Université de Caen / Basse-Normandie

Département d'informatique **École doctorale SIMEM**

UFR Sciences

Systèmes d'agents normatifs : concepts et outils logiques

THÈSE

présentée et soutenue publiquement le 13 décembre 2002

pour l'obtention du

Doctorat de l'UNIVERSITÉ de CAEN

Spécialité : Informatique

(Arrêté du 30 mars 1992)

par

Tiberiu STRATULAT

Composition du jury

Rapporteurs :	Jacques Ferber, professeur d'université	Université de Montpellier
	Camilla Schwind, chercheur CNRS	CNRS Marseille
Examinateurs :	François Bourdon, professeur d'université	Université de Caen
	Amal El Fallah-Seghrouchni, professeur d'université	Université Paris X
	Philippe Mathieu, professeur d'université	Université de Lille
Directeur :	Patrice Enjalbert, professeur d'université	Université de Caen

Groupe de Recherche en Informatique, Image, Instrumentation de Caen — CNRS UMR 6072

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier tout d'abord Patrice Enjalbert, à qui je dois une reconnaissance toute particulière. J'ai trouvé en lui le maître dont tout disciple a besoin : compétence, disponibilité, enthousiasme, mille encouragements et soutien moral quand on commence à douter de soi-même, amitié. Les résultats de cette thèse sont le fruit d'une étroite collaboration avec lui. Je le remercie de m'avoir encouragé et guidé toutes ces années.

Je suis très reconnaissant à Françoise Clérin qui a été l'un de mes principaux interlocuteurs à l'Université de Caen. Je le remercie pour ses remarques et les nombreuses discussions que nous avons eues, pour son grand soutien et sa disponibilité. Je remercie également Gérard Becher pour l'intérêt qu'il a porté à ce travail, tous les membres du groupe Smile pour les différents échanges d'idées durant ma présence à Caen ainsi que tout le laboratoire GREYC pour le cadre de travail très agréable.

Je remercie François Bourdon de m'avoir confié la responsabilité de cette thèse, pour ses idées, pour ses remarques et son soutien au début de mes travaux.

Je remercie les rapporteurs de ces travaux, Camilla Schwind et Jacques Ferber, qui me font l'honneur de leur évaluation.

Je remercie Amal El Fallah-Seghrouchni et Philippe Mathieu qui ont bien voulu s'intéresser à cette thèse et qui ont accepté de participer au jury de thèse.

Ce travail n'aurait pu être effectué sans le soutien de France Télécom R&D. Je tiens à remercier Marc Girault, Fabrice Clerc et Anne Lille qui a eu la charge de suivre cette étude. J'exprime ma gratitude à toutes les personnes qui ont contribué au bon déroulement de mon stage, et plus particulièrement à Eric Malville, Michel Milhau et Emannuel Hym.

Ma reconnaissance va également à Marek Sergot qui m'a fait l'honneur de m'inviter à participer au projet IST-ALFEBIITE et à Brahim Chaib-draa pour le très bon accueil lors de mon séjour à l'Université Laval au Québec.

Je voudrais également remercier Pierre Siegel de m'avoir permis de bien préparer la soutenance de cette thèse et Jacques Labiche, sans l'aide de qui, je n'aurais jamais pu la commencer.

Enfin, je remercie chaleureusement tous ceux qui, en dehors des laboratoires de recherche, m'ont accompagné et soutenu, de loin ou de près. Ils sont nombreux, à commencer par ma famille, et ce travail leur appartient à tous. La part qui revient à Nuca est sans doute la plus substantielle. Elle a toujours su me motiver et me donner les forces nécessaires pour arriver jusqu'à la fin de cette thèse. Mes remerciements à Marius, le visionnaire, celui qui m'a guidé toute ma jeunesse et qui m'a fait partager non seulement sa passion pour la programmation, mais également découvrir l'intelligence artificielle et la programmation logique, dans des temps difficiles pour la Roumanie où l'informatique en était à sa préhistoire. Je salue l'immense aide de mon frère Sorin et le soutien de Sandrine qui a corrigé le manuscrit.

Table des matières

Partie I Concepts et architectures liés à l'intéraction normative

Table des figures

Première partie

Concepts et architectures liés à l'intéraction normative

Chapitre 1

Introduction

Le but de cette thèse et d'étudier l'aspect normatif de l'interaction sociale entre agents et de proposer les outils (logiques) adéquats. Cet effort s'appuie sur les résultats obtenus au cours des deux dernières décénis par les recherches dans le domaine de systèmes multi-agents (SMA).

L'intérêt croissant manifesté pour l'étude des SMA est dû au développement des applications dans un contexte distribué et en particulier à l'Internet. L'architecture d'un tel système est inévitablement basée sur des composants logiciels distribués, conçus spécialement pour résoudre ensemble un problème. Ainsi l'objectif principal des travaux liés aux SMA est de concevoir des méthodes efficaces d'interaction (coordination) entre ces composants afin d'obtenir une certaine caractéristique fonctionnelle ou de faire émerger une certaine propriété du système entier. C'est un problème nouveau et difficile car, de plus, ces composants sont hétérogènes, c.à.d., ils sont conçus par des personnes différentes qui, pour les réaliser, font appel à des méthodes et technologies différentes (concepts, architectures, langages de programmation, etc.). Ce problème sort du cadre du modèle classique computationnel où il s'agissait de résoudre, de façon unitaire, un problème sur un système de calcul qui facilitait le contrôle de son exécution. En conséquence, les travaux sur les SMA proposent des théories et des outils qui permettent de comprendre et de faciliter la construction des structures d'interaction entre ces composants hétérogènes et distribués.

Souvent, les démarches utilisées dans la conception des SMA sont pluridisciplinaires, car les concepts qu'on véhicule font déjà l'objet d'étude dans des domaines très divers (e.g. IA, économie, sociologie, philosophie, psychologie, linguistique, logique, domaine juridique, etc.). La tâche du chercheur informaticien, qui se propose de construire effectivement des outils informatiques, est double. D'abord, il doit aller aux sources afin de comprendre les concepts et la façon dont on s'en sert dans un domaine particulier. Ensuite, il doit comprendre la pertinence et l'applicabilité de ces concepts à son domaine pour faire son choix.

Dans cette thèse nous adoptons aussi une démarche pluridisciplinaire et nous avons choisi d'étudier les normes et le paradigme social, comme source principale d'inspiration pour décrire et structurer l'interaction entre agents. Ainsi, les problèmes que nous nous proposons de traiter sont comme suit :

1. comprendre et expliquer ce qu'est une norme et comment on peut s'en servir dans un contexte multi-agents ;
2. décrire et étudier d'autres concepts (par exemple dans le domaine social) qui sont directement liés à l'utilisation des normes dans les SMA ;
3. proposer des structures architecturales afin de permettre l'utilisation des normes ;
4. préciser les éléments composant une norme et proposer les moyens (formalismes) pour les décrire ;
5. proposer des outils pour la construction des composants d'un SMA normatif.

