par la spécialiste (source de connaissances) ayant provoqué l'action et est exprimée sous une forme d'importance relative. La spécialiste peut en effet préciser que l'action effectuée est plus importante que, par exemple, toute action concernant un certain niveau du blackboard. Elle peut également demander explicitement à ce que l'événement soit placé en tête — événement le plus important — ou en queue — événement le moins important — de la liste.

2. Un événement associé à la suppression d'une hypothèse ferait référence à une donnée qui n'existe plus dans le blackboard.

Un type d'objet *événement* a été défini, chaque événement créé étant une instance de ce type et étant caractérisé principalement par :

- Le *niveau* du blackboard dans lequel le changement a été effectué.
- Le *nœud* du blackboard (nom de l'hypothèse) sur lequel porte le changement.
- Le *type* de l'action effectuée (création, modification ou remplacement).

  La liste des *attributs-mis-a-jour* et des *liens-mis-a-jour* du nœud qui ont effectivement été créés ou modifiés.

Reprenons l'exemple des véhicules stationnés dans un parking. La figure 2.5 signale que la couleur du *véhicule-2* a changé.

```
Evénement-5
    Niveau                : véhicule
    Noeud                 : véhicule-2
    Type                  : remplacement
    Attributs-mis-à-jour  : (couleur)
    Liens-mis-à-jour      : ()
```

Figure 2.5. Structure d'un événement.

Nous verrons en 2.5 comment et à quel niveau ces événements influencent le mécanisme de contrôle.

2.4 Les sources de connaissances

Une source de connaissances est un module indépendant expert dans un sous-domaine du domaine de départ. Nous l'appellerons par la suite *spécialiste*. Son rôle est d'essayer de contribuer à la résolution du problème à partir des informations disponibles dans le blackboard et des connaissances qu'elle

renferme. Sa participation se traduira par une mise à jour du blackboard. Ce comportement est illustré par la figure 2.6.

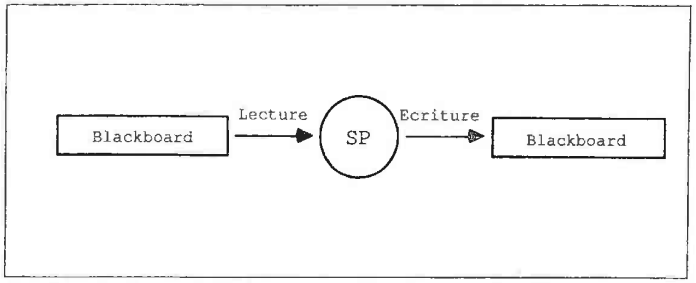


Figure 2.6. E/S d'une spécialiste.

De par son expertise, une spécialiste est capable de spécifier à l'avance le type de données du blackboard qui l'intéressent. Il lui est en effet possible de définir un filtre sur le blackboard ou sur une partie du blackboard[3] qui lui permettra, à chaque activation, de déterminer son contexte de travail (sous-ensemble de nœuds du blackboard). Ainsi, au lieu de raisonner sur tout le blackboard, la spécialiste oriente ses activités vers la région de données qu'elle doit traiter.

De plus, dans cette première étape, nous avions choisi de "remonter" la partie condition[4] d'une spécialiste au niveau du mécanisme de contrôle pour deux raisons :

1. Eviter le parcours de toutes les spécialistes pour retrouver celles qui ont cette partie condition satisfaite.
2. Supprimer les problèmes de conflits qui se posent pour connaître l'ordre dans lequel les spécialistes activables seraient à exécuter.

Le corps de la spécialiste est alors réduit à sa seule partie action. Cette dernière est composée d'une base de règles de production respectant le formalisme ATOME.

3. La spécialiste peut en effet limiter le filtrage sur les hypothèses du blackboard qui ont provoqué son activation et qui lui sont de ce fait passées en *focalisation d'attention* par le mécanisme de contrôle.

4. Cf. chapitre 1.1 de la partie I.

Nous caractériserons une spécialiste par les propriétés citées dans la figure 2.7. Le comportement lié à son activation varie en fonction de son mode de fonctionnement et est décrit dans la figure 2.8.

1. *Nom :* identification de la spécialiste.
2. *Variables locales :* leur principal rôle est de permettre la spécification du filtre de la spécialiste. Différentes fonctions de filtrage sont associées à chaque variable. Ces variables sont liées lors de l'activation de la spécialiste et sont manipulées dans sa base de règles (ordre 0+).
3. *Base de règles :* corps de la spécialiste.
4. *Mode de fonctionnement :* cette propriété permet de préciser le mode d'interprétation de la base de règles — simple, multiple ou cyclique.

Figure 2.7. Caractéristiques d'une spécialiste.

1. La spécialiste crée son contexte de travail.

   Les variables locales de la spécialiste sont liées aux hypothèses filtrées sur le blackboard et constituent alors son contexte de travail.

2. Cas où son mode de fonctionnement est :

   Simple : Activer la règle déclenchable la plus importante.

   Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

   Multiple : Activer toutes les règles déclenchables en un seul passage.

   Le système n'effectue qu'un seul parcours de la base de règles.

   Cyclique : Activer toutes les règles déclenchables jusqu'à saturation.

   Le système boucle sur le mode multiple.

Figure 2.8. Activation d'une spécialiste.

2.5 Le mécanisme de contrôle

Les connaissances de contrôle sont représentées sous forme de *sources de connaissances de contrôle* organisées en deux niveaux hiérarchiques :

- Le niveau le plus bas est composé de méta-sources de connaissances — *les tâches* — qui contiennent des informations sur les capacités des SCs spécialistes à résoudre un sous-but particulier. Au cours de la résolution du problème, ce sont ces méta-sources qui décideront quelles spécialistes seront à activer et dans quel ordre.
- Au niveau le plus haut, il y a une seule source de connaissances — *la stratégie* — qui analyse la qualité de la solution courante pour déterminer vers quelle région de données le système doit se focaliser, quelles tâches sont à activer et dans quel ordre.

Avant de définir plus précisément ces SCs de contrôle, nous présentons les structures de contrôle utilisées dans ATOME. Nous ferons ensuite une synthèse sur les différentes activités des sources de connaissances du système et sur l'architecture résultante, puis nous décrirons sa boucle de base.

2.5.1 Structures de contrôle

On distingue deux structures de contrôle, la *liste des événements* consultée par les tâches et le *résumé du blackboard* analysé par la stratégie.

La liste des événements :

Elle contient tous les changements du blackboard qui n'ont pas encore participé à la résolution du problème. Nous rappelons que, dans ce chapitre, seules les actions liées à une création, une modification ou un remplacement d'une hypothèse dans le blackboard engendrent des événements. Ces derniers sont immédiatement positionnés dans la liste des événements en fonction de leur importance (cf. 2.3). La figure 2.9 illustre ce mécanisme. Cette liste est consultée par les tâches pour rechercher la présence d'événements qui ont eu lieu et qui peuvent provoquer l'activation de sources de connaissances spécialistes.

Le résumé du blackboard :

Il mémorise tous les nœuds du blackboard considérés comme importants ou nécessaires. Au lieu de parcourir tout le blackboard, la stratégie consultera cette structure de contrôle; ceci afin de gagner en temps

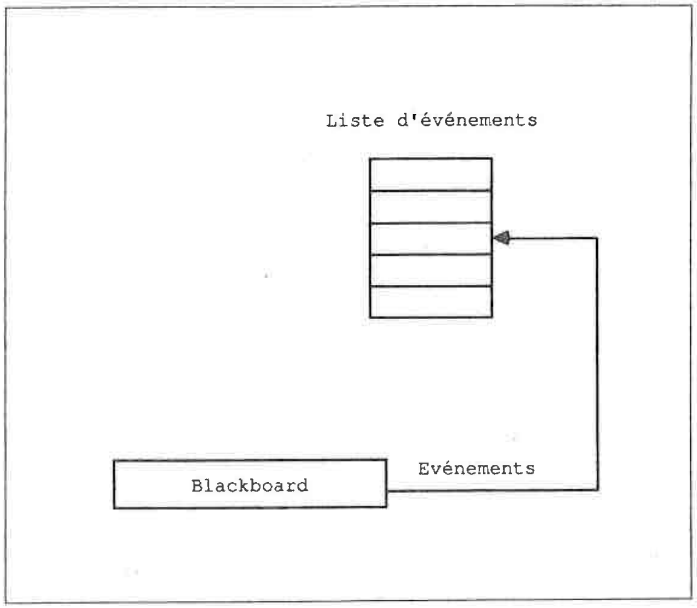


Figure 2.9. Liste des événements.

d'exécution. Le résumé du blackboard lui fournira ainsi une vue d'ensemble, globale de la solution courante qui sera déterminée en ne filtrant que les hypothèses pertinentes du blackboard, capables d'influencer le comportement du système.

Le filtre qui permet de construire ce résumé relève de l'expertise de l'application en cours de développement avec ATOME. Il consiste à spécifier *a priori* sous une forme déclarative les conditions sous lesquelles les hypothèses d'un certain niveau du blackboard peuvent être placées dans le résumé. Un niveau du blackboard peut ne pas apparaître dans son résumé si aucune condition d'importance ne lui est associée.

2.5.2 Les tâches

Une tâche regroupe et coordonne les activités d'un sous-ensemble de spécialistes du système, ceci afin de résoudre un sous-problème du problème de départ. Des spécialistes peuvent être placées sous la responsabilité d'une même tâche pour deux raisons :

1. soit parce qu'elles sont *compétitives* : elles sont souvent en conflit. C'est le cas par exemple où elles essaient de travailler sur les mêmes données.
2. soit parce qu'elles sont *coopératives* : elles sont souvent amenées à travailler ensemble. C'est le cas par exemple où les résultats de l'une servent de données pour l'autre.

Une même spécialiste peut intervenir dans plusieurs tâches.

Une tâche active des spécialistes en s'appuyant sur les changements du blackboard qui n'ont pas encore participé à la résolution du problème (liste des événements), sur les hypothèses qui ont permis son activation, et sur les connaissances qu'elle renferme et qui sont exprimées sous forme de règles de production.

Un exemple type de règle d'une tâche est donné dans la figure 2.10.
La partie gauche de la règle teste la présence d'événements dans la liste des

```
SI
    tel(s) type(s) d'événement(s) est(sont) arrivé(s)
ALORS
    SC4 SC1 SC2
```

où SC1 SC2 et SC4 sont des sources de connaissances spécialistes.

Figure 2.10. Règle type d'une tâche.

événements et l'état de variables locales à la tâche ou globales. Sa partie droite contient une ou plusieurs spécialistes à exécuter séquentiellement lorsque la règle est déclenchée. En l'occurrence, la satisfaction de la partie gauche de la règle de la figure 2.10 provoque l'activation des spécialistes SC4, suivie de SC1, suivie de SC2. Le comportement global de la tâche consiste donc essentiellement à lire des informations dans la liste des événements et à activer en conséquence des sources de connaissances spécialistes comme l'indique la figure 2.11.

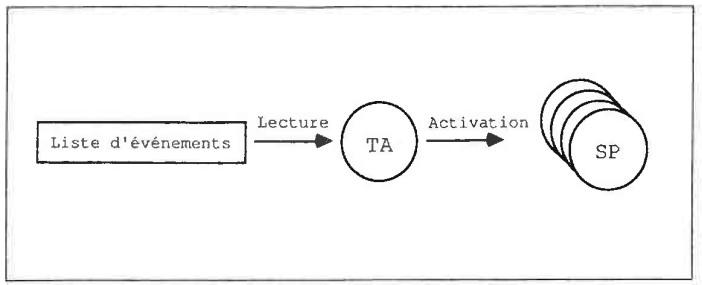


Figure 2.11. E/S d'une tâche.

Il existe deux modes de raisonnement au niveau des tâches :

Mode dirigé par les événements :

La tâche choisit l'événement en tête de la liste des événements - l'événement le plus important - et recherche une ou plusieurs règles déclenchables avec cet événement. A chaque déclenchement de règles, des spécialistes sont activées, des actions sont effectuées dans le blackboard et, de ce fait, de nouveaux événements sont engendrés. Les plus importants seront placés en tête de la liste des événements et seront donc traités en premier lieu. Ainsi, une tâche dirigée par les événements *réagit* à chaque changement important dans le blackboard (cf. figure 2.12). La ou les spécialistes activées pourront disposer, si elles le désirent, de l'hypothèse ayant permis leur activation (l'hypothèse associée à l'événement en cours de traitement). Celle-ci lui sera fournie par la tâche sous forme de *focalisation de contrôle*.

Mode dirigé par les règles :

La tâche cherche à activer sa règle la plus importante[5]. A cette fin, elle consultera la liste des événements pour retrouver ceux qui vérifient la partie gauche de la règle. De même que précédemment, chaque déclenchement de règles entraîne l'activation de spécialistes et la mise à jour de la liste des événements. Une tâche dirigée par les règles traite les événements par groupes : ceux qui satisfont la partie gauche d'une même règle

5. Comme pour les spécialistes, chaque règle est affectée d'un coefficient d'importance.

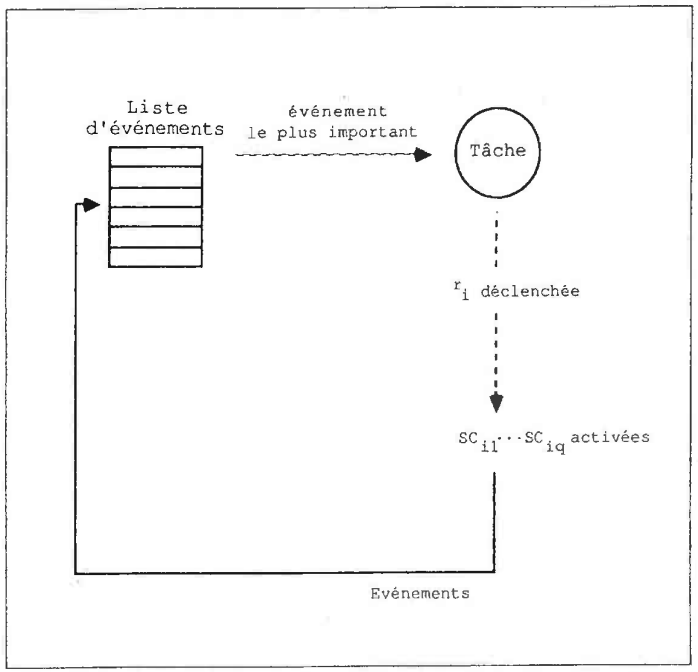


Figure 2.12. Cycle de base du mode dirigé par les événements.

sont traités ensemble, indépendamment de leur importance (cf. figure 2.13). La ou les spécialistes activées peuvent disposer, si elles le désirent, des hypothèses ayant permis leur activation. Celles-ci leur seront fournies par la tâche sous forme de *focalisation de contrôle*.

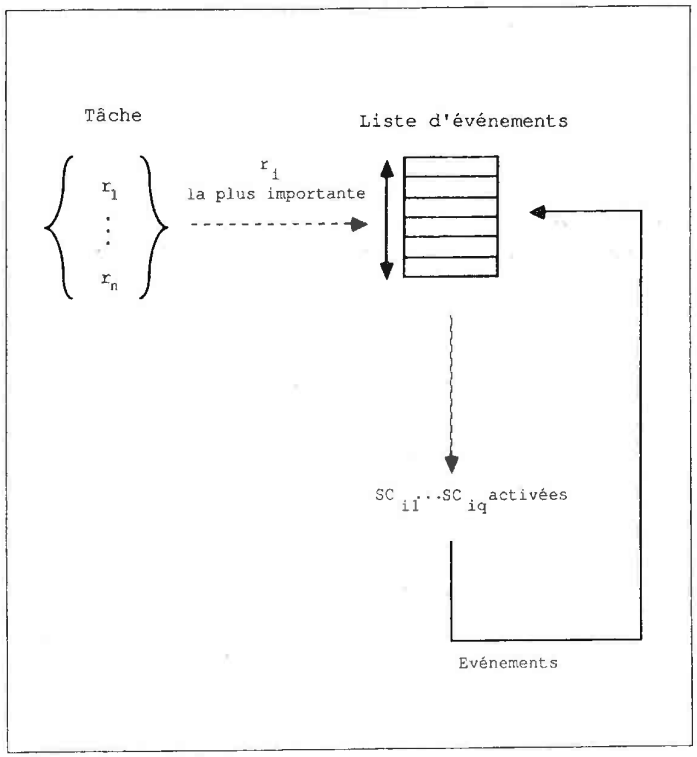


Figure 2.13. Cycle de base du mode dirigé par les règles.

La figure 2.14 résume l'ensemble des propriétés caractérisant une tâche. Le comportement lié à l'activation d'une tâche varie, d'une part, selon son *mode de raisonnement* — *dirigé par les événements* ou *dirigé par les règles* — et, d'autre part, selon son *mode de fonctionnement*. Ce dernier est lui-même dépendant du mode de raisonnement de la tâche. Ce comportement est décrit dans les figures 2.15 et 2.16.

1. *Nom :* identification de la tâche.
2. *Variables locales :* variables locales de la tâche. Ces variables n'ont pas le même rôle que pour les spécialistes. Elles permettent de mémoriser une valeur ou un calcul intermédiaire mais ne constituent pas le contexte de travail principal de la tâche. Ce dernier est caractérisé par l'état de la liste des événements au moment de l'activation de la tâche.

   Ces variables sont visibles par les spécialistes qui sont sous la responsabilité de cette tâche.
3. *Base de règles :* corps de la tâche.
4. *Mode de raisonnement :* indique si la tâche est dirigée par les événements ou par les règles.
5. *Mode de fonctionnement :* varie en fonction du mode de raisonnement de la tâche.

   Tâche dirigée par les événements :

   Le mode de fonctionnement dans ce cas peut être simple, multiple, cyclique-simple ou cyclique-multiple.

   Tâche dirigée par les règles :

   Dans ce cas, le mode de fonctionnement de la tâche peut être simple, multiple ou cyclique.

Figure 2.14. Caractéristiques d'une tâche.

La figure 2.15 montre que les événements ayant permis l'activation de règles d'une tâche dirigée par les événements sont retirés de la liste des événements au fur et à mesure de leur utilisation. Lors de l'activation de cette tâche, le rôle du système est de réagir aux actions les plus importantes dans le blackboard et de les traiter rapidement. Ainsi, lorsque ces actions ont été prises en compte, elles peuvent être retirées de la liste des événements.

1. La tâche sélectionne l'événement le plus important.

   Il s'agit du premier événement de la liste des événements non encore traité par la tâche.
2. Si aucune règle n'est activable avec cet événement, le *marquer* comme "traité" et retourner en 1.,

   Sinon, cas où le mode de fonctionnement de la tâche est :

   Simple :

   (a) Activer la première règle déclenchable avec cet événement.
   Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

   (b) Le *retirer* de la liste des événements.

   Multiple :

   (a) Activer toutes les règles déclenchables en un seul passage avec cet événement.
   Le système n'effectue qu'un seul parcours de la base de règles.

   (b) Le *retirer* de la liste des événements.

   Cyclique-simple :

   (a) Activer la première règle déclenchable avec cet événement.

   (b) Le *retirer* de la liste des événements.

   (c) Retourner en 1.

   Boucle sur le mode simple pour chaque événement de la liste des événements.

   Cyclique-multiple :

   (a) Activer toutes les règles déclenchables en un seul passage avec cet événement.

   (b) Le *retirer* de la liste des événements.

   (c) Retourner en 1.

   Boucle sur le mode multiple pour chaque événement de la liste des événements.

Figure 2.15. Activation d'une tâche dirigée par les événements.

Cas où son mode de fonctionnement est :

Simple : Activer la règle déclenchable la plus importante.

Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

Multiple : Activer toutes les règles déclenchables en un seul passage.

Le système n'effectue qu'un seul parcours de la base de règles.

Cyclique : Activer toutes les règles déclenchables jusqu'à saturation.

Le système boucle sur le mode multiple.

Figure 2.16. Activation d'une tâche dirigée par les règles ou de la stratégie.

En revanche, lors de l'activation d'une tâche dirigée par les règles, le rôle du système est d'essayer d'activer des règles importantes de la tâche en recherchant les événements appropriés. Il n'y a aucune raison qui justifierait leur suppression de la liste des événements à la fin de l'activation de la tâche. Ce ménage doit cependant être fait, d'une part pour ne pas laisser le système s'attarder sur d'anciennes actions et négliger les plus récentes, et, d'autre part pour limiter la taille de la liste des événements. Aussi avons nous choisi de retirer ces événements de la liste à la fin du *cycle* au cours duquel la tâche a été activée. La notion de cycle sera définie en 2.5.3. Nous pouvons cependant préciser que, durant un cycle, plusieurs tâches peuvent être activées séquentiellement.

2.5.3 La stratégie

Le rôle de la stratégie est de superviser le déroulement du processus de résolution du problème posé. En fonction de l'avancement du système, elle choisit les points particuliers du problème à traiter. Pour ce faire, elle s'appuie sur une vue globale de la solution courante et coordonne les activités des tâches. Cette vue globale lui est fournie par le résumé du blackboard.

Les connaissances de la stratégie sont exprimées sous forme de règles de production dont un exemple type est donné dans la figure 2.17. La partie gauche de la règle teste la présence d'hypothèses dans le résumé du blackboard et l'état de variables locales à la stratégie ou globales. La partie droite contient une ou plusieurs tâches à exécuter séquentiellement lorsque la règle est déclenchée. En l'occurrence, la satisfaction de la partie gauche de la règle de la figure 2.17 provoque l'activation des tâches TA5, suivie de TA1, suivie de TA2. L'activation

d'une règle de la stratégie représente un *cycle* du fonctionnement du système. La stratégie fournit aux tâches les hypothèses ayant permis leur activation en tant que *focalisation de contrôle* pour leur permettre éventuellement d'orienter leurs travaux vers des événements portant sur ces hypothèses par exemple.

Globalement, le comportement de la stratégie consiste donc essentiellement à lire des informations dans le résumé du blackboard et à activer en conséquence des tâches. La figure 2.18 illustre ce comportement.

```
SI
    tel(s) type(s) d'hypothèse(s) existe(ent) dans le résumé du blackboard
ALORS
    TA5 TA1 TA2

où TA1 TA2 et TA5 sont des tâches.
```

Figure 2.17. Règle type de la stratégie.

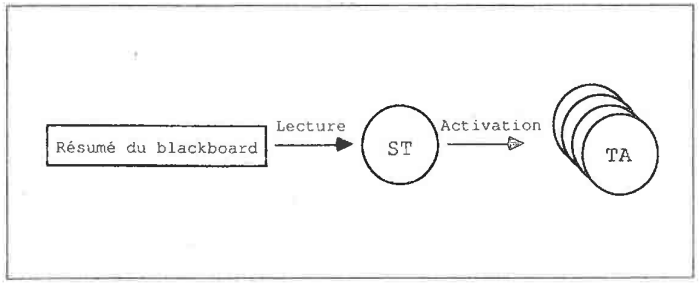


Figure 2.18. E/S de la stratégie.

La figure 2.19 résume l'ensemble des propriétés caractérisant la stratégie. Le comportement lié à son activation varie en fonction de son mode de fonctionnement. Il est similaire à celui d'une tâche dirigée par les règles (cf. figure 2.16).

1. *Nom* : identification de la stratégie.
2. *Variables locales :* variables locales de la stratégie. Ces variables ont le même rôle que celles des tâches, le contexte de travail de la stratégie étant le résumé du blackboard.

   Elles représentent en fait les variables globales du système et sont visibles par toutes les tâches et spécialistes.
3. *Base de règles :* corps de la stratégie.
4. *Mode de fonctionnement :* il peut être simple, multiple ou cyclique.

Figure 2.19. Caractéristiques de la stratégie.

2.5.4 Architecture d'ATOME

En résumé, les activités des différentes sources de connaissances d'ATOME peuvent être exprimées sous la forme suivante :

- Les SCs *spécialistes*

  ont accès au blackboard (données initiales et hypothèses),

  peuvent

  créer de nouvelles hypothèses dans le blackboard,
  changer des valeurs d'attributs d'hypothèses,
  changer des liens entre hypothèses,
  supprimer des hypothèses,

  et *envoient* les résultats de leurs actions dans la liste des événements.

- Les *tâches*

  ont accès aux événements de la liste des événements,

  et *peuvent* appeler les SCs spécialistes appropriées pour résoudre un point particulier du problème.

- La *stratégie*

  a accès au résumé du blackboard,

  et *peut* activer une ou plusieurs tâches.

La figure 2.20 décrit l'architecture globale d'ATOME qui en résulte. Nous définissons dans la suite la boucle de contrôle de base d'un système développé avec ATOME.

2.5.5 Boucle de contrôle de base

Cette boucle de base montre le fonctionnement global d'un système-ATOME[6] pour lequel toutes les composantes de base ont été définies selon les diverses expertises liées au domaine d'application. Elle constitue un cycle d'exécution du système :

1. La stratégie choisit la règle la plus importante ayant sa partie gauche satisfaite et la déclenche.

   L'activation de cette règle provoque l'exécution d'une séquence de tâches, éventuellement de fonctions Lisp et/ou l'arrêt du système.

   Les tâches activées disposeront des nœuds sélectionnés par la stratégie (*première focalisation de contrôle*).

2. La tâche courante (selon son mode de raisonnement) provoque l'activation de spécialistes, éventuellement l'exécution de fonctions Lisp et/ou l'interruption de cette tâche.

   Elle peut se focaliser sur les événements associés aux nœuds sélectionnés par la stratégie.

   Lorsqu'elle a fini de travailler, la tâche redonne le contrôle à la stratégie.

   Les spécialistes activées disposeront des événements sélectionnés par cette tâche et de ce fait, des nœuds associés (*seconde focalisation de contrôle*).

3. La spécialiste courante est activée. Elle peut créer, modifier ou supprimer des hypothèses dans le blackboard, exécuter des programmes Lisp et/ou demander de s'arrêter.

   Elle peut se focaliser sur les nœuds passés en *première* et/ou *seconde focalisation de contrôle*.

   Lorsqu'elle a fini de travailler, la spécialiste redonne le contrôle à la tâche appelante.

4. Fin du cycle.

6. Système développé avec ATOME.

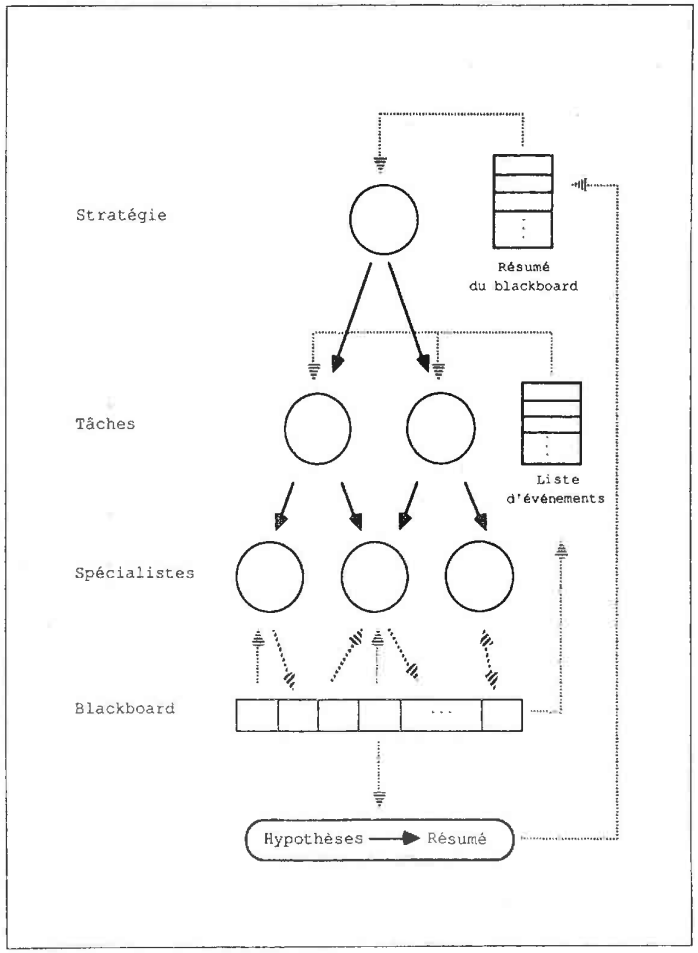


Figure 2.20. Architecture d'ATOME.

Le système s'arrête lorsqu'aucune règle de la stratégie n'est déclenchable ou si une demande explicite a été effectuée par la stratégie. La figure 2.21 illustre cette boucle de contrôle de base.

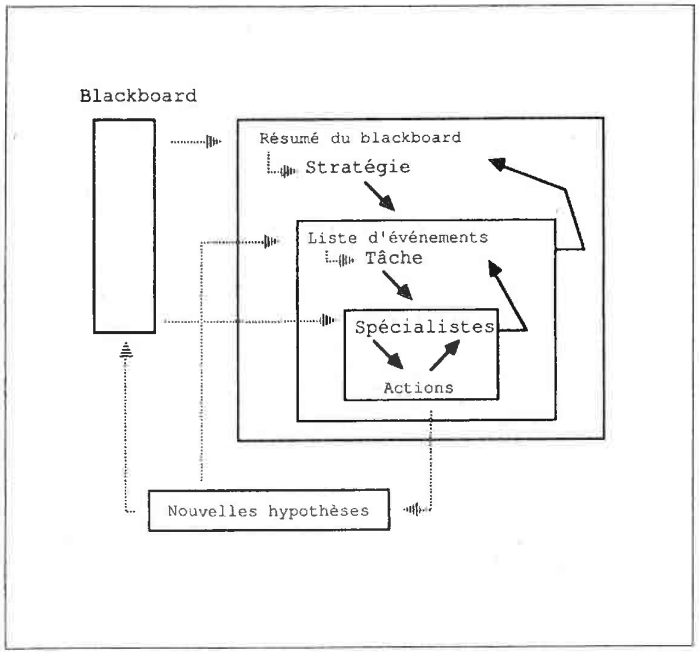


Figure 2.21. Boucle de contrôle de base.

2.6 Evaluation de la première étape

Notre but étant de définir une architecture efficace et facile d'expression en vue de l'étendre ensuite vers une structure plus souple, un contrôle hiérarchique nous a paru être un bon point de départ. D'après notre étude bibliographique et d'après la manière dont les différents systèmes fondés sur le modèle

du blackboard ont abordé le mécanisme de contrôle, une structure à deux niveaux de sources de connaissances de contrôle nous a paru suffisante. Pour des raisons d'efficacité, nous avons dû définir deux structures de contrôle : la liste d'événements et le résumé du blackboard. Sans ces dernières, les SCs de contrôle (la stratégie et les tâches) devraient accéder directement au blackboard pour recueillir les informations leur permettant d'organiser leurs activités. Etant donné la quantité d'hypothèses mémorisées dans le blackboard, et le fait que ces hypothèses sont souvent incomplètes et incertaines, le temps nécessaire pour retrouver les plus pertinentes deviendrait trop important. De plus, la stratégie et les tâches auraient une vue trop "détaillée" de la solution courante et n'auraient pas suffisamment de recul vis à vis de l'avancement du système pour le faire évoluer correctement. Il est donc indispensable de fournir à chacune d'elles une vue globale de ce qu'il y a dans le blackboard d'une part, et de ce qui s'est passé dans ce dernier d'autre part. Le résumé du blackboard fournit ainsi à la stratégie les éléments à prendre en considération dans le blackboard tandis que la liste des événements indique aux tâches les différents mouvements qui ont eu lieu dans le blackboard et qui n'ont pas encore été traités.

Envisager plus de deux niveaux pour le mécanisme de contrôle demanderait de définir des structures de contrôle supplémentaires dont les éléments fournissent une autre vision de la solution courante. Nous pensons (de plus) que le développement d'une application basée sur le modèle du blackboard est un problème en soi; décomposer le contrôle en deux niveaux de SCs permet de structurer et d'organiser les connaissances mais en définir davantage risque d'en augmenter la complexité.

Voyons les avantages et les limites que présente l'architecture d'ATOME à la fin de cette première étape.

2.6.1 Les avantages

Les avantages sont principalement liés aux buts que nous nous étions fixés, à savoir la *facilité d'expression* et l'*efficacité*. Nous ajouterons à ces deux points un facteur supplémentaire qui concerne la souplesse dans la *conception* de l'outil développé.

1. *Facilité d'expression :*

   ATOME reflète en quelque sorte la manière dont un groupe d'experts aborde un problème. Le responsable, schématisé par la stratégie, supervise le déroulement du processus de résolution du problème : il consulte régulièrement l'état d'avancement des travaux en ne s'appuyant que sur des informations synthétisées et schématisées par le résumé du blackboard. En fonction de cet état, le responsable choisit les points du problème vers

   lesquels il faut se focaliser et demande aux responsables qui organisent et contrôlent ces points particuliers de les résoudre. Ces responsables sont schématisés par les tâches. Chacun d'eux a sa propre équipe d'experts pour résoudre sa partie du problème. Ces experts sont schématisés par les SCs spécialistes.

   ATOME permet donc une représentation explicite et modulaire des connaissances de contrôle et une formalisation structurée du problème.

2. *Efficacité :*

   L'efficacité du système est liée à la hiérarchie entre les SCs de contrôle et les SCs du domaine, conduisant à une boucle de base qui ne se limite pas à l'activation d'une seule source de connaissances comme dans les architectures à contrôle procédural ou à base de blackboard (cf. chapitres 2 et 4 de la partie II). A chaque cycle, la stratégie active une séquence de tâches et chaque tâche active plusieurs séquences de spécialistes. La focalisation de contrôle joue également un rôle prépondérant dans l'efficacité car elle permet de focaliser les activités des SCs vers une région de données particulière. Les structures de contrôle permettent également au mécanisme de contrôle d'agir rapidement en lui fournissant une vue globale à la fois sur le contenu du blackboard et sur les changements qui apparaissent dans ce dernier et qui n'ont pas encore participé à la résolution du problème.

3. *Souplesse dans la conception*

   ATOME est un système ouvert pour lequel il est aisé d'ajouter ou de supprimer des composantes. Nous avons profité de cette souplesse pour étendre les fonctionnalités d'ATOME dans les étapes ultérieures.

2.6.2 Les limites

Les limites de l'architecture obtenue portent essentiellement sur les quatre points suivants : manque de souplesse dans le *comportement* du système, absence de préconditions dans les spécialistes, mise à jour de la liste des événements et création d'événements, ainsi que la représentation des données dans un seul blackboard.

1. *Souplesse dans le comportement*

   La limite la plus importante de cette architecture est le manque de souplesse dans le comportement du système. Les SCs de contrôle exigent une spécification précise et fine de l'expertise dans la résolution de conflits d'intervention des différentes spécialistes. En effet, l'ordre d'exécution

des spécialistes doit être partiellement connu en fonction de leurs conditions d'activation (cf. figures 2.10). Or, dans de nombreuses applications, cette expertise ne peut être définie que pour certaines phases de la résolution du problème. HEARSAY-II en est exemple type : la première phase consiste à activer séquentiellement quatre spécialistes émettant successivement des hypothèses sur les niveaux *signal paramétré*, *segment*, *syllabe* et *mot*; la seconde phase consiste ensuite à sélectionner de façon *opportuniste* les spécialistes exécutables (cf. chapitre 2 de la partie II).

2. *Préconditions d'une spécialiste*

   La seconde limite porte sur l'absence de préconditions dans la définition des spécialistes. Nous avons en effet précisé que la partie condition d'une spécialiste est "remontée" au niveau des tâches. Les prémisses de règles des tâches spécifient les conditions sous lesquelles une séquence de spécialistes *pourra* travailler. Dans la réalité, rien ne permet d'affirmer que chaque spécialiste effectuera réellement un travail. En effet, ces conditions portent sur des événements, c'est-à-dire sur ce qui s'est passé dans le blackboard et qui n'a pas encore été traité, mais ne tiennent pas compte des hypothèses associées à ces événements dans leur détail. En fait, ces conditions permettent de définir une séquence de spécialistes *susceptibles* de travailler.

   Ces conditions ne peuvent constituer à part entière la partie condition d'une spécialiste telle qu'elle a été définie dans le chapitre 2 de la partie I[7].

   Elles représentent en quelque sorte un *trigger* un peu plus élaboré que celui défini dans BB1[8] puisqu'elles permettent de préciser non seulement les spécialistes auxquels le système doit s'intéresser mais aussi l'ordre dans lequel elles doivent être exécutées. Nous pensons qu'il est nécessaire de restaurer une partie précondition associée à chaque spécialiste afin que cette dernière puisse juger elle-même si elle est effectivement capable de travailler lorsqu'elle est sollicitée.

3. *Evénements et liste des événements*

7. Nous rappelons que la partie condition d'une SC décrit les situations dans lesquelles la SC peut contribuer à l'avancement de la résolution du problème. Généralement, cette partie requiert une configuration particulière des hypothèses dans le blackboard.

8. Nous avons vu dans la description de BB1 (cf. chapitre 4 de la partie II) que la partie condition d'une spécialiste est décomposée en deux champs : le *trigger* spécifiant quand le système doit s'intéresser à cette SC et la *précondition* qui décrit les conditions supplémentaires que la SC doit vérifier pour effectivement travailler.

Le troisième problème que nous ne devons pas négliger fait référence aux événements et à la mise à jour de la liste des événements. Plusieurs faiblesses apparaissent dans ce contexte.

(a) Comme nous l'avons précisé dans ce chapitre, il est nécessaire de faire le ménage dans la liste des événements. Ainsi, les événements ayant été utilisés par des tâches sont retirés de la liste soit à la fin de l'activation de la tâche concernée, soit à la fin du cycle durant lequel cette dernière a été activée.

Or, cet événement aurait pu intéresser d'autres tâches à des cycles ultérieurs. Peut-être était-il prématuré de le supprimer ? Quels seraient les critères qui permettraient de savoir si un événement particulier doit être supprimé ou non ? Le cas échéant, quand faut-il le retirer de la liste d'événements ?...

(b) Lorsque deux événements se rapportent au même type de changement d'un même attribut d'une même hypothèse du blackboard par exemple, le système actuel manipule ces deux événements indépendamment de leur lien. Le second événement pourrait cependant remettre en cause le précédent, ou pourrait conduire à une même action.

(c) Il faut étendre la notion d'événement aux suppressions d'hypothèses dans le blackboard, car ces dernières peuvent influencer les stratégies à suivre dans le comportement du système.

(d) Chaque action relative à une création, une modification ou un remplacement d'une hypothèse dans le blackboard engendre un événement. Pourtant certaines actions peuvent n'être d'aucun intérêt pour le mécanisme de contrôle.

4. *Représentation des données*

   Le dernier point que nous citerons en tant que limite est la structure du blackboard lui-même. Nous nous sommes en effet limités à un seul blackboard. Pour organiser davantage les données du système, il nous paraît intéressant de permettre l'utilisation de plusieurs blackboards.

2.7 Conclusion

Nous avons présenté dans ce chapitre la première étape de nos travaux sur le mécanisme de contrôle dans les architecture de blackboard. Dans cette optique, nous avons développé un premier noyau d'ATOME, outil d'aide au développement de système d'IA basé sur le modèle du blackboard. L'architecture

que nous avons définie est fondée sur un contrôle hiérarchique et présente un certain nombre d'avantages que nous voulons conserver et de limites que nous voulons corriger. Le chapitre suivant présente les extensions et corrections effectuées sur ATOME dans ce sens.

Cette étape a abouti à la réalisation de la version 1.0 d'ATOME [188] opérationnelle depuis février 1987. Cette version a été développée en Le_Lisp version 15.2 sur station de travail SUN et est opérationnelle sur toutes les machines où Le_Lisp version 15.2 est disponible.

Cette étape a fait également l'objet des publications [117,167,193] et des rapports internes [188,190].

3
Vers une architecture souple et paramétrable

Ce chapitre décrit la seconde étape de nos travaux durant laquelle nous avons défini et mis en œuvre un mécanisme de contrôle de type *hybride à multi-phases*. Pour ce faire, nous nous sommes basés sur les résultats obtenus à l'étape précédente (cf. chapitre 2) et sur les buts que nous nous étions fixés, à savoir : obtenir une architecture à la fois souple, paramétrable, efficace et d'expression facile.

Dans ce chapitre, nous dressons d'abord une liste des principaux changements mis en œuvre dans ATOME. Ensuite, pour chaque composante d'ATOME, nous présentons et justifions les diverses extensions et corrections effectuées. Nous définissons ensuite le modèle hybride à multi-phases qui en résulte. Nous terminons par une conclusion sur l'architecture obtenue et les différents buts atteints.

3.1 Approche générale

Nous avons étendu ATOME en :

1. permettant l'utilisation de *plusieurs* blackboards (cf. § 3.2 et § 3.5.2);
2. permettant aux spécialistes de *gérer* elles-mêmes la création des événements associés à leurs actions (cf. § 3.3);
3. autorisant la création d'événements associés aux *suppressions* d'hypothèses (cf. § 3.3);
4. associant une partie *précondition* aux spécialistes (cf. § 3.4 et § 3.6);
5. divisant la liste d'événements en plusieurs listes *locales* aux tâches (cf. § 3.5.1 et § 3.6);

6. *synthétisant* les événements qui signalent des actions similaires (cf. § 3.5.1 et § 3.6);

7. ajoutant un mode de raisonnement *opportuniste* au niveau des tâches (cf. § 3.6.3 et § 3.6.4);

8. définissant la notion de blackboard *à long terme* (cf. § 3.8).

3.2 Les blackboards

Certaines applications nécessitent l'utilisation de plusieurs blackboards pour structurer davantage les données et les hypothèses du système. Le système VISIONS [111], par exemple, définit une *mémoire à court terme*, blackboard contenant l'état de l'interprétation d'une image spécifique et une *mémoire à long terme*, blackboard contenant les connaissances *a priori* d'un modèle général donné. Le système CRYSALIS (cf. chapitre 3 de la partie II) divise le blackboard en deux plans hiérarchiques séparant ainsi données obtenues après prétraitement et hypothèses sur l'interprétation en cours. Le système DVMT, décrit au chapitre 2 de la partie II, définit également plusieurs blackboards : le *blackboard des données* mémorisant les hypothèses et les données du système, et le *blackboard des buts* contenant les buts que ce dernier cherche à résoudre. Dans le cadre de ces exemples, il est à remarquer que les différents blackboards utilisés dans un même système contiennent des éléments de type *hypothèse*, *but* ou *donnée*, deux blackboards différents contenant des entités de types différents.

Mais l'utilisation de plusieurs blackboards ne se limite pas à ce cas. On peut en effet envisager qu'un sous-ensemble des sources de connaissances du domaine d'un système souhaitent placer leurs résultats partiels dans une zone de données privées (blackboard local accessible à toutes les SCs de cet ensemble) et ne communiquer aux autres SCs du système que les hypothèses nécessaires (par l'intermédiaire d'un blackboard global accessible à toutes les SCs du système). Le blackboard local contiendra alors toutes les hypothèses intermédiaires qui ont permis d'émettre les hypothèses du blackboard global. Il constitue en fait une zone de communication privilégiée entre des spécialistes. La figure 3.1 met en évidence ce type de comportement.

Il nous est donc indispensable, pour satisfaire ces besoins, de permettre l'utilisation de plusieurs blackboards dans le développement d'une application avec ATOME. La définition des niveaux et de leurs éléments reste presque similaire à ce qui a été défini dans le chapitre précédent. Il suffit, en fait, de préciser pour chaque niveau ou hypothèse le nom du blackboard auquel il (elle) appartient. Une instance d'un niveau pourra représenter différents concepts dans les blackboards. Des liens pourront bien sûr être définis entre

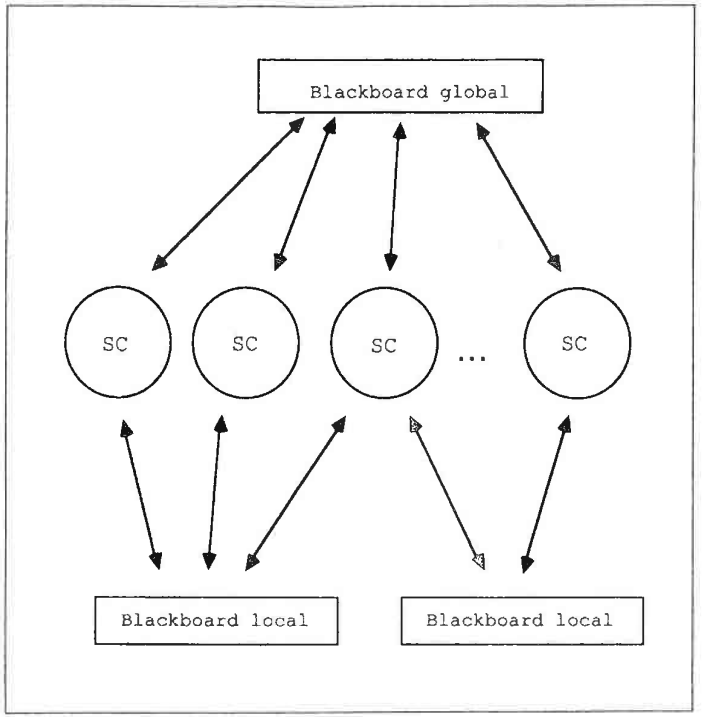


Figure 3.1. Utilisation de blackboards locaux et globaux.

des éléments d'un même blackboard ou de blackboards différents.

