Contribution à l'étude d'un agent rationnel : spécification en logique intentionnelle et implantation

Jean-Pierre Muller

THESE

Présentée à

L'UNIVERSITE SCIENTIFIQUE, TECHNOLOGIQUE ET MEDICALE DE GRENOBLE
et à
L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir le grade de
DOCTEUR de l'INPG
(arrêté ministériel du 5 juillet 1984)
«Informatique»

par

Jean-Pierre MULLER

ooooo

Contribution à l'étude d'un agent rationnel : Spécification en logique intensionnelle et implantation

ooooo

Thèse soutenue le 11 Décembre 1987 devant la commission d'examen.

G. VEILLON	Président
J. SIFAKIS	Rapporteur
M. BORILLO	
Ph. JORRAND	Examinateurs
J.C. LATOMBE	
A. LUX	

UNIVERSITE Joseph FOURIER (GRENOBLE I)

Président de l'Université :
M. PAYAN Jean Jacques

Année Universitaire 1987 - 1988

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées

LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du Solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BLANCHI J.Pierre	STAPS
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie

HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
NEMOZ Alain	Thermodymamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	EEA. IUT 1

PROFESSEURS de 2ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
PEFFEN René	Métallurgie IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

PROFESSEURS DE PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital sud
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie-Obstétrique	Hopital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Georges LESPINARD

Année 1988

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Marc	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FOGGIA Albert	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GREVEN Hélène	IUFA
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOUBERT Jean-Claude	ENSPG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSIEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PARIAUD Jean-Charles	ENSEEG
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
RAMEAU Jean-Jacques	ENSEEG
RENAUD Maurice	UFR PGP
ROBERT André	UFR PGP
ROBERT François	ENSIMAG
SABONNADIERE Jean-Claude	ENSIEG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VEILLON Gérard	ENSIMAG
ZADWORNY François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Madeleine
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
GHIBAUDO Gérard
HAMAR Sylvaine
HAMAR Roger
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
PASTUREL Alain
PLA Fernand
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S
Directeurs de recherche 1ère Classe

CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
LANDAU Ioan
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
COURTOIS Bernard
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GUELIN Pierre
JOUD Jean-Charles
KLEITZ Michel
KOFMAN Walter
KAMARINOS Georges
LEJEUNE Gérard
LE PROVOST Christian
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MUNIER Jacques
PIAU Monique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
WACK Bernard

Personnalités agrées à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

E.N.S.E.E.G

CHATILLON Christian
HAMMOU Abdelkader
MARTIN GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G

BOREL Joseph

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

ROWE Alain

E.N.S.I.M.A.G

COURTIN Jacques

E.F.P.

CHARUEL Robert

C.E.N.G

CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIB Maurice
VINCENDON Marc

Laboratoires extérieurs

C.N.E.T

DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

Résumé

Cette étude porte sur l'architecture globale d'un agent rationnel. Nous appelons agent un système autonome, c'est-à-dire capable d'agir sans intervention extérieure. Nous faisons de plus l'hypothèse qu'il est rationnel, c'est-à-dire que l'ensemble de ses activités extérieures, perceptives et déductives sont justifiables par ses buts.

Cette étude comprend deux parties, la spécification et l'implantation.

La première partie est un essai de spécification des notions essentielles pour un agent rationnel à savoir, le raisonnement sur les connaissances, le raisonnement sur les actions et la capacité de faire quelque chose, et l'organisation de ses buts. Nous utilisons les logiques intensionnelles comme outil formel pour l'expression de cette spécification.

La deuxième partie décrit le système CHAOS qui implante une partie des idées que nous avons développées dans la spécification. Cette implantation montre la faisabilité d'une approche globale d'un agent rationnel.

Mots-Clés : agent rationnel
autonomie
logiques intensionnelles

Abstract

This dissertation studies global architecture of a rational agent. An agent is defined as an autonomous system which means a system able to act without external control. An additional hypothesis concerns its rationality which means all its external, perceptive and deductive activities are goal directed.

This study is composed of two parts, the specification and the implementation.

The first part proposes a specification of the essential notions for a rational agent : the reasoning about knowledge, the reasoning about action and the ability to act, and the goal structure. We use the intensional logics as a formal tool to express this specification.

The second part describes the system CHAOS which implements a subset of the ideas developped in the specification. This implementation shows the feasability of a global approach to a rational agent.

Keywords : rational agent
autonomy
intensional logics

Remerciements

Je tiens à remercier:

Le Professeur Gérard VEILLON, directeur de l'ENSIMAG, pour avoir accepté de présider mon jury de thèse.

Le Professeur Jean-Claude LATOMBE, mon directeur de thèse, pour m'avoir fait confiance en acceptant que je traite ce sujet.

M. Joseph SIFAKIS, directeur de recherche du CNRS, pour avoir accepté d'être rapporteur et dont l'influence sur mon essai de formalisation fut aussi brève qu'inestimable. Les imprécisions de la formalisation restent les miennes.

M. Mario BORILLO, directeur de recherche du CNRS, pour avoir également accepté d'être mon rapporteur et pour l'intérêt qu'il a porté à mon travail.

M. Augustin LUX, chargé de cours à l'ENSIMAG, pour sa lecture patiente et constructive des différentes versions de ma thèse et pour m'avoir appris les règles de la rédaction scientifique.

M. Philippe JORRAND, directeur du LIFIA, pour avoir accepté d'être membre du jury.

Mlle Isabelle URECH, une amie très chère, pour les jours et les nuits qu'elle a passés à relire ma thèse et à en corriger le français, ainsi que pour son soutien moral dont l'importance est inestimable.

Tous les gens du LIFIA, pour leur support technique et surtout scientifique par les longues discussions que nous avons eues au cours de ces années de recherche.

Tous les gens du groupe d'Intelligence Artificielle du SRI, pour les deux mois que j'ai passé là-bàs et qui ont été l'occasion d'échanges fructueux.

Mes parents, et tous mes amis, pour m'avoir supporté dans les jours difficiles, et m'avoir redonné courage lorsque j'ai douté de moi.

Jean-Pierre Müller

Sommaire

Liste des figures

Chapitre 1

Introduction

1.1 Problématique

L'Intelligence Artificielle concerne la compréhension de la nature du comportement intelligent. Le but ultime est une théorie de l'intelligence qui tienne compte d'un tel comportement et guide la création d'êtres artificiels qui en soient capables. L'Intelligence Artificielle est donc à la fois une science et une ingénierie.

Une caractéristique essentielle du comportement de tout être intelligent est l'*autonomie*. C'est dans ce cadre que se place notre travail.

Supposons que nous ayons un robot jardinier. Il est muni d'un certain nombre de capteurs lui permettant d'analyser son environnement et notamment le temps qu'il fait et l'état du jardin. D'autre part, il est capable de se déplacer et de manipuler des objets tels qu'une lance d'arrosage ou un rateau. La présence de capteurs et d'effecteurs sont les conditions physiques de l'autonomie.

Si notre robot est vraiment autonome, nous ne devons pas sans arrêt lui dire ce qu'il doit faire. Cela suppose qu'il est capable de déterminer de lui-même ce qu'il a à faire en fonction des circonstances. Son rôle consiste dans notre cas à empêcher le gazon de se dessécher et à le maintenir à une certaine hauteur. Nous pouvons encore lui imposer un entretien personnel pour qu'il ne se dégrade pas. Les tâches qui lui incombent déterminent deux types de comportement :

- un comportement d'observation pour vérifier régulièrement l'évolution de l'humidité et de la hauteur du gazon. Il cherche donc de lui-même à savoir si il y a quelque chose à faire ou non.

- un comportement d'exécution pour arroser ou tondre le gazon lorsque le robot s'aperçoit que c'est nécessaire.

Dans ces deux types de comportement, la perception et l'exécution sont étroitement entremêlées. En effet, pour observer l'état du gazon lorqu'il est à l'intérieur, notre robot doit sortir ou regarder par la fenêtre. Pour pouvoir regarder par la fenêtre, il doit aller jusqu'à la fenêtre et peut-être ouvrir les volets si ceux-ci sont fermés. De la même façon, pour pouvoir arroser le gazon, il doit regarder son environnement pour décider comment arroser et vérifier si tout se passe bien.

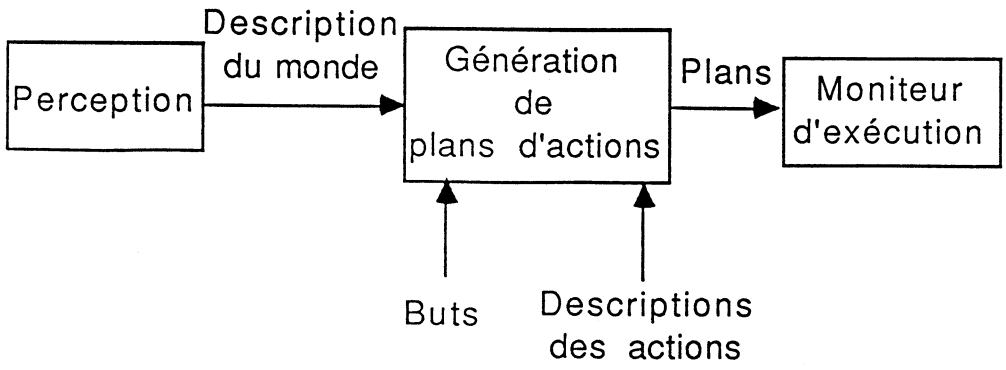

Figure 1.1: Architecture classique

Pour résumer, notre robot est autonome pour les raisons suivantes :

- il possède les moyens de se mettre en rapport avec son environnement par un ensemble de capteurs et d'effecteurs.
- il oriente ses activités vers la réalisation des tâches qui lui sont assignées.
- il est capable de s'adapter aux modifications de l'environnement.

Sa capacité d'adaptation suppose qu'il est capable de coordonner perception et exécution afin d'une part de savoir que faire et comment le faire et d'autre part de vérifier les effets de ses actions pour réagir à des surprises éventuelles.

Un sous-problème important est la production d'un tel comportement. Elle suppose l'existence de *buts* et d'*actions* permettant de satisfaire les buts. Plus fondamentalement, elle suppose des *connaissances* sur la capacité de chaque action à résoudre tel ou tel but et sur la capacité du robot à réaliser effectivement ces actions. Dans ce cadre, la *génération de plans d'actions* offre un ensemble de techniques permettant :

- la mise en correspondance des actions et des buts,
- la construction de plans exécutables.

La génération de plans d'actions classique, telle qu'elle est décrite par Nilsson dans [34], est considérée comme une boîte noire à laquelle nous fournissons en entrée des buts, une description du monde actuel et une description des actions et qui nous fournit en sortie un plan exécutable permettant d'atteindre ces buts. Ces considérations nous fournissent la structure illustrée par la figure 1.1. Dans ce schéma, la génération de plans est indépendante de la perception chargée de construire la description du monde d'une part et de l'exécution effective des plans d'autre part.

Cette architecture ne peut donc convenir pour construire notre système autonome pour les raisons suivantes :

- elle n'est pas conçue pour engendrer ses propres buts,

- elle ne permet pas la coordination étroite entre la perception et l'exécution,
- toute modification de l'environnement et tout échec à l'exécution demande une replanification.

Confronté à la complexité et à la mouvance du monde réel, ce type d'architecture devient vite inutilisable car le système passe son temps à replanifier. De plus, il est impossible d'obtenir une description complète d'un univers de complexité réaliste. Cette même complexité rend également irréaliste une planification complète d'une tâche car la prévision de ce qui peut se passer devient d'autant moins fiable que nous nous éloignons dans le futur. Il est donc nécessaire d'intégrer le contrôle de la perception et de l'exécution au processus de génération de plans d'actions.

Dans ce cadre, nous proposons une structure globale régie par ce que nous appelons le *principe de rationalité*. En vertu de ce principe, le choix des actions par le système est justifiable en termes des buts qu'elles permettent de réaliser. Il en est ainsi des actions sur le monde extérieur mais aussi des actions perceptives et déductives. Les actions perceptives et déductives sont justifiées par des buts d'acquisition de connaissances, ce qui nécessite le raisonnement explicite sur les connaissances du système. Ce principe nous permet d'intégrer la perception au processus de génération de plans d'actions.

Pour assurer la capacité d'adaptation de la génération de plans d'actions aux modifications des connaissances qu'il possède du monde extérieur et de lui-même, il faut étudier :

- les interactions connaissance-but pour que la génération adapte ses buts aux modifications de l'environnement,
- les interactions connaissance-action pour que le raisonnement sur les actions prennent en compte les connaissances nécessaires à la réalisation des actions,
- les interactions action-connaissance pour que la génération de plans s'adapte à l'effet des actions sur ses connaissances et puisse intégrer le contrôle d'exécution.

Comme base de cette étude, nous proposons une notion de but dont les interactions avec les connaissances permettent de modéliser comment l'échec ou le succès de l'exécution d'une action d'une part et ce que le système perçoit d'autre part influencent le processus de génération de plans d'actions.

Nous abordons cette structure sur le plan théorique en proposant une formalisation intégrant le raisonnement sur les connaissances de la génération de plans permettant ainsi d'exprimer les diverses interactions. Nous décrirons alors le prototype que nous avons implanté.

1.2 Démarche

Dans ce paragraphe, nous décrivons brièvement la démarche que nous désirons suivre. Elle est illustrée par la figure 1.2. Le point de départ de cette démarche

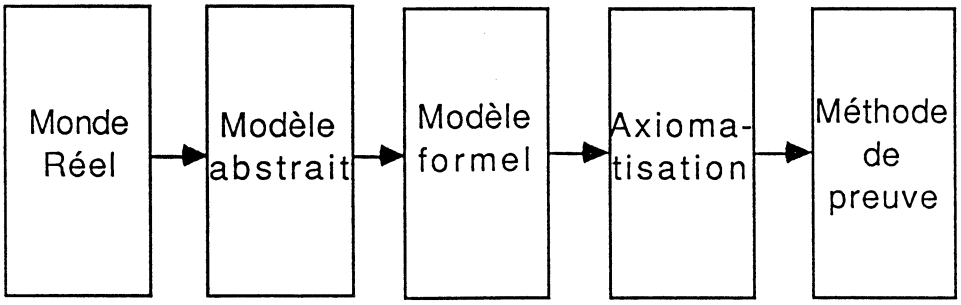

Figure 1.2: Méthode

est le *monde réel* dont nous supposons l'existence. Le *modèle abstrait* est l'idée que nous nous faisons du monde réel et notamment l'ensemble des objets que nous voulons décrire ainsi que les hypothèses simplificatrices. Il s'exprime par des concepts intuitifs. Le *modèle formel* est la traduction dans un formalisme des notions décrites dans le modèle abstrait. Il s'exprime à l'aide d'objets mathématiques. L'*axiomatisation* est l'expression dans un langage des propriétés du modèle formel. Enfin, des *méthodes de preuve* doivent être mises au point pour pouvoir tester la formalisation.

A cette démarche s'ajoute la nécessité de s'assurer que l'axiomatisation est équivalente au modèle formel c'est-à-dire que tout ce qui est productible formellement correspond à ce modèle et réciproquement. En logique, ce sont les propriétés de complétude et consistance respectivement.

Nous suivons cette démarche tout au long de cette thèse. Toutefois, nous n'allons pas plus loin que l'axiomatisation et notamment, nous ne nous attachons pas à prouver l'équivalence entre notre axiomatisation et le modèle formel pour un certain nombre de raisons. La raison principale est que nous cherchons non seulement à modéliser une partie du comportement intelligent mais aussi à construire un prototype opérationnel. Aller jusqu'au bout de la formalisation et construire un démonstrateur de théorèmes nous auraient certainement permis de valider notre formalisation mais ne nous auraient en aucun cas fourni un programme produisant le comportement souhaité. Néanmoins, nous élaborons une justification intuitive par comparaison avec des formalismes existants. D'autre part, nous décrivons un prototype qui fonctionne sur les principes introduits.

1.3 Agent rationnel

La première étape de la démarche consiste en l'élaboration du modèle abstrait. Nous prenons en considération dans le monde réel les êtres capables d'un comportement intelligent, qu'ils soient naturels ou artificiels. Nous appelons ces êtres, des *agents*.

Il nous faut maintenant spécifier ce que nous entendons par comportement in-

telligent. Ce phénomène est particulièrement difficile à décrire puisqu'il résulte en fait de tout un ensemble de caractéristiques qui, mises ensemble, constituent ce que nous appelons l'*intelligence* chez l'être humain. Etant donné que l'Intelligence Artificielle cherche à réaliser des systèmes capables d'exhiber un tel comportement, la littérature s'est attachée à cerner un certain nombre de principes qui sous-tendent ce comportement, et parmi eux la notion de *rationalité* introduite notamment par Newell [33].

1.3.1 Principe de rationalité

Nous définissons le principe de rationalité de la façon suivante :

Définition 1 *Un agent rationnel cherche à effectuer les actions les plus pertinentes pour atteindre ses buts étant donné la connaissance qu'il possède de ses actions et de l'environnement.*

Un agent qui obéit à ce principe sera appelé un *agent rationnel.*

Nous retrouvons dans cette définition le vocabulaire introduit dans le paragraphe 1.1. Toutefois, il nous faut expliciter ce que nous entendons par action pertinente. En effet, la définition de la rationalité dans [33] ne l'introduit pas.

La pertinence d'une action est fonction :

- de la contribution de l'action à la satisfaction des buts,
- d'un jugement sur l'effectivité de l'action du point de vue de son coût, de ses conséquences futures, du coût du raisonnement sur ses conséquences possibles, etc.

Ces deux conditions introduisent deux aspects du concept de rationalité. Le comportement est dit rationnel pour deux raisons :

- parce que les actions faites contribuent à la satisfaction des buts,
- parce que parmi toutes les actions possibles pour réaliser les buts, les actions exécutées sont les meilleures.

A partir de cette définition de la rationalité, nous pouvons esquisser les éléments d'une structure fonctionnelle qui la mette en oeuvre. Nous avons les deux composantes suivantes :

- une composante *intentionnelle* qui gère la mise en correspondance des actions et des buts (première nuance du concept de rationalité),
- une composante *décisionnelle* qui juge l'effectivité des actions (deuxième nuance du concept de rationalité).

L'ensemble de ces composantes est lié au monde extérieur par les actions perceptives et les actions sur le monde extérieur. Nous appelons ces dernières des *actions externes*.

Par rapport à cette décomposition, nous pouvons réexprimer le sujet de notre thèse comme étant l'étude des interactions entre les modifications de la connaissance qu'un agent possède du monde extérieur et la composante intentionnelle. Puisque ce sont ces interactions qui nous intéressent dans cette thèse, le comportement ne sera rationnel que dans la mesure où les actions effectuées contribueront à satisfaire des buts. Par contre, les actions choisies pourront ne pas être les meilleures.

1.3.2 Architecture croyance-désir-intention

Un certain nombre de chercheurs en Intelligence Artificielle (notamment Konolige [18], Bratman [5] et Nilsson [35]) proposent d'articuler l'architecture d'un agent autonome autour de trois notions :

- les *croyances* B qui portent sur le monde extérieur et l'agent lui-même,
- les *désirs* D qui sont les conditions que l'agent cherche à satisfaire,
- les *intentions* I qui sont des désirs que le système a explicitement choisi de réaliser.

Les désirs et les intentions sont deux aspects de la notion de but.

Des désirs peuvent être contradictoires. Par exemple, notre robot jardinier peut simultanément désirer arroser le gazon et le tondre alors qu'il est impossible de satisfaire conjointement ces deux désirs. Il peut également désirer se rendre à un endroit et en même temps désirer éviter un obstacle ce qui va le dévier de sa route.

Les intentions sont le résultat d'une délibération. Par exemple notre robot peut formuler l'intention de tondre le gazon. Cette intention exclut la possibilité d'avoir simultanément l'intention de l'arroser mais le robot peut très bien formuler l'intention d'arroser après avoir fini de tondre.

L'ensemble des intentions qui contribuent à satisfaire un but donné constitue un *plan* pour ce but. Un plan peut être partiel. Par exemple, l'intention de tondre le gazon peut être décomposée le moment venu en de nouvelles intentions comme celles d'aller chercher la tondeuse, de l'amener sur la place de travail, de mettre de l'essence, etc. Ce lien entre les intentions et les plans est formalisé notamment par Pollack dans [36].

1.3.3 Synthèse

Les notions introduites dans l'architecture croyance-désir-intention permettent de prendre en compte les deux aspects du concept de rationalité ainsi que nous allons

le montrer en décrivant les deux composantes que nous avons proposées en termes de ces notions.

La composante intentionnelle est une fonction :

$$B \times D \times I \rightarrow D$$

Etant donné les connaissances que le système possède du monde extérieur, elle assure la correspondance des actions et des buts (désirs ou intentions). Par exemple, si notre robot désire (ou a l'intention) de tondre le gazon, il désirera prendre la tondeuse électrique ou la tondeuse à essence. La distinction entre désirs et intentions est une différence de priorité pour la composante intentionnelle. Cette modélisation de la composante intentionnelle suppose qu'il existe des désirs a priori à partir desquels tous les autres sont dérivés. Nous appelons ces désirs les *finalités* du système. Pour notre robot jardinier, la finalité est d'entretenir le gazon.

La composante décisionnelle est une fonction :

$$B \times D \times I \rightarrow I$$

C'est le processus de délibération qui choisit parmi les désirs ceux qu'il faut satisfaire. Ce choix dépend des connaissances du système et des intentions déjà formulées. Les intentions jouent ici un rôle de filtre. Ce rôle a été mis en évidence par Bratman [3,4].

L'ensemble de ces deux composantes assure la construction de plans permettant de satisfaire les buts selon les critères de rationalité que nous avons introduits mais n'assure pas l'exécution de ces plans. Pour cela, il faut prendre en compte une autre propriété de certaines intentions. Les intentions *causent* l'exécution des actions. Cette propriété importante a été mise en évidence par certains philosophes et notamment par Searle dans [42,43]. L'ensemble de ces considérations est résumé dans la figure 1.3. C'est sur cette architecture que nous nous basons dans cette thèse.

1.4 Finalisation

Comme nous l'avons vu, les désirs fournissent les raisons pour que l'agent agisse. Nous disons qu'ils *motivent* les activités de l'agent. Puisque nous intégrons perception, exécution et génération de plans d'actions, l'agent n'agit pas seulement sur son environnement extérieur mais aussi sur l'état de ses connaissances par la perception et la déduction. Par exemple, si notre robot veut que le gazon soit à une certaine hauteur, il désire savoir quelle est la hauteur du gazon. De même, si la hauteur est trop grande, il désire couper le gazon. Un désir est donc à l'origine de deux autres désirs :

- le *désir de savoir* si le désir est satisfait ou non,
- le *désir d'être capable* de faire une action permettant de satisfaire son désir si il n'est pas déjà satisfait.

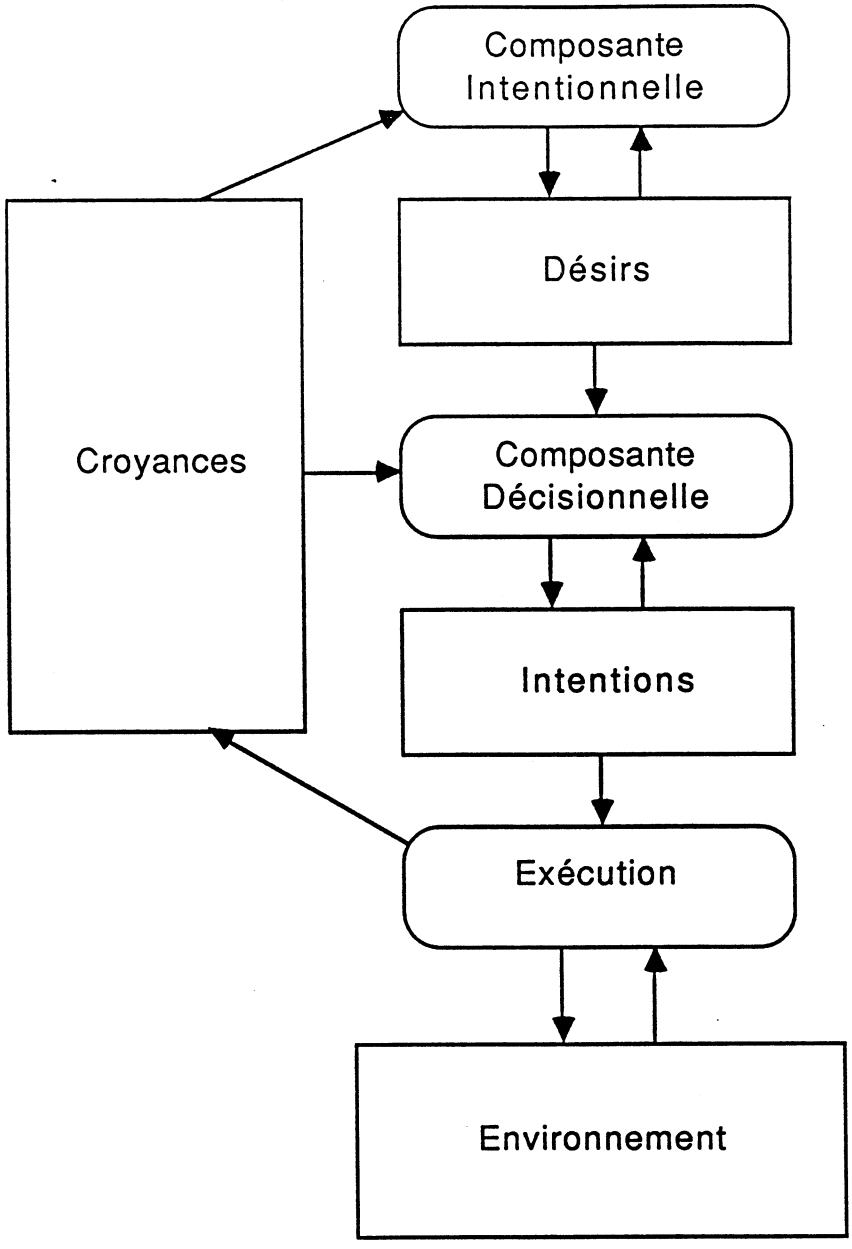

Figure 1.3: Architecture d'un agent rationnel

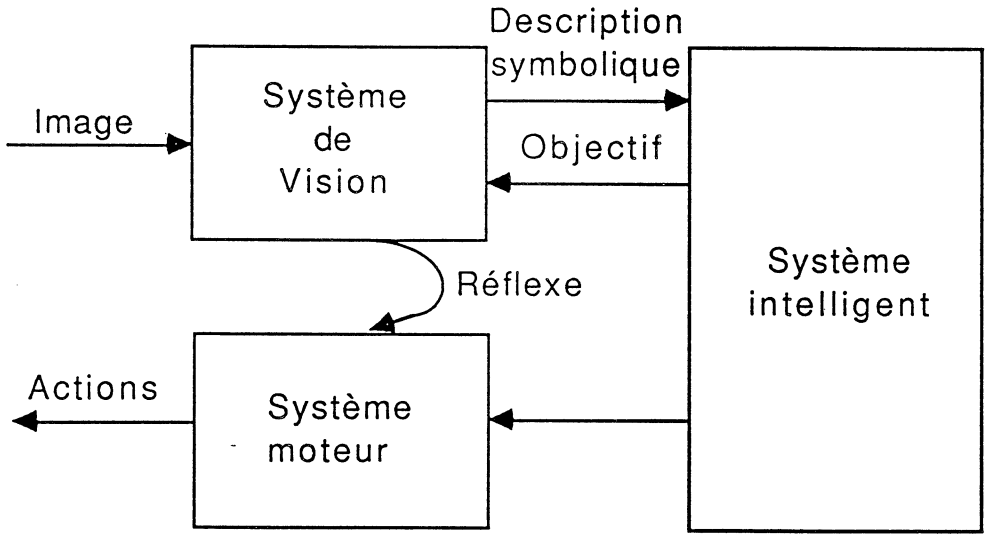

Figure 1.4: La vision dans un système intelligent

Nous pouvons ainsi motiver les comportements d'observation et d'exécution.

Cette modélisation a des conséquences importantes sur la façon dont nous prenons en compte la perception. Dans sa thèse, Lux [23] propose une architecture d'un système intelligent afin d'y situer la place de la Vision par Ordinateur (voir figure 1.4). Ce schéma est extensible à la perception en général. Il s'en dégage deux types de perception :

- la perception réflexe,
- la perception générale.

Dans le premier cas, la perception est directement reliée au système moteur. Dans le second cas, le système de perception est soumis au contrôle de ce que Lux appelle les objectifs et que nous avons appelé les désirs. En d'autres termes, la perception est alors motivée.

Dans notre travail, étant donné la place prépondérante de la composante intentionnelle, nous considérons que la perception est entièrement sous son contrôle.

Une première conséquence est l'élimination de la perception réflexe. En effet, seule la perception finalisée entre dans le cadre que nous nous sommes fixé. Notons que la perception réflexe n'est pas incompatible avec notre approche. Elle sort cependant de notre sujet.

Une deuxième conséquence est l'introduction de la perception sous la forme d'actions. En effet, la perception aura pour but d'obtenir des connaissances sur l'état du monde extérieur. Elle sera donc activée de la même façon qu'une action est exécutée c'est-à-dire selon les besoins du système. C'est pourquoi nous introduisons les actes perceptifs, planifiables comme toute autre action.

Une troisième conséquence concerne le processus de déduction qui prend place sur la base des données perçues pour déterminer certaines informations. En effet, les informations sont acquises pour tester les prémisses qui permettront de tirer un certain nombre de conclusions. Le désir d'obtenir une de ces conclusions motive donc à la fois la vérification des prémisses et l'exécution des pas d'inférence. Pour intégrer cette activité à l'ensemble, les déductions doivent également être considérées comme des actions.

La perception est alors modélisable par des actions perceptives planifiables au même titre que les actions sur le monde extérieur. Les avantages de cette approche sont nombreux :

- la satisfaction d'un désir peut aussi bien comprendre la planification d'actions perceptives pour pouvoir agir que d'actions externes pour pouvoir percevoir. Pour reprendre notre robot, il peut regarder si il n'y a pas d'obstacles pour aller quelque part et de la même façon il peut aller dans le jardin pour pouvoir vérifier si il est dans l'état souhaité.

- l'interprétation des données perçues pose moins de problèmes puisque l'agent se focalise sur les signes qui lui fournissent les informations qu'il cherche.

Le premier point montre comment nous cherchons à prendre en compte la coordination entre la perception et l'exécution que nous considérons comme essentielle pour un agent autonome.

1.5 Contenu du rapport

La discussion qui précède nous permet de cerner les activités du système que nous voulons explicitement motiver, à savoir :

- les actions sur le monde extérieur,

- les actions perceptives,

- les actions déductives.

D'autre part, la décomposition d'un désir dans les désirs de savoir si le désir est satisfait ou non et d'être capable de faire une action permettant de satisfaire ce désir nous pose deux problèmes importants :

- que veut dire savoir quelque chose,

- que veut dire être capable de quelque chose.

Nous traitons le premier problème dans le chapitre suivant en introduisant une modélisation des croyances d'un agent. Etre capable de faire quelque chose est formalisé dans le chapitre 3 qui traite du raisonnement sur les actions. Le chapitre 4 modélise les désirs et montre comment motiver le comportement d'un agent

rationnel par coordination des actions perceptives, déductives et externes. Finalement, le chapitre 5 présente le système CHAOS qui implante concrètement les idées exposées dans les chapitres précédents et montre la faisabilité de notre approche. Nous concluons en dégageant quelques principes que nous jugeons essentiels pour assurer l'autonomie d'un agent artificiel.