Dans la suite, nous décrivons notre démarche pour résoudre ces problèmes. La première partie de cette thèse étudie la problématique du domaine des SMA et les concepts nécessaires à l'utilisation des normes. On commence par expliquer pourquoi on s'intéresse aux normes en examinant le conflit conceptuel entre deux notions importantes pour la conception des SMA, l'*autonomie* et le *contrôle*. Les normes seront proposées comme une solution possible de compromis à ce conflit. Comme les normes font partie intégrante du paradigme social, nous montrons dans le chapitre 3 en quoi consiste ce paradigme et comment les notions de dépendance sociale, organisation et rôle nous permettent de mieux structurer et décrire l'interaction entre agents. Afin de mieux expliquer le sens que nous donnons à la notion de norme, nous étudions dans le chapitre 4 la façon dont on s'en sert et les concepts normatifs utilisés ailleurs dans des applications liées au domaine juridique, à la sécurité informatique, aux systèmes distribués ou même dans les SMA. Dans le chapitre 5, nous présentons nos choix et nos solutions concernant les problèmes liés à l'interaction normative. Les structures architecturales résultantes ont été regroupées sous le nom de Système d'Agents Normatifs (SAN). Nous montrons aussi quels sont les acteurs majeurs dans un tel système et quels sont les besoins en termes d'outils pour les construire.

Lors de la description d'une norme et en fonction du choix que nous faisons sur son contenu, on fait intervenir des concepts tels que l'action, le temps, les obligations et la notion d'agence. Ainsi, nous consacrons la deuxième partie à montrer d'une manière critique comment sont formalisés ces concepts dans la littérature (e.g. logique du temps et de l'action, logique déontique) et quels sont les inconvénients majeurs des formalismes correspondants (chapitres 6 et 7). Dans le chapitre 8, nous proposons un modèle temporel pour décrire l'interaction normative dans lequel nous précisons les motivations de nos choix sur les formalismes utilisés et dans le chapitre 9, nous montrons son applicabilité. Plus précisément, nous mettons en évidence l'aspect opérationnel du modèle afin de produire les outils nécessaires à la construction des composants d'un SAN : l'observation des comportements des agents, la détection des violations et l'ordonnancement déontique.

Chapitre 2

Systèmes multi-agents, préliminaires

2.1 Introduction

Pour comprendre les caractéristiques et le comportement des agents normatifs, on doit d'abord étudier les propriétés et les problèmes liés aux agents ordinaires en général. Dans ce chapitre, nous présentons quelques aspects liés à la modélisation et à la conception des SMA. On commence par choisir une définition pour le concept d'agent qui nous convient le mieux. Ensuite, nous présentons certaines caractéristiques des agents qui permettent d'introduire les normes dans les SMA, comme l'autonomie, le contrôle, l'anthropomorphisme et la rationalité. Nous finissons par une présentation de quelques solutions proposées dans la littérature au problème central des SMA, la coopération.

2.2 Définitions du terme *agent*

Le terme agent reste encore très controversé à propos de sa signification. Même si les efforts de standardisation des aspects opérationnels de la conception SMA sont de plus en plus reconnus [FIPA, 2000], toute tentative d'unifier les différentes théories sur les agents ou de trouver un dénominateur commun semble se trouver, pour l'instant, sous le spectre de l'échec [Wooldridge et Jennings, 1995].

Le terme *agent* a été utilisé principalement dans le domaine de la philosophie de l'esprit bien avant qu'il soit utilisé par les théories liées aux SMA ou plus généralement par l'informatique. Certains philosophes l'ont employé pour éluder l'utilisation du mot *humain* dans les différents modèles philosophiques exprimant une image abstraite du monde réel. Comme la philosophie et l'intelligence artificielle ont souvent des objets d'étude communs, ce terme s'est imposée naturellement dans les modèles proposés par le domaine de SMA qui traitent de l'interaction calculatrice entre différents éléments physiques ou virtuels où l'humain cotoie des composantes logicielles.

Plus généralement, en informatique, l'origine de ce terme est souvent oubliée et suscite beaucoup de polémiques autour de son sens, notamment en raison d'une utilisation abusive et inappropriée. On peut utiliser ce mot pour désigner pratiquement tout élément ou produit logiciel. Par exemple, on appelle agents les processus à synchroniser dans certains algorithmes qui utilisent le calcul distribué ou parallèle, sans avoir en réalité besoin d'étudier leur interaction, coopération, etc. Un autre exemple est celui où la notion d'agent décrit une composante logicielle qui a comme fonction d'assister l'utilisateur humain dans l'utilisation d'un produit logiciel [Microsoft, 1999].

Plusieurs définitions [Wooldridge et Jennings, 1995; Shoham, 1993] ont été proposées dans le domaine des SMA, notamment par Ferber, pour qui un agent est une entité physique ou virtuelle [Ferber, 1995] :

1. capable d'agir sur elle-même et sur son environnement ;
2. capable de percevoir son environnement (avec une représentation partielle) ;
3. capable de communiquer avec d'autres agents ;
4. qui poursuit un objectif individuel ;
5. qui possède des compétences et peut éventuellement se reproduire ;
6. dont le comportement est la conséquence de toutes les propriétés énumérées ci-dessus.

Cette définition n'est ni la meilleure ni la plus mauvaise. On la préfère parce qu'elle délimite une classe d'agents à laquelle on s'intéresse dans la suite. Pourtant, on note ici que toute définition peut être limitative et risque de restreindre les domaines d'application des systèmes multi-agents proposés.

2.3 Caractéristiques d'un agent

2.3.1 Autonomie

L'*autonomie* est un concept central dans la définition du comportement des agents. Elle a été initialement utilisée dans la robotique pour décrire les capacités des robots à réagir de manière adéquate aux événements inattendus qui peuvent apparaître dans l'environnement physique réel. Par exemple, les robots doivent se déplacer physiquement entre deux points différents tout en évitant les éventuels obstacles. Ce n'est que plus tard, au moment de l'apparition du domaine des SMA, qui étudie l'interaction entre plusieurs composantes hétérogènes, qu'on parle d'agents autonomes. Par exemple, pour Demazeau et Müller [Demazeau et Müller, 1990], un agent autonome est un agent dont l'existence ne se justifie pas par l'existence des autres agents. Pour Castelfranchi [Castelfranchi, 1995], un agent autonome s'inscrit dans la définition proposée par Ferber, citée ci-dessus, mais ils y ajoutent la notion que l'agent agit sans l'intervention des humains ou des autres agents, ayant un contrôle sur ses actions et son état interne.

A notre avis, il y a deux raisons principales qui expliquent l'utilisation du concept d'autonomie dans la modélisation et la conception des SMA. La première concerne le besoin de flexibilité et d'adaptabilité du système face aux nouvelles situations. L'adaptabilité est une propriété fondamentale, nécessaire dans les domaines fortement dynamiques. Un agent doit se plier aux demandes ou contraintes externes afin de réaliser ses objectifs ou même de survivre dans le milieu où il se trouve. En effet, par l'introduction de la notion d'agent autonome et *situé*, on a généralisé le concept de robot et d'environnement physique. On parle d'un agent comme étant un robot qui n'a pas forcément un corps physique. Il est situé parce qu'il est capable de percevoir l'environnement (physique ou virtuel) dans lequel il vit par l'intermédiaire des capteurs et de modifier cet environnement par l'intermédiaire des effecteurs (actions).

La notion d'autonomie est définie donc, par rapport à la capacité de l'agent d'exécuter une action à sa propre initiative (pro-actif). On suppose que les agents logiciels peuvent contrôler leur comportement afin d'atteindre leurs buts et de décider s'ils aident ou empêchent les autres de réaliser eux-mêmes leurs objectifs. Avec ces hypothèses, l'autonomie s'exprime en termes de degrés de *liberté*, ce qui revient à dire qu'un agent est libre de décider de ce qu'il fait et libre de percevoir son monde.