3.3 Les événements

Les événements représentent des messages envoyés au mécanisme de contrôle par les spécialistes pour le prévenir de leurs actions dans les blackboards. Comme nous l'avons présenté dans la première version (cf. chapitre 2), toute action relative à la création, à la modification ou au remplacement d'une hypothèse dans le blackboard engendre systématiquement un événement qui est placé dans la liste des événements. Or certaines de ces actions peuvent représenter des étapes intermédiaires (des calculs par exemple) nécessaires à la spécialiste en cours d'activation pour émettre des hypothèses plus significatives vis à vis de la résolution de problème. Ces actions peuvent de ce fait n'intéresser en rien le mécanisme de contrôle, en l'occurrence les tâches. Elles peuvent même les distraire en engendrant des événements inutiles. Il est donc nécessaire de gérer explicitement la création des événements en caractérisant les actions devant les engendrer.

Pour résoudre ce problème, deux approches nous semblent possibles :

1. *Caractériser ces actions lors de la définition du ou des blackboards du système-*ATOME *en cours de développement.*

   Cette approche consiste à associer à chaque attribut, niveau ou blackboard du système, les types de mises à jour qui doivent conduire à la création d'événements. Par exemple, *toute action relative à la création d'une hypothèse dans un blackboard particulier à un niveau spécifique doit engendrer un événement.*

   Cette approche est quelque peu rigide. En effet, la création des événements est systématique dès qu'une action porte sur l'attribut, le niveau ou le blackboard en question (cf. figure 3.2).

2. *Permettre à chaque spécialiste du système-*ATOME *de gérer ses propres actions.*

   De par son expertise ou ses connaissances et par sa vision de la solution courante (vision locale sur les blackboards), chaque spécialiste effectue des actions dans les blackboards. Dans ce contexte, elle est la seule capable de juger l'importance de ses propres actions et donc de décider s'il est nécessaire de prévenir le mécanisme de contrôle en engendrant des événements. Cette approche est plus souple que la précédente. Un même type d'action dans un blackboard peut en effet engendrer ou non un événement selon la spécialiste à l'origine de cette action et selon le contexte ayant conduit à cette action (cf. figure 3.3).

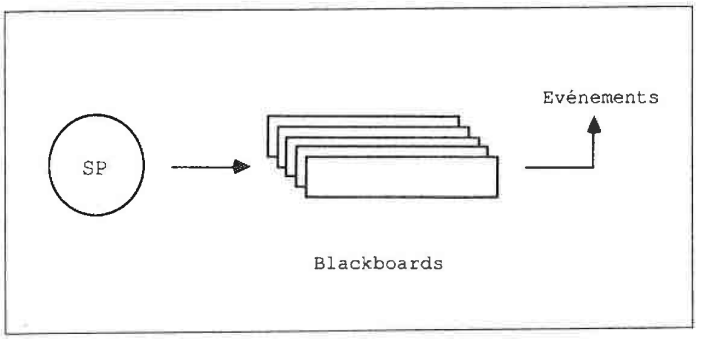


Figure 3.2. Création des événements systématique pour un attribut, un niveau ou un blackboard donné.

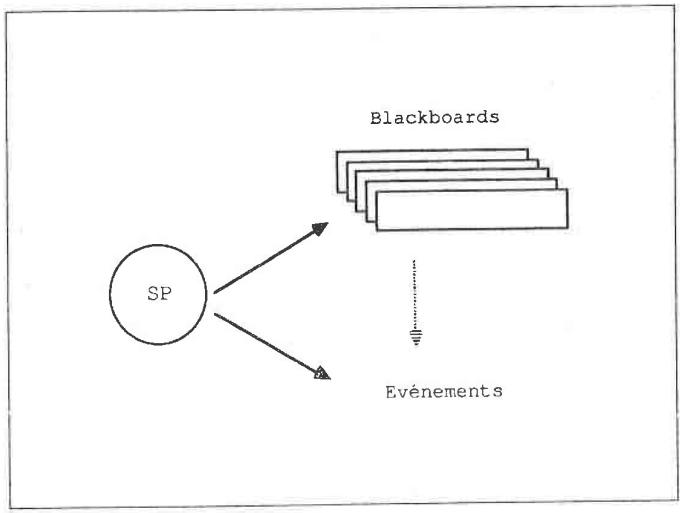


Figure 3.3. Création des événements à la demande des spécialistes.

La seconde approche fournit aux spécialistes une certaine autonomie en leur permettant de gérer elles-mêmes la création des événements associés à leurs propres actions. Pour ces raisons, nous avons choisi cette méthode pour la gestion des événements dans un système-ATOME.

Dans la version précédente d'ATOME, une spécialiste associait à chacune de ses actions une importance relative. Celle-ci permettait au système de calculer la position à laquelle l'événement devait être placé dans la liste des événements. Maintenant, lorsque cette importance n'est pas précisée par la spécialiste, l'événement correspondant à l'action n'est pas engendré.

D'autre part, nous avons étendu la notion d'événement à la suppression d'hypothèses dans le(s) blackboard(s). En effet, certaines de ces suppressions peuvent influencer les décisions du mécanisme de contrôle dans sa conduite du système et doivent de ce fait lui être signalées. Le problème lié à la création d'événements pour des suppressions d'hypothèses concerne la focalisation de contrôle. En effet, lorsqu'un événement quelconque permet l'activation d'une spécialiste, l'hypothèse associée à cet événement est passée en *seconde focalisation de contrôle* à la spécialiste (cf. § 2.5.5). Or, lorqu'une suppression d'hypothèse est demandée dans un blackboard, l'hypothèse n'est pas seulement retirée du blackboard, mais elle est aussitôt détruite. Nous avons donc été contraints de changer ce principe. Lorsque la suppression d'une hypothèse est demandée, l'hypothèse est immédiatement retirée du blackboard; tous les événements associés à cette hypothèse sont également retirés de la liste d'événements; l'hypothèse n'est détruite physiquement que si sa suppression n'engendre aucun événement ou lorsque l'événement engendré est traité.

Nous avons vu au chapitre 2 de cette partie la structure d'un événement. Cette structure n'a pas changé, pour cette seconde étape de notre thèse, même si dans le cas d'une suppression d'hypothèses, seuls les champs *niveau*, *nœud* et *type* de cet événement auront une valeur :

- *Niveau :* contiendra le niveau préfixé par le nom du blackboard auquel l'hypothèse supprimée appartenait.
- *Nœud :* contiendra le nom de l'hypothèse supprimée.
- *Type :* suppression.

3.4 Les sources de connaissances

Comme nous l'avons précisé au chapitre précédent en citant les limites de l'architecture d'ATOME issue de la première étape, il est nécessaire de restituer la partie précondition associée à chaque spécialiste. En effet, lorsque le

mécanisme de contrôle sélectionne une ou plusieurs spécialistes afin de les activer, il leur signale en fait que le contexte actuel leur est favorable et peut les intéresser[1]. C'est ensuite aux spécialistes de juger et de vérifier que toutes les conditions sont requises pour leur permettre effectivement de travailler.

Ces conditions sont exprimées dans la partie *précondition* de chaque spécialiste sous forme de tests sur l'état des blackboards ou d'une partie des blackboards. Si ces conditions sont vérifiées, nous dirons que la spécialiste passe de l'état *pré-sélectionnée* à l'état *activable*. Nous distinguons la *phase de vérification de la précondition* d'une spécialiste de la *phase d'activation* de cette même spécialiste. Toute phase d'activation d'une spécialiste est précédée d'une phase de vérification de sa précondition (cf. figure 3.4).

En revanche, le mécanisme de contrôle peut demander l'exécution de la phase de vérification de la précondition d'une spécialiste indépendamment de sa phase d'activation. Cet enchaînement entre les deux phases est lié au mode de raisonnement de la tâche ayant sollicité la spécialiste. Nous reviendrons sur cet aspect par la suite (cf. § 3.6).

Nous avons vu au chapitre précédent que la phase d'activation d'une spécialiste est composée de deux étapes : création de son contexte de travail puis exécution du corps de la spécialiste. De façon similaire, la phase de vérification de la précondition d'une spécialiste sera effectuée en deux étapes : création d'un *contexte préliminaire* obtenu par filtrage sur les blackboards ou sur une partie des blackboards[2] puis test de la validité de la précondition. Ce contexte préliminaire représente un sous-ensemble du contexte de travail de la spécialiste. Il contient la vue partielle de l'état des blackboards nécessaire à la spécialiste pour vérifier sa précondition.

Les caractéristiques d'une spécialiste telles qu'elles ont été décrites au chapitre précédent ont évolué comme l'indique la figure 3.5[3]. La phase de vérification de la précondition d'une spécialiste est décrite par la figure 3.6. La phase d'activation d'une spécialiste, quant à elle, a peu changé par rapport à l'étape précédente : la création du contexte de travail doit prendre en considération le fait qu'un contexte préliminaire a déjà été évalué lors de la phase de vérification de la précondition de la spécialiste (cf. figure 3.7).

1. Le mécanisme de contrôle "réveille" les spécialistes susceptibles d'être intéressées par les derniers changements des blackboards.
2. La spécialiste peut en effet limiter le filtrage sur les hypothèses des blackboards qui ont permis sa pré-sélection : ces hypothèses lui sont passées en *focalisation d'attention* par le mécanisme de contrôle.
3. Nous serons amenés à étendre de nouveau les propriétés d'une spécialiste afin de permettre l'intégration d'un mode opportuniste dans ATOME (cf. § 3.6.3).

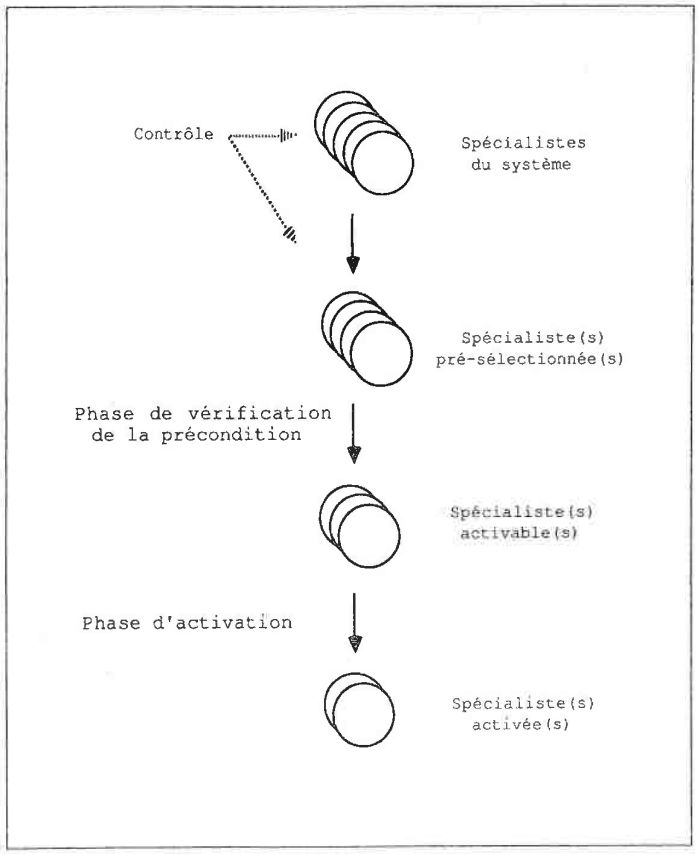


Figure 3.4. Phases de fonctionnement des spécialistes.

1. *Nom :* identification de la spécialiste.
2. *Variables locales :* leur principal rôle est de permettre la spécification des filtres de la spécialiste. Différentes fonctions de filtrage sont associées à chaque variable. Cet ensemble de variables est divisé en deux parties :
   (a) *Variables de précondition :*
   Ces variables sont liées lors de la phase de vérification de la précondition de la spécialiste et restent accessibles lors de sa phase d'activation. Elles sont donc manipulées dans sa précondition ainsi que dans sa base de règles.
   (b) *Variables d'action :*
   Ces variables sont liées lors de la phase d'activation de la spécialiste et sont manipulées dans sa base de règles (ordre 0+).
3. *Précondition :* liste de conditions à vérifier par la spécialiste pour devenir *activable.*
4. *Base de règles :* corps de la spécialiste.
5. *Mode de fonctionnement :* cette propriété permet de préciser le mode d'interprétation de la base de règles - simple, multiple ou cyclique.

Figure 3.5. Caractéristiques d'une spécialiste.

1. La spécialiste crée son contexte préliminaire.
   Les variables de précondition de la spécialiste sont liées aux hypothèses filtrées sur les blackboards et constituent alors son contexte préliminaire.
2. Validité de la précondition :
   Vérifie si les conditions spécifiées dans la précondition de la spécialiste sont satisfaites.

Figure 3.6. Phase de vérification de la précondition d'une spécialiste.

1. La spécialiste crée son contexte de travail.

   Les variables d'action de la spécialiste sont liées aux hypothèses filtrées sur les blackboards. Les variables de préconditions ont déjà été évaluées auparavant et restent accessibles durant cette phase. L'ensemble des variables locales ainsi liées constituent le contexte de travail de la spécialiste.

2. Cas où son mode de fonctionnement est :

   Simple : Activer la règle déclenchable la plus importante.
   Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

   Multiple : Activer toutes les règles déclenchables en un seul passage.
   Le système n'effectue qu'un seul parcours de la base de règles.

   Cyclique : Activer toutes les règles déclenchables jusqu'à saturation.
   Le système boucle sur le mode multiple.

Figure 3.7. Phase d'activation d'une spécialiste.

3.5 Structures de contrôle

Les structures de contrôle utilisées dans ATOME sont composées de *listes d'événements* et de *résumés des blackboards.*

3.5.1 Les listes d'événements

Durant la première étape, l'utilisation d'une seule liste d'événements commune à toutes les tâches a mis en évidence les problèmes suivants :

- *Présence d'événements perturbateurs*

  Tous les événements qui n'ont pas encore participé à la résolution du problème sont mémorisés dans la liste des événements. Toutes les tâches accèdent à cette liste pour avoir une vue globale sur ce qui s'est passé dans le blackboard[4] et sur ce qui n'a pas encore été traité. A partir de ces informations, des spécialistes seront activées.

  Or, certains événements de la liste des événements peuvent n'avoir aucun intérêt pour certaines tâches et en revanche être pertinents pour d'autres.

4. Nous rappelons que durant cette première étape, ATOME permettait de ne définir qu'un seul blackboard. Les problèmes que nous soulevons dans ce paragraphe restent inchangés lors de l'utilisation de plusieurs blackboards.

  Considérons par exemple le domaine de la parole. Une tâche travaillant au niveau du signal ne sera pas intéressée par les changements se rapportant au niveau phrase. Pour retrouver les événements liés au niveau signal, elle sera amenée à parcourir la liste des événements et sera donc perturbée et distraite par tous ceux qui ne concernent pas son niveau d'intérêt.

- *Suppression intempestive des événements de la liste des événements*

  Les événements ayant été utilisés par une tâche sont retirés de la liste des événements soit à la fin de l'activation de la tâche concernée (cas où la tâche est dirigée par les événements (cf. chapitre 2)), soit à la fin du cycle durant lequel cette dernière a été activée (cas où la tâche est dirigée par les règles (cf. chapitre 2)).

  Or, certains des événements supprimés auraient pu intéresser d'autres tâches et conduire à des actions intéressantes. Ainsi, la suppression d'événements de la liste des événements engendre une perte d'informations parfois pénalisante. De par cette suppression, les tâches deviennent compétitives vis à vis de l'accès aux événements dans la liste des événements. Ce type de conflit ne peut être résolu par la stratégie, dont le rôle est d'activer les tâches en fonction de l'avancement de la résolution du problème : elle n'a pas les connaissances requises pour résoudre ce conflit.

  Ce problème représente en fait le dual du précédent. En effet, dans le premier cas, l'information est surabondante par rapport aux besoins des tâches tandis que, dans le second cas, elle est manquante.

La figure 3.8 illustre l'accès des tâches à la liste des événements.

Afin de résoudre les problèmes décrits précédemment, nous avons introduit la notion de *liste des événements locale à une tâche.* Chaque tâche d'un système-ATOME[5] dispose de sa propre liste d'événements. Elle peut spécifier le type d'informations dont elle a besoin et qui doivent lui être signalées par l'intermédiaire de sa liste d'événements locale. Elle effectue ainsi un tri sur les différents messages envoyés par les spécialistes sous forme d'événements et ne sera pas distraite par ceux qui ne la concernent pas. Il est à remarquer que, d'une part, les spécialistes trient leurs actions en signalant aux tâches celles qu'elles ont jugées nécessaires et que, d'autre part, chaque tâche sélectionne parmi les actions signalées celles qui l'intéressent. Ces deux niveaux de filtrage sur les événements à créer et sur leur propagation vers les tâches réduisent considérablement le nombre d'événements à traiter par le système et pour chaque tâche.

5. Système développé avec ATOME.

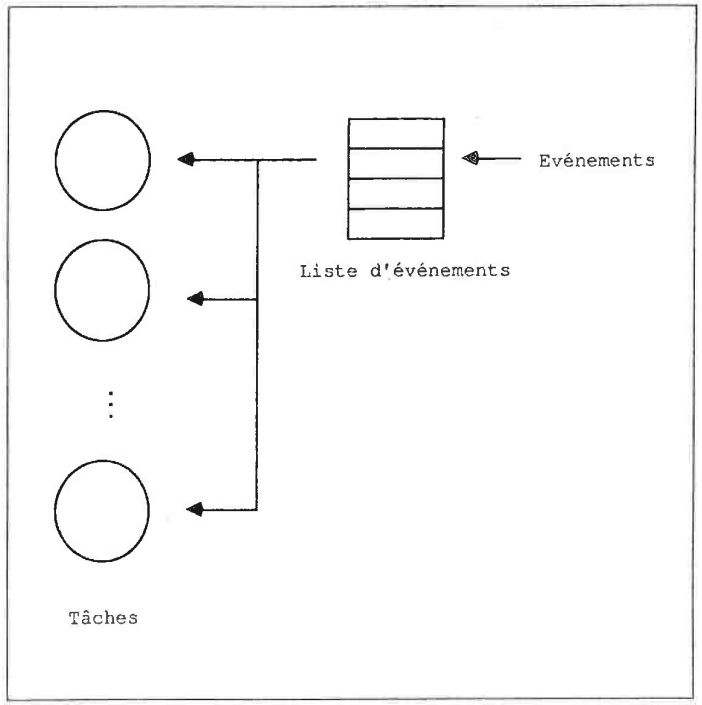


Figure 3.8. Liste d'événements commune à toutes les tâches.

Les événements utilisés lors de l'activation d'une tâche sont retirés de sa liste d'événements locale à la fin de cette activation. Les événements appartenant à plusieurs listes d'événements ne sont bien sûr retirés que de la liste de la tâche concernée. Ils restent de ce fait accessibles aux autres tâches. Ainsi, la liste des événements d'une tâche contient les informations qui intéressent la tâche et qu'elle n'a pas encore traitées. Ce couple (tâche, liste d'événements) constitue alors un système autonome comme l'indique la figure 3.9. Le module *propagation événements-vers-tâches* est chargé de filtrer les événements envoyés par les spécialistes afin de les propager vers les tâches intéressées. Pour ce faire, il dispose des connaissances fournies par les tâches quant aux types d'informations

susceptibles de les intéresser.

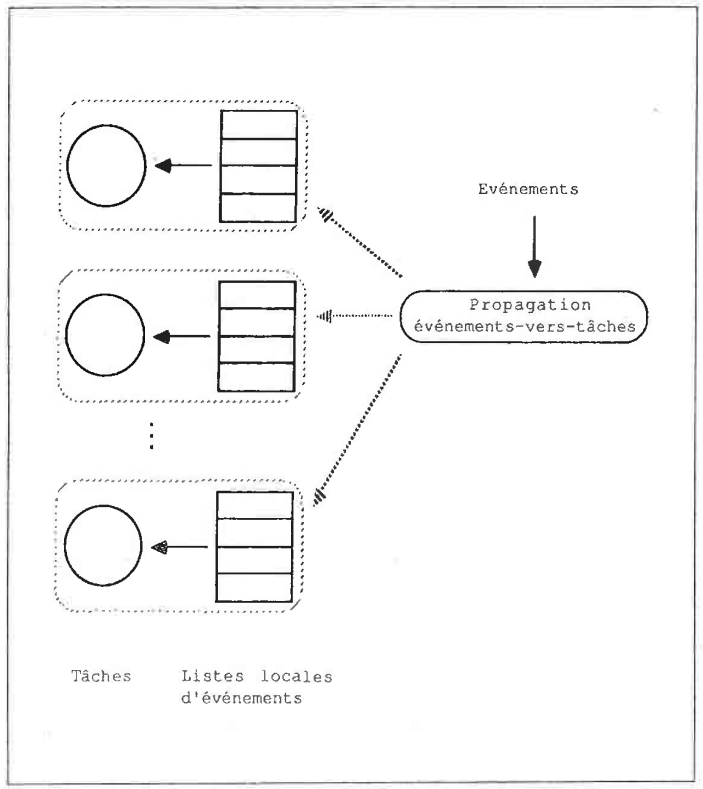


Figure 3.9. Listes des événements locales.

La position à laquelle un événement est placé dans une liste des événements est déterminée à partir de l'importance relative affectée à l'événement par la spécialiste l'ayant engendré.

Nous avons également introduit la notion de *synthèse d'événements* d'une même liste d'événements. Lorsqu'un événement est placé dans la liste d'événements d'une tâche, le système compare cet événement à ceux déjà existants dans cette liste. Il recherche si ce dernier peut ou non être synthétisé avec l'un d'eux. Pour cela, il vérifie si les deux événements :

- concernent la même hypothèse;
- représentent le même type d'action : soit modification, soit remplacement[6].

L'événement résultant de la synthèse de deux événements sera défini par les propriétés suivantes :

- *Niveau :* niveau de l'hypothèse sur laquelle portent les deux événements synthétisés.
- *Nœud :* nom de cette hypothèse.
- *Type-action :* modification ou remplacement selon le type de l'action à laquelle les deux événements synthétisés font référence.
- *Attributs-mis-à-jour :* contient l'union des attributs mis à jour cités dans les deux événements synthétisés.
- *Liens-mis-à-jour :* contient l'union des liens mis à jour cités dans les deux événements synthétisés.

Il sera placé dans la liste des événements à la position du plus important des deux événements synthétisés. L'événement *evt-j* de la figure 3.10 vient d'être propagé vers la liste d'événements locale à la tâche T. Cet événement est synthétisable avec l'événement *evt-i* déjà existant dans cette liste. Nous appelons *evt-k* l'événement résultant de cette synthèse. L'importance relative de *evt-j* est évaluée afin de déterminer la position qu'aurait occupée cet événement par rapport à *evt-i* et d'en déduire la position qui sera affectée à *evt-k*. *evt-i* sera bien sûr retiré de cette liste d'événements.

Une synthèse d'événements peut conduire à une synthèse entre des événements résultant eux-mêmes d'une synthèse et/ou des événements directement

6. On ne crée ou ne supprime qu'une seule fois une hypothèse. De ce fait, on ne rencontrera jamais plusieurs événements relatifs à une même création ou à une même suppression. Le problème de la synthèse d'événements ne se pose donc pas dans ce cas.

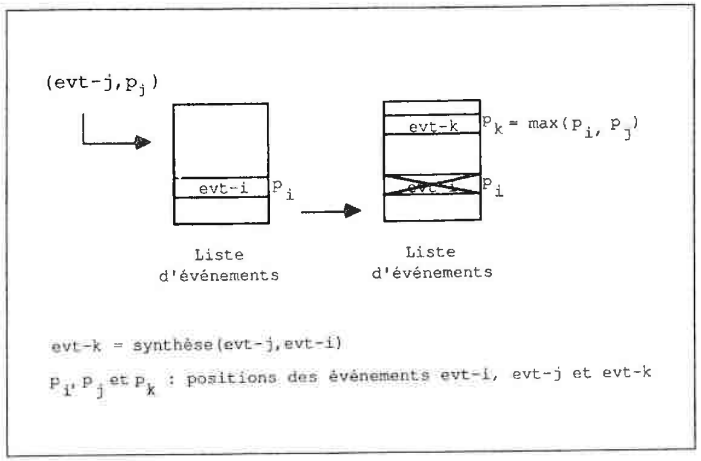


Figure 3.10. Synthèse de deux événements dans la liste d'événements de la tâche T.

envoyés par les spécialistes. Elle ne sera pas possible entre deux événements se rapportant l'un à une modification et l'autre à un remplacement d'une même hypothèse car ces deux actions n'ont pas le même comportement[7].

Elle permet de diminuer le nombre d'événements dans les listes d'événements locales aux tâches sans perdre de l'information d'une part, et de regrouper les informations faisant référence à des mêmes types d'action dans les blackboards d'autre part.

3.5.2 Résumés des blackboards

La notion de résumé du blackboard définie au chapitre précédent a été généralisée afin de prendre en compte l'utilisation de plusieurs blackboards dans un système-ATOME. A chaque blackboard est associé un résumé des hypothèses importantes qu'il renferme et qui peuvent intéresser le mécanisme de contrôle,

7. La modification conserve les anciennes valeurs des attributs d'une hypothèse tandis que le remplacement écrase ces valeurs.

en l'occurrence la stratégie. Celle-ci dispose ainsi d'une vue d'ensemble de la solution courante mémorisée dans les blackboards.

Les filtres, qui permettent de construire les *résumés des blackboards*, relèvent de l'expertise d'une application donnée. Ils permettent de spécifier *a priori* sous une forme déclarative les conditions sous lesquelles les hypothèses d'un certain niveau d'un blackboard ou d'un certain blackboard peuvent être placées dans le résumé correspondant.

3.6 Les tâches

Nous rappelons qu'une tâche est une méta-source de connaissances dont le rôle est d'effectuer un *contrôle local* sur un groupe de spécialistes en gérant l'ordre et l'instant d'intervention de ces dernières. Chaque tâche dispose d'une liste d'événements locale lui permettant d'avoir une vue de l'ensemble des changements du blackboard qu'elle n'a pas encore traités et qui sont susceptibles de l'intéresser. La tâche a en effet préalablement défini un filtre caractérisant les types d'événements qui doivent lui être signalés. L'ensemble des événements envoyés par les spécialistes à toutes les tâches sont automatiquement triés afin d'être propagés uniquement vers les tâches intéressées. La synthèse des événements dans la liste des événements d'une tâche permet de réduire le nombre d'éléments que cette tâche doit traiter et de regrouper les informations relatives à des actions similaires dans les blackboards. La liste des événements d'une tâche fournit donc des informations *condensées* sur ce qui s'est passé dans le blackboard et n'a pas encore été traité.

Nous avons défini au chapitre précédent une règle type d'une tâche. La forme d'expression de cette règle reste la même (cf. figure 3.11) mais le sens qui lui est attribué a changé. En effet, la partie gauche de la règle teste la présence d'événements dans *la liste d'événements locale* à la tâche et l'état de variables locales à la tâche ou globales. En revanche, sa partie droite contient une ou plusieurs spécialistes à activer de façon *séquentielle* ou *opportuniste* selon le *mode de raisonnement* de la tâche.
Une tâche est caractérisée par l'ensemble des propriétés citées dans la figure 3.12.
Il existe actuellement trois modes de raisonnements au niveau des tâches : une tâche peut être *dirigée par les événements*, *dirigée par les règles* ou *opportuniste*.

3.6.1 Tâche dirigée par les événements

Le principe de base de ce mode de raisonnement est similaire à celui décrit au chapitre précédent. Les principales différences sont liées au fait que la tâche

SI
tel(s) type(s) d'événement(s) est(sont) arrivé(s)

ALORS
SC4 SC1 SC2

où SC1 SC2 et SC4 sont des sources de connaissances spécialistes.

Figure 3.11. Règle type d'une tâche.

1. *Nom :* identification de la tâche.
2. *Filtre sur les événements :* caractérise les types d'événements qui intéressent la tâche.
3. *Variables locales :* variables locales de la tâche. Ces variables n'ont pas le même rôle que pour les spécialistes. Elles permettent de mémoriser une valeur ou un calcul intermédiaire mais ne constituent pas le contexte de travail principal de la tâche. Ce dernier est caractérisé par l'état de sa liste des événements locale au moment de son activation.

   Ces variables sont visibles par les spécialistes qui sont sous la responsabilité de cette tâche.
4. *Base de règles :* corps de la tâche.
5. *Mode de raisonnement :* indique si la tâche est dirigée par les événements, par les règles ou opportuniste.
6. *Mode de fonctionnement :* varie en fonction du mode de raisonnement de la tâche.

   Tâche dirigée par les événements :
   Le mode de fonctionnement dans ce cas peut être simple, multiple, cyclique-simple ou cyclique-multiple.

   Tâche dirigée par les règles :
   Dans ce cas, le mode de fonctionnement de la tâche peut être simple, multiple ou cyclique.

   Tâche opportuniste :
   Il n'y a pas de mode de fonctionnement dans ce cas.

Figure 3.12. Caractéristiques d'une tâche.

a sa propre liste d'événements et que les spécialistes sont dotées d'une précondition. La tâche choisit donc l'événement en tête de sa liste des événements - l'événement le plus important pour cette tâche - et recherche une ou plusieurs règles déclenchables avec cet événement. A chaque déclenchement de règles, des spécialistes sont sollicitées pour travailler *séquentiellement*. Si la précondition d'une des spécialistes n'est pas vérifiée, celle-ci n'est pas activée. De nouvelles actions sont effectuées dans les blackboards par les spécialistes activées. Ces spécialistes signalent aux tâches les actions qu'elles ont jugées pertinentes pour influencer le comportement du système en engendrant des événements. Parmi ces événements, chaque tâche sélectionne ceux qui l'intéressent. Certains événements sont éventuellement propagés vers la tâche en cours d'activation[8]. Les plus importants sont placés en tête de sa liste des événements, certains sont synthétisés avec d'autres et le cycle recommence (cf. figures 3.13 et 3.14).

La tâche dirigée par les événements *réagit* aux changements importants dans le blackboard, cette importance étant à *relativiser* par rapport aux besoins de la tâche. Un même événement peut avoir une importance différente dans deux listes d'événements différentes selon sa position dans chacune des listes.

De même qu'à l'étape précédente, la ou les spécialistes activées peuvent disposer, si elles le désirent, de l'hypothèse ayant permis leur activation[9].

3.6.2 Tâche dirigée par les règles

De la même façon qu'à l'étape précédente, chaque tâche cherche à activer sa règle la plus importante (cf. chapitre 2). Pour cela, elle consulte sa liste d'événements locale pour retrouver ceux qui vérifient la partie gauche de la règle. De même que dans le mode dirigé par les événements, chaque déclenchement de règles conduit à une activation *séquentielle* de spécialistes (celles qui ont leur précondition vérifiée) et à la mise à jour des listes d'événements du système. Les événements sont traités par groupes[10] et indépendamment de leur importance.

La ou les spécialistes activées peuvent disposer, si elles le désirent, des hypothèses ayant permis leur activation. Celles-ci leur sont fournies par la tâche en tant que focalisation de contrôle.

8. Il est possible qu'aucun événement n'intéresse cette tâche ou qu'au contraire, tous les événements l'intéressent.
9. Hypothèse associée à l'événement qui a réveillé la spécialiste. Celle-ci lui est fournie par la tâche en tant que focalisation de contrôle.
10. Ceux qui satisfont la partie gauche d'une même règle sont traités ensemble.

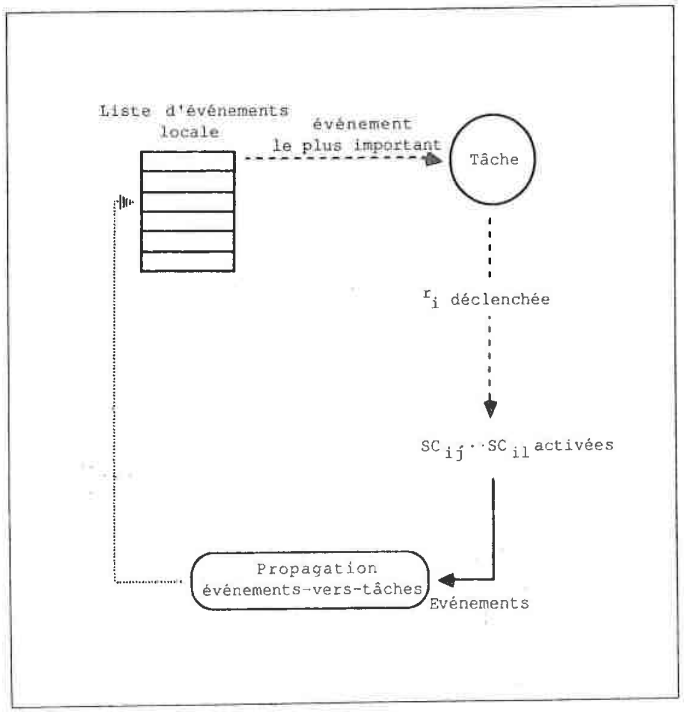


Figure 3.13. Cycle élémentaire d'une tâche dirigée par les événements

1. La tâche sélectionne l'événement le plus important.

   Il s'agit du premier événement de sa liste des événements locale non encore traité par la tâche.

2. Si aucune règle n'est activable avec cet événement, le *marquer* comme "traité" et retourner en 1.,

   Sinon, cas où le mode de fonctionnement de la tâche est :

   Simple :

   (a) Activer la première règle déclenchable avec cet événement.
   Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

   (b) Le *retirer* de sa liste des événements.

   Multiple :

   (a) Activer toutes les règles déclenchables en un seul passage avec cet événement.
   Le système n'effectue qu'un seul parcours de la base de règles.

   (b) Le *retirer* de sa liste des événements.

   Cyclique-simple :

   (a) Activer la première règle déclenchable avec cet événement.

   (b) Le *retirer* de sa liste des événements.

   (c) Retourner en 1.

   Boucle sur le mode simple pour chaque événement de la liste des événements locale à la tâche.

   Cyclique-multiple :

   (a) Activer toutes les règles déclenchables en un seul passage avec cet événement.

   (b) Le *retirer* de sa liste des événements.

   (c) Retourner en 1.

   Boucle sur le mode multiple pour chaque événement de la liste des événements locale à la tâche.

Figure 3.14. Algorithme d'activation d'une tâche dirigée par les événements.

Le cycle élémentaire de la tâche correspondant au déclenchement d'une de ses règles est illustré par la figure 3.15. L'algorithme d'activation d'une tâche dirigée par les règles est décrit dans la figure 3.16.

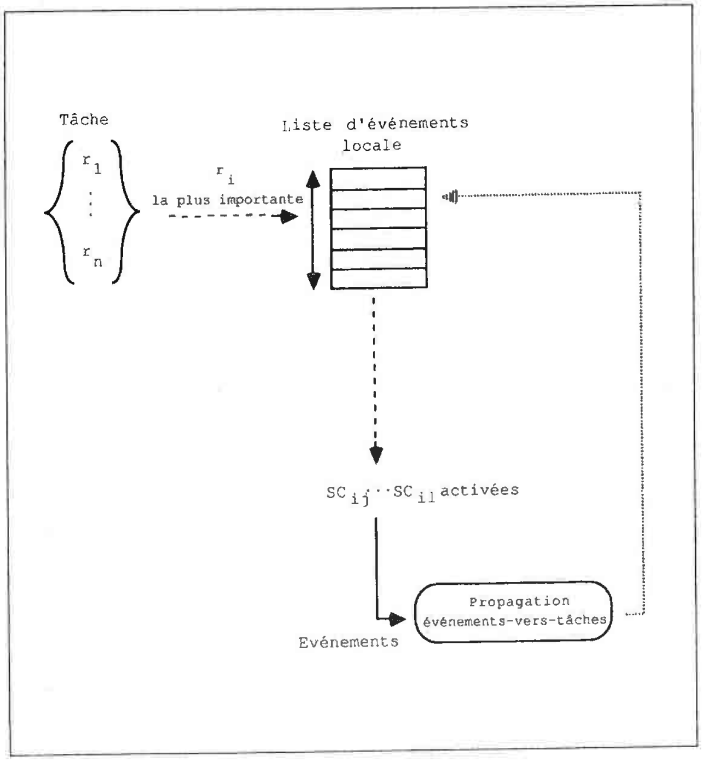


Figure 3.15. Cycle élémentaire d'une tâche dirigée par les règles.

Cas où son mode de fonctionnement est :

Simple : Activer la règle déclenchable la plus importante.

Un coefficient d'importance est associé à chaque règle, les règles à coefficients égaux étant classées dans leur ordre de définition.

Multiple : Activer toutes les règles déclenchables en un seul passage.

Le système n'effectue qu'un seul parcours de la base de règles.

Cyclique : Activer toutes les règles déclenchables jusqu'à saturation.

Le système boucle sur le mode multiple.

Figure 3.16. Algorithme d'activation d'une tâche dirigée par les regles.

3.6.3 Tâche opportuniste

Ce mode de raisonnement n'existait pas à l'étape précédente. Nous l'avons défini afin d'introduire une certaine souplesse dans le fonctionnement des tâches. En effet, nous avons vu qu'une tâche dirigée par les événements ou par les règles spécifie dans la partie droite de ses règles une ou plusieurs spécialistes à activer *séquentiellement*. Le mode opportuniste permettra de retrouver *dynamiquement* l'instant et l'ordre dans lequel des spécialistes en conflit doivent être activées. Ce mécanisme sera d'une grande utilité lorsque le flux de contrôle (instant et ordre d'intervention des spécialistes) n'est pas bien connu et ne peut être spécifié à l'aide de tâches dirigées par les événements ou par les règles. Ce cas peut se présenter par exemple en début de développement d'une application ou au cours du développement d'une application faisant intervenir des connaissances de contrôle relatives à une expertise mal maîtrisée.

Le mode de raisonnement opportuniste est fondé sur la notion d'*agendas* et sur la notion de *priorités*. Une priorité est une valeur affectée à chaque élément en attente dans les agendas afin de permettre l'accès au plus prioritaire (cf. figure 3.17).

Principe de fonctionnement

Une tâche opportuniste recherche, pour chacun des événements de sa liste des événements locale, l'ensemble des règles déclenchables avec cet événement. L'activation d'une règle *réveille* les spécialistes figurant dans sa partie droite. La tâche prévient ainsi les spécialistes que cet événement peut les intéresser et éventuellement leur permettre de travailler. La règle de la tâche représente en fait la spécification du *trigger* des spécialistes qu'elle contient en partie droite.

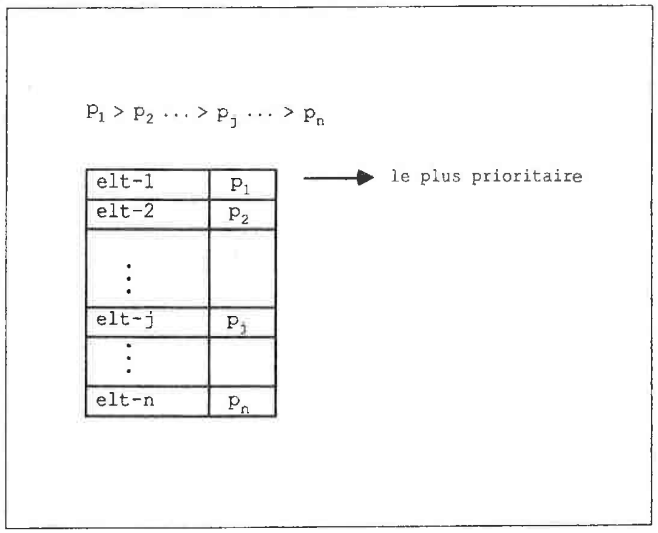


Figure 3.17. Agenda dont les éléments sont rangés par priorité décroissante.

notion définie dans BB-1 au chapitre 4 de la partie II et rappelée au chapitre précédent.

Les spécialistes *éveillées* sont alors placées dans un agenda sous la forme d'*Instances de SPécialistes* (ISPs), chacune mémorisant le nom de la spécialiste concernée ainsi que l'événement qui l'a réveillée[11]. Cet agenda appelé *agendas des* ISPs *sélectionnées* regroupe donc toutes les ISPs susceptibles de vouloir travailler par la suite. Nous dirons qu'il mémorise l'ensemble des *travaux potentiels* dont la tâche dispose.

Après avoir déterminé toutes les ISPs en parcourant entièrement sa liste des événements, la tâche retire de cette liste les événements qui ont permis de réveiller des spécialistes et recherche parmi les ISPs en attente dans l'agenda des

11. Une ISP représente l'équivalent d'une ISC définie au chapitre 1 de la partie II.

ISPs sélectionnées celles qui peuvent effectivement travailler, c'est-à-dire celles dont la précondition est vérifiée[12]. Nous dirons que ces ISPs sont *activables* et qu'elles constituent des *activités potentielles*. Les ISPs activables sont placées dans un autre agenda appelé *agenda des ISPs exécutables* et sont retirées de l'agenda des ISPs sélectionnées. L'agenda des ISPs exécutables regroupe ainsi l'ensemble des activités potentielles en concurrence pour travailler.

La tâche doit alors choisir l'une de ces ISPs activables et l'exécuter. Ce choix est basé sur la priorité affectée aux ISPs qui est calculée à partir d'un ensemble d'*heuristiques* et de la *règle d'intégration* indiquant comment prendre en compte ces heuristiques. Cette dernière est fournie à la tâche par la stratégie (cf. § "Calcul des priorités").

La tâche active la spécialiste associée à l'ISP la plus prioritaire en lui fournissant l'événement qui l'a réveillée en tant que *focalisation de contrôle*. L'exécution de cette spécialiste provoque la création de nouveaux événements qui sont propagés vers les tâches. Les nouveaux événements reçus par la tâche en cours d'activation éveilleront ou activeront d'autres spécialistes en plus de celles déjà présentes dans les agendas. L'arrêt de la tâche est provoqué lorsque l'agenda des ISPs exécutables est vide.

Ce mécanisme de fonctionnement constitue le cycle de base de la tâche appelé *cycle local*. Il est illustré par la figure 3.18. En résumé, chaque cycle local à une tâche opportuniste consiste, d'une part, à déterminer tous les travaux potentiels et toutes les activités potentielles de la tâche, puis, d'autre part, à exécuter l'activité potentielle la plus prioritaire, c'est-à-dire l'ISP exécutable la plus prioritaire (cf. figure 3.19).

Maintenance des agendas

Un système de maintenance des agendas est défini de manière à vérifier si la précondition d'une ISP de l'agenda des ISP exécutables est toujours valide, sinon cette ISP est replacée dans l'agenda des ISPs sélectionnées. Ces vérifications sont effectuées à chaque cycle local de la tâche.