Chapitre 2

Modélisation des connaissances

2.1 Introduction

Il est essentiel d'avoir une théorie de la connaissance pour deux raisons :

- un système intelligent doit avoir des connaissances qui représentent le monde extérieur et lui-même,
- un système intelligent doit être capable de déterminer ce qu'il sait et ce qu'il ne sait pas.

Le mot "connaissance" possède plusieurs sens. Une connaissance peut être soit une information qui représente quelque chose, soit une relation entre une information et l'objet auquel l'information se rapporte. Par exemple, nous pouvons dire qu'un agent sait que le ciel est bleu pour dire qu'il possède une information sur la couleur du ciel. Nous pouvons également dire qu'il sait que le ciel est bleu parce qu'il possède cette information et qu'elle est vraie.

Nous appelons *croyance* une connaissance dans l'acception d'une information qui représente quelque chose. Nous utilisons également le mot "représentation". Nous devons encore préciser ce que nous entendons par représenter. Lorsque nous avons la croyance que le ciel est bleu, nous ne voulons pas dire que nous avons une structure symbolique dont le sens est que le ciel est bleu. Par exemple, une phrase écrite sur un morceau de papier ne représente rien puisqu'elle n'a un sens que lue par quelqu'un. Il en est de même des formules logiques. Par contre la pensée humaine représente puisqu'elle est intrinsèquement à propos de quelque chose. Une telle propriété est importante dès lors que nous cherchons à construire un agent autonome. En effet, c'est la capacité de représentation qui lui permet de se mettre en rapport avec le monde extérieur.

Pour bien faire la distinction entre les croyances et les objets symboliques auxquels nous pouvons attribuer un sens, nous appelons ces derniers des *descriptions*. Ainsi, la branche de l'Intelligence Artificielle qui s'appelle "représentation des connaissances" devrait plutôt s'appeler "description des connaissances" dans notre méthodologie puisqu'elle construit essentiellement des objets symboliques

(réseaux sémantiques ou formules logiques) dont le sens est donné par les concepteurs des systèmes ou les experts du domaine.

Nous disons également qu'une croyance *représente* alors qu'une description *signifie* ou *a un sens.*

Nous réservons le mot *connaissance* à la description de la relation entre une représentation d'un objet et l'objet représenté. La description de cette relation dépend de la possibilité d'observer conjointement une croyance et l'objet de cette croyance. Si nous nous plaçons en observateur extérieur, nous disons qu'un agent sait que le ciel est bleu si et seulement s'il possède cette croyance sur la couleur du ciel et si le ciel est effectivement bleu. Si nous nous mettons à la place de l'agent possédant cette croyance, nous n'avons plus accès à l'objet lui-même puisque la croyance résulte de la perception du monde extérieur. Lorsqu'un agent dit de lui-même qu'il sait que le ciel est bleu, il ne décrit pas "objectivement" cette relation mais émet un jugement sur la validité de sa représentation. Ainsi il dit qu'il sait quelque chose s'il est sûr de la véracité de son information. Le terme croyance change également de sens puisque ce n'est plus la description objective d'une présence d'informations mais de nouveau un jugement sur la véracité de cette information. Ainsi un agent dit qu'il croit quelque chose s'il n'est pas sûr que cette croyance corresponde à la réalité.

Le problème est d'attribuer une structure formelle à la notion de croyance afin de donner un sens précis à des phrases telles que "l'agent croit que le ciel est bleu". Les logiciens et les chercheurs en Intelligence Artificielle ont proposé un certain nombre de modèles :

- la sémantique des mondes possibles décrit les croyances d'un agent par l'ensemble des mondes que l'agent estime possibles dans un monde donné. Nous disons que l'agent croit que le ciel est bleu si la proposition "le ciel est bleu" est vraie dans tous les mondes possibles.

- l'approche syntaxique décrit les croyances par l'ensemble des phrases syntaxiquement dérivables par un moteur d'inférence. Nous disons que l'agent croit que le ciel est bleu si la phrase "le ciel est bleu" appartient à cet ensemble.

- l'approche des automates situationnels décrit les croyances par une relation entre l'environnement et l'état d'un automate. Nous disons que l'agent croit que le ciel est bleu si l'automate est dans un état donné chaque fois que le ciel est bleu.

Finalement notre approche décrit les croyances par la donnée partielle de la dénotation qui lie les descriptions aux objets dont l'agent est capable de parler. Nous disons que l'agent croit que le ciel est bleu si la phrase "le ciel est bleu" dénote la valeur vraie.

Dans toutes ces formalisations, nous disons que l'agent sait que le ciel est bleu s'il croit que le ciel est bleu et que le ciel est bleu.

La discussion qui suit porte sur les avantages et inconvénients que possèdent chacun de ces formalismes dans l'expression et la résolution des problèmes suivants :

- le problème de la représentation,
- le problème de l'incomplétude,
- le problème de l'expression.

En ce qui concerne le premier problème, nous devons bien distinguer entre :

- le problème du sens : comment pouvons-nous donner un sens à un objet ?
- le problème de la représentation : comment quelque chose peut-elle représenter ?

Le premier problème a été largement étudié par les logiciens et les linguistes. Le deuxième problème fait plutôt partie du domaine de la philosophie. Il nous paraît cependant essentiel si nous voulons construire un agent autonome. En effet, la première propriété que doit posséder un tel agent est de pouvoir se mettre en rapport avec le monde extérieur et donc de posséder des croyances dans le sens que nous avons esquissé plus haut.

Afin de formaliser le raisonnement sur la connaissance, nous devons tenir compte des limitations des agents humains ou artificiels. Parce que leurs ressources sont limitées, ils possèdent les propriétés suivantes :

- leur connaissance est partielle. C'est pourquoi il est nécessaire de raisonner à propos de ce que sait ou ne sait pas un agent.
- ils sont logiquement incomplets. Ils ne peuvent pas déduire certaines choses parce qu'ils ne connaissent pas les règles pour y parvenir.
- ils sont déductivement incomplets. Même avec les bonnes règles, ils sont incapables de déduire toutes les conclusions possibles car ils sont limités en place et en temps.

Alors que le premier point est purement sémantique, les deux derniers points font une référence explicite à un processus de calcul sous-jacent qui prend place dans la tête de l'agent.

Le dernier problème concerne les descriptions que nous pouvons faire des croyances d'un agent. Dans le contexte où nous nous plaçons, nous nous intéressons plutôt à exprimer qu'un agent *sait si* le ciel est bleu qu'à exprimer qu'un agent *sait que* le ciel est bleu. De la même façon nous nous intéressons à exprimer qu'un agent *sait quel* est le numéro de téléphone de quelqu'un plutôt qu'à exprimer que l'agent *sait que* le numéro de téléphone est le 245.67.78. Le formalisme doit donc pouvoir exprimer naturellement :

- que l'agent *sait si*,
- que l'agent *sait quoi*,
- que l'agent *sait que*.

Nous allons d'abord présenter les autres approches, à savoir :

- la sémantique des mondes possibles,
- l'approche syntaxique,
- l'approche des automates situationnels.

et en discuter les limites relativement aux trois problèmes que nous venons de citer. Nous présentons ensuite notre formalisation.

2.2 Les autres approches

2.2.1 La sémantique des mondes possibles

La sémantique des mondes possibles est un modèle formel très général qui a été développé par Kripke [19] pour traiter les notions de nécessité et de possibilité. Ces deux notions sont représentées respectivement par les opérateurs $\Box$ et $\Diamond$. Ainsi, la phrase $\Box p$ se lit: il est nécessaire que p. La logique classique étendue avec les opérateurs $\Box$ et $\Diamond$ s'appelle une *logique modale.* Ces opérateurs s'appellent des *opérateurs modaux.*

L'introduction de tels opérateurs a posé de graves problèmes aux philosophes (voir les discussions dans [22]) pour leur attribuer une sémantique. En effet, la phrase:

$$\Box p$$

peut être vraie ou fausse indépendamment de la valeur de vérité de p, contrairement à $\neg p$ par exemple qui est vrai si p est faux et faux si p est vrai. Autrement dit, les opérateurs modaux ne sont pas des *fonctions* de vérité. De tels opérateurs sont appelés *opérateurs intensionnels*[1].

Le modèle formel proposé par Kripke ne s'applique pas seulement à la nécessité et à la possibilité mais à toute notion exprimable à l'aide d'opérateurs intensionnels. Nous l'utilisons notamment pour formaliser le raisonnement sur les actions et sur les désirs ainsi que nous le verrons dans les chapitres suivants. Le lecteur trouvera une analyse exhaustive des logiques intensionnelles dans [32].

Le premier, Hintikka [16] a proposé l'utilisation de la sémantique des mondes possibles pour formaliser le raisonnement sur les croyances. Pour cela, il introduit l'opérateur épistémique $[B]$ qui se lit "l'agent croit que". La logique classique augmentée de cet opérateur est appelée *logique épistémique.* Intuitivement, les croyances d'un agent sont représentées par l'ensemble des mondes que l'agent estime possibles étant donné la représentation partielle qu'il possède de la réalité. Cette sémantique ne fait aucune hypothèse sur la nature de ces représentations.

[1] le terme intensionnel avec un s est utilisé par opposition à l'extentionalité qui caractérise toute fonction. Il ne faut pas le confondre avec intentionnel avec un t qui, dans la langue courante, qualifie ce qui est fait volontairement, ni même avec l'intentionalité des philosophes qui caractérise ce qui représente.

Le cas propositionnel

Pour présenter l'approche des mondes possibles, nous suivons la démarche habituelle de la logique, à savoir la présentation du langage suivi de:

- la théorie des modèles dans laquelle nous introduisons respectivement:
  - la structure de modèle,
  - la définition de l'interprétation des phrases du langage relativement à la structure de modèle,
  - la notion de validité.
- la théorie de la déduction.

Le langage Dans le cas propositionnel, le *langage* est donné par l'ensemble des formules construites à partir d'un ensemble fini de variables propositionnelles :

$$p, q, r, \ldots$$

à l'aide des connectifs logiques :

$$\neg, \wedge, \vee, \rightarrow, \ldots$$

et de l'opérateur épistémique $[B]$ de la façon suivante :

- les variables propositionnelles sont des formules bien formées (fbfs).
- si P et Q sont des fbfs, alors les formules :

  $$\neg P, P \wedge Q, P \vee Q, P \rightarrow Q, \ldots$$

  sont des fbfs.
- si P est une fbf, la formule épistémique :

  $$[B]P$$

  (qui se lit : l'agent croit que P) est une fbf.
- il n'y a pas d'autres fbfs.

Nous avons utilisé des majuscules pour désigner les formules du langage.

La sémantique La sémantique des mondes possibles repose sur la donnée d'une relation sur les mondes possibles. Intuitivement, cette relation décrit les mondes que l'agent estime possibles lorsqu'il est dans un monde donné. Autrement dit, uRv si et seulement si v est un monde possible quand l'agent est dans le monde u.

Formellement, Kripke[19] introduit une *structure de modèle* sous la forme d'un triplet $M = (S, \phi, R)$ où :

S : est un ensemble de mondes possibles.

ϕ : est une assignation de valeurs de vérité aux variables propositionnelles pour chaque monde possible :

$$\phi : P \times S \rightarrow \{Vrai, Faux\}$$

C'est la fonction de *dénotation.*

R : est une relation binaire sur les mondes possibles.

Une *interprétation* est le choix d'un M et d'un monde s de S dans lequel nous nous plaçons.

Nous définissons la sémantique des phrases du langage en caractérisant la notion de modèle. Rappelons que les *modèles* d'une formule sont les interprétations qui la rendent vraie. Pour cela nous utilisons la relation $\models$ (qui se lit "est un modèle pour").

Nous avons la sémantique suivante :

- $(M, s) \models P$ si et seulement si $\phi(P, s) = vrai$,
- $(M, s) \models P \wedge Q$ si et seulement si $(M, s) \models P$ et $(M, s) \models Q$ et ainsi de suite pour les différents connectifs logiques,
- $(M, s) \models [B]P$ si et seulement si $(M, t) \models P$ pour tous les t tels que (s, t) appartient à R.

Le dernier point de la définition exprime qu'un agent croit que p si et seulement si p est vrai dans tous les mondes possibles pour l'agent dans le monde s. Notons que p n'est pas forcément vrai dans le monde choisi ainsi que l'illustre la figure 2.1. Dans cette figure, p est vrai dans tous les mondes possibles et également dans le monde choisi. Récursivement, l'agent peut choisir un des mondes possibles et raisonner sur les mondes qu'il estimerait possibles s'il était dans ce monde. Ceci permet de donner un sens à des formules de la forme $[B][B]p$ qui exprime que l'agent croit qu'il croit que p.

Finalement, nous avons la notion usuelle de *validité.* Une formule p est valide si et seulement si elle est vraie dans toutes les interprétations c'est-à-dire si chaque interprétation est un modèle de la formule. Nous notons la validité par la formule :

$$\models p$$

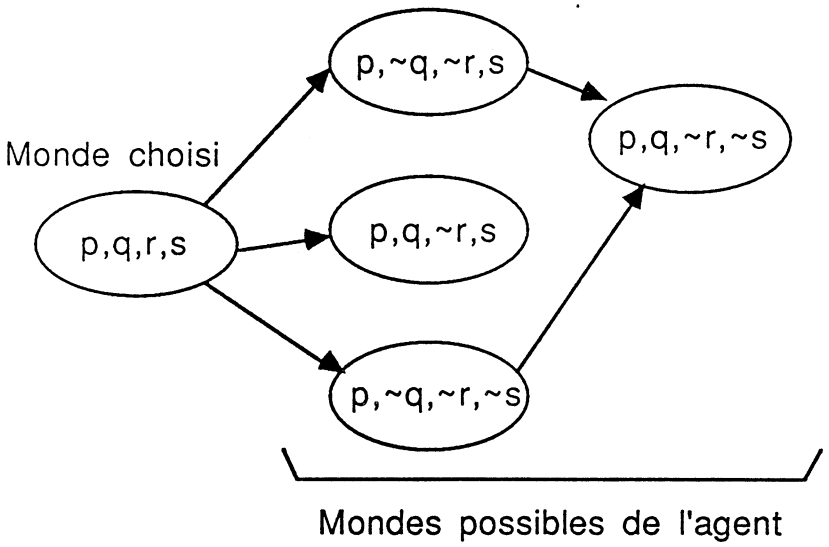

Figure 2.1: La sémantique des mondes possibles

L'axiomatique L'*axiomatisation* de cette formalisation permet de rendre explicite les propriétés des croyances que nous pouvons exprimer avec ce formalisme. Les axiomes sont les suivants :

1. les tautologies du calcul propositionnel
2. $([B]p \wedge [B](p \rightarrow q)) \rightarrow [B]q$

Nous avons également les deux règles d'inférence suivantes :

1. si $\vdash p$ et $\vdash p \rightarrow q$ alors $\vdash q$
2. si $\vdash p$ alors $\vdash [B]p$

où le symbole $\vdash$ exprime la dérivabilité de la formule[2]. Cette axiomatisation définit le système K.

Montrons intuitivement que l'axiomatisation correspond à la théorie des modèles présentée précédemment. La deuxième règle d'inférence est une conséquence directe de la notion de validité. En effet, si p est valide alors nous avons que :

$$\forall M \forall u, (M, u) \models p$$

d'où nous pouvons déduire que :

$$\forall v \forall u, (vRu) \rightarrow (M, u) \models p$$

[2] $\vdash$ est une notion syntaxique alors que $\models$ est une notion sémantique. La complétude et la consistance permettent d'établir des liens entre ces deux notions. Le système axiomatique est complet si et seulement si $\models p$ implique $\vdash p$. Il est consistant si et seulement si $\vdash p$ implique $\models p$

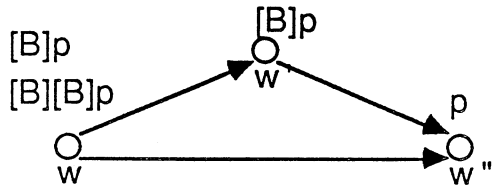

Figure 2.2: Axiome d'introspection positive

donc par définition :

$$\forall v, (M, v) \models [B]p$$

Nous nous apercevons par l'axiome 2 que l'agent sait toutes les conséquences de ce qu'il sait. Donc par l'axiome 1 et par la règle d'inférence 2, un agent est *omniscient.*

Nous pouvons caractériser davantage les croyances d'un agent en introduisant les axiomes suivants :

- l'axiome de connaissance : $[B]p \rightarrow p$
- l'axiome d'introspection positive : $[B]p \rightarrow [B][B]p$
- l'axiome d'introspection négative : $\neg[B]p \rightarrow [B]\neg[B]p$

Le premier axiome fait de la croyance d'un agent une connaissance puisque tout ce qu'il croit est vrai. Dans ce cas, l'opérateur épistémique se note $[K]$. Le deuxième et le troisième axiome affirment respectivement que l'agent sait quand il sait et qu'il sait quand il ne sait pas. Le système K auquel nous ajoutons l'axiome de connaissance s'appelle T. Le système T plus l'introspection positive nous permet d'obtenir $S4$. Enfin le système $S4$ auquel nous ajoutons l'axiome d'introspection négative donne $S5$.

Nous pouvons montrer que ces axiomes ne se déduisent pas des axiomes précédents. En effet, dans notre théorie des modèles nous avons posé une relation R quelconque. Or, à ces axiomes correspondent des contraintes sur R.

Ainsi, l'axiome de connaissance est vrai si R est *réflexive.* Intuitivement, si nous imposons que quel que soit le monde choisi s dans lequel nous nous plaçons, s est un monde possible du point de vue de l'agent, alors ce qu'il croit est forcément vrai dans le monde choisi. Ceci modélise la notion de connaissance que nous avons introduit dans le paragraphe 2.1.

Si nous considérons l'axiome d'introspection positive, la relation R doit être *transitive.* En effet, $[B]p$ est vrai dans s si p est vrai dans tous les mondes accessibles directement de s. D'autre part $[B][B]p$ est vrai si p est vrai dans tous les mondes accessibles de tous les mondes accessibles à partir de s c'est-à-dire accessibles par un chemin de longueur 2 dans le graphe des mondes possibles (voir figure 2.2). Si p est vrai dans tout monde accessible par un chemin de longueur 1 implique qu'il est vrai dans tout monde accessible par un chemin de longueur 2 alors la relation est transitive. En effet, il ne suffit pas que la relation soit dense, ce qui impliquerait

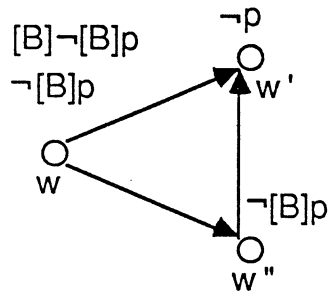

Figure 2.3: Axiome d'introspection négative

l'existence d'au moins un chemin de longueur 2 mais pas de tous. Donc $S4$ exige que la relation soit réflexive et transitive.

Enfin, pour l'axiome d'introspection négative la relation R doit être *euclidienne*. Si dans un monde s, $[B]p$ est faux, alors il existe un monde accessible de s, disons t dans lequel p est faux (voir figure 2.3). Si nous voulons par l'implication, que dans tout monde t' accessible de s, $[B]p$ soit faux, alors on doit avoir tRt', donc la relation doit être euclidienne. Dans $S5$, la relation R devient une relation d'équivalence sur les mondes possibles.

Quantification

Voyons maintenant ce qu'il advient du formalisme si nous étendons notre logique à la logique du premier ordre.

Nous ajoutons au langage les variables:

$$x, y, z, \ldots,$$

les termes de la forme:

$$f(t_0, \ldots, t_n), n \geq 0$$

où $t_0, \ldots, t_n$ sont des termes ou des variables, et les prédicats de la forme:

$$p(t_0, \ldots, t_n), n \geq 0$$

où $t_0, \ldots, t_n$ sont des termes ou des variables.

Les fbfs sont définies récursivement de la façon suivante:

- les prédicats sont des formules bien formées (fbfs)
- si P et Q sont des fbfs, alors les formules :

$$\neg P, P \wedge Q, P \vee Q, P \rightarrow Q, \ldots$$

sont des fbfs.

- si P est une fbf, les formules:

  $$\forall x P$$

  et

  $$\exists x P$$

  sont des fbfs.

- si P est une fbf, la formule épistémique :

  $$[B]P$$

  (qui se lit : l'agent croit que P) est une fbf.

- il n'y a pas d'autres fbfs.

Il faut alors modifier la structure de modèle de façon à introduire un domaine de discours D_s pour chaque monde possible s. Soit D l'union de tous les D_s, ϕ doit alors attribuer à chaque terme du langage un élément de D. Rappelons que ϕ dépend de chaque monde possible. Une interprétation est toujours le choix d'un des mondes possibles. En conséquence, son domaine de discours est considéré comme l'ensemble des objets qui existent dans le monde choisi.

L'introduction des quantificateurs dans cette formalisation pose plusieurs problèmes que nous exposons ici et dont nous expliquons les solutions. Pour comprendre les problèmes d'interprétation de l'opérateur épistémique, comparons les deux phrases suivantes :

$$\exists x([B]p(x))$$

$$[B]\exists x(p(x))$$

Dans le deuxième cas il suffit que $p(x)$ soit vrai dans chaque monde possible pour au moins un objet dans ce monde possible. Ce n'est pas forcément toujours le même. Par contre, dans le premier cas il faut que $p(x)$ soit vrai dans tous les mondes possibles pour le même objet. Sémantiquement, il faut que cet objet appartienne à l'intersection de tous les D_s. Syntaxiquement, il faut s'assurer que le terme t tel que $[B]p(t)$, dénote le même objet dans tous les mondes possibles. Le problème est de caractériser formellement ce cas sinon la quantification à travers l'opérateur modal n'a pas de sens. Il n'est effectivement pas suffisant que x puisse être remplacé par un terme quelconque parce qu'un même terme peut désigner des objets différents dans des mondes possibles différents.

Pour résoudre ce problème, nous introduisons une collection de constantes dont nous garantissons qu'elles ont la même dénotation dans tous les mondes possibles. C'est-à-dire que :

$$\forall u \forall v, \phi(cst, u) = \phi(cst, v)$$

Nous appellons ces constantes des ***désignateurs rigides***. Ces constantes spéciales sont souvent identifiées avec les ***noms standard*** (notamment par Moore dans [30]) qui sont des constantes dont nous connaissons le sens. Cependant, si les noms

standard sont effectivement des désignateurs rigides, les désignateurs rigides ne garantissent pas que nous savons de quoi nous parlons mais seulement que nous parlons de la même chose. Nous pouvons le voir par la formule ci-dessus qui affirme l'identité des dénotations mais ne dit pas quelle est la dénotation de la constante.

Un autre problème est l'attribution d'une valeur de vérité à un prédicat qui porte sur un individu qui n'existe pas dans tous les mondes possibles. Il y a plusieurs façons de résoudre ce problème :

1. rendre identiques tous les domaines de discours,
2. rendre les prédicats indéfinis pour certaines valeurs,
3. étendre l'assignation des valeurs de vérité de façon à attribuer $faux$ pour les individus qui n'existent pas dans le monde considéré.

La première solution satisfait la formule de Barkan :

$$\forall x([B]p(x)) \rightarrow [B]\forall x(p(x))$$

ainsi que son inverse :

$$[B]\forall x(p(x)) \rightarrow \forall x([B]p(x))$$

En effet, dans ce contexte, ces deux formules expriment respectivement que l'agent ne peut parler que des objets du monde choisi (mais pas forcément de tous) et que l'agent connaît tous les objets du monde choisi (mais pas uniquement ceux-là). Les deux phrases conjointes expriment alors que l'agent sait des choses à propos de tous les objets du monde choisi et seulement ceux-ci. Ce cas se produit lorsque tous les domaines de discours coïncident. Nous perdons donc une certaine souplesse. Notamment, si nous considérons que le monde choisi est le monde réel dans un état donné, un agent ne peut pas parler d'objets qui n'y existent pas.

La seconde solution a le désavantage d'introduire une troisième valeur de vérité. Quant à la troisième, elle a l'avantage de garder une grande généralité sans pour cela introduire de complications. C'est la solution choisie notamment par Kripke[19].

Critique

Un premier défaut de cette formalisation est l'omniscience. Dans l'introduction nous avons posé que les croyances d'un agent doivent être incomplètes. Ce modèle formel ne permet donc pas d'en tenir compte. Fagin et Halpern[11] proposent de pallier à l'hypothèse d'omniscience introduite par cette formalisation en ajoutant à la structure de modèle M un ensemble A de formules décrivant soit les phrases dont l'agent est capable de parler, soit les phrases à propos desquelles l'agent peut dire quelque chose en un temps déterminé. Ainsi nous tenons effectivement compte des incomplétudes citées plus haut en introduisant un aspect syntaxique. Cependant, il faut alors un mécanisme qui puisse construire à l'avance l'ensemble des formules auxquelles l'agent aura le temps de répondre, ce qui est peu réaliste.

Un autre défaut de ce formalisme est son indépendance de la représentation des croyances. En effet, si la sémantique des mondes possibles est attrayante, notamment pour les problèmes de référence des noms, il est difficile d'imaginer comment implanter les mondes possibles spécifiant les croyances du système. La solution généralement adoptée est la description des connaissances par des structures symboliques (réseaux sémantiques ou formules logiques). Ces structures ne représentent rien (au sens du paragraphe 2.1) dans la mesure où leur sens n'appartient qu'au concepteur. Cependant, il est possible de formaliser ce que veut dire savoir p pour un agent possèdant un ensemble d'assertions $base(u)$ dans un état du monde u. Il suffit de dire que l'agent sait p si p est vrai dans tous les modèles de $base(u)$ ce qui revient à définir la relation R de la façon suivante :

$$uRv \textit{ si et seulement si } v \models base(u)$$

Si l'agent sait p, il doit agir en accord avec sa connaissance. Or p peut très bien ne pas être codé explicitement dans $base(u)$. Nous sommes alors obligés soit de supposer qu'un agent peut réagir à des connaissances implicites ce qui est pour le moins curieux, soit de dériver toutes les conséquences ce qui est irréalisable en pratique.

L'approche syntaxique cherche à corriger ces défauts en reposant explicitement sur les structures symboliques et les capacités inférencielles d'une base de connaissances.

2.2.2 L'approche syntaxique

La structure de modèle

Cette approche a été étudiée par Konolige [17]. Les croyances d'un agent ne sont plus représentées par un ensemble de mondes possibles mais par un *sous-système des croyances* (belief subsystem). Il est constitué :

- d'axiomes,
- de règles d'inférence,
- d'une stratégie de contrôle.

ainsi qu'il est montré dans la figure 2.4. Ce sous-système définit implicitement un ensemble de croyances par l'ensemble des théorèmes qu'il peut dériver. Appelons cet ensemble B. Nous supposons que les assertions de B sont munies d'une sémantique mais cette sémantique n'est pas prise en compte dans la formalisation.

Quelques hypothèses sont nécessaires pour cerner cet ensemble :

- les connaissances sont consistantes : nous ne pouvons pas avoir p et $\neg p$.

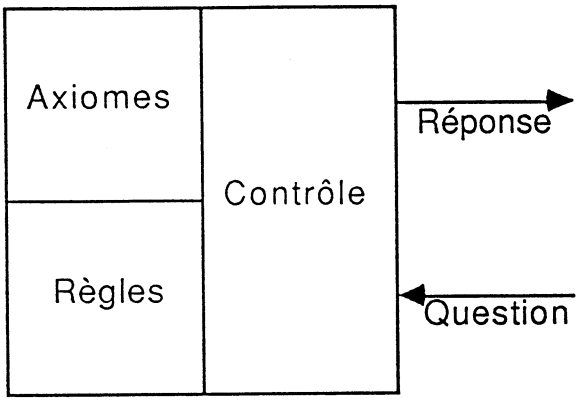

Figure 2.4: Le sous-système des croyances

- le système est déductivement clos : si $p_1, \ldots, p_n$ sont dans le système des croyances et la règle $p_1, \ldots, p_n \vdash q$ y est aussi, alors q est dans le système. Ceci impose que la stratégie de contrôle soit complète[3].
- les connaissances peuvent être partielles : il n'est pas nécessaire d'avoir p ou $\neg p$.

Nous avons également des contraintes sur les règles d'inférence. Elles doivent être :

effectives : l'applicabilité d'une règle est calculable

locales : le nombre de prémisses est fixe et fini

correctes : la règle ne contredit pas la sémantique du langage de l'agent

La formalisation

Dans cette approche, nous devons distinguer deux langages :

- le langage du système déductif : c'est-à-dire le langage interne à l'agent dans lequel sont exprimés les axiomes et les règles d'inférence.
- le méta-langage que nous utilisons pour parler des croyances de l'agent.

Le langage interne appelé L_0 doit être un langage du premier ordre. L'interprétation de ce langage est une valuation (une attribution de valeurs de vérité) standard respectant la sémantique des connectifs logiques et des quantificateurs. Nous supposons donc qu'il existe une fonction de dénotation ϕ qui définit notamment la validité des règles d'inférence. Cependant, seules les phrases du langage nous intéressent.

[3]Ici la complétude est une complétude syntaxique, c'est-à-dire que la stratégie est capable de dériver tous les théorèmes possibles. Ceci ne veut pas dire que les théorèmes couvrent tout ce qui est vrai ce qui correspondrait à la complétude logique.

Pour construire le méta-langage, il faut d'abord étendre le langage L_0 en associant à chaque constante a la formule $\bullet a$ dont nous verrons la sémantique plus tard. Le langage résultant s'appelle $L_0^\bullet$. Le méta-langage L^B nous permet de parler des connaissances de l'agent. Il est défini par les règles suivantes :

- L^B contient L_0,
- si p est une formule de $L_0^\bullet$ alors $[K]p$ est une formule de L^B.

Nous ne pouvons pas avoir d'opérateurs épistémiques emboîtés contrairement au cas des mondes possibles. Enfin, nous ne pouvons utiliser les constructions avec l'opérateur $\bullet$ qu'à l'intérieur d'une formule épistémique.

Si les formules de L_0 sont exclusivement closes (sans variable libre), la sémantique du langage est relativement simple. Une interprétation de L^B est une valuation respectant la sémantique des connectifs logiques et des quantificateurs. Elle assigne aux formules de la forme $[K]p$ la valeur *vrai* si la formule p fait partie de l'ensemble B des croyances de l'agent et la valeur $faux$ si la formule p n'en fait pas partie.

Si nous faisons l'hypothèse que les règles d'inférence sont complètes, c'est-à-dire que nous pouvons dériver tout ce qui est vrai étant donnée la sémantique de L_0, nous obtenons les mêmes propriétés que pour le modèle des mondes possibles, c'est-à-dire que nous avons :

- l'axiome: $[K]p \wedge [K](p \rightarrow q) \rightarrow [K]q$,
- la règle d'inférence: si $\vdash p$ alors $\vdash [K]p$.

Cependant, la possibilité d'avoir des règles incomplètes constitue une généralisation par rapport à la sémantique des mondes possibles.

Quantification

Les choses deviennent plus compliquées lorsque nous cherchons à quantifier par-dessus un opérateur épistémique. Prenons de nouveau le cas des deux phrases suivantes :

$$\exists x([K]p(x))$$
$$[K]\exists x(p(x))$$

La dernière phrase est vraie si cette phrase existe telle quelle dans l'ensemble des croyances B. Par contre, la première phrase exprime qu'il existe un objet à propos duquel l'agent croit (ou sait) que p est vrai. Cette phrase est donc satisfaite s'il existe dans le sous-système des croyances une phrase $P(c)$ où c dénote l'objet voulu. Pour pouvoir l'exprimer, il faut pouvoir identifier qu'une constante désigne un objet donné. Cette identification est formalisée par une fonction ν qui fait correspondre aux éléments du domaine de discours les noms que l'agent utilise pour les désigner. Ces noms sont appelés des ***constantes identifiables*** par Konolige. En utilisant cette fonction ν, nous disons que $\exists x([K]p(x))$ si et seulement s'il existe un individu k tel que $p(\nu(k))$ est dans le sous-système des croyances de l'agent.