Une deuxième motivation pour l'utilisation de l'autonomie concerne le point de vue du concepteur du système. Le concepteur dispose d'un ensemble d'agents et a comme objectif de résoudre un problème par l'intermédiaire de l'interaction des agents. Comme on l'a déjà vu, le terme agent a été proposé pour occulter les origines variées des différents participants à l'interaction, qui peuvent être des humains ou des logiciels. Dans le cas des agents logiciels, il s'agit des programmes conçus par différentes personnes qui utilisent des techniques variées de conception. Si le concepteur du système veut obtenir (modifier) un certain type de comportement de la part d'un agent, alors il lui est difficile de connaître les détails de construction de chaque agent logiciel, voire impossible dans le cas des agents humains. En bref, l'autonomie indique qu'un éventuel observateur n'a qu'un modèle incomplet du modèle interne de l'agent. Dans certaines situations, un agent doit être considéré comme étant une boîte noire dont on connaît seulement la description fonctionnelle. Cela protège aussi l'agent : en considérant ses capacités de contrôler l'accès à ses ressources, on réduit les possibilités de l'influencer.

Pour mettre en évidence l'aspect autonome des agents, on compare souvent les comportements des agents et les objets de la programmation orientée-objet. Par exemple, l'appel d'une méthode d'un objet renvoie toujours un résultat qui est décrit par un algorithme déterministe implémenté par la méthode en cause. Par contre, si on pose une question à un agent, on peut obtenir un comportement non déterministe caractérisé par une réponse ou une absence de réponse, un refus, ou une proposition de négociation pour obtenir une réponse. Le point de vue du concepteur peut ainsi être étendu aux agents en général. Chaque agent doit considérer les autres agents comme étant autonomes, c.à.d. qu'il connaît seulement la description fonctionnelle de leur comportement.

2.3.2 Intelligence et anthropomorphisme

Le fameux test de Turing [Turing, 1950] donne, d'une certaine manière, une définition de l'intelligence d'un artefact (ou l'intelligence artificielle). Un artefact est intelligent si, en observant les comportements d'un humain et de l'artefact, sans savoir qui est l'un et qui est l'autre, on ne peut pas dire (après l'observation) qui est l'humain et qui est l'artefact. Cette définition est donnée d'une façon subjective parce qu'elle est introduite par rapport à l'humain qui est considéré implicitement comme ayant un comportement intelligent.

Dans les SMA on étudie principalement l'interaction entre plusieurs éléments d'un système tout en ignorant leurs origines diverses. On peut donc avoir, dans le même système, des artefacts et des humains qui interagissent les uns avec les autres sans faire de discrimination entre eux. De ce point de vue nous pensons que le but du domaine des SMA s'inscrit dans la perspective annoncée par le test de Turing et consiste à proposer des artefacts aussi intelligents que les humains, au moins en ce qui concerne l'interaction.

Dans cette thèse, on s'inscrit dans le droit fil d'une tradition déjà bien établie en IA, celle qui décrit les agents en termes d'états mentaux comme les croyances, les désirs, les buts [McCarthy, 1979; Cohen et Levesque, 1990; Shoham, 1993; Rao et Georgeff, 1995]. Le philosophe Daniel Dennett [Dennett, 1978] utilise le terme *système intentionnel* pour les entités dont le comportement peut être prédit par l'attribution des capacités mentales. Pour Cohen, qui construit une théorie sur l'intentionnalité, un agent n'agit que selon ses intentions. Il choisit des buts qui sont réalisables et abandonne ceux qui sont irréalisables. Le choix des buts est le résultat d'un processus cognitif basé sur les connaissances dont l'agent dispose. On remarque que dans ce type d'approche l'intelligence est implicite. Faisant l'hypothèse des capacités cognitives s'avere être suffisante pour considérer un agent intelligent. De plus, cette hypothèse est très forte, car elle suppose que nous savons toujours ce que nous voulons faire. On note l'apparition d'une autre théorie (voir [Drogoul, 1993]) qui explique l'intelligence d'un système par sa simple capacité d'interagir avec son environnement sans avoir des buts construits *a priori*.

De toute façon, dans la construction des SMA, la motivation de cette démarche de type anthropomorphique consiste à offrir des outils plus expressifs aux concepteurs des systèmes d'agents et plus proches de la façon dont nous, les humains, décrivons les comportements des autres humains. Par exemple, quand on dit d'une personne qu'elle croit ou veut telle ou telle chose, on emploie souvent des expressions qui traitent une machine comme étant intelligente ou ayant un esprit ou une âme : « la voiture ne veut pas démarrer » ou « l'ordinateur connaît la réponse ». On le fait naturellement pour notre confort dans nos activités ou pour optimiser les efforts de représentation et de compréhension, les pensées, etc. La même pratique appliquée aux agents considérés dans un contexte social (voir plus bas) s'avère utile pour représenter les interactions sociales où les agents tiennent compte de la présence d'autres agents.

2.3.3 Réactivité

La *réactivité* d'un agent est définie comme la capacité de percevoir et de réagir aux modifications survenues dans son environnement. Le type de réactivité est traité de manière différente en fonction de la manière dont l'agent est construit (voir la section 2.4 sur les types d'agents). Si les agents sont capables de construire eux-mêmes des buts à atteindre sans l'aide de stimuli reconnaissables, on parle de pro-activité (en anglais *pro-activeness*).

2.3.4 Sociabilité

La *sociabilité* signifie que les agents sont capables d'interagir avec d'autres agents, même des humains. Cela nous a conduit à étudier la conception SMA du point de vue de la perspective sociale. La perspective sociale [Gasser, 1991] dans les SMA met l'accent sur les interactions qui ont lieu dans le système ou le groupe d'agents, plutôt que sur la construction interne de l'agent individuel. En général, dans ce type de démarche, on ne considère pas que le social met en danger l'autonomie des agents. On se concentre plus sur leur organisation (structure) efficace pour produire un système qui a un comportement global désiré, que sur leur interaction avec l'environnement. Les agents ont une représentation non seulement de l'environnement et de leurs propres caractéristiques mais aussi des autres agents.

Pourtant, dans la pratique, il existe des cas où l'autonomie des agents est une propriété importante à préserver, car on ne peut pas ignorer le fait que le social implique l'introduction des contraintes sur le comportement des agents qui parfois sont en contradiction avec leurs objectifs.

2.3.5 Hétérogénéité

Une des applications (ou plutôt des métaphores) préférées dans le domaine des SMA est la modélisation des Systèmes Ouverts (SO). Un tel système [Hewitt, 1991] est caractérisé par des propriétés en permanente modification et en conséquence difficilement contrôlables :

- les SO sont dans une interaction continue avec leur environnement ;
- les SO ont des composantes dynamiques (les agents et les ressources entrent et sortent du système pendant son exécution), hétérogènes (conçues par diverses personnes avec des méthodes différentes à des moments différents), distribuées (sur plusieurs machines), multiples.

Le concepteur d'un SO est confronté donc au problème de la mise en relation des différentes architectures, ontologies, types de représentations (de connaissances), langages de communication, etc. Lors de l'application de la méthodologie SMA à la conception des SO, la notion d'agent se substitue à la composante ordinaire du SO impliquant souvent la prise en compte de l'hypothèse de l'*hétérogénéité*.

Par opposition à l'hétérogénéité, l'*homogénéité* reflète l'idée d'un concepteur commun pour tous les agents, montrant qu'il n'y a pas de différences entre eux. Les agents homogènes utilisent le même type de représentation, le même langage de communication, etc. Pourtant, l'hypothèse de l'homogénéité n'implique pas la non-existence de conflits entre les (buts des) agents.