Afin de limiter le nombre d'ISPs manipulées dans les agendas, nous avons introduit une propriété supplémentaire qu'il est possible de définir pour une spécialiste : spécification de ses *conditions de rejet*. Cette propriété permet d'exprimer les situations pour lesquelles une spécialiste en association avec

12. Précondition de la spécialiste associée à l'ISP.

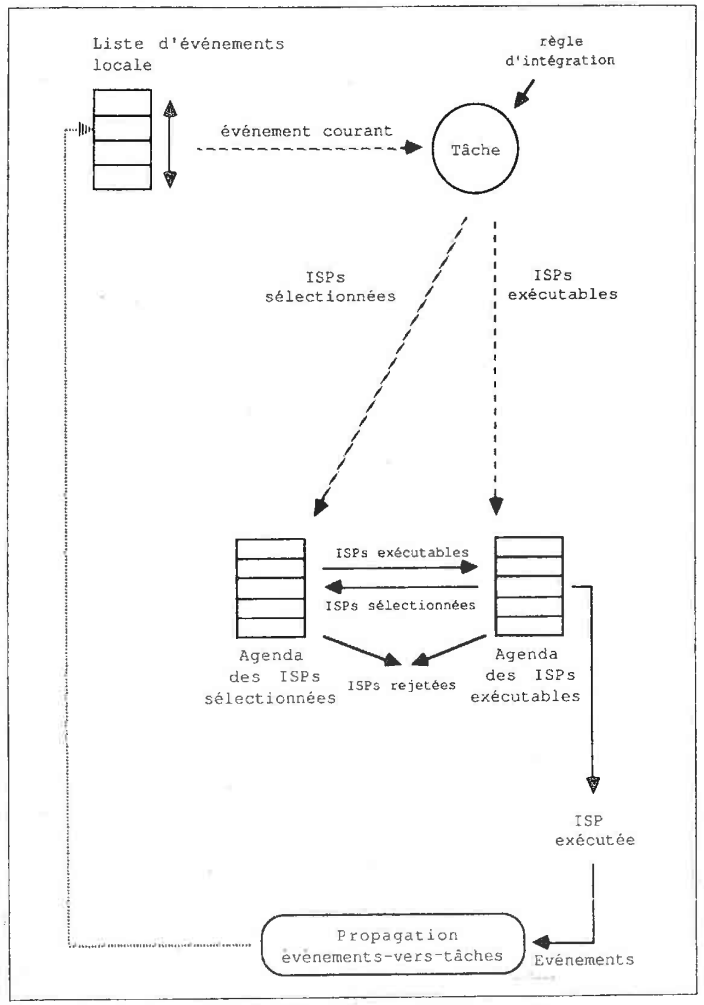


Figure 3.18. Cycle local d'une tâche opportuniste.

1. *Phase de création des* ISPs *:*

   Pour chaque événement <*evt*> de la liste d'événements

   Pour chaque règle <*r*>

   SI <*r*> est activable avec <*evt*>

   ALORS

   Pour chaque spécialiste <*sp*> de la partie droite de <*r*>

   (a) Créer une <ISP>=<*sp*> ⊕ <*evt*>.

   (b) L'ajouter dans l'agenda des ISPs *sélectionnées.*

2. *Phase de mise à jour des agendas :*

   (a) Pour chaque ISP de l'agenda des ISPs *sélectionnées*

   SI les conditions de rejet de cette ISP sont vérifiées

   ALORS la supprimer de l'agenda des ISPs sélectionnées

   SINON

   Si ses préconditions sont vérifiées alors la déplacer vers l'agenda des ISPs *exécutables.*

   (b) Pour chaque ISP de l'agenda des ISPs *exécutables*

   SI les conditions de rejet de cette ISP sont vérifiées

   ALORS la supprimer de l'agenda des ISPs exécutables

   SINON

   Si ses préconditions ne sont plus vérifiées alors la déplacer vers l'agenda des ISPs *sélectionnées.*

3. *Phase de sélection et d'exécution de l'*ISP *la plus prioritaire :*

   Pour chaque ISP de l'agenda des ISPs *exécutables*

   Appliquer la règle d'intégration en cours à cette ISP.

   (a) Activer l'ISP la plus prioritaire.

   (b) La supprimer de l'agenda des ISPs exécutables.

   (c) Aller en 1.

La tâche s'arrête quand l'agenda des ISPs exécutables est vide.

Figure 3.19. Activation d'une tâche opportuniste.

l'événement qui l'a réveillée n'est plus capable de fournir un travail et est à retirer définitivement des agendas. Elle n'est pas redondante par rapport à la précondition d'une spécialiste car la vérification de cette précondition permet de faire passer une ISP d'un agenda à un autre mais pas de la retirer définitivement des agendas. L'exemple du voyageur de commerce présenté au chapitre 5 met en évidence un cas typique d'utilisation de ce mécanisme.

Ces conditions de rejet sont exprimées sous forme de tests sur l'état des blackboards ou d'une partie des blackboards. En fait, le filtre associé à la précondition d'une spécialiste pour déterminer son contexte préliminaire sera étendu à ces conditions de rejet. Le contexte préliminaire sera ainsi évalué soit pendant la phase de vérification de la précondition d'une spécialiste, soit pendant sa *phase de test des conditions de rejet* (cf. figure 3.20). Les variables de préconditions permettant de définir le filtre sur les blackboards pour construire ce contexte préliminaire seront appelées dans la suite *variables de contexte.*

1. La spécialiste crée son contexte préliminaire.

   Les variables de contexte de la spécialiste sont liées aux hypothèses filtrées sur les blackboards et constituent alors son contexte préliminaire.

2. Validité des conditions de rejet :

   Teste si les conditions spécifiées sont satisfaites.

Figure 3.20. Phase de test des conditions de rejet d'une spécialiste.

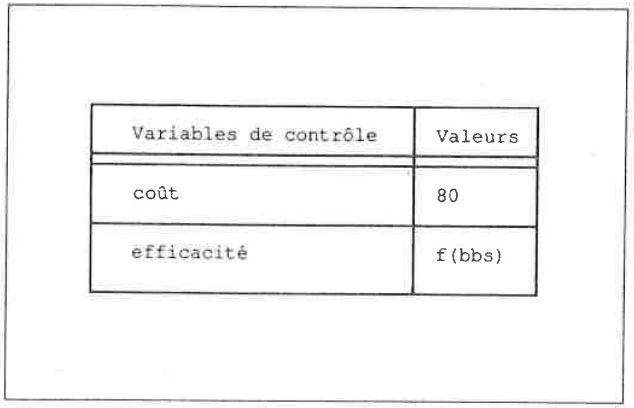
Calcul des priorités

Nous avons précisé que la priorité d'une ISP est calculée à partir d'heuristiques et de la règle d'intégration à appliquer sur ces heuristiques fournie à la tâche par la stratégie. Ces heuristiques sont basées sur des paramètres qui peuvent être attribués à chaque spécialiste et de ce fait aux ISPs associées. Ces paramètres représentent des facteurs discriminants des spécialistes pour une application donnée. Ils peuvent caractériser par exemple l'efficacité, la rapidité ou l'importance d'une spécialiste. La plupart des outils pour développer des systèmes d'IA figent ces facteurs à l'avance. Nous permettons à tout utilisateur d'ATOME de définir lui-même ces composants sous forme de *variables de contrôle*, les *heuristiques* à appliquer permettant de favoriser un composant ou de moduler son influence dans le calcul des priorités, ainsi que les *règles d'intégration* permettant de favoriser ou combiner les heuristiques définies dans le système.

1. *Variables de contrôle :*

   Nous désignons par *variable de contrôle* tout paramètre associé à une spécialiste et pouvant intervenir dans le calcul des priorités des ISPs. Chaque variable de contrôle sera affectée d'une valeur. Cette valeur pourra être *constante* durant toute l'exécution du système ou *évolutive*, dépendante du contexte dans lequel le système évolue : état des blackboards, événement associé à la spécialiste au moment où la priorité est calculée, importance de cet événement, durée de l'exécution, cycle d'exécution,...

   Les éléments à prendre en compte et leur intervention dans le calcul de cette valeur sont définis par l'utilisateur. La figure 3.21 montre que le paramètre *coût* de la spécialiste SC1 vaut *80* tandis que son efficacité dépend de l'état des blackboards.



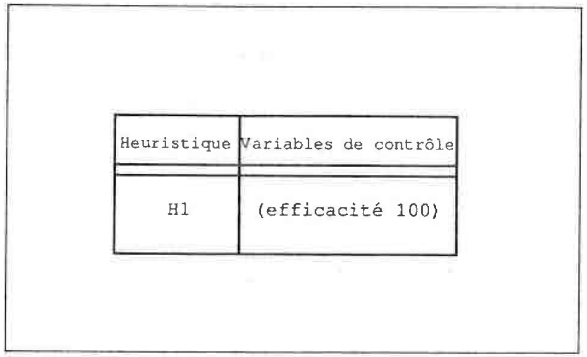
Variables de contrôle	Valeurs
coût	80
efficacité	f(bbs)

Figure 3.21. Paramètres associés à la spécialiste SC1.

2. *Heuristiques :*

   Le but d'une heuristique est de favoriser ou défavoriser l'influence de certains paramètres dans le calcul des priorités des ISPs. Un exemple simple serait de définir une heuristique H1 favorisant les ISPs les plus efficaces.

Une heuristique est définie par l'ensemble des variables de contrôle qu'elle souhaite prendre en compte et par les *poids* qu'elle leur affecte. La figure 3.22 fournit une représentation possible de H1 qui consiste à ne considérer que le paramètre *efficacité* de l'ISP et à lui affecter le poids maximum.



Heuristique	Variables de contrôle
H1	(efficacité 100)

Figure 3.22. Heuristique H1.

L'utilisateur définit ainsi un ensemble d'heuristiques à utiliser durant l'exécution du système. Les heuristiques à prendre en compte lors de l'exécution d'une tâche opportuniste sont spécifiées par les règles d'intégration données par la stratégie.

3. *Règles d'intégration :*

   Une règle d'intégration fournit à une tâche opportuniste la règle à appliquer pour évaluer la priorité des ISPs. L'ISP la plus prioritaire sera celle dont la priorité est la plus élevée.

   Une règle d'intégration consiste à favoriser ou défavoriser certaines heuristiques. Elle est définie par un ensemble d'heuristiques à appliquer à l'ISP et par le poids qu'elle leur affecte. Un exemple simple serait de définir une règle d'intégration R1 qui considérerait l'ISP la plus efficace comme la plus prioritaire. La figure 3.23 fournit une représentation possible de R1 qui consiste à ne considérer que l'heuristique H1 définie précédemment et à lui affecter le poids maximum.

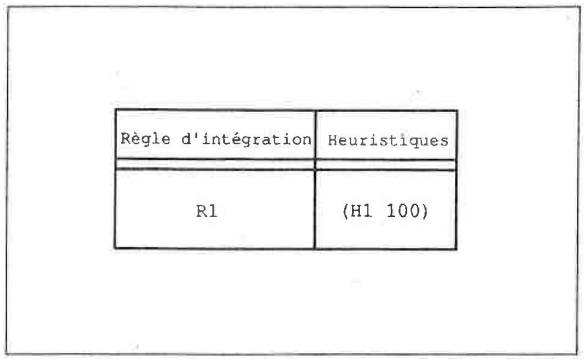


Figure 3.23. Règle d'intégration R1.

L'utilisateur définit ainsi un ensemble de règles d'intégration des heuristiques à utiliser durant l'exécution du système. Elles sont mises à la disposition de la stratégie qui, lorsqu'elle est amenée à activer une tâche opportuniste, lui indique la règle d'intégration à appliquer. Une même tâche peut être activée plusieurs fois avec des règles d'intégration différentes.

En résumé, le calcul de la priorité d'une ISP est effectué en plusieurs étapes. Il consiste dans un premier temps à appliquer les différentes heuristiques Hj spécifiées dans la règle d'intégration Rk fournie à la tâche. Pour chacune de ces heuristiques, il s'agit d'évaluer les variables de contrôle Vi de l'ISP qui sont spécifiées dans l'heuristique. Les résultats obtenus sont ensuite combinés en fonction des poids p_i^j affectés à ces variables de contrôle et fournissent ainsi une évaluation de l'ISP face à l'heuristique Hj considérée. Les résultats obtenus pour chacune des heuristiques sont ensuite combinés selon la règle d'intégration et fournissent ainsi la priorité de l'ISP comme l'indique la figure 3.24.

En fait, une même règle d'intégration peut proposer plusieurs combinaisons possibles d'heuristiques en les classant par ordre de préférence. La combinaison appliquée pour une ISP donnée sera la première pour laquelle les heuristiques qu'elle fait intervenir peuvent être évaluées : toutes les variables de contrôle qui interviennent dans ces heuristiques ont été définies pour cette ISP.

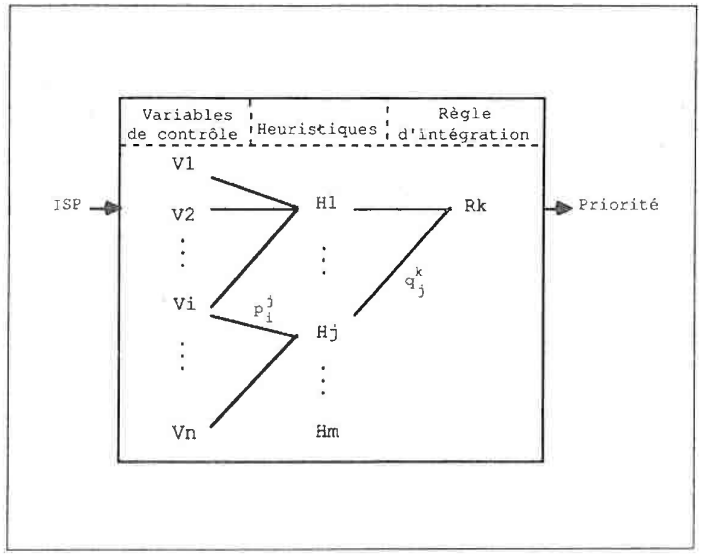


Figure 3.24. Calcul de la priorité d'une ISP.

Il est à remarquer que la tâche calcule ou recalcule les priorités des ISPs activables à chaque cycle local, c'est-à-dire après l'activation d'une seule ISP. Le fonctionnement d'une tâche opportuniste est donc coûteux en temps de calcul. Nous nous sommes intéressés à ce problème et avons introduit la notion de *stabilité* des heuristiques. Une heuristique *stable* ne sera évaluée qu'une seule fois pour une même ISP. Cette valeur sera mémorisée et restituée lorsque la priorité de l'ISP sera recalculée. En revanche, une heuristique *instable* sera réévaluée à chaque fois que la priorité de l'ISP sera recalculée. Cette notion de stabilité permet en particulier de distinguer les heuristiques prenant en compte des variables de contrôle constantes ou évolutives.

3.6.4 Comparaison des tâches

Les tâches dirigées par les événements ou par les règles sont plus efficaces que les tâches opportunistes mais exigent une spécification plus précise et plus fine de l'expertise dans la résolution de conflit d'intervention des différentes SCs spécialistes, car le contexte et l'ordre d'exécution des spécialistes doivent être précisés. En contrepartie, elles conduisent à une certaine rigidité du système. Les tâches opportunistes sont plus souples dans la mesure où l'expertise ne déclare que le contexte d'intervention des spécialistes, leur ordre d'activation étant déterminé dynamiquement en fonction des paramètres associés aux spécialistes, des heuristiques et des règles d'intégration à appliquer.

Une tâche opportuniste est utile lorsque le flux de contrôle n'est pas connu. Elle peut de plus servir de *phase d'apprentissage* pour mieux cerner l'expertise impliquée dans le flux de contrôle et être éventuellement réécrite par la suite sous forme de tâche dirigée par les événements ou par les règles si l'expertise acquise le permet, ceci pour des raisons d'efficacité.

Une tâche dirigée par les événements permet de traiter les événements au fur et à mesure de leur apparition dans la liste des événements locale à la tâche en fonction de leur importance, tandis que les tâches dirigées par les règles sont adaptées au contrôle de spécialistes déclenchables avec plusieurs événements en contexte indépendamment de leur importance.

Il est à remarquer que le déclenchement d'une règle d'une tâche n'a pas le même sens si la tâche est opportuniste ou dirigée par les événements ou par les règles. En effet, dans le premier cas, il conduit à la création d'ISPs tandis que dans les deux autres cas, il conduit à des activations de spécialistes.

D'autre part, les événements utilisés par une tâche, quel que soit son mode de raisonnement, sont retirés de sa liste des événements à la fin de son activation. Ceci ne perturbe en rien les autres tâches qui peuvent continuer à accéder à cet événement s'il appartenait à leur liste d'événements locale. Ceci permet également à une tâche de ne pas s'attarder sur des événements qu'elle a déjà traités au cours d'une activation antérieure.

Chaque tâche constitue donc un système *autonome* capable, à partir de sa propre liste des événements, de son propre mode raisonnement et de l'expertise qu'elle renferme, de gérer un sous-ensemble des spécialistes d'un système-ATOME.

3.7 La stratégie

La stratégie dispose des *résumés des blackboards* pour évaluer l'avancement du système et déterminer vers quelles régions de données le système doit se focaliser, les problèmes à résoudre et la manière de les résoudre. Elle permet d'exprimer un *contrôle global* du système. Plusieurs SCs de type stratégie peuvent être associées à un système-ATOME mais ne peuvent pas être utilisées simultanément. Ceci permet de tester différentes configurations du système afin de choisir la meilleure.

Nous avons défini au chapitre précédent une règle type de la stratégie. La forme d'expression de cette règle a changé dans la mesure où la stratégie doit fournir aux tâches opportunistes une règle d'intégration à appliquer. La partie gauche de la règle teste la présence d'hypothèses dans les résumés des blackboards. La partie droite contient une ou plusieurs tâches à activer séquentiellement lorsque la règle est déclenchée. Si l'une de ces tâches est opportuniste, une règle d'intégration lui sera associée. Une même tâche opportuniste peut apparaître plusieurs fois dans les règles de la stratégie avec une règle d'intégration différente. Nous donnons dans la figure 3.25 l'exemple d'une règle de la stratégie où la tâche opportuniste TA1 est appelée deux fois respectivement avec les règles d'intégration R1 et R2. Le déclenchement de cette règle provoque l'activation de TA1 avec pour règle d'intégration R1, puis l'activation de TA2 et enfin une nouvelle activation de TA1 avec la règle d'intégration R2.

SI
tel(s) type(s) d'hypothèse(s) existe(ent) dans les résumés des blackboards

ALORS
(TA1 . R1) TA2 (TA1 . R2)

où TA1 est une tâche opportuniste, et TA2 une tâche dirigée par les événements ou par les règles.

Figure 3.25. Exemple d'une règle de la stratégie.

La figure 3.26 illustre la structure d'ATOME face à toutes les extensions décrites précédemment.

3.8 Mémoire à long terme

Le modèle du blackboard est prévu pour être utilisé dans des applications complexes faisant intervenir des milliers d'hypothèses. Un système-ATOME n'échappe pas à cette règle. Parmi ces hypothèses, certaines sont constamment évolutives tandis que d'autres sont moins souvent sollicitées par le système. Parmi ces dernières, certaines ne sont plus traitées depuis longtemps. Elles

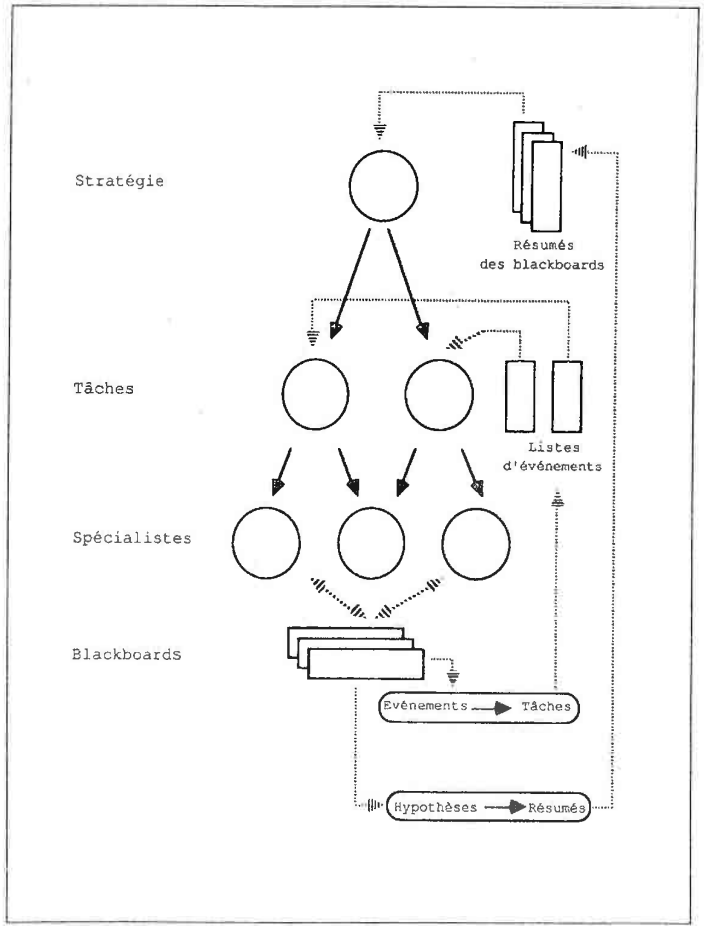


Figure 3.26. Nouvelle architecture d'ATOME.

peuvent être considérées *a priori* comme *anciennes* et *non productives*. Ce sont des hypothèses qui alourdissent les temps de parcours dans le(les) blackboard(s) même si elles ne sont plus utilisées. Cependant, elles ne doivent pas être supprimées dans la mesure où elles contiennent de l'information potentiellement utile. Il est donc indispensable de prévoir un mécanisme de ménage dans le(les) blackboard(s) qui permette de reléguer cette information sans la *perdre* définitivement.

Pour ces raisons, nous avons introduit dans ATOME la notion de *mémoire à long terme*. Un *système de gestion des hypothèses* est défini afin de détecter les hypothèses anciennes et non productives des blackboards, de les supprimer des blackboards et de les stocker dans une mémoire à long terme. En fait, chaque blackboard susceptible de contenir de telles hypothèses se verra attribuer une mémoire à long terme que nous appellerons *blackboard à long terme*. Nous avons choisi d'implanter chaque blackboard à long terme sous forme de fichier afin de récupérer la place mémoire qu'occupaient ces hypothèses. On pourrait envisager de l'implanter sous forme de base de données ou sous forme de blackboard au sens habituel du terme.

Chaque blackboard à long terme est accessible en lecture par les spécialistes du système; ainsi, l'information n'est pas perdue (cf. figure 3.27). Cependant, ces accès devront être formulés explicitement par les spécialistes.

C'est à l'utilisateur d'ATOME de spécifier les blackboards pour lesquels ce type de ménage doit être fait. Il doit également définir les critères selon lesquels les hypothèses d'un blackboard sont anciennes et non productives. Ces critères sont fondés sur le nombre de cycles durant lesquels une hypothèse n'a pas été traitées[13]. Ce ménage sera effectué périodiquement sur les blackboards. Une *période*[14] sera en effet associée à chaque blackboard. Le cycle à partir duquel les opérations de ménage seront lancées sur ce blackboard doit également être précisé. La figure 3.28 indique que le ménage sera effectué à partir du $20^{ième}$ cycle sur le blackboard *solution*. Cette opération sera ensuite reproduite tous les *5* cycles. Les hypothèses de ce blackboard qui n'ont pas été modifiées depuis les *10* derniers cycles seront transférées dans le blackboard à long terme associé au blackboard *solution* lors de ces opérations de ménage.

13. Un cycle représente la boucle de base du système.
14. Exprimée en nombre de cycles.

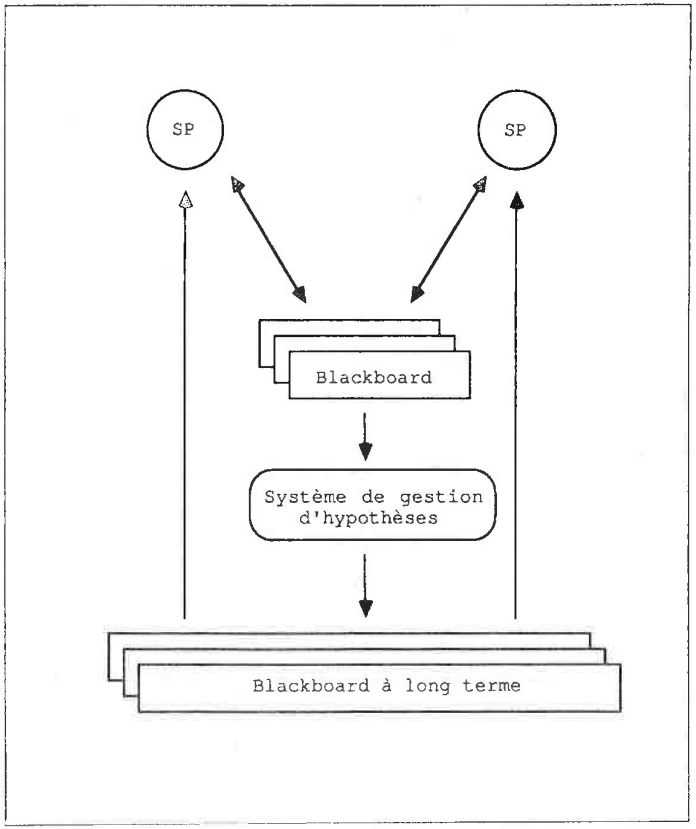


Figure 3.27. Blackboards à long terme.

Blackboard	Cycle-début	Période	Durée-non-maj
solution	20	5	10

Figure 3.28. Ménage du blackboard *solution*.

3.9 Architecture résultante

Le but de ce paragraphe est de faire une synthèse sur l'architecture résultante d'ATOME face aux extensions décrites précédemment.

ATOME définit trois types de sources de connaissances :

1. *les spécialistes* :

   Ce sont des modules autonomes et indépendants entre eux, qui regroupent les connaissances liées au sujet traité. Chaque spécialiste est experte pour un problème bien particulier du domaine d'application.

   En communiquant à travers un ou plusieurs blackboards, elles construisent ensemble les éléments de la solution d'une façon incrémentale et opportuniste. La qualité de la contribution de chaque spécialiste dépend de la qualité des données qu'elle a en entrée ainsi que de l'expertise qu'elle renferme. Etant donné que chacune des spécialistes n'a qu'une vue partielle du problème et de la solution courante, une coopération est nécessaire entre elles afin de combiner leurs résultats partiels en une solution finale.

   Les spécialistes sont *principalement* constituées de deux parties :

   (a) leur précondition qui définit les configurations du blackboard pour lesquelles la spécialiste devient *activable*,

(b) leur corps constitué de leur base de règles[15].

L'appel d'une spécialiste provoque une suite d'actions dans les blackboards : créations de nouvelles hypothèses, modifications, remplacements ou suppressions d'anciennes hypothèses.
Seules les actions jugées importantes par la spécialiste sont signalées au mécanisme de contrôle via ce que l'on appelle des *événements*. Le fait de juger qu'une action doit être ou non signalée au mécanisme de contrôle, en l'occurrence aux tâches, fait partie de l'expertise fournie par l'utilisateur.

Les événements constituent ainsi le moyen de communication utilisé par les spécialistes pour prévenir les tâches des dernières modifications importantes qu'elles ont effectuées sur la solution courante — les blackboards.

2. *les tâches* :

   Ce sont des méta-sources de connaissances qui renferment une partie des connaissances liées au contrôle - une tâche fournit un contrôle local pour coordonner le travail d'un ensemble de spécialistes. Pour ce faire, chacune d'elles dispose d'une structure de contrôle locale, appelée *liste des événements* où sont stockés les changements du blackboard qui l'intéressent.
   En fonction du contenu de sa liste d'événements, chaque tâche sélectionne un ensemble de spécialistes et un ordre dans lequel elles vont être exécutées. Cet ordre peut être donné explicitement par la tâche, cas des tâches dirigées par les événements ou par les règles, ou calculé à l'aide d'heuristiques, cas des tâches opportunistes.

3. *la stratégie* :

   C'est une méta-méta-source de connaissances qui fournit un contrôle global au système. Son rôle est de gérer le travail des tâches en fonction de la qualité de la solution courante stockée dans les blackboards. Pour mesurer cette qualité, la stratégie dispose de structures de contrôle appelées *résumés des blackboards* qui lui offre une vue globale et synthétisée de la configuration des blackboards. Les résumés des blackboards ne contiennent que les informations jugées importantes pour la suite du processus de résolution du problème.

La figure 3.30 décrit les différentes composantes d'ATOME. La figure 3.29 résume le flux de contrôle dans ATOME : la stratégie consulte les résumés des blackboards et sélectionne une série de tâches à activer. La tâche en cours d'activation consulte sa liste d'événements, choisit un ensemble de spécialistes

15. Nous verrons dans le chapitre 7 de cette partie que le corps d'une spécialiste pourra également prendre la forme d'un programme, ou d'un système expert développé à l'aide d'outils d'IA tels que SAGANE ou IROISE.

et les active en fonction de son type de contrôle. La spécialiste en cours d'activation consulte les régions des blackboards qui l'intéressent et émet de nouvelles hypothèses ou modifie des hypothèses déjà existantes. Certaines hypothèses engendrent des événements qui sont filtrés et propagés vers les listes d'événements des tâches intéressées. Ces événements sont éventuellement synthétisés. Les hypothèses sont filtrées pour être ou non placées dans les résumés des blackboards ou en être retirées. Puis, le cycle recommence.

Quand la stratégie (respectivement une tâche) active une séquence de tâches (respectivement un ensemble de spécialistes), elle passe le contrôle à la tâche (respectivement la spécialiste) courante et se met en attente. Lorsque la dernière tâche (respectivement spécialiste) a terminé son exécution, elle redonne le contrôle à la stratégie (respectivement à la tâche mère) qui ensuite activera une séquence d'autres tâches (respectivement spécialistes), etc.

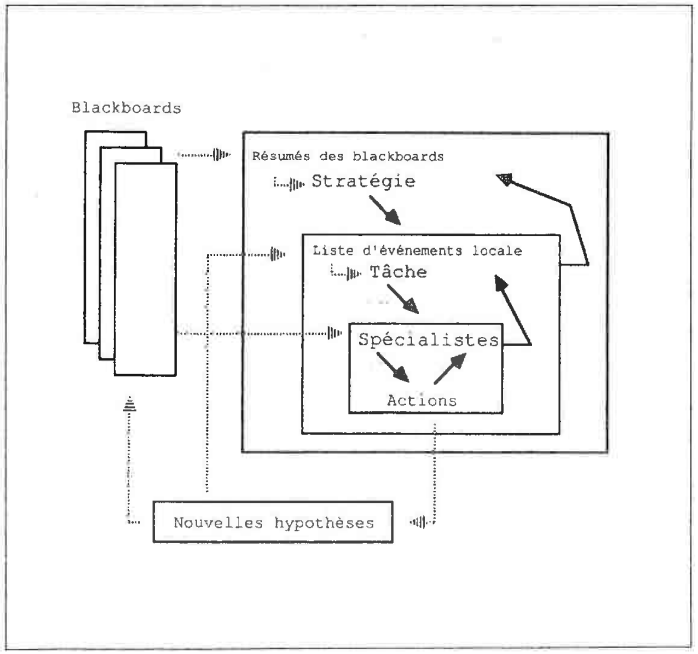


Figure 3.29. Flux de contrôle dans ATOME

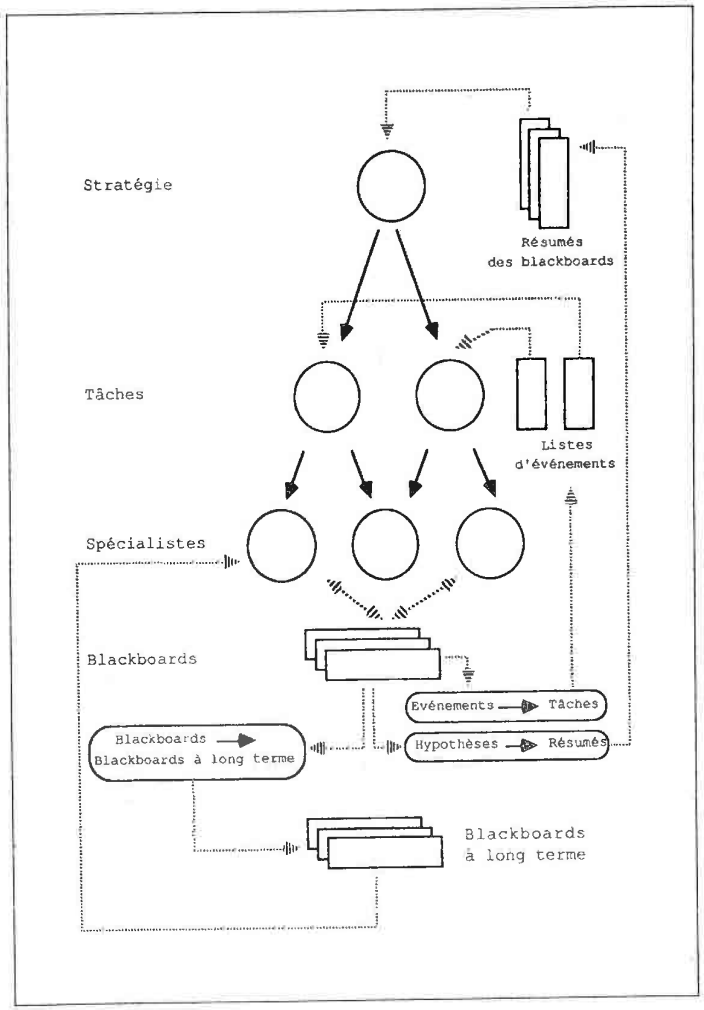


Figure 3.30. Organisation des sources de connaissances dans ATOME.

3.10 Le modèle hybride à multi-phases

Notre but dans cette section est de définir un modèle à partir de l'architecture que nous avons mise en œuvre dans ATOME. Ce modèle est basé sur le fait que la résolution d'un problème est généralement divisée en plusieurs étapes, un cas extrême étant de résoudre le problème en une seule étape. De nombreux systèmes sont basés sur ce principe. Le modèle du contrôle à base de blackboard, par exemple, oriente la résolution du problème vers une suite d'objectifs à résoudre (cf. BB-1 [121]). ATOME résoud le problème par l'activation successive de tâches, chacune caractérisant un sous-problème à traiter.

A partir de ce principe, les systèmes BBB [18] et ERASMUS [16] ont introduit la notion de *phases*. Ils définissent *a priori* la séquence de phases que le système devra suivre, ainsi que les SCs actives durant chaque phase. Ils utilisent un mécanisme de contrôle *uniforme* pour toute la résolution du problème.

Nous respectons cette notion de phases dans la mesure où l'activation d'une tâche matérialise une phase dont les SCs actives sont les spécialistes placées sous la responsabilité de la tâche. Cependant, dans ATOME, le sous-ensemble des SCs actives durant une phase n'est pas défini *a priori*, il est déterminé dynamiquement par la stratégie qui, en fonction de l'état de la solution courante, choisit une séquence de tâches à activer.

De plus, dans ATOME, chaque tâche a son propre mode de raisonnement selon lequel elle coordonne les activités des spécialistes : la tâche peut en effet être dirigée par les événements, dirigée par les règles ou opportuniste. Chaque tâche applique donc un type de contrôle particulier adapté au sous-problème qu'elle doit traiter. Nous dirons que le mécanisme de contrôle dans ATOME est *hybride*.

Dans ATOME, la stratégie effectue un *contrôle global* de la résolution du problème afin de faire évoluer le système de phase en phase. Les tâches proposent différents types de *contrôles locaux* appropriés pour chaque phase en cours de résolution. Nous appellerons le mécanisme de contrôle dans ATOME : *contrôle hybride à multi-phases*. Nous classerons ce type de contrôle en plus des types de contrôle définis dans la partie II, à savoir : contrôle procédural, contrôle hiérarchique et contrôle à base de blackboard.

A partir de ce contrôle hybride à multi-phases, nous pouvons déduire le *modèle hybride à multi-phases*.

Le modèle hybride à multi-phases

Soit $\mathcal{P}$ le problème à résoudre.

Soit $\mathcal{PH}$ l'ensemble des phases PH_i de résolution du problème $\mathcal{P}$.

$$\mathcal{PH} = \{PH_i \mid i=0, 1, 2\ldots\}$$

Soit $\mathcal{S}$ l'ensemble des différents états de la solution courante du problème à chaque phase.
Soit $\mathcal{A}$ l'ensemble des actions potentielles du système.
Soit $\mathcal{C}$ l'ensemble des types de contrôles locaux utilisés durant chaque phase.

La résolution du problème $\mathcal{P}$ est définie par la suite des phases $PH_0, PH_1\ldots$. Chaque phase est caractérisée entre autres par l'état de la solution courante S_i, par un sous-ensemble $\mathcal{A}_i$ des activités potentielles du système et par le type de contrôle local utilisé C_i.

La fonction de contrôle *Création-phase* définit une phase à partir des éléments cités précédemment par :

$$\begin{array}{rccc} \textit{Création-phase}: & \mathcal{S}\times\mathcal{A}\times\mathcal{C} & \longrightarrow & \mathcal{PH} \\ & (S_i,\mathcal{A}_i,C_i) & \longrightarrow & PH_i \end{array}$$

Le passage d'une phase à une autre sera obtenu en appliquant une fonction *Méta-contrôle* définie par :

$$\begin{array}{rccc} \textit{Méta-contrôle}: & \mathcal{PH} & \longrightarrow & \mathcal{PH} \\ & PH_i & \longrightarrow & PH_{i+1} \end{array}$$

Le contrôle hybride à multi-phases face à ce modèle

Dans ce paragraphe, nous montrons comment ce modèle est implanté dans ATOME. Pour cela, nous faisons une correspondance entre les différents éléments qui interviennent dans le modèle et les composantes de base d'un système-ATOME :

- S_i représente l'état des blackboards au début de la phase PH_i.
- $\mathcal{A}_i$ représente l'ensemble des spécialistes en concurrence durant cette phase.
- C_i représente le mode de raisonnement impliqué durant cette phase.

La fonction *Création-phase* correspond à l'activation d'une tâche. En effet, à partir de l'état des blackboards (S_i), des spécialistes placées sous sa responsabilité ($\mathcal{A}_i$) et de son propre mode de raisonnement (C_i), une tâche gère la résolution du problème durant une phase PH_i.

La fonction *Méta-contrôle* sera, quant à elle, représentée par la stratégie. Cette dernière ordonne l'activation des tâches, faisant ainsi passer le système d'une phase à la phase suivante.

Le nombre total de phases n'est pas connu *a priori*, il est lié au nombre d'activations de tâches et ne sera déterminé que lorsque la stratégie arrêtera le système.

La figure 3.31 dresse un tableau récapitulatif de cette correspondance entre le modèle et son implantation dans ATOME.

S_i	Blackboards
$\mathcal{A}_i$	Spécialistes
C_i	Type de contrôle
Création-phase	Activation d'une tâche
Méta-contrôle	Activation de la stratégie

Figure 3.31. Tableau de correspondance.

3.11 Conclusion

Nous avons montré tout au long de ce chapitre comment les corrections et les extensions effectuées sur ATOME ont corrigé les limites rencontrées à l'étape précédente. Nous avons également défini un modèle hybride à multi-phases qui

traduit le comportement d'un système-ATOME. Cette étape a fait l'objet des publications [166,168,164] et des rapports internes [165,189]. Elle a abouti à la réalisation des versions d'ATOME :

1. Version 2.0 développée en Le_Lisp version 15.2 sur station de travail SUN et opérationnelle depuis janvier 1988 sur toutes les machines où Le_Lisp version 15.2 est disponible.

2. Version 2.0 développée en Common-Lisp sur machine Lisp (Symbolics) et opérationnelle depuis juin 1988. Cette version est dotée d'un interface utilisateur.

3. Version 3.0 développée en Common-Lisp sur machine Lisp (Symbolics) et opérationnelle depuis décembre 1988 [27,28]. Elle est également opérationnelle sur stations de travail SUN et HP9000. Cette dernière est une version révisée et optimisée de la version précédente (2.0/Common-Lisp).

La mise en œuvre des deux dernières versions d'ATOME a été partiellement financée par le GERDSM[16].

Dans la suite, nous faisons une synthèse sur les différentes entités manipulées par ATOME et les opérations qui portent sur ces dernières. Nous traitons ensuite un exemple complet d'utilisation d'ATOME. Les étapes trois et quatre de notre thèse qui consistent à introduire un mécanisme explicatif et à étendre le formalisme de représentation des spécialistes dans ATOME seront décrites après cet exemple. Ces deux étapes n'affectent en rien le type de contrôle d'ATOME. Aussi, nous préférons développer ces différents points avant de faire un bilan complet sur ce dernier.

16. Groupe d'Etude et de Recherche en Détection Sous-Marine (Toulon).

4
Entités manipulées par ATOME

Le but de ce chapitre est de fournir au lecteur un inventaire des différentes entités manipulées par ATOME et des opérations définies sur ces dernières. Nous distinguons les entités de type *niveau*, *hypothèse*, *blackboard*, *résumé d'un blackboard*, *blackboard à long terme*, *spécialiste*, *tâche*, *stratégie*, *règle*, *événement*, *liste d'événements* et *agenda*. Nous terminons ce chapitre par une brève présentation du schéma de traduction que nous avons suivi pour implanter ces entités et opérations.

4.1 Entités et opérations

- *Niveau :*

  Une entité de type niveau est une composante du blackboard permettant de représenter la solution du problème à un certain niveau de détail. Elle est caractérisée par un ensemble de champs de type attribut ou lien. Un attribut d'un élément du niveau peut avoir plusieurs valeurs alternatives provenant de plusieurs sources d'informations et affectées d'un coefficient de vraisemblance numérique ou égal à *indéfini*. Un lien représente une relation entre deux hypothèses d'un même niveau ou de deux niveaux différents et a obligatoirement un lien inverse.

  Les opérations définies sur une entité de type niveau sont :

  - la définition d'un niveau dans un blackboard;
  - la recherche d'éléments dans un niveau d'un blackboard vérifiant des conditions particulières;
  - l'ajout ou le retrait d'une hypothèse dans un niveau d'un blackboard;
  - la définition d'un filtre associé à un niveau d'un blackboard, permettant de caractériser les hypothèses à considérer comme importantes.

- *Hypothèse :*

  Une entité de type hypothèse est caractérisée par son niveau et par les valeurs de ses attributs et liens. Une hypothèse est un élément de la solution du problème.

  Les opérations définies sur le type hypothèse sont :

  - la création d'une nouvelle hypothèse dans un niveau d'un blackboard. Celle-ci est effectuée à partir du nom du niveau auquel cette hypothèse appartiendra ainsi que des valeurs des attributs et des liens à mettre à jour;
  - la lecture et la mise à jour de la valeur d'un attribut ou d'un lien d'une hypothèse. La première opération est effectuée à partir du nom de l'hypothèse et de l'attribut ou du lien à consulter. Pour la seconde opération, il faut en plus préciser les nouvelles valeurs de l'attribut ou du lien considéré;
  - la suppression d'une hypothèse. Celle-ci nécessite comme donnée le nom de l'hypothèse à supprimer;
  - l'accès à une hypothèse. Celle-ci est réalisée à l'aide des fonctions de recherche dans les blackboards ou dans leurs niveaux. Il est également possible d'accéder à une hypothèse grâce à un nom qui lui aurait été affecté lors de sa création;
  - l'évaluation de l'importance d'une hypothèse.

- *Blackboard :*

  Une entité de type blackboard est caractérisée par un ensemble de niveaux. Les blackboards contiennent la solution du problème.

  Les opérations définies sur ce type d'entité sont :

  - la création d'un blackboard;
  - la recherche dans un blackboard d'éléments vérifiant des conditions particulières.

- *Résumé d'un blackboard :*

  Une entité de type résumé d'un blackboard est caractérisée par un ensemble de niveaux (sous-ensemble des niveaux du blackboard associé). Un résumé d'un blackboard contient uniquement les hypothèses importantes de ce dernier. La spécification des conditions sous lesquelles une hypothèse d'un certain niveau du blackboard peut être considérée comme importante relève de l'expertise de l'application en cours de développement avec ATOME.

  Les opérations définies sur ce type d'entité sont :

  - l'ajout ou le retrait d'une hypothèse du résumé d'un blackboard;
  - la recherche dans le résumé d'un blackboard d'éléments vérifiant des conditions particulières.

- *Blackboard à long terme :*

  Une entité de type blackboard à long terme a pour rôle de mémoriser les hypothèses anciennes et non productives du blackboard. Elle est accessible uniquement en lecture et de façon explicite par les spécialistes.

  Les opérations définies sur ce type d'entité sont :

  - la création d'un blackboard à long terme;
  - l'ajout d'une hypothèse dans un blackboard à long terme;
  - la recherche d'éléments dans un blackboard à long terme. Celle-ci permet de retrouver des éléments vérifiant certaines conditions.

- *Spécialiste :*

  Une entité de type spécialiste est une source de connaissances renfermant une partie de l'expertise du domaine. Elle est caractérisée principalement par son nom, sa précondition, ses conditions de rejet et sa partie action.