Cette fonction ν possède des propriétés intéressantes notamment par rapport à la fonction de dénotation ϕ. D'abord, ν est un inverse partiel de ϕ, par exemple :

$$\phi(3+4) = 7\textit{(le nombre)}$$

$$\nu(7) = 7\textit{(le symbole)} \neq 3+4$$

Ceci est dû au fait que la fonction de dénotation porte sur des phrases du langage et 7 est syntaxiquement différent de $3+4$. Les constantes pour lesquelles ν est l'inverse de ϕ sont appelées des *noms standard.*

Les caractéristiques que l'on peut donner à ν dépendent du modèle abstrait que nous cherchons à formaliser. ν peut être partielle si nous voulons qu'il existe des objets à propos desquels l'agent ne sait rien. ν peut également être une relation si des objets différents pour l'agent peuvent en fait être le même individu pour un observateur extérieur.

Nous pouvons définir l'expression $\bullet a$ à partir de ν. En effet, $\bullet a$ est utilisé pour construire la constante identifiable qui correspond à l'objet dont on veut parler. Ce n'est donc rien d'autre que $\nu(\phi(a))$.

Cette vision des choses donne une interprétation différente de la formule de Barkan :

$$\forall x([K]p(x)) \rightarrow [K]\forall x(p(x))$$

ainsi que de son inverse :

$$[K]\forall x(p(x)) \rightarrow \forall x([K]p(x))$$

La quantification à l'extérieur de l'opérateur modal est interprétée dans les mondes possibles comme portant sur l'ensemble des objets du monde choisi ainsi que nous l'avons vu précédemment. Elle exprime donc quelque chose sur les objets dont l'agent peut parler. Dans le cas syntaxique décrit par Konolige, la formule de Barkan inverse exprime que la fonction ν est totale et la formule de Barkan affirme que si la fonction ν est totale alors elle couvre tous les objets dont l'agent peut parler.

Critique

Si cette approche permet de tenir compte des incomplétudes citées plus haut, les connaissances d'un agent ne représentent toujours pas. En effet, la fonction ν est l'attribution par l'observateur d'une sémantique à certains noms que le système manipule et non pas une sémantique attribuée par le système lui-même.

2.2.3 L'approche des automates situationnels

Rosenschein [39] propose de définir objectivement les connaissances d'un agent par une relation entre l'état interne de l'agent et l'état de l'environnement. L'agent et l'environnement sont modélisés par des automates d'états finis. La notion d'état trouve donc un sens très précis. Nous appelons cette approche celle des automates situationnels (situated-automata).

Le modèle

Soit l'automate $M = (S, \Sigma, A, \delta, \lambda, s_0)$ tel que :

- S est un ensemble d'états,
- Σ est un ensemble d'entrées (stimuli),
- A est un ensemble de sorties (actions),
- $\delta : S \times \Sigma \rightarrow S$ est la fonction de transition,
- $\lambda : S \rightarrow A$ est la fonction de sortie,
- $s_0 \epsilon S$ est l'état initial.

Si le monde est dans l'état w, nous pouvons le décrire par des conditions $\phi(w)$ organisées dans un treillis tel que $\phi \sqsubseteq \phi'$ si et seulement si ϕ' est plus général que ϕ. Nous distinguons ϕ_0 comme étant la condition la plus forte que satisfait le monde lorsque l'automate est dans l'état s_0.

Les événements perçus sont liés aux modifications de l'environnement et donc aux changements des conditions associées. Nous appelons ϕ/σ la plus forte post-condition après que l'entrée σ ait été perçue. Nous pouvons étendre cette définition à $\overline{\sigma}$ qui décrit une séquence d'entrées. $\phi/\overline{\sigma}$ est alors la plus forte postcondition après que la séquence $\overline{\sigma}$ ait été perçue.

Les *conditions connaissables* par l'automate sont les plus fortes postconditions pour chacune des séquences d'entrées possibles. Notamment les conditions qui ne sont postconditions d'aucune séquence d'entrées ne peuvent pas être connues de l'automate. Donc toutes les conditions exprimables par un observateur extérieur ne sont pas forcément connaissables par l'automate.

A chaque état s, nous pouvons associer l'ensemble L_s des séquences d'entrées qui y mènent à partir de l'état initial. Nous pouvons également construire la condition du monde qui correspond à L_s et qui n'est rien d'autre que la disjonction des postconditions associées à chaque séquence de L_s. Cette condition peut être vue comme l'information que possède le système de son environnement quand il est dans l'état s. La relation R dans la sémantique des mondes possibles peut alors être définie de la façon suivante : uRv si l'automate est dans le même état s quand le monde est dans l'état u ou l'état v, c'est-à-dire si u et v satisfont la même postcondition associée à L_s. Il est facile de vérifier que R ainsi définie rend la logique définissant la connaissance du système isomorphe à $S5$. En effet, R est une relation d'équivalence. Autrement dit, l'agent sait toutes les conséquences de ce qu'il sait, il sait quand il sait et quand il ne sait pas, etc. Toutefois, cette omniscience ne porte pas sur toutes les conditions mais seulement sur celles qui sont connaissables.

Critique

Nous obtenons un système qui objectivement représente le monde dans lequel il est. Il ne possède pas de structures symboliques qui codent explicitement cette information. Avec cette sémantique, un système est "capable de savoir" sans avoir à possèder un "langage de la pensée" dont l'existence est une des hypothèses les plus universellement admises dans la communauté de l'Intelligence Artificielle. Cependant cette formalisation possède plusieurs inconvénients :

- la nécessité d'avoir une description complète de l'automate et de l'environnement,
- la nécessité d'avoir des systèmes synchrones.

De plus, s'il est possible d'abstraire des connaissances d'un automate existant, il est difficile d'imaginer comment synthétiser un automate ayant à posséder certaines connaissances du monde dès lors qu'il devient un peu complexe.

Le résultat qui nous intéresse ici est la caractérisation d'un contenu sémantique par une relation entre l'agent et l'environnement. En effet, cette caractérisation nous fournit une condition sous laquelle nous pouvons dire que quelque chose (ici l'état d'un automate) *représente* le monde extérieur.

2.3 Notre approche

2.3.1 Introduction

L'approche des mondes possibles comme l'approche syntaxique ont l'inconvénient de déboucher sur des réalisations informatiques des croyances d'un système dont la sémantique est externe au système. En conséquence, une même description dans un système donné et dans un monde donné peut posséder des informations différentes selon l'interprétation que l'utilisateur ou l'observateur lui donne. Notre solution est de décrire explicitement la notion de dénotation. Etant donné que les connaissances de l'agent sont partielles, la fonction de dénotation est elle-même partielle.

Nous proposons d'implanter les connaissances d'un agent comme la donnée conjointe des éléments suivants :

- le domaine de discours D,
- le langage de description L,
- la description partielle de la fonction de dénotation de L dans D.

Le domaine de discours

Le *domaine de discours* est l'ensemble des objets dont le système peut parler, notamment les actions qu'il peut exécuter, les fonctions qu'il peut calculer et les objets qu'il peut percevoir ou construire. Nous disons que ces actions *représentent* des actions parce qu'elles sont exécutables. De même, ces fonctions *représentent* des fonctions parce qu'elles peuvent être appliquées à des arguments et rendre un résultat. D'une certaine manière les actions et les fonctions *sont* des actions et des fonctions (par opposition à des noms pour des actions ou des fonctions) parce qu'elles sont opératoires. Pour les objets percevables ou constructibles, il est plus délicat de montrer intuitivement qu'ils représentent quelque chose. Nous disons qu'ils *représentent* des objets ou des états du monde dans la mesure où certaines descriptions dénotent certains objets si et seulement si le système et le monde extérieur sont dans un certain état. Cette condition leur assigne un contenu sémantique de la même manière qu'un état d'automate dans la formalisation de Rosenschein possède un contenu sémantique. La sémantique de ces objets ne dépend donc plus seulement d'un sens attribué par le concepteur du système, mais est en relation directe avec des états de l'environnement.

Remarquons que nous pouvons imaginer aisément comment implanter un tel domaine de discours. Les fonctions et les actions sont implantables sous forme de code exécutable. Les autres objets peuvent être implantés par n'importe quelle structure de données pourvu qu'elle satisfasse à au moins une des conditions de "sémanticité" suivantes :

- elles sont constructibles (calculables) par une fonction,
- elles peuvent être assignées à une expression par un acte perceptif,
- elles peuvent être assignées à une expression par un acte déductif.

Ainsi, si nous avons les fonctions arithmétiques courantes, les nombres sont des objets du domaine de discours par la première condition. Si à l'expression "couleur du ciel", un acte perceptif peut assigner le symbole "bleu", "bleu" est un élément du domaine de discours. Mais ces conditions interdisent de dire que "bleu" est un élément du domaine de discours si aucun capteur n'est capable d'identifier la couleur bleue ou si cette couleur n'est pas constructible ou déductible à partir d'autres représentations.

Le langage de description

Le *langage de description* est un langage de termes. C'est-à-dire que les expressions sont de la forme :

$$f(t_1, \ldots, t_n), n \geq 0$$

Un terme sans argument dénote un objet c'est-à-dire une action, une fonction ou un autre objet. Un terme composé dénote le résultat de l'application de la dénotation

du nom de la fonction à la dénotation de chacun de ses arguments. Nous avons donc un langage fonctionnel.

Dans la suite, nous prenons la liberté d'écrire les termes avec un ou deux arguments en forme préfixée ou infixée respectivement lorsque la notation s'en trouve allègée.

La fonction de dénotation

Nous avons vu que dans l'approche des mondes possibles comme dans l'approche syntaxique, il est indispensable d'avoir deux types de constantes :

- les constantes symboliques,
- les désignateurs rigides.

Il est en effet difficile de parler des croyances d'un agent à propos d'un objet si nous ne pouvons pas garantir soit que nous parlons d'un objet précis (constante identifiable) soit que nous parlons bien du même objet (désignateur rigide). Nous allons plus loin et nous posons qu'un agent ne peut pas savoir ou croire quelque chose à propos d'une proposition ou d'un terme s'il ne sait pas ce que cette proposition ou ce terme signifie. C'est pourquoi nous allons définir la sémantique de notre formalisation des croyances d'un agent à partir de la description explicite de la fonction de dénotation.

2.3.2 Situation

Nous appelons *situation*, la structure de modèle qui formalise la représentation des connaissances que nous voulons mettre en oeuvre.

Une *situation* est un triplet $S = (L, D, \mu)$ où :

L : est un langage de description.

D : est le domaine de discours. Il est formé des fonctions de E^n dans E où E est un ensemble d'objets et $n \geq 0$. Les fonctions de E^0 dans E s'identifient à E lui-même.

μ : est une fonction de dénotation de L dans D avec les propriétés suivantes :

partialité : μ n'est pas définie pour toutes les phrases du langage L.

extentionalité : si μ est défini pour les formules $f(t_0, \ldots, t_n)$, f et t_0 à t_n alors la contrainte suivante doit être satisfaite :

$$\mu(f(t_0, \ldots, t_n)) = \mu(f) \circ (\mu(t_0), \ldots, \mu(t_n))$$

où $\circ$ est l'opérateur d'application.

Etant donné que les règles d'inférence sont représentées comme des actions, elles ne font pas partie de notre modèle.

En logique classique, les formules bien formées dénotent exclusivement les valeurs de vérité. Il est possible de faire de L un langage logique en introduisant dans D les valeurs de vérité et les fonctions booléennes. Il faut également introduire les contraintes suivantes sur μ :

$$\begin{aligned} \mu(\wedge) &= \textit{fonction et logique} \\ \mu(\vee) &= \textit{fonction ou logique} \\ \mu(\neg) &= \textit{fonction négation logique} \\ &\vdots \end{aligned}$$

La contrainte d'extentionalité garantit que toute valuation est alors compatible avec la sémantique des connectifs logiques. Nous appelons une telle situation une *situation logique*. Dans la suite, nous utilisons uniquement des situations logiques.

Pour montrer que notre modèle inclut la structure de modèle des mondes possibles, nous introduisons la notion d'*extension* d'une situation :

Définition 2 *Une* extension *d'une situation $S = (L, D, \mu)$, notée $ext(S)$, est une situation $S' = (L, D, \mu')$ avec les propriétés suivantes :*

- *μ' est complète : $\forall l \epsilon L, \mu'(l)$ est défini.*
- *μ' coïncide avec μ : $\forall l, défini(\mu(l)) \rightarrow \mu'(l) = \mu(l)$.*

S peut avoir zéro, une ou plusieurs extensions :

- S n'a pas d'extension si et seulement si μ est *implicitement* contradictoire. Par exemple, si nous avons simultanément :

$$\begin{aligned} \mu(p) &= \textit{true} \\ \mu(\rightarrow) &= \textit{l'implication logique} \\ \mu(p \rightarrow q) &= \textit{true} \\ \mu(\neg q) &= \textit{true} \end{aligned}$$

  aucune extension n'est possible parce que toute extension assigne à la fois *vrai* et $faux$ à q. Notons que μ ne peut pas être *explicitement* contradictoire parce qu'elle doit être une fonction.

- S possède une seule extension si et seulement si $ext(S) = S$, donc quand elle est le point fixe de l'opérateur d'extension. Dans ce cas, la fonction μ est complète. S correspond alors exactement à une interprétation au sens de la logique classique.

- S possède normalement plus d'une extension. Une situation logique avec les bonnes contraintes peut être vue comme une représentation de l'ensemble des mondes possibles compatibles avec la connaissance d'un agent. Nous notons $\{ext(S)\}$ l'ensemble des extensions de S.

Une situation est considérée comme une représentation de l'ensemble des mondes possibles que possède un agent grâce à la partialité de sa fonction de dénotation. Cependant, elle prend aussi en compte les incomplétudes logiques et déductives puisqu'il n'y a pas construction explicite de ce que l'agent peut déduire de ses connaissances.

Notre approche inclut aussi l'approche syntaxique. En effet, avoir une proposition p dans une base de connaissances représente implicitement que le fait p est vrai pour l'agent. Nous pouvons l'exprimer explicitement dans notre modèle en posant que :

$$\mu(p) = true$$

dans la situation. De la même façon, la déduction de la proposition q dans le modèle syntaxique est représentée par une action qui transforme une situation S_0 dans laquelle $\mu(q)$ n'est pas définie dans la situation S_1 où $\mu(q) = true$. Il se trouve que $\{ext(S_1)\} \subseteq \{ext(S_0)\}$, ce qui correspond à l'intuition que l'ajout de connaissances réduit le nombre d'alternatives pour décrire le monde extérieur (voir [21] pour une discussion sur ce point). Notre modèle inclut donc le modèle formel sur lequel repose l'approche syntaxique.

2.3.3 Méta-situation

Après avoir modélisé les croyances d'un agent, nous allons définir quand un agent sait et quand il ne sait pas. Pour cela, nous utilisons un langage dont la sémantique est basée sur la notion de situation. Etant donné que nous pouvons de nouveau en faire un langage de description, nous décrivons notre formalisation de la connaissance sur la connaissance sous la forme d'une méta-situation.

Une *méta-situation* est un quadruplet $MS = (ML, MD, S, M\mu)$ où :

S : est une situation dont le langage est appelé L et la fonction de dénotation, μ.

ML : est un langage de termes dont les expressions de base sont de la forme :

$$f(t_1, \ldots, t_n), n \geq 0$$

appelées *termes simples.* Nous y ajoutons les expressions de la forme :

$$[B]exp$$

et

$$[Believe]exp$$

où exp est une phrase du langage L de la situation S. Ces dernières expressions sont appelées des *termes épistémiques.* $[B]exp$ exprime que l'agent possède une croyance à propos de exp et $[Believe]exp$ exprime que l'agent croit que exp. Nous introduisons également les expressions :

$$'exp$$

pour désigner les phrases du langage L.

MD : est le domaine de discours qui peut être distinct de D mais doit contenir les valeurs de vérité.

$M\mu$: est une fonction de dénotation de ML dans MD :

- partielle
- extentionnelle pour les termes simples
- S-compatible, à savoir:
  - si $M\mu([B]exp)$ est définie alors $M\mu([B]exp) = vrai$ si et seulement si $\mu(exp)$ est définie.
  - si $M\mu([Believe]exp)$ est définie alors $M\mu([Believe]exp) = vrai$ si et seulement si $\mu(exp) = vrai$.
  - $M\mu('exp) = exp$ où exp est une phrase de L.

Nous avons choisi comme langage ML le langage L, étendu avec quelques opérateurs épistémiques. Cependant, les deux langages ne doivent pas être confondus.

Regardons la correspondance que nous pouvons faire entre cette formalisation et le modèle abstrait que nous avons présenté dans le paragraphe 2.1 :

- soit la phrase :

  $$[B]bleu(ciel)$$

  Elle est vraie si et seulement si $\mu(bleu(ciel)) = vrai$ ou $\mu(bleu(ciel)) = faux$. Autrement dit, elle est vraie si et seulement si l'agent *sait si* le ciel est bleu.

- soit la phrase :

  $$[B]no\text{-}téléphone(Paul)$$

  Elle est vraie si et seulement si $\mu(no\text{-}telephone(Paul))$ est défini donc si l'agent *sait quel* est le numéro de téléphone de Paul.

- soit la phrase :

  $$[Believe]bleu(ciel)$$

  Cette phrase est vraie si $\mu(bleu(ciel)) = vrai$ c'est-à-dire si l'agent *sait que* le ciel est bleu.

Nous sommes donc capable d'exprimer qu'un agent *sait-si*, *sait-quoi* et *sait que*. La dernière expression correspond à ce qui est exprimable dans les autres approches. Les deux premières expressions sont très utiles lorsqu'un agent doit planifier ses activités perceptives et déductives. D'habitude, un agent planifie un acte perceptif pour *savoir si* une proposition est vraie ou fausse et non pour savoir spécifiquement qu'elle est vraie. La même chose se produit pour la déduction motivée par le désir d'acquérir de la connaissance à propos de quelque chose indépendamment de ce que nous allons obtenir.

Nous modélisons le "savoir que" en disant que savoir que p c'est savoir que la phrase p dénote vrai. L'introduction des valeurs booléennes dans D nous permet

de modéliser le "savoir si" par l'opérateur $[B]$ quand il porte sur un prédicat ou une formule logique plus générale. En effet, $[B]p \wedge q$ si et seulement si l'agent *sait si* $p \wedge q$ est vrai.

Pour pouvoir exprimer qu'un agent sait que le numéro de téléphone de Jean est, par exemple, 67.45.12, nous avons besoin d'introduire l'*équivalence.*

Pour cela, nous introduisons la contrainte suivante sur μ :

$$\mu(=) = \textit{l'identité}$$

Cette sémantique fait du symbole $=$ une relation d'équivalence sur L. En effet, nous avons que: $\mu(a = b)$ est vrai si et seulement si $\mu(a)$ est identique à $\mu(b)$. Nous avons respectivement que:

- $\mu(exp) = \mu(exp)$ pour toutes les expressions exp,
- $\mu(exp_1) = \mu(exp_2)$ implique que $\mu(exp_2) = \mu(exp_1)$,
- et $\mu(exp_1) = \mu(exp_2)$ et $\mu(exp_2) = \mu(exp_3)$ implique que $\mu(exp_1) = \mu(exp_3)$.

La relation $=$ devient donc une relation d'équivalence.

Introduisons un ensemble de constantes identifiables dont la dénotation est connue par une description adéquate de μ. Si nous admettons que 67.45.12 est une constante identifiable, nous pouvons exprimer que l'agent sait que le numéro de téléphone de Jean est 67.45.12 :

$$[Believe](\textit{téléphone(Jean)} = 67.45.12)$$

qui est vrai si $\mu(telephone(Jean))$ est identique à $\mu(67.45.12)$. Nous pouvons également dire que l'agent sait si le numéro de téléphone de Jean est le 67.45.12 :

$$[B](\textit{téléphone(Jean)} = 67.45.12)$$

qui est vraie si $\mu(\textit{téléphone(Jean)})$ et $\mu(67.45.12)$ sont tous les deux définis et sont soit égaux, soit différents.

2.3.4 Auto-situation

Le choix d'un langage similaire à la fois comme langage de l'agent et comme méta-langage sur les croyances de l'agent n'est pas innocent. En effet, nous voulons que l'agent puisse exprimer la connaissance qu'il possède de ses propres croyances. L'étape suivante consiste donc à faire du méta-langage, le langage de l'agent.

Pour cela, nous introduisons la notion d'*auto-situation* qui n'est rien d'autre qu'une méta-situation de la forme :

$$AS = (L, D, AS, \mu)$$

c'est-à-dire une méta-situation qui s'auto-décrit.

La fonction μ pour les termes épistémiques est définie par l'ensemble des contraintes suivantes :

$$\begin{array}{lcl} \mu([B]exp) = vrai & \leftrightarrow & \mu(exp) \textit{ est défini} \\ \mu([B]exp) = faux & \leftrightarrow & \mu(exp) \textit{ n'est pas défini} \\ \mu([Believe]exp) = vrai & \leftrightarrow & \mu(exp) = vrai \\ \mu([Believe]exp) = faux & \leftrightarrow & \mu(exp) \textit{ n'est pas défini ou } \mu(exp) \neq vrai \end{array}$$

Etant donné qu'une situation ne décrit pas les conclusions que l'agent peut potentiellement tirer des faits qu'il possède, ce sont bien des contraintes sur la valeur de μ si elle est définie pour le terme épistémique. Elles n'imposent pas du tout à μ d'être défini pour ces formules.

Une conséquence est la possibilité d'attribuer un sens à des formules contenant des opérateurs épistémiques emboîtés. Ces contraintes impliquent de plus les axiomes d'introspection positive :

$$[Believe]exp \rightarrow [Believe][Believe]exp$$

et négative :

$$\neg[Believe]exp \rightarrow [Believe]\neg[Believe]exp$$

Nous avons aussi les analogues pour l'opérateur $[B]$:

$$[B]exp \rightarrow [Believe][B]exp$$

$$\neg[B]exp \rightarrow [Believe]\neg[B]exp$$

Encore une fois, cela ne veut pas dire que si, par exemple, $\mu([B]exp) = vrai$ alors $\mu([Believe][B]exp)$ est défini et vaut $vrai$. Cela veut seulement dire que l'agent peut validement tirer cette conclusion s'il en a besoin. Le fait d'en avoir besoin trouvera une définition précise dans le chapitre 4.

2.3.5 Comparaison

Il est ici intéressant de comparer cette modélisation d'une part à la formalisation de l'introspection chez Konolige, d'autre part à la logique autoépistémique de Moore[31].

Nous avons vu en effet que pour Konolige comme dans notre méta-langage, nous ne pouvons pas avoir d'opérateurs épistémiques emboîtés les uns dans les autres. En fait, Konolige lève cette restriction en permettant à un agent d'avoir une description de lui-même donc de se modéliser lui-même sous la forme d'un sous-système de croyances. Cette approche est intéressante car elle permet à un agent d'avoir un modèle de lui-même qui soit faux ou incomplet et donc aussi de se modéliser ayant un modèle erroné ou partiel de lui-même. Nous introduisons également cette caractéristique. Cependant, les contraintes sur μ font que l'idée que se fait un agent de lui-même ne peut pas être fausse. Elle peut par contre être incomplète de par l'incomplétude de μ.

La logique autoépistémique de Moore se place dans un contexte un peu différent puisqu'il propose une logique qui prend en compte la non-monotonie inhérente aux systèmes qui travaillent sur des hypothèses. Alors que McDermott introduit la non-monotonie par un opérateur de consistance [27,29], Moore décrit une formalisation à l'aide d'opérateurs épistémiques tels que :

$$[Know]p$$

est un théorème si et seulement si p est un théorème.

$$\neg[Know]p$$

est un théorème si et seulement si p n'en est pas un.

Ces contraintes sont très similaires à celles que nous utilisons pour contraindre μ. La différence réside dans la clôture déductive que nous n'introduisons pas pour tenir compte de l'incomplétude déductive. Une conséquence importante se déduit de cette comparaison : les contraintes imposées sur notre fonction de dénotation introduisent la non-monotonie dans notre système. En effet, tout ajout d'une information exp change la dénotation de l'expression $[K]exp$ correspondante si elle existe. Cependant, dans Moore, $[Know]p$ est déductible si p est dans l'ensemble de tous les théorèmes déductibles ce qui suppose que nous les avons déjà tous déduits. Nous avons donc une circularité qui rend la théorie non récursivement énumérable. Dans notre cas, $[K]exp$ ne dépend que des informations explicitement représentées dans une situation et non pas de tout ce qui en est dérivable.

2.4 Résumé

Dans ce chapitre, nous nous sommes intéressés à la façon de représenter et de décrire les croyances d'un agent. Les croyances sont codées par des représentations. Ces représentations doivent avoir un certain nombre de caractéristiques. Un langage est basé sur ces représentations et constitue donc un langage sur les croyances.

Les caractéristiques de nos représentations sont les suivantes :

- les représentations représentent.
- les représentations reflètent les incomplétudes suivantes :
  - la partialité,
  - l'incomplétude logique,
  - l'incomplétude déductive.
- les représentations permettent d'exprimer facilement qu'un agent sait ce que dénote une expression quelconque.

Les croyances d'un agent sont représentées par une *situation* $S = (D, L, \mu)$:

D : le domaine de discours est un univers fonctionnel,

L : le langage est un langage de termes,

μ : une fonction partielle de L dans D.

A partir de la notion de situation, nous pouvons attribuer une sémantique aux expressions :

- savoir que,
- savoir quoi,
- savoir si.

Pour cela nous introduisons une situation décrivant une autre situation, formalisée par une *méta-situation* $MS = (ML, MD, S, M\mu)$:

S : une situation (L, D, m).

ML : un langage de termes avec les opérateurs $[B]$ (savoir-si, savoir-quoi) et $[Believe]$ (savoir-que).

MD : un domaine de discours contenant les phrases de L.

$M\mu$: une fonction partielle de ML dans MD telle que

- $M\mu([B]p) = vrai$ si et seulement si $\mu(p)$ est défini
- $M\mu([Believe]p) = vrai$ si et seulement si $\mu(p) = vrai$

Si $[B]p$ est vrai alors l'agent *sait si* p est vrai ou non lorsque p dénote une valeur de vérité et *sait quel* est p si p dénote un objet autre qu'une valeur de vérité. Si $[Believe]p$ est vrai alors l'agent *sait que* p est vrai.

Finalement, un agent exprimant les connaissances qu'il possède sur ses propres croyances est modélisé par une méta-situation se décrivant elle-même, formalisée par une *auto-situation* $AS = (AL, AD, AS, \mu)$.

Chapitre 3

Modélisation des actions

3.1 Introduction

Supposons que notre agent veuille prendre le train. Nous pouvons décrire l'action de prendre le train par l'expression :

prendre(train,heure,quai)

Pour en être arrivé là, l'agent doit déjà savoir que prendre le train lui permet de satisfaire un de ses désirs, par exemple, de se rendre quelque part. Ceci implique qu'il puisse raisonner sur les *effets* des actions. Le deuxième problème qui se pose à notre agent est de rendre cette action physiquement exécutable. Pour cela, il doit, par exemple, être sur le quai, avoir un ticket valable, etc. Il doit donc raisonner sur l'*exécutabilité* de l'action. Mais ces conditions ne sont pas encore suffisantes pour qu'il soit capable de prendre le train. Il faut encore qu'il sache comment faire pour prendre un train, qu'il sache quel train prendre et qu'il sache à quelle heure et sur quel quai le train part. Il doit donc *connaître* l'action qu'il va exécuter. Ce n'est que si il connaît l'action et si l'action est exécutable qu'il est effectivement *capable* de l'exécuter.

La description des effets d'une action repose sur la représentation des actions sous forme de transitions. Nous verrons que la *logique dynamique* [37,32] dont la sémantique est basée sur les mondes possibles, convient bien à ce type de représentation. Cependant, étant donné qu'à chaque monde est associé notamment l'ensemble des objets dont nous pouvons parler (son domaine de discours) et que les actions sont des transformations d'un monde dans un autre, les actions sont donc extérieures au domaine de discours. Nous disons que ce sont des *actions-transition.*

Le raisonnement sur les actions elles-mêmes ne se situe pas sur le même niveau que le raisonnement sur les effets d'une action. En effet, dans ce dernier cas nous devons être capable de construire une action, de savoir si nous la connaissons et de raisonner sur son exécutabilité. Les actions doivent donc faire partie du domaine de discours. Nous appelons ces actions des *actions-objet.* Cela veut dire aussi que

nous avons des termes dans notre langage qui dénotent des actions-objet. Nous allons donc travailler sur trois niveaux :

- les termes dénotant des actions qui sont des objets syntaxiques,
- les actions-objet qui sont des objets du domaine de discours,
- les actions-transition qui modélisent la dynamique des actions.

Les descriptions peuvent dénoter des actions-objet et à chaque action-objet il doit correspondre une action-transition qui en décrit les effets. Nous avons déjà formalisé dans le chapitre précédent la fonction qui fait passer des termes aux objets du domaine de discours et défini la sémantique de la connaissance dans ce cadre. D'autre part, nous pouvons considérer qu'il existe une bijection qui associe une transition à chaque action-objet. Cela signifie qu'il existe pour chaque action du domaine de discours, une description de cette action en termes d'effets et d'exécutabilité.

Les problèmes proviennent du fait que d'une part les descriptions des actions sous forme de transitions dépendent fortement de la connaissance que l'agent possède de ces actions. D'autre part, les actions peuvent modifier les connaissances qu'un agent possède des actions. A notre connaissance, il n'existe pas de formalisation de ces interdépendances.

Dans une première partie, nous présentons les travaux existants dans ce domaine. Un premier paragraphe traite de la logique dynamique. Puis nous présentons la formalisation que Moore propose dans sa thèse [30]. Cette formalisation permet d'exprimer les liens entre les connaissances et les actions et notamment de formaliser la notion d'être capable de faire une action que nous avons présentée intuitivement. Elle nous permet également d'expliquer les différentes difficultés qu'il nous faut considérer.

La formalisation de Moore suffit à notre propos, mais elle se base sur la sémantique des mondes possibles pour la logique épistémique. Dans la seconde partie, nous esquissons une adaptation de cette formalisation à notre modélisation de la connaissance et montrons intuitivement qu'elle permet d'exprimer ce dont nous avons besoin dans le cadre de cette thèse. Nous n'avons pas la prétention de pousser cette formalisation jusqu'au bout et encore moins de résoudre la problématique des interactions entre les connaissances et les actions.

Nous concluons par une comparaison de notre approche avec la génération hiérarchique de plans d'actions.

3.2 Les autres approches

Une action-transition modifie à la fois le monde extérieur et les connaissances que le système possède du monde extérieur. Nous introduisons d'abord la logique dynamique qui nous permet de décrire les effets sur le monde, et l'exécutabilité d'une action. Nous présentons ensuite l'approche de Moore [30] pour intégrer le raisonnement sur les connaissances et le raisonnement sur les actions.

3.2.1 Logique dynamique

La sémantique de la logique dynamique repose sur les mondes possibles. Intuitivement, un monde possible représente un état de l'univers. Notons que cette modélisation des événements est plus naturelle que dans le cas des opérateurs épistémiques. En effet, les croyances d'un agent sont représentées par un ensemble de mondes possibles. Cependant, dans ce cas, il est difficile de faire correspondre une entité concrète à la notion de monde possible. Par contre, en logique dynamique, nous pouvons considérer qu'un monde possible est la représentation d'un état de l'univers à un instant donné. Nous nous limitons dans cette présentation à la logique dynamique propositionnelle.