2.4 Types d'agents

La notion d'agent se réfère par définition à quelqu'un qui agit. On classifie [Wooldridge et Jennings, 1995] souvent les types d'agents en fonction de leur *comportement* individuel. Ainsi, on peut avoir trois grandes classes d'agents : cognitifs, réactifs et hybrides. Tous les modèles que nous présentons ont la même architecture basée sur le cycle *observation - décision - action*. L'observation est possible grâce à la présence des senseurs pour capter les informations venues du monde externe. Quant à l'action, elle est possible grâce aux effecteurs, pour modifier l'état du monde ou pour agir dans ce monde. Le module de décision, qui choisit l'action à exécuter à partir de l'information générée par les senseurs, est la partie qui fait la différence entre ces trois types d'agents.

2.4.1 Agents cognitifs

Les SMA de type cognitif suivent le paradigme proposé par l'IA symbolique. On considère qu'on peut créer un agent en mettant ensemble différents modules cognitifs qui ont comme base une représentation symbolique du monde. Le moteur d'exécution d'un agent est souvent caractérisé par un cycle réactif (dans le sens proposé par Pnueli [Pnueli, 1986]) de type perception - délibération - action. L'agent est doté des senseurs capables de générer l'information utilisée pour la mise à jour des connaissances sur le monde. Un module de raisonnement (planification) symbolique est capable, à partir de ce modèle, de prendre la décision concernant l'action qui sera exécutée par la suite.

Les architectures de ce type plus connues sont les architectures BDI[1]. Les connaissances (*beliefs*), les désirs et les intentions d'un agent sont les notions cognitives primitives et caractérisent l'état de l'agent. Le modèle d'un agent BDI est représenté comme un système composé de structures dynamiques de données correspondant aux notions cognitives de base et d'une file d'événements gardant tous les événements que l'agent doit traiter. Les événements sont considérés comme les entrées du système produits par différentes sources telles que les informations venues de l'extérieur de l'agent (événements externes) ou les modifications de l'état de l'agent (événements internes). Les sorties du système sont des actions atomiques exécutées par l'intermédiaire d'une fonction $execute$. A partir de son état courant et du contenu de la file d'événements, le système sélectionne et exécute les *options* qui correspondent à des procédures, règles de production, tâches, plans ou automates à états finis. Les actions une fois exécutées

1. Acronyme pour *Beliefs - Desires - Intentions*.

```
BDI-iterpreter
initialize-state() ;
do
    selected-events := event-selector(event-queue) ;
    options := option-generator(selected-events,B,G,I) ;
    selected-options := deliberate(options, B,G,I) ;
    update-intentions(selected-options, I) ;
    execute(I) ;
    get-new-external-events() ;
    drop-successful-attitudes(B,G,I) ;
    drop-impossible-attitudes(B,G,I) ;
until quit
```

FIG. 2.1 – *L'interpréteur BDI*

peuvent générer à leur tour des événements, qui, ainsi générés permettent à nouveau d'invoquer d'autres options, et ainsi de suite.

Le moteur d'exécution d'un agent est décrit par un interpréteur contenant une exécution en boucle dont la description est présentée dans la figure 2.1.[2]

Au début du cycle, on exécute une sélection parmi les événements de la queue. Dans les systèmes comme PRS [Georgeff et Lansky, 1987] et son successeur dMARS [d'Iverno *et al.*, 1997], le sélecteur d'événements sélectionne un seul événement. Le générateur d'options lit les événements sélectionnés et retourne les meilleures options (alternatives de plans) qui sont déclenchées par ces événements. Dans l'étape de délibération, on sélectionne les plans à exécuter. Pour des raisons d'efficacité, le processus de délibération fait appel à des méta-plans (ou stratégies). Le résultat sera déposé dans la structure afférente aux intentions. Si, parmi ces intentions, il s'en trouve une pour l'exécution immédiate d'une action atomique, l'agent l'exécute. Tous les éventuels événements externes générés pendant l'exécution d'un cycle seront ajoutés dans la queue d'événements. Les événements internes sont ajoutés au moment de leur création. Ensuite, l'agent modifie les structures d'intentions et de buts en effaçant les buts et les intentions satisfaites, ainsi que les buts impossibles ou les intentions irréalisables.

PRS et dMARS sont les systèmes les plus connus implémentant le modèle BDI et utilisés dans des applications pratiques des SMA [Georgeff et Rao, 1996]. Haddadi et Sundermeyer [Haddadi et Sundermeyer, 1996] décrivent IRMA comme l'une des premières architectures BDI. Les agents IRMA sont

2. On remarque qu'un agent peut avoir des désirs contradictoires. Pour simplifier on ne traite que les désirs mutuellement consistants et qu'on appelle buts (*goals*). Dans la figure 2.1 on note avec G l'ensemble des buts de l'agent.

composés de quatre structures de données : une bibliothèque de plans et des structures pour représenter les croyances, les intentions et les désirs. Cinq processus traitent ces structures :

- un module de raisonnement qui met à jour les croyances actuelles de l'agent,
- un analyseur *means-ends* qui à partir de l'ensemble des croyances va générer les options (les plans) répondant le mieux aux intentions de l'agent,
- un analyseur d'opportunités qui observe l'environnement et décide d'autres options éventuelles,
- un processus de filtrage qui assure que l'option choisie est cohérente avec l'ensemble courant d'intentions,
- un processus de délibération qui fait le choix entre des options concurrentes.

COSY [Haddadi, 1996], une autre architecture BDI, enrichit le modèle d'origine avec des composantes pour la coopération basée sur la notion d'engagement social et sur des protocoles de communication. Et enfin, les agents ARCHON [Cockburn et Jennings, 1996] sont construits sur la base du modèle BDI étendu, car ils prennent en compte l'existence explicite des autres agents dans leurs processus de décision.

Sadek [Sadek, 1991] propose une théorie formelle de l'interaction rationnelle par la communication, qui est basée aussi sur les notions de croyances et d'intentions. Le résultat pratique de cette théorie a donné naissance au système ARTIMIS [Bretier et Sadek, 1996] qui implémente un seul agent offrant une interface capable de dialoguer avec un interlocuteur humain, et utilisable par exemple pour la réalisation des serveurs vocaux qui offrent divers services d'informations (i.e. météo). ARTIMIS essaie d'inférer les intentions de son interlocuteur et d'agir en conformité. Le langage de communication pour dialoguer avec l'utilisateur, ARCOL, est le précurseur de FIPA-ACL, le langage de communication retenu par FIPA.

Les principales difficultés posées par la conception d'agents cognitifs sont liées à la représentation symbolique de leur univers et aux mécanismes de raisonnement. Les formalismes et les techniques de représentation de connaissances proposent des algorithmes qui se révèlent très coûteux pour assurer une bonne réactivité du système ou pour garantir un résultat dans un intervalle de temps fixé. Ces caractéristiques sont souvent très importantes pour rendre les agents opérationnels dans des applications réelles.

2.4.2 Agents réactifs

Étant données les difficultés que la démarche cognitive introduit, les agents réactifs ont été proposés comme une alternative possible. Le principe utilisé dans la construction des architectures réactives est basé sur l'idée de l'émergence d'un comportement complexe du système grâce à la coexistence des comportements individuels simples. Un agent réactif n'a pas de représentation de son environnement ou des autres agents. Son comportement est décrit par des règles simples du type stimulus/réponse. Alors que les agents cognitifs sont peu nombreux, l'intelligence étant localisée au niveau de chaque agent (un agent est en fait un expert), le nombre d'agents réactifs est plus significatif. Dans le cas réactif, l'intelligence ne se manifeste (émerge) qu'au niveau du système.