  Les opérations définies sur ce type d'entité sont :

  - la définition d'une spécialiste;
  - l'activation de la partie action (corps) d'une spécialiste;
  - l'évaluation de la précondition d'une spécialiste;
  - l'évaluation des conditions de rejet d'une spécialiste;
  - l'instanciation d'une spécialiste (cf. entité "instance de spécialiste").

- *Tâche :*

  Une entité de type tâche est une méta-source de connaissances qui fournit un contrôle local au système en gérant un sous-ensemble de spécialistes. Elle est caractérisée principalement par son nom, son mode de raisonnement, son mode de fonctionnement et sa base de règles.

  Les opérations définies sur ce type d'entité sont :

  - la définition d'une tâche;
  - la spécification des types d'événements susceptibles d'intéresser une tâche;

– l'activation d'une tâche.

- *Stratégie :*

  Une entité de type stratégie est une méta-méta-source de connaissances qui fournit un contrôle global au système en gérant l'ensemble des tâches. Elle est caractérisée principalement par son nom, son mode de fonctionnement et sa base de règles.

  Les opérations définies sur ce type d'entité sont :

  – la définition d'une stratégie;

  – l'activation d'une stratégie.

- *Règle :*

  Une entité de type règle est une règle de production caractérisée principalement par sa partie gauche, sa partie droite et son coefficient d'importance. Elle constitue la composante élémentaire d'une source de connaissances de type stratégie, tâche ou spécialiste.

  Les opérations définies sur ce type d'entité sont :

  – l'évaluation de l'importance d'une règle;

  – l'évaluation de la partie gauche d'une règle;

  – l'activation de la partie droite d'une règle.

- *Evénement :*

  Une entité de type événement est un signal envoyé par les spécialistes pour prévenir des mises à jour qu'elles ont effectuées dans le blackboard, à savoir, création d'une nouvelle hypothèse, modification ou suppression d'une ancienne hypothèse. Elle est caractérisée principalement par un niveau d'un blackboard, une hypothèse, un type d'action, une liste d'attributs et de liens mis à jour.

  Les opérations définies sur ce type d'entité sont :

  – la création d'un événement;

  – la suppression d'un événement;

  – la synthèse de deux événements.

- *Liste d'événements :*

  Une entité de type liste d'événements est une structure de contrôle utilisée par les tâches afin de mémoriser les changements du blackboard qui

les intéressent. Chaque tâche a sa propre liste d'événements. Elle est caractérisée par l'ensemble des événements qu'elle contient.

Les opérations définies sur ce type d'entité sont :

– l'insertion d'un événement dans une liste d'événements à une position donnée;

– la suppression d'un événement d'une liste d'événements;

– la recherche d'événements dans une liste d'événements. Cette recherche permet de retrouver des événements vérifiant des conditions particulières.

- *Instance de spécialiste (ISP) :*

  Une entité de type instance de spécialiste représente une activité potentielle d'une spécialiste dans le contexte de l'événement qui l'a réveillée. Une ISP est caractérisée par le nom de la spécialiste et l'événement concernés et par une priorité qui lui est affectée.

  Les opérations définies sur ce type d'entité sont :

  – la création d'une ISP;

  – l'accès à la spécialiste associée à l'ISP;

  – l'accès à l'événement associé à l'ISP;

  – le calcul de la priorité d'une ISP;

  – la suppression d'une ISP.

- *Agenda :*

  Une entité de type agenda est une liste d'attente qui regroupe soit les ISPs réveillées (agenda des ISPs sélectionnées), soit les ISPs activables (agenda des ISPs activables). Ces agendas sont manipulés par les tâches opportunistes. Une entité de type agenda est caractérisée par l'ensemble des ISPs qu'elle contient.

  Les opérations définies sur ce type d'entité sont :

  – la création d'un agenda;

  – le tri des entrées d'un agenda;

  – l'insertion et la suppression d'une entrée dans un agenda.

4.2 Schéma de traduction

Les entités de type niveau, hypothèse, blackboard, résumé d'un blackboard, spécialiste, tâche, stratégie, règle, événement et instance de spécialiste ont été implantées sous forme de tables. En revanche, les entités de type liste d'événements et agenda ont été implantées sous forme de listes chaînées et les entités de type blackboard à long terme sous forme de fichiers.

Les différentes opérations sur ces entités ont été envisagées sous forme de procédures.

L'implantation des différentes versions d'ATOME ne nous a pas posé de problèmes majeurs car nous avions relativement bien spécifié l'architecture adoptée, les structures de données manipulées par le système et les opérations portant sur ces différentes structures.

5
Le problème du voyageur de commerce avec ATOME

Le but de ce chapitre est de décrire un exemple complet d'utilisation d'ATOME. Nous étudierons étape par étape le problème du voyageur de commerce. Cet exemple, qui consiste à trouver le plus petit circuit que devra suivre le voyageur de commerce pour se rendre dans un ensemble de villes en ne passant qu'une fois et une seule dans chaque ville, n'est pas un problème de type blackboard mais constitue un élément de référence dans ce domaine.

Nous souhaitons ici fournir au lecteur un exemple facile à aborder, que l'on puisse traiter entièrement dans ce chapitre, et qui permette de bien comprendre les mécanismes de base d'ATOME et de donner un aperçu de sa syntaxe. Cet exemple nous permettra entre autres de montrer comment améliorer les performances d'un système-ATOME en choisissant un contrôle approprié parmi ceux proposés par ATOME.

Les applications réelles en début de développement avec ATOME sont abordées dans le bilan de nos travaux à la fin de la thèse (cf. chapitre 8).

5.1 Résolution du problème

Il faut dans un premier temps définir les *villes* que le voyageur de commerce doit visiter. Les villes seront ensuite regroupées deux à deux de manière à former des *arcs* non orientés[1]. Chaque arc traduit un passage possible du voyageur d'une ville à une autre. Les arcs représentent les composants élémentaires du *chemin* à trouver. La résolution du problème est effectuée de manière incrémentale : à chaque étape de la résolution, le meilleur arc sera sélectionné et ajouté au chemin solution. Cet arc ne sera pas forcément lié aux arcs déjà présents dans le chemin solution. Le chemin solution est donc construit arc par arc. Les critères permettant de sélectionner le meilleur arc sont basés sur

1. En recherche opérationnelle, on parlerait plutôt d'arête.

deux *heuristiques*. La première consiste à favoriser l'arc le plus court. La seconde consiste à favoriser l'arc le plus extérieur : l'arc qui relie les deux villes les plus éloignées du "barycentre" de toutes les villes. Ces deux heuristiques sont à prendre en compte simultanément dans le choix d'un arc. Un *poids* leur est appliqué afin de moduler leur influence respective. La première heuristique sera affectée d'un poids de trois unités tandis que la seconde d'un poids de deux unités. Ces valeurs ainsi que les heuristiques manipulées par le système sont celles utilisées par BB-1 pour résoudre ce problème. La résolution du problème étant fondée sur des heuristiques ne garantit pas une solution minimale.

Dans la suite, nous présentons l'architecture du système-ATOME résolvant ce problème en définissant ses blackboards, ses spécialistes, ses tâches et sa stratégie. Nous décrivons ensuite le comportement de ce système lors de son exécution. Nous en présentons également une seconde version qui diffère de la précédente par son mode de raisonnement au niveau des tâches. Nous évaluons ensuite ces deux versions en présentant une étude comparative de leurs résultats.

5.2 Définition des blackboards

La première étape dans le développement d'une application avec ATOME est de définir les blackboards qui seront manipulés par le système et leurs niveaux respectifs. En l'occurrence, nous avons défini deux blackboards : le blackboard *tour-info* et le blackboard *solution*.

Dans ATOME, ces blackboards sont définis par :

```
($defblackboards solution tour-info)
```

5.2.1 Tour-info

Ce blackboard contient les données de départ du système, c'est-à-dire, l'ensemble des villes à visiter. Chaque ville est caractérisée par ses coordonnées dans un plan géographique et par son nom. Un champ supplémentaire lui sera associé afin de détecter le nombre de passages du voyageur de commerce dans la ville. Chaque départ d'une ville ou arrivée dans une ville sera comptabilisé comme un passage dans cette ville. Le voyageur ne devant se rendre qu'une fois et une seule dans chaque ville, l'une des conditions pour que le chemin obtenu soit correct est que le nombre de passages dans la ville soit égal à deux : une seule arrivée dans une ville et un seul départ de cette ville sont autorisés.

Le blackboard *tour-info* est constitué d'un seul niveau *villes*. Ce niveau est composé des attributs *nom*, *x*, *y* et *nb-passages* caractérisant chacune des villes.

Il est à remarquer qu'aucun lien n'est défini dans ce blackboard.

Ce niveau est défini avec ATOME par l'appel de la fonction suivante :

```
($defniveau  tour-info.villes  ; Blackboard.niveau
             nom               ; Nom de la ville
             x                 ; Première coordonnée de la ville
             y                 ; Deuxième coordonnée de la ville
             nb-passages)      ; Nombre de passages dans la ville
```

Chaque élément de ce niveau sera représentatif d'une ville et l'ensemble de ces éléments constituera l'ensemble des villes à visiter. Le blackboard *tour-info* contiendra donc les données du problème.

5.2.2 Solution

Le blackboard *solution* contiendra d'une part l'ensemble des arcs reliant les villes deux à deux, et d'autre part, la solution partielle du problème, c'est-à-dire le chemin solution en cours de construction. Le blackboard *solution* est donc composé des deux niveaux *arcs* et *chemin*. Le niveau *arcs* contiendra tous les arcs possibles reliant deux villes tandis que le niveau *chemin* contiendra le chemin solution. Chaque arc sera caractérisé par les deux villes qu'il relie, la distance qui les sépare et son extériorité. Le chemin solution sera caractérisé par les arcs qui le composent et les villes qui ont déjà été visitées, c'est-à-dire les villes qui appartiennent déjà au chemin solution. D'autres informations telles que la liste des villes à visiter seront mémorisées dans ce niveau. Les attributs *de-la-ville* et *à-la-ville* représentent des pointeurs vers une ville du blackboard *tour-info*. L'attribut *chemin* représente un lien avec le niveau *chemin* du blackboard *solution* dont le lien inverse est *arcs*. Le lien *chemin* permet éventuellement de retrouver les arcs, éléments de la solution.

Ces niveaux sont définis avec ATOME par :

```
($defniveau  solution.arcs
             de-la-ville   ; Ville où débute l'arc
             à-la-ville    ; Ville où termine l'arc
             distance      ; Distance entre les deux villes
             extériorité   ; Extériorité de l'arc par rapport à l'ensemble des villes
             chemin)       ; Lien avec le chemin solution
```

```
($defniveau  solution.chemin
             arcs          ; Arcs appartenant au chemin solution
             à-visiter     ; Pointeur vers les villes à visiter
             visitées      ; Villes déjà visitées
             centre-x      ; Coordonnée x du "barycentre" des villes
             centre-y      ; Coordonnée y du "barycentre" des villes
             max-x         ; Valeur maximale de x rencontrée
             max-y         ; Valeur maximale de y rencontrée
             longueur      ; Longueur du chemin solution
             min-x         ; Valeur minimale de x rencontrée
             min-x)        ; Valeur minimale de y rencontrée
```

Les attributs *chemin* et *arcs* de ces deux niveaux sont déclarés en tant que liens par la fonction ATOME suivante :

```
($deflien/inverse arcs chemin solution.chemin solution.arcs)
```

Ce blackboard est évolutif au cours de la résolution du problème. Il représente l'état de la solution courante au problème posé.

Nous avons montré dans cet exemple comment définir deux blackboards. Il aurait été possible de ne définir qu'un seul blackboard contenant à la fois les données du problème et la solution courante. Ce blackboard aurait été composé des trois niveaux : *villes*, *arcs* et *chemin*.

5.3 Les spécialistes

Nous distinguons deux types de spécialistes, celles qui sont utilisées pour initialiser le problème et celles qui résolvent ce problème.

5.3.1 Spécialistes d'initialisation

Nous avons défini deux spécialistes : la première, appelée *Init-villes*, crée l'ensemble des villes à visiter dans le blackboard *tour-info*. La seconde, appelée *Init-résolution*, crée les arcs entre les villes et effectue des calculs préliminaires.

Init-villes

Cette spécialiste n'est utilisée que pour définir les villes à visiter. Elle ne nécessite aucun traitement particulier et aucune condition particulière d'activation. En fait, lorsqu'elle est sollicitée par le mécanisme de contrôle, elle sera toujours capable de créer des villes dans le blackboard *tour-info*. Il est évident que le

mécanisme de contrôle ne l'appellera que lors de la phase d'initialisation du système.

Cette spécialiste n'a ni variable de contrôle, ni variable de contexte, ni variable d'action. Sa précondition est toujours vraie.

Elle est définie dans ATOME par :

```
($defspecialiste  Init-villes  ; Nom de la spécialiste
                  ()           ; Pas de variable de contrôle
                  ()           ; Pas de variable de contexte
                  ()           ; Pas de variable d'action
                  (t)          ; Précondition
                  simple)      ; Mode de fonctionnement
```

Sa partie action est composée d'une seule règle dont la partie gauche est toujours vraie et la partie droite initialise le blackboard *tour-info*. Nous prendrons l'exemple où le voyageur de commerce doit visiter dix villes : Seattle, Los Angeles, Chicago, New York, Miami, Kansas City, Dallas, Las Vegas, Boston et San Francisco. Pour étendre cet exemple à d'autres villes ou changer certaines villes, il suffit de remettre à jour cette spécialiste sans modifier le reste du système.

Cette spécialiste fournit le *nom* de chaque ville ainsi que leur position géographique exprimée sous forme de coordonnées x et y. L'attribut *nb-passages* de chaque ville est initialisé à 0.

Il est à remarquer que les fonctions ATOME manipulent séparément les attributs et les liens des nœuds créés ou mis à jour. La spécialiste n'associe aucune importance relative aux différentes villes créées, ce qui signifie qu'aucun événement signalant la création de ces villes ne sera engendré.

La règle de cette spécialiste est définie avec ATOME par :

```
($defregle  Init-villes                                  ; Nom de la SC
            sp                                           ; Type de la SC : spécialiste
            (t)                                          ; Partie gauche
            (                                            ; Partie droite
             ($proposer-creation  tour-info.villes       ; Création d'une ville dans
                                  ((nom ''Seattle'')     ; le blackboard tour-info
                                   (x 23)                ; au niveau villes
                                   (y 160)               ; Attributs mis à jour
                                   (nb-passages 0))
                                  ()                     ; Liens mis à jour
                                  ())                    ; Pas d'événement
```

```
($proposer-creation  tour-info.villes
                     ((nom ''Los Angeles'')
                      (x 25)
                      (y 45)
                      (nb-passages 0))
                     ()
                     ())

($proposer-creation  tour-info.villes
                     ((nom ''Chicago'')
                      (x 250)
                      (y 125)
                      (nb-passages 0))
                     ()
                     ())
...)
100)                                    ; Importance de la règle
```

La figure 5.1 représente la carte résultante de la création de ces villes en prenant en compte leurs coordonnées respectives.

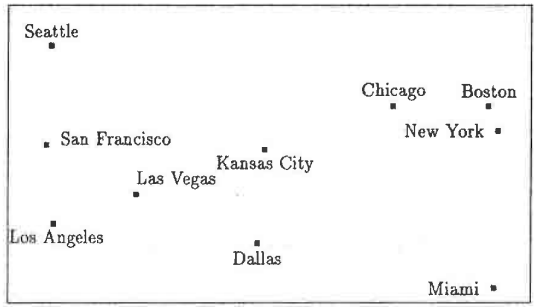


Figure 5.1. Villes à visiter.

Init-résolution

Cette spécialiste initialise le chemin solution qui évoluera durant la résolution du problème et crée l'ensemble des arcs possibles entre ces villes. De même que la spécialiste précédente, elle ne nécessite aucune condition particulière

d'activation.

Elle n'a ni variables de contrôle, ni variables de contexte et sa précondition est toujours vraie. En revanche, elle définit un certain nombre de variables d'action qu'elle manipulera dans ses règles. La plus intéressante dans cet exemple est la variable LISTE-DES-VILLES à laquelle est associé un filtre sur le blackboard *tour-info*. Ce filtre permet de retrouver l'ensemble des villes de ce blackboard. Cette variable sera liée à ces villes lorsque la spécialiste *Init-résolution* évaluera son contexte de travail.

La spécialiste *Init-résolution* est définie à l'aide d'ATOME par l'appel à la fonction suivante :

```
($defspecialiste  Init-résolution                    ; Nom de la spécialiste
                  ()                                 ; Pas de variables de contrôle
                  ()                                 ; Pas de variables de contexte
                  (                                  ; Variables d'action
                   (LISTE-DES-VILLES
                     ($chercher* tour-info.villes))  ; Filtre
                   DISTANCE
                   EXTERIORITE
                   TOTAL
                  )
                  (t)                                ; Précondition
                  multiple)                          ; Mode de fonctionnement
```

La première règle de cette spécialiste initialise le chemin solution en créant un nœud au niveau *chemin* du blackboard *solution*. Ce nœud sera d'ailleurs mémorisé dans la variable MON-CHEMIN dès sa création[2]. L'attribut *à-visiter* de ce nœud se verra affecter la liste des villes à visiter tandis que ses attributs *centre-x*, *centre-y*, *max-x*, *max-y*, *min-x* et *min-y* seront mis à jour à partir d'un certain nombre de calculs. Ces calculs permettent de définir un "barycentre" fictif des villes du blackboard *tour-info* et de détecter les valeurs minimales et maximales des coordonnées de ces villes. Ils ne seront pas détaillés ici; nous signalerons cependant les endroits où ils interviennent.

L'initialisation du chemin solution est en fait réalisée en deux étapes : une création et une mise à jour. La création engendrera un événement car la spécialiste précise son importance relative par **($tete)**, ce qui n'est pas le cas de la mise à jour. Cette distinction est faite car la spécialiste ne veut signaler au mécanisme de contrôle que la mise à jour de l'attribut *à-visiter* du chemin créé. Les autres attributs représentent pour elle des éléments intermédiaires non représentatifs de la solution en elle-même. Après cette initialisation, la

2. Nous verrons plus loin que cette variable est déclarée dans la stratégie en tant que variable globale.

première règle de la spécialiste *Init-résolution* visualise la carte des villes à visiter à l'écran.

Cette règle est décrite avec ATOME par :

```
($defregle  Init-résolution
            sp
            (t)                                    ; Partie gauche
            (                                      ; Partie droite
             ($proposer-creation  solution.chemin  ; Création du chemin so
                                  ((à-visiter
                                    (list LISTE-DES-VILLES
                                          'indéfini)))
                                  ()
                                  ($tete)          ; Importance de l'événem
                                  MON-CHEMIN)      ; Variable

             ($proposer-remplacement      MON-CHEMIN          ; Modification du
                                          ((centre-x (...))   ; Calculs
                                           (centre-y (...))
                                           (min-x (...))
                                           (min-y (...))
                                           (longueur 0)
                                           (max-x (...))
                                           (max-y (...)))
                                          ()
                                          ())              ; Pas d'événemen
             (initialise-affichage-carte)
            )
            100)                          ; Importance de la règle
```

La seconde règle utilise la LISTE-DES-VILLES comme contexte. Elle recherche toutes les paires possibles parmi ces villes et crée les arcs correspondants. Pour chaque arc, la distance entre les villes qu'il relie et l'extériorité de cet arc sont calculés. De même que précédemment, ces calculs ne seront pas détaillés mais seront signalés. Chaque arc créé engendre un événement.

Cette règle est définie par la fonction ATOME suivante :


```
($defregle  Init-résolution
            sp
            (t)
            (
             (do ((DE-LA-VILLE (car LISTE-DES-VILLES)
                               (car AUX-VILLES))
                  (AUX-VILLES (cdr LISTE-DES-VILLES)
                              (cdr AUX-VILLES)))
                 ((null DE-LA-VILLE))
               (dolist (A-LA-VILLE AUX-VILLES)
                 (setq DISTANCE
                       (distance DE-LA-VILLE A-LA-VILLE)
                       EXTERIORITE
                       (exteriorite DE-LA-VILLE
                                    A-LA-VILLE
                                    MON-CHEMIN))
                 ($proposer-creation solution.arcs
                                     ((de-la-ville DE-LA-VILLE)
                                      (à-la-ville A-LA-VILLE)
                                      (distance DISTANCE)
                                      (extériorité EXTERIORITE))
                                     ()
                                     ($tete))))
            )
            100)
```

5.3.2 Spécialistes de résolution

La résolution du problème consiste à construire le chemin arc par arc. Pour ce faire, il est nécessaire de définir une spécialiste dont le rôle est d'ajouter un arc au chemin sans créer de boucle sauf pour le chemin complet. Nous devons donc distinguer deux spécialistes. La première, *Ajouter-un-arc* ajoute un arc au chemin si celui-ci ne conduit pas à une boucle et la seconde, *Ajouter-arc-final*, ajoute le dernier arc.

Ajouter-un-arc

Cette spécialiste travaillera dans le contexte d'un arc donné. Celui-ci lui sera fourni en focalisation de contrôle et sera accessible par l'appel à la fonction ATOME suivante :

```
($noeud-evt-selectionne)
```

Les conditions pour lesquelles cette spécialiste pourra fournir un travail sont les suivantes :

1. Les deux villes composant l'arc que doit ajouter cette spécialiste n'ont pas déjà été visitées (nombre de passages[3] égal à deux).

2. L'arc ne formera pas de boucle dans le chemin.

Ces conditions sont exprimées sous forme de préconditions associées à la spécialiste. Certaines variables de contexte manipulées dans ces conditions seront définies.

Cette spécialiste est caractérisée par des variables de contrôle dont la valeur dépend directement des attributs *distance* et *extériorité* de l'arc à ajouter. Nous verrons l'utilisation de ces variables dans le paragraphe 5.5.

Cette spécialiste est définie dans ATOME par :

```
($defspecialiste  Ajouter-un-arc
                  (                                    ; Variables de contrôle
                   (EXTERIORITE
                    ($valeur extériorité
                             ($noeud-evt-selectionne)))
                   (R-DISTANCE
                    (float
                     (/ 1000.0
                        ($valeur
                           distance
                           ($noeud-evt-selectionne)))))
                  )
                  (                                    ; Variables de contexte
                   (VILLES-VISITEES
                    ($valeur visitées
                             MON-CHEMIN))
                   (ARC
                    ($noeud-evt-selectionne))
                   (DE-LA-VILLE
                    ($valeur de-la-ville ARC))
                   (A-LA-VILLE
                    ($valeur à-la-ville ARC))
                  )
                  ()                                   ; Pas de variables d'action
```

3. Cf. § 5.2.1.


```
                  (                                    ; Précondition
                   (< ($valeur nb-passages
                               DE-LA-VILLE)
                      2)
                   (< ($valeur nb-passages
                               A-LA-VILLE)
                      2)
                   (non-boucle DE-LA-VILLE
                               A-LA-VILLE
                               ($val-lien*
                                  arcs
                                  MON-CHEMIN))
                  )
                  simple)
```

La fonction *non-boucle* utilisée dans la définition de cette spécialiste teste si l'arc à ajouter ne fait pas une boucle avec les arcs déjà présents dans le chemin. Ces derniers sont retrouvés à partir de la relation de type *lien* qui existe entre les arcs et le chemin solution.

La partie action de cette spécialiste est constituée d'une règle qui ajoute l'arc à la solution. Tout ajout d'un arc dans le chemin conduit à la mise à jour des attributs *longueur* et *visitées* du chemin. Un lien est établi entre l'arc et le chemin : le lien *arcs* du nœud MON-CHEMIN est mis à jour, le lien inverse *chemin* de l'arc correspondant sera automatiquement modifié et prendra pour valeur MON-CHEMIN. Les deux villes constituant l'arc verront leur attribut *nb-passages* incrémenté de un. Le chemin en cours d'évolution sera visualisé à l'écran.

Cette règle est représentée dans ATOME par :

```
($defregle  Ajouter-un-arc
            sp
            (ARC)
            (($proposer-remplacement  MON-CHEMIN
                                      ((longueur
                                        (+ ($valeur
                                              longueur
                                              MON-CHEMIN)
                                           ($valeur
                                              distance
                                              ARC)))
```

```
                        (visitées
                           (list
                             (union
                               (list
                                  DE-LA-VILLE
                                  A-LA-VILLE)
                               ($valeur visitées
                                        MON-CHEMIN))
                             'indéfini)))
                        ()
                        ())                                  ; Pas d'événem

($proposer-modification MON-CHEMIN
                        ()
                        ((arcs ARC))  ; Liens mis à jour
                        ())

($proposer-remplacement DE-LA-VILLE
                        ((nb-passages
                           (1+ ($valeur nb-passages
                                        DE-LA-VILLE))))
                        ()
                        ())

($proposer-remplacement A-LA-VILLE
                        ((nb-passages
                           (1+ ($valeur nb-passages
                                        A-LA-VILLE))))
                        ()
                        ())
(afficher MON-CHEMIN))
100)
```

Ajouter-arc-final

Cette spécialiste ajoute le dernier arc au chemin. Son activation requiert un certain état du chemin solution[4] : il ne doit manquer qu'un seul arc à ce chemin. Cette condition est exprimée dans sa précondition. Son rôle est de rechercher les deux villes qu'il reste à lier afin de fermer le chemin en ajoutant l'arc correspondant.

Une variable de contrôle PRIORITE lui sera associée. Cette spécialiste est considérée prioritaire par rapport à tout autre travail envisageable car elle finit le chemin[5].

4. Etat du blackboard *solution*.

5. Le fait qu'elle soit activable signifie que sa précondition est vérifiée et donc que la solution est presque déterminée.

Cette spécialiste est définie avec ATOME par :

```
($defspecialiste  Ajouter-arc-final
                  ((PRIORITE 100))                  ; Variables de contrôle
                  ((NB-VILLES                       ; Variables de contexte
                      (length
                        ($valeur à-visiter
                                 MON-CHEMIN))))
                  (                                 ; Variables d'action
                   (DEUX-VILLES
                     (remove-if
                      #'(lambda (ville)
                          (=
                            (eval
                              '($valeur
                                nb-passages
                                ,ville))
                            2))
                       ($valeur à-visiter
                                MON-CHEMIN)))

                   (DE-LA-VILLE
                     (car DEUX-VILLES))
                   (A-LA-VILLE
                     (cadr DEUX-VILLES))
                   (ARC
                     ($chercher solution.arcs
                                (or ($dans
                                      ($val-attr de-la-ville
                                                 DEUX-VILLES))
                                    ($dans
                                      ($val-attr à-la-ville
                                                 DEUX-VILLES))))))
                  (                                                  ; Précondition
                   (= (1- NB-VILLES)
                      (length ($val-lien*
                                arcs
                                MON-CHEMIN)))
                  )
                  cyclique)
```

La partie action de cette spécialiste est constituée d'une seule règle dont le rôle est d'ajouter l'arc final au chemin. Cette règle est similaire à celle décrite pour la spécialiste *Ajouter-un-arc* excepté pour sa partie gauche qui est toujours vraie (t).

5.4 Les tâches

Deux tâches ont été définies. La première initialise le problème en coordonnant les activités des deux spécialistes d'initialisation *Init-villes* et *Init-résolution*. La seconde gère la résolution du problème proprement dite, à savoir, trouver le chemin. Elle coordonne les activités des spécialistes *Ajouter-un-arc* et *Ajouter-arc-final*.

5.4.1 Tâche d'initialisation

Cette tâche, appelée *Init*, est dirigée par les règles. Ce mode de raisonnement ici n'est pas très représentatif car la tâche n'a qu'une seule règle et ne traite aucun événement : l'initialisation est une phase obligatoire. De ce fait, cette tâche ne demande à être avertie d'aucune action dans les blackboards et ne requiert aucun événement dans sa liste des événements.

Elle est définie par la fonction ATOME suivante :

```
($deftache  Init      ; Nom de la tâche
            ()        ; Pas de variables locales
            ()        ; Pas de Filtre sur les événements
            regles    ; Mode de raisonnement
            simple)   ; Mode de fonctionnement
```

La règle de cette tâche est définie à l'aide d'ATOME par :

```
($defregle  Init              ; Nom de la SC
            ta                ; Type de la SC : tâche
            (t)               ; Partie gauche
            (                 ; Partie droite
             Init-villes
             Init-résolution
            )
            100)              ; Importance
```

Il n'y a pas de réelle compétition entre les deux spécialistes *Init-villes* et *Init-résolution*. Elles sont plutôt coopératives dans la mesure où l'activation de la seconde nécessite l'activation de la première. Elles seront donc activées séquentiellement comme l'indique la règle de la tâche.

5.4.2 Tâche de résolution

Cette tâche, appelée *Construire-chemin*, a un mode de raisonnement opportuniste. Elle a pour rôle de coordonner les travaux et activités potentiels

des spécialistes *Ajouter-un-arc* et *Ajouter-arc-final* afin d'obtenir la solution au problème posé. Ce mode de raisonnement est nécessaire car l'ordre dans lequel les arcs seront ajoutés au chemin n'est pas connu *a priori*. Cette tâche est principalement intéressée par les événements faisant référence aux diverses créations dans le blackboard *solution* aux niveaux *chemin* et *arcs*. Ce sont ces types d'événements qui seront signalés à cette tâche et seront mémorisés dans sa liste d'événements locale.

Cette tâche est définie avec ATOME par :

```
($deftache  Construire-chemin
            ()
            (                                  ; Filtre des événements
             ($est type-action ''creation'')
             ($parmi niveau
                     '(solution.chemin
                       solution.arcs))
            )
            opportuniste)                      ; Mode de raisonnement
                                               ; Pas de mode de fonctionnement
```

La première règle de cette tâche construit tous les travaux ou activités potentiels relatifs à la spécialiste *Ajouter-un-arc*. Elle crée en effet les ISPs liées à cette spécialiste pour chaque événement de la liste des événements locale à la tâche faisant référence à la création d'un arc. La seconde règle crée les ISPs liées à la spécialiste *Ajouter-arc-final* pour chaque événement de la liste des événements locale à la tâche faisant référence à la création d'un chemin solution.

Ces règles sont définies avec ATOME par :

```
($defregle  Construire-chemin
            ta
            (                                  ; Partie gauche
             ($evt-tel-que
              ($est niveau 'solution.arcs))
            )
            (                                  ; Partie droite
             Ajouter-un-arc
            )
            100)
```

```
($defregle  Construire-chemin
            ta
            (                                   ; Partie gauche
             ($evt-tel-que
              ($est niveau 'solution.chemin))
            )
            (                                   ; Partie droite
             Ajouter-arc-final
            )
            100)
```

5.5 La stratégie

La stratégie doit organiser la résolution du problème. En fait, le problème du voyageur de commerce n'est pas suffisamment complexe pour faire intervenir ce niveau de stratégie dans le raisonnement. Cependant, pour montrer une utilisation possible de cette méta-source de contrôle et des résumés des blackboards, nous avons défini une stratégie simple. Elle consiste à demander l'initialisation du système en appelant la tâche *Init* si l'exécution débute, et à lancer la résolution du problème en appelant la tâche *Construire-chemin* s'il y a effectivement des villes à visiter. La stratégie arrête la résolution du problème lorsque toutes les villes à visiter ont été visitées.

Seul, un résumé du blackboard *solution* sera nécessaire à la stratégie. Ce résumé devra contenir le chemin solution si l'ensemble des villes à visiter n'est pas vide. Pour ce faire, nous caractérisons les hypothèses (nœuds) importantes du blackboard *solution* comme étant les hypothèses du niveau *chemin* pour lesquelles il existe encore des villes à visiter.

Ceci est exprimé avec ATOME par :

```
($defnoeuds-importants solution.chemin

    (not (null ($val-attr à-visiter))))
```

La stratégie est définie avec ATOME par :

```
($defstrategie  Stratégie       ; Nom de la stratégie
                (MON-CHEMIN)    ; Variable locale à l'application
                cyclique)       ; Mode de fonctionnement
```

La stratégie est composée de trois règles de production. La première initialise le système, la seconde a pour rôle de construire le chemin et la dernière arrête l'exécution du système. Elles sont définies avec ATOME par :


```
($defregle  Stratégie                   ; Nom de la SC
            st                          ; Type de la SC
            (                           ; Partie gauche
             (equal ($cycle-courant) 0) ; si début d'exécution
            )
            (                           ; Partie droite
             Init
            )
            100)
```

```
($defregle  Stratégie
            st
            (
             ($au-niveau solution.chemin)  ; Résumé non vide
            )
            (
             (Construire-chemin .  R1)     ; R1 règle d'intégration
            )
            100)
```

```
($defregle  Stratégie
            st
            (
             ($au-niveau solution.chemin
                         ($egal ($val-attr visitées)
                                ($val-attr à-visiter)))
            )
            (
             ($quitter-st)
            )
            100)
```

La figure 5.2 décrit l'architecture globale du système-ATOME ainsi développé.

La règle d'intégration R1 fournie à la tâche *Construire-chemin* par la stratégie fait intervenir les deux heuristiques que nous avons présentées au début de ce chapitre, à savoir H1 favorisant les arcs les plus courts et H2 favorisant les arcs les plus extérieurs. Elle fait également intervenir une troisième heuristique H3 qui favorise les spécialistes les plus prioritaires. Ces heuristiques sont définies en fonction des variables de contrôle R-DISTANCE, EXTERIORITE et PRIORITE par les fonctions ATOME suivantes :

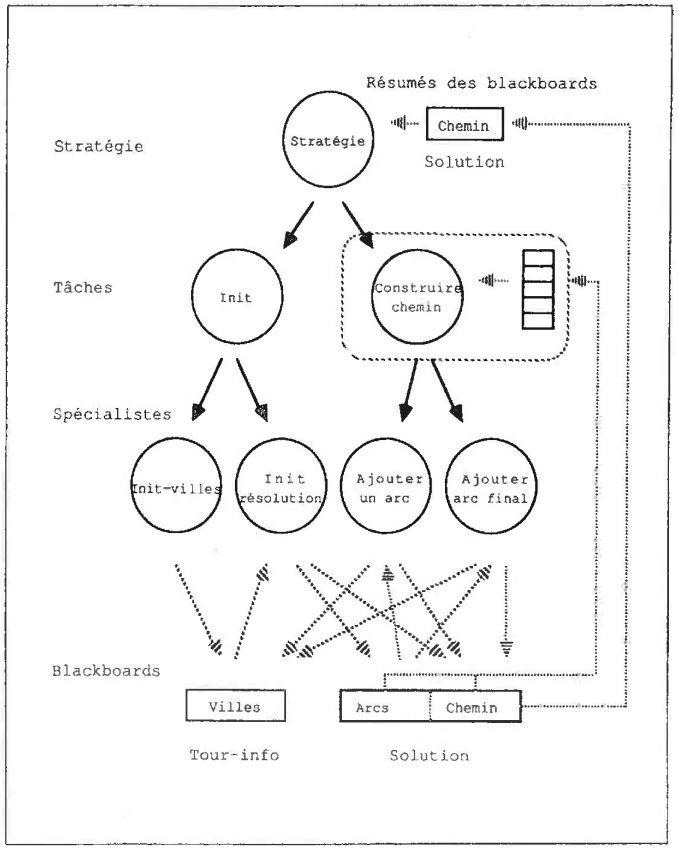


Figure 5.2. Architecture du système-ATOME résolvant le problème du voyageur de commerce.


```
($defheuristique  H1                ; Nom de l'heuristique
                  stable
                  (R-DISTANCE 1))   ; Couple variable-poids

($defheuristique  H2
                  stable
                  (EXTERIORITE 1))

($defheuristique  H3
                  stable
                  (PRIORITE 1))
```

R1 est définie par :

```
($defregle-integration  R1                ; Nom de la règle d'intégration
                        ((H1 3) (H2 2))   ; Première combinaison
                        ((H3 1)))         ; Deuxième combinaison
```

5.6 Exécution et première amélioration

5.6.1 Exécution

L'exécution du système commence par l'activation de la stratégie. Cette dernière déclenche en l'occurrence sa première règle et donne le contrôle à la tâche *Init*. La règle de cette tâche est déclenchée et active séquentiellement les deux spécialistes *Init-villes* et *Init-résolution*.

Init-villes initialise le blackboard *tour-info* en créant les dix villes dans son niveau *villes*.

Init-résolution initialise le chemin solution en créant le nœud MON-CHEMIN au niveau *chemin* du blackboard *solution*. Elle crée également les quarante cinq arcs possibles dans le niveau *arcs* du blackboard *solution*. Ces créations engendrent des événements qui sont envoyés à la tâche *Construire-chemin*. Les dix villes étant à visiter, MON-CHEMIN est mémorisé dans le résumé du blackboard *solution*.

Le contrôle retourne ensuite à la tâche *Init* qui le redonne à la stratégie. La stratégie active alors la tâche *Construire-chemin* en lui donnant R1 comme règle d'intégration.

Cette tâche va parcourir sa liste des événements locale et créera autant

d'ISPs que d'événements satisfaisant ses prémisses de règles. Ainsi, quarante six ISPs seront créées : une illustrant un travail potentiel de la spécialiste *Ajouter-arc-final* avec l'événement associé à la création du chemin solution; les autres associant la spécialiste *Ajouter-un-arc* à chaque création d'un arc.

Au départ, l'agenda des ISPs sélectionnés contient l'ISP qui consiste à ajouter l'arc final tandis que l'agenda des ISPs exécutables contient une offre d'ajout pour chaque arc dans le chemin. Le calcul des priorités permet de sélectionner l'ISP à activer et d'ajouter le meilleur arc. Au fur et à mesure de la résolution du problème, les ISPs associées à des arcs bouclant le chemin ou sollicitant un passage de trop dans une ville passe de l'agenda des ISPs exécutables vers l'agenda des ISPs sélectionnées[6]. Ces ISPs resteront alors dans l'agenda des ISPs sélectionnées jusqu'à la fin de la résolution du problème. Etant donné qu'il n'y a pas de retour-arrière dans le raisonnement, les arcs ajoutés au chemin solution le sont une fois pour toute et ne sont jamais remis en cause, tout arc qui conduit à une boucle dans le chemin et à un passage de trop dans une ville, bouclera le chemin ou sollicitera un passage de trop dans la ville jusqu'à la fin de l'exécution.

D'autre part, l'ISP associée à la spécialiste *Ajouter-arc-final* du dernier arc deviendra activable et passera dans l'agenda des ISPs exécutables lorsque les neuf premiers arcs auront été sélectionnés et ajoutés au chemin. Les figures 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12 et 5.13 illustrent l'exécution. La dernière figure fournit le chemin trouvé.

Il est important de remarquer que les heuristiques intervenant dans les calculs de priorité des ISPs sont stables et ne seront évaluées qu'une seule fois par ISP.

6. Leur précondition n'est plus vérifiée.

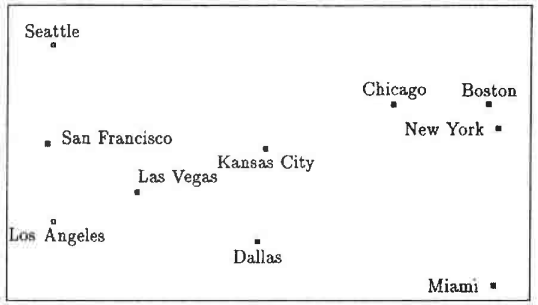


Figure 5.3. Etat initial.

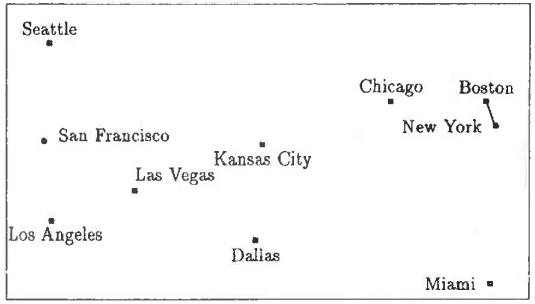


Figure 5.4. Arc New York - Boston.

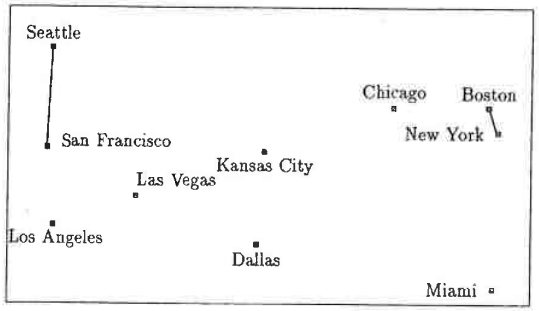


Figure 5.5. Arc Seattle - San Francisco.

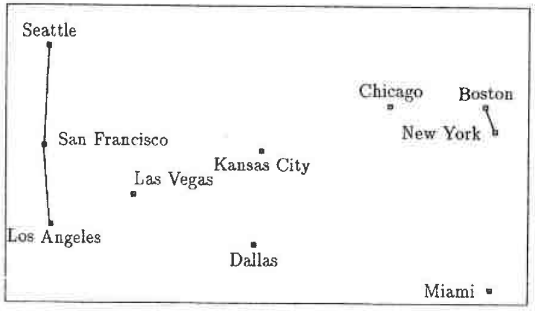


Figure 5.6. Arc Los Angeles - San Francisco.

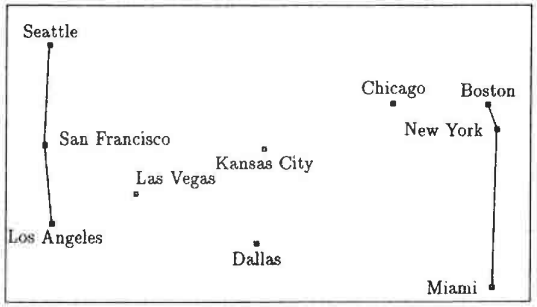


Figure 5.7. Arc New York - Miami.

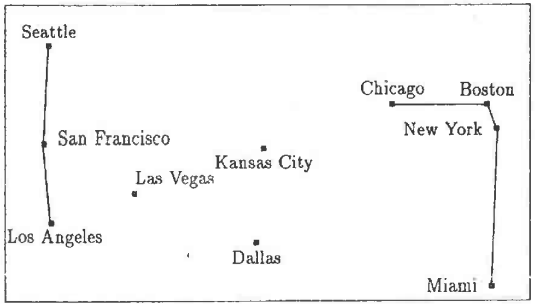


Figure 5.8. Arc Chicago - Boston.

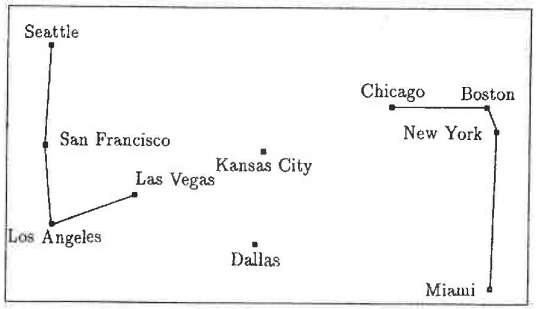


Figure 5.9. Arc Los Angeles - Las Vegas.

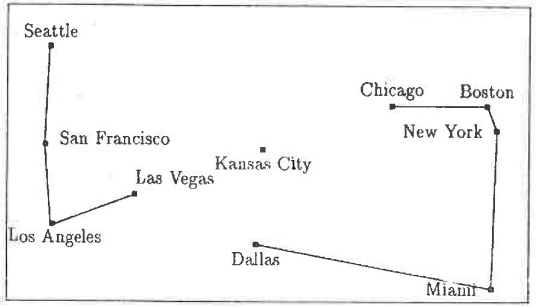


Figure 5.10. Arc Miami - Dallas.

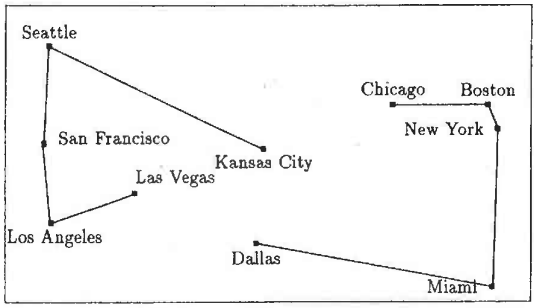


Figure 5.11. Arc Seattle - Kansas City.