Le langage

Le *langage* de la logique dynamique propositionnelle est le calcul des propositions auquel nous ajoutons les opérateurs modaux :

$$[a_1], \ldots, [a_n]$$

pour chacune des actions a_1 à a_n. L'expression :

$$[a_i]p$$

se lit : après l'action a_i, p est vrai.

La sémantique

La *structure de modèle* est un n-uplet $M = (S, \phi, R_{a_1}, \ldots, R_{a_n})$ où :

S : est un ensemble d'états

ϕ : est une valuation des propositions pour chacun des états

$R_{a_1}, \ldots, R_{a_n}$: est un ensemble de relations sur les mondes possibles associées aux actions a_1 à a_n.

Notons qu'une relation peut décrire aussi bien une transition d'un état vers un autre que la transition d'un état vers plusieurs autres. Dans ce dernier cas, cette relation décrit un événement *non-déterministe* alors que dans le premier, l'événement est strictement *déterministe*. Par exemple, le jet d'un dé est une action non-déterministe puisqu'elle peut aboutir dans différents états du monde selon la face qui apparaît.

Une *interprétation* est définie par le choix d'un M et d'un état s dans S. La sémantique des formules modales dans une interprétation est telle que :

$$(M, s) \models [a_i]p$$

si et seulement si

$$(M, t) \models p$$

pour tout t tel que (s,t) appartient à R_{a_i}. Nous voyons que la définition est similaire à celle que nous avons donnée pour la sémantique des opérateurs épistémiques dans la sémantique des mondes possibles.

Dans cette définition de l'opérateur, il peut arriver qu'il n'y ait aucune transition à partir d'un état donné s. La définition dit seulement que si il y a une transition pour une action a d'un état s vers un ensemble d'états $t_1, \ldots, t_n$ et qu'une formule p est vraie dans chacun de ces états, alors la formule $[a]p$ est vraie. Nous appelons cette définition de l'opérateur intensionnel $[a]$, la définition *faible*. L'opérateur ne décrit alors que les effets de l'action dans le sens que la formule:

$$p \rightarrow [a]q$$

affirme que p est une condition pour avoir q vrai après a mais p n'est pas une condition suffisante pour que a s'exécute.

Pour guarantir que l'action a s'exécute effectivement, il faut ajouter qu'il existe au moins une transition. Nous avons alors que:

$$(M,s) \models [a_i]p$$

si et seulement si il existe au moins un monde possible t tel que (s,t) appartienne à R_{a_i} et:

$$(M,t) \models p$$

pour tout t tel que (s,t) appartient à R_{a_i}. Nous obtenons alors ce que nous appelons la définition *forte*. Dans ce cas, la formule:

$$p \rightarrow [a]q$$

exprime que p est à la fois une condition pour que l'action a soit exécutable et qu'elle ait l'effet q.

Nous pouvons en déduire une définition de l'exécutabilité d'une action. Une action a est *exécutable* dans un monde possible s, ce qui se note:

$$(M,s) \models executable(a)$$

si et seulement si il existe au moins un monde possible t tel que (s,t) appartienne à R_a. Cette définition s'exprime dans la logique dynamique de la façon suivante:

$$executable(a) \leftrightarrow <a> vrai$$

où:

- $(M,s) \models vrai$ dans tous les mondes possibles,
- $\neg[a]\neg p \leftrightarrow <a> p$.

Finalement, une formule est *valide* si elle est vraie pour tout choix de s, c'est-à-dire dans tous les états du monde.

L'axiomatisation

L'*axiomatisation* de la logique dynamique s'obtient en ajoutant aux axiomes du calcul propositionnel l'axiome suivant :

$$[a](p \rightarrow q) \rightarrow ([a]p \rightarrow [a]q)$$

et pour chaque action a_i, la règle d'inférence :

$$si \vdash p \ alors \vdash [a_i]p$$

Comme pour la logique épistémique, les règles d'inférence sont une conséquence directe de la notion de validité. L'axiome et les règles d'inférence garantissent que :

- tout événement produit toutes les conséquences de ses effets,
- ce qui est universellement vrai reste toujours vrai.

La logique dynamique étend les possibilités de description des actions en permettant d'écrire les événements sous la forme :

- de séquence : $a_1; a_2$,
- de choix non-déterministe : $a_1 \cup a_2$,
- de test : $p?$,
- d'itération : $a^\star$.

L'axiomatique suivante permet de rendre compte de la sémantique que nous voulons donner à ces expressions :

test : $[p?]q \Leftrightarrow p \rightarrow q$

composition : $[a; b]p \Leftrightarrow [a][b]p$

union : $[a \cup b]p \Leftrightarrow [a]p \wedge [b]p$

itération :

$[a^\star]p \rightarrow p$ (réflexivité)
$[a^\star]p \rightarrow [a][a^\star]p$ (pas d'itération)
$p \wedge [a^\star](p \wedge [a]p) \rightarrow [a^\star]p$ (induction)

Remarquons que si nous donnons une description de ces axiomes en termes de relations sur les mondes possibles, nous voyons que la définition du test est telle que $s \models [p?]q$ est vrai si et seulement si $sR_{p?}s$. Par contre, la relation est vide si p est faux dans s. Donc un test faux n'envoie dans aucun monde. Notamment si p est faux quand nous cherchons à faire $p?; a$, l'événement a ne se produira jamais. La notion d'événement conditionnel s'exprime donc aisément.

Conclusion

La logique dynamique nous a permis d'exprimer deux notions importantes dont nous avons besoin:

- les effets des actions a_i qui sont décrits à l'aide des opérateurs modaux $[a_i]$ munis de leur définition faible,
- l'exécutabilité d'une action.

Dans cette sémantique de la logique dynamique, les actions ne font pas partie du domaine de discours. Elles ne sont décrites que par les opérateurs modaux et donc les relations sur les états. Il n'est donc pas possible de parler de la connaissance qu'un agent peut posséder d'une action. Un tel discours se situe à un méta-niveau. C'est ce que nous allons aborder dans le prochain paragraphe.

3.2.2 L'approche de Moore

Nous avons vu dans l'exemple introduisant ce chapitre qu'un agent doit savoir un certain nombre de choses pour agir. Dans le cas de prendre un train, il doit savoir, entre autre, à quelle heure et de quel quai part son train. Réciproquement, certaines actions peuvent apporter de la connaissance, par exemple, les actions perceptives. Malheureusement, ces deux types de raisonnement peuvent difficilement se traiter au même niveau puisque le raisonnement sur la connaissance d'une action suppose que les actions font partie du domaine de discours (voir chapitre 2), alors que la logique dynamique considère les actions comme des transitions. Pour résoudre ce problème, Moore propose une formalisation en calcul des prédicats du premier ordre de la sémantique des mondes possibles pour respectivement la logique épistémique et la logique dynamique. Il est alors possible d'exprimer les interactions entre les deux logiques dans ce méta-langage.

Comme dans l'approche syntaxique, nous avons deux langages:

- un langage objet qui est le calcul des prédicats du premier ordre muni des opérateurs intensionnels des logiques épistémique et dynamique.
- un méta-langage qui est le calcul des prédicats du premier ordre sans opérateur intensionnel et qui nous permet de décrire le langage objet.

Pour distinguer entre ces deux langages, Moore a choisi des écritures différentes pour les opérateurs logiques et les quantificateurs des deux langages. Pour être consistant avec notre notation, nous utilisons les opérateurs:

$$\vee, \wedge, \neg, \rightarrow, \leftrightarrow, \forall, \exists$$

pour le langage objet et les opérateurs:

$$or, and, \sim, \Rightarrow, \Leftrightarrow, all, exists$$

pour le méta-langage. D'autre part, les variables du méta-langage sont typées, c'est-à-dire qu'elles ne portent que sur des sous-ensembles du domaine de discours.

Nous présentons successivement la formalisation dans le langage objet, du calcul des prédicats du premier ordre, puis des logiques épistémique et dynamique. Finalement nous introduisons la formalisation des interactions entre les actions et la connaissance.

La sémantique

La sémantique du langage objet est définie dans le méta-langage par le prédicat:

$$T(w, p)$$

où w est un monde possible et p est une formule du langage objet.

Les connectifs logiques du langage objet trouvent une définition immédiate en termes des connectifs du méta-langage, de la façon suivante:

$$all\ w\ p, T(w, \neg p) \Leftrightarrow (\sim T(w, p))$$

$$all\ w\ p1\ p2, T(w, p1 \vee p2) \Leftrightarrow (T(w, p1)\ or\ T(w, p2))$$

$$all\ w\ p1\ p2, T(w, p1 \wedge p2) \Leftrightarrow (T(w, p1)\ and\ T(w, p2))$$

$$all\ w\ p1\ p2, T(w, p1 \rightarrow p2) \Leftrightarrow (T(w, p1) \Rightarrow T(w, p2))$$

$$all\ w\ p1\ p2, T(w, p1 \rightarrow p2) \Leftrightarrow (T(w, p1) \Rightarrow T(w, p2))$$

Les variables p, $p1$ et $p2$ portent uniquement sur les phrases du langage objet.

Les quantificateurs sont formalisés de la façon suivante:

$$all\ w\ p\ v, T(w, \forall v, p) \Leftrightarrow (all\ s, T(w, P[\phi(s)/v]))$$

$$all\ w\ p\ v, T(w, \exists v, p) \Leftrightarrow (exists\ s, T(w, P[\phi(s)/v]))$$

où v porte sur les noms de variables. Notons que s porte sur le domaine de discours du langage objet ce qui est conforme à la sémantique que nous voulons donner aux quantificateurs. Cependant, les objets du domaine de discours ne faisant pas partie du langage objet, nous avons besoin d'une fonction ϕ qui transforme un objet du domaine de discours en un terme dont la dénotation dans le monde possible est l'objet lui-même. Moore est même plus exigeant en faisant de ϕ une fonction indépendante des mondes possibles. Elle fournit donc un terme dont la dénotation est la même dans tous les mondes possibles, c'est-à-dire un désignateur rigide (voir paragraphe 2.2.1).

A partir du moment où nous incluons les mondes possibles dans le domaine de discours du méta-langage, nous pouvons aisément exprimer la sémantique de l'opérateur épistémique en introduisant une relation:

$$K(a, w_1, w_2)$$

qui est vraie si et seulement si w_2 est une description compatible avec les connaissances de l'agent a dans le monde w_1. Etant donné que nous ne nous intéressons qu'à un seul agent, nous omettons le premier argument. Nous obtenons alors les définitions suivantes:

$$all\ w\ p, T(w, [K]p) \Leftrightarrow (all\ w', K(w, w') \Rightarrow T(w', p))$$

$$all\ w\ p, T(w, <K>\ p) \Leftrightarrow (exists\ w', K(w, w')\ and\ T(w', p))$$

Nous voyons l'utilité de la définition forte de Moore pour la quantification lorsqu'un quantificateur porte sur une variable qui se trouve dans un contexte intensionnel. En effet, nous obtenons, par exemple, que:

$$all\ w\ p\ v, T(w, \exists v, [K]p) \Leftrightarrow (exists\ s, all\ w', K(w, w')\ and\ T(w', P[\phi(s)/v]))$$

Le fait que $\phi(s)$ soit le désignateur rigide qui dénote s guarantit que nous parlons du même s dans tous les mondes w' et par la même occasion le s dont nous affirmons l'existence dans w.

Pour la logique dynamique, nous introduisons pour chaque action a_i la relation:

$$R(a_i, w_1, w_2)$$

qui est vraie si et seulement si l'action a_i fait passer de l'état du monde décrit par le monde possible w_1 à l'état décrit par le monde possible w_2. Nous avons alors les formules suivantes pour les opérateurs dynamiques correspondant aux actions:

$$all\ w\ p\ a, T(w, [a]p) \Leftrightarrow (all\ w', R(D(w, a), w, w') \Rightarrow T(w', p))$$

$$all\ w\ p\ a, T(w, <a>\ p) \Leftrightarrow (exists\ w', R(D(w, a), w, w')\ and\ T(w', p))$$

Dans cette formulation, a est une variable qui porte sur l'ensemble des termes dénotant des actions. Or, la relation R ne dépend pas du terme mais de l'action dénotée par le terme. C'est pourquoi Moore introduit la fonction:

$$D(w, a)$$

qui doit nous fournir la dénotation du terme a dans le monde w. Cette fonction est liée à la fonction ϕ de la façon suivante:

$$all\ w\ s, D(w, \phi(s)) = s$$

Nous voyons apparaître dans cette formule, les liens entre les termes décrivant les actions et les actions-objet par l'intermédiaire de la fonction D ainsi que la correspondance entre les actions-objet et leur description en tant que transition grâce au prédicat R.

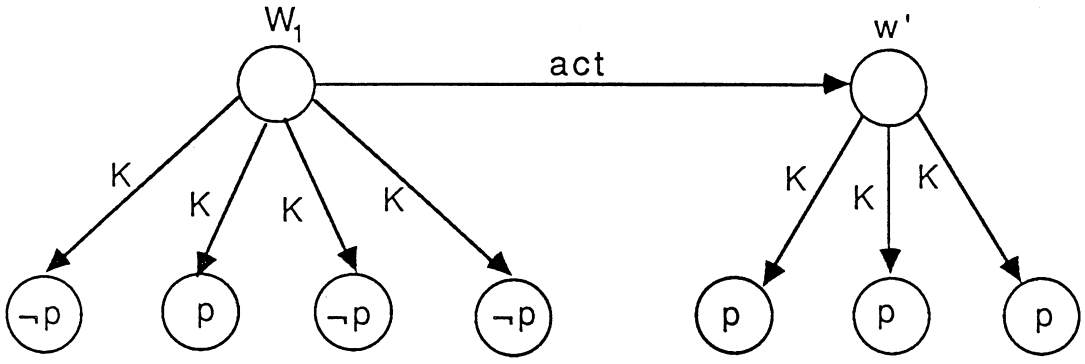

Figure 3.1: Influence d'une action sur la connaissance

Les interactions

Etudions l'influence d'une action sur les connaissances. A l'aide de la formulation du paragraphe précédent, nous pouvons exprimer la sémantique de la formule:

$$[act][K]p$$

dans un monde W_1 de la façon suivante:

$$T(W_1, [act][K]p) \Leftrightarrow$$
$$(all\ w', R(D(W_1, act), W_1, w') \Rightarrow (all\ w'', K(w', w'') \Rightarrow T(w'', p))$$

Cette formule est illustrée par la figure 3.1 dans le cas d'une action déterministe. Si, nous supposons que l'agent ne connaît pas p avant l'action, nous avons des descriptions possibles de W_1 dans lesquelles p est faux et d'autres dans lesquelles p est vrai. Après exécution, p est devenu vrai dans toutes les descriptions que l'agent possède du nouvel état du monde.

Voyons maintenant la sémantique que nous pouvons donner à une phrase telle que:

$$[K][act]p$$

dans un monde W_1. Nous obtenons la formule suivante:

$$T(W_1, [K][act]p) \Leftrightarrow$$
$$(all\ w', K(W_1, w') \Rightarrow (all\ w'', R(D(w', act), w', w'') \Rightarrow T(w'', p))$$

La figure 3.2 permet d'illustrer cette formule. Nous avons deux descriptions possibles du monde par l'agent, w'_1 et w'_2. Nous avons respectivement que:

$$D(w'_1, act) = e_1$$

$$D(w'_2, act) = e_2$$

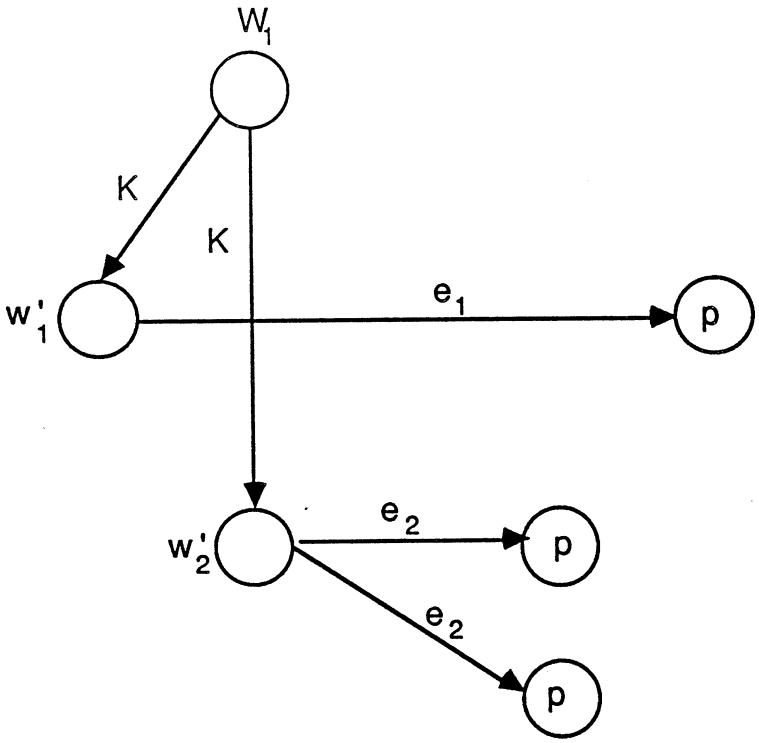

Figure 3.2: Interaction connaissance-action

Chacune des actions dénotées par *act* a pour effet p, nous avons donc:

$$[act]p$$

dans chacun des mondes possibles w'_1 et w'_2. Donc, nous pouvons dire que dans cette figure, w'_1 et w'_2 étant des descriptions possibles de W_1 pour l'agent, l'agent sait que $[act]p$. Nous avons donc que:

$$[K][act]p$$

Cet exemple illustre que cette formule exprime que l'action *act* a l'effet p quelle que soit la dénotation de *act* dans chacune des descriptions que l'agent se fait du monde extérieur. Nous y reviendrons plus tard.

L'exécutabilité

Finalement l'exécutabilité d'une action s'exprime par la formule:

$$\begin{aligned} &all\ w\ act, T(w, executable(act)) \leftrightarrow \\ &exists\ w', R(D(w, act), w, w')\ et\ T(w', true) \end{aligned}$$

en sachant que:

$$all\ w, T(w, true)$$

Nous voyons encore une fois que la phrase $executable(act)$ est vraie dans le langage objet si et seulement si toute action-objet dénotée par le terme *act* est exécutable, c'est-à-dire qu'il lui correspond au moins une transition.

Etre capable

Nous avons dit au début de ce chapitre que nous voulions modéliser la notion d'être capable de faire une action. Nous voyons ici que savoir que $[act]p$ n'est pas suffisant pour que l'agent puisse être capable de faire l'action *act*. En effet, cette description d'action dénote des actions différentes dans des mondes différents. L'agent ne sait donc pas exactement de quelle action il s'agit, ce qui est pourtant indispensable si il veut pouvoir l'exécuter. Moore introduit l'opérateur:

$$[can](act, p)$$

pour exprimer que l'agent est capable d'exécuter l'action *act* avec l'effet p. Il le définit de la façon suivante:

$$all\ w\ act\ p, T(w, [K](\phi(D(w, act)) = act \wedge [act]p)) \Rightarrow T(w, [can](act, p)) \quad (3.1)$$

Si l'action s'écrit sous la forme d'une séquence, c'est-à-dire que le terme est de la forme $a_1; a_2$, alors nous avons:

$$all\ w\ a_1\ a_2, T(w, [can](a_1; a_2, p)) \Leftrightarrow T(w, [can](a_1, [a_1][can](a_2, p)))$$

Cette dernière formule peut s'écrire directement dans le langage objet:

$$[can](a_1; a_2, p) \leftrightarrow [can](a_1, [a_1][can](a_2, p))$$

ainsi que la première:

$$(\exists x, [K](act = x \wedge [act]p)) \rightarrow [can](act, p)$$

La définition de l'opérateur $[can]$ n'est pas complète puisqu'il nous manque l'implication:

$$[can](act, p) \rightarrow (\exists x, [K](act = x \wedge [act]p))$$

Montrons que cette implication est fausse dans le cas général. Tout d'abord, en supposant qu'une séquence $(a_1; a_2)$ est connue, nous avons:

$$\exists x[K](a_1; a_2 = x) \leftrightarrow (\exists x_1[K](a_1 = x_1) \wedge \exists x_2[K](a_2 = x_2))$$

D'autre part, si nous avons:

$$[a_1]\exists x[K](a_2 = x \wedge [a_2]p)$$

nous pouvons en déduire que:

$$[a_1][can](a_2, p)$$

De la même façon, si nous avons:

$$\exists x[K](a_1 = x \wedge [a_1][can](a_2, p))$$

alors:

$$[can](a_1, [can](a_2, p))$$

et donc l'agent est capable de faire la séquence d'actions:

$$[can](a_1; a_2, p)$$

Or, nous n'avons pas forcément:

$$\exists x[K](a_1; a_2 = x)$$

parce que a_2 n'a besoin d'être connu qu'après l'exécution de a_1. Nous n'avons donc pas l'équivalence dans la formule 3.1. Par contre, nous pouvons donner la condition sous laquelle l'équivalence existe:

$$([can](act, p) \leftrightarrow (\exists x, [K](act = x \wedge [act]p))) \leftrightarrow \neg\exists a_1, a_2, act = a_1; a_2$$

Cette dernière condition est tellement importante que nous introduisons une nouvelle notion pour cela. Nous appelons action *primitive* une action telle que:

$$\neg\exists a_1, a_2, act = a_1; a_2$$

Notons qu'un terme a qui n'est pas de la forme $a_1; a_2$, c'est-à-dire sans symbole de séquence, n'est pas forcément une action primitive. Si le terme a n'est pas primitif alors nous avons:

$$exists\ w\ e_1\ e2, D(w, a) = (e_1; e_2)$$

Mais étant donné que la fonction ϕ est supposée définie pour tous les objets du domaine de discours du langage objet, nous avons alors:

$$exists\ w\ e_1\ e_2, T(w, a = (\phi(e_1); \phi(e_2)))$$

Conclusion

La formalisation de Moore permet de modéliser les notions dont nous avons besoin. Nous allons maintenant décrire le passage de la sémantique des mondes possibles à notre formalisation des opérateurs épistémiques. Nous esquissons ensuite un modèle pour donner une sémantique aux formules de notre langage étendu avec les opérateurs de la logique dynamique.

3.3 Notre approche

Nous avons montré dans le paragraphe 2.3.2 qu'une situation représente l'ensemble des mondes possibles au sens de la sémantique de Kripke lorsqu'elle n'est pas implicitement contradictoire. Pour adapter la formalisation de Moore à la nôtre, il suffit intuitivement de remplacer les descriptions possibles d'un état du monde donné par une situation. C'est ce que nous présentons dans un premier paragraphe. Nous explorons ensuite les conséquences sur les effets des actions sur les croyances, l'exécutabilité des actions et la notion d'être capable de faire une action.

3.3.1 Extension de la notion de situation

Nous obtenons alors pour la logique dynamique propositionnelle augmentée de nos opérateurs épistémiques $[B]$ et $[Believe]$, la structure de modèle $M = (W, \Phi, R_{a_1}, \ldots, R_{a_n})$ où:

W: est un ensemble de couples $w = (m, s)$ où m est un monde possible au sens de Kripke et s est une situation, c'est-à-dire un triplet (L, D, μ).

Φ: est une valuation des propositions pour chaque w.

$R_{a_1}, \ldots, R_{a_n}$: est un ensemble de relations sur les couples w associées aux différentes actions.

Une interprétation est comme d'habitude le choix d'un M et d'un couple (m, s) dans W. Nous définissons la sémantique de la façon suivante:

- $(M,(m,s)) \models P$ si et seulement si $\phi(P,(m,s)) = vrai$
- $(M,(m,s)) \models P \wedge Q$ si et seulement si $(M,(m,s)) \models P$ et $(M,(m,s)) \models Q$ et ainsi de suite pour les différents connectifs logiques
- $(M,(m,s)) \models [a_i]P$ si et seulement si $(M,w) \models P$ pour tous les couples w tels que $((m,s),w)$ appartient à R_{a_i}
- $(M,(m,s)) \models [B]P$ si et seulement si $\mu_s(P)$ est défini
- $(M,(m,s)) \models [Believe]P$ si et seulement si $\mu_s(P) = vrai$

Nous avons noter μ_s la fonction μ associée à la situation s.

Il faut remarquer que nous ne pouvons pas écrire une formule de la forme:

$$[B][act]p$$

parce que cela suppose que l'agent est capable de parler des actions, ce que nous n'avons pas encore introduit. Mais, nous pouvons exprimer qu'une action *act* apporte une connaissance à propos d'un terme quelconque t par la formule:

$$[act][B]t$$

En proposant cette formalisation, nous nous sommes mis à la place d'un observateur extérieur ayant une connaissance parfaite et complète de chaque monde possible et de chaque situation associée à chaque monde possible. Si nous voulons modéliser les connaissances partielles d'un agent sur les actions, il nous faut faire les restrictions suivantes:

- l'agent ne connaît pas tous les mondes possibles. W est donc fini.
- l'agent n'a qu'une connaissance partielle de chacun des mondes possibles. Chaque élément de W est une situation et non pas un couple (monde possible,situation).
- l'agent ne se trouve que dans un seul état à la fois. Il faut donc spécifier dans quelle situation il se trouve.

Nous obtenons une nouvelle notion de situation sous la forme d'un n-uplet $S = (L, D, W, \mu, R_{a_1}, \ldots, R_{a_n})$ où:

L: est un langage de termes auquel nous avons ajouté les opérateurs $[B]$,$[Believe]$ et $[a_i]$ pour chaque action a_i

D: est le domaine de discours

W: est un ensemble de situations S' de la forme $(L, D, W, \mu', R_{a_1}, \ldots, R_{a_n})$

μ: est une fonction de L dans D, partielle, extensionnelle sauf pour les opérateurs $[B]$,$[Believe]$ et $[a_i]$, et telle que:

- $\mu([B]exp) = vrai$ si et seulement si $\mu(exp)$ est défini,
- $\mu([Believe]exp) = vrai$ si et seulement si $\mu(exp) = vrai$,
- $\mu([act]exp) = vrai$ si et seulement si $\mu'(exp, s') = vrai$ pour tout s' tel que (S, s') appartient à $R_{\mu(act)}$ (nous avons désigné par μ' la fonction de dénotation associée à s').

La dernière condition sur μ pose un problème si $\mu(act)$ est inconnu. En fait, si $\mu(act)$ n'est pas défini, la formule $\mu([act]p) = vrai$ exprime quelque chose à propos du monde extérieur mais aussi à propos de la connaissance que l'agent peut avoir de act. Dans la sémantique des mondes possibles, nous avons vu que cette formule exprime que p est vrai après toute action que act peut dénoter dans chacune des descriptions possibles que l'agent peut faire du monde dans lequel nous nous plaçons. Par la notion d'extension d'une situation, nous avons une sémantique équivalente dans notre formalisation.

Si $\mu(act)$ est connu, alors l'action correspond à une transition vers un ou plusieurs mondes possibles dont l'agent possède une connaissance partielle. Dans ce cas, si $\mu(p)$ est défini dans chacun des mondes possibles connus de l'agent et vaut vrai, alors la formule $\mu([act]p)$ doit valoir vrai si elle est définie.

Une première différence avec le formalisme de Moore est l'existence de relations différentes dans chaque situation. Il est clair que la description des transitions entre les mondes possibles change lorsque la connaissance sur les actions change. Dans le formalisme de Moore, cette interdépendance doit apparaître par des liens entre la relation K et les relations R. Ces liens n'ont pas été explicités dans le travail de Moore mais il est clair que sa formalisation offre un cadre dans lequel il serait possible de les exprimer. Nous n'avons pas l'intention d'aborder ce problème complexe dans le cadre de cette thèse. Nous nous contentons de montrer intuitivement que nous sommes capables d'exprimer les mêmes notions que Moore.

3.3.2 Les effets des actions sur les croyances

Nous illustrons par la figure 3.3 la différence entre un état du monde extérieur et une situation qui en est une description partielle. Si après l'action a, p est vrai, cela ne veut pas dire qu'après l'action a l'agent sache que p est vrai. Cependant, une action modifie toujours les croyances qu'un agent possède du monde extérieur.

Certaines actions modifient ces croyances explicitement. Ce sont les actes perceptifs et déductifs. Si un agent regarde la couleur du ciel et que le ciel est bleu, l'effet de cette action perceptive sur ses connaissances est, par exemple, que $\mu(couleur(ciel)) = bleu$. De même, si un agent déduit que l'herbe est sèche parce qu'il fait beau, l'effet de cette action déductive est, par exemple, que $\mu(sec(herbe)) = vrai$.

Mais les autres actions doivent également modifier les connaissances que l'agent possède du monde extérieur même si elles ne le font pas explicitement. En effet, si un agent sait que l'herbe est sèche à un certain moment, cette information n'est plus pertinente après avoir arrosé. Donc l'action d'arroser le gazon modifie non seulement le monde extérieur mais doit aussi modifier les connaissances de l'agent.

Les effets explicites

Nous pouvons exprimer l'effet des actions sur les croyances de l'agent, puisque la relation d'accessibilité porte sur les situations qui décrivent les croyances de

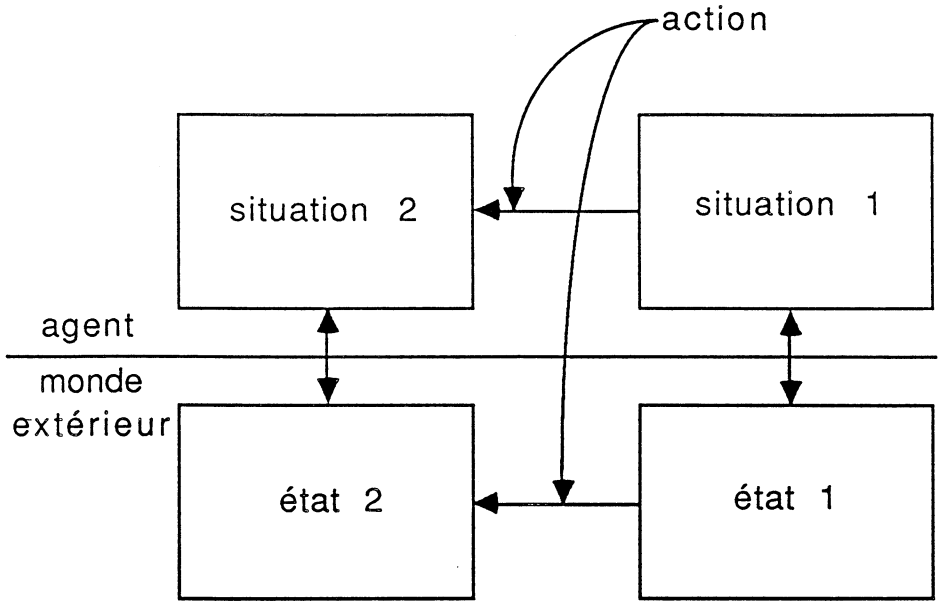

Figure 3.3: Exécution d'une action

l'agent. La formule :

$$[a][B]couleur(ciel)$$

peut être interprétée de la façon suivante : soit S une situation :

$$S \models [a][B]couleur(ciel)$$

si et seulement si $S' \models [B]couleur(ciel)$ pour tous les S' tels que (S, S') appartient à $R_\mu(a)$, c'est-à-dire lorsque $\mu'(couleur(ciel))$ est défini dans chaque situation S'.

Nous pouvons donc décrire les actions permettant d'acquérir de l'information. Par exemple nous pouvons dire :

$$[percevoir('couleur(ciel))][B]couleur(ciel)$$

qui exprime que l'action de percevoir la couleur du ciel nous permet de savoir quelle est sa couleur. Notons que nous avons écrit l'argument précédé d'un apostrophe car l'acte de perception dépend de la phrase dont nous cherchons la dénotation et non pas de la couleur du ciel que nous cherchons justement à établir.