Parmi les premiers travaux sur les agents réactifs, on peut citer ceux de Brooks dans la robotique [Brooks, 1986]. Pour construire un robot intelligent, il propose une architecture hiérarchique (*subsumption architecture*) composée de plusieurs composantes (couches) réactives qui sont en compétition les unes avec les autres pour contrôler le robot. Les niveaux inférieurs s'occupent des comportements primitifs, par exemple, la mannière d'éviter un obstacle. Les niveaux supérieurs peuvent rendre inactifs les niveaux inférieurs en filtrant leurs entrées et sorties (effecteurs). Une architecture similaire a été proposée par Steels [Steels, 1995], qui, au lieu d'utiliser le principe de *subsumption*, applique la coopération (ou la compétition) entre les différentes couches. On cite aussi les travaux d'Agre et Chapman [Agre et Chapman, 1987] qui proposent Pengi, une architecture réactive utilisable dans la planification. Ferber propose le modèle des *éco-agents* qui trouve son inspiration dans le comportement des sociétés d'insectes dans la nature. Un éco-agent est un agent très simple qui cherche toujours à satisfaire ses besoins. S'il est dérangé dans son activité, il agresse les autres. Le modèle a été utilisé pour la résolution distribuée de problèmes [Ferber et Drogoul, 1992].

2.4.3 Agents hybrides

Les caractéristiques présentées pour les deux grands types d'agents, cognitifs et réactifs, proposent des solutions apparemment diamétralement opposées. Pourtant, elles peuvent être vues comme étant complémentaires. Afin de construire une architecture qui répond au mieux (temps de réponse, précision ou efficacité) à un problème, on peut combiner les deux types d'approche pour construire des architectures d'agents plus souples qu'on appelle hybrides. Dans une telle architecture, un agent est composé de modules qui gèrent indépendamment la partie réflexe (réactive) et réfléchie (cognitive) du comportement de l'agent. Le problème central reste de trouver le mécanisme idéal de contrôle assurant un bon équilibre et une bonne coordination entre ces modules. Nous citons TouringMachines [Ferguson, 1992] et InteRRap [Müller, 1996] qui sont les exemples les plus connus d'architectures hybrides.

2.5 Coopération et coordination

L'objectif principal de la recherche SMA est de proposer des solutions à un problème comme étant le résultat des activités (en commun) réalisées par les agents. Dans un SMA chaque agent exerce un contrôle sur ses actions et gère ses interactions avec les autres agents. Les notions de coordination et de coopération entre agents sont alors très importantes pour décrire ou comprendre comment on arrive à obtenir ce résultat commun. Les deux concepts décrivent en effet la même chose : les mécanismes qui contraignent les agents à ajuster leur comportement pour participer à une activité commune. Vu de l'extérieur, un agent se coordonne avec les autres, vu de l'intérieur, l'agent coopère.

Les mécanismes de coordination vont donc déterminer la dynamique à l'intérieur du système, ainsi que ses propriétés externes. Ces mécanismes dépendent :

- du type d'agent qui participe à l'interaction, et plus précisément de son attitude par rapport à la coopération : coopératif, opportuniste, antagoniste, etc.
- de la façon dont l'environnement est prédictible ou manifeste une certaine stabilité ;
- du degré de décomposabilité des activités ou des tâches et du fait qu'elles sont bien définies.

Dans la littérature SMA, plusieurs mécanismes de coordination des agents ont été proposés. Durfee et al. [Durfee *et al.*, 1987] décrivent trois mécanismes : la planification, l'organisation et l'échange de méta-informations. L. Gasser [Gasser, 1992] rajoute à ceux-ci la synchronisation explicite et plus tard la négociation. Dans la suite, nous détaillons quelques mécanismes que nous considérons importants du point de vue de cette thèse.

2.5.1 Planification

La planification faisait partie du domaine de recherche de l'IA, bien avant qu'on s'intéresse aux aspects distribués. Elle consiste à trouver dans un espace d'états un chemin liant deux états appelés état initial et état final. Dans l'approche classique, ce processus de recherche est réalisé par un seul planificateur (ou agent). Si l'environnement change seulement à cause de l'action de l'agent il, s'agit d'une planification *statique*. L'agent connaît tous les états possibles de son environnement et tous les événements qu'il peut générer pour le modifier. Si d'autres événements (inattendus) peuvent modifier l'état du système, on parle alors d'une planification *dynamique*. Dans ce cas, l'agent doit observer l'impact de ces événements sur l'exécution de son plan et si nécessaire, reconsidérer son déroulement (voir [Russell et Norvig, 1995]). Afin de résoudre ce problème, deux types de planificateurs ont été proposés dans la littérature : *hiérarchique* et *réactif*. La planification hiérarchique repose sur la hiérarchisation du plan sur plusieurs niveaux d'abstraction, du plus général (les grandes actions à accomplir) au plus spécifique (les actions de base ou les mieux spécifiées). Dans la planification réactive, on connaît par avance quels sont les plans, le problème consistant à choisir le bon plan dans une situation donnée. La difficulté majeure de cette méthode vient de l'impossibilité de prévoir à l'avance toutes les situations possibles.

Dans le contexte multi-agents, la planification est vue comme un mécanisme de coordination, où les agents ont des plans décrivant leurs actions futures et les interactions par rapport à un objectif commun. Dans ce cas, tous les agents qui y participent s'engagent à agir conformément aux plans. On obtient ainsi une meilleure cohérence entre leurs activités grâce au fait que les agents savent à l'avance quelles sont les actions à exécuter par eux-mêmes et par les autres et quelles sont les interactions qui vont avoir lieu dans le système [Jennings, 1996].

En fonction de la manière dont le plan est généré, on distingue deux approches principales dans la planification multi-agents : la planification *centralisée* et la planification *distribuée*. Dans la planification centralisée, un plan commun est conçu par un agent central qui a une perspective globale sur le problème à résoudre. Le rôle de cet agent est de préciser les actions des différents agents et de gérer les éventuels conflits. L'inconvénient de cette approche réside dans le fait qu'en pratique, dans les applications réelles, les agents sont physiquement distribués et donc un agent central est inapproprié comme solution. Dans la planification distribuée, chaque agent est responsable de la construction de son propre plan. La coordination est réalisée par l'échange d'informations sur les plans partiels, en évitant ainsi les conflits potentils. On remarque que cette approche donne seulement une direction d'étude. Même si elle est mieux adaptée aux domaines d'applications SMA, en réalité, beaucoup de choses restent à faire pour réaliser une gestion effective des comportements des agents tout en évitant les conflits. Pour une bonne présentation des différentes approches qui existent dans ce domaine nous suggérons [Durfee, 1998].

2.5.2 Organisation

Dans les SMA, la notion d'organisation sous-entend un ensemble de relations structurelles entre plusieurs rôles. Quand un agent accepte de jouer un certain rôle dans une organisation, il s'engage à se conformer au comportement que le rôle décrit, ainsi qu'aux relations que celui-ci implique. Une structure organisationnelle a donc un très fort aspect coercitif qui, en fonction de son degré de manifestation, différencie les divers types de relations existantes. Par exemple, les structures de responsabilité peuvent définir les rôles nécessaires à la réalisation de certaines tâches. On réduit ainsi le niveau d'incertitude locale à un agent en lui offrant des informations sur les comportements potentiels des agents qui l'entourent. Un autre exemple concerne les relations d'autorité, qui indiquent quel rôle (agent) peut donner des ordres à d'autres agents. Ces relations sont donc utilisées pour décrire la façon de résoudre un problème ou simplement d'éviter les conflits.