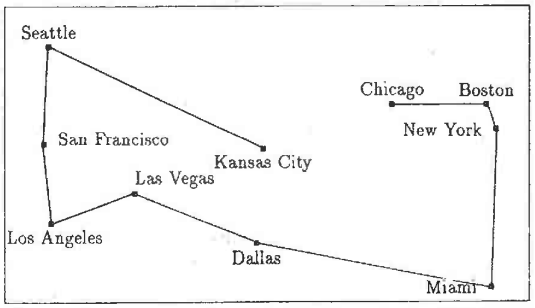


Figure 5.12. Arc Dallas - Las Vegas.

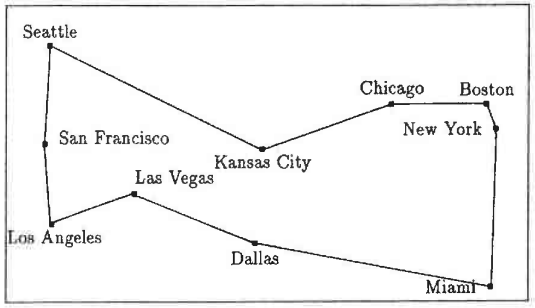


Figure 5.13. Arc Chicago - Kansas City.

5.6.2 Apport des conditions de rejet

Nous avons vu que, lors de l'exécution du système, les ISPs associées à des arcs bouclant le chemin ou sollicitant un passage de trop dans une ville sont placés définitivement dans l'agenda des ISPs sélectionnées. Cependant, les préconditions de ces ISPs seront testées à chaque cycle local de la tâche lors de la phase de mise à jour des agendas. Aussi, il est plus intéressant d'ajouter des conditions de rejet à la spécialiste *Ajouter-un-arc* qui permettraient de rejeter définitivement ces ISPs des agendas lorsqu'elles sont vérifiées. Ceci limiterait considérablement les parcours dans les agendas. Nous avons effectué cette extension en spécifiant les conditions de rejet suivantes :

1. L'une des deux villes composant l'arc que doit ajouter cette ISP a déjà été visitée (nombre de passages égal à deux).

2. Ou l'arc forme une boucle dans le chemin.

5.7 Extensions

L'utilisation du mode opportuniste dans la tâche *Construire-chemin* nous a permis de mettre au point une première version du voyageur de commerce. Cependant, elle conduit à des parcours des agendas à chaque fois qu'un arc est ajouté au chemin solution pour détecter les ISPs sélectionnées, activables ou

rejetées. Nous allons montrer dans cette partie comment le *mode dirigé par les événements* peut améliorer les performances du système.

Etant donné que les heuristiques liées au choix d'un arc à ajouter (indépendamment de l'arc final) sont directement calculées à partir d'attributs de cet arc et sont non évolutives[7], il nous a semblé intéressant de faire intervenir ces heuristiques dans le classement des événements associés à la création des arcs dans la liste locale des événements de la tâche *Construire-chemin*.

Ainsi, ces événements seraient préalablement classés dans l'ordre où les arcs devraient être testés pour être ajoutés au chemin.

Afin de permettre un tel classement, il faut effectuer certains changements dont les plus importants sont décrits dans la suite.

5.7.1 Les blackboards

Nous avons ajouté un attribut *total* dans le niveau *arc* du blackboard *solution* qui mémorisera le résultat de l'application de la première combinaison de la règle d'intégration R1, à savoir `((H1 3) (H2 2))`.

5.7.2 Spécialiste Init-résolution

Cette spécialiste envoyait systématiquement les événements associés à la création d'un arc en tête des listes des événements des tâches. Cette importance relative des événements est changée. En effet, sa valeur **`($tete)`** précisée dans chaque appel de la fonction ATOME **`($proposer-creation solution.arcs...)`** sera remplacé par :

```
($avant-evt-tel-que
    (< ($valeur total ($noeud-evt-courant)) TOTAL))
```

où TOTAL est une variable d'action à ajouter à la spécialiste contenant, pour l'arc créé, la valeur de son attribut total.

Cette importance relative signifie que l'événement engendré sera placé avant le premier des événements se référant à un arc moins prioritaire que l'arc qui vient d'être créé.

Ainsi, dans chaque liste des événements où cet événement sera propagé, en l'occurrence la liste des événements locale à la tâche *Construire-chemin*, les événements relatifs à la création des arcs seront classés par importance décroissante. L'événement le plus important correspondant au premier arc à ajouter au chemin.

7. Valeur constante pour un arc donné.

5.7.3 Tâche Construire-chemin

Cette tâche est réécrite en mode dirigé par les événements avec un mode de fonctionnement cyclique-multiple. Les seuls événements dont elle aura besoin sont ceux relatifs à la création des arcs. Son filtre sur les événements a donc changé également.

Cette tâche est redéfinie avec ATOME par :

```
($deftache  Construire-chemin
            ()
            (                                   ; Filtre des événements
             ($est type-action ''creation'')
             ($est niveau 'solution.arcs)
            )
            evenements                          ; Mode de raisonnement
            cyclique-multiple)                  ; Mode de fonctionnement
```

Elle sera composée d'une seule règle qui consiste à demander l'activation séquentielle des spécialistes *Ajouter-un-arc* et *Ajouter-arc-final* si l'événement le plus important concerne le niveau *arcs* du blackboard *solution*, sachant qu'une spécialiste dont la précondition n'est pas vérifiée est ignorée.

Cette règle est définie avec ATOME par :

```
($defregle  Construire-chemin
            ta
            (                                   ; Partie gauche
             ($evt-tel-que
              ($est niveau 'solution.arcs))
            )
            (                                   ; Partie droite
             Ajouter-un-arc
             Ajouter-arc-final
            )
            100)
```

5.7.4 La stratégie

Lorsqu'elle appelle la tâche *Construire-chemin*, la stratégie n'a plus à lui fournir la règle d'intégration R1.

5.7.5 Le comportement

Seules les créations d'arcs sont signalées à la tâche *Construire-chemin*. Sa liste des événements n'est composée que des événements associés à ces créations

classés par ordre de priorité décroissante. Ainsi, l'événement le plus important correspond à l'arc le plus prioritaire à traiter en premier. Le mode dirigé par les événements permet à la tâche de traiter ces événements un par un en fonction de leur importance.

A chaque activation de règles, la tâche sollicite les spécialistes *Ajouter-un-arc* et *Ajouter-arc-final*. Les préconditions de ces spécialistes ne sont jamais satisfaites en même temps. Durant l'ajout des huit premiers arcs, seule la spécialiste *Ajouter-un-arc* est activée. En revanche, l'ajout de l'avant-dernier arc par cette spécialiste valide la précondition de la spécialiste *Ajouter-arc-final*. Dans ce cas, les deux spécialistes *Ajouter-un-arc* et *Ajouter-arc-final* sont activées séquentiellement.

Au fur et à mesure que les événements sont traités dans la liste des événements, le chemin est construit. La spécialiste *Ajouter-arc-final* arrêtera l'exécution en appelant la fonction **($quitter-st)**.

5.8 Résultats

Des mesures ont été faites afin de comparer les résultats obtenus pour les deux modes de raisonnement de la tâche *Construire-chemin*, à savoir opportuniste ou dirigé par les événements.

Les mesures suivantes ont été effectuées sur la version Common-Lisp/Symbolics d'ATOME - v3.0 [28,27]. Les premiers résultats obtenus avec la version Common-Lisp/Symbolics d'ATOME - v2.0 sont décrits dans [29].

5.8.1 Mode opportuniste

`Temps d'exécution :`

Version compilée : 6.8 secondes.
Version interprétée : 14.9 secondes.

`Nombre de nœuds créés dans les blackboards : 56.`

- *Solution* :

  Niveaux : *chemin* : 1.
  arcs : 45.

- *Tour-info* :

  Niveaux : *villes* : 10.

Nombre de nœuds propagés vers les résumés des blackboards : 1.

- *Solution* :

  Niveaux : *chemin* : 1.

Nombre de règles activées des SCs du domaine : 13.

Spécialistes	:	*Init-villes*	: 1.
		Init-résolution	: 2.
		Ajouter-un-arc	: 9.
		Ajouter-arc-final	: 1.

Nombre de règles activées des SCs du contrôle : 50.

Tâches	:	*Init*	: 1.
		Construire-chemin	: 46.
Stratégie	:	*Stratégie*	: 3.

Nombre d'événements créés : 46.

Spécialistes : *Init-résolution* : 46.

Nombre d'événements propagés : 46.

Vers les tâches : *Construire-chemin* : 46.

Nombre d'événements synthétisés : 0.

Nombre d'événements utilisés : 46.

Dans les tâches : *Construire-chemin* : 46.

Nombre d'ISPs créées : 46.

Spécialistes	:	*Ajouter-un-arc*	: 45.
		Ajouter-arc-final	: 1.

Nombre d'ISPs activées : 10.

Spécialistes	:	*Ajouter-un-arc*	: 9.
		Ajouter-arc-final	: 1.

Nombre d'activations des spécialistes : 12.

Init-villes	:	1.
Init-résolution	:	1.
Ajouter-un-arc	:	9 (sous forme d'ISP).
Ajouter-arc-final	:	1 (sous forme d'ISP).

5.8.2 Mode dirigé par les événements

Temps d'exécution :

Version compilée	:	3.8 secondes.
Version interprétée	:	7.0 secondes.

Nombre de nœuds créés dans les blackboards : 56.

- *Solution* :

  Niveaux : *chemin* : 1.
  arcs : 45.

- *Tour-info* :

  Niveaux : *villes* : 10.

Nombre de nœuds propagés vers les résumés des blackboards : 1.

- *Solution* :

  Niveaux : *chemin* : 1.

Nombre de règles activées des SCs du domaine : 13.

Spécialistes	:	*Init-villes*	: 1.
		Init-résolution	: 2.
		Ajouter-un-arc	: 9.
		Ajouter-arc-final	: 1.

Nombre de règles activées des SCs du contrôle : 19.

Tâches	:	*Init*	: 1.
		Construire-chemin	: 16.
Stratégie	:	*Stratégie*	: 2.

Nombre d'événements créés : 46.

Spécialistes : *Init-résolution* : 46.

Nombre d'événements propagés : 45.

Vers les tâches : *Construire-chemin* : 45.

Nombre d'événements synthétisés : 0.

Nombre d'événements utilisés : 16.

Dans les tâches : *Construire-chemin* : 16.

`Nombre d'activations des spécialistes : 12.`

Init-villes : 1.
Init-résolution : 1.
Ajouter-un-arc : 9.
Ajouter-arc-final : 1.

5.8.3 Comparaison

Cette étude confirme l'efficacité du mode de raisonnement dirigé par les événements par rapport au mode opportuniste : 3.8 vs 6.8 secondes en version compilée, et 7.0 vs 14.9 secondes en version interprétée[8].

Le mode opportuniste dépense la plupart de son temps d'exécution dans la création d'ISPs, dans la vérification des conditions de rejet et des préconditions des ISPs et dans les calculs de leur priorité.

Les deux modes créent le même nombre de nœuds dans les blackboards (dont un seul est mémorisé dans les résumés), engendrent le même nombre d'événements, provoquent le même nombre d'activations des spécialistes et déclenchent le même nombre de règles de ces dernières.

La différence réside principalement dans le nombre de règles déclenchées des SCs de contrôle : 19 vs 49. La tâche opportuniste active 46 règles contre 16 règles activées lorsque la tâche est dirigée par les événements. Tous les événements de la liste d'événements de la tâche opportuniste déclenchent une règle pour créer une ISP d'où les 46 ISPs. En revanche, seuls les 16 plus importants événements sont utilisés lorsque le mode de raisonnement est dirigé par les événements.

Parmi les 46 ISPs créés par la tâche opportuniste, seulement 10 sont activées.

Le nombre d'événements propagés vers les tâches est similaire pour les deux types de tâches : 45 vs 46.

5.9 Conclusion

Cet exemple a montré, d'une part, une utilisation d'ATOME et, d'autre part, comment le mode opportuniste peut être utilisé pour les premières étapes de mise au point d'une application. Les changements effectués sur les spécialistes pour changer de mode de raisonnement de la tâche dans un mode dirigé par les

8. D'après [100], BB-1 consomme environ 30 secondes pour créer les premiers ISCs lors du premier cycle de résolution du problème.

événements concernent essentiellement la position des événements à engendrer.

Il a également mis en évidence l'efficacité du mode de raisonnement dirigé par les événements par rapport au mode opportuniste. En effet, le mode dirigé par les événements ne nécessite pas un parcours de la liste des événements mais prend en considération leur importance respective. Le mode opportuniste quant à lui offre plus de souplesse mais ceci au dépend de l'efficacité. En effet, la liste des événements est régulièrement parcourue afin de rechercher tous les travaux potentiels du système et d'en sélectionner les meilleurs. Les règles de contrôle sont plus souvent sollicitées dans ce mode que celles du mode dirigé par les événements[9].

La tâche opportuniste fait intervenir des heuristiques définies explicitement tandis que la tâche dirigée par les événements les manipulent de façon implicite par l'intermédiaire de la position des événements.

Dans les deux cas, le problème a été résolu en deux phases : une phase d'initialisation matérialisée par l'activation de la tâche *Init* et une phase de construction du chemin solution matérialisée par l'activation de la tâche *Construire-chemin*.

9. La tâche opportuniste prend son temps pour réfléchir !!

6
Le mécanisme explicatif dans ATOME

La puissance d'un système d'IA ne se limite pas aux connaissances qu'il renferme ni à la manière dont ces connaissances sont organisées et exploitées. Elle est également liée à la capacité du système à expliquer son raisonnement, aussi bien au développeur qu'à l'utilisateur final. Le développeur sera intéressé par des explications sur le comportement du système et sur ses déductions afin d'évaluer les résultats obtenus et d'affiner ou de réorganiser les connaissances du système. L'utilisateur final, quant à lui, aura surtout besoin d'explications liées directement au domaine d'application et exprimées dans un vocabulaire qui lui est familier. Pour être complet, le mécanisme explicatif dans un système d'IA doit prendre en compte ces deux aspects.

Les systèmes à base de blackboard n'échappent pas à cette règle : en fait le mécanisme explicatif est encore plus justifié pour eux que pour tout autre système d'IA "classique" [15,244], étant donné la diversité des sources de connaissances, et la complexité des types de raisonnement et de contrôle qu'ils utilisent.

La troisième étape de notre thèse consiste à introduire dans ATOME un tel mécanisme. Dans ce chapitre, nous définissons les principales entités que le mécanisme explicatif doit prendre en compte dans toute architecture de blackboard. Nous décrirons ensuite l'approche que nous avons adoptée dans ATOME. Ce dernier étant un outil de mise au point de systèmes d'IA, nous nous sommes limités à l'étude d'un mécanisme explicatif pour le développeur d'une application.

6.1 Entités de base d'une architecture de blackboard

Les entités fondamentales dans une architecture de blackboard sont : les *hypothèses* du blackboard, les *sources de connaissances*, les *modules de contrôle*.

les *structures de contrôle* et les *événements*.

Toutes ces entités interagissent au cours de l'exécution du système. En effet, les *sources de connaissances* mettent à jour les *hypothèses* du blackboard en engendrant des *événements*. Ces derniers sont stockés dans des *structures de contrôle*. En fonction du contenu de ces structures, des *modules de contrôle* choisissent des actions à exécuter et le cycle recommence. Ces interactions établissent des relations entre les entités, comme l'indique la figure 6.1.

Aussi, pour être complet, un mécanisme explicatif dans une architecture de blackboard, doit être capable de :

1. *Prendre en compte ces entités ainsi que les relations entre elles.*

   Lors de l'exécution du système, le mécanisme explicatif doit être capable de reconnaître les différentes entités qui sont intervenues dans la résolution du problème. Il doit en effet détecter et repérer les hypothèses émises dans le blackboard, les événements engendrés, l'évolution de l'état des structures de contrôle ainsi que les activités des sources de connaissances et des modules de contrôle. Il doit également définir les relations entre ces entités en suivant les interactions qui apparaissent au cours de l'évolution du système.

2. *Les représenter sous une forme appropriée.*

   Le mécanisme explicatif ayant alors connaissance des différentes entités et relations, qui lui sont nécessaires pour retrouver ce qui s'est passé durant l'exécution du système, doit disposer d'une ou plusieurs structures de données adaptées afin de mémoriser ces informations.

3. *Restituer toutes les informations qui en découlent.*

   A partir de cette ou de ces structures de données, le mécanisme doit pouvoir reconstituer un historique complet et détaillé de l'exécution du système. Il doit également être capable de restituer les informations plus élémentaires concernant la participation des différentes entités à l'exécution du système.

Dans la suite, nous traitons le cas particulier d'ATOME. Pour ce faire, nous allons décrire plus précisément les différentes entités fondamentales à prendre en compte dans ATOME ainsi que les relations qui existent entre elles. Nous étudierons également la structure de données retenue pour stocker les informations explicatives engendrées durant l'exécution d'une application, ainsi que les moyens utilisés pour restituer ces informations au développeur de l'application.

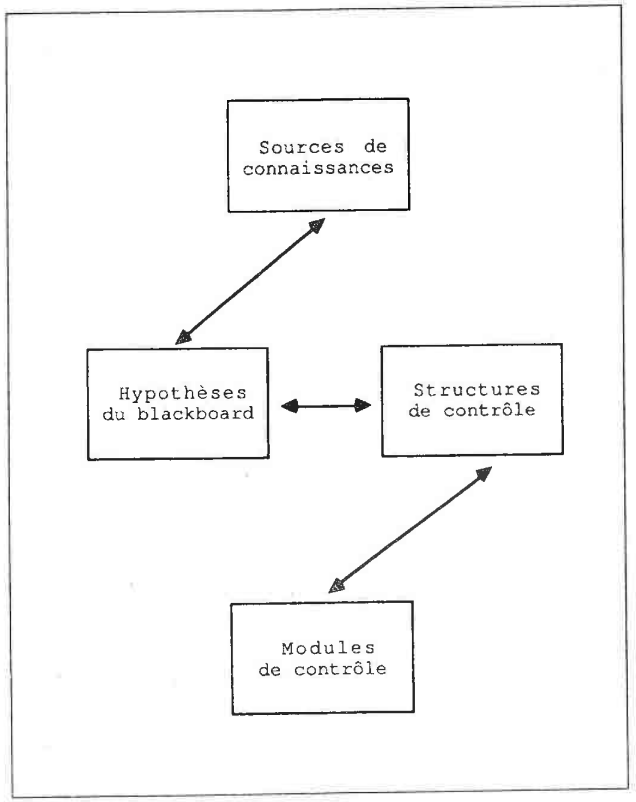


Figure 6.1. Relations entre les entités fondamentales d'une architecture de blackboard.

6.2 Le mécanisme explicatif dans ATOME

Le mécanisme explicatif dans ATOME (cf. figure 6.2) est composé d'une structure de données stockant les informations liées aux différentes entités décrites précédemment sous forme d'un blackboard appelé *blackboard explicatif* et d'un ensemble de transactions que l'utilisateur[1] peut formuler en tant que requêtes au blackboard explicatif. Le module chargé de traiter ces transactions est appelé *module explicatif*.

6.2.1 Le blackboard explicatif

Il y a quatre points essentiels à souligner dans le fonctionnement d'ATOME pour comprendre l'origine de la structure de blackboard retenue pour le stockage des informations explicatives :

1. Le déclenchement d'une règle de la stratégie est conditionné par l'existence de certaines hypothèses dans les résumés des blackboards, et se traduit par l'activation d'une séquence de tâches (cf. figure 6.3).

2. L'activation d'une tâche dirigée par les événements ou par les règles entraîne le déclenchement d'un certain nombre de règles. Ces déclenchements sont conditionnés par l'existence d'événements donnés, et conduisent à l'activation de séquences de spécialistes (cf. figure 6.4).

3. L'activation d'une tâche opportuniste ne se traduit pas directement par des déclenchements de règles, mais par la création d'instances de spécialistes (cf. figure 6.5).

4. L'activation d'une spécialiste entraîne le déclenchement d'un certain nombre de règles. Ces déclenchements sont conditionnés par l'état de certaines hypothèses dans les blackboards, et se traduisent par la production d'événements. (cf. figure 6.6).

Il apparaît de façon évidente que chacune des informations explicatives à stocker se rapporte plus particulièrement à l'une des cinq entités fondamentales d'ATOME : la stratégie, les tâches, les spécialistes, les événements et les hypothèses. Il devient donc relativement naturel de songer à représenter l'ensemble des informations explicatives du système par un blackboard divisé en cinq niveaux, représentant chacun l'une de ces entités fondamentales (cf. figure 6.7). Chacune des entrées de ce blackboard modélise en fait un granule d'information

1. Dans notre cas, l'utilisateur est le développeur d'un système d'IA à l'aide d'ATOME.

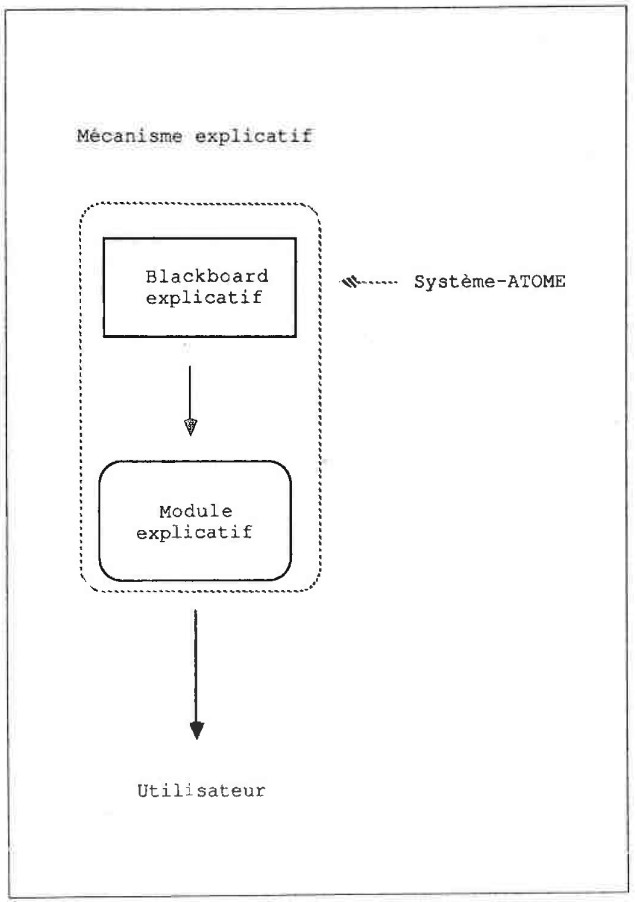


Figure 6.2. Mécanisme explicatif dans ATOME.

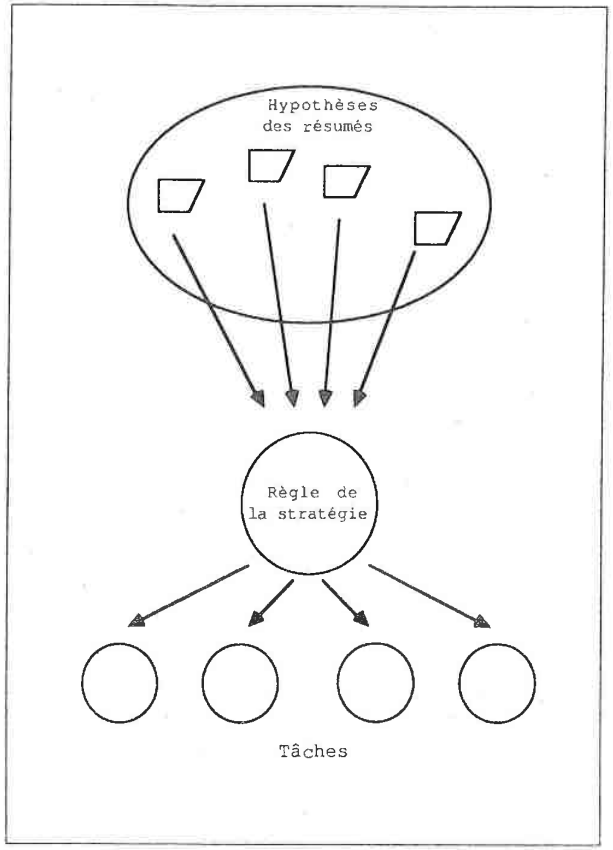


Figure 6.3. Relations entre la stratégie, les résumés des blackboards et les tâches.

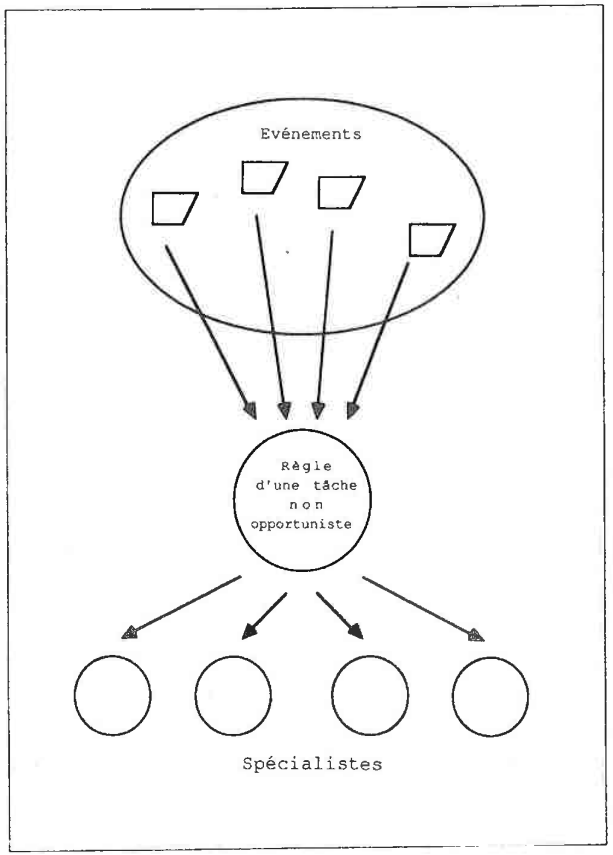


Figure 6.4. Relations entre une tâche dirigée par les événements ou par les règles, sa liste d'événements et les spécialistes.

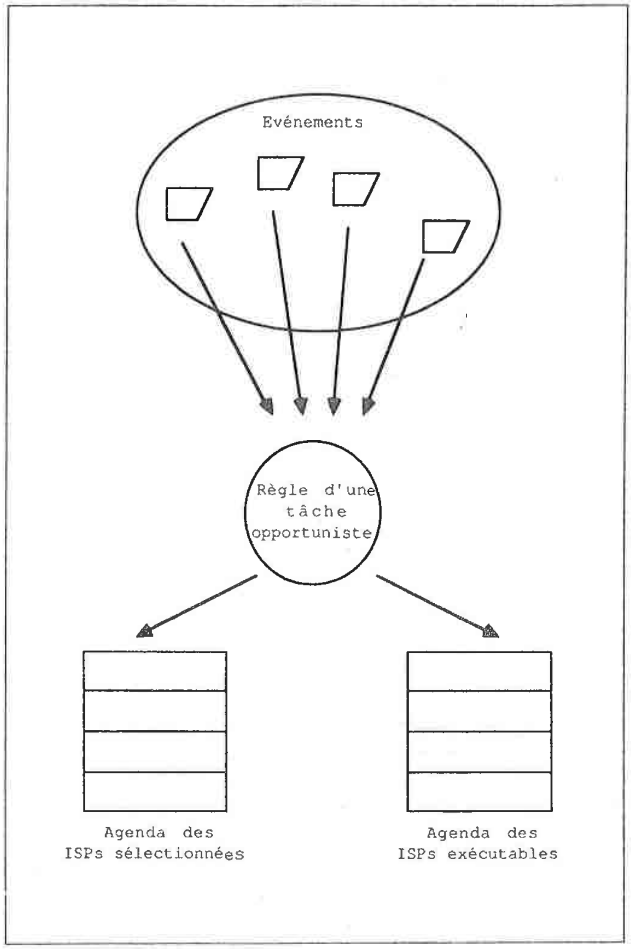


Figure 6.5. Relations entre une tâche opportuniste, sa liste d'événements et les agendas des instances de spécialistes.

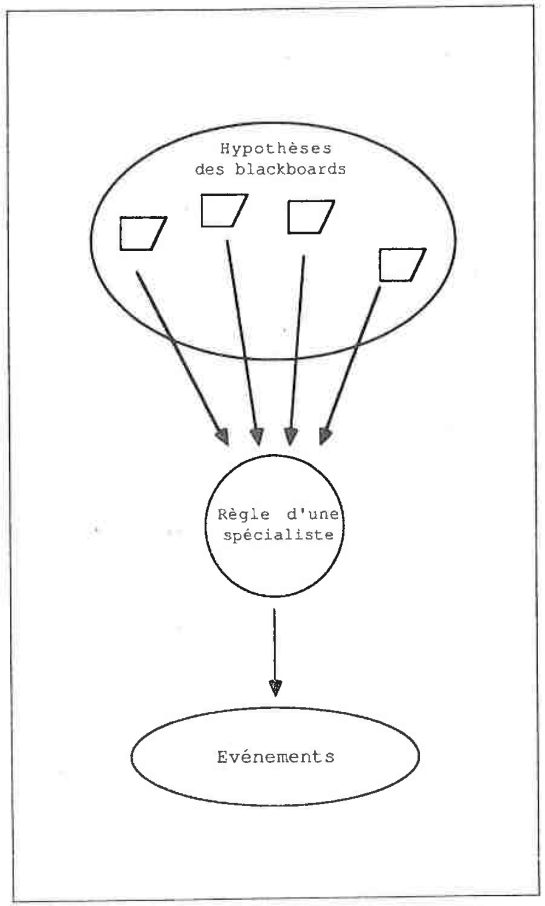


Figure 6.6. Relations entre une spécialiste, les blackboards et les événements.

explicative associé à l'entité fondamentale correspondante. Les relations entre ces entités fondamentales seront quant à elles schématisées par des liens entre les niveaux de ce blackboard.

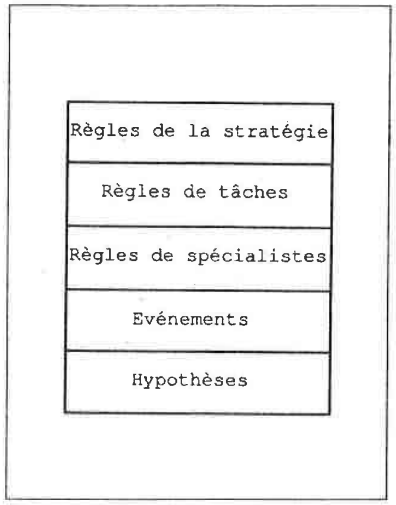


Figure 6.7. Blackboard explicatif à cinq niveaux.

Après implantation et test de cette structure, elle est apparue insuffisante pour représenter correctement l'historique d'une exécution du système. En effet, elle ne permet pas de :

- mémoriser certains types de faits, en particulier l'activation de sources de connaissances du domaine ou de contrôle dont aucune règle n'a été déclenchée;
- détecter simplement certains comportements d'exception. Par exemple, les activations successives de la même source de connaissances sont assimilées à une unique activation.

Ces limites justifient une réorganisation complète du blackboard explicatif. Celui-ci compte actuellement huit niveaux (cf. figure 6.8) :

1. Au niveau *stratégie* figurent les activations de stratégie (il n'y en n'a qu'une par exécution) caractérisées par leur nom, et des liens vers les règles déclenchées.
2. Au niveau *règles de la stratégie* figurent les déclenchements des règles de stratégie caractérisés par leur nom, leur cycle, des liens vers la stratégie mère, les hypothèses des résumés des blackboards ayant satisfait leurs prémisses, et les tâches activées.
3. Au niveau *tâches* figurent les activations de tâches caractérisées par leur nom, des liens vers la règle de la stratégie mère, et vers les règles déclenchées (si la tâche est dirigée par les événements ou par les règles) ou sollicitées (si la tâche est opportuniste).
4. Au niveau *règles de tâches* figurent les déclenchements de règles de tâches caractérisés par leur nom, des liens vers la tâche mère, vers les événements ayant satisfait leurs prémisses, et vers les spécialistes activées.
5. Au niveau *spécialistes* figurent les activations de spécialistes caractérisées par leur nom, leur priorité, l'hypothèse leur servant de contexte et ISP associés (si l'activation a été faite sous le contrôle d'une tâche opportuniste), des liens vers la règle de la tâche mère, vers les hypothèses ayant satisfait leurs préconditions, et vers les règles déclenchées.
6. Au niveau *règles de spécialistes* figurent les déclenchements de règles de spécialistes caractérisés par leur nom, des liens vers la spécialiste mère, vers les hypothèses ayant satisfait leur prémisse, et vers les hypothèses créées, modifiées ou supprimées.
7. Au niveau *hypothèses* figurent les hypothèses ayant transité dans les blackboards caractérisées par leur nom, leur niveau, leur blackboard, leurs cycles de présence dans le résumé du blackboard correspondant, des liens vers les règles de spécialistes qui les ont manipulées, vers les spécialistes dont elles ont satisfait les préconditions, et vers les règles de la stratégie ou de spécialistes dont elles ont satisfait les prémisses.
8. Au niveau *événements* figurent les événements créés lors du déroulement du système caractérisés par leur nom, leur type d'action (création, modification, remplacement ou suppression), les attributs et liens manipulés, des liens vers la règle de la spécialiste les ayant engendrés, vers l'hypothèse associée, vers les tâches auxquelles ils ont été acheminés, vers les règles de tâches dont ils ont satisfait les prémisses.

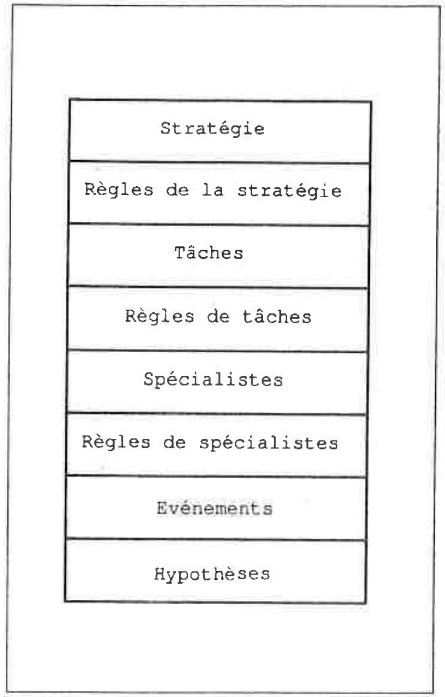


Figure 6.8. Le blackboard explicatif dans ATOME.

Les manières d'exploiter cette structure sont nombreuses : menus ou commandes interprétées pour accéder aux différents granules d'information, reconstitution chronologique de la consultation... Le type d'informations que restitue ATOME et la manière dont le module explicatif retrouve ces informations sont décrits ci-dessous (cf. § 6.2.2).

6.2.2 Le module explicatif

Dans ATOME, le module explicatif se charge d'exécuter les commandes effectuées par l'utilisateur en vue d'obtenir des informations relatives à l'évolution d'une entité (par exemple la stratégie), un composant d'une entité (par exemple une règle de la stratégie), ou l'historique de la dernière exécution du système. Pour ce faire, le module explicatif accède au niveau correspondant du blackboard explicatif et parcourt une partie de ce blackboard lorsque les informations requises concernent une entité ou une composante de cette entité. Il a également la possibilité de parcourir tout le blackboard explicatif pour restituer un historique d'exécution.

Si, par exemple, la transaction formulée correspond à la recherche d'informations concernant un événement donné, alors en accédant au niveau *événements* du blackboard explicatif, le module explicatif est capable de restituer :

- Le type d'action que représente cet événement — création, modification, remplacement, ou suppression.
- L'hypothèse et le blackboard sur laquelle porte l'action.
- Les attributs et liens de l'hypothèse qui ont été mis a jour.
- La spécialiste qui a engendré cet événement.
- Les tâches vers lesquelles cet événement a été acheminé.
- Les règles de tâches activées avec cet événement, et par conséquent les spécialistes déclenchées avec ce dernier comme contexte.

L'exemple de la figure 6.9 fait référence au problème du voyageur de commerce développé au chapitre 5. Il illustre la création d'un événement. La figure 6.10 décrit son utilisation. Ces informations sont retrouvées à partir des différents liens établis entre les entités fondamentales du blackboard explicatif.

Le module explicatif procédera de la même façon si l'information correspond à la recherche de l'historique d'une entité, d'une composante de celle-ci ou du système complet.

```
STRATEGIE Stratégie REGLE règle-1
TACHE Init REGLE règle-1
SPECIALISTE Init-résolution REGLE règle-2
CYCLE 1


CREATION DE L'EVENEMENT evt-2
PORTANT SUR LA CREATION DU NOEUD solution.arcs-1
AYANT MIS EN JEU LES ATTRIBUTS

   de-la-ville
   à-la-ville
   distance
   extériorité

PROPAGE VERS LA TACHE

   Construire-chemin
```

Figure 6.9. Création d'un événement.


```
UTILISATION DE L'EVENEMENT evt-2


UTILISE PAR LA TACHE Construire-chemin
POUR DECLENCHER SA REGLE règle-1

 ET

CREER L'ISP isp-45
ASSOCIEE A LA SPECIALISTE Ajouter-un-arc


AU COURS DU CYCLE cycle-2
```

Figure 6.10. Utilisation d'un événement.

6.2.3 Architecture résultante d'ATOME

L'architecture globale d'ATOME définie dans le chapitre précédent évolue comme l'indique la figure 6.11. Lorsqu'un système développé avec ATOME est en cours d'exécution, le *module explicatif* détecte, repère et mémorise dans le *blackboard explicatif* toutes les informations explicatives associées aux entités de base du système. A la fin de l'exécution et à la demande du développeur, le module explicatif restituera ces dernières en parcourant le graphe que constitue le contenu du blackboard explicatif.

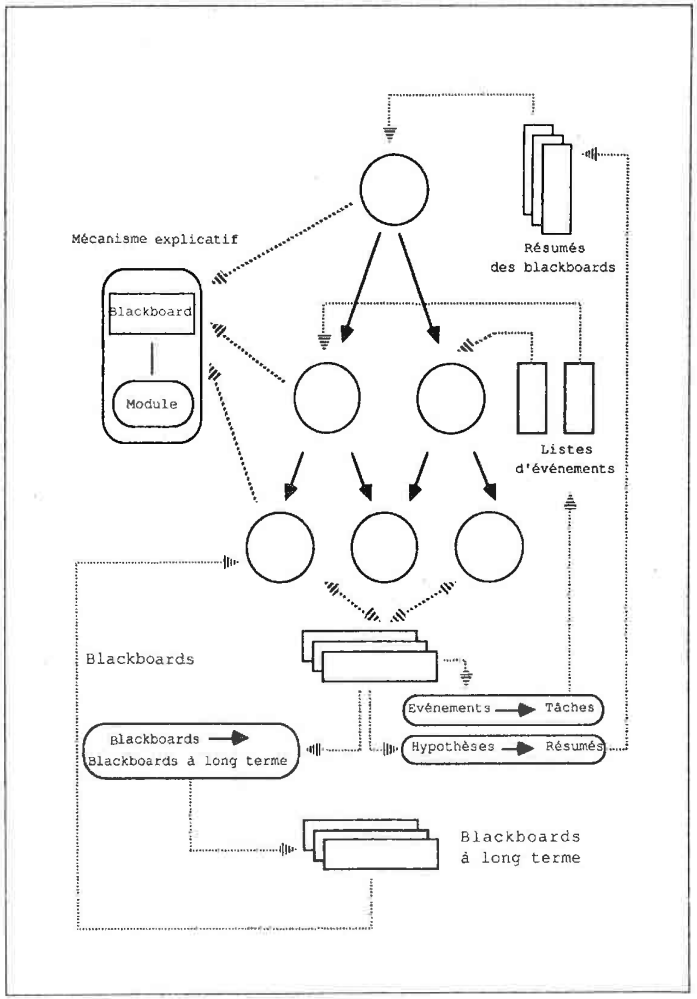


Figure 6.11. Architecture d'ATOME.

6.3 Conclusion

Le blackboard explicatif constitue un support puissant pour tout mécanisme d'explications dans les architectures de blackboard. Néanmoins, une structure aussi complexe soulève des problèmes d'occupation de mémoire et d'efficacité. En effet, le blackboard explicatif contient autant d'événements que le système a engendré, autant sinon plus d'hypothèses que le blackboard des données, etc. Par conséquent, un mécanisme explicatif utilisant une structure de blackboard pour stocker les informations explicatives se trouve être un grand consommateur de mémoire. Durant l'exécution du système, le blackboard explicatif est mis à jour à chaque fois qu'une action est faite sur une entité ou sur l'une de ses composantes. Par conséquent, le mécanisme explicatif se trouve aussi être un grand consommateur de temps d'exécution.

Il reste malgré tout utile sinon nécessaire pour des systèmes de mise au point tel qu'ATOME. C'est un moyen puissant d'aide au développement d'une application d'IA qui permet une communication homme-machine à la fois sophistiquée et souple durant la mise au point de cette application. Il permet en effet au développeur, suite à une exécution du système, d'en retracer la chronologie, ou d'isoler la participation de chacun des constituants de l'application.

Il est intéressant de pouvoir le désactiver dans des systèmes finaux où généralement la rapidité dans le temps de réponse est un objectif primordial.

Cette étape a fait l'objet d'un stage de DESS IA chez Cognitech [143]. Elle a donné lieu à deux versions d'ATOME :

1. Version 2.1 sans interface graphique, développée en Le_Lisp version 15.2 sur station de travail SUN et opérationnelle depuis juin 1988;
2. Version 2.1 avec interface graphique, développée en Le_Lisp version 15.2 sur station de travail SUN et opérationnelle depuis juillet 1988. L'interface graphique a été mise en œuvre à l'aide de l'interface virtuelle de CEWB : atelier de développement industriel de systèmes experts [38].

7

Représentation hétérogène des spécialistes dans ATOME

En IA, nombreux sont les chercheurs, concepteurs ou utilisateurs qui sentent le besoin d'intégrer dans la même architecture, d'une part, des systèmes procéduraux conventionnels et, d'autre part, des systèmes basés sur une connaissance déclarative. En effet, s'il est justifié d'utiliser des langages déclaratifs pour traiter des données symboliques, une solution algorithmique est souvent plus appropriée au traitement des données brutes.

Un cas typique de problèmes nécessitant de telles architectures est celui de l'interprétation de formes complexes : parole, images, ou signaux acoustiques. En effet, ce type de problèmes requiert une coopération des techniques de reconnaissances des formes et d'IA, et combine par conséquent des traitements numériques — algorithmes de segmentation et paramétrisation des données brutes, et d'apprentissage des formes modèles —, avec un raisonnement basé sur des connaissances.

Durant la quatrième étape de notre thèse, nous nous sommes intéressés à cet aspect dans ATOME dans le souci de permettre la coopération de sources d'informations hétérogènes. A titre expérimental, nous nous sommes intéressé au cas où les spécialistes peuvent être soit des programmes C [153], soit des systèmes à base de connaissances respectant le formalisme d'ATOME (ordre 0+) (cf. chapitres 2 et 3 de cette partie), le formalisme de SAGANE (ordre 0) [11], ou le formalisme d'IROISE (ordre 1) [107].

Nous commençons ce chapitre par une description du principe de base que nous avons défini pour permettre la mise en œuvre d'une coopération de SCs hétérogènes dans une architecture de blackboard. Nous présentons ensuite la réalisation effectuée qui consiste à appliquer ce principe de base dans le cadre d'ATOME afin d'enrichir le formalisme d'expression de ses spécialistes. Pour cela, nous avons réalisé une communication entre ATOME, les environnements de développement de systèmes experts — SAGANE et IROISE, et le langage C.

7.1 Principe

Dans cette section, nous définissons ce que nous entendons par *architecture homogène* et *architecture hétérogène* dans un système fondé sur le modèle du blackboard. Nous présentons ensuite pour ces deux types d'architectures les principes selon lesquels les SCs peuvent accéder au blackboard et être exécutées. Nous définirons plus particulièrement les principes que nous avons introduits pour permettre la communication entre les SCs d'une architecture hétérogène.

- *Architecture homogène :*

  Dans une architecture homogène, toutes les SCs ont la même structure et respectent le même formalisme. La représentation interne des objets du blackboard et les fonctions d'accès en lecture et écriture à ces objets font partie du langage de ces SCs. En plus de son rôle de zone de communication, le blackboard constitue la mémoire de travail des SCs : il contient les données initiales, les résultats intermédiaires et finaux de chacune d'elles.