Nous pouvons aussi acquérir des informations par des actions de test. Par exemple, nous pouvons tester si le gaz du four est mis en l'allumant. S'il s'allume, le gaz est mis. L'action peut être décrite physiquement de la façon suivante :

$$mis(gaz) \rightarrow [allume(four)]allumé(four)$$

$$\neg mis(gaz) \rightarrow [allume(four)]\neg allumé(four)$$

En supposant que nous pouvons percevoir directement que le four est allumé, l'acquisition d'information s'en déduit aisément. En effet,

$$[allume(four)](allumé(four) \rightarrow mis(gaz))$$

$$[allume(four)](\neg allumé(four) \rightarrow \neg mis(gaz))$$

A partir de l'état résultant, il est alors possible de déduire quel est l'état du gaz et donc de le savoir par introspection.

Les effets implicites

Il nous reste à traiter le cas des actions qui n'apportent pas explicitement de l'information. Soit la description suivante de l'action :

$$[action]p$$

Nous avons trois possibilités pour tenir compte de l'exécution de l'action *action* :

- dans la nouvelle situation, nous avons que $\mu(p) = vrai$,
- nous indiquons d'une façon ou d'une autre que p peut être devenu vrai,
- nous ne savons pas si p est effectivement devenu vrai et donc p devient inconnu.

Dans tous les cas, si p dénote $faux$ avant l'exécution de l'action, il faut détruire cette dénotation et toutes les dénotations qui en ont été déduites. En effet, le système est non-monotone dans ses déductions pour un certain nombre de raisons et notamment parce qu'il évolue dans le temps. Donc, à partir du moment où nous prenons en compte la dimension temporelle du système, il est normal qu'un mécanisme de rétraction d'informations s'ajoute à la modélisation que nous avons faite. Nous en parlons plus amplement dans l'implantation que nous avons réalisée (voir chapitre 5).

La première possibilité suppose que la connaissance de l'agent est parfaite et donc que les actions s'exécutent toujours parfaitement. Cette modélisation ne laisse aucune place à l'erreur. En fait, si l'agent possède au départ une connaissance suffisamment complète de l'univers, il n'a plus besoin de percevoir quoi que ce soit puisqu'il peut tout prédire parfaitement. Il est clair que cette option n'est pas réaliste puisqu'une action peut échouer ou avoir des résultats imprévisibles.

La seconde possibilité n'est pas exprimable dans notre formalisme. En effet, elle introduit une connaissance incertaine. Nous avons discuté dans l'introduction du chapitre 2, que savoir et croire pouvait être un jugement sur les croyances de l'agent. Ici, il serait intéressant que l'agent puisse croire que p est vrai en sous-entendant que cette croyance peut être abandonnée à la première contradiction. Elle peut également exprimer une attente, auquel cas l'information doit être vérifiée. Mais

l'agent s'attend à percevoir que p est vrai. Nous voyons apparaître ici toute une gamme d'attitude de l'agent vis-à-vis d'une information. Notre modélisation étant incapable d'en tenir compte, nous n'en parlons donc pas davantage.

Puisque les croyances du système peuvent être fausses et qu'il n'est pas possible d'exprimer des jugements sur ces croyances, la dernière possibilité est donc la seule qui s'offre à nous. L'agent ne peut plus croire dans les conséquences des descriptions affectées par l'action. Nous verrons plus loin que cette possibilité fonctionne parfaitement, étant donné les interactions avec les actions et les désirs.

Discussion

Nous voyons que les effets d'une action sur les connaissances d'un agent peuvent être très importantes. Situons cette modélisation par rapport aux problèmes du raisonnement sur les actions. McCarthy a mis au jour trois problèmes fondamentaux du raisonnement sur les actions, à savoir :

le "ramification problem" : qui concerne le raisonnement sur ce qui change lorsque nous exécutons une action et notamment sur les réactions en chaîne.

le "frame problem" : qui concerne le raisonnement sur ce qui ne change pas lorsque nous effectuons une action.

le "qualification problem" : qui concerne l'applicabilité des actions.

Le "ramification problem" est justement ce que nous prenons en compte en décrivant les effets des actions sur les connaissances du système. Pour détruire toutes les informations sur ce qui a été affecté par les actions et toutes les informations qui en sont déduites, nous calculons en fait l'ensemble des descriptions qui portent sur ce qui est modifié par une action.

Le "frame problem" apparaît également. En effet, non seulement le système ne peut pas objectivement savoir si les effets connus de ses actions sont effectivement vérifiés dans le nouvel état, mais en plus, il ne peut pas savoir si les descriptions qui ne sont pas citées restent valides dans le nouvel état. Donc, toute action devrait avoir pour effet d'effacer toute information sur le monde extérieur étant donné qu'a priori tout peut avoir changé. En ne détruisant que l'information affectée par l'action exécutée, nous faisons implicitement l'hypothèse que tout ce qui n'est pas connu comme ayant changé, ne change pas. Ceci s'exprime par la formule:

$$(p \wedge [a]\neg[Believe]\neg p) \rightarrow [a]p$$

Le "qualification problem" vient de l'impossibilité, en général, de fournir une condition nécessaire et suffisante à l'exécutabilité d'une action. Il peut toujours se trouver quelque chose auquel nous n'avons pas pensé et qui peut empêcher l'action. Cette situation peut même être créée volontairement si nous ne voulons pas tester toutes les conditions par manque de temps ou pour toute autre raison. Ainsi nous ne faisons pas une révision technique complète à chaque fois que nous prenons la voiture. Ici, nous en tenons compte implicitement en supposant qu'une action peut échouer.

3.3.3 L'exécutabilité d'une action

Pour exprimer que p est est une précondition de a, nous écrivons la formule :

$$p \leftrightarrow executable(a)$$

p doit être une condition nécessaire et suffisante pour être une condition d'exécutabilité de a. En effet, si nous avons seulement :

$$executable(a) \rightarrow p$$

alors p ne garantit pas l'exécutabilité de a. D'autre part, si nous avions seulement que :

$$p \rightarrow executable(a)$$

alors p ne définit pas exactement l'exécutabilité de l'action a.

Etant donné que a peut être un terme dont la dénotation n'est pas connue a priori, cette expression est très forte. En effet, elle exprime que si p est vrai avant a alors a est exécutable quelle que soit la façon de faire a. Soient $b1$ et $b2$ deux façons de faire a :

$$p1 \rightarrow (a = b1)$$

et

$$p2 \rightarrow (a = b2)$$

ainsi que $e1$ et $e2$ les conditions d'exécutabilité de $b1$ et $b2$ respectivement, alors nous avons que :

$$p \leftrightarrow (p1 \wedge e1) \vee (p2 \wedge e2)$$

Donc, l'expression d'une précondition pour une description d'action quelconque est en général redondante parce qu'elle doit être exhaustive. Les seules actions pour lesquelles la condition d'exécutabilité n'est pas déductible d'autres informations, sont les actions que nous appelons *actions primitives.*

Nous pouvons généraliser la notion d'action primitive aux actions dont nous avons fixé une fois pour toutes la manière de les faire. C'est en général le cas en logique dynamique qui est beaucoup utilisée pour étudier la sémantique des langages de programmation. Les primitives correspondent alors aux instructions du langage et aux routines de support qui sont composées de plusieurs instructions machine. Elles possèdent donc des conditions d'utilisation strictes.

Caractérisons une action primitive dans notre modélisation. Soit une action de la forme $b_1; b_2$, elle ne peut pas être primitive puisque c'est une séquence de deux actions. Une action primitive doit donc apparaître sous la forme d'un terme $act(t_1, \ldots, t_n)$ où t_1 à t_n sont les paramètres de l'action. Cette formulation ne garantit cependant pas encore qu'elle est primitive car elle peut être équivalente à une action qui ne l'est pas. Par contre si la dénotation de act est connue, c'est-à-dire si act est un désignateur rigide alors une action exécutable est associée à ce nom et nous pouvons affirmer que $act(t_1, \ldots, t_n)$ dénote une action primitive. Nous avons donc que :

$$primitive(act(t_1, \ldots, t_n)) \leftrightarrow [B]act$$

Donc si $[B]act$ est vrai alors la condition d'exécutabilité d'une action s'exprime par la formule :

$$p \leftrightarrow executable(act(t_1, \ldots, t_n))$$

L'exécutabilité des autres actions s'en déduit par les implications suivantes :

$$(a_1 = a_2) \wedge executable(a_1) \rightarrow executable(a_2)$$

et pour la séquence :

$$executable(a_1) \wedge [a_1]executable(a_2) \rightarrow executable(a_1; a_2)$$

Notons que cette dernière définition nous permet d'avoir une séquence de deux actions dont la première assure l'exécutabilité de la seconde. Notamment, l'exécutabilité de la seconde action n'a besoin d'être établie qu'après l'exécution de la première action.

3.3.4 Etre capable de faire une action

Etant donné que nous avons soigneusement distingué la description des effets, des conditions d'exécutabilité, nous avons dans notre formalisation la formule :

$$executable(act) \wedge [B]act \rightarrow [can]act$$

L'avantage de cette formulation est double :

- elle ne dépend pas des effets de l'action qui n'interviennent pas dans cette notion. Nous séparons bien exécutabilité et description des effets.
- elle ne fait pas intervenir explicitement de quantificateur existentiel. La sémantique est donc plus facile à donner.

Cependant, si *act* est une séquence de deux actions, cette formule ne définit "être capable" que si nous connaissons la seconde action avant même d'avoir exécuté la première. Or, il est possible que la première action nous apporte des connaissances nous permettant de déterminer ce qu'est la seconde action de la même façon que la première action peut rendre la seconde exécutable. Il nous faut donc définir l'opérateur $[can]$ dans le cas d'une séquence de la façon suivante :

$$[can]a_1; a_2 \leftrightarrow [can]a_1 \wedge [a_1][can]a_2$$

La formule :

$$executable(act) \wedge [B]act \rightarrow [can]act$$

n'est donc applicable que si l'action est primitive.

3.4 Discussion

Nous pouvons faire un parallèle entre le processus par lequel un agent peut chercher à connaître une action et la génération hiérarchique de plans d'actions. Dans l'un et l'autre des cas, il ne s'agit pas de découvrir comment une action peut être décomposée en sous-actions mais de rendre explicite une information qui existe implicitement dans les connaissances données par l'utilisateur.

Si nous avons l'équivalence suivante :

$$a = a_1; a_2$$

et que nous cherchons à faire l'action a, nous cherchons d'abord à connaître l'action a ainsi que nous l'avons expliqué dans l'introduction de ce chapitre. Si a est équivalent à la séquence $a_1; a_2$ ($\mu(a = a_1; a_2) = vrai$, nous pouvons connaître a si nous arrivons à spécifier a_1 et a_2 respectivement. Nous avons bien l'équivalent de la décomposition d'un problème en sous-problèmes. Remarquons que nous pouvons aussi utiliser l'équivalence dans l'autre sens. En effet, si nous cherchons à déterminer $a_1; a_2$, nous pouvons le faire en déterminant l'action a. Nous obtenons alors un processus dans lequel plusieurs sous-problèmes peuvent se réduire en un seul. C'est une généralisation de la structure strictement arborescente des générateurs hiérarchiques de plans d'actions.

Etudions le cas où l'équivalence dépend du contexte. Pour cela, prenons l'exemple suivant :

Prendre(A) = Descendre-bras;Serrer-pince;Monter-bras

Ces deux actions ne peuvent être équivalentes que sous certaines conditions, notamment si le bras est au-dessus de A et si A est libre. L'équivalence des deux actions est donc subordonnée aux circonstances. Nous avons vu dans le paragraphe sur l'exécutabilité (voir 3.3.3) que la condition d'exécutabilité de *Prendre(A)* doit contenir la condition d'équivalence.

Pour mieux faire comprendre le lien entre cette condition et ce que nous appelons une précondition en génération de plans d'actions, voyons schématiquement les représentations classiques.

Dans les générateurs hiérarchiques de plans d'actions, la description d'une action est la suivante (Cf. NOAH[41] et NONLIN[47]) :

α
précondition : p
corps : β
effet : q

Cette représentation exprime approximativement que l'action α peut se décomposer dans le plan β avec la précondition p et l'effet q.

Nous avons vu dans le paragraphe sur la logique dynamique (cf 3.2.1) que la précondition p peut exprimer à la fois la condition pour que l'action s'exécute et

pour qu'elle ait l'effet $\cdot q$ si nous prenons la spécification forte de la relation de transition. Nous voyons maintenant apparaître la précondition comme condition pour que β soit équivalente à α.

Nous pouvons en déduire que la notion de précondition en génération classique de plans d'actions recouvre les significations suivantes :

- c'est la condition pour que faire β soit équivalent à faire α. Par exemple, si nous sommes au bord d'une route, lever le pouce nous permet de faire de l'auto-stop.

- c'est la condition pour que q soit vrai après β (ou α puisque ces actions deviennent équivalentes). Par exemple, si la lampe est éteinte, après avoir appuyé sur l'interrupteur, la lampe est allumée.

- c'est la condition physique pour que β (ou α) soit exécutable. Par exemple, si le cube A est libre, prendre le cube A est exécutable.

Si un agent désire être capable de faire une action *act* c'est-à-dire désire satisfaire la formule :

$$[can]act$$

Il doit :

- connaître l'action *act* ce qui l'amène éventuellement à décomposer cette action en sous-actions afin de déterminer comment la faire. Nous avons donc une justification épistémique de la décomposition hiérarchique.

- rendre *act* exécutable en satisfaisant les préconditions physiques.

Le processus de génération de plans d'actions est donc entièrement inclus dans le raisonnement sur la capacité d'un agent à faire quelque chose ainsi que nous allons le voir de façon détaillée dans le chapitre suivant.

Chapitre 4

Modélisation des désirs

4.1 Introduction

L'agent doit coordonner rationnellement ses activités de planification, perception, exécution et déduction. Dans le chapitre 1, nous avons distingué trois notions permettant de motiver les activités d'un agent :

- les finalités,
- les désirs,
- les intentions.

Nous avons précisé également que les intentions forment un sous-ensemble des désirs choisis par un processus décisionnel, et que les finalités sont des désirs de plus haut niveau. Nous avons donc choisi de faire de la notion de désir, la notion centrale contrôlant les activités du système.

Chaque désir est responsable d'une partie de l'activité d'ensemble du système. Si le comportement est rationnel dans le sens limité que nous avons proposé dans l'introduction, un certain nombre de régularités peuvent être observées. Par exemple, si notre robot désire que le gazon soit mouillé, il regarde tôt ou tard l'état du gazon pour savoir si il est déjà mouillé ou non. De même, si le gazon est sec, il l'arrose. Or ces actions sont causées par des intentions et ces intentions sont issues des désirs du système (voir 1.3.2). Donc si le robot désire que le gazon soit mouillé, il a aussi le désir de savoir si il est mouillé ou le désir de l'arroser. Il produit alors les différentes intentions qui causent les actions observées.

Le but de ce chapitre est :

- de déterminer l'ensemble des désirs qui doivent être présents pour pouvoir obtenir le comportement décrit ci-dessus,
- de déterminer dans quelles conditions ces désirs sont présents,
- de comprendre comment ils s'articulent.

Nous appelons *contraintes de rationalité* l'ensemble des relations qui lient les désirs aux connaissances que le système possède du monde extérieur et aux autres désirs.

Dans la suite, nous imposons la contrainte purement syntaxique qu'un désir ne porte que sur une proposition. Cette formulation exprime que la proposition doit être satisfaite. En conséquence, nous excluons l'expression directe des désirs suivants :

- le désir de faire quelque chose, par exemple de marcher,
- le désir d'un objet, par exemple lorsque l'on désire une nouvelle voiture,

étant donné que le premier désir porte sur une action et le second sur un objet.

Nous avons bien précisé que notre hypothèse exclut l'expression directe, mais cela ne veut pas dire que ce ne soit pas exprimable. De par la discussion que nous avons faite dans le chapitre précédent sur les actions, pour que l'agent fasse quelque chose, il faut qu'il en soit capable. Donc si il désire marcher, il doit être capable de marcher. Ce désir implique qu'il désire savoir comment marcher et rendre cette action exécutable. Donc, le désir de marcher peut s'exprimer par le désir d'être capable de marcher.

Dans le deuxième cas, nous pouvons désirer posséder un objet ou désirer en avoir un momentanément à disposition. Il est différent de posséder un marteau ou d'avoir besoin d'un marteau pour planter un clou dans le mur. Dans tous les cas, cette situation peut s'exprimer sous la forme d'une action d'acquisition de l'objet et donc sous la forme d'un désir d'être capable de faire cet acte d'acquisition.

Nous présentons les contraintes de rationalité dans le paragraphe suivant. Elles nous suffisent pour spécifier ce que nous voulons faire. Notamment, dans le cadre que nous nous sommes fixé, l'agent n'a pas besoin de raisonner sur ses propres désirs. Nous n'essayons donc pas de donner une sémantique formelle aux formules de la forme:

$$[desire]p$$

mais nous exprimons quelles formules de ce type doivent être vraies simultanément et sous quelles conditions.

Cependant, nous présentons une tentative de formalisation des désirs par la sémantique des mondes possibles puis nous suggérons une sémantique formelle dans laquelle les désirs sont interprétés en termes d'états des noeuds d'un réseau qui matérialise les contraintes de rationalité.

Nous concluons par la comparaison de notre modélisation avec l'analyse de Allen[1] sur le raisonnement sur les intentions d'un agent pour la reconnaissance de plans.

4.2 Les contraintes de rationalité

Les conditions sous lesquelles un désir doit être présent, peuvent être vues de plusieurs façons selon l'usage que nous voulons en faire :

- si nous cherchons à comprendre les motivations d'un agent, il est utile de savoir que le désir de savoir la valeur de vérité d'une proposition peut être dû au désir que cette proposition soit vérifiée ou, de la même façon, que le désir d'identifier un objet est dû au désir de le manipuler.

- si nous voulons construire un agent qui se comporte de façon rationnelle dans le sens que ses activités sont orientées par ses motivations, ces conditions peuvent nous servir de spécification pour engendrer des désirs à partir d'autres désirs.

Dans l'un ou l'autre cas, ces conditions apparaissent comme des contraintes sur les désirs possibles de l'agent, c'est pourquoi nous les avons appelées des contraintes de rationalité.

Nous allons exprimer ces contraintes par un ensemble de formules logiques qui exprime dans quelles conditions un agent désire ou ne désire pas quelque chose. Nous disons qu'un agent désire p si la formule $[desire]p$ est vraie et qu'il ne désire pas p si elle est fausse. Provisoirement, nous ne distinguons pas entre le désir de quelque chose et la valeur de vérité de la formule qui décrit ce désir puisque nous ne nous intéressons pas dans ce paragraphe à la sémantique formelle des désirs.

4.2.1 Satisfaction d'un désir

Un agent ne désire pas quelque chose qui est déjà réalisé. La première contrainte exprime donc dans quelle condition un agent ne désire pas quelque chose :

$$cond \rightarrow \neg[desire]cond \tag{4.1}$$

Cette contrainte est essentielle à la cohérence de notre modèle. Dans toutes les contraintes qui suivent, un désir n'est présent que si la condition n'est pas déjà vérifiée ou si elle est inconnue. Cette condition lie donc l'existence d'un désir aux connaissances que le système possède du monde extérieur et de lui-même. Dans la suite, nous n'écrivons pas systématiquement cette condition afin de ne pas alourdir l'écriture.

Si le système désire satisfaire une condition, cette dernière peut être inconnue ou fausse. Ces deux états conditionnent deux types d'activité :

- les activités d'acquisition de connaissances si la condition n'est pas connue du système,

- les activités de planification si la condition est fausse.

Nous allons donc présenter dans quelles conditions apparaissent les désirs qui sont à l'origine de ces deux types d'activité. Nous présentons ensuite le processus de planification lui-même.

4.2.2 Motivation de l'acquisition de connaissances

Nous avons dit dans l'introduction (chapitre 1) que notre robot jardinier ne reste pas inactif même si le gazon est déjà mouillé puisqu'il cherche en fait à vérifier si le gazon est effectivement mouillé afin de mettre à jour ses représentations du monde extérieur. A l'origine de cette activité, il faut donc le désir d'acquérir une information. Une contrainte lie le désir de satisfaire une condition au désir de savoir si la condition est satisfaite :

$$[desire]cond \wedge \neg[B]cond \rightarrow [desire][B]cond \tag{4.2}$$

Le désir de savoir si la condition est vraie ou fausse motive les déductions et les actes perceptifs afin d'acquérir l'information. Si la condition est déjà vraie, il ne désire pas la réaliser et par conséquent, il n'a pas besoin de savoir si elle vraie ou non.

Une remarque s'impose si le désir porte déjà sur une condition de la forme $[B]cond$. Parce que le système sait exactement ce qu'il sait et ne sait pas, il est toujours vrai que $[B][B]cond$. Donc un désir de la forme :

$$[desire][B][B]cond$$

n'existe jamais. Cette situation est une conséquence importante et souhaitable de notre modélisation. En effet, le but essentiel du désir de savoir quelque chose est d'acquérir de l'information soit sur les actions, soit sur le monde extérieur. Savoir si l'on sait quelque chose ne nous apporte aucune information utile. Nous serions dans la situation de la personne à qui nous demandons si il connaît l'heure et qui répondrait : "oui" sans plus de détail. De plus, nous avons expliqué dans le chapitre 2 que si $[B]cond$ alors $[B][B]cond$ donc il suffit d'avoir un désir de la forme :

$$[desire][B]cond$$

Le désir de savoir quelque chose ne motive pas seulement des actes perceptifs ou déductifs. Un cas particulier est lié à la propriété d'extentionalité de notre langage (voir 2.3.2) qui dit que :

$$\mu(f(t_1,\ldots,t_n)) = \mu(f) \circ (\mu(t_1),\ldots,\mu(t_n)) \tag{4.3}$$

Par conséquent, nous pouvons connaître $f(t_1,\ldots,t_n)$ si nous connaissons la fonction f et les arguments t_1 à t_n. Donc il est possible de satisfaire $[B]f(t_1,\ldots,t_n)$ si nous avons $\mu([B]f) = vrai$, $\mu([B]t_1) = vrai$, etc. De ces remarques nous dérivons donc les contraintes suivantes :

$$\begin{array}{l} [desire][B]f(t_1,\ldots,t_n) \rightarrow [desire][B]f \\ [desire][B]f(t_1,\ldots,t_n) \rightarrow [desire][B]t_1 \\ \vdots \\ [desire][B]f(t_1,\ldots,t_n) \rightarrow [desire][B]t_n \end{array} \tag{4.4}$$

4.2.3 Motivation de la planification

Si un agent désire qu'une condition *cond* soit vraie et qu'une action a pour effet de rendre *cond* vraie, alors il désire faire cette action. Pour pouvoir faire une action, il doit d'abord prouver qu'il est capable de la faire. Nous obtenons la contrainte suivante :

$$[desire]cond \wedge \neg cond \wedge [act_i]cond \rightarrow [desire][can]act_i \tag{4.5}$$

Remarquons que le désir de faire une action ne signifie pas que l'agent l'exécutera. L'exécution effective dépend en dernier ressort du processus de décision (et donc du choix) qui formule l'intention.

Etudions plus en détail la condition liée à ce désir de faire une action. Etant donné que le désir qu'une condition soit vraie est à l'origine du désir de savoir si cette condition est vraie ou fausse, le système sait tôt ou tard si la condition est fausse. Si il est incapable d'obtenir l'information, la contrainte ne sera pas applicable. Mais, il peut très bien ne pas savoir qu'il existe une action qui permette de la réaliser, c'est -à-dire que:

$$\mu([act_i]cond)$$

n'est pas défini. Par hypothèse de rationalité, l'agent ne tire aucune conclusion des connaissances qu'il possède sans motivation. Donc il peut ne jamais savoir qu'une action produit l'effet désiré même si cette conclusion est déductible des connaissances qu'il possède. Il faut donc qu'il désire trouver une action permettant de réaliser la condition. C'est pourquoi, nous devons ajouter la contrainte suivante :

$$[desire]cond \wedge \neg cond \rightarrow [desire][act_i]cond \tag{4.6}$$

pour toutes les actions act_i possibles. Il faut comprendre par cette formule que le système désire qu'il existe une action produisant l'effet souhaité.

Il s'agit maintenant d'énumérer les conditions sous lesquelles ce désir peut être satisfait. Nous pourrons alors en déduire les désirs qui doivent dépendre du désir qu'il existe une action réalisant un certain effet. Nous pouvons déterminer quatre conditions :

$$[act_i]cond$$

$$p \rightarrow [act_i]cond$$

$$[act_i]q \wedge (q \rightarrow cond)$$

$$(p \rightarrow [act_i]q) \wedge (q \rightarrow cond)$$

Dans le premier cas, le désir est trivialement satisfait par les connaissances du système. Il n'apparaît donc pas à cause de la contrainte 4.1. Dans les trois autres cas, il est nécessaire que le système puisse motiver une activité d'acquisition d'information afin de pouvoir déduire l'effet d'une action. La contrainte de rationalité 4.2 s'applique tout naturellement à ce cas. Donc, si le système désire qu'il

existe une action qui produise un certain effet, il désire savoir si il connaît une telle action, ce qui se traduit par la contrainte :

$$[desire][act_i]cond \rightarrow [desire][B][act_i]cond$$

qui n'est donc qu'une instance de la contrainte 4.2. Le désir qu'il existe une certaine action amène donc l'agent à désirer savoir si elle existe.

4.2.4 Etre capable de faire une action

Etudions les conséquences de la contrainte 4.5. La motivation du raisonnement sur l'exécutabilité des actions découle de la définition de l'opérateur $[can]$ (voir 3.3.4). Cette définition repose sur les notions d'action primitive et d'action non-primitive.

Une action est primitive si la fonction qui la décrit est connue :

$$[B]act \rightarrow primitive(act(t_1, \ldots, t_n))$$

Toute action équivalente à une action primitive est primitive :

$$a = b \wedge primitive(b) \rightarrow primitive(a)$$

Une séquence n'est pas une action primitive :

$$\neg primitive(b_1; b_2)$$

Finalement, toute action équivalente à une action non-primitive est non-primitive :

$$a = b \wedge \neg primitive(b) \rightarrow \neg primitive(a)$$

Actions primitives

Si une action est primitive, nous sommes capables de la faire si elle est exécutable. Le système doit désirer qu'une action soit exécutable si elle est primitive :

$$[desire][can]action \wedge primitive(action) \rightarrow [desire]executable(action) \quad (4.7)$$

Le système doit également savoir de quelle action il s'agit (préconditions mentales) :

$$[desire][can]action \wedge primitive(action) \rightarrow [desire][B]action \quad (4.8)$$

La satisfaction conjointe de ces deux désirs entraîne la satisfaction de$[can]action$.

Actions non-primitives

Nous devons également traiter le cas des actions composées. Etre capable de faire une telle action est défini par (voir 3.3.4) :

$$[can]b_1; b_2 \leftrightarrow [can]b_1 \wedge [b_1][can]b_2$$

Donc nous avons également les contraintes suivantes :

$$\begin{array}{l} [desire][can]b_1; b_2 \rightarrow [desire][can]b_1 \\ [desire][can]b_1; b_2 \rightarrow [desire][b_1][can]b_2 \end{array} \quad (4.9)$$

Nous pouvons aussi avoir une description d'action sans séquence qui soit équivalente à une action composée. Nous avons alors que :

$$\begin{array}{l} [desire][can]a \wedge \neg primitive(a) \wedge a = b_1; b_2 \rightarrow [desire][can]b_1 \\ [desire][can]a \wedge \neg primitive(a) \wedge a = b_1; b_2 \rightarrow [desire][b_1][can]b_2 \end{array} \quad (4.10)$$

L'ensemble des formules 4.7 et 4.8 s'applique aux actions primitives alors que les formules 4.9 et 4.10 concernent les actions non-primitives. Pour savoir quelles formules appliquer, le système doit désirer savoir si l'action est primitive ou non. Nous avons donc la contrainte additionnelle suivante :

$$[desire][can]a \rightarrow [desire][B]primitive(a) \quad (4.11)$$

Si $primitive(a)$ est faux, il existe une séquence $b_1; b_2$ telle que $a = b_1; b_2$ et le second ensemble de contraintes (4.9 et 4.10) est applicable puisque la condition peut être vérifiée sans avoir à motiver d'autres acquisitions de connaissances.

Exemple Montrons l'utilisation de ces contraintes dans le cas d'une action perceptive. Nous avons vu un exemple d'action perceptive dans le chapitre précédent (voir 3.3.2) où nous avions :

$$[percevoir('couleur(ciel))][B]couleur(ciel)$$

Si le système possède le désir suivant :

$$[desire][B]couleur(ciel)$$

le système a le désir d'être capable de faire cet acte perceptif :

$$[desire][can]percevoir('couleur(ciel))$$

Pour cela il faut que l'action de percevoir soit connue ce qui est trivialement vérifié si *percevoir* est une action primitive du système, $'couleur(ciel)$ dénotant l'expression elle-même. D'autre part, il faut que le désir suivant soit satisfait :

$$[desire]executable(percevoir('couleur(ciel)))$$

Si l'exécutabilité est déjà satisfaite, le système est donc capable de faire cette action. Par contre si la condition n'est pas satisfaite, il lui faut savoir si l'action est exécutable :

$$[desire][B]executable(percevoir('couleur(ciel)))$$

Le problème est maintenant de déduire cette exécutabilité. Etant donné que les règles d'inférence ne font pas partie de notre modélisation, il faut les décrire à l'aide d'actions déductives. Donc nous devons exploiter des connaissances de la forme :

$$p \rightarrow q$$

de façon à ce que le système cherche à vérifier ou à tester la condition p pour pouvoir déduire q. La règle de Modus Ponens rend cette implication équivalente aux deux descriptions suivantes :

$$p \leftrightarrow executable(denoter('q, vrai))$$

$$[denoter('q, vrai)][Believe]q$$

Nous utilisons le symbole d'équivalence puisqu'une condition d'exécutabilité est nécessaire et suffisante. L'action *denoter* permet d'introduire une nouvelle dénotation qui ici envoie la phrase q dans la valeur de vérité *vrai*.

Prenons un autre cas. Si nous avons l'équivalence suivante :

$$ouvert(volets) \leftrightarrow executable(percevoir('couleur(ciel)))$$

L'équivalence nous dit que $executable(percevoir('couleur(ciel)))$ a toujours la même valeur de vérité que $ouvert(volets)$. Le désir suivant en découle :

$$[desire][can]denoter(\ 'executable(percevoir('couleur(ciel))),\ ouvert(volets))$$

d'où :

$$[desire]executable(\ denoter('executable(percevoir('couleur(ciel))),\ ouvert(volets)))$$

et

$$[desire][B]denoter('executable(percevoir('couleur(ciel))), ouvert(volets))$$

Le premier désir est satisfait puisque c'est précisément ce qu'affirme l'équivalence. Par contre, le second n'est satisfait que si les désirs suivants sont satisfaits :

$$[desire][B]denoter$$

$$[desire][B]'executable(percevoir('couleur(ciel)))$$

$$[desire][B]ouvert(volets)$$

à cause de la contrainte 4.4. Le deuxième désir est toujours satisfait puisque :

$$\mu('executable(percevoir('couleur(ciel)))) = executable(percevoir('couleur(ciel)))$$

Si nous supposons que le système connaît l'action ***denoter***, le système désire savoir si les volets sont ouverts ce qui est exactement le comportement souhaité.

Un problème survient si les volets ne sont pas ouverts. En effet, les contraintes introduites jusqu'à présent assurent que le système vérifie que les volets soient ouverts, mais elles ne motivent pas l'ouverture des volets si il s'avère qu'ils sont fermés. C'est ce problème auquel nous allons apporter une solution dans la section suivante.