On remarque ici qu'il y a un très important lien entre les structures relationnelles et la structure de la tâche ou de l'activité à réaliser. Par exemple, le fait d'être responsable d'une tâche (complexe et décomposable) implique que l'agent en cause est désigné comme ayant la possibilité de coordonner les différentes sous-tâches et éventuellement de donner des ordres aux agents supposés les exécuter. Dans les organisations de type hiérarchique [Fox, 1981], pour chaque tâche, un responsable contrôle la réalisation et coordonne les efforts des agents du niveau inférieur. En fonction de leur composition on connaît deux types de hiérarchies : fonctionnelle et de produit. Dans le cas d'une *hiérarchie fonctionnelle*, les agents qui ont les mêmes caractéristiques ou compétences sont réunis dans un département fonctionnel. A chaque département on attribue un gestionnaire fonctionnel. Un bureau exécutif est chargé de décider quelles sont les tâches à réaliser pour produire tous les produits de l'organisation et les départements appropriés pour les exécuter. Une fois la tâche déléguée au gestionnaire du département fonctionnel, celui-ci décide quel agent de son groupe sera finalement désigné. Dans le cas d'une *hiérarchie de pro-*

duit, les agents sont répartis dans des divisions qui correspondent à des lignes de produit. Chaque division possède un gestionnaire, appelé gestionnaire de produit, qui est spécialisé pour un type de produit particulier. Chaque gestionnaire contrôle entièrement les tactiques utilisées pour la réalisation de ce type de produit. Il décide quelles sont les tâches à réaliser et attribue ces tâches aux acteurs appropriés de sa division (voir également la section 3.6 sur les rôles).

En général, les structures relationnelles entre les rôles ne changent pas très souvent, et donc on peut les traiter comme étant des connaissances communes partagées par tous les agents. La façon dont on les représente dépend de la manière dont une organisation est définie. De ce point de vue, plusieurs travaux ont été proposés et classifiés en deux catégories. D'abord ceux qui considèrent l'organisation comme une entité externe et coercitive (représentable même par un objet ou un agent), et ceux qui la considèrent comme existant seulement dans les engagements et les expectatives des agents [Chaib-draa *et al.*, 1992].

2.5.3 Un exemple : le réseau contractuel (*Contract Net Protocol*)

Le réseau contractuel [Smith, 1980] a fait l'objet de nombreuses études et analyses. Il est souvent cité comme un exemple de structure organisationnelle qui utilise comme mécanisme de coordination le principe de la négociation. Même s'il ne propose pas un modèle formel de négociation, son mécanisme de fonctionnement est celui du marché [Malone, 1987] dans lequel tous les agents, acheteurs ou fournisseurs, sont en contact les uns avec les autres.

Ce système repose sur un mécanisme d'allocation de tâches régi par le protocole d'offres. Un agent peut jouer deux rôles par rapport à une tâche : de *manager* (gestionnaire) et de *contractor* (contractant). Le manager est responsable du contrôle de l'exécution et le contractant de l'exécution effective. Le protocole est composé de trois étapes. Dans la première étape, l'appel d'offres (*task announcement*), le manager à qui on a confié la réalisation de la tâche fait une annonce de tâches aux agents du système. Les agents contractants analysent dans une deuxième étape leurs capacités et leur intérêt par rapport à la réalisation de la tâche donnée. Si le résultat de cette analyse est positif, ils peuvent soumettre une offre (*bid*) au manager. Ensuite, dans la troisième étape, le manager choisit un ou plusieurs agents parmi les contractants qui ont répondu et les informe de son choix (*announced award*).

2.6 Conclusions et discussions

Dans ce chapitre nous avons présenté succinctement quelques aspects liés à la conception des systèmes multi-agents. Nous avons mis en évidence certaines caractéristiques telles que l'autonomie, la rationalité, la réactivité, la sociabilité et la coopération, qui nous aideront à présenter par la suite l'aspect normatif de l'interaction entre agents. La normativité sera envisagée comme une solution possible pour résoudre le problème de la coopération, réalisable par l'intermédiaire d'un compromis qui simultanément protège l'autonomie des agents et contraint leur comportement.

La thèse centrale qui est à la base de toutes nos motivations, dans l'explication du choix des normes, considère qu'il peut y avoir une différence entre le comportement *idéal* qu'on veut obtenir d'un agent et le comportement *réel* qu'on obtient finalement de lui. Ainsi, on met en évidence le conflit existant dans la façon de construire les SMA, entre la volonté de contrôler les comportements des agents (afin de les coordonner) et le besoin d'avoir des agents autonomes.[3] Les normes, en tant que descriptions du comportement idéal, ainsi que tous les mécanismes qui permettent leur application, seront considérées comme une solution de *compromis* pour résoudre le conflit entre l'autonomie et le contrôle. On laisse les agents autonomes, c.a.d. faire ce qu'ils veulent, mais tout en surveillant ce qu'ils doivent faire. Le contrôle peut se réaliser à différents degrés, en fonction du caractère plus ou moins restrictif de la norme.

Un autre aspect lié à l'utilisation des normes est l'aspect *informatif*. Une norme est une source d'informations pour les composants d'un système. Elle donne des indications sur la façon dont un agent doit agir, sur ce qu'on veut obtenir de lui, ou sur les conditions dans lesquelles il est en légalité. Ces indications peuvent venir d'une autorité du système ou d'un autre agent ordinaire. Elles peuvent être l'expression d'un but général, dit social, qui concerne tout le système ou exprimer le désir personnel d'un agent individuel.

Par conséquent, l'un de nos objectifs est d'étudier les concepts et les structures permettant l'utilisation des normes dans le monde des agents. Notre démarche consiste d'abord à choisir le type d'agent idéal participant à l'interaction normative. Même si la normativité et la rationalité sont deux notions orthogonales, comme nous allons le voir dans les chapitres suivants, nous optons pour le cas des agents rationnels BDI[4]. La raison de notre choix est que le modèle BDI a montré sa pertinence pour modéliser le comportement d'un agent, même si en pratique la complexité qu'il implique est difficile à gérer.

Ensuite, nous étudierons comment on peut enrichir le modèle de l'agent et la structure d'un SMA avec des concepts qui tiennent plus du domaine social, par exemple les relations de dépendance sociale ou la façon dont un agent s'intègre dans une organisation sociale. Une autre étude permettra de comprendre la notion de norme et la façon dont un agent peut prendre en compte la présence de celle-ci. Le résultat de cette étude va nous amener à fournir des structures architecturales qui permettent l'interaction normative, en particulier la construction des SAN.

3. A noter l'antagonisme conceptuel qui existe entre les notions de contrôle et d'autonomie : la première contraint, la deuxième libère.

4. Ce choix n'implique en aucun cas que nous considérons que les agents réactifs ne peuvent pas être aussi des agents normatifs. Pourtant, notons qu'on peut se demander si un agent réactif peut assurer les structures nécessaires à l'utilisation des normes.

Chapitre 3

Aspects sociaux

3.1 Introduction

Dans ce chapitre, nous allons montrer comment les systèmes naturels (et en particulier les systèmes sociaux) peuvent être utilisés comme source d'inspiration par les domaines de l'Intelligence Artificielle et des SMA dans leur tentative pour modéliser les systèmes artificiels.

La méthode générale employée consiste en plusieurs étapes. D'abord, on essaie de comprendre, d'une façon intuitive, le fonctionnement des systèmes naturels. En partant de l'observation simple, empirique, on identifie les entités de base et les relations entre celles-ci. Le résultat de cette observation sera utilisé à l'étape suivante, l'analyse rationnelle ou philosophique. Son rôle est de produire un modèle qui explique la réalité et généralise les résultats observés par l'intermédiaire d'une abstraction en employant un langage rigoureux et précis. Souvent, le langage employé est celui de la logique. Ce choix présente des avantages importants, confirmés au cours des années : la précision des termes utilisés, le pouvoir d'abstraction, de généralisation et de calcul symbolique, la rigueur et la concision de ses énoncés, l'existence des outils pour l'automatisation du raisonnement logique. Le modèle ainsi créé peut être instancié soit pour fournir des outils de simulation de la réalité qu'il exprime, soit pour construire des systèmes artificiels qui imitent ou héritent des caractéristiques des systèmes naturels. Souvent ce processus peut être itératif, dans la mesure où après une confrontation analytique entre le système naturel et son approximation artificielle on reprend la méthode pour affiner le niveau de notre compréhension du monde réel et de son applicabilité dans la conception des artéfacts.