  L'activation d'une SC dans ce type d'architecture peut être résumée en deux étapes :

  1. *Création du contexte de travail :*

     La SC commence par créer son contexte de travail en filtrant sur le blackboard le type de données qui l'intéressent en entrée.

  2. *Exécution du corps de la* SC *:*

     A partir des données mémorisées dans son contexte de travail, la SC engendre des résultats dans le blackboard (cf. figure 7.1). Ces résultats correspondent à des créations de nouvelles hypothèses, des modifications et/ou des suppressions d'anciennes hypothèses du blackboard. La mise à jour du blackboard est effectuée tout au long de l'exécution de la SC. De ce fait, le blackboard sert de mémoire de travail à la SC.

- *Architecture hétérogène :*

  Dans une architecture hétérogène, les SCs n'ont pas forcément la même structure. En effet, elles peuvent être de nature diverse — procédurale ou déclarative —, et/ou écrites dans des langages différents. Par conséquent, la structure interne des objets du blackboard et les fonctions d'accès en lecture et écriture à ces objets ne sont pas obligatoirement compatibles avec le langage de chaque SC.

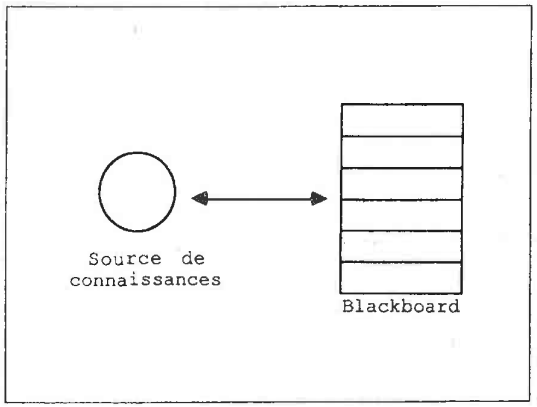


Figure 7.1. Accès au blackboard dans une architecture homogène.

Pour permettre une communication entre plusieurs SCs hétérogènes à travers le blackboard tout en garantissant leur indépendance, nous avons choisi l'approche suivante :

- Les SCs dont le langage est compatible avec la représentation interne des objets du blackboard et avec les fonctions d'accès à ces objets auront le même comportement que les SCs d'une architecture homogène.

- En revanche, les SCs dont le langage n'est pas compatible avec la représentation interne des objets du blackboard et avec les fonctions d'accès à ces objets auront accès au blackboard de façon globale et uniquement *avant* et *après* exécution. Elles disposeront d'une *mémoire de travail locale* (éventuellement exprimée sous forme de base de faits) pour engendrer et mémoriser leurs résultats durant leur exécution. Elles se verront également associer un *module d'entrée* et un *module de sortie* définis à l'aide de *fonctions génériques* leur permettant de transposer les informations du blackboard dans leur langage et réciproquement (cf. figure 7.2).

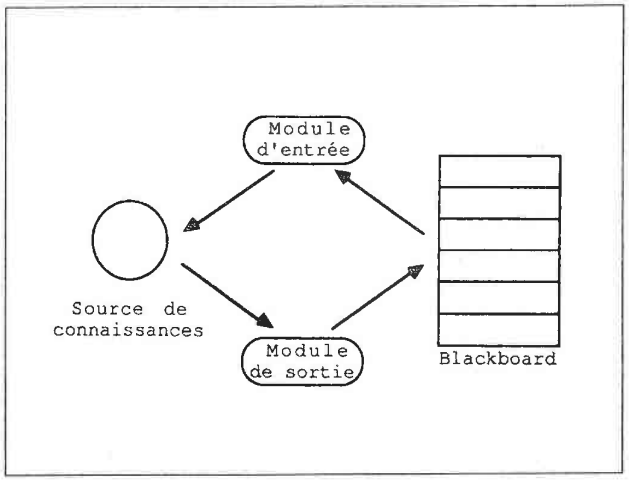


Figure 7.2. Accès au blackboard dans une architecture hétérogène.

Dans le premier cas, le blackboard a un rôle multiple : il sert à la fois de zone de communication et de mémoire de travail aux SCs. L'activation des SCs est alors composée des deux étapes définies précédemment pour une architecture homogène.

Dans le second cas, le blackboard ne sert que de zone de communication. L'activation d'une SC est alors divisée en trois étapes :

1. *Création du contexte de travail :*

   Le module d'entrée de la SC commence par créer le contexte de travail de cette dernière en filtrant sur le blackboard le type de données qui l'intéressent et en effectuant les transformations nécessaires sur ces données pour rendre leur structure interne compatible avec le langage de la SC.

   Il les stocke ensuite dans la mémoire de travail locale de la SC.

2. *Exécution du corps de la* SC *:*

   L'exécution de la SC conduit à une mise à jour de sa mémoire de travail locale. Aucun accès au blackboard n'est effectué durant cette exécution.

3. *Transformation des résultats :*

   Une partie de la mémoire de travail de la SC — généralement les résultats finaux — va être transformée par le module de sortie de la SC afin de rendre la structure des objets engendrés homogène avec celle du blackboard et les stocker dans ce dernier.

L'architecture d'ATOME telle que nous l'avons décrite dans les chapitres précédents est homogène : chaque spécialiste est exprimée dans un même formalisme d'ordre 0+[1]. Dans le but d'étendre ATOME vers une architecture hétérogène, nous avons appliqué le principe décrit précédemment.

7.2 Réalisations

Nous avons intégré dans ATOME les outils d'aide au développement de systèmes experts SAGANE d'ordre 0 et IROISE d'ordre 1 ainsi que le langage C. Ces trois formalismes ont été choisi d'une part, parce qu'ils complétaient le formalisme proposé par ATOME et d'autre part, parce qu'une expérience acquise sur ces systèmes a facilité considérablement cette intégration [186]. Une spécialiste dans ATOME peut donc être développée selon le formalisme-ATOME, à l'aide de SAGANE ou IROISE, ou en C selon le besoin.

Pour réaliser la communication entre les spécialistes d'un système-ATOME, plusieurs primitives ont été définies afin de permettre :

- La définition d'une spécialiste en précisant son type : ATOME, SAGANE, IROISE ou C[2].
- La récupération d'objets d'un blackboard pour les stocker dans la mémoire de travail d'une spécialiste de type SAGANE, IROISE ou C.
- La transformation de la représentation interne des objets des blackboards, en une représentation compatible avec le langage d'une spécialiste de type SAGANE, IROISE ou C.
- Le stockage d'objets dans un blackboard à partir de la mémoire de travail d'une spécialiste de type SAGANE, IROISE ou C. Ce stockage peut éventuellement engendrer des événements.

1. Nous désignerons par la suite ce formalisme par *formalisme*-ATOME.
2. ATOME étant lui-même écrit en Lisp, la définition d'une spécialiste sous forme de programme Lisp ne nécessite aucun mécanisme spécifique.

- La transformation de la représentation interne des objets de la mémoire de travail d'une spécialiste de type SAGANE, IROISE ou C en une représentation compatible avec la structure des blackboards.

Par exemple, lorsque la spécialiste est de type SAGANE, elle définit les *paramètres* destinés à la récupération et à la transformation des objets du blackboard. Pour ce faire, elle leur associe un filtre sur le blackboard qui lui donne accès à la valeur d'un attribut d'une hypothèse [9] (cf. figure 7.3).

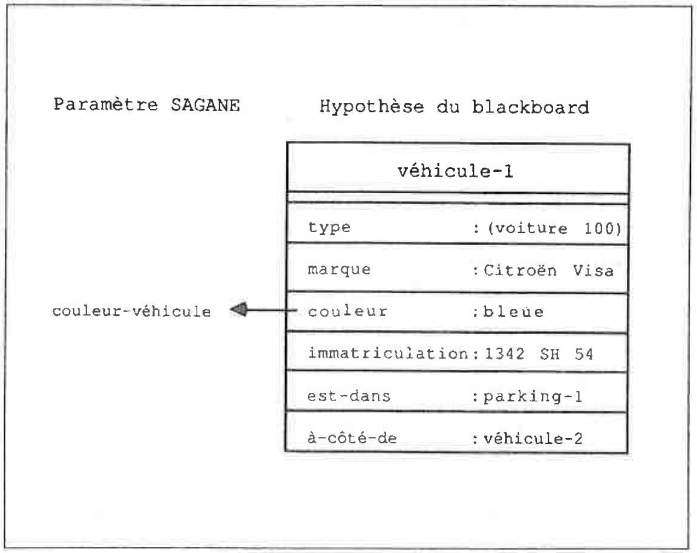


Figure 7.3. Récupération et transformation d'objets du blackboard.

Inversement, il est possible de récupérer les valeurs de plusieurs paramètres de cette spécialiste et de les affecter à des attributs d'hypothèses du blackboard (cf. figure7.4).

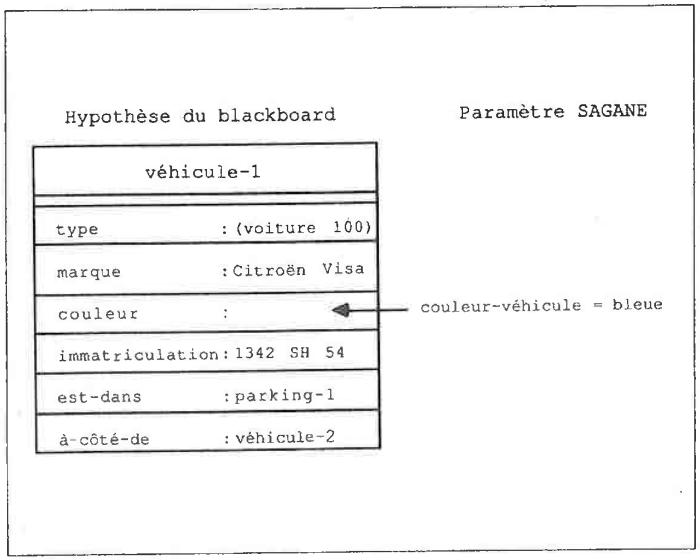


Figure 7.4. Transformation et stockage de paramètres SAGANE dans le blackboard.

La figure 7.5 représente la structure d'une spécialiste d'ATOME de type SAGANE, IROISE ou C, en appliquant le principe que nous avons défini pour la communication de SCs dans une architecture hétérogène. La structure d'une spécialiste respectant le formalisme-ATOME sera la même que celle décrite dans le chapitre 3. L'architecture résultante de la connexion d'ATOME avec ces formalismes est donnée en figure 7.6.

1. *Nom :* identification de la spécialiste.
2. *Type :* ATOME, SAGANE, IROISE, ou C.
3. *Variables locales :* leur principal rôle est de permettre la spécification des filtres de la spécialiste. Différentes fonctions de filtrage sont associées à chaque variable. Cet ensemble de variables est divisé en deux parties :
   (a) *Variables de précondition :*
   Ces variables sont liées lors de la phase de vérification de la précondition de la spécialiste et restent accessibles lors de sa phase d'activation. Elles sont donc manipulées dans sa précondition ou dans sa partie action.
   (b) *Variables d'action :*
   Ces variables sont liées lors de la phase d'activation de la spécialiste et sont manipulées dans sa partie action. Dans le cas où la spécialiste est de type SAGANE, IROISE ou C, ces variables contiennent les résultats des transformations effectuées par le module d'entrée de la spécialiste sur les hypothèses du blackboard filtrées.
4. *Module d'entrée :* ce module n'est défini que pour les spécialistes de type SAGANE, IROISE ou C. Il est intégré dans la définition des variables d'action de la spécialiste.
5. *Précondition :* liste de conditions à vérifier par la spécialiste pour devenir *activable.*
6. *Action :* corps de la spécialiste. Il peut être sous forme déclarative (ATOME, SAGANE ou IROISE), soit sous forme procédurale (programme C).
7. *Module de sortie :* ce module n'est défini que pour les spécialistes de type SAGANE, IROISE ou C. Il permet de convertir les résultats de la spécialiste en hypothèses du blackboard en engendrant éventuellement des événements.
8. *Mode de fonctionnement :* cette propriété n'est définie que pour une spécialiste de type ATOME. Elle permet de préciser le mode d'interprétation de sa base de règles - simple, multiple ou cyclique.

Figure 7.5. Structure d'une spécialiste hétérogène dans ATOME.

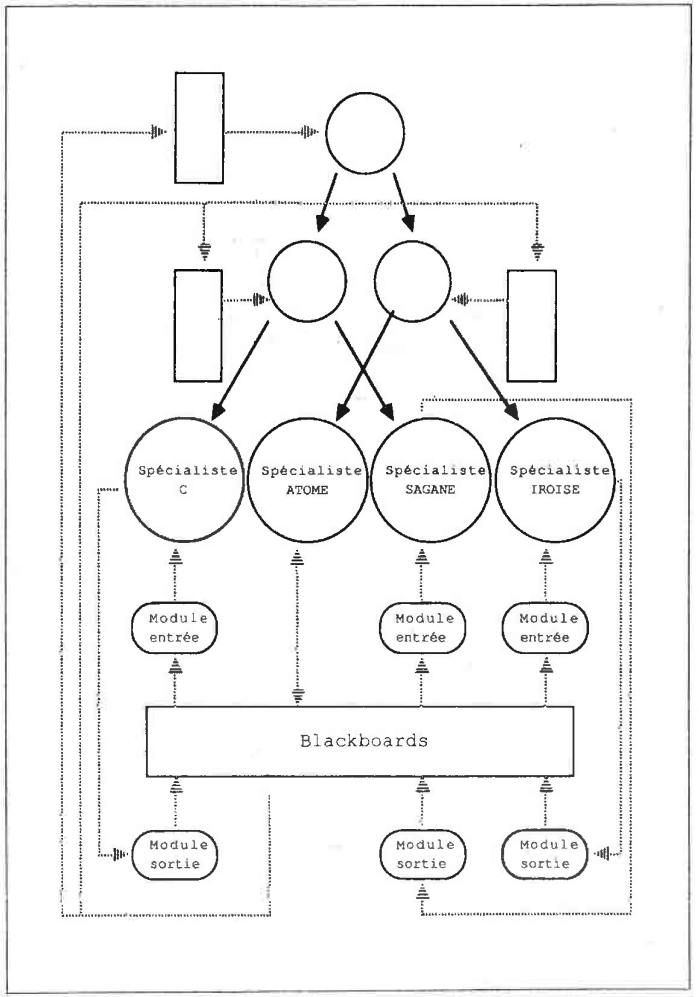


Figure 7.6. Architecture hétérogène d'ATOME.

7.3 Conclusion

Nous terminons ce chapitre par une présentation des nouvelles caractéristiques d'ATOME qui résultent de l'enrichissement du formalisme d'expression de ses spécialistes et par une présentation des difficultés rencontrées.

- *L'architecture d'*ATOME *est hétérogène :*

  ATOME fait partie des outils d'IA permettant de faire coopérer des sources d'informations qu'elles soient de type procédural (spécialiste C), ou de type déclaratif (spécialiste ATOME, SAGANE ou IROISE). Le type déclaratif regroupe différents formalismes de représentation des connaissances d'ordres 0, 0+ et 1. Cette architecture permet à tout utilisateur d'ATOME de coder les sources de connaissances dans le formalisme le plus adapté à ses besoins. Une spécialiste développée à l'aide de SAGANE ou IROISE, ou dans le langage C accède aux blackboards en début et en fin d'exécution, c'est-à-dire de façon globale. En revanche, une spécialiste écrite dans le formalisme-ATOME travaille essentiellement sur les blackboards.

- ATOME *est un système ouvert :*

  D'autres outils peuvent être facilement intégrés à ATOME. Pour cela, il suffit de s'inspirer des primitives ayant permis sa communication avec SAGANE, IROISE et C.

  Inversement, si une application ne nécessite pas l'intégration de tous les formalismes d'expression de spécialistes proposés par ATOME, il est possible de sélectionner uniquement les formalismes nécessaires sans perturber le système.

- Difficultés rencontrées :

  L'intégration de SAGANE et IROISE nous a permis de ne pas réécrire des environnements déjà existants par ailleurs. Mais, la charge mémoire que représentent les interfaces développés autour de ces outils nous a posé des problèmes. Il n'est pas possible de disposer de tous ces environnements en même temps. Aussi, il faut distinguer la phase de développement du corps d'une spécialiste écrite avec l'un de ces outils, la phase de développement des autres caractéristiques associées à cette spécialiste et sa phase d'exécution. Le développement du corps d'une telle spécialiste sera effectué et testé indépendamment d'ATOME en simulant les hypothèses à lire ou écrire dans le(s) blackboard(s). Lorsque cette phase sera terminée, il faudra écrire les différents modules d'entrée et de sortie de cette spécialiste à partir des primitives proposées par ATOME afin d'effectuer la connexion avec ce dernier. Il faudra également spécifier les autres

caractéristiques de cette spécialiste telles que ses variables de préconditions, ses préconditions... Tout se passe comme si ATOME dispose de l'exécutable du corps de cette spécialiste d'une part, et fournit d'autre part les moyens de définir ses autres propriétés. Au cours de l'exécution d'un système-ATOME, la différence entre les différents types de spécialistes n'apparaîtra que lors de leur exécution effective[3]. Si elle respecte le formalisme-ATOME, l'activation de la spécialiste est effectuée en deux phases : *création du contexte de travail* et *exécution du corps de la* SC. Sinon, elle est effectuée en trois phases : *création du contexte de travail*, *exécution du corps de la* SC et *transformation des résultats*.

Cette étape a fait l'objet d'un stage de DESS IA chez Cognitech [9].

Les idées décrites dans cette étape ont été implantées dans la version 2.0 d'ATOME (Le_Lisp) et sont opérationnelles depuis juillet 1988.

3. Après vérification de sa précondition et si le mécanisme de contrôle a décidé d'activer cette spécialiste.

8
Bilan d'ATOME

Le but de ce chapitre est de résumer les propriétés essentielles d'ATOME concernant sa généricité, son efficacité, sa souplesse et sa modularité. Nous présentons d'abord les différentes possibilités proposées par ATOME pour représenter les spécialistes, pour les faire communiquer ainsi que pour coordonner leurs activités. Nous montrons ensuite comment ATOME intègre efficacité, souplesse et modularité dans son organisation et exploitation des connaissances.

La plupart des chercheurs dans le domaine des architectures de blackboard considèrent qu'actuellement, il y a deux grandes classes d'architectures de blackboard : les architectures descendant de HEARSAY-II, et les architectures descendant de HASP/SIAP. Aussi nous avons choisi de faire une comparaison entre BB-1, descendant de la première catégorie, et ATOME, descendant de la seconde catégorie.

Nous terminons ce chapitre par les extensions d'ATOME envisagées à court, moyen et long terme.

8.1 Généricité

Nous présentons dans ce paragraphe toutes les caractéristiques et particularités qui font d'ATOME un système générique pour un développement incrémental et souple de systèmes d'IA.

- En permettant l'utilisation d'un nombre quelconque de blackboards, ATOME propose ainsi plusieurs types de communication entre les spécialistes (cf. chapitre 1 de la partie I) :
  - *Communication par partage d'informations :*
    Dans ce cas, il suffit d'utiliser un seul blackboard qui servira d'unique moyen de communication et de coordination entre les différentes spécialistes du système. Ce blackboard contiendra à tout instant du

processus de résolution du problème, les données initiales, les résultats intermédiaires émis par chaque spécialiste, ainsi que la solution finale au problème posé (cf. figure 8.1).

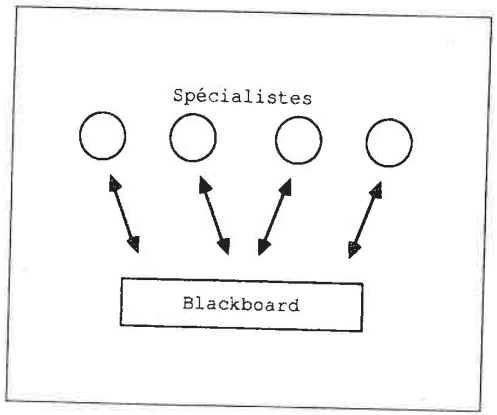


Figure 8.1. Communication par partage d'informations dans ATOME.

- *Communication par envoi d'informations :*

  ATOME permet également la mise en œuvre de systèmes d'IA adoptant une communication inter-spécialistes basée sur des envois d'informations entre ces dernières. Pour ce faire, il suffit de déclarer autant de blackboards que de spécialistes dans le système. Chacun des blackboards est associé à une spécialiste et constitue alors une zone de données privée à celle-ci. Elle y recoit des informations envoyées par les autres spécialistes et y effectue ses propres traitements, avant d'en envoyer les résultats à d'autres spécialistes. Le blackboard sert ainsi de *boîte à lettre* où la spécialiste reçoit ses messages et de *mémoire de travail* où elle effectue ses travaux locaux (cf. figure 8.2).

- *Communication hybride :*

  Une communication hybride entre les différentes spécialistes du système consiste à combiner les deux types de communication précédents de telle sorte qu'une spécialiste ou un sous-ensemble des spécialistes disposent d'un ou plusieurs blackboards pour communiquer entre elles. Elles pourront ainsi placer leurs données initiales et leurs résultats partiels dans un blackboard local accessible uniquement à ces spécialistes, et ne communiquer aux autres spécialistes que les hypothèses jugées nécessaires par l'intermédiaire d'un blackboard global accessible à toutes les spécialistes du système. Le blackboard local contiendra alors les résultats intermédiaires ayant permis d'émettre les hypothèses du blackboard global (cf. figure 3.1 du chapitre 3 de la partie III).

- Afin de répondre aux besoins d'applications d'IA nécessitant une représentation hétérogène des connaissances, ATOME offre plusieurs formalismes pour coder les spécialistes :

  - *Un formalisme déclaratif :*

    Les spécialistes codées à l'aide d'un formalisme déclaratif — systèmes à base de connaissances —, peuvent être de type ATOME (ordre 0+), SAGANE (ordre 0) ou IROISE (ordre 1).

    Les spécialistes ATOME utilisent les blackboards non seulement comme zone de communication mais également comme mémoire de travail. En revanche, les spécialistes SAGANE ou IROISE disposent d'une mémoire de travail privée pour effectuer leurs traitements. Dans ce cas, les blackboards servent uniquement de moyen de communication et de coordination entre les spécialistes.

  - *Un formalisme procédural :*

    ATOME permet également l'intégration des systèmes conventionnels — programmes C ou Lisp — qui sont généralement utilisés pour des phases de traitements numériques (algorithmiques) d'une application.

- Avec son contrôle hybride à multi-phases ATOME offre à tout utilisateur trois types de tâches — dirigées par les événements, dirigées par les règles ou opportunistes — ce qui lui permet de :

  - tester différents types de contrôle pour une tâche;
  - combiner ces types de contrôle;
  - adopter le contrôle le plus approprié durant chaque phase de résolution d'un problème.

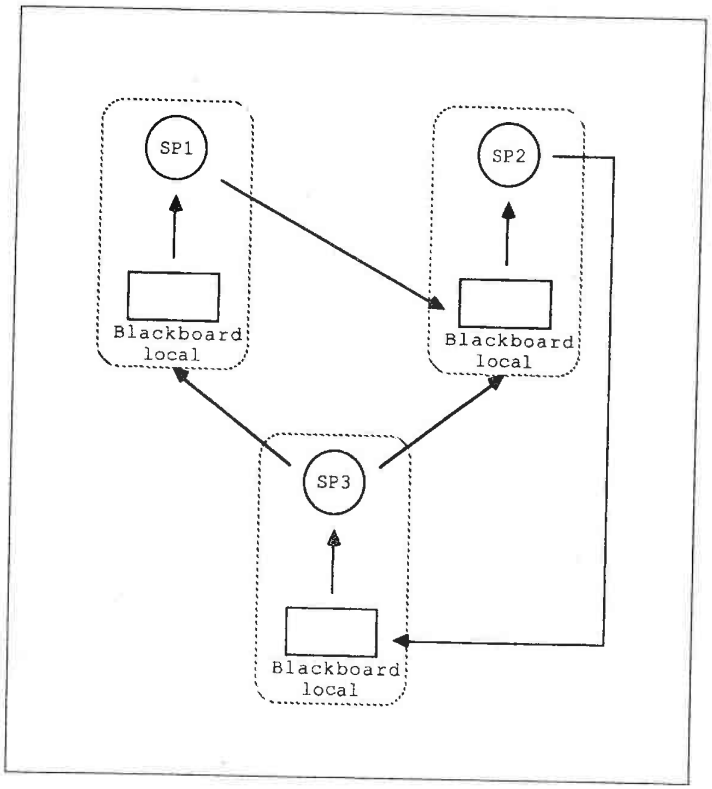


Figure 8.2. Communication par envoi d'informations dans ATOME.

8.2 Efficacité

Les systèmes à base de blackboard sont généralement renommés pour leur inefficacité dans leur fonctionnement et dans leur temps de réponse. Afin de corriger cette limite, ATOME permet la mise en œuvre de systèmes à base de blackboard efficaces aussi bien dans leur fonctionnement que dans leur temps de réponse. Pour ce faire, il définit un mécanisme de contrôle permettant à l'utilisateur de gérer au mieux les structures de contrôle — listes d'événements, résumés des blackboards et blackboards à long terme — tout en lui donnant la possibilité d'orienter les activités du système vers une région de données ou un sous-ensemble de spécialistes à l'aide de la notion de focalisation de contrôle.

- ATOME réduit le nombre d'événements manipulés par le système et par conséquent le temps de raisonnement des tâches en :
  - permettant aux spécialistes de gérer elles-mêmes la création d'événements associés à leurs propres actions;
  - synthétisant les événements portant sur les mêmes types de changements d'une même hypothèse.
- Il réduit également la quantité d'informations stockées dans les blackboards en :
  - permettant à chaque spécialiste de disposer d'une *mémoire de travail locale* où elle mémorise ses informations privées et ne stocke dans le blackboard que celles nécessaires pour la coopération avec les autres spécialistes[1];
  - associant à chaque blackboard un *blackboard à long terme* qui contient les hypothèses non traitées depuis un certain temps. Les blackboards sont parcourus périodiquement afin de rechercher ces hypothèses et les mémoriser dans les blackboards à long terme correspondants. Les spécialistes si elles le désirent peuvent toujours accéder à ces blackboards à long terme, mais uniquement en lecture. Ce mécanisme est bien sûr optionnel.
- Il réduit aussi l'espace de travail des sources de connaissances, qu'elles soient du domaine ou de contrôle en :
  - associant à chaque blackboard, un résumé qui contient uniquement les données jugées importantes pour la poursuite des activités du système. Au lieu de parcourir entièrement les blackboards, la stratégie

1. Cette propriété n'a de sens que pour les spécialistes de type SAGANE, IROISE ou C.

accède uniquement à ces résumés, ce qui diminue considérablement son espace de recherche;

- associant à chaque tâche une liste d'événements locale qui ne mémorise que les changements pertinents pour cette tâche. Chaque tâche spécifie le type d'événements qui doivent lui être signalés et placés dans sa liste d'événements;
- associant à chaque spécialiste un *contexte de travail* évalué à chaque activation de cette dernière. Il lui est en effet possible de définir un filtre sur les blackboards qui lui permettra, à chaque activation, de déterminer ce contexte de travail (sous-ensemble d'hypothèses des blackboards). Ainsi au lieu de raisonner sur tous les blackboards, la spécialiste oriente ses activités vers la région de données qu'elle doit traiter.

- Il permet également aux tâches et aux spécialistes de se focaliser sur des données spécifiques en :
  - signalant aux tâches activées, les hypothèses du résumé du blackboard ayant permis leur activation. Ces hypothèses sont transmises par la stratégie aux tâches activées en tant que focalisation de contrôle;
  - signalant aux spécialistes, les hypothèses associées aux événements ayant permis leur activation. Ces hypothèses sont transmises par les tâches aux spécialistes activées en tant que focalisation de contrôle.

8.3 Souplesse

ATOME est un système générique qui intègre la souplesse aussi bien dans la conception que dans le comportement d'un système-ATOME :

- *Souplesse dans la conception :*
  - ATOME est un système ouvert pour lequel il est aisé d'ajouter ou de supprimer des composantes. Nous avons profité de cette souplesse pour étendre les fonctionnalités d'ATOME en enrichissant son formalisme d'expression des spécialistes;
  - Afin de permettre la comparaison de diverses stratégies de contrôle ainsi que l'analyse de plusieurs configurations et organisations des spécialistes, les composantes d'ATOME sont facilement interchangeables;

  - ATOME est facile d'expression : ce facteur est lié à la clarté de l'architecture globale d'ATOME — séparation entre les connaissances liées au domaine et celles liées au contrôle, et hiérarchisation des sources de connaissances. Il est par conséquent accessible à une large communauté d'utilisateurs;
  - Afin d'aider l'utilisateur dans la mise au point de son application, ATOME propose un mécanisme explicatif permettant :
    * La prise en compte de toutes les entités fondamentales utilisées par le système, c'est-à-dire les événements, les blackboards, les spécialistes, les tâches, la stratégie et les structures de contrôle.
    * Leur représentation sous une forme appropriée — à l'aide d'un blackboard explicatif — en tenant compte des relations qui existent entre ces entités.
    * La restitution de toutes les informations relatives à l'évolution de chaque entité (par exemple, une spécialiste) ou d'une composante de cette entité (par exemple, une règle de cette spécialiste).

    Ainsi, l'utilisateur pourra évaluer et tester diverses configurations et organisations de son système.

- *Souplesse dans le comportement :*
  - La plupart des systèmes à base de blackboard utilisent un contrôle *uniforme* pour toutes les étapes de résolution d'un problème. Ils appliquent le même type de contrôle sans tenir compte du fait que ce dernier pourrait être adapté pendant certaines phases et inadapté pendant d'autres. Par conséquent, ils se sont avérés destinés uniquement à une classe de problèmes — généralement la première classe pour laquelle ils étaient développés — et sont difficilement réutilisables pour d'autres types d'application sans de grandes modifications. Pour corriger ces limites, ATOME propose un contrôle appelé *hybride à multi-phases*. Il subdivise le processus de résolution du problème en une suite de *phases* représentant les différentes étapes d'exécution du système. Durant chaque phase, seul un sous-ensemble de sources de connaissances est pris en compte par le mécanisme de contrôle qui applique alors *le contrôle approprié* pour coordonner leurs activités. A cet effet, ATOME offre trois types de contrôle : dirigé par les règles, dirigé par les événements, et opportuniste;
  - ATOME offre également un mécanisme de contrôle à plusieurs niveaux de granularité : la stratégie effectue un *contrôle global* pour la conduite globale du système en raisonnant sur une vue abstraite

et synthétisée de la solution courante (mémorisée dans les résumés des blackboards). Les tâches, quant à elles, effectuent un *contrôle local* pour gérer les activités des spécialistes qui se trouvent sous leur responsabilité en se basant uniquement sur les récents changements qui ont eu lieu (mémorisés dans les listes d'événements locales);

- L'utilisateur d'ATOME peut affecter à chaque spécialiste du système une liste de paramètres tels que son efficacité et son coût ou tout autre paramètre discriminant permettant de caractériser cette spécialiste;
- L'utilisateur a également la possibilité de définir plusieurs heuristiques qui permettent de favoriser ou de combiner certains paramètres;
- Pour évaluer la contribution potentielle d'une spécialiste vis à vis d'un ensemble d'heuristiques complémentaires ou compétitives, l'utilisateur a la possibilité de définir des règles d'intégration. Celles-ci identifient l'ensemble des heuristiques *actives* du système et définissent la manière dont ces heuristiques doivent être combinées;
- Les poids affectés à une heuristique peuvent évoluer en fonction de l'état d'avancement de la résolution d'un problème;
- Une même tâche (opportuniste) peut être activée plusieurs fois avec des règles d'intégration différentes.

8.4 Modularité

ATOME est un système modulaire dans la mesure où il est composé de plusieurs modules opérant sur diverses structures de données. En effet :

- Seules les spécialistes ont accès en lecture et en écriture aux données stockées dans les différents blackboards.
- Elles ne communiquent leurs contributions partielles qu'à travers les blackboards, ce qui assure une certaine évolutivité du système.
- Chaque tâche accède à sa liste d'événements locale en vue de coordonner les spécialistes se trouvant sous son contrôle. Elle possède une certaine connaissance sur ces spécialistes ainsi que sur leur capacité de résoudre un sous-problème particulier du problème posé. En revanche elle ignore totalement les autres spécialistes du système.
- La stratégie accède aux résumés des blackboards en vue de coordonner les activités des tâches. Elle a connaissance de toutes les tâches.

- Un conflit d'activation entre deux ou plusieurs entités (par exemple, des règles ou des sources de connaissances) est toujours résolu par l'entité qui englobe celle-ci. Par exemple, un conflit entre plusieurs règles d'une source de connaissances[2] est résolu par la source de connaissances en question. Un conflit entre plusieurs spécialistes est résolu par une tâche tandis qu'un conflit entre plusieurs tâches est résolu par la stratégie.

8.5 Comparaison entre ATOME et BB-1

8.5.1 Similitudes

Bien que l'architecture globale de BB-1 diffère de celle d'ATOME, cela n'empêche pas certaines similitudes entre les deux systèmes.

En effet, le niveau *stratégie* du blackboard de contrôle de BB-1 et la *stratégie* en tant que source de connaissances dans ATOME jouent le même rôle. Les deux concepts décrivent la conduite que doit suivre le système pour résoudre son problème. Seule la manière dont cette description est faite change : dans le premier cas, la stratégie est implantée sous forme de suite d'*objectifs* à atteindre alors que dans le second cas, elle est implantée sous forme de sous-problèmes à résoudre. En effet, le niveau *objectif* du blackboard de contrôle dans BB-1 implante une partie de la stratégie du système. De même une *tâche* dans ATOME matérialise la manière dont un sous-problème doit être résolu.

Le niveau *heuristique* de BB-1 enregistre les heuristiques qui implantent un objectif, décrivant ainsi comment ordonner l'activation d'un ensemble de sources de connaissances compétitives. Dans ATOME l'ordre dans lequel les spécialistes vont être exécutées peut être donné explicitement ou calculé à l'aide d'*heuristiques*.

La figure 8.3 montre la similitude qui existe entre les niveaux du blackboard de contrôle de BB-1 et la hiérarchie du contrôle dans ATOME.
Pour implanter un contrôle similaire à celui de BB-1 dans ATOME, une solution consiste à définir autant de tâches *opportunistes* que d'objectifs dans le blackboard de contrôle de BB-1. Ainsi, à un objectif BB-1, on associe une tâche opportuniste d'ATOME. Les heuristiques ainsi que la règle d'intégration qui implante un objectif donné seront les mêmes que pour la tâche correspondante. La stratégie d'ATOME consiste alors à activer en séquence les tâches dans le même ordre que celui des objectifs qui implante la stratégie de BB-1.

2. La stratégie, une tâche ou une spécialiste.

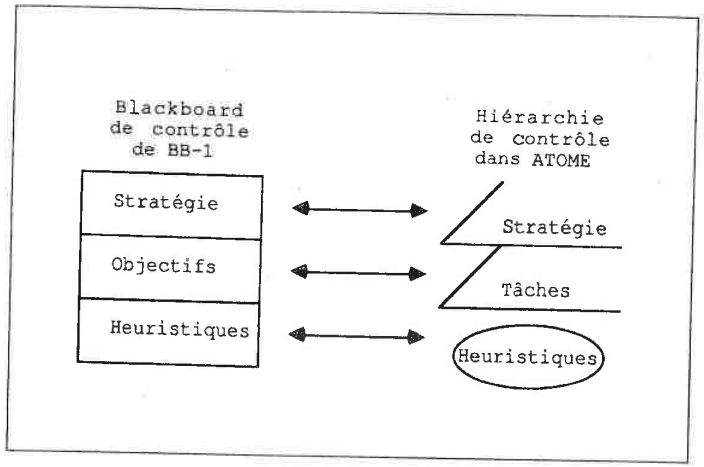


Figure 8.3. Similitudes entre le mécanisme de contrôle de BB-1 et ATOME.

8.5.2 Différences

Les différences entre BB-1 et ATOME sont évidemment nombreuses. En effet :

- Les SCs de contrôle de BB-1 gèrent d'une façon *indirecte* les SCs du domaine en opérant sur le blackboard de contrôle alors que dans ATOME les tâches opèrent *directement* sur les spécialistes qu'elles ont sous leur responsabilité.

- Dans BB-1, toutes les SCs qu'elles soient du domaine ou de contrôle sont compétitives alors que dans ATOME elles sont hiérarchisées.

- Quand un objectif de BB-1 est actif, toutes les SCs qu'elles soient du domaine ou de contrôle sont compétitives alors que dans ATOME, durant l'activation d'une tâche, seul un sous-ensemble de spécialistes est considéré par le système, ce qui conduit à plus d'efficacité dans l'organisation et l'exécution du système.

- A chaque cycle de résolution du problème, BB-1 ne déclenche qu'une seule SC à la fois, alors qu'ATOME permet l'activation aussi bien d'une seule spécialiste (dans le cas d'une tâche opportuniste) que d'une séquence de spécialistes (dans le cas d'une tâche dirigée par les événements ou par les règles) à la fois.

- BB-1 présente un contrôle uniforme durant toutes les phases de résolution d'un problème, alors qu'ATOME adapte le contrôle approprié à chaque phase.

- A l'inverse d'ATOME toutes les SCs de BB-1 qu'elles soient du domaine ou de contrôle ont un unique mode de fonctionnement : multiple, qui consiste à déclencher toutes les règles activables en un seul parcours de la base de règle de la SC.

- Toutes les SCs du domaine de BB-1 sont homogènes. ATOME a étendu le formalisme d'expression des spécialistes pour permettre la réalisation de systèmes d'IA à multi-sources de connaissances hétérogènes.

- Dans BB-1, toutes les actions sur le blackboard engendrent des événements. De plus, afin de sélectionner une ISC à activer, toutes les SCs du système sont confrontées à tous les événements engendrés durant le dernier cycle. ATOME permet aux spécialistes et aux tâches de contrôler la création et le traitement des événements, ce qui conduit à une organisation du contrôle plus efficace que BB-1.

- Dans BB-1, seuls les niveaux *objectif* et *heuristique* du blackboard de contrôle interviennent lors du processus de résolution d'un problème. En effet, le niveau *stratégie* du blackboard de contrôle ne sert que pour indiquer les objectifs que le système désire atteindre et l'ordre dans lequel ces objectifs doivent être traités. Cet ordre est d'ailleurs fixé à l'avance lors de la définition de la stratégie. De plus, quand un objectif a été traité par le mécanisme de contrôle, il n'est plus reconsidéré par le système.

  Dans ATOME, l'ordre dans lequel les tâches (qui jouent le même rôle que les objectifs de BB-1) sont activées varie en fonction de l'état de la solution courante et du processus de résolution du problème, ce qui amène plus de souplesse dans le choix de l'ordre dans lequel les sous-problèmes doivent être examinés.

- BB-1 stocke toutes les informations (qu'elles soient liées au domaine d'application ou au contrôle) dans le blackboard. Par conséquent, toutes les sources de connaissances du système aussi bien du domaine ou de contrôle ont la même représentation syntaxique et opèrent de la même manière, ce qui offre une certaine uniformité et élégance au système.

En revanche, pour des raisons d'efficacité, ATOME stocke uniquement les informations liées au domaine dans les blackboards. Les informations liées au contrôle, quant à elles, sont stockées dans des structures de contrôle : listes d'événements et résumés des blackboards. Par conséquent, les SCs de contrôle (stratégie et tâches) n'ont pas la même structure que les SCs du domaine (spécialistes). L'utilisateur est ainsi confronté à s'adapter aux différentes syntaxes des SCs du système.

8.6 Extensions à court et moyen terme

Dans ce paragraphe, nous décrivons quelques extensions possibles et réalisables à court ou à moyen terme sur ATOME.

8.6.1 Raisonnement dirigé par les buts

Le raisonnement utilisé dans ATOME est dirigé par les données, c'est-à-dire que les différentes spécialistes ne réagissent qu'à l'apparition de changements dans le blackboard pouvant les activer.

Il serait intéressant d'étendre ce raisonnement avec un raisonnement dirigé par les buts en définissant par exemple un nouveau type de tâches *dirigées par les buts* ou en l'intégrant dans la boucle de base d'ATOME. Pour ce faire, plusieurs problèmes se posent :

- *Quel genre de buts intégrer dans* ATOME *?*

  Par exemple, un but consisterait à étendre une région particulière d'un blackboard donné, créer une hypothèse à un certain niveau avec certaines conditions ou formuler des transactions de type : y a t-il une hypothèse vérifiant certaines conditions données ?

- *Comment représenter les buts et où les stocker ?*

  Les buts peuvent être soit définis à l'avance et donnés comme paramètres aux sources de connaissances (stratégie, tâches et spécialistes), soit engendrés durant leur activation.

  Ils peuvent être stockés dans des structures de contrôle telles que *listes de buts* [208], ou dans un *blackboard de buts* [176].

- *Qui doit les traiter ?*

  Les buts peuvent être traités par les spécialistes (buts locaux), les tâches (buts intermédiaires) ou la stratégie (buts globaux).

- *Comment les traiter ?*

  Deux solutions se proposent pour traiter un but donné : le système bloque l'activation de la SC qui a émis un but en attendant qu'il soit résolu ou continue son activation jusqu'à ce qu'elle ait fini de travailler.

8.6.2 Extensions des résumés des blackboards

Le but des résumés des blackboards est, d'une part, de fournir une vue synthétisée du contenu des blackboards et, d'autre part, de contenir moins d'entrées que les blackboards, ceci afin de permettre au mécanisme de contrôle, en l'occurrence la stratégie, d'être efficace dans son raisonnement.

Actuellement, les résumés des blackboards contiennent des données au même niveau de détail que les blackboards. Il serait plus intéressant que ces résumés contiennent des informations plus abstraites que celles des blackboards. Par exemple, des moyennes ou des historiques des données des blackboards. De plus, le filtre qui permet d'identifier les entrées d'un blackboard à stocker dans le résumé correspondant est fixe pendant toute la résolution du problème.

Il pourrait être plus intéressant de définir une source de connaissances dont le rôle serait :

1. d'adapter le filtre à appliquer sur les hypothèses des blackboards pour détecter les plus importantes en fonction de l'état d'avancement de la résolution du problème;

2. de synthétiser les hypothèses sélectionnées dans le but de les mémoriser dans le résumé sous une forme plus abstraite.

8.6.3 Stratégie opportuniste

De la même manière que nous avions défini une tâche opportuniste, il serait intéressant de définir un nouveau type de stratégie — une stratégie opportuniste — qui consisterait à utiliser des agendas, des heuristiques et des règles d'intégration en vue de sélectionner la meilleure tâche parmi un ensemble de tâches possibles. Ceci conduirait à une résolution de problèmes plus opportuniste qu'à l'état actuel. Ainsi en fonction du contenu des résumés des blackboards, la stratégie sélectionnerait un ensemble de tâches et un ordre dans lequel elles seraient exécutées. Cet ordre pourrait être donné explicitement par la stratégie ou calculé à l'aide d'heuristiques.

8.6.4 Extensions du mécanisme explicatif

Actuellement, le mécanisme explicatif stocke des informations concernant l'évolution de chaque entité manipulée par ATOME. Ce mécanisme explicatif permet ainsi de retrouver ces informations à différents niveaux de granularité. Cependant, il ne prend pas en compte les besoins de l'utilisateur vis à vis de la granularité souhaitée au moment du stockage des informations. Autrement dit, certaines informations du blackboard explicatif risquent de ne pas être utilisées. Ces informations occupent inutilement de la place mémoire et ralentissent le fonctionnement du système.

Il serait donc plus judicieux de concevoir un mécanisme explicatif capable de ne mémoriser que l'information nécessaire relative aux besoins de l'utilisateur.

8.6.5 Compilation de règles

Jusqu'à présent, le souci d'efficacité du contrôle a été envisagé par des techniques d'abstraction, de partition et de structuration des données, par organisation hiérarchique des sources de connaissances, et par utilisation du contrôle le plus approprié pour chaque phase de résolution du problème.

En plus de ces techniques, nous pensons qu'il serait intéressant d'étudier l'utilisation des techniques classiques telles que les réseaux de discrimination de type RETE [95,106], notamment pour optimiser l'activation de la stratégie, d'une tâche ou d'une spécialiste.

8.7 Extensions à long terme

Les extensions d'ATOME que nous envisageons actuellement concernent l'intégration d'un *raisonnement adaptatif* permettant de mettre en œuvre des systèmes d'IA devant fonctionner en temps réel.