4.2.5 Réalisation d'une condition

Les contraintes que nous avons données jusqu'à présent nous amènent à tester systématiquement les conditions permettant de déduire certaines informations, notamment qu'une action est exécutable ou qu'elle a un certain effet. Cependant, l'agent doit également chercher à réaliser une de ces conditions pour pouvoir exécuter une action ou pour qu'elle ait l'effet voulu. En effet, il est différent de désirer que la condition soit vraie ou de désirer que la condition soit connue comme vraie. Par exemple :

$$[desire]ouvert(volets)$$

exprime que le système désire que la condition $ouvert(volets)$ soit vraie, et

$$[desire][Believe]ouvert(volets)$$

exprime que nous désirons savoir que cette condition est vraie.

Nous voulons dans le cas général pouvoir décider de seulement tester une condition ou de la réaliser. La décision doit se prendre au niveau du contrôle où il doit être possible d'abandonner un désir lorsqu'un simple test donne des résultats satisfaisants. Le problème se situe donc au niveau décisionnel et sort du cadre que nous nous sommes fixé. Nous prenons donc l'option la moins limitative en engendrant systématiquement le désir de réaliser la condition.

Nous avons supposé jusqu'ici qu'une action est toujours exécutable. Une action *act* peut ne pas l'être dans deux cas :

- soit le système possède l'information explicite que l'action n'est pas exécutable :

  $$\mu(executable(act)) = faux$$

- soit il existe une règle de la forme :

  $$p \rightarrow executable(act)$$

  qui, si nous prenons en compte la sémantique opérationnelle de Modus Ponens, est équivalente à :

  $$\mu(executable(denoter('executable(act), p))) = vrai$$

Le premier cas est incontournable. Si le système sait explicitement qu'une action n'est pas exécutable, il est impossible de la rendre exécutable. Néanmoins, cette assertion peut être utile dans un contexte multi-agent. En effet, il est possible qu'un agent A ne puisse pas exécuter une action mais qu'un autre agent B en soit capable. Dans ce cas, il faut planifier une requête pour que l'agent B exécute l'action.

Dans le second cas, nous voyons que p doit être vrai pour que l'action *act* soit exécutable. Nous obtenons la contrainte suivante :

$$\begin{array}{l} [desire]executable(action) \wedge \\ \neg executable(action) \wedge \\ executable(denoter('executable(action), cond)) \\ \rightarrow [desire]cond \end{array} \tag{4.12}$$

Le système a donc le désir de rendre vraie toute condition qui rend une action exécutable. Il revient à la composante décisionnelle (voir 1.3.1) de décider de satisfaire effectivement cette condition.

Le système raisonne sur l'exécutabilité de trois types d'action :

- les actions externes dont l'effet est de satisfaire une proposition qui porte sur le monde extérieur,
- les actions perceptives dont l'effet est de satisfaire une proposition de la forme $[B]terme$ où *terme* est une proposition ou doit dénoter un objet du monde extérieur,
- les actions déductives dont l'effet est d'établir une condition interne au système, à savoir :
  - l'exécutabilité d'une action,
  - la connaissance de l'effet d'une action.

En conséquence, le système essaie non seulement de satisfaire les préconditions d'une action externe ou perceptive mais aussi de satisfaire une condition dont il peut déduire :

- qu'une action est exécutable,
- qu'une action a un certain effet.

Nous sommes donc capables d'une part de réaliser les conditions dans lesquelles une action a l'effet désiré et d'autre part de réaliser les conditions dont nous pouvons déduire de façon arbitrairement compliquée qu'elles rendent une action possible.

4.3 Vers une sémantique formelle

4.3.1 La sémantique des mondes possibles

Essayons d'introduire une sémantique à la notion de désir. Si un agent désire que le gazon soit mouillé, nous pouvons exprimer ce fait par la formule :

$$[desire]mouillé(gazon)$$

Dans cette expression, le désir est un opérateur intensionnel car la valeur de vérité de cette formule dépend non seulement de la valeur de vérité de la proposition :

$$mouillé(gazon)$$

mais aussi d'une certaine attitude de l'agent vis-à-vis de cette proposition. La sémantique sur les mondes possibles vient tout de suite à l'esprit puisque cette modélisation permet de rendre compte des phénomènes liés aux opérateurs intensionnels.

Soit donc une structure de modèle $M = (S, \phi, R)$ (comme définis dans les paragraphe 2.2.1 et 3.2.1), la formule :

$$[desire]mouillé(gazon)$$

est vérifiée dans une interprétation (M, s) si la proposition $mouillé(gazon)$ est vraie dans tous les mondes t tels que (s, t) appartient à R. C'est exactement la même formulation que pour la logique épistémique et la logique dynamique.

Pour que cette sémantique corresponde effectivement à la formalisation de la notion de désir, il faut que (s, t) soit dans R si et seulement si t est un monde souhaitable du point de vue de l'agent quand il est dans l'environnement décrit par s. Il reste alors à déterminer ce que veut dire un monde souhaitable. Nous disons qu'un monde souhaitable est un monde dans lequel un maximum de propositions désirées et non-contradictoires sont vérifiées.

Les axiomes de cette théorie ne sont rien d'autre que les contraintes de rationalité que nous avons énumérées dans le paragraphe précédent.

Comme dans la sémantique des mondes possibles de la logique épistémique, cette approche permet de rendre compte des phénomènes propres à l'intensionalité de l'opérateur modal mais ne dit rien sur la représentation des désirs. Si nous voulons pouvoir construire un agent muni de désirs, il est essentiel que nous puissions imaginer une méthode d'implantation du modèle. Ceci est à mettre en parallèle avec notre modélisation des connaissances dans laquelle nous avons développé un modèle concret satisfaisant un certain nombre de contraintes. C'est ce que nous faisons dans le prochain paragraphe.

4.3.2 La structure des désirs

Nous proposons dans ce paragraphe une alternative à la sémantique des mondes possibles. Nous esquissons d'abord une structure qui obéisse aux contraintes de

rationalité et qui puisse servir de modèle pour une sémantique formelle des désirs. Finalement, nous proposons une définition de $\mu([desire]p)$ en termes de cette structure.

Les désirs dépendent les uns des autres. En effet, certains désirs n'apparaissent que si d'autres sont présents. Ainsi le désir de savoir p n'est présent que si le système désire réaliser p. Cela suggère de représenter les désirs dans un graphe orienté où :

- chaque noeud représente un désir et inversément un désir n'est représenté que par un noeud,
- chaque arc représente la dépendance entre deux désirs.

Le graphe est orienté parce que les contraintes sont exprimées par des implications ce qui montre que les dépendances ne sont pas réciproques. Un désir ne peut pas dépendre de lui-même. Le graphe obtenu est donc sans circuit.

Dans un graphe sans circuit, il existe un ou plusieurs noeuds n'ayant pas de prédécesseurs. Ces noeuds correspondent à ce que nous avons appelé les *finalités* dans le chapitre introductif.

A chaque noeud du réseau est attaché:

- la condition sur laquelle porte le désir,
- une étiquette qui est *actif* ou *inactif* selon que le désir est présent ou absent.

Etant donné que les contraintes que nous avons énoncées déterminent quand un agent désire quelque chose, nous devons pouvoir en dériver les conditions sous lesquelles un désir est actif ou inactif.

A un instant donné, la condition p associée à un noeud du réseau est vraie, fausse ou inconnue. Le comportement attendu est :

$\mu(p) = inconnu$: il faut acquérir l'information. Le désir de savoir p engendre une activité d'acquisition d'information,

$\mu(p) = faux$: il faut raisonner sur les actions qui permettent de réaliser cette condition,

$\mu(p) = vrai$: le désir ne produit aucune activité puisque sa condition est déjà satisfaite.

L'état actif ou inactif associé à un désir doit tenir compte de ce comportement. Nous avons les définitions suivantes :

Un désir est *actif* si :

- au moins un de ses supérieurs hiérarchiques est actif ou si il n'a pas de supérieurs,
- et tous ses prédécesseurs temporels sont satisfaits,

- et sa condition est inconnue ou fausse.

Dans le cas contraire, le désir est *inactif.*

Les finalités n'ont pas de supérieurs et donc sont actifs jusqu'à ce qu'ils soient satisfaits (c'est-à-dire que leur condition associée soit vraie) et redeviennent actifs chaque fois que la condition devient fausse ou inconnue.

Nous pouvons maintenant proposer une sémantique formelle. Soient:

- S une situation avec le langage L étendu avec l'opérateur intensionnel $[desire]$ et la fonction de dénotation μ,
- G un graphe orienté sans circuit dont chaque noeud est étiqueté par un couple (p, s) où:
  - p est une formule de L,
  - s est "actif" ou "inactif" selon les définitions données plus haut,

alors $\mu([desire]p) = vrai$ si et seulement si il existe un noeud $(p, actif)$ dans G.

La structure obtenue correspond à l'arborescence que construit un générateur hiérarchique et non-linéaire de plans d'actions, à la différence près que les actes déductifs et perceptifs sont inclus naturellement dans le plan. D'autre part, à chaque noeud du réseau et à certains liens sont associées des conditions qui sont vraies, fausses ou inconnues selon l'état des connaissances de l'agent à un moment donné. Il est clair que cette connaissance varie au fur et à mesure que le système évolue dans le temps, c'est-à-dire au fur et à mesure que les actions sont exécutées.

4.4 Comparaison

Très peu de travaux ont été faits sur la modélisation des désirs d'un agent. Le seul domaine dans lequel une telle analyse ait été développée, est la compréhension de la langue naturelle et notamment la reconnaissance de plans. Pour notre comparaison, nous nous basons essentiellement sur les travaux de Allen [1].

Présentation des travaux de Allen

De la même façon que nous introduisons plusieurs opérateurs modaux pour décrire la connaissance d'un agent, Allen introduit un certain nombre de constructions :

A BELIEVE P : pour décrire qu'un agent A croit que P est vrai.

A KNOW P : si l'agent A sait que P.

A KNOWIF P : si l'agent A sait si P est vrai ou non

A KNOWREF D : si l'agent A connaît la dénotation du terme D

Les descriptions suivantes expriment le point de vue d'un observateur extérieur sur les intentions d'un agent A. L'intention qu'un agent A fasse une action Act est exprimée par le prédicat :

$$WANT(A,Act)$$

Il s'agit ici d'une intention car Allen raisonne sur ce que l'agent veut effectivement faire et non pas sur ce qu'il désire faire. En contrepartie, pour faire de l'inférence de plans, Allen doit émettre des hypothèses sur les intentions possibles de l'agent. Nous retrouvons la plupart des contraintes que nous avons citées plus haut mais avec les implications dans l'autre sens puisqu'il faut cette fois-ci deviner ce que l'agent veut faire à partir des actions observées. Les liens entre les différentes intentions s'articulent autour des notions classiques en génération de plans d'actions, à savoir :

- les préconditions,
- les effets,
- la notion de sous-plan.

Le lien entre les préconditions d'une action et l'action est exprimé par la règle suivante :

$$SBAW(P) \rightarrow SBAW(ACT) \tag{4.13}$$

si P est une précondition de ACT. $SBAW(P)$ est un raccourci pour S $BELIEVE$ $WANT(A,P)$. Cette règle exprime que si l'agent S pense que l'agent A veut que P soit vrai, alors S peut en déduire que l'agent A veut faire l'action dont P est une précondition.

Une action et ses effets sont également liés. Ce lien s'exprime par la règle :

$$SBAW(ACT) \rightarrow SBAW(E) \tag{4.14}$$

si E est un effet de ACT. Donc si l'agent A veut faire une action, c'est probablement qu'il veut réaliser un des effets de l'action.

Enfin, les liens entre une sous-action et l'action dont elle fait partie sont exprimés par des formules de la forme :

$$SBAW(B) \rightarrow SBAW(ACT) \tag{4.15}$$

où B est une sous-action de ACT.

L'originalité des travaux de Allen repose sur la prise en compte de la communication entre agents. Si un agent pose une question, il est facile d'en déduire qu'il cherche une certaine information. A partir de là, un certain nombre d'autres intentions peuvent être inférées.

En ce qui concerne les propositions, nous avons les deux règles suivantes :

$$SBAW(A\ KNOWIF\ P) \rightarrow SBAW(P) \tag{4.16}$$

$$SBAW(A\ KNOWIF\ P) \rightarrow SBAW(\neg P) \tag{4.17}$$

Elles expriment que si l'agent veut savoir si une condition est vraie ou fausse, il veut probablement que cette condition ait une certaine valeur de vérité.

Un autre ensemble de règles permet de traiter les références de termes :

$$SBAW(A\ KNOWIF\ P(a)) \rightarrow SBAW(A\ KNOWREF\ x{:}P(x)) \tag{4.18}$$

$$SBAW(A\ KNOWREF\ x{:}D(x)) \rightarrow SBAW(P(x{:}D(x))) \tag{4.19}$$

La première règle (4.18) exprime que si l'agent veut savoir si P est vrai pour un certain objet, c'est peut être qu'il veut savoir quel objet a la propriété P. La seconde règle (4.19) exprime que si un agent veut savoir quel objet possède une certaine propriété, c'est qu'il veut faire quelque chose avec cet objet (i.e. satisfaire une certaine propriété P pour cet objet).

Les autres formules décrites par Allen concernent le raisonnement multi-agent et ne sont donc pas comparables directement avec notre formalisme.

Discussion

Une première remarque concerne l'utilisation de divers opérateurs pour exprimer la connaissance qu'un agent possède. Nous avons introduit les opérateurs $[B]$ et $[Believe]$ pour exprimer le savoir-quoi et le savoir-si d'une part et le savoir-que d'autre part. Allen considère aussi que cette panoplie d'opérateurs est nécessaire pour décrire avec la souplesse voulue les échanges d'information puisqu'il introduit *KNOWREF* et *KNOWIF* d'une part et *KNOW* d'autre part.

Nous pouvons noter quelques différences entre notre formalisation et celle de Allen :

- chez Allen, un agent peut tout aussi bien vouloir une proposition qu'une action. Il y a donc une certaine ambiguïté que nous avons résolue en utilisant la notion d'être capable.
- Allen utilise la notion STRIPS de précondition, ce qui ne permet pas de distinguer entre la réalisation d'une condition pour rendre l'action exécutable et la réalisation d'une condition pour avoir l'effet voulu (voir 3.4).

La règle 4.13 concernant le lien entre les préconditions d'une action et l'action, correspond aux désirs suivants dans notre formalisation. Le désir :

$$[desire][can]act$$

implique que le système désire que l'action soit exécutable :

$$[desire]executable(act)$$

Si p est la condition qui falsifie l'exécutabilité de act, nous avons le désir :

$$[desire]p$$

Cette liaison nous permet de mettre au jour une autre déficience du modèle de Allen. En effet, il suffit dans son cas que p soit une précondition de *act*. Or l'agent peut très bien ne pas connaître cette précondition. Dans notre formalisation, ces désirs ne coexistent que si l'agent sait si p est vrai ou faux et plus particulièrement si il sait que p est faux.

D'autre part, le formalisme de Allen ne permet pas de détecter une activité qui a pour but d'acquérir de l'information sur l'action elle-même.

En ce qui concerne l'acquisition d'information, la règle 4.16 est l'exacte parallèle de :

$$[desire]p \rightarrow [desire][B]p$$

L'équivalent de la règle 4.17 s'obtient par la contrainte de rationalité liée à l'extentionalité (4.4). En effet, par application de la contrainte de rationalité 4.2, nous obtenons :

$$[desire][B]\neg p$$

puis par extentionalité de la négation qui n'est qu'une fonction comme une autre dans notre langage, nous obtenons conjointement que :

$$[desire][B]\neg$$

et

$$[desire][B]p$$

ce qui donne l'équivalent de la règle 4.17. Etant donné que nous ne distinguons pas entre savoir-si et savoir-quoi dans notre formalisation, nous n'avons pas besoin de règles supplémentaires alors que Allen doit donner des règles distinctes pour connaître le référant d'une expression (*KNOWREF*). En effet, si l'agent veut que $p(a)$ soit vrai, nous avons :

$$[desire]p(a)$$

ce qui induit qu'il veut connaître $p(a)$:

$$[desire][B]p(a)$$

et par extentionalité, il veut connaître a :

$$[desire][B]a$$

Notre formalisation ne contredit pas la théorie proposée par Allen. Elle apporte des nuances supplémentaires liées d'une part à la distinction systématique entre l'état du monde et les connaissances que le système possède de cet état et d'autre part à la formalisation que nous avons faite des actions. En effet, elle repose sur la logique dynamique et l'analyse des interactions connaissance-action et non pas sur le modèle proposé pour les générateurs classiques de plans d'actions.

Chapitre 5

Le système CHAOS

5.1 Présentation générale

Le système CHAOS implante les principes et les modèles que nous avons développés dans les chapitres précédents avec un certain nombre de restrictions que nous expliciterons. Nous retrouvons les trois structures de base d'un agent rationnel :

- les croyances : implantation de la notion de situation,
- les désirs : organisés dans un graphe orienté sans circuit,
- les intentions,

ainsi que les deux modules fonctionnels suivants :

- la composante intentionnelle,
- la composante décisionnelle.

Dans notre implantation, la dernière composante est simplifiée puisque nous ne voulions pas approfondir les problèmes de décision.

Le système CHAOS a été réalisé en CommonLisp sur un VAX/750 tournant UNIX. Nous avons extensivement utilisé la notion de package fourni par CommonLisp. CHAOS se compose de 21 packages dont les principaux sont :

dénotation : qui gère l'essentiel de la notion de situation.

désir : qui gère le graphe des désirs.

intention : qui choisit et exécute les actions. Il inclut la composante décisionnelle et les intentions.

interpréteur : qui correspond à la composante intentionnelle.

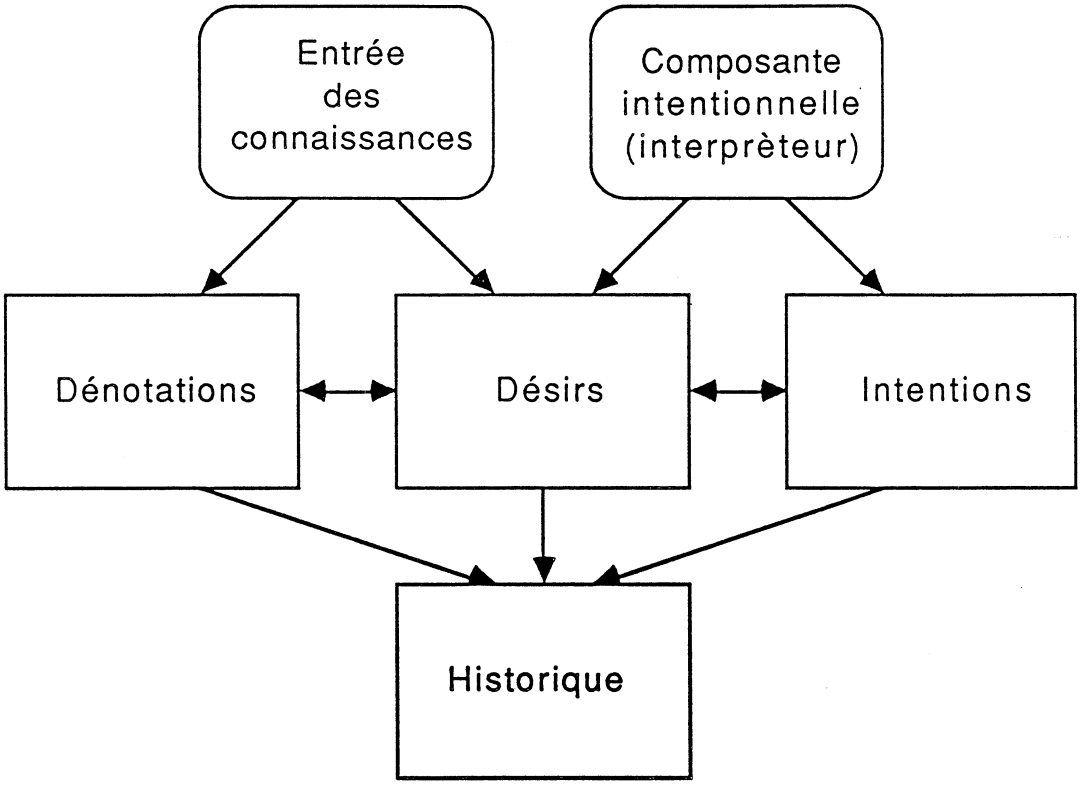

Figure 5.1: Architecture de CHAOS

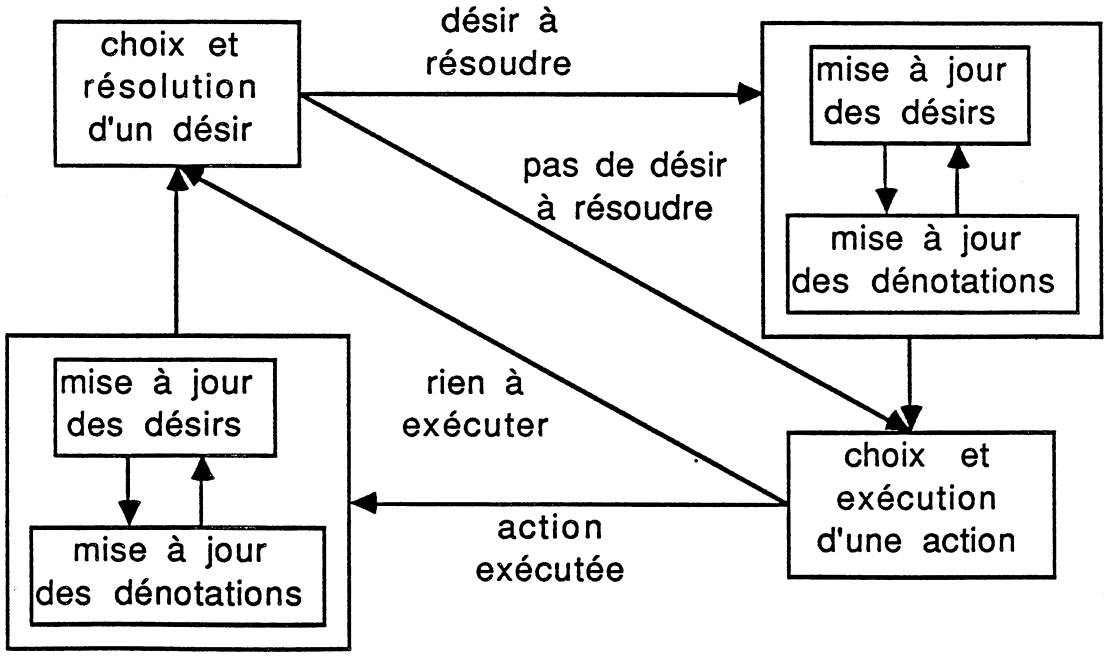

Figure 5.2: Le cycle d'exécution

Pour gèrer les mises à jour, un gestionnaire d'historique a été implanté. Il permet aussi d'obtenir des explications sur ce qui s'est passé. Cette architecture est résumée dans la figure 5.1.

Le cycle d'exécution est illustré par la figure 5.2. La composante intentionnelle choisit et résout des désirs. Elle génère de nouveaux désirs qui modifient le graphe. Pour calculer son état, le graphe des désirs interroge les croyances du système ce qui peut modifier certaines croyances et par effet de bord d'autres parties du graphe des désirs.

Lorsque l'état du système s'est stabilisé, la composante décisionnelle choisit une action disponible et l'exécute. Cette exécution a des répercutions sur ce que le système sait du monde extérieur ainsi que nous l'avons décrit dans le chapitre 3. Ces modifications peuvent changer l'état du graphe des désirs. Le cycle reprend dès que le graphe des désirs s'est de nouveau stabilisé.

Nous allons expliquer en détail ce que nous venons de schématiser. A la suite de quoi, nous présentons l'exemple du robot jardinier qui nous permet de montrer concrètement ce que le système est capable de faire. Mais auparavant, nous allons décrire comment un utilisateur peut exprimer les connaissances nécessaires au système pour qu'il sache ce qu'il doit faire et comment le faire.

5.2 Les croyances

Comme nous l'avons dit dans l'introduction de ce chapitre, les croyances du système sont une implantation à peu près directe de la notion de situation. Elle est donc constituée :

- du domaine de discours,
- du langage,
- de la description partielle de la fonction de dénotation.

Nous allons décrire successivement chacune de ces parties, étant donné que l'utilisateur peut intervenir dans chacune de ces composantes pour exprimer les connaissances dont le système a besoin.

5.2.1 Domaine de discours

Une grande partie du domaine de discours est prédéfinie. Il contient notamment :

- les valeurs de vérités,
- les nombres,
- certaines fonctions arithmétiques et booléennes ainsi que les opérations de comparaison,

- les actions primitives du système.

L'utilisateur peut définir d'autres objets de domaine de discours sous la forme d'ensembles ordonnés ou non et de nouvelles fonctions.

Nous allons décrire les actions primitives et leur usage. Elles définissent les capacités fondamentales du système.

Etant donné que ce système ne fonctionne qu'en simulation, les seules capacités concrètes du système sont la lecture des données entrées par l'utilisateur et l'affichage à l'écran de conseils sur les actions à exécuter. De façon interne, le système doit également être capable de déduire certaines informations des connaissances qu'il possède. Les capacités de base de notre système se composent donc de trois actions primitives :

- demander à l'utilisateur la dénotation d'une expression,
- demander à l'utilisateur de faire quelque chose,
- déduire la dénotation d'une expression.

Ce sont les seules actions qui existent et il n'est pas possible d'en définir d'autres. Si nous voulons connecter CHAOS avec de vrais capteurs et effecteurs, il faut actuellement modifier le système en définissant une nouvelle action pour chaque capteur et pour chaque effecteur.

Pour pouvoir décrire ces différentes actions, nous avons prédéfini un certain nombre de fonctions qui envoyent leurs paramètres dans l'ensemble des actions. Nous avons les fonctions suivantes :

(do 'exp) : qui construit l'action de faire ce qui est décrit. Cette action va afficher l'expression à l'écran. Cette expression doit décrire une action à demander à l'utilisateur. Ce dernier signale si il l'a exécutée ou non. Nous nous servons donc de l'utilisateur comme d'un effecteur.

(ask 'exp) : qui construit l'action de demander la dénotation de l'expression. A l'exécution de cette action, notre système affiche l'expression et demande la valeur à l'utilisateur. La valeur donnée est introduite comme étant la dénotation de l'expression. Nous nous servons donc de l'utilisateur comme d'un capteur.

(affect 'exp val) : qui construit l'action d'attribuer à l'expression la valeur donnée, mettant à jour la fonction de dénotation. Cette action est utilisée pour traduire la sémantique opérationnelle des règles d'inférence. Elle n'est donc pas directement accessible à l'utilisateur.

Notre système est donc capable :

- de percevoir,
- d'agir,

- de déduire.

Il faut remarquer qu'il n'existe pas d'actions pour détruire une dénotation. Cette destruction n'est faite qu'à l'exécution d'une action.

5.2.2 Le langage

Le langage dans lequel l'utilisateur peut exprimer les connaissances possède de fortes restrictions par rapport au langage L que nous avons élaboré dans les chapitres précédents. Nous présentons d'abord la syntaxe du langage. Nous explicitons ensuite la sémantique que nous lui donnons et notamment comment ces connaissances sont utilisées. Nous discutons finalement pourquoi certaines expressions de L ont été exclues du langage.

la syntaxe

Nous appelons *composants* les éléments non-décomposables de notre syntaxe. Ils comprennent:

- les nombres entiers et réels,
- les lambda-expressions de Lisp (Exemple: (lambda (x) (+ x 4))),
- les symboles,
- les expressions simples définies ci-dessous et précédées de l'apostrophe.

L'ensemble des *expressions simples* est défini récursivement de la façon suivante:

- les composants sont des expressions simples,
- si exp_1 à exp_n sont des expressions simples et f est un symbole différent des symboles *and*, *or*, *not*, $\rightarrow$, $\leftrightarrow$ ou *executable*, alors $f(exp_1 \ldots exp_n)$ est une expression simple. (Remarque: il n'y a pas de virgule entre les paramètres)

L'ensemble des *expressions booléennes* est défini de la façon suivante:

- les expressions simples sont des expressions booléennes,
- si exp_1 et exp_n sont des expressions booléennes alors $or(exp_1\ exp_2)$, $and(exp_1\ exp_2)$, $\rightarrow (exp_1\ exp_2)$, $\leftrightarrow (exp_1\ exp_2)$ et $not(exp_1)$ sont des expressions booléennes.

Pour des raisons de simplification d'écriture, l'utilisateur peut déclarer des symboles comme étant:

- *infixés* si ils sont utilisés comme des noms de fonction à deux arguments,
- *préfixés* ou *postfixés* si ils sont utilisés comme des noms de fonction à un seul argument.

Il faut également assigner à chaque symbole une *priorité* pour décider si une expression de la forme:

$$3 + 5 * 7$$

est équivalente à $+(3 * (57))$ ou bien à la formule $*(+(35)7)$. De plus, il faut leur assigner une *associativité* pour savoir si, par exemple, la formule:

$$a + b + c$$

est équivalente à $+(+(ab)c)$ ou à $+(a + (bc))$. Les opérateurs usuels sont prédéclarés.

Le langage d'expression des connaissances fourni à l'utilisateur est composé :

- des expressions simples ou de la forme *not exp* où *exp* est une expression simple, par exemple des équivalences et des relations :

$$temperature = 67$$

$$opposé(gauche\ droite)$$

- des formules de la forme $[desire]exp$ où *exp* est une expression simple, par exemple :

$$[desire]etat(gazon) = mouille$$

- des formules de la forme $[act]exp$ ou $[act][B]exp$ où *act* et *exp* sont des expressions simples, par exemple :

$$[percevoir('couleur(ciel))][B]couleur(ciel)$$

- des formules de la forme *executable(exp)* où *exp* est une expression simple et qui expriment l'exécutabilité inconditionnelle d'une action, par exemple:

$$executable(do\ 'ouvrir(bouche))$$

- des implications de la forme $bexp \rightarrow exp$, $bexp \rightarrow not\ exp$ ou $bexp \rightarrow [act]exp$ où *bexp* est une expression booléenne et *exp* et *act* sont des expressions simples. Nous pouvons à l'aide de ces implications exprimer des lois sur le monde extérieur :

$$couleur(ciel) = bleu\ ou\ taille(nuages) = petit \rightarrow beau(temps)$$

ou décrire les effets des actions:

$$eteinte(lampe) \rightarrow [(do\ 'appuyer(interrupteur))]allume(lampe)$$

- des équivalences de la forme $bexp_1 \leftrightarrow bexp_2$ ou $bexp_1 \leftrightarrow executable(exp)$ où $bexp_1$ et $bexp_2$ sont des expressions booléennes et *exp* est une expression simple. Elles permettent d'exprimer entre autre les conditions d'exécutabilité:

$$dehors(robot) \leftrightarrow executable(do\ 'arroser)$$

Utilisation des connaissances

Certaines données sont transformées pour tenir compte de leur interprétation en tant que dénotation :

- la relation $r(a_1, .., a_n)$ définit une situation où $\mu(r(a_1, .., a_n)) = vrai$.
- la négation *not* $r(a_1, .., a_n)$ définit une situation où $\mu(r(a_1, .., a_n)) = faux$.
- l'équivalence $(a = b)$ où b est un désignateur rigide définit une situation dans laquelle $\mu(a) = b$ et inversément si a est un désignateur rigide. Si les deux sont des désignateurs rigides, l'équivalence est soit triviale (par exemple, $1 = 1$) soit fausse (par exemple, $1 = 2$).
- l'équivalence : $(a = b)$ où ni a ni b ne sont des désignateurs rigides est traduite en :

$$(executable(affect('a, b))$$

$$([affect('a, b)][B]a)$$

$$(executable(affect('b, a))$$

$$([affect('b,a)][B]b)$$

  Ces formules reflètent la sémantique sous-jacente à la déclaration que la dénotation de a est la dénotation de b. En effet, cette équivalence peut être utilisée pour déterminer a en déterminant b ou réciproquement. Si nous prenons l'action *affect('a,b)*, la précondition mentale nous oblige à connaître la dénotation de l'action et des arguments. La dénotation de $affect$ est donnée une fois pour toutes. L'expression $'a$ dénote l'expression elle-même. Il reste à déterminer la dénotation de b. A l'exécution, l'expression a dénote alors ce que dénote b. C'est exactement ce que nous voulons pour déterminer a à partir de b.