Ce que nous allons présenter dans la suite sont des directions d'étude qui appartiennent à la fois à l'IA et aux domaines connexes dans lesquels elles trouvent leur inspiration, comme la sociologie, la psychologie sociale, la philosophie, la philosophie juridique. Ces études essaient d'appliquer la méthode de modélisation présentée ci-dessus pour expliquer d'une part ce qu'est un système social artificiel et d'autre part quels sont les éléments qu'on *peut* emprunter aux systèmes naturels comme par exemple l'action sociale ou le caractère normatif des lois sociales. Par « peut » on comprend l'analyse de l'utilité (ou de l'expressivité) de l'introduction de ces concepts dans les systèmes artificiels, ainsi que la complexité calculatoire que cette démarche implique.

3.2 Systèmes sociaux artificiels

Le premier concept que nous décrivons est celui du système social artificiel (SSA). Avant d'analyser plus en détail les raisons justifiant l'introduction de ce concept, nous examinerons les définitions des termes qui le composent : social, système et artificiel. Même si leur emploi ne pose aucun problème à la compréhension du discours dans la pratique courante, nous pensons que la présentation de certains aspects définitionnels aide à mieux illustrer l'utilité de ce concept.

Le mot *social* est principalement utilisé en rapport avec une société ou une collectivité humaine. Parmi les différentes définitions [5] du terme société nous retenons :

> 1. Mode de vie propre à l'homme et à certains animaux, caractérisé par une association organisée d'individus en vue de l'intérêt général.
> 2. Ensemble d'individus vivant en groupe organisé ; milieu humain dans lequel quelqu'un vit, caractérisé par ses institutions, ses lois, ses règles.

Dans ces deux définitions on remarque d'abord la perspective d'ensemble d'une société, la notion de social supposant la considération d'un individu dans un contexte global et non en isolation. La deuxième remarque concerne la notion d'organisation, d'association organisée ou d'institution, c.a.d le fait qu'une société a par définition une structure qui décrit des règles de fonctionnement et des relations entre les individus qui la composent. Le troisième élément est lié à la motivation donnée à ce type de vie humaine, en collectivité, utilisé en vue d'un intérêt général qui transcende les intérêts individuels.

La notion de *système* est en général utilisée pour caractériser un ensemble d'éléments [6] organisé ou structuré d'une certaine manière. En fait, ce qui est intéressant dans l'acception du terme système est lié à l'*analyse systémique*, comme analyse d'ensemble. L'analyse systémique [7] :

> [...] envisage les éléments [...] non pas isolément mais globalement, en tant que parties intégrantes d'un ensemble dont les différents composants sont dans une relation de dépendance réciproque.

En mettant ensemble les deux termes système et social, nous retrouvons en fait la notion de société. La clé de notre présentation se trouve dans leur association avec le terme artificiel - « qui est produit par une technique humaine, et non par la nature ».

5. cf. [Larousse, 1995]
6. En grecque *sustêma* signifie *ensemble*.
7. cf. [Larousse, 1995]

L'étude des SMA (comme exemple de système artificiel utilisé en informatique) a pour principal objectif de proposer des solutions à un problème résultant d'une activité coopérative entre plusieurs agents. Un élément majeur est donc de comprendre comment on arrive à faire se coordonner les agents afin de résoudre le problème, ou comment on impose un intérêt général (la résolution du problème) aux agents qui sont autonomes et par conséquent ont des intérêts spécifiques individuels. Comme notre objectif est de prendre pour base le mode de fonctionnement des systèmes naturels on étudie les similitudes entre les deux types de systèmes et on essaie d'identifier les éléments transférables vers les systèmes artificiels.

D'après les remarques faites sur les définitions ci-dessus et en employant la méthodologie générale présentée dans l'introduction, les caractéristiques des SSA sont :

1. les agents artificiels doivent être considérés dans un contexte global où l'action de chaque agent peut influencer les comportements des autres ;
2. les interactions entre les agents se font dans un cadre organisé (qu'on appelle organisation, agence ou institution) caractérisé par une structure bien définie ;
3. il y a toujours un but général (commun) qui transcende l'intérêt propre de chaque agent ;
4. il existe un ensemble de règles ou lois générales qui contraignent le comportement des agents afin d'atteindre le but général.

On remarque que en fin de compte tout SMA peut être considéré comme étant plus ou moins social. Tout dépend de la façon dont le système en cause présente ou non les quatre caractéristiques, la plus importante étant la dernière qui impose la présence explicite de lois sociales.

Comme nous l'avons déjà précisé, les résultats obtenus dans le domaine de la sociologie peuvent être une bonne source d'inspiration pour la construction des SMA. La sociologie essaie d'expliquer l'existence de ce qu'on appelle l'ordre social parmi les membres d'une collectivité qui sont plus ou moins libres (autonomes) et présentent des relations d'interdépendance. Comme la recherche sur les SMA a pour but de trouver des mécanismes améliorant l'efficacité des interactions, la sociologie ne peut que de lui apporter des solutions complémentaires.

Les travaux de[Moses et Tennenholtz, 1995] et [Shoham et Tennenholtz, 1995] ont été parmi les premiers à montrer clairement le caractère social des interactions entre agents et son rôle dans la conception des SMA. Dans ces articles, les auteurs mettent en évidence la différence entre le processus *off-line* de la conception d'un système et celui de son exécution *on-line*. Dans le cas de l'exécution on-line, les agents sont obligés de percevoir l'environnement, de prendre des décisions et d'agir en temps réel. Étant donné que souvent, dans la pratique, un agent ne dispose que de ressources limitées (temps, pouvoir de calcul, etc.) et que les problèmes qu'il doit gérer sont difficiles (en général de complexité NP), le concepteur d'un SMA doit intervenir dans la résolution du problème afin de compenser les limites de l'agent, par un processus d'analyse off-line. Dans le cadre de l'analyse off-line, on dispose de plus de temps et de ressources. On a donc la possibilité de résoudre un problème difficile ou de proposer des solutions pour éviter l'apparition des conflits. Une fois qu'une décision off-line a été prise (avant le commencement de l'activité on-line), les résultats sont soit de type logiciel, soit de type construction physique. Dans les deux cas, l'impact sur les agents va générer des contraintes.

La contrainte ainsi créée σ est appelée *loi sociale* et elle a pour rôle de restreindre l'ensemble d'actions possibles réalisables par un agent. Du point de vue du concepteur, une loi sociale va éliminer tous les ensembles d'actions ou leurs combinaisons connues comme étant négatives pour obtenir la fonctionnalité globale désirée. Du point de vue de l'agent, une loi sociale permet de faire plus facilement des prédictions sur le comportement des autres agents. Ainsi, un système social peut interdire à un invité à un anniversaire de manger tout le gâteau, laissant aux autres la possibilité, s'ils le souhaitent, d'en manger un morceau. Un exemple plus complexe proposé dans l'article cité traite de la convention de circuler à droite sur la route.