L'intégration des contraintes temps réel dans les systèmes à multi-sources de connaissances représente un problème délicat à résoudre. En effet, à la différence des approches classiques temps réel, les tâches (travaux) à exécuter sont interdépendantes et contribuent à une partie de la résolution du problème. Il n'est pas possible de préciser à l'avance la priorité d'une tâche ni les ressources qui lui seront nécessaires, ces caractéristiques évoluant en fonction de l'avancement de la résolution du problème et de l'état de la solution courante. Les seuls travaux à notre connaissance qui vont en ce sens sont ceux de Lesser, Pavlin et Durfee [180]. Leur projet consiste à intégrer des contraintes temps réel dans le système DVMT. Leur approche consiste à fournir au système des mécanismes lui permettant de détecter si la meilleure solution — la plus complète, précise et certaine — ne peut être obtenue dans les délais requis, de sélectionner en conséquence les stratégies nécessaires et appropriées et de les appliquer dans le but de trouver une solution moins bonne que la précédente mais acceptable en respectant les délais imposés. Cette étude est fondée sur trois classes de raisonnements :

1. Eliminer certaines données en entrée/sortie des sources de connaissances.
2. Se baser sur une vue plus abstraite des données.
3. Définir des sources de connaissances qui regroupent et résument un ensemble de sources de connaissances et substituent ces dernières.

Les expériences préliminaires effectuées sur DVMT montrent que cette voie est peut-être à approfondir et représente un sujet ouvert de recherche à l'heure actuelle.

Afin d'être en mesure de traiter des problèmes d'IA avec des contraintes temps réel, nous pensons qu'il serait donc intéressant d'étendre ATOME par des mécanismes adaptés à un raisonnement contraint par le temps en vue d'intégrer :

- *Raisonnement concurrent de plusieurs sources de connaissances :*

  Dans certaines applications d'IA, il est possible d'envisager l'activation en parallèle de tâches ou de spécialistes. L'objectif du raisonnement concurrent est de caractériser les relations qui existent entre les sources de connaissances et de détecter les situations qui permettent de les déclencher en parallèle, ceci dans le but de diminuer le temps de réponse du système.

- *Raisonnement interruptible par des événements externes :*

  Parmi les axes actuels de recherche en IA, figure le développement de systèmes capables de prendre en compte l'évolution du monde extérieur [134]. Le but du raisonnement interruptible est de permettre l'accès aux événements externes, d'identifier leur type — *urgence*, *très-important...*, *peu-important* et *fausse alarme* — et de les traiter en conséquence.

- *Raisonnement en temps limité :*

  Le système doit être capable de prendre comme paramètre important, le temps qui lui est attribué pour fournir sa réponse. Il peut le considérer, d'une part, comme une ressource dans la mesure où il dispose d'un certain temps pour résoudre le problème et, d'autre part comme une contrainte dans la mesure où il doit respecter les délais imposés.

Le système doit être capable d'ordonner les raisonnements décrits auparavant par priorité, urgence ou tout autre critère adapté à la prise en compte de situations critiques. Nous appellerons cette approche *raisonnement adaptatif* : le système oriente et adapte son fonctionnement en fonction des contraintes de temps, de son degré de parallélisme, et de l'évolution du monde extérieur.

8.8 ATOME au CRIN

Trois applications sont en cours de développement au CRIN avec ATOME. La première concerne l'identification de sous-marins en collaboration avec le GERDSM[3]. La seconde application entre dans le cadre de la compréhension de l'écriture et fait l'objet du projet GRAPHEIN [36]. La dernière est une application d'aide à la décision en maintenance de voirie urbaine en collaboration avec la mairie de Nancy [92]. Ces applications ne font que débuter et sont en phase de spécification ou de début de réalisation avec ATOME. Aussi, il est encore prématuré de décrire leur architecture complète face à celle d'ATOME, nous nous limitons donc à une brève description de ces dernières en fonction des informations dont nous disposons actuellement.

D'autre part, ATOME sert de cadre de travail pour des projets relatifs aux raisonnements approximatif, hypothétique et temporel. L'objectif de ces raisonnements est de permettre le traitement de données incomplètes et/ou évolutives dans le temps.

8.8.1 Applications

Nous tenons à préciser que les développements de ces applications avec ATOME sont récents et ne peuvent fournir des résultats définitifs concernant la validation d'ATOME.

Identification de sous-marins

Cette application se situe dans le contexte de l'interprétation des signaux sous-marins pour l'identification des sources de bruit les ayant engendrés. Pour ce faire, les experts se basent sur la description des signaux analysés, de l'environnement dans lequel ils se situent, des sources de bruit potentielles (hélice ou moteur diesel), et des contraintes structurelles et fonctionnelles qui les régissent. Une version préliminaire de cette application a été développée à l'aide de l'outil ORA [19] et a mis en évidence le besoin d'une architecture de blackboard.

3. Groupe d'Etude et Recherche en Détection Sous-Marine (Toulon).

Le développement de cette application avec ATOME et BB-1 est en cours d'étude au CRIN et au GERDSM.

Aide à la décision

Le but du système est de dresser un planning des travaux à exécuter sur le réseau routier de la ville de Nancy. Les travaux sont répartis en deux catégories principales :

- la *maintenance* qui consiste à assurer l'entretien courant des rues par de petits travaux ponctuels ne nécessitant pas de gros moyens techniques et financiers;
- la *réfection* qui suppose de faire appel à une entreprise spécialisée pour un chantier important. Il faut alors décider si la rue est à refaire complètement (depuis les fondations) ou simplement en surface (uniquement le tapis).

L'objectif de la mairie de Nancy est d'aboutir à une amélioration de l'état global du réseau urbain. L'étendue des interventions possibles sera limitée par un budget annuel à respecter. Il faut donc intervenir juste et bien. Les choix des rues à traiter et de la réfection pour chacune d'elles (type, matériaux et techniques à mettre en œuvre) représentent un problème crucial à résoudre. Ces choix doivent aller dans le sens de la politique que s'est fixée la ville, à savoir favoriser le confort et la sécurité de ses habitants, en faisant intervenir le cas échéant des contraintes d'esthétique.

Une approche de type blackboard s'est avérée adaptée et est en cours d'étude à l'aide d'ATOME.

Les différentes tâches mises en évidence dans ce système sont relatives au diagnostic des dégradations de chaque rue, à la thérapie à appliquer, aux choix des matériaux et à la décision à prendre face au budget fixé. L'ordre et l'instant d'intervention de ces tâches n'est pas connu *a priori*. Par exemple, une dégradation spécifique dans une rue peut conduire à une thérapie immédiate tandis que dans d'autres cas, il est nécessaire d'évaluer toutes les dégradations dans une rue avant d'en connaître la cause et l'origine et d'en déduire une thérapie. Un certain nombre de facteurs tels que la densité du trafic ou l'état des réseaux souterrains, peuvent également influencer les choix dans les thérapies à appliquer et les rues à réparer. Une stratégie globale pour gérer ces tâches existe donc.

Ce projet est en cours de développement au CRIN [92] et permettra de valider et perfectionner le mécanisme de contrôle dans ATOME.

Reconnaissance de l'écriture

Le but du projet GRAPHEIN est de réaliser un système générique de compréhension de l'écriture où les caractères sont imprimés et multifontes. Il doit intégrer différentes méthodes d'analyse et choisir la meilleure selon le contexte dans lequel il est utilisé.

Ce projet est ambitieux et représente un axe de recherche important dans le domaine de la compréhension de l'écriture. Aussi, bien que les principales tâches du système soient déterminées (mesure, prétraitement, segmentation, analyse et reconnaissance), les travaux actuels se focalisent essentiellement sur la segmentation. Il s'agit d'être capable de choisir les techniques de segmentation les mieux adaptées et de les appliquer, en fonction de la page à segmenter, des connaissances *a priori* sur le texte à reconnaître et des connaissances acquises lors de la segmentation de pages antérieures.

Une maquette a été développée avec ATOME [36,17]. L'exemple consiste à segmenter deux pages d'un document de deux façons différentes. L'idée est de tenir compte des connaissances extraites lors de la segmentation de la première page pour segmenter plus efficacement la deuxième. Seules, les tâches *mesure*, *prétraitement*, et *segmentation* ont été implantées. La stratégie globale du système ne sera élaborée que lorsque toutes les tâches seront intégrées. Pour le moment, son seul rôle est de lancer pour chaque page, les tâches mesure et prétraitement pour analyser si le texte est incliné et le redresser le cas échéant. Elle active ensuite la tâche segmentation.

Des spécialistes de macro-segmentation et de micro-segmentation ont été définies et placées sous la responsabilité de cette tâche. L'expertise que renferme cette tâche est mal connue et fait l'objet de la recherche en cours dans le projet. Aussi, un outil tel qu'ATOME a permis de tester diverses stratégies et d'en fixer quelques unes.

Cette maquette utilise un seul blackboard appelé *physique* contenant les divers résultats sur la structure physique du texte. Les niveaux de ce blackboard sont : texte, page, segment, ligne, mot physique, bloc et caractère physique. Un second blackboard *logique* sera défini lors de l'intégration des tâches de reconnaissance et d'analyse du texte afin de représenter sa structure logique : texte, chapitre, paragraphe, phrase, mot et caractère. Pour plus de précisions sur cette maquette, se référer à [36].

L'état actuel d'avancement de ce projet ne nous permet pas de prétendre qu'il a validé ATOME. Cependant, il nous permet de certifier qu'ATOME est un bon outil de mise au point. De plus, son architecture a été un bon support de travail pour formaliser ce projet.

8.8.2 Raisonnements approximatif, hypothétique et temporel dans ATOME

Les intégrations des raisonnements approximatif, hypothétique et temporel dans ATOME ont été étudiées indépendamment les unes des autres. L'objectif visé à long terme est de les combiner dans un seul et même raisonnement [191, 192].

Raisonnement approximatif

Dans un premier temps, nous nous sommes intéressés à intégrer dans ATOME un raisonnement approximatif fondé sur la théorie de *Dempster-Shafer* [225, 248,139]. Ceci dans le but de traiter les données imprécises et incertaines. L'objectif à plus long terme de ces travaux est de prendre en compte la pluralité des sources de connaissances d'ATOME qui peuvent exprimer leur incertitude dans des langages divers.

Raisonnement hypothétique

Le raisonnement hypothétique a pour objectif de permettre la résolution d'un problème même si certaines valeurs des données de ce problème sont inconnues. Il propose une solution du problème en fonction d'hypothèses émises sur ces valeurs en explorant les différentes alternatives possibles.

L'intégration de ce type de raisonnement dans ATOME est en cours d'étude [23, 168]. Une approche de type ATMS[4] a été adoptée. Le principe de base de cette approche est d'explorer en parallèle les différentes suppositions effectuées sur les valeurs inconnues [155,156].

Raisonnement temporel

Dans le but d'aborder des domaines d'applications où les données manipulées sont évolutives dans le temps, un module de raisonnement temporel a été intégré dans ATOME. Ce module permet de :

- Garder trace de l'évolution du monde et raisonner sur une longue période de temps.
- De représenter les notions de précédence ou de simultanéité d'un fait par rapport à un autre.

4. Assumption-based Truth Maintenance System : système de maintien de la vérité fondé sur la notion de supposition.

- Planifier dans le temps.

La représentation du temps adoptée est basée sur la manipulation d'intervalles associés aux valeurs des attributs des hypothèses des blackboards. Mais afin de pouvoir raisonner sur des événements répétitifs, la notion de période a été introduite ainsi que les relations symboliques pouvant exister entre deux intervalles ou deux périodes [117,168,190,193].

Afin de limiter la quantité d'informations temporelles dans les blackboards, il est possible de construire un historique par aggrégation de ces informations.

Les informations temporelles sont souvent incomplètes et en constante évolution (une information vraie à un instant donné peut se révéler fausse à un instant postérieur). Afin de gérer cet aspect non-monotone de l'univers, il est nécéssaire d'introduire un module de maintien de la cohérence. Une approche TMS[5] [64] est en.cours d'études. Cette approche entre dans le cadre du raisonnement hypothétique défini précédemment. A l'inverse d'un ATMS, elle explore les suppositions effectuées sur les valeurs inconnues les unes après les autres en autorisant des retours-arrières.

8.9 ATOME à l'extérieur du CRIN

ATOME a été diffusé dans plusieurs laboratoires en France : au CERT à Toulouse pour une application de télémanipulation [24], à l'université de Lyon 1 pour une application de diagnostic et une application dans le domaine de la CAO (Conception Assistée par Ordinateur) et à l'Ecole Nationale Supérieure de Télécommunications (ENST) pour une application de traitement de signal. Il est également aux laboratoires GTE aux Etats-Unis, à l'école polytechnique fédérale de Lausanne (EPFL) et sera bientôt au Canada à l'université de Montréal.

5. Truth Maintenance System : système de maintien de la vérité.

Conclusion générale

Dans cette thèse, nous avons décrit et caractérisé le modèle du blackboard pour gérer la coopération entre les agents d'un univers multi-agents. Les systèmes que nous avons présentés ont été choisis, d'une part, pour leur notoriété et, d'autre part, pour leur apport dans le domaine. Le classement de ces systèmes par type de contrôle *procédural*, *hiérarchique* ou *à base de blackboard* a mis en évidence :

- *La souplesse du modèle dans la représentation du mécanisme de contrôle :*

  En effet, selon les caractéristiques et objectifs visés, le concepteur d'un système à base de blackboard pourra implanter le mécanisme de contrôle soit sous forme de programme (cf. HEARSAY-II) renfermant toutes les stratégies de contrôle nécessaires pour gérer les activités des sources de connaissances, soit sous forme de méta-sources de connaissances (cf. CRYSALIS) avec chacune d'elles gérant *directement* un ensemble de sources de connaissances ou soit sous forme de sources de connaissances de contrôle gérant *indirectement* l'ensemble des sources de connaissances à travers un second blackboard de contrôle (cf. BB-1).

- *Le manque d'un contrôle universel :*

  Nous avons montré que ces trois types de contrôle n'ont pu concilier *efficacité* et *souplesse* qui constituent les deux facteurs les plus importants à prendre en compte lors de la réalisation d'un système à base de blackboard. Les contrôles *procédural* et *hiérarchique* sont efficaces dans leur résolution du problème mais sont moins souples que le contrôle *à base de blackboard*. Ce dernier offre un comportement plus opportuniste et plus souple que les deux précédents mais manque d'efficacité.

  De plus, leur *uniformité* impose une résolution de problèmes rigide dans la mesure où le contrôle n'est pas adapté à chaque étape de cette résolution.

  Afin de corriger ces limites, nous avons défini un contrôle *hybride à multi-phases* qui intègre efficacité et souplesse dans un même environnement et

permet d'adapter le contrôle à utiliser au fur et à mesure de l'avancement de la résolution du problème.

Nous avons implanté ce type de contrôle en développant l'outil ATOME décrit dans la troisième partie de cette thèse et pour lequel nous avons dressé un bilan complet.

Dans la suite, nous présentons une conclusion d'ordre générale sur les architectures de blackboard en insistant sur leur structure à la fois modulaire et souple et sur les études en cours ou à approfondir concernant la formalisation d'un problème nécessitant l'utilisation d'une architecture de blackboard, l'apport du modèle du blackboard en programmation, l'intégration du parallélisme, les liens éventuels avec les bases de données et enfin avec les modèles connexionistes.

Modularité

La modularité est une propriété inhérente dans les architectures de blackboard. En effet, le modèle du blackboard permet de séparer l'espace de solutions (le blackboard), les connaissances qui opèrent sur cet espace (les sources de connaissances) et les mécanismes qui gèrent ces connaissances (le contrôle). La modularité est également respectée au sein de chacune de ces entités. En effet, l'espace des solutions est divisé en plusieurs niveaux d'abstraction permettant ainsi de considérer la solution sous plusieurs niveaux de détail. Les connaissances liées au domaine sont partagées en plusieurs modules autonomes et indépendants. Le mécanisme de contrôle, quant à lui, est généralement constitué de plusieurs modules de contrôle opérant sur des structures de contrôle.

Le modèle du blackboard offre des architectures faciles à tester et à maintenir. En effet, les sources de connaissances peuvent être facilement testées séparément avant d'être intégrées dans le système. La création d'une nouvelle source de connaissances ou la mise à jour d'une ancienne source de connaissances n'a pas de grand effet sur les autres sources de connaissances ni sur le mécanisme de contrôle. De même que la mise à jour de ce dernier n'engendre pas automatiquement une mise à jour des sources de connaissances ou du blackboard. Par conséquent, les architectures de blackboard s'avèrent robustes et fiables.

Nouvelles voies de recherche

Les recherches en IA concernant les architectures de blackboard sont relativement récentes et nombreux sont les problèmes non encore abordés ou résolus. Nous citons ici les voies de recherche qui nous semblent les plus prometteuses dans ce domaine :

1. *Formalisation des problèmes :*

   Malgré cette structure à la fois souple et modulaire du modèle du blackboard, la complexité des problèmes qu'il permet de traiter nécessite une certaine méthode de développement. Malheureusement, à l'heure actuelle, il est très difficile dans les architectures de blackboard, et plus généralement en IA, de définir des techniques théoriques ou pratiques sur la manière dont de tels systèmes doivent être conçus. Bien que des systèmes génériques tels que BB-1, GBB, AGE et ATOME se soient efforcés de fournir à l'utilisateur une certaine démarche à suivre pour formaliser son problème [100,8,147,188], nous pensons qu'étant donné le grand nombre d'applications ayant utilisé le modèle du blackboard avec succès, les architectures de blackboard ont atteint un stade de maturité où il est intéressant de mener des recherches dans le but de définir des méthodes de conception comme celles développées pour la conception de bases de données [60].

   Cette voie de recherche permettrait probablement de mieux cerner le modèle du blackboard, les entités qu'il met en jeu ainsi que les relations qui existent entre ces entités.

2. *Programmation centrée blackboard :*

   Le modèle du blackboard respecte le principe "diviser pour régner" dans la mesure où :

   - les connaissances du domaine sont réparties en plusieurs sources de connaissances;
   - l'espace de recherche est divisé en plusieurs blackboards eux-mêmes décomposés en niveaux d'abstraction. Chacun des niveaux peut contenir plusieurs éléments de la solution;
   - les connaissances de contrôle sont réparties en modules ou sources de connaissances.

   Aborder un problème avec une "approche blackboard" consisterait à détecter les composantes de base (les sources de connaissances, le blackboard et le mécanisme de contrôle) qui interviennent dans la résolution du problème, à les considérer et à les concevoir *indépendamment* les unes des autres.

Nous pensons que cette approche peut être utilisée dans les phases d'analyse et de spécification d'un problème au même titre que les approches classiques de programmation [210].

3. *Parallélisme dans les architectures de blackboard :*

   Exécuter en parallèle plusieurs sources de connaissances semble à première vue être un mécanisme intrinsèque dans une architecture de blackboard. Aussi, dès le début des recherches dans ce domaine, certains systèmes ont choisi cette voie pour résoudre les conflits d'intervention des sources de connaissances [90,206,81].

   Les résultats donnés dans le cadre de ces travaux confirment la difficulté d'améliorer les performances d'un système par parallélisation [206]. Ils ont montré que la rapidité d'exécution d'un système à contrôle centralisé n'a pas été améliorée par les méthodes de parallélisation. Ils ont également prouvé qu'un système basé essentiellement sur le parallélisme sans contrôle conduit à de nombreux traitements en parallèle et à une mauvaise maîtrise du comportement du système qui aboutit à une solution non satisfaisante au problème. Plus la granularité du parallélisme est fine[6], plus les composantes du système deviennent interdépendantes et plus il est difficile de concilier efficacité d'exécution du système et cohérence de la solution.

   Nous pensons que, pour améliorer les performances des architectures de blackboard, il est effectivement nécessaire d'introduire un certain degré de parallélisme. L'approche que nous jugeons très prometteuse consiste à distribuer le contrôle entre les sources de connaissances de telle sorte qu'elles soient suffisamment intelligentes pour gérer elles-mêmes les activités relatives à leurs besoins, leur interaction se faisant par exemple par négociations à travers le blackboard [163, 187].

   Cette voie de recherche permettrait lorsque les machines supports et le type d'applications le permettent, d'avoir des systèmes fonctionnant dans un environnement multi-processeurs avec de faibles temps de réponse.

4. *Liens avec les bases de données :*

   La structure du blackboard pourrait être assimilée à une base de données dont les fonctions d'accès seraient gérées par un SGBD[7]. L'apport des techniques de bases de données aux architectures de

6. Tests en parallèle des prémisses de règles, déclenchement en parallèle des règles activables d'une source de connaissances ou activation en parallèle de plusieurs sources de connaissances exécutables.

7. Système de Gestion de Base de Données.

blackboard se situe donc au niveau de l'organisation des données dans le blackboard et de l'accès à ces dernières.

Nous pensons qu'à leur tour, les architectures de blackboard peuvent être d'un apport intéressant pour les bases de données distribuées ou hétérogènes. En effet, les types de contrôle que ces architectures mettent en œuvre pour coordonner les activités des sources de connaissances peuvent aider à gérer les transactions effectuées sur les bases de données.

5. *Le connexionisme dans les systèmes à base de blackboard :*

   Jusqu'alors, nous avons envisagé que les sources de connaissances peuvent être soit des programmes conventionnels, soit des systèmes à base de connaissances. Nous pensons qu'il serait intéressant, par exemple, d'étudier le cas où une source de connaissances repose sur un modèle connexioniste. Ceci permettrait d'envisager des systèmes "intelligents" comportant des modules connexionnistes spécialisés, contrôlés par des systèmes symboliques [229].

Annexe A
Conférences spécialisées sur les architectures de blackboard

La recherche dans le domaine des systèmes à base de blackboard a donné lieu à trois conférences spécialisées : la première a eu lieu les 12 et 13 juin 1986 à Pittsburgh (Pennsylvanie); la seconde a eu lieu l'année suivante le 13 juillet à Seattle (Washington); la troisième a eu lieu le 24 août 1988 à Saint Paul (Minnesota).

Année 1986

La majorité des systèmes présentés lors de cette conférence concernait des applications dans le domaine de la robotique, notamment, le système NAVLAB de l'université de Carnegie-Mellon et le projet "Flying Eye Robot" du centre d'intelligence artificielle de Boeing. Ces derniers sont plutôt à classer parmi les systèmes de synchronisation de processus parallèles que de systèmes à base de blackboard.

Parmi les autres travaux présentés, nous pouvons citer les systèmes DVMT de l'université de Massachusetts qui intègre un mécanisme de planification dans sa résolution de problème, ABE de Teknowledge et AGORA de l'université de Carnegie-Mellon où le blackboard est l'un des moyens de communication interprocessus proposé, et GBB de l'université de Massachusetts qui s'inspire beaucoup des techniques de bases de données pour réaliser les opérations d'accès au blackboard.

N'ayant pas assisté à cette conférence, nous ne pouvons pas donner de précisions sur les problèmes et les points importants qui ont pu être soulevés, analysés et/ou discutés.

Année 1987

En 1987, la conférence a été organisée en cinq pôles d'intérêt :

1. *Architecture de blackboard :*

   Etant donné le nombre croissant des systèmes d'IA portant l'étiquette "blackboard", il était nécessaire de faire le point sur ce qui peut être considéré comme un système à base de blackboard ou non. Ceci faisait l'objet de cette première session.

   Les critères retenus pour définir une architecture de blackboard sont liés aux trois composants de base que nous avons définis au chapitre 1.1 de la partie I, à savoir, la structure du blackboard, les sources de connaissances du domaine et un mécanisme de contrôle. Ce dernier joue un rôle primordial dans cette définition.

2. *Contrôle :*

   Durant la seconde session, les discussions portaient sur le problème du contrôle dans les systèmes à base de blackboard. Parmi les points soulevés, nous pouvons retenir la difficulté de concilier souplesse et efficacité dans de tels systèmes.

   En particulier, nous pouvons citer la présentation de GBB-1, système intégré dans GBB, qui améliore la performance du contrôle de type BB-1 tout en gardant sa souplesse. Cependant, le manque d'efficacité qui subsiste dans ce type de contrôle a été souligné lors des discussions.

3. *Parallélisme :*

   Cette session était destinée à répondre aux questions suivantes : les systèmes à base de blackboard sont ils (facilement) parallélisables ? quel degré de parallélisme faut-il adopter ?

   Nous pouvons retenir de cette session les travaux menés à l'université de Stanford sur le système POLYGON. Ces travaux ont montré que, plus la granularité du parallélisme est fine, plus les entités du systèmes deviennent interdépendantes et plus le système est difficile à maîtriser.

4. *Performance et temps réel :*

   L'objet de cette session était d'étudier la possibilité d'utiliser le modèle du blackboard dans des applications faisant intervenir des contraintes de temps réel.

   De même que la session précédente, cette session a montré que le problème reste ouvert.

5. *Environnements :*

   Cette dernière session concernait les environnements de développement de systèmes à base de blackboard.

   De nombreux environnements ont été présentés dont OPUS d'IntelliCorp , BBC de l'université de Pennsylvanie, GEST de l'institut technologique de Géorgie, BB-1 de l'université de Stanford et ATOME du CRIN/INRIA-Lorraine.

   Il est à remarquer qu'il n'existe aucun outil industriel pour le développement de systèmes d'AI à base de blackboard. Les différents outils présentés dans cette session et les outils actuels sont des produits de recherche.

Année 1988

La conférence spécialisée de 1988 a été organisée en trois sessions :

1. *Contrôle :*

   Cette session était consacrée à analyser les problèmes d'efficacité et de souplesse du mécanisme de contrôle dans les architectures de blackboard.

   Parmi les idées retenues, nous pouvons citer les travaux développés à l'université de Massachusetts sur le système DVMT et son raisonnement dirigé par les buts. Les concepteurs de ce système ont défini un ensemble de relations qui peuvent exister entre des buts (assistance, compétition, coopération, inhibition...) et qui permettent de diminuer considérablement le nombre de buts à traiter.

   Une autre idée à retenir était le "contrôle hybride à multi-phases" et son implantation dans ATOME présenté dans cette thèse.

   Nous pouvons également citer l'approche de type BB-1 implantée dans le système d'ordonnancement SONIA de l'université de Paris VI.

2. *Parallélisme et temps réel :*

   La seconde session a commencé par une présentation des recherches menées par l'université de Massachusetts concernant les problèmes de temps réel et de parallélisme dans les systèmes GBB et DVMT. Cette présentation était une réflexion de fond sur les diverses possibilités et niveaux de parallélisme envisagés.

   Une approche pour la prise en compte d'événements externes dans BB-1 a été également présentée. Ceci a été appliqué pour implanter

un système de surveillance permanente des pulsations cardiaques d'un malade placé dans un service de soins intensifs.

3. *Applications :*

   La dernière session a été consacrée aux applications de type blackboard. Parmi les systèmes présentés, nous pouvons citer ESPRIT-KRITIC de Framentec dans le domaine des télécommunications, CIM de Boeing dans le domaine de l'avionique et ABLS de l'université de New York à Buffalo pour la reconnaissance de codes postaux.

Annexe B
Glossaire des termes ATOME

Agenda des ISPs exécutables
: Liste d'attente qui regroupe toutes les instances de spécialistes activables (ayant leur précondition satisfaite).

Agenda des ISPs sélectionnées
: Liste d'attente qui regroupe toutes les instances de spécialistes réveillées (susceptibles de vouloir travailler) dont la précondition n'est pas ou n'est plus satisfaite.

ATOME
: Acronyme de **A**nother **TO**ol for Developing **M**ulti-**E**xpert Systems.

Attribut
: Champ lié à un niveau du blackboard. Il permet de caractériser les hypothèses de ce niveau. Un attribut d'une hypothèse peut avoir plusieurs valeurs alternatives provenant de plusieurs sources d'informations et affectées d'un coefficient de vraisemblance numérique ou égal à *indéfini*.

Blackboard
: Structure de données globale à toutes les spécialistes. Cette structure est divisée en plusieurs niveaux d'abstraction. Elle mémorise les données initiales, les résultats intermédiaires ainsi que les résultats finaux de l'application. ATOME permet l'utilisation de plusieurs blackboards.

Blackboard à long terme
: Structure de données qui mémorise les hypothèses anciennes et non productives du blackboard. Elle est accessible uniquement en lecture et de façon explicite par les spécialistes.

Blackboard explicatif
: Structure de données qui mémorise les informations explicatives sur l'exécution d'un système-ATOME (activation de sources de connaissances, création d'hypothèses, événements engendrés...).

Coefficient d'importance
Coefficient attribué à chaque règle d'une source de connaissances afin de définir sa priorité d'exécution.

Coefficient de vraisemblance
Coefficient attribué à chaque valeur d'un attribut d'une hypothèse. Il est numérique ou égal à *indéfini*.

Conditions de rejet
Champ de la spécialiste qui précise les conditions sous lesquelles une instance de cette spécialiste ne pourra fournir aucun résultat.

Contexte de travail
Sous-ensemble d'hypothèses du blackboard constituant la région de données à traiter par une spécialiste. Ce contexte est déterminé à chaque activation de cette dernière par filtrage sur le blackboard.

Contrôle hybride à multi-phases
Type de contrôle implanté dans ATOME qui consiste à diviser la résolution d'un problème en plusieurs phases en appliquant un contrôle approprié pour chacune d'elles.

Cycle
Un cycle dans ATOME représente l'activation d'une règle de la stratégie.

Dirigé par les événements
Mode de raisonnement utilisé au niveau des tâches. Une tâche dirigée par les événements sélectionne l'événement le plus important de sa liste d'événements locale et parcourt sa base de règles en vue d'activer les spécialistes exécutables avec cet événement.

Dirigé par les règles
Mode de raisonnement utilisé au niveau des tâches. Une tâche dirigée par les règles sélectionne la règle la plus importante de sa base de règles et parcourt sa liste d'événements locale en vue d'activer les spécialistes activables avec les événements qui satisfont la partie gauche de cette règle.

Evénement
Signal envoyé par les spécialistes pour prévenir des mises à jour qu'elles ont effectuées dans le blackboard, à savoir, création d'une nouvelle hypothèse, modification ou suppression d'une ancienne hypothèse.

Filtre
Spécification du type des données du blackboard qui intéressent une spécialiste. Ce filtre est utilisé pour déterminer le contexte de travail de la spécialiste.

Focalisation de contrôle
Ensemble d'hypothèses du résumé du blackboard qui ont permis l'activation de tâches ou ensemble d'hypothèses du blackboard associées aux événements qui ont permis l'activation de spécialistes.

Heuristique
Règle de combinaison des variables de contrôle. Elle est appliquée aux ISPs lors du raisonnement opportuniste.

Hypothèse
Objet situé à un niveau d'un blackboard.

Hypothèse importante
Hypothèse du résumé du blackboard. La spécification des conditions sous lesquelles une hypothèse d'un certain niveau du blackboard peut être considérée comme importante relève de l'expertise de l'application en cours de développement avec ATOME.

Instance de spécialiste (ISP)
Activité potentielle d'une spécialiste dans le contexte de l'événement qui l'a réveillée.

Lien
Relation entre deux hypothèses d'un même niveau ou de deux niveaux différents. Chaque lien a obligatoirement un lien inverse.

Liste d'événements
Structure de contrôle utilisée par les tâches afin de mémoriser les changements du blackboard qui les intéressent.

Mécanisme explicatif
Ensemble des structures et des modules qui permettent de stocker et de restituer les informations explicatives.

Mode de fonctionnement
Champ associé à chaque SC respectant le formalisme ATOME et qui détermine la manière dont ses règles vont être traitées lors de son activation.

Mode de raisonnement
Propriété associée à chaque tâche et qui détermine la manière dont celle-ci coordonne l'ensemble des spécialistes se trouvant sous son contrôle (cf. dirigé par les événements, dirigé par les règles et opportuniste).

Niveau
Composante du blackboard permettant de représenter ce dernier sous plusieurs niveaux de détail. Généralement, un blackboard est composé de plusieurs niveaux.

Niveau d'abstraction
Voir niveau.

Niveau du blackboard
Voir niveau.

Nœud
Voir hypothèse.

Opportuniste
Mode de raisonnement utilisé au niveau des tâches. Une tâche opportuniste parcourt sa liste d'événements locale et sa base de règles en vue de créer des ISPs. La stratégie fournit à la tâche une règle d'intégration afin que cette dernière sélectionne la meilleure instance de spécialiste : celle qui répond au mieux à cette règle d'intégration.

Précondition
Champ de la spécialiste qui précise les conditions sous lesquelles cette spécialiste devient activable (il peut être statique ou dépendant de l'état des blackboards).

Priorité
Coefficient attribué à une ISP exécutable. L'ISP la plus prioritaire (qui a le plus grand coefficient) est sélectionnée pour être activée.

Propagation des événements
Les événements engendrés par les spécialistes sont reçus par un module de propagation intégré dans ATOME chargé de propager les événements uniquement vers les tâches intéressées. Ce module dispose des connaissances fournies par les tâches quant aux types d'informations susceptibles de les intéresser.

Règle d'intégration
Règle de combinaison des heuristiques. Elle est appliquée aux ISPs lors du raisonnement opportuniste.

Résumé du blackboard
Structure de contrôle mémorisant uniquement les hypothèses jugées importantes pour la suite du processus de résolution du problème. Elle évite à la stratégie de parcourir tout le blackboard.

SC de contrôle
Voir stratégie et tâche.

SC du domaine
Voir spécialiste.

Source de connaissances (SC)
Module autonome, c'est-à-dire indépendant des autres sources de

connaissances, dont le rôle est soit de résoudre un sous-problème du problème de départ (SC du domaine) soit coordonner cette résolution (SC de contrôle).

Spécialiste
SC du domaine d'application. Elle renferme une partie de l'expertise et est destinée à résoudre un problème bien précis.

Stratégie
SC de contrôle qui fournit à un système-ATOME un contrôle global. Son rôle est de gérer les activités de toutes les tâches du système.

Synthèse
Opération de regroupement des événements qui signalent des actions similaires.

Système-ATOME
Système d'IA développé à l'aide d'ATOME.

Tâche
SC de contrôle qui fournit à un système-ATOME un contrôle local. Son rôle est de gérer un sous-ensemble des spécialistes du système.

Variable de contrôle
Paramètre associé à une spécialiste tel que son efficacité, sa fiabilité...

Il peut être statique ou dépendant de l'état des blackboards.

Bibliographie

[1] *Proceedings of the 1988 Workshop on Distributed Artificial Intelligence*, Lake Arrowhead, California, May 22-25, 1988.

[2] *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, S^t Paul, Minnesota, August 24, 1988.

[3] *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[4] Knowledge Systems Laboratory 85, incorporating the Heuristic Programming Project. Department of Computer Science, Stanford University, 1985.

[5] N. Aiello. A Comparative Study of Control Strategies for Expert Systems: AGE Implementation of Three Variations of PUFF. In *Proceedings of the* 3^{rd} *National Conference on Artificial Intelligence (*AAAI-83*)*, pages 1-4, Washington, D. C., 1983.

[6] N. Aiello. AGEPUFF*: A Simple Event-Driven Program* - AGE *Example Series: Number 2.* Technical Report HPP-81-25, Heuristic Programming Project, Computer Science Department, Stanford University, June 1981.

[7] N. Aiello, C. Bock, H. P. Nii, and W. C. White. *Joy of* AGE*ing: An introduction to the* AGE-1 *System.* Technical Report HPP-81-23, Heuristic Programming Project, Computer Science Department, Stanford University, October 1981.

[8] N. Aiello, C. Bock, H. P. Nii, and W. C. White. AGE *Reference Manual:* AGE-1. Technical Report HPP-81-24, Heuristic Programming Project, Computer Science Department, Stanford University, October 1981.

[9] D. Albertoli. *Enrichissement des sources de connaissances* ATOME. Rapport de stage, COGNITECH-CRIN-ISIAL, Avril-Juillet 1988.

[10] K. M. Andress and A. C. Kak. Evidence Accumulation and Flow of Control in a Hierarchical Spatial Reasoning System. *AI Magazine*, 9(2):75–94, 1988.

[11] T. Andro, J. C. Coriton, H. mahe, and P. Mariot. SAGANE : *Manuel de référence.* Technical Report, COGNITECH, Décembre 1987.

[12] K. E. Arzen. Expert Systems for Process Control. In *Proceedings of the 1st Conference on Artificial Intelligence in Engineering*, pages 1127–1138, Southampton, U. K., April 1986.

[13] R. Balzer, L. D. Erman, P. London, and C. Williams. HEARSAY-III: A Domain-Independent Framework for Expert Systems. In *Proceedings of the 1st National Conference on Artificial Intelligence (AAAI-80)*, pages 108–110, Stanford, California, 1980.

[14] J. A. Barnett. How Much is Control Knowledge Worth? A Primitive Example. *Artificial Intelligence*, 22:77–89, 1984.

[15] D. R. Barstow, N. Aiello, R. O. duda, L. D. Erman, C. L. Forgy, D. Gorlin, R. D. Greiner, D. B. Lenat, P. E. London, J. McDermott, P. N. Nii, P. Politakis, R. Reboh, S. Rosenschein, A. C. Scott, W. V. Melle, and S. M. Weiss. *Building Expert Systems*, chapter "Languages and Tools for Knowledge Enginnering", pages 283–345. Addison-Wesley, 1983.

[16] L. S. Baum, R. T. Dodhiawala, and V. Jagannathan. The Erasmus System. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[17] A. Belaid, B. Maître, H. Lâasri, and Y. Chenevoy. GRAPHEIN: A Blackboard-Based System for Character Understanding. *Submitted to the International Workshop on Raster Imaging and Digital Typography (*RIDT-89*)*, Lausanne, Switzerland, October 12-13, 1989.

[18] M. Benda, L. S. Baum, R. T. Dodhiawala, and V. Jagannathan. Boeing Blackboard System. In *Proceedings of The* AAAI *Sponsored Workshop on High Level Tools*, October 1986.

[19] E. Benzaquin, J. F. Gagey, P. Le Blé, and G. Sagué. ORA - *Evolution d'un outil dirigé par les données et les modèles.* Rapport interne, GERDSM - Toulon, 1987.

[20] R. Bisiani, F. Allen, A. Forin, R. Lerner, and M. Bauer. *Distributed Artificial Intelligence*, chapter "The Architecture of the Agora Environment", pages 99–117. Pitman, 1987.

[21] A. Bonnet. *L'intelligence artificielle, promesses et réalités.* InterEditions, 1984.

[22] A. Bonnet, J. P. Haton, and J. M. Truong-Ngoc. *Systèmes experts, vers la maîtrise technique.* InterEditions, 1986.

[23] A. C. Boury. *Etude de l'intégration du raisonnement hypothétique dans* ATOME. Rapport de DEA, CRIN/INRIA-Lorraine, Septembre 1988.

[24] G. A. Boy and N. Mathé. Using Non-Monotonic Reasoning for Operator Assistant Systems in Reactive Environments. Presented at the ESA-ESTEC Workshop on Artificial Intelligence Applications on Space Projects, November 15-17, 1988.

[25] G. Brajnik, G. Guida, M. Pighin, and C. Tasso. ADL: A Language and Tool for Designing Blackboard System Architectures. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 1–17, S^t Paul, Minnesota, August 24, 1988.

[26] J. A. Brugge. ABC*: A Knowledge-Based System for Determining Structural Components of Proteins.* MSAI Practicum Report, Knowledge Systems Laboratory, Computer Science Department, Stanford University, May 27 1987.

[27] S. Brunessaux, H. Lâasri, and B. Maître. ATOME *version common lisp 3.0 : Dictionnaire des fonctions et variables.* Rapport technique 89-R-023, CRIN/INRIA-Lorraine, Fe'vrier 1989.

[28] S. Brunessaux, H. Lâasri, and B. Maître. ATOME *version common lisp 3.0 : Manuel d'utilisation.* Rapport technique 89-R-032, CRIN/INRIA-Lorraine, Fe'vrier 1989.

[29] S. Brunessaux, H. Lâasri, and B. Maître. *Tuning the Traveller Salesman Problem within* ATOME. Technical Report 88-R-083, CRIN/INRIA-Lorraine, September 1988.

[30] S. Calac and F. Monnet. IKAROS - *Intelligence and Knowledge Aided Recognition of Speech.* Esprit Project 954, Division Informatique, CRCGE, Laboratoires de Marcoussis, July 1987.

[31] N. F. Carver, V. R. Lesser, and D. L. McCue. Focusing in Plan Recognition. In *Proceedings of the 4th National Conference on Artificial Intelligence (*AAAI-84*)*, pages 42–48, Austin, Texas, 1984.

[32] J. Chailloux, M. Devin, F. Dupont, J. M. Hullot, B. Serpette, and J. Vuillemin. *Le_Lisp version 15.2 - Le manuel de référence.* Technical Report, INRIA-Roquencourt, Mai 1986.

[33] B. Chandrasekaran and S. Mittal. *Advances in Computers*, chapter "Conceptual Representation of Medical Knowledge for Diagnosis by a Computer: MDX and Related Systems", pages 217–293. Academic Press, New York, 1983.

[34] E. Charniak and D. McDermot. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, Massachusetts, 1985.

[35] W. Chehire and T. de Lastic Saint-Jal. KIRK : Un environnement de dèveloppement de systèmes experts et une application : l'interprétation de photographies aériennes. *Actes des 8èmes journées internationales sur les systèmes experts et leurs applications (Avignon-88)*, tome 3, pages 391-408, Avignon, 30 Mai-03 Juin, 1988.

[36] Y. Chenevoy. *Segmentation contextuelle de documents imprimés multipolices.* Rapport de DEA 88-R-092, CRIN/INRIA-Lorraine, Septembre 1988.

[37] W. J. Clancey and C. Bock. *Representing Control Knowledge as Abstract Tasks and Metarules.* Technical Report KSL-85-16, Knowledge Systems Laboratory, Computer Science Department, Stanford University, August 1985.

[38] M. Clerget. Un atelier de développement industriel de systèmes experts. *Actes des 11èmes Journées francophones sur l'informatique (Architectures avancées pour l'intelligence artificielle)*, pages 237-245, Nancy, 12-13 Janvier 1989.

[39] A. Collinot. Revising the BB-1 Basic Control Loop to Control the Behavior of Knowledge Sources. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 19–36, S^t Paul, Minnesota, August 24, 1988.

[40] S. E. Conry and R. A. Meyer. *Multistage Negotiation in Distributed Planning.* Technical Report 86-67, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, December 1986.

[41] O. Corby. BB-1 en SMECI. *Actes du 6ème Congrès* AFCET-RFIA (RFIA-87), tome 1, pages 581-586, Antibes, 16-20 Novembre 1987.

[42] D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem-Solving Networks.* PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, December 1982.

[43] D. D. Corkill. Design Alternatives for Parallel and Distributed Blackboard Systems. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 89–106, S^t Paul, Minnesota, August 24, 1988.

[44] D. D. Corkill, K. Q. Gallagher, and P. M. Johnson. Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, pages 18–23, Seattle, Washington, July 1987.

[45] D. D. Corkill, K. Q. Gallagher, and P. M. Johnson. *From Prototype to Product: Evolutionary Development within the Blackboard Paradigm.* Technical Report 86-46, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, October 1986.

[46] D. D. Corkill, K. Q. Gallagher, and K. E. Murray. GBB: A Generic Blackboard Development System. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pages 1008–1014, Philadelphia, Pennsylvania, August 1986. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 26, pages 503-517, Addison-Wesley, 1988).

[47] D. D. Corkill and V. R. Lesser. The Use of Meta-Level Control for Coordination in A Distributed Problem-Solving Network. In *Proceedings of the 3rd National Conference on Artificial Intelligence (AAAI-83)*, pages 748–756, Washington, D. C., 1983.

[48] D. D. Corkill, V. R. Lesser, and E. Hudlicka. Unifying data-directed and goal-directed control: an example and experiments. In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI-82)*, pages 143–147, Pittsburgh, Pennsylvania, 1982.

[49] I. D. Craig. A Distributed Blackboard Architecture. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[50] I. D. Craig. *Distributed Control In A Blackboard System.* PhD thesis, Department of Computing, Univeristy of Lancaster, September 1987.

[51] I. D. Craig. The Ariadne-1 Blackboard System. *The Computer Journal*, 29(3):235–240, 1986.

[52] Crocus. *Systèmes d'exploitation des ordinateurs.* Dunod, Avril 1975.