- l'implication : $(aand\ b \rightarrow c)$ est traduite en :

$$(a\ and\ b \leftrightarrow executable(affect('c, true))$$

$$([affect('c, true)][B]c)$$

$$([affect('c, true)][Believe]c)$$

  Nous pouvons remarquer que ces assertions spécifient que si a et b sont connus comme vrais par le système, alors l'action de déduire que c est vrai est exécutable. L'effet est alors une croyance sur c.

L'utilisateur peut donc introduire de nouvelles dénotations à l'aide du langage défini.

Quelques restrictions

Par la syntaxe présentée ci-dessus, nous avons exclu la plupart des expressions contenant l'opérateur épistémique $[B]$. En effet, nous voulons que l'utilisateur fournisse au système des connaissances sur le monde extérieur et non pas sur le système lui-même. Or, si nous prenons une expression de la forme:

$$[B]exp$$

cette information décrit la connaissance que le système possède de sa propre connaissance. Cette formule n'apporte rien de nouveau si $\mu(exp)$ est déjà défini. Elle peut même être contradictoire si $\mu(exp)$ n'est pas défini. Une telle expression est donc inutile puisqu'elle ne dit rien sur exp. Pour prendre un exemple de la vie courante, si nous déclarons à quelqu'un qu'il connaît le temps qu'il fait à New York, cela peut être vrai ou faux mais cette déclaration ne lui apporte aucune information sur le temps à New York.

Une information de la forme:

$$exp_1 \ and \ \neg[B]exp_2 \rightarrow exp_3$$

est encore plus dangereuse puisqu'en faisant semblant de décrire le monde extérieur, elle explicite en fait quelle hypothèse (ici exp_3) peut être faite en absence d'information ($\neg[B]exp_2$). Par exemple, nous pouvons avoir que:

$$pleut\text{-}sur(Paris) \ and \ pres(Paris, Vincennes) \ and \ \neg[B]\neg pleut\text{-}sur(Vincennes) \rightarrow pleut\text{-}sur(Vincennes)$$

Si le système obtient l'information que *pleut-sur(Vincennes)* est faux, la conclusion *pleut-sur(Vincennes)* doit être rétractée. Le système devient donc non-monotone ce qui pose des problèmes techniques non-négligeables. C'est pourquoi nous avons limité l'utilisation de l'opérateur $[B]$ à la seule description des effets des actions.

En effet, nous pouvons considérer qu'une expression de la forme:

$$exp_1 \rightarrow [act][B]exp_2$$

apporte une information objective sur l'effet d'une action sur les connaissances. Une telle formule redeviendrait gênante si nous cherchions à l'utiliser pour prédire le futur. En effet, elle affirme que nous obtiendrons une connaissance, mais elle ne dit pas laquelle. Dans notre prototytpe, nous n'utilisons cette information que pour savoir à quoi l'action peut servir. Elle ne nous pose donc aucun problème. C'est pour la même raison que nous avons également restreint l'utilisation des opérateurs de la logique dynamique.

Finalement, les opérateurs $[can]$ et $[desire]$ ne servent qu'au fonctionnement interne du système à l'exclusion des désirs initiaux que l'utilisateur est obligé de définir si nous voulons que le système fasse quelque chose.

5.2.3 Gestion des dénotations

La gestion des dénotations s'articule autour de trois fonctionalités :

- l'introduction de nouvelles dénotations,
- l'interrogation,
- la destruction.

L'introduction de nouvelles dénotations peut se faire de trois façons :

- par assertion de l'utilisateur à l'aide du langage décrit plus haut,
- par exécution d'une action perceptive (*ask*),
- par exécution d'une action déductive (*affect*).

Toute modification de la situation est signalée aux désirs puisque ceux-ci doivent s'adapter aux connaissances que le système possède à un moment donné. Les désirs doivent donc être avertis de toutes les modifications des propositions sur lesquelles ils portent. Nous verrons plus loin la gestion de ce mécanisme.

L'interrogation du module des dénotations ne consiste pas seulement en une recherche pure et simple dans l'ensemble des dénotations existantes à un moment donné. Le module essaie dans la mesure du possible de calculer sa réponse si la dénotation n'est pas connue. Ainsi si nous entrons les assertions suivantes :

$$a = 3$$

$$c = 4$$

$$b = 2$$

L'expression :

$$b = c - 2 and\ a < c$$

est calculable et vaut *vrai*. En effet, les fonctions $=$, $-$ et $<$ étant prédéfinies, le système est tout à fait capable d'établir la valeur de cette expression sans avoir à faire de déduction. Etant donné que l'utilisateur peut définir ses propres fonctions, il est donc possible de définir algorithmiquement les parties du système qui demandent des calculs compliqués ou qui tout simplement s'expriment mieux fonctionnellement. Les dénotations des expressions calculées sont à leur tour stockées. Ainsi, si l'utilisateur demande la dénotation de $c - 2$, elle n'a pas besoin d'être recalculée. Ce stockage a pour fonction essentielle de signaler aux désirs que l'état des connaissances du système a changé. Evidemment, il permet aussi de ne pas avoir à recalculer plusieurs fois les mêmes expressions.

La destruction d'une dénotation nécessite une mise à jour de la situation. Si nous reprenons l'exemple précédent et si nous détruisons la dénotation de l'expression c, alors les expressions $c - 2$, $b = c - 2$ et $b = c - 2 and\ a < c$ perdent également leur dénotation. De la même façon, toutes les informations qui

ont pu être déduites à partir de ces dénotations par l'exécution d'actes déductifs cessent d'être valides. Finalement, toutes les informations acquises par des actes perceptifs qui permettent de déduire cette information doivent être effacées. En effet, détruire une information qui peut être redéduite à partir d'autres informations n'a pas d'utilité. Le système de maintenance de cohérence fait donc une mise à jour complète de la situation. Pour ce faire, il a besoin de savoir pour chaque dénotation quelles sont les informations qui l'influencent et quelles sont les informations qui sont influencées par ces dernières. La gestion de la destruction peut trouver toutes ces informations dans l'historique qui connaît tous les événements qui se sont produits et leurs causes.

5.3 Gestion des désirs

Il existe deux types de liens entre les désirs :

- les dépendances hiérarchiques,
- les dépendances temporelles.

La hiérarchie des désirs est organisée en un graphe ET/OU. Les désirs sont des noeuds OU puisque chaque branche qui en est issue est une façon de résoudre le problème posé par le désir. Nous avons également des noeuds ET dont la résolution dépend de la résolution de chacune des sous-branches. Les noeuds ET relient les désirs les uns aux autres.

Les dépendances temporelles permettent de forcer l'ordre dans lequel les désirs sont résolus.

Plus précisément, chaque désir (noeud OU) est à l'origine de deux graphes :

- le graphe d'acquisition d'information,
- le graphe de résolution.

Le premier graphe est actif si la proposition sur laquelle porte le désir est inconnue. Le second est actif si la proposition sur laquelle porte le désir est fausse. Enfin les deux graphes sont inactifs, si la proposition est satisfaite.

Si le système désire que le cube a soit sur le cube b (*desire(sur(a b))*), nous aurons les deux sous-graphes fondés sur :

$$desire([B]sur(a\ b))$$

et

$$desire([can]poser(a\ b))$$

si nous supposons que l'action $poser(a\ b)$ permet de réaliser $sur(a\ b)$. Etant donné que le système sait toujours quand il sait ou ne sait pas, la formule $[B]sur(a\ b)$ ne peut être que vraie ou fausse. Un désir de cette forme ne possède donc jamais de sous-graphe d'acquisition d'information.

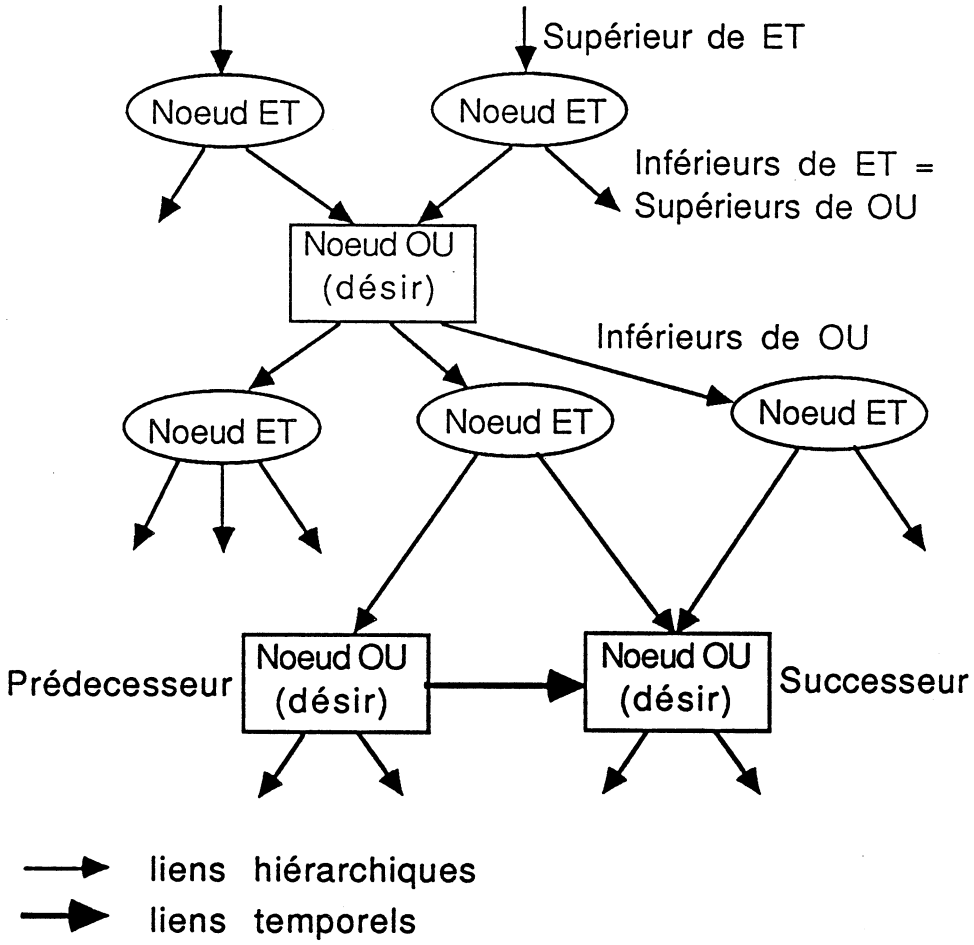

Figure 5.3: Liens entre les noeuds

Chaque noeud du graphe possède un état qui résume les influences des autres noeuds du graphe qui agissent sur lui. Nous allons expliquer en détail pour les noeuds OU et les noeuds ET comment leur état est calculé. Pour cela nous avons besoin d'introduire une terminologie pour désigner les différentes relations qui existent entre les noeuds du graphe (voir figure 5.3). Nous appelons :

supérieurs : les noeuds ET dont un noeud OU dépend hiérarchiquement ou le noeud OU dont un noeud ET dépend hiérarchiquement.

inférieurs : les noeuds ET qui dépendent hiérarchiquement d'un noeud OU ou les noeuds OU qui dépendent hiérarchiquement d'un noeud ET

prédécesseurs : les noeuds OU qui précèdent un noeud OU

successeurs : les noeuds OU qui suivent à un noeud OU

Remarquons que nous avons toujours un noeud ET entre deux noeuds OU dans les liens de dépendances. D'autre part, les dépendances temporelles n'existent

qu'entre des noeuds OU.

5.3.1 Etats d'un noeud OU

Nous avons vu que les sous-graphes sont actifs ou inactifs selon que certains désirs portent sur des propositions vraies, fausses ou inconnues. Nous allons tenir compte de ce phénomène en associant à chaque noeud du graphe une étiquette. Pour les noeuds OU, cette étiquette dépend :

- de ses supérieurs,
- de ses prédécesseurs,
- de la dénotation de la proposition sur laquelle il porte,
- de ses inférieurs.

Nous avons défini les états possibles d'un noeud OU afin de séparer soigneusement les influences multiples qui s'exercent sur lui.

Les influences de ses supérieurs et de ses prédécesseurs déterminent la prise en considération du noeud. Le noeud est :

dormant : si tous ses supérieurs sont inactifs ou si au moins un prédécesseur n'est pas satisfait.

actif : si il n'est pas dormant (notamment s'il n'a aucun supérieur).

Le noeud OU n'est pas pris en considération (dormant) si il ne sert à rien ou si il doit attendre que d'autres désirs soient réalisés avant lui.

La proposition sur laquelle le noeud OU porte, détermine lequel des sous-graphes est pertinent. Si le noeud est actif, le noeud est alors :

satisfait : si la proposition est vraie. Dans ce cas, les deux sous-graphes doivent devenir dormants.

attente-acquisition : si la proposition est inconnue. Le graphe de résolution doit alors être dormant et le graphe d'acquisition doit être actif.

attente-résolution : si la proposition est fausse. Le graphe d'acquisition doit être dormant et le graphe de résolution doit être actif.

La détermination de cet état du noeud OU influence l'état des sous-graphes correspondants ainsi que des successeurs. Cette influence doit donc être propagée vers le bas.

Finalement, l'influence des sous-graphes doit être prise en compte pour déterminer complètement l'état d'un noeud OU. Cet état est :

échoué : si il est en attente d'acquisition et cette acquisition a échoué ou si il est en attente de résolution et toutes les alternatives de résolution ont échoué.

actif : dans les conditions suivantes :

- il est en attente-résolution et au moins une des alternatives de résolution a réussi,
- il est en attente-résolution et il n'y a pas d'alternative existante,
- il est en attente-acquition et il n'y a pas de sous-graphe d'acquisition.

L'échec détermine les désirs pour lesquels il n'y a pas de solution tandis que l'état actif détermine les feuilles du graphe de recherche de solution. Les noeuds actifs soit sont des noeuds à résoudre lorsqu'il n'y a pas encore d'alternatives, soit déterminent des actions exécutables.

La détermination de cet état final peut modifier l'état des noeuds supérieurs et nécessite donc une propagation vers le haut.

Pour résumer, nous avons donc six états possibles pour un noeud OU :

- dormant,
- satisfait,
- échoué,
- attente-acquisition,
- attente-résolution,
- actif.

5.3.2 Etats d'un noeud ET

Les noeuds ET relient les noeuds OU entre eux et leur confèrent la structure hiérarchique. Pour faciliter le calcul des états des noeuds du graphe, ils sont aussi munis d'un état qui résume les influences sur cette partie du graphe. Leur état dépend seulement de deux influences :

- l'influence du noeud supérieur,
- l'influence des noeuds inférieurs.

De nouveau, les états permettent de distinguer les influences exercées. Ainsi, le noeud ET est dans l'état :

dormant : dans les cas suivants :

- son supérieur est dormant,
- il est dans le sous-graphe d'acquisition d'information et le supérieur est en attente-résolution,
- il est dans le sous-graphe de résolution et le supérieur est en attente-acquisition.

actif : si il n'est pas dormant.

Ces états peuvent modifier le statut des noeuds OU inférieurs. Ils nécessitent donc une propagation vers le bas.

Après propagation vers le bas et si il est déjà actif, le noeud ET peut passer dans les états suivants selon le statut des noeuds inférieurs :

satisfait : si tous ses inférieurs sont satisfaits.

échoué : si au moins un de ses inférieurs a échoué.

Il reste dans son état dormant ou actif dans tous les autres cas. Finalement, toute modification de son statut nécessite une propagation vers le haut afin de signaler l'échec ou la satisfaction du noeud.

5.3.3 Interactions avec les autres structures

Les modifications du graphe des désirs influencent les deux autres structures :

- les dénotations,
- les intentions.

Comme nous l'avons vu dans la section précédente, toute interrogation du module des dénotations peut provoquer la détermination d'un ensemble de sous-expressions. Ces expressions peuvent être des conditions sur lesquelles portent des désirs. La modification de la dénotation de ces expressions est donc signalée aux désirs correspondants. Des modifications dans l'étiquetage du graphe des désirs peuvent alors surgir. Le principe suivant permet d'assurer un comportement correct de ces diverses propagations :

> le graphe des désirs ne peut se mettre à jour que lorsque les dénotations sont complètement à jour.

Cette condition assure que le calcul des étiquettes du graphe se base sur une situation cohérente. Elle suppose que tous les avis de modification engendrés par le module de dénotation pour le graphe des désirs sont stockés dans une queue d'attente pour n'être envoyés qu'après mise à jour de la situation.

En ce qui concerne les intentions, les sous-graphes de résolution ont comme racine le désir d'être capable de faire une action. Lorsqu'un tel désir est satisfait, l'action correspondante peut être exécutée par le système puisqu'elle est à la fois parfaitement déterminée et exécutable dans les circonstances présentes. Cette condition est donc signalée à la composante décisionnelle pour qu'elle puisse choisir l'action à faire et qu'elle puisse l'exécuter.

5.4 Module d'exécution

Le module d'exécution tient à jour l'ensemble des actions que le système est capable de faire à un moment donné. Une action est :

disponible : si le système est capable de faire cette action et si elle sert à quelque chose (le désir $[can]action$ est dans l'état satisfait).

indisponible : si le système est incapable de faire cette action (le désir $[can]action$ n'est pas satisfait) ou si cette action ne sert à rien (le désir $[can]action$ est dormant).

exécutée : si l'action a été exécutée avec succès.

échouée : si l'action a été exécutée sans succès.

Les deux premières conditions sont signalées par le graphe des désirs. Les deux dernières permettent au système de ne pas choisir deux fois la même action. Une action échoue si l'utilisateur est incapable de l'exécuter (par exemple, si une machine est en panne). Cependant l'action peut très bien être exécutée avec succès mais ne pas avoir l'effet souhaité.

Une action disponible est choisie de façon arbitraire puis exécutée. Si son exécution s'est réalisée sans problème, le module calcule les effets de l'action sur ses connaissances. Il détruit la partie de son modèle du monde extérieur modifié par l'action. Ces modifications sont signalées au graphe des désirs par le module de dénotation. L'effet principal est la réactivation des sous-graphes d'acquisition d'information afin de reconstruire le modèle sur le nouvel état du monde extérieur.

5.5 L'interpréteur

L'interpréteur a essentiellement deux fonctions :

- la construction du graphe des désirs,
- l'appel du module d'exécution.

5.5.1 Construction du graphe

La construction du graphe des désirs consiste dans le choix d'un noeud actif et de sa résolution de la façon suivante :

- si le noeud $[desire]p$ est en attente d'acquisition d'information, il faut construire le sous-désir $[desire][B]p$ de savoir quelle est la valeur de vérité de la proposition.
- si le noeud $[desire]p$ est en attente de résolution, il faut créer le sous-désir $[desire][can]action$ pour toutes les actions telles que $[action]p$. Trois conditions demandent un traitement particulier :

- une condition de la forme $[B]f(t_1, \ldots, t_n)$ peut être résolue si les conditions $[B]f$ et $[B]t_1$ à $[B]t_n$ sont résolues,
- une condition de la forme $[can]action$ est résolue si les conditions $[B]action$ et $executable(action)$ sont résolues,
- une condition de la forme $executable(action)$ est résolue si le système peut rendre vraie la condition qui rend $executable(action)$ fausse (p tel que $[can]affect('executable(action)\ p)$).

Il faut noter qu'après un certain temps de fonctionnement, le graphe est entièrement construit et cette phase de l'interpréteur n'aura plus rien à faire. Nous pouvons assimiler cette phase à une *compilation incrémentale* des connaissances sous forme d'un graphe.

5.5.2 Appel du module d'exécution

L'appel du module d'exécution teste si une action est disponible afin de l'exécuter. Lorsque le graphe est entièrement construit, c'est la seule phase qui subsiste afin d'acquérir des informations sur le monde extérieur et d'agir en conséquence.

5.5.3 Conditions d'arrêt

Si il ne reste plus de noeud à résoudre et qu'aucune action n'est plus disponible, le système suppose que l'état du monde extérieur est devenu stable et il s'arrête. En effet, étant donné que le système, par hypothèse, est le seul à agir sur le monde extérieur, il s'arrête dès que l'ensemble de ses désirs est satisfait.

Pour prendre en compte dans une certaine mesure la possibilité que le monde puisse changer indépendamment du système, nous avons ajouté à cette implantation la notion de *persistance* d'un fait. Nous pouvons déclarer que la dénotation d'une expression n'est valide que sur une certaine période de temps. Par défaut cette période est infinie. Au bout de cette période, le système efface la partie du modèle qui dépend de ce fait ce qui réactive le graphe des désirs afin de remettre à jour son modèle du monde extérieur. Ce mécanisme est implanté par un échéancier dans lequel le système planifie la destruction des faits à durée de validité finie. L'interpréteur ne s'arrête alors que lorsqu'il n'y a plus d'événements en attente dans l'échéancier.

5.6 Historique

L'historique est une représentation de l'évolution du système. Il enregistre tous les événements qui se produisent avec leurs causes et leurs effets. Ces événements sont soit des actions externes, soit des actes perceptifs ou déductifs. Pour structurer ces informations, l'historique implante les notions suivantes :

- les *faits* sont des énoncés qui peuvent être connus ou inconnus sur des intervalles de temps. Ces intervalles sont contigus puisqu'un fait cesse d'être inconnu au moment où il devient connu et inversément.
- les *événements* désignent l'ensemble des occurrences possibles d'une action.
- les *instants* représentent les occurrences datées des événements.
- les *intervalles* représentent la connaissance ou la non-connaissance d'un fait entre deux dates.

En ce qui concerne les intervalles, une des bornes peut être indéterminée. La borne inférieure est indéterminée si le fait est inconnu depuis toujours jusqu'à une certaine date. La borne supérieure est indéterminée si le fait est connu ou inconnu depuis une certaine date jusqu'à maintenant. Il est impossible qu'un fait soit connu depuis toujours puisque les informations les plus anciennes datent au moins de l'initialisation du système. Un fait qui a été et est toujours inconnu n'a jamais été pris en considération par le système et ne peut donc avoir donné lieu à aucune représentation. Donc, une des bornes d'un intervalle est toujours définie.

L'échelle des dates dépend de la plus petite unité de temps représentable dans l'implantation. Sur notre implantation, elle est exprimée en année, mois, jour, heure, minute, seconde et soixantième de seconde.

L'historique enregistre les dépendances entre ces différentes notions. Ainsi, un événement est *permis* (enablement) par un ou plusieurs intervalles. La cessation ou l'apparition d'un intervalle est *causé* par un événement. Ces dépendances sont créées et exploitées par le module de dénotation. Ce dernier doit toujours signaler avec l'apparition et la cessation quel événement en est à l'origine et qu'est-ce qui a rendu possible cet événement. Les dépendances sont utilisées pour mettre à jour le modèle du monde extérieur.

5.7 Exemple

5.7.1 Les connaissances

Pour illustrer le fonctionnement du système et montrer qu'il a bien le comportement souhaité, nous allons reprendre l'exemple du robot jardinier. Nous supposons dans cet exemple, que les symboles *mouille, sec, beau, pluvieux, dedans, dehors, ouvert* et *ferme* sont des désignateurs rigides.

Exprimons d'abord la raison de vivre de notre robot :

$$desire(etat(gazon) = mouille)$$

Cette assertion crée immédiatement un premier désir qui est dans l'état actif puisque dans l'état initial, le système ne possède aucune connaissance sur le monde extérieur.

Nous donnons également à notre système quelques connaissances sur les relations entre le temps et l'état du gazon :

$$etat(temps) = pluvieux \rightarrow etat(gazon) = mouille$$

$$etat(temps) = beau \rightarrow etat(gazon) = sec$$

Ces assertions sont transformées en description d'actes déductifs. Elles sont donc équivalentes aux assertions :

$$etat(temps) = pluvieux \leftrightarrow executable(affect('etat(gazon)\ mouille))$$

$$etat(temps) = beau \leftrightarrow executable(affect('etat(gazon)\ sec))$$

avec les effets :

$$[affect('etat(gazon)\ mouille)][B]etat(gazon)$$

$$[affect('etat(gazon)\ sec)][B]etat(gazon)$$

Les capacités perceptives du système sont décrites par trois actions qui permettent respectivement de regarder l'état du temps, l'état des volets et la position actuelle du robot :

$$etat(volets) = ouvert \leftrightarrow executable(ask\ 'etat(temps))$$

$$[(ask\ 'etat(temps))][B]etat(temps)$$

$$position(moi) = dedans \leftrightarrow executable(ask\ 'etat(volets))$$

$$[(ask\ 'etat(volets))][B]etat(volets)$$

$$executable(ask\ 'position(moi))$$

$$[(ask\ 'position(moi))][B]position(moi)$$

Remarquons qu'à part l'observation de la position du robot, les actes perceptifs sont tous munis de préconditions. Il n'est en effet pas toujours possible de capter n'importe quoi dans n'importe quelle condition. La prise en compte de cette propriété importante n'a jamais été possible dans aucun générateur de plans d'actions mais s'exprime naturellement dans notre système.

Finalement, les capacités d'action du système sont les suivantes :

$$position(moi) = dehors \leftrightarrow executable(do\ 'arroser(gazon))$$

$$[(do\ 'arroser(gazon))]etat(gazon) = mouille$$

$$position(moi) = dedans \leftrightarrow executable(do\ 'ouvrir(volets))$$

$$[(do\ 'ouvrir(volets))]etat(volets) = ouvert$$

$$position(moi) = dedans \leftrightarrow executable(do\ 'sortir)$$

$$[(do\ 'sortir)]position(moi) = dehors$$

Notre robot est donc capable de sortir, d'ouvrir les volets et d'arroser le gazon.

5.7.2 Le fonctionnement

Dans une première phase, l'interpréteur va construire le graphe d'acquisition de l'information sur l'état du gazon étant donné que cet état est inconnu dans l'état initial. Cet état peut être déduit de l'état du temps. Le temps peut être observé par la fenêtre à condition que les volets soient ouverts. Enfin, le système peut savoir si les volets sont ouverts si il se trouve à l'intérieur de la pièce. Nous obtenons le graphe de la figure 5.4. Le système est toujours capable de se situer d'après les descriptions que nous avons données dans le paragraphe précédent. C'est pourquoi, le système peut savoir immédiatement qu'il est capable de demander où il se trouve ($[can](ask\ 'position(moi))$).

Etant donné que le système ne peut connaître l'état des volets ($[can](ask\ 'etat(volets)$) que si et seulement si il se trouve à l'intérieur, il déduit de sa position que l'observation de l'état des volets est exécutable. L'action suivante est donc de demander si les volets sont ouverts ou fermés. Les volets étant fermés, le système est incapable de regarder si le temps est beau ou pluvieux. Dans la figure 5.5, cette situation se traduit par le fait que $executable(ask\ 'etat(temps))$ est faux.

L'action de demander l'état du temps n'est pas exécutable parce que les volets ne sont pas ouverts. Donc pour résoudre ce problème, il faut rendre vraie la proposition $etat(volets) = ouvert$. Nous savons déjà que les volets sont fermés ainsi qu'il est indiqué dans la figure 5.6 par un rectangle arrondi autour de la proposition $[B]etat(volets)$. Le système va donc chercher à ouvrir les volets puisque c'est la seule action qui permette de changer l'état des volets. Cette action est exécutable à condition d'être à l'intérieur de la pièce. Or nous savons déjà que nous sommes à l'intérieur. Donc, l'action de déduire que l'action est exécutable est elle-même exécutable. Le système est donc capable d'ouvrir les volets résolvant si tout se passe bien le problème qui empêchait le système de regarder par la fenêtre. Nous supposons qu'il n'y a pas de coup de vent malencontreux qui ferme les volets entre le moment où l'action est exécutée et le moment où le système vérifie le nouvel état des volets. Si cela se produisait, le système replanifierait l'ouverture des volets.

Finalement, le système est capable de regarder quel temps il fait. Le temps s'avère beau. Le système peut donc en déduire que le gazon est sec. Nous nous trouvons alors dans la situation décrite par la figure 5.7. Le gazon n'est pas mouillé ce qui pose donc un nouveau problème à notre robot jardinier.

L'action qui permet de mouiller le gazon est l'arrosage. Le système va donc raisonner sur la possibilité d'arroser le gazon. Mais cette action n'est exécutable qui si le robot est dehors. Or le système sait déjà qu'il se trouve dedans. Il se pose donc un nouveau problème, à savoir de rendre exécutable l'action d'arroser.

Pour résoudre ce problème, le système doit se trouver dehors ce qui est réalisable par l'action de sortir. Cette action n'est exécutable que si le robot se trouve dedans ce que le robot sait déjà (voir figure 5.9). Le système en déduit donc qu'il est capable de sortir.