Formellement, Shoham et Tennenholtz utilisent des automates finis pour modéliser un agent. Un agent est capable d'exécuter un ensemble fini A d'actions. Le résultat d'une action a dans un état s est décrit par une fonction de transition $T(s,a)$. La loi sociale est définie comme une contrainte sur cette fonction de transition. On note S_σ le système S d'agents se trouvant sous l'influence de σ. S_σ est un *système normatif* si ses agents n'exécutent que des actions légales ou conformes à σ. Dans la vision de Moses, Shoham et Tennenholz, un système normatif doit être muni d'une structure interne garantissant la présence de comportement légaux. Par conséquence, ils introduisent le *système social* comme un cas particulier d'un système normatif, qui garantit que l'état dans lequel le système peut se trouver est un état considéré comme « socialement acceptable ». De plus, un système social doit garantir que les buts sociaux peuvent être atteints, c.a.d. que pour chaque but considéré comme « socialement acceptable », de tout agent appartenant à S_σ, il existe un plan qui *garantie* sa réalisation, qui est *efficace* et dont le calcul est *tractable* (i.e. complexité polynomiale).

Un premier élément résultant de la complexité des systèmes sociaux est le fait que le problème de la recherche d'une loi sociale σ (c.a.d. de trouver les contraintes à appliquer sur les actions possibles utilisées pour la réalisation d'un état « socialement acceptable »), si elle existe, est NP-complet. Le résultat n'est pas complètement négatif, parce que ce processus est réalisé off-line. De plus, dans certains cas on peut l'améliorer. Par exemple, prenons le cas particulier des agents *bornés* pour qui le nombre de transitions possibles pour changer leur état est limité par $O(log(n))$ où n représente le nombre total d'états d'un agent. Les restrictions suivantes seront nécessaires et suffisantes pour obtenir une complexité linéaire :

- le nombre d'états d'un agent à gérer par une loi sociale est limité par une constante c ;
- le plan (s'il existe) utilisé par un agent pour atteindre son but est déterministe ;
- le plan (s'il existe) doit être court.

Pourtant, dans le cas général des agents non-bornés, on n'a aucune structure sur l'ensemble d'états, d'actions ou de lois sociales. Pour garantir la complexité en temps linéaire, les conditions ci-dessus ne sont que suffisantes.

Une autre amélioration proposée dans [Shoham et Tennenholtz, 1995] concerne la modularité d'un agent. Intuitivement, on considère qu'un agent est composé de plusieurs éléments, et donc l'ensemble de ses états est le produit cartésien des ensembles d'états de chaque élément le composant. Par exemple, dans le cas de la représentation d'un robot, ces éléments peuvent décrire sa position, son orientation et la position de son bras.

Ainsi, une première restriction nécessaire pour garantir une complexité linéaire concerne la modularité des états, notamment, en supposant qu'il n'existe pas plus de $O(log(n))$ composantes, chacune avec un nombre constant d'états.[8]

La modularité des états implique la modularité des actions. Cela permet d'appliquer une deuxième restriction, c.a.d. de considérer un nombre réduit (une constante) de lois sociales révélateur du changement d'état d'une composante particulière. Les SSA contenant des agents dont la structure respecte les deux restrictions de modularité sont appellées des SSA *modulaires*. Le problème de la recherche d'une loi sociale pour les SSA modulaires est polynomial.

Le contexte où on obtient ce genre de résultat est, selon nous, très réducteur et rigide : des agents homogènes obéissant à des lois homogènes. Dans les conclusions de l'article [Shoham et Tennenholtz, 1995], les auteurs mettent en évidence cet aspect, en s'interrogeant sur ce qui se passe dans le cas des agents « tricheurs », où on est en présence d'une transgression des lois sociales.

Notons que le modèle de SSA proposé par Moshe, Shoham et Tennenholtz est principalement lié au processus de conception des SMA. Le contrôle centralisé est remplacé par des lois sociales. Cette approche, peut être considérée comme étant classique ou la plus couramment employée dans la pratique, car elle suppose d'enrégimenter les agents : les manifestations des agents obéissent complètement à la vision d'un concepteur humain, résultat d'une analyse *a priori*. Pourtant, dans les systèmes sociaux naturels, il existe d'autres types d'expressions, par exemple normatives, où les lois sociales sont plus libérales, leur but principal étant d'influencer les comportements des individus. Nous traiterons plus loin cette démarche, qui est en fait notre point de vue principal.

3.3 Action rationnelle et action sociale

En employant le terme agent on pense implicitement à quelqu'un qui agit. Par conséquent on caractérise les propriétés d'un agent par rapport à ses actions. Toujours dans le cadre de notre démarche, qui se bqse sur l'observation du monde réel, nous nous proposons d'examiner à present la notion d'action dans un contexte social.

8. n représente le nombre total d'états de l'agent.

3.3.1 Rationalité et action

Souvent, dans le monde SMA, on fait la différence entre les agents rationnels et les agents réactifs (cf. chapitre 2). Cette catégorisation est basée sur une théorie de l'action rationnelle qui explique la raison d'agir pour les premiers par la notion d'intention, i.e. [Bratman, 1987; Cohen et Levesque, 1990; Searle, 1983]. Cette approche s'inscrit dans une longue tradition de la philosophie de l'esprit qui se préoccupe de la relation entre la rationalité, la connaissance et l'action, étant donné que l'efficacité d'une action dépend de la qualité des prédictions rationnelles, et que la connaissance acquise dépend du succès de l'action. La rationalité est un concept difficile à définir. On se contente ici de rappeler deux définitions. D'après Platon, être rationnel c'est agir seulement en conformité avec ses connaissances ou avec sa vision et compréhension du monde. D'après Kant, la rationalité c'est avoir une raison pour agir.

Plus récemment, dans les études sur la relation action - rationalité, on constate l'apparition d'une typologie où les hypothèses sur les types d'actions sont différentes. La plus commune est l'action *téléologique*, par laquelle l'agent poursuit un but intervenant avec des moyens appropriés pour assurer l'apparition d'un état de choses propice, conformément au choix opéré entre plusieurs alternatives. Le modèle correspondant à ce type d'action se retrouve dans la théorie des jeux et de la décision. Il emploie la perspective utilitariste (de l'obtention d'un maximum d'utilité), et il s'avère efficace non seulement dans l'économie, mais aussi dans la recherche sociologique et la psychologie sociale. C'est le modèle le plus souvent utilisé en informatique, voire dans le domaine de l'IA et des SMA.

Le deuxième type concerne l'action *réglementée par des normes*. Dans ce cas, le modèle suppose l'existence d'un groupe social dont les membres acceptent certaines valeurs communes et réalisent des comportements désirables. La notion de rôle que nous étudierons plus en détail dans cette thèse est directement liée à ce modèle.

Le troisième type d'action est celui de l'action *auto-réflexive*. Dans ce cas, on suppose que les partenaires d'une action collective représentent les uns pour les autres une catégorie de public. Ce modèle, encore en discussion, est utilisé pour expliquer l'image (subjective) de nous-mêmes qu'on veut dévoiler aux autres (i.e. émotions, sentiments).

Dans la succession des modèles d'action présentés ci-dessus, on utilise des hypothèses ontologiques de plus en plus complexes : l'individu rapporté à l'univers, la collectivité humaine rapportée à l'univers social et la subjectivité de l'acteur (humain) rapportée à l'univers public. Cette typologie nécessite des critères de rationalité correspondants, et donc de plus en plus nuancés. Au moment de l'application d'un modèle « naturel » au monde artificiel, on est obligé de prendre en compte et d'adapter ces critères.

3.3.2 Action sociale

La manière la plus réductrice d'étudier l'action dans un contexte