[53] M. L. Das. Knowledge Base for Structural Design. In *Proceedings of the 1st Conference on Artificial Intelligence in Engineering*, pages 33–44, Southampton, U. K., April 1986.

[54] R. Davis. Meta-Rules: Reasoning about Control. *Artificial Intelligence*, 15(3):179–222, 1980.

[55] R. Davis. Report on the Second Workshop on Distributed AI (1981). SIGART *Newsletter*, 80:13–23, 1982.

[56] R. Davis. Report on the Workshop on Distributed AI (1980). SIGART *Newsletter*, 73:42–52, 1980.

[57] R. Davis and R. G. Smith. Negotiation as a Metaphor for Distributed Problem-Solving. *Artificial Intelligence*, 20:63–109, 1983.

[58] K. S. Decker. Distributed Problem-Solving Techniques: A Survey. IEEE *Transactions on Systems, Man, and Cybernetics*, SMC-17(5):729–740, 1987.

[59] J. R. Delaney. Multi-System Report Integration Using Blackboards. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[60] C. Delobel and M. Adiba. *Bases de données et systèmes relationnels*. Dunod, 1982.

[61] J. R. Dixon. An Architecture for Application of Artificial Intelligence to Design. In *Proceedings of the* 21^{th} *Design Automation Conference*, pages 634–640, 1984.

[62] R. T. dodhiawala and G. R. Cross. Analysis of Cosmic Ray Tracks using Distributed Problem-Solving. *Pattern Recognition Letters*, 4:471–476, December 1986.

[63] R. T. Dodhiawala, V. Jagannathan, and L. S. Baum. Erasmus System Design: Performance Issues. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[64] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[65] B. A. Draper, R. T. Collins, J. Brolio, A. R. Hanson, and E. M. Riseman. Issues in the Development of a Blackboard-Based Schema System for Image Understanding. In *Blackboard Systems*, pages 189–218, R. Engelmore and T. Morgan, editors, Addison-Wesley, 1988.

[66] B. A. Draper, R. T. Collins, J. Brolio, A. R. Hanson, and E. M. Riseman. The Schema System. *Submitted to the International Journal of Computer Vision*, September 1988.

[67] E. H. Durfee. *A Unified Approach to Dynamic Coordination Planning Actions and Interactions in a Distributed Problem-Solving Network*. PhD thesis. Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, September 1987.

[68] E. H. Durfee. *An Approach to Cooperation: Planning and Communication in a Distributed Problem-Solving Network*. Technical Report 86-09, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, March 1986.

[69] E. H. Durfee and V. R. Lesser. Incremental Planning to Control a Blackboard Based Problem Solver. In *Proceedings of the* 5^{th} *National Conference on Artificial Intelligence (*AAAI-86*)*, pages 58–64, Philadelphia, Pennsylvania, August 1986. (Also presented at the Carnegie-Mellon University Workshop on Blackboard Systems for Robot Control Perception and Control, June 12-13, 1986).

[70] E. H. Durfee and V. R. Lesser. Incremental Planning to Control a Time-Constrained, Blackboard-Based Problem-Solver. IEEE *Transactions on Aerospace and Electronic Systems*, 1988.

[71] E. H. Durfee and V. R. Lesser. Using Partial Global Plans to Coordinate Distributed Problem-Solvers. In *Proceedings of the* 10^{th} *International Joint Conference on Artificial Intelligence (*IJCAI-87*)*, pages 875–883, Milan, Italy, 1987.

[72] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Coherent Cooperation Among Communicating Problem-Solvers. IEEE *Transactions on Computers*, C-36(11):1275–1291, November 1987.

[73] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperation through Communication in a Distributed Problem Solving Network. In *Distributed Artificial Intelligence*, pages 29–58, M. N. Huhns, editor, Pitman, 1987.

[74] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Increasing Coherence in A Distributed Problem-Solving Network. In *Proceedings of the* 9^{th} *International Joint Conference on Artificial Intelligence (*IJCAI-85*)*, pages 1025–1030, Los Angeles, California, 1985.

[75] A. Elfes and S. N. Talukdar. A Distributed Control System for the CMU Rover. In *Proceedings of the* 8^{th} *International Joint Conference on Artificial Intelligence (*IJCAI-83*)*, pages 830–833, Karlsruhe, West Germany, 1983.

[76] R. Engelmore and T. Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

[77] R. Engelmore and A. Terry. Structure and Function of the CRYSALIS System. In *Proceedings of the* 6^{th} *International Joint Conference on Artificial Intelligence (*IJCAI-79*)*, pages 250–256, Tokyo, Japan, 1979.

[78] R. S. Engelmore. *Structure and Function of the* CRYSALIS *System*. Technical Report HPP-79-16, Heuristic Programming Project, Computer Science Department, Stanford University, 1979.

[79] R. S. Engelmore, A. J. Morgan, and H. P. Nii. *Blackboard Systems*, chapter "Hearsay-II", pages 25–29. Addison-Wesley, 1988.

[80] J. R. Ensor and J. D. Gabbe. Transactional Blackboards. *Artificial Intelligence in Engineering*, 1(2):80–84, 1986.

[81] J. R. Ensor and J. D. Gabbe. Transactional Blackboards. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 340–344, Los Angeles, California, 1985. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 24, pages 465-473, Addison-Wesley, 1988).

[82] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and R. D. Reddy. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM *Computing Surveys*, 12:213–253, 1980. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 3, pages 31-86, Addison-Wesley, 1988).

[83] L. D. Erman, J. S. Lark, and F. Hayes-Roth. *Engineering Intelligent Systems: Progress Report on* ABE. Technical Report TTR-TSE-86-102, Teknowledge, Inc., May 1986. (Also presented at the Carnegie-Mellon University Workshop on Blackboard Systems for Robot Control Perception and Control, June 12-13, 1986).

[84] L. D. Erman and V. R. Lesser. A Multi-Level Organization for Problem-Solving Using Many Diverse Cooperating Sources of Knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI-75)*, pages 483–490. Tbilisi, Georgia, USSR, 1975.

[85] L. D. Erman and V. R. Lesser. *Computer Vision Systems*, chapter "System Engineering Techniques for Artificial Intelligence Systems", pages 37–45. A. Hanson and E. Riseman, Academic Press, New York, 1978.

[86] L. D. Erman and V. R. Lesser. *Trends in Speech Recognition*, chapter "The HEARSAY-II Speech Understanding System: A Tutorial". Prentice Hall, 1980.

[87] L. D. Erman, P. E. London, and S. F. Fickas. The Design and an Example use of HEARSAY-III. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 409–415, Vancouver, British Columbia, 1981. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 13, pages 281-295, Addison-Wesley, 1988).

[88] M. R. Fehling and L. D. Erman. Report on the Third Annual Workshop on Distributed AI (1982). SIGART *Newsletter*, 1982:3–12, 1982.

[89] M. R. Fehling, S. Forrest, and B. M. Wilber. The Heuristic Control Virtual Machine. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[90] R. D. Fennell and V. R. Lesser. Parallelism in Artificial Intelligence Problem-Solving: A Case Study of HEARSAY-II. IEEE *Transactions on Computers*, C-26(2):98–111, 1977.

[91] J. Ferber and M. Ghallab. Problématiques des univers multi-agents intelligents. *Actes des journées nationales sur l'intelligence artificielle* PRC/IA (88), pages 295-320, Toulouse, 1988.

[92] C. Ferraris. *Vers un système expert d'aide à la décision en maintenance de voirie urbaine.* Rapport de DEA, CRIN/INRIA-Lorraine, Septembre 1988.

[93] R. Fikes. Integrating Blackboards into the KEE Object-Oriented Modeling Environment. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[94] A. Florescu, E. P. M. V. Liempd, and H. Velthijsen. BLONDIE-III: *An Environment for Distributing Problem-Solving.* Intern Rapport 88 DNL/16, Netherlands PTT, Dr. Neher Laboratories, June 1986.

[95] L. Forgy. RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem. *Artificial Intelligence*, 19:17–37, 1982.

[96] S. Forrest, J. S. Lark, and L. D. Erman. Representing and Reasoning About the Control of Concurrent and Distributed Actions. In *Proceedings of the 1988 Workshop on Distributed Artificial Intelligence*, Lake Arrowhead, California, May 22-25, 1988.

[97] G. C. Fox and P. C. Messina. Advanced Computer Architectures. *Scientific American, Trends In Computing*, 1:16–23, September 1988. Reprinted from the October 1987 Issue.

[98] A. Garvey. *Implementing Diverse Forms of Control Knowledge in Multiple Control Architectures.* Technical Report KSL-87-40, Knowledge Systems Laboratory, Computer Science Department, Stanford University, May 1987.

[99] A. Garvey, C. Cornelius, and B. Hayes-Roth. Computational Costs versus Benefits of Control Reasoning. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[100] A. Garvey, M. Hewett, M. V. Johnson Jr., R. Schulman, and B. Hayes-Roth. BB-1 *User Manual - Common* LISP *Version 2.0.* Technical Report KSL-86-61, Knowledge Systems Laboratory, Computer Science Department, Stanford University, August 1987.

[101] A. Garvey and B. Hayes Roth. An Empirical Analysis of Explicit vs. Implicit Control Architectures. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 31–58, S^t Paul, Minnesota, August 24, 1988.

[102] L. Gasser, C. Braganza, and N. herman. *Distributed Artificial Intelligence*, chapter "MACE: A Flexible Testbed for Distributed AI Research", pages 119–152. Pitman, 1987.

[103] L. Gasser, C. Braganza, and N. Herman. *Reading in Distributed Artificial Intelligence*, chapter "Implementing Distributed AI Systems Using MACE", pages 445–450. Morgan Kaufmann, 1988.

[104] M. R. Genesereth and D. E. Smith. *Meta-Level Architecture.* Technical Report HPP-81-6, Heuristic Programming Project, Computer Science Department, Stanford University, December 1982.

[105] M. P. Georgeff. A Framework for Control in Production Systems. In *Proceedings of the* 6^{th} *International Joint Conference on Artificial Intelligence (*IJCAI-79*)*, pages 328–334, Tokyo, Japan, 1979.

[106] M. Ghallab. Compilation de bases de connaissances. *Actes des journées nationales sur l'intelligence artificielle* PRC/IA (88), pages 231-253, Toulouse, 1988.

[107] M. Gilloux, C. Tarridec, and M. A. Simon. *Manuel de référence* IROISE. Technical Report, CNET Lannion.

[108] J. F. Gilmore, M. F. Melinda, A. L. Stevenson, and M. X. Rabin. TESS - The Tactical Expert System. In SPIE *Applications of Artificial Intelligence* III, Orlondo, Florida, April 1986.

[109] Y. Gong and J. P. Haton. Une société de spécialistes à niveaux multiples pour l'interprétation de signaux. *Actes du* $6^{ème}$ *congrès* AFCET-RFIA (RFIA-87), tome 1, pages 245-257, Antibes, 16-20 Novembre 1987.

[110] P. E. Green. *Distributed Artificial Intelligence*, chapter "AF: A Framework for Real-Time Distributed Cooperative Problem-Solving", pages 153–175. Pitman, 1987.

[111] A. Hanson and E. Riseman. VISIONS: A Computer System for Integrating Scenes. In *Computer Vision Systems*, pages 303–333, A. Hanson and E. Riseman (Eds.), Academic Press, New York, 1978.

[112] A. R. Hanson and E. M. Riseman. *The* VISIONS *Image Understanding System.* Technical Report 86-62, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, December 1986.

[113] J. P. Haton. Compréhension de la parole et intelligence artificielle : l'état des recherches. *Actes du séminaire en dialogue homme-machine à composante orale*, pages 11-12, Nancy, octobre 1984.

[114] J. P. Haton. Intelligence Artificielle et Compréhension Automatique de la Parole : Etat des Recherches et Comparaison avec la Vision par Ordinateur. *Techniques et Sciences Informatiques*, 4(5):265–287, 1985.

[115] J. P. Haton. *Knowledge-Based Methodology in Pattern Recognition and Understanding.* Technical Report 691, INRIA-Lorraine, July 1987.

[116] J. P. Haton. Les systèmes à base de connaissances en reconnaissance et en interprétation de formes. *Actes de Cognitiva-87*, tome 2, pages 73-80, Paris, 18-22 Mai 1987.

[117] J. P. Haton, B. Maître, H. Lâasri, and T. Mondot. ATOME: Another TOol for developing Multi-Expert Systems. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[118] B. Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence*, 26:251–321, July 1985. (Also appeared as Technical Report HPP-83-38, Heuristic Programming Project, Computer Science Department, Stanford University, June 1983).

[119] B. Hayes-Roth. *A Multi-Processor Interrupt-Driven Architecture for Adaptive Intelligent Systems.* Technical Report KSL-87-31, Knowledge Systems Laboratory, Computer Science Department, Stanford University, June 1987.

[120] B. Hayes-Roth. *Dynamic Control Planning in Adaptive Intelligent Systems.* Technical Report KSL-87-67, Knowledge Systems Laboratory, Computer Science Department, Stanford University, November 1987.

[121] B. Hayes-Roth. BB-1*: An Architecture for Blackboard Systems that Control, Explain, and Learn About their Own Behavior.* Technical Report HPP-84-16, Heuristic Programming Project, Computer Science Department, Stanford University, December 1984.

[122] B. Hayes-Roth. *The Blackboard Architecture: A General Framework for Problem-Solving?* Technical Report HPP-83-30, Heuris-

tic Programming Project, Computer Science Department, Stanford University, May 1983.

[123] B. Hayes-Roth, B. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan, and O. Jardetzky. PROTEAN: Deriving Protean Structure From Constraints. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pages 904–909, Philadelphia, Pennsylvania, 1986. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 20, pages 417-431, Addison-Wesley, 1988).

[124] B. Hayes-Roth, B. B. Buchanan, O. Lichtarge, M. Hewett, R. Altman, J. Brinkley, C. Cornelius, B. Duncan, and O. Jardetzky. *Elucidating Protein Structure from Constraints in* PROTEAN. Technical Report KSL-85-35, Knowledge Systems Laboratory, Computer Science Department, Stanford University, October 1985.

[125] B. Hayes-Roth, A. Garvey, M. V. Johnson Jr., and M. Hewett. *A Layered Environment for Reasoning about Action.* Technical Report KSL-86-38, Knowledge Systems Laboratory, Computer Science Department, Stanford University, November 1986.

[126] B. Hayes-Roth and F. Hayes-Roth. A Cognitive Model for Planning. *Cognitive Science*, 3:275–310, 1979.

[127] B. Hayes-Roth, F. Hayes-Roth, S. Rosenschein, and S. Cammarata. *Blackboard Systems*, chapter "Modeling Planning as an Incremental, Opportunistic Process", pages 231–244. Addison-Wesley, 1988.

[128] B. Hayes-Roth and M. Hewett. *Blackboard Systems*, chapter "BB-1: An Implementation of the Blackboard Control Architecture", pages 297–313. Addison-Wesley, 1988.

[129] B. Hayes-Roth, M. V. Johnson, A. Garvey, and M. hewett. Application of the BB-1 Blackboard Control architecture to Arrangement Assembly Tasks. *Artificial Intelligence in Engineering*, 1(2):85–94, 1986.

[130] B. Hayes-Roth, M. V. Johnson, A. Garvey, and M. Hewett. *Blackboard Systems*, chapter "Building Systems in the BB* Environment", pages 543–560. Addison-Wesley, 1988.

[131] F. Hayes-Roth, L. D. Erman, S. Fouse, J. S. Lark, and J. Davidson. ABE: *A Cooperative Operating System and Development Environment.* Technical Report TTR-ISE-88-105, Teknowledge, Inc., May 1988.

[132] F. Hayes-Roth and V. R. Lesser. Focus of Attention in the Hearsay-II Speech Understanding System. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 27–35, Cambridge, Massachusetts, 1977.

[133] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors. *Building Expert Systems.* Addison-Wesley, Readings, Massachusetts, 1983.

[134] M. Hewett and B. Hayes-Roth. Real-Time I/O in Knowledge-Based Systems. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 101–117, S^t Paul, Minnesota, August 24, 1988.

[135] M. Hewett and B. Hayes-Roth. The BB-1 Architecture: A Software Engineering View. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[136] C. Hewitt. Viewing Control Structures as Patterns Of Passing Messages. *Artificial Intelligence*, 8(3):323–364, 1977.

[137] C. Hewitt and H. Liberman. *Design Issues in Parallel Architectures for Artificial Intelligence.* Technical Report AI Memo No. 750, MIT Artificial Intelligence Laboratory, November 1983.

[138] E. Hudlicka. *Diagnosing Problem-Solving System Behavior.* PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, February 1986.

[139] R. C. Hughes and J. N. Maksym. Acoustic Signal Interpretation: Reasoning with Non-Specific and Uncertain Information. *Pattern Recognition*, 18(6):475–483, 1985.

[140] M. N. Huhns, editor. *Distributed Artificial Intelligence.* Pitman, London, Los Altos, California, 1987.

[141] M. N. Huhns, U. Mukhopadhyay, and L. M. Stephens ans R. D. Bonnell. *Distributed Artificial Intelligence*, chapter "DAI for Document Retrieval: The MINDS Project", pages 249–283. Pitman, 1987.

[142] M. N. Huhns, L. M. Stephens, and D. B. Lenat. Cooperation for DAI through Common-Sense Knowledge. In *Proceedings of the 1988 Workshop on Distributed Artificial Intelligence*, Lake Arrowhead, California, May 22-25, 1988.

[143] P. Jacquot. *Interface graphique d'*ATOME. Rapport de stage, COGNITECH-CRIN-ISIAL, Avril-Juillet 1988.

[144] V. Jagannathan. Realizing the Concurrent Blackboard Model. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 119–131, S^t Paul, Minnesota, August 24, 1988.

[145] V. Jagannathan, L. S. Baum, and R. T. Dodhiawala. Designing a Distributed Blackboard System with Erasmus. In *Proceedings of*

the AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[146] P. M. Johnson, D. D. Corkill, and K. Q. Gallagher. Integrating BB-1 Style Control into the Generic Blackboard System. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[147] P. M. Johnson, K. Q. Gallagher, and D. D. Corkill. GBB *Reference Manual Version 1.2.* Technical Report 87-120, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, June 1988.

[148] J. Jones, M. Millington, and P. Ross. *Blackboard Systems*, chapter "A Blackboard Shell in PROLOG", pages 533–542. Addison-Wesley, 1988. (Also appeared in Proceedings of the 7^{th} European Conference on Artificial Intelligence, Brighton, United Kingdom, 1986, pages 428-436).

[149] G. L. Steele Jr. *Common Lisp, the Language.* Digital Press, 1984.

[150] M. V. Johnson Jr. and B. Hayes-Roth. Integrating Diverse Reasoning Methods in the BB-1 Blackboard Control architecture. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[151] K. Kaiser, L. Baum, D. Blevins, B. Miller, and V. Jagannathan. Adapting the Blackboard Model for Cokpit Information Management. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 193–209, S^t Paul, Minnesota, August 24, 1988.

[152] S. E. Keene. *Object-Oriented Programming in Common-Lisp.* Addison-Wesley, 1989.

[153] B. W. Kernighan and D. M. Ritchie. *The* C *Programming Language.* Prentice Hall, 1987.

[154] J. H. Kim, D. W. Payton, and K. E. Olin. An Expert System for Object Recognition in Natural Scenes. In *Proceedings of the* 1^{st} *Conference on Applications of* AI, pages 170–175, Denver, Colorado, 1984.

[155] J. De Kleer. An assumption-Based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[156] J. De Kleer. Extending the ATMS. *Artificial Intelligence*, 28:163–196, 1986.

[157] J. Knell. *The Heuristic Control Virtual Machine: An Introduction.* Technical Report, Teknowledge, Inc., November 1987.

[158] A. Koenig and E. Crochon. TRAM : Tableau noir pour robotique autonome mobile. *Actes des* $8^{èmes}$ *journées internationales sur les systèmes experts et leurs applications (Avignon-88)*, tome 1, pages 493-510, Avignon, 30 Mai-03 Juin, 1988.

[159] W. A. Kornfeld and C. E. Hewitt. The Scientific Community Metaphor. IEEE *Transactions on Systems, Man, and Cybernetics*, SMC-11(1):24–33, January 1981.

[160] H. Lâasri. Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et réalisations — Contribution personnelle. Thèse de l'université de Nancy 1, Février 1989.

[161] H. Lâasri. *Quelques systèmes* IA *fondés sur le modèle du blackboard.* Rapport interne 86-R-133, CRIN/INRIA-Lorraine, 1986.

[162] H. Lâasri. OMSC *: Outil d'aide au développement de systèmes à Multi-Sources de Connaissances.* Rapport de DEA 86-R-133, CRIN/INRIA-Lorraine, Septembre 1986.

[163] H. Lâasri and B. Maître. A Negotiation Approach to the Blackboard-Based Problem-Solving. *Submitted to the* 11^{th} *International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[164] H. Lâasri, B. Maître, and J. P. Haton. Hybrid Control to achieve Efficiency and Flexibility in Blackboard-Based Systems. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 59–72, St. Paul, Minnesota, August 24, 1988.

[165] H. Lâasri, B. Maître, and J. P. Haton. *Opportunism and Efficiency in Meta-Level Blackboard Architectures:* ATOME *Case Study.* Technical Report 88-R-007, CRIN/INRIA-Lorraine, 1988.

[166] H. Lâasri, B. Maître, and J. P. Haton. Organisation, coopération et exploitation des connaissances dans les architectures de blackboard : Cas d'ATOME. *Actes des* $8^{èmes}$ *journées internationales sur les systèmes experts et leurs applications (Avignon-88)*, tome 3, pages 371-390, Avignon, 30 Mai-03 Juin, 1988.

[167] H. Lâasri, B. Maître, and J. P. Haton. ATOME : Outil d'aide au développement de systèmes multi-experts. *Actes du* $6^{ème}$ *Congrès* AFCET-RFIA (RFIA-87), tome 2, pages 749-759, Antibes, 16-20 Novembre 1987.

[168] H. Lâasri, B. Maître, T. Mondot, F. Charpillet, and J. P. Haton. ATOME: A Blackboard Architecture with Temporal and Hypothetical Reasoning. In *Proceedings of the* 8^{th} *European Conference*

on Artificial Intelligence (ECAI-88), pages 5–10, Munich, W. Germany, August 01-05, 1988. (Extended version appeared as INRIA Research Report no. 855, June 1988).

[169] W. L. Lakin, J. A. H. Miles, and C. D. Byrne. *Blackboard Systems*, chapter "Intelligent Data Fusion for Naval Command and Control", pages 443–458. Addison-Wesley, 1988.

[170] J. S. Lark. *Module-Oriented Programming in* ABE*: Modules and Abstract Datatypes.* Technical report TTR-ISE-87-103b, Teknowledge, Inc., June 1987.

[171] J. L. Laurière. *Intelligence artificielle : Résolution de problèmes par l'homme et la machine.* Eyrolles, 1986.

[172] J. L. Laurière. Représentation et utilisation des connaissances - première partie : les systèmes experts. *Techniques et Sciences Informatiques*, 1(1):25–42, 1982.

[173] J. L. Laurière. Représentation et utilisation des connaissances - seconde partie : représentation des connaissances. *Techniques et Sciences Informatiques*, 1(2):109–133, 1982.

[174] D. Lenat, R. Davis, J. Doyle, M. Genesereth, I. Goldstein, and H. Schrobe. *Building Expert Systems*, chapter "Reasoning about Reasoning", pages 219–239. Addison-Wesley, Massachusetts, 1983.

[175] V. R. Lesser and D. D. Corkill. Functionally Accurate, Cooperative Distributed Systems. IEEE *Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–97, 1981.

[176] V. R. Lesser and D. D. Corkill. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*, 4(3):15–33, Fall 1983. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 18, pages 353-386, Addison-Wesley, 1988).

[177] V. R. Lesser, D. D. Corkill, and E. H. Durfee. *An Update on the Distributed Vehicle Monitoring Testbed.* Technical Report 87-111, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts, October 1987.

[178] V. R. Lesser, D. D. Corkill, J. A. Hernandez, and R. C. Whitehair. Goal Relationships in Blackboard Architectures. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 73–87, S^t Paul, Minnesota, August 24, 1988.

[179] V. R. Lesser and L. D. Erman. *Blackboard Systems*, chapter "A Restropective View of the Hearsay-II Architecture", pages 87–121.

Addison-Wesley, 1988. (Also appeared in A. Hanson and E. Riseman, editors, *Computer Vision Systems*, pages 790-800, Academic Press, New York, 1978).

[180] V. R. Lesser, J. Pavlin, and E. H. Durfee. Approximative Processing in Real-Time Problem-Solving. *AI Magazine*, 9(1):49–61, 1988.

[181] J. Levy. BBC: A Blackboard Generating System. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[182] H. Lieberman. *A Preview of Act1.* Technical Report AI Memo No. 625, MIT Artificial Intelligence Laboratory, June 1981.

[183] C. Ling, R. Dodhiawala, D. Nguyen, and T. Skillman. A Parallel VLSI Architecture for Blackboard Systems. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[184] B. J. Lippolt and H. Velthuijsen. BLONDIE*: A Blackboard Shell.* Memorandum 1404 DNL/86, Netherlands PTT, Dr. Nether Laboratories, November 1986.

[185] B. Maître. Coopération dans un univers multi-agents basée sur le modèle du blackboard : Etudes et réalisations — Contribution personnelle. Thèse de l'université de Nancy 1, Février 1989.

[186] B. Maître. *Extensions des fonctionnalités d'un progiciel de développement de systèmes experts.* Rapport de fin d'étude, Cognitech - Ecole Centrale de Lyon, Avril-Juin 1986.

[187] B. Maître and H. Lâasri. A Distributed Approach to Control Problem-Solving in Blackboard Systems. *Submitted to the Specialized Conference on Second Generation Expert Systems*, Avignon-89, Avignon, 1989.

[188] B. Maître and H. Lâasri. *Manuel d'Utilisation d'*ATOME *: Version 1.0.* Rapport Technique 87-T-099, CRIN/INRIA-Lorraine, Novembre 1987.

[189] B. Maître, H. Lâasri, and J. P. Haton. *A Blackboard-Based Shell for Multi-Level Knowledge Organization.* Technical Report 88-R-025, CRIN/INRIA-Lorraine, 1988.

[190] B. Maître, H. Lâasri, and T. Mondot. *Manuel d'Utilisation d'*ATOME *: Version 1.1.* Rapport Technique 87-T-110, CRIN/INRIA-Lorraine, Novembre 1987.

[191] B. Maître, H. Lâasri, T. Mondot, F. Charpillet, and J. P. Haton. Coordination de sources de connaissances opérant dans un univers incomplet et évolutif : Etudes et réalisations dans ATOME. *Soumis à la conférence spécialisée sur les systèmes experts de deuxième génération*, Avignon-89, Avignon, 1989.

[192] B. Maître, H. Lâasri, T. Mondot, F. Charpillet, and J. P. Haton. Knowledge Sources Coordination in an Incomplete and Evolutive Universe: Studies and Achievements in ATOME. *Submitted to the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[193] B. Maître, H. Lâasri, T. Mondot, and J. P. Haton. ATOME: Advanced Tool for Multi-Level Knowledge Organization. *2nd International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, Charleston, South Carolina, May 3-5, 1988.

[194] J. N. Maksym, A. J. Bonner, C. A. Dent, and G. L. Hemphill. Machine Analysis of Acoustical Signals. *Pattern Recognition*, 18:459–463, 1983.

[195] G. Mayer. Knowledge Clusters: Knowledge Source Organization with Integral Control for Blackboard Architectures. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[196] K. R. Mazer. Automated Interpretation of Sensor Data for Evaluating In-Situ Conditions. In *Proceedings of the 1st Conference on Artificial Intelligence in Engineering*, pages 861–886, Southampton, U. K., April 1986.

[197] D. McCracken. *A Production System Version of the* HEARSAY-II *Speech Understanding System*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, April 1978.

[198] N. Nandhakumar and J. K. Aggarwal. The Artificial Intelligence Approach to Pattern Recognition - A Perspective and Overview. *Pattern Recognition*, 18(6):383–389, 1985.

[199] H. P. Nii. *An Introduction to Knowledge Engineering, Blackboard Model, and* AGE. Technical Report HPP-80-29, Heuristic Programming Project, Computer Science Department, Stanford University, March 1980.

[200] H. P. Nii. Blackboard Systems (Part 1) - The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. *AI Magazine*, 7(2):38–53, 1986.

[201] H. P. Nii. Blackboard Systems (Part 2) - Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective. *AI Magazine*, 7(3):82–106, 1986.

[202] H. P. Nii. *Research on Blackboard Architectures at the Heuristic Programming Project.* Technical Report KSL-85-24, Knowledge Systems Laboratory, Computer Science Department, Stanford University, May 1985.

[203] H. P. Nii. CAGE *and* POLIGON*: Two Frameworks for Blackboard-Based Concurrent Problem Solving.* Technical Report KSL-86-41, Knowledge Systems Laboratory, Computer Science Department, Stanford University, April 1986.

[204] H. P. Nii. *Signal-to-Symbol Transformation: Reasoning in the* HASP/SIAP *Program.* Technical Report HPP-83-44, Heuristic Programming Project, Computer Science Department, Stanford University, Stanford, California, December 1983.

[205] H. P. Nii and N. Aiello. AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence (*IJCAI-79*)*, pages 645–655, Tokyo, Japan. 1979. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 12, pages 251-280, Addison-Wesley, 1988).

[206] H. P. Nii, N. Aiello, and J. Rice. *Blackboard Systems*, chapter "Frameworks for Concurrent Problem-Solving: A Report on CAGE and POLIGON", pages 475–501. Addison-Wesley, 1988.

[207] H. P. Nii and E. A. Feigenbaum. *Pattern-Directed Inference Systems*, chapter "Rule-Based Understanding Of Signals", pages 483–501. Academic Press, New York, 1978.

[208] H. P. Nii, E. A. Feigenbaum, J. J. Anton, and A. J. Rockmore. Signal-to-Symbol Transformation: HASP/SIAP Case Study. *AI Magazine*, 3(2):23–35, 1982. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 6. pages 135-157. Addison-Wesley, 1988).

[209] N. J. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.

[210] C. Pair, R. Mohr, and R. Schott. *Construire les algorithmes.* Dunod, 1988.

[211] H. D. Parunak. *Distributed Artificial Intelligence*, chapter "Manufacturing Experience with the Contract Net", pages 285–310. Pitman, 1987.

[212] H. E. Pattison, D. D. Corkill, and V. R. Lesser. *Distributed Artificial Intelligence*, chapter "Instantiating Descriptions of Organizational Structures", pages 59–96. Pitman, 1987.

[213] J. Pavlin. Predicting the Performance of Distributed Knowledge-Based Systems: A Modeling Approach. In *Proceedings of the 3rd National Conference on Artificial Intelligence (AAAI-83)*, pages 314–319, Washington, D. C., 1983.

[214] J. Pavlin and D. D. Corkill. Selective Abstraction of AI System Activity. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 264–268, Austin, Texas, 1984.

[215] G. Pearson. Mission Planning within the Framework of the Blackboard Model. In *Proceedings from Expert Systems in Government Conference*, McLean, Virginia, 1985. (Also appeared in R. S. Engelmore and A. Morgan, editors, *Blackboard Systems*, chapter 21, pages 433-442, Addison-Wesley, 1988).

[216] G. Pearson and D. Kuan. Mission Planning System for Autonomous Vehicle. In *Proceedings of the 2nd Conference on Artificial Intelligence Applications*, IEEE, Miami Beach, Florida, December 11-13, 1985.

[217] P. Raulefs. Towards a Blackboard Architecture for Real-Time Control of Dynamic Systems. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[218] D. Reynolds. "MUSE: A Toolkit for Embedded, Real-Time AI". In *Blackboard Systems*, pages 519–532, R. Engelmore and T. Morgan, editors, Addison-Wesley, 1988.

[219] J. P. Rice. POLIGON*: A System for Parallel Problem-Solving.* Technical Report KSL-86-19, Knowledge Systems Laboratory, Computer Science Department, Stanford University, April 1986.

[220] J. P. Rice. *The* POLIGON *User's Manual - Release 7.1.* Technical Report KSL-86-10, Knowledge Systems Laboratory, Computer Science Department, Stanford University, October 1987.

[221] E. Rich. *Intelligence Artificielle.* Masson, 1987.

[222] S. P. Roth, S. D. Tynor, and J. F. Gilmore. GEST - Development of a Blackboard Expert System Tool. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[223] M. D. Rychener and R. B. Alcantara. A Rule-Based Blackboard System. SIGART *Newsletter*, 92:101–102, April 1985.

[224] R. E. Seviora and P. Dasiewicz. A Blackboard-Based Robot Position Estimator. *Microprocessing and Microprogramming*, 18:89–96, 1986.

[225] G. Shafer and R. Logan. Implementing Dempster's Rule for Hierarchical Evidence. *Artificial Intelligence*, 33:271–298, 1987.

[226] S. A. Shafer. A Communications Database for Robot Navigation. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[227] D. D. Sharma and N. S. Sridharan. Knowledge-Based Real-Time Control: A Parallel Processing Perspective. In *Proceedings of the AAAI-88 Workshop on Blackboard Systems*, pages 163–168, S^t Paul, Minnesota, August 24, 1988.

[228] S. C. Sok. Evolutivité du logiciel. Thèse de 3ème cycle en Informatique, Université de Nancy 1, Septembre 1980.

[229] F. Fogelman Soulié. Applications des méthodes connexionnistes en intelligence artificielle. *Actes des* 11èmes *Journées francophones sur l'informatique (Architectures avancées pour l'intelligence artificielle)*, pages 101-116, Nancy, 12-13 Janvier 1989.

[230] D. S. Spain. Application of Artificial Intelligence to Tactical Situation Assessment. In *Proceedings of the 16th* EASCON, pages 457–464, 1983.

[231] S. N. Srihari, C. H. Wang, P. W. Palumbo, and J. J. Hull. Recognizing Address Blocks on Mail Pieces: Specialized Tools and Problem-Solving Architecture. *AI Magazine*, 25–40, Winter 1987.

[232] D. Sriram. DESTINY: A Model for Integrated Structural Design. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*. Seattle, Washington, July 13, 1987.

[233] R. O. Stenerson. Integrating AI into the avionics engineering environment. IEEE *Computer*, 88–91, February 1986.

[234] A. Tailor. MXA - A Blackboard Expert System Shell. In *Blackboard Systems*, pages 315–333, R. Engelmore and T. Morgan, editors, Addison-Wesley, 1988.

[235] S. Talukdar and E. Cardozo. Blackboards, Lateral Relations and Large-Scale Organizations. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[236] A. Terry. *Blackboard Systems*, chapter "Using Explicit Strategic Knowledge to Control Expert Systems", pages 159–188. Addison-Wesley, 1988.

[237] A. Terry. *The* CRYSALIS *Project: Hierarchical Control of Production Systems.* Technical Report HPP-83-19, Heuristic Programming Project, Computer Science Department, Stanford University, Stanford, California, May 1983.

[238] C. Thorpe, M. H. Herbert, T. Kanade, and S. A. Shafer. Vision and Navigation for the Carnegie-Mellon Navlab. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, May 1988.

[239] S. D. Tynor, S. P. Roth, and J. F. Gilmore. GEST: The Anatomy of a Blackboard Expert System Tool. In *Proceedings of the* 7^{th} *International Workshop on Expert Systems and their Applications (Avignon-87)*, pages 793–807, Avignon, May 13-15, 1987.

[240] W. K. Utt, J. B. Anderson, F. J. Bollag, and S. D. Nystrom. An Expert System for Data Reduction. In *Proceedings of the* 2^{nd} *Conference on* AI *Applications*, pages 120–124, Miami Beach, Florida, December 11-13, 1985.

[241] H. Velthijsen. *Manual for* BLONDIE - *An Expert System Shell with a Blackboard Architecture.* Intern Rapport 87 DNL/35, Netherlands PTT, Dr. Neher Laboratories, December 1987.

[242] H. Velthijsen. *Manual for* BLONDIE-II - *A Parallel Blackboard Shell.* Intern Rapport 87 DNL/36, Netherlands PTT, Dr. Neher Laboratories, December 1987.

[243] H. Velthuijsen. A Parallel Blackboard System for Robot Control. In *Proceedings of the* AAAI-87 *Workshop on Blackboard Systems*, Seattle, Washington, July 13, 1987.

[244] D. A. Waterman and F. Hayes-Roth. *Building Expert Systems*, chapter "An Investigation of Tools for Building Expert Systems", pages 169–215. Addison-Wesley, Reading, Massachusetts, 1983.

[245] T. E. Weymouth. *Using Object Descriptions in A Schema Network for Machine Vision.* PhD thesis. Department of Computer and Information Science, University of Massachusetts, Amherst. Massachusetts, May 1986.

[246] T. E. Weymouth, J. S. Griffith, A. R. Hanson, and E. M. Riseman. Rule-Based Strategies for Image Interpretation. In *Proceedings of the* 3^{rd} *National Conference on Artificial Intelligence (*AAAI-83*)*, pages 429–432, Washington, D. C., 1983.

[247] M A. Williams. *Blackboard Systems*, chapter "Hierarchical Multi-Expert Signal Understanding", pages 387–415. Addison-Wesley. 1988.

[248] L. A. Zadeh. A Simple View of the Dempster-Shafer Theory of Evidence and its Implication for the Rule of Combination. *AI Magazine*, 7(2):85–90, 1986.

[249] R. Zanconato. BLOBS - An Object-Oriented Blackboard System Framework for Reasoning in Time. In *Blackboard Systems*, pages 335–345, R. Engelmore and T. Morgan, editors, Addison-Wesley, 1988.

Index

acteurs 11
action 95, 124
activation opportuniste 136
activation séquentielle 136
agenda 34, 75, 143
agendas 22, 39
agent 1, 14
architecture d'ATOME 112
architecture de blackboard 15, 19, 235
architecture hétérogène 224
architecture homogène 224
attribut 95

base de données 258
blackboard 10, 59, 62, 71, 93, 95, 99
blackboard à long terme 155
blackboard de contrôle 38, 69
blackboard des buts 51
blackboard des données 53
blackboard du domaine 69
blackboard explicatif 208
blackboard global 122
blackboard local 122
blackboards 122
boucle de contrôle de base 115

coefficient d'importance 101
coefficient de vraisemblance 95
communication hybride 236
compétitives 32, 104
concurrence 11
concurrentes 21, 69
condition-action 21
conditions de rejet 76, 144
conflits 2, 11
connexionisme 259
contexte de travail 76, 100, 127
contrôle 11, 19
contrôle à base de blackboard 22, 69
contrôle explicite 37
contrôle global 35, 93
contrôle hiérarchique 22, 57, 93
contrôle hybride 22, 161
contrôle implicite 37
contrôle local 35, 93
contrôle procédural 22, 39
contrôle uniforme 161
coopération 11, 14
coopératives 21, 104
création d'une hypothèse 95
cycle 110, 111

déclaratif 21
data flow 11
dimension 79
dirigé par les événements 105, 136
dirigé par les buts 246
dirigé par les règles 105, 138

efficacité 85, 117
envoi d'informations 10, 236
espace du blackboard 79
événement 21, 23, 45, 59, 98, 124, 126
expertise hétérogène 9

facilité d'expression 85, 116
filtre 100, 129

flux de contrôle 25, 66
focalisation d'attention 35
focalisation de contrôle 35, 105, 106, 111
formalisme-ATOME 100, 232

gestionnaires 59

heuristiques 39, 72, 144, 147, 243
hiérarchie du blackboard 46
hiérarchique 102
hybride à multi-phases 26, 92, 121, 161
hypothèse 19, 41, 71, 95, 122

importance d'un événement 102
importance d'une hypothèse 102
indépendantes 21
instance de SC 51
interpréteur 75

lien 95
liste d'événements 93, 102, 104
liste locale d'événements 131

mécanisme de communication 10
mécanisme de contrôle 2, 10, 23, 31, 91
mécanisme explicatif 205
mémoire de travail 224, 225
méta-sources de connaissances 22, 34
modèle du blackboard 16, 19, 255
mode de fonctionnement 101, 108
mode de raisonnement 108
modification d'un attribut 95
modification d'un lien 95
modules de contrôle 37, 59
moniteur du blackboard 45
multi-agents 9, 13, 15, 255
multi-experts 4, 13
multi-SCs 91

négociation 12, 258
niveau d'abstraction 15, 41
niveau d'entrée 32
niveau de sortie 32
niveau du blackboard 24, 95
nœud 102, 126

objectif 72, 243
objet structuré 95
opportuniste 85, 142, 247
ordonnanceur 39, 51, 69, 76
ordre 0 92, 223
ordre 0+ 92, 223
ordre 1 92, 223

parallélisme 258
partage d'informations 10, 235
partie action 41
partie condition 41, 73
phases 80, 161
plan de contrôle 69
planificateur 53
précondition 73, 118, 126
priorité 147
procédural 21
propagation des événements 132

règle d'intégration 144, 147
règle d'une spécialiste 100
règle d'une tâche 104
règle de la stratégie 110
résolution de problème 1, 12, 21, 110, 1
résumé du blackboard 93, 102, 110
résumés des blackboards 136
rôle du contrôle 21, 32
raisonnement adaptatif 250
raisonnement approximatif 250, 253
raisonnement concurrent 249
raisonnement hypothétique 250, 253
raisonnement interruptible 249
raisonnement limité 249
raisonnement temporel 250, 253
remplacement d'un attribut 95
remplacement d'un lien 95
response-frame 43

société d'experts 11
solution courante 110

souplesse 85, 117
source de connaissances 9, 39, 99
sources de connaissances 21, 47
spécialiste 99
spécialistes 21, 93
stimulus-frame 43
stratégie 59, 71, 93, 102, 110, 243
stratégies de contrôle 39
structures de contrôle 23, 37, 59, 93, 102, 130
suppression d'une hypothèse 95
synthèse d'événements 134
système-ATOME 93
système d'IA 1, 4, 93, 205
système d'IA distribué 50
système multi-agents 1
système ouvert 85, 232

tâche 104, 243
tâches 64, 93, 102
tableau noir 10
temps réel 19, 248
transfert d'informations 2
trigger 73
type de contrôle 26, 37, 53, 83

uniformité 85
unité 79

variables de contrôle 147
voyageur de commerce 171

zone de communication 10, 21, 224
ATMS 253
ATOME 26, 38, 93, 223, 243
BB-1 16, 19, 38, 71, 243
CRYSALIS 19, 38, 62
DVMT 19, 39, 48
GBB 19, 78
HASP/SIAP 19, 38, 57
HEARSAY-II 16, 19, 37, 39
IROISE 92, 223
ISC 34, 51
ISP 143
SAGANE 92, 223
SCs de contrôle 38, 69, 102
SCs du domaine 38, 69
TMS 254

Résumé

Notre travail s'inscrit dans le cadre du projet SYCO du CRIN/INRIA-Lorraine. Ce projet concerne la réalisation de systèmes à base de connaissances et leur application à la compréhension.

Cette thèse, effectuée au sein de l'équipe RFIA (Reconnaissance des Formes et Intelligence Artificielle), présente le résultat de notre travail sur la résolution de problèmes nécessitant la coopération de plusieurs agents "intelligents". Nous traitons plus particulièrement le cas où cette coopération est réalisée par partage d'information à travers une zone de données commune aux différents agents (modèle du blackboard).

Notre travail a consisté à définir un mécanisme de contrôle à la fois efficace et souple pour gérer cette coopération. Après avoir énuméré et évalué les réalisations déjà existantes, nous présentons l'approche que nous avons implantée dans ATOME : outil d'aide au développement de systèmes multi-agents.

L'architecture d'ATOME est organisée en quatre plans hiérarchiques :

1. les *blackboards* qui, à chaque instant de la résolution du problème, contiennent les résultats fournis par les différents agents. Ces blackboards contiennent également les données initiales du problème;
2. les *spécialistes* qui renferment les connaissances du domaine et constituent les agents du système;
3. les *tâches* qui fournissent un *contrôle local* au système en gérant un sous-ensemble de spécialistes;
4. une *stratégie* qui fournit un *contrôle global* au système en gérant l'ensemble des tâches.

Pour terminer, nous présentons un bilan de nos travaux et nos réflexions sur le modèle du blackboard.

Mots clés :

architecture de blackboard, coopération, coordination, mécanisme de contrôle, résolution de problème, sources de connaissances, univers multi-agents.