L'effet de sortir est d'effacer les informations concernant la position du robot

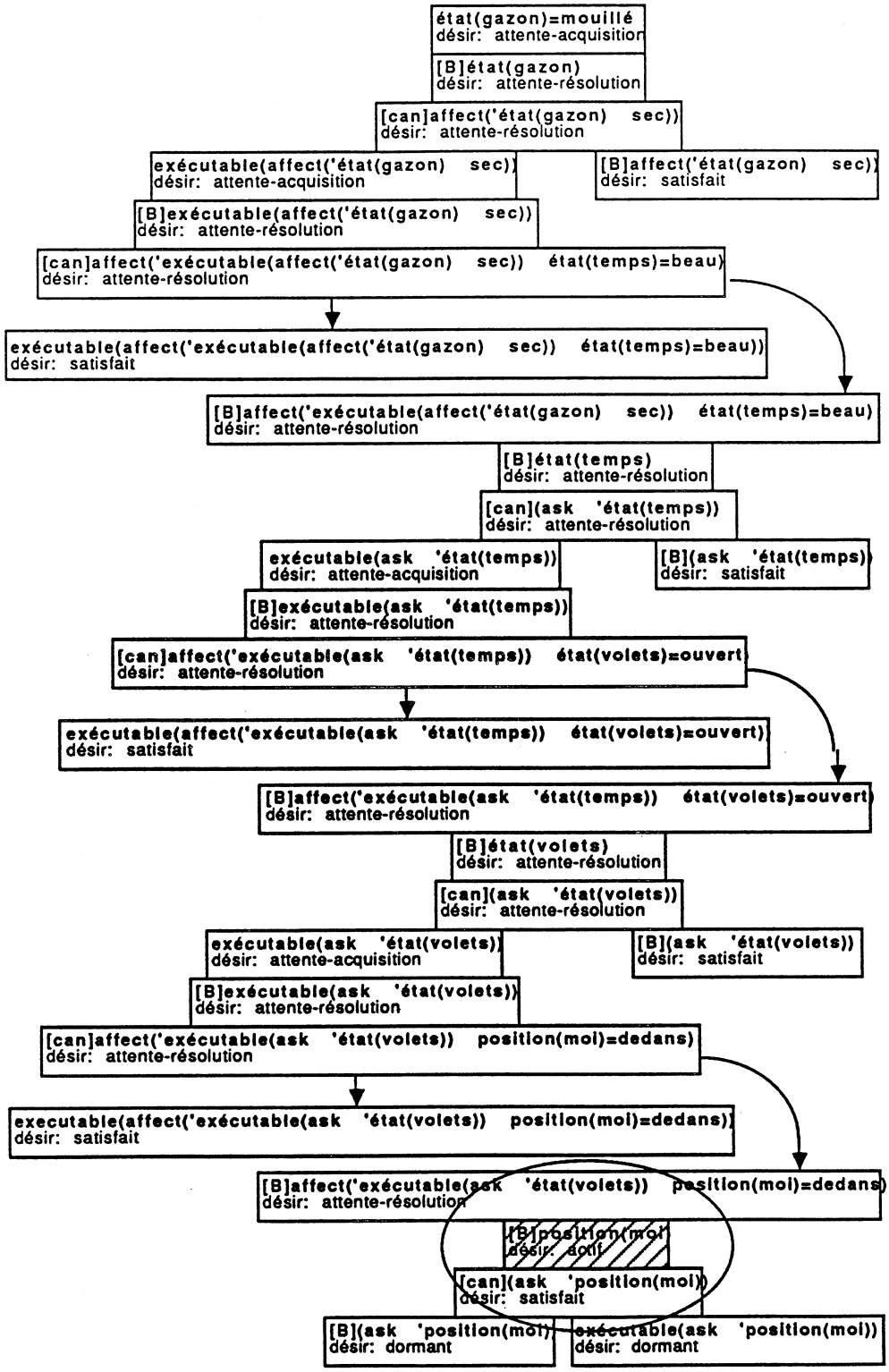

Figure 5.4: Arborescence d'acquisition d'information

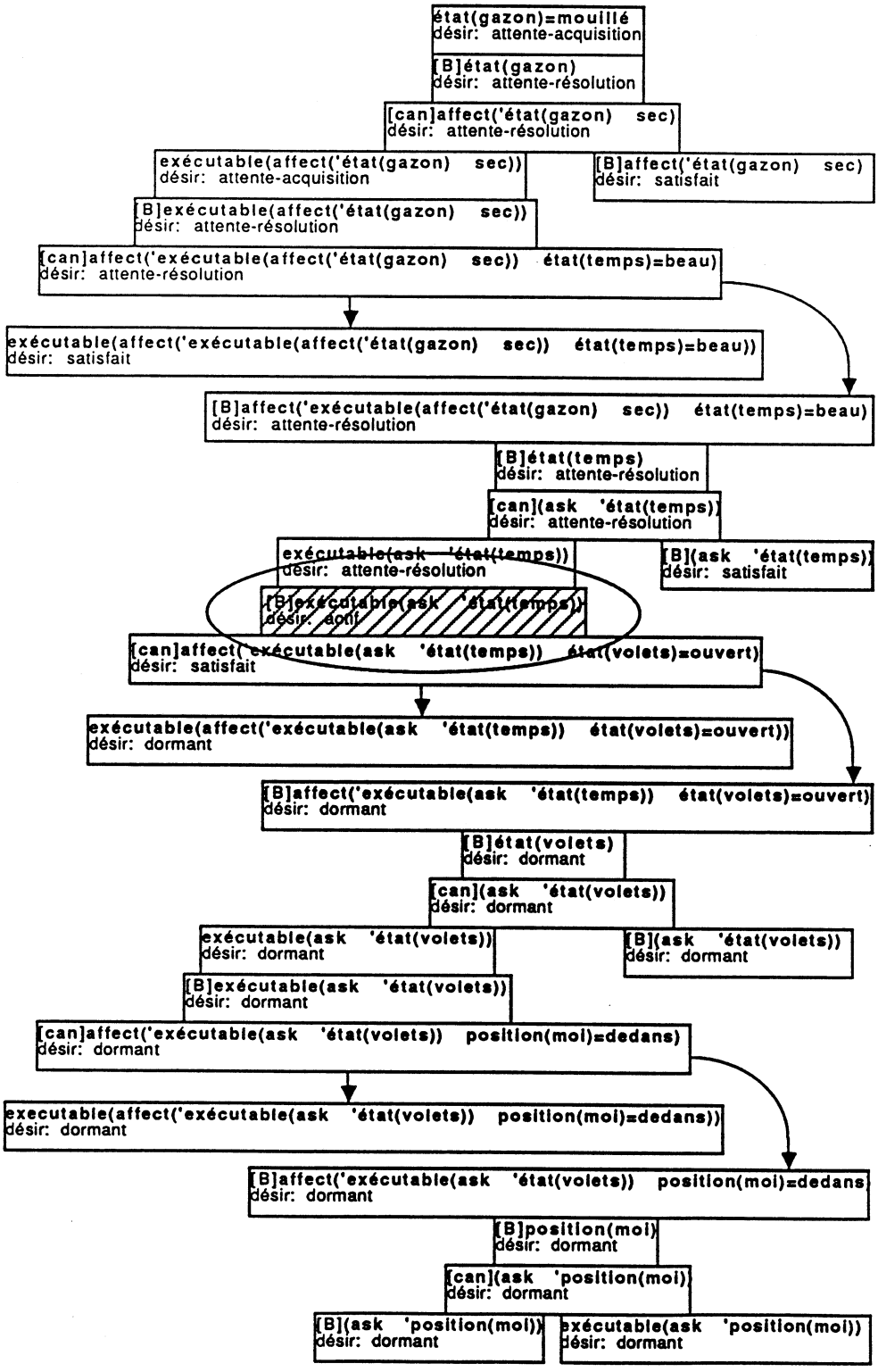

Figure 5.5: Exécutabilité du pas d'inférence

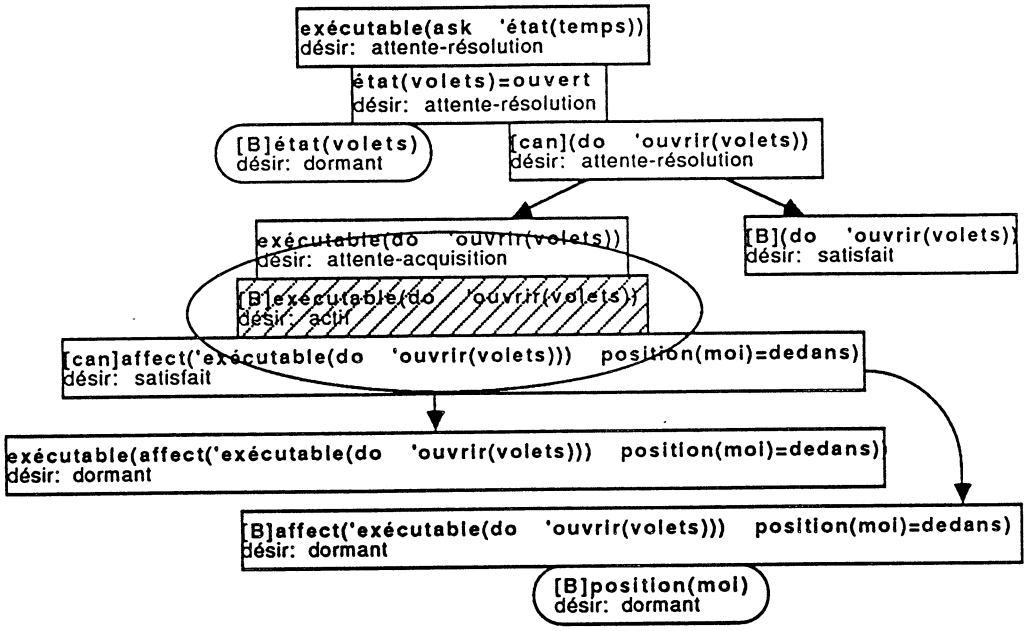

Figure 5.6: Planification de l'ouverture des volets

puisque cette position a changé. Mais la position a pu ne pas changer dans le sens souhaité. Le modèle concernant la position du robot est donc effacé ce qui active le graphe d'acquisition d'information correspondant. Si l'action a réussi dans le sens que le robot se trouve effectivement dehors, le système est alors en mesure d'arroser le gazon ce qui changera l'état du gazon. Ce changement doit être confirmé par l'acquisition d'information sur l'état du gazon. Si le gazon est bien mouillé, le système s'arrête à moins que nous lui disions que le gazon ne reste pas éternellement mouillé.

5.8 Discussion

Nous articulons la discussion de notre prototype autour des points suivants :

- l'expressivité du modèle,
- l'adaptabilité du système au monde extérieur.

5.8.1 Expressivité

L'expressivité découle du modèle théorique que nous avons introduit. Ainsi nous sommes capables de décrire des actions qui modifient soit le monde extérieur soit les connaissances que le système possède du monde extérieur ce qui permet :

- de réaliser une condition pour pouvoir

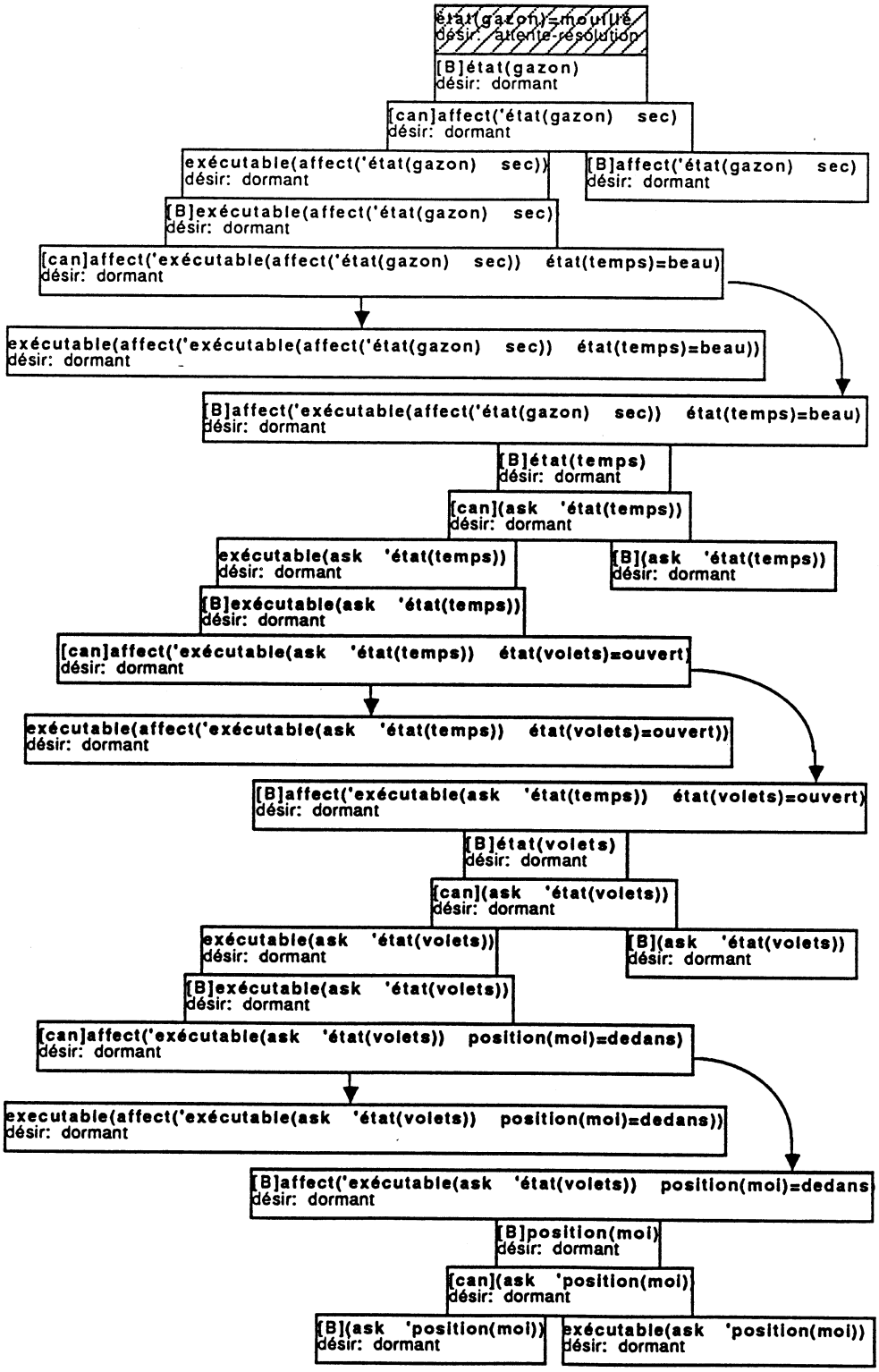

Figure 5.7: Fin de l'acquisition d'information

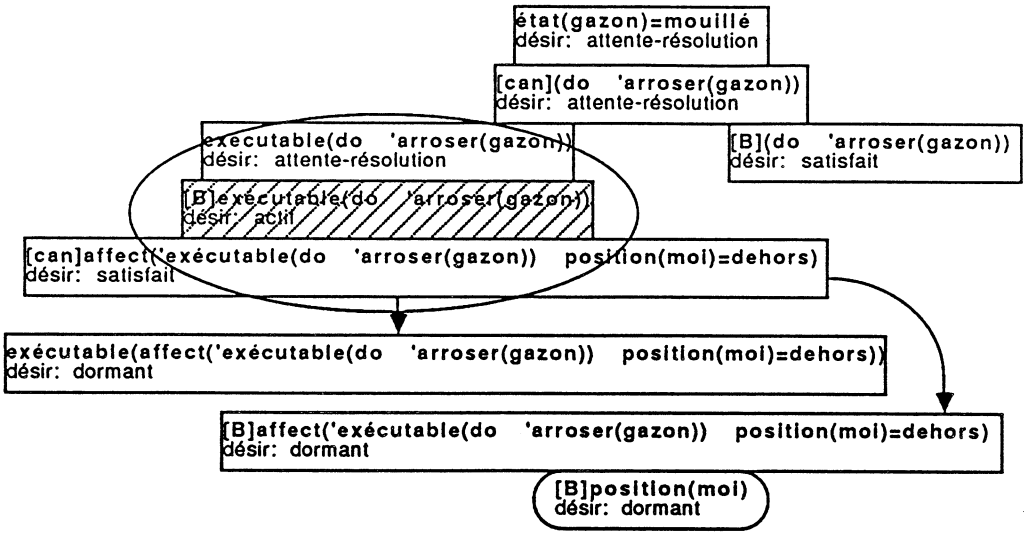

Figure 5.8: Exécutabilité de l'arrosage

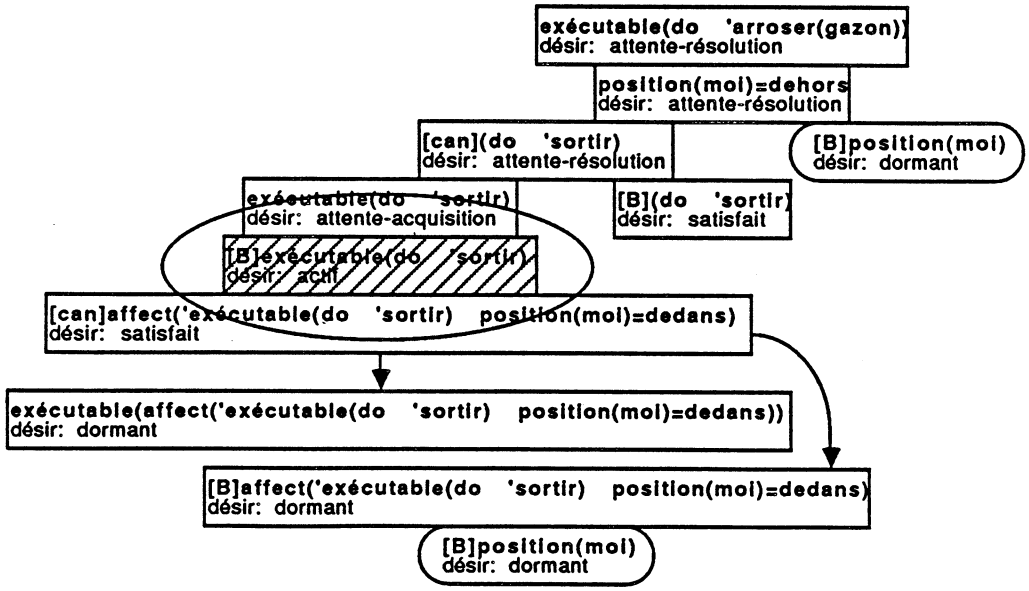

Figure 5.9: Exécutabilité de la sortie

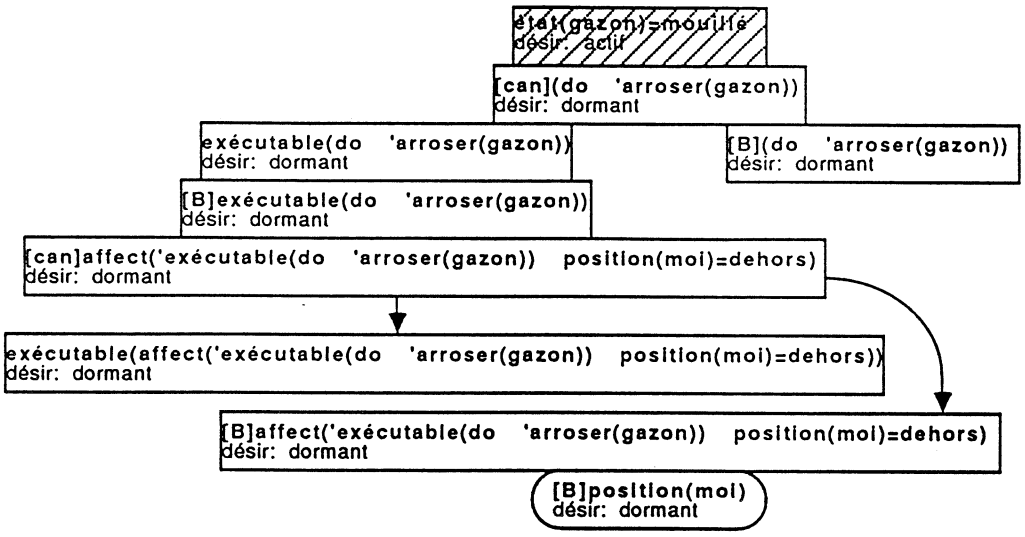

Figure 5.10: Arrosage du gazon

  - déduire,
  - percevoir,
  - rendre une action exécutable,
  - permettre à une action de produire l'effet voulu.

- d'acquérir de l'information pour savoir
  - comment faire une action,
  - si une action est exécutable,
  - si une action a bien l'effet voulu.

Nous profitons donc d'une part de l'intégration complète des actions perceptives, déductives et réelles. D'autre part, nous exploitons la distinction que nous faisons entre les conditions d'exécutabilité, les conditions épistémiques qui permettent de savoir comment faire une action et les conditions sous lesquelles une action produit un certain effet.

5.8.2 Adaptabilité

Etant donné que nous avons supposé que le système était seul à pouvoir changer l'état du monde extérieur, CHAOS ne peut tenir compte que des échecs de l'exécution de ses actions. En effet, l'effacement de la partie du modèle qui concerne ce qui a été modifié par l'action provoque une réacquisition de l'information. Si les désirs à l'origine de l'action ne sont toujours pas satisfaits, les actions correspondantes peuvent être réexécutées jusqu'à ce qu'elles réussissent. Si d'autre part

les conditions d'exécutabilité ont changé, le système essaie soit de les établir de nouveau, soit de faire d'autres actions.

Nous pouvons regretter deux choses :

- que le système ne s'aperçoive pas des modifications qui ne concernent pas directement l'action qui vient d'être faite,
- que le système ne puisse pas réagir à des modifications de l'univers qui se feraient indépendamment de lui-même.

La première critique est liée au problème du raisonnement sur les actions. En général nous pouvons faire l'hypothèse qu'une action n'a qu'un effet localisé sur son environnement. Nous avons défini la frontière de ce qui change comme étant l'ensemble des faits liés aux effets connus de l'action. Nous avons déjà dit qu'une position extrême est de considérer que les choses ont changé comme prévu et que l'autre extrême est de considérer que tout peut avoir changé. La bonne solution dépend essentiellement du temps que le système peut consacrer d'une part au diagnostic de ce qui s'est produit, d'autre part à l'effort qu'il veut ou qu'il a le temps d'investir. Nous pouvons faire le classement suivant par rapport au but et à l'effort à investir :

1. vérifier si l'objectif est atteint.
2. vérifier si tous les effets connus de l'action sont atteints.
3. diagnostiquer tout ou partie des anomalies pour vérifier les effets non-prévus de l'action.

Nous nous sommes placés dans le second cas.

En ce qui concerne la prise en compte des modifications indépendantes du système, deux réponses sont possibles. La première réponse est de dire que même si notre prototype ne le fait pas, il est *en principe* capable de le faire. Dans notre modélisation nous avons exclu la perception ascendante. Cependant, le mécanisme qui met à jour les croyances du système et le graphe des désirs lorsqu'une action est exécutée est parfaitement extensible au cas de la perception d'un événement extérieur. Nous pouvons facilement imaginer qu'un capteur perçoive une modification d'un fait et efface la dénotation correspondante ce qui aurait pour effet de mettre à jour la situation, d'activer le sous-graphe d'acquisition d'information correspondant et donc de tenir compte du nouvel état du monde extérieur. Si nous voulons conserver une perception uniquement descendante, la seconde réponse est l'introduction du raisonnement sur le futur. Si le système était capable de raisonner sur ce qui va se passer notamment sur ses propres actions, il pourrait aussi prévoir l'occurrence d'événements extérieurs et donc planifier la perception des modifications qui en découlent.

Chapitre 6

Conclusion

L'autonomie est un passage obligé dans la quête de l'intelligence artificielle[1]. Notre contribution dans cette voie est d'avoir :

- proposé une modélisation des connaissances qui permette de tenir compte de l'interprétation des structures symboliques et des limites inhérentes à un agent réel,
- montré comment les interactions entre les croyances et les actions peuvent donner lieu à une formulation logique qui soutend la génération de plans d'actions,
- proposé une notion de désir qui permette de mettre le processus de génération de plans d'actions en relation avec les modifications des connaissances qu'un système possède du monde extérieur,

et d'avoir montré la faisabilité de cette approche par la réalisation du système CHAOS.

Nous pouvons tirer de cette étude un certain nombre de suggestions sur les conditions de l'autonomie d'un système artificiel.

Une première condition est d'avoir dans un système des représentations du monde extérieur. Or, des structures symboliques ne peuvent représenter que si le système est capable de les lier à son univers extérieur par des capteurs et des effecteurs. En effet, une structure ne représente que si elle est constructible à partir de données-capteur, si elle permet d'agir sur le monde extérieur et enfin si elle contribue à la satisfaction des buts du système. Ceci fait de tous les domaines en contact avec le monde réel tel que la Robotique ou la Conduite de Procédé, des champs d'expérimentation privilégiés pour l'Intelligence Artificielle.

Une deuxième condition est d'être capable de coordonner efficacement exécution, perception et prise de décision. Dans ce cadre, nous nous sommes basés sur l'architecture croyance-désir-intention proposée récemment par divers chercheurs

[1]nous écrivons ce terme avec des minuscules car il ne s'agit pas ici du nom du domaine mais de son objectif.

en Intelligence Artificielle. Cette architecture possède l'avantage de fournir une structure globale dans laquelle s'intègrent génération de plans d'actions et processus de décision. Nous nous sommes attachés à exploiter les liens croyance-action et croyance-désir afin d'assurer la coordination perception-exécution et nous en avons montré à la fois les avantages et les limites.

Cependant, beaucoup de travail reste à faire et notamment :

- l'intégration du processus de décision par la modélisation des liens entre les désirs et les intentions et la prise en compte du contexte dans ce processus,
- le raisonnement sur le temps dont les interactions avec le raisonnement sur les actions et sur les croyances sont complexes.

Enfin, relevons que le modèle de production d'un comportement autonome que nous proposons peut être utilisé dans le raisonnement multi-agent. En effet, il permet, sur la base des croyances et des désirs d'un agent, d'inférer son comportement. Il peut donc être utilisé pour raisonner sur le comportement d'autres agents afin d'assurer une coordination. En fournissant un modèle de l'influence de l'environnement sur le processus de génération du comportement, il permet aussi à un agent d'influencer le comportement des autres. Il peut donc servir de base à diverses formes de coopération.

Bibliographie

[1] J.F.Allen, C.R.Perrault.
Analysing Intention in Utterances.
Artificial Intelligence - 15(1980) - pp. 143-178

[2] J.F.Allen.
Towards a General Theory of Action and Time.
Artificial Intelligence - 23(1984) - pp.123-154

[3] M.Bratman.
Taking Plans Seriously.
Social Theory and Practice, Vol. 9, No 2-3, (Summer-Fall 1983)

[4] M.Bratman.
Two Faces of Intention.
The Philosophical Review, XCIII, No 3 (July 1984)

[5] M.Bratman, K.Konolige, D.J.Israel & M.E.Pollack
Rational Behaviour in Resource-Bounded Agents.
SRI - NSF Project Proposal - 1986

[6] R.A.Brooks
Achieving Artificial Intelligence Through Building Robots
MIT - AIM 899 - May 1986

[7] E.Charniak, D.McDermott.
Introduction to Artificial Intelligence.
Addison Wesley - 1985

[8] P.R.Cohen, H.J.Levesque.
Persistence, Intention and Commitment.
SRI - Draft - Sept. 1986

[9] Y.Descottes.
Représentation et Exploitation de Connaissances Expertes en Génération de Plans d'Actions.
INPG - Thèse - Décembre 1981

[10] J.Doyle.
A Model for Deliberation, Action and Introspection.
MIT-AI-TR-581, May 1980

[11] R.Fagin, J.Halpern.
Belief,Awareness and Limited Reasoning: Preliminary Report.
IJCAI-85 - Los Angeles - pp.480-490

[12] A.I.Goldman.
A Theory of Human Action
Prentice Hall, Englewood Cliffs, N.J., 1970

[13] J.Y.Halpern,Y.Moses.
A Guide to the Modal Logics of Knowledge and Belief: Preliminary Draft.
IJCAI-85 - Los Angeles - pp.480-490

[14] A.R.Haas.
A Syntactic Theory of Belief and Action.
Artificial Intelligence - 28(1986) - pp.245-292

[15] P.J.Hayes.
The Naive Physics Manifesto.
1979

[16] J.Hintikka.
Knowledge and Belief.
Ithica, New York: Cornell University Press

[17] K.Konolige.
A Deduction Model of Belief and its Logics.
SRI - T.N. 326 - Aug. 1984

[18] K.Konolige.
Experimental Robot Psychology.
SRI - T.N. 363, 1985

[19] S.A.Kripke.
Semantical Analysis of Modal Logic.
Zeitschrift für mathematische Logik und Grundlagen der Mathematik, Bd 9 s.67-96 (1963)

[20] T.H.Kuhn
La structure des révolutions scientifiques
Flammarion - 1983

[21] H.J.Levesque.
Foundations of a Functional Approach to Knowledge Representation.
Artificial Intelligence - 23(1984) - pp.155-212

[22] L.Linsky and Co.
Reference and Modality.
Ed. L.Linsky.
Oxford University Press - 1971

[23] A.Lux.
Algorithmique et Contrôle en Vision par Ordinateur.
INPG - Thèse - 1985

[24] J.McCarthy.
First Order Theories of Individual Concepts and Propositions.
Machine Intelligence 9, Ellis Horwood, 1979

[25] J.McCarthy. ?

[26] J.McCarthy, P.Hayes.
Some Philosophical Problems from the StandPoint of Artificial Intelligence.
in Readings in Artificial Intelligence - Tioga 1981

[27] D. McDermott & J. Doyle.
Nonmonotonic Logic I.
Artificial Intelligence - ?(1980)

[28] D. McDermott.
A Temporal Logic for Reasoning about Processes and Plans
Yale University - Research Report no 196 - March 1981

[29] D. McDermott.
Nonmonotonic Logic II: Nonmonotonic Modal Theories.
JACM - Vol.29, no 1, Jan. 1982, pp.33-57

[30] R.C.Moore.
Reasoning about Knowledge and Action.
SRI - T.N. 191 - Oct. 1980

[31] R.C.Moore.
Semantical Considerations on Nonmonotonic Logic.
IJCAI-83 - Karlsruhe - pp.272-279

[32] J.P.Müller.
Etude des logiques modales pour la modélisation du raisonnement d'un agent intelligent en univers réel.
LIFIA - Rapport de Recherche n.47, IMAG - Rapport de Recherche n.603, Mars 1986

[33] A.Newell.
The Knowledge Level.
Artificial Intelligence - 18(1982) - pp.87-127

[34] N.Nilsson.
Principles of Artficial Intelligence.
Tioga 1980

[35] N.Nilsson.
Intelligent Communicating Agents.
Stanford University - Project Proposal - 1986

[36] M.E.Pollack.
Infering Domain Plans in Question-Answering.
University of Pennsylvania - Thesis - 1986

[37] V.R.Pratt.
Six Lectures in Dynamic Logic.
MIT/LCS/TM-117 - Dec. 1979

[38] S.S.Rosenschein.
Plan Synthesis: A Logical Perspective.
Proc. IJCAI-81 - Vancouver - pp.331-337

[39] S.S.Rosenschein.
Formal Theories of Knowledge in AI and Robotics.
SRI - Technical Note no 362 - 10 Sep. 1985

[40] G.Ryle.
Knowing how and Knowing that.
Hutchinsons's University Library - 1955

[41] E.D.Sacerdoti.
A Structure for Plans and Behavior.
SRI - T.N. no 109 - Aug 1975

[42] J. R. Searle.
The Intentionality of Intention and Action.
Cognitive Science, 4(1980), pp.47-70

[43] J. R. Searle.
L'intentionalité.
Les éditions de minuit - 1985

[44] R.C.Stalnaker.
Inquiry.
Bradford Books 1984

[45] M.Stefik.
Planning with Constraints (MOLGEN: Part 1)
Artificial Intelligence - 16(1981) - pp.111-140

[46] M.Stefik.
Planning and Meta-Planning (MOLGEN: Part 2)
Artificial Intelligence - 16(1981) - pp.141-170

[47] A.Tate.
NONLIN: a Hierarchical Non-linear Planner.
Edinburgh University - Jun. 1976

[48] R.W.Weyhrauch.
A Prologomena to a Theory of Mechanized Formal Reasoning.
Artificial Intelligence - 13(1980) - pp.133-170

Résumé

Cette étude porte sur l'architecture globale d'un agent rationnel. Nous appelons agent un système autonome, c'est-à-dire capable d'agir sans intervention extérieure. Nous faisons de plus l'hypothèse qu'il est rationnel, c'est-à-dire que l'ensemble de ses activités extérieures, perceptives et déductives sont justifiables par ses buts.

Cette étude comprend deux parties, la spécification et l'implantation.

La première partie est un essai de spécification des notions essentielles pour un agent rationnel à savoir, le raisonnement sur les connaissances, le raisonnement sur les actions et la capacité de faire quelque chose, et l'organisation de ses buts. Nous utilisons les logiques intensionnelles comme outil formel pour l'expression de cette spécification.

La deuxième partie décrit le système CHAOS qui implante une partie des idées que nous avons développées dans la spécification. Cette implantation montre la faisabilité d'une approche globale d'un agent rationnel.

Mots-Clés : agent rationnel
autonomie
logiques intensionnelles

Abstract

This dissertation studies global architecture of a rational agent. An agent is defined as an autonomous system which means a system able to act without external control. An additional hypothesis concerns its rationality which means all its external, perceptive and deductive activities are goal directed.

This study is composed of two parts, the specification and the implementation.

The first part proposes a specification of the essential notions for a rational agent : the reasoning about knowledge, the reasoning about action and the ability to act, and the goal structure. We use the intensional logics as a formal tool to express this specification.

The second part describes the system CHAOS which implements a subset of the ideas developped in the specification. This implementation shows the feasability of a global approach to a rational agent.

Keywords : rational agent
autonomy
intensional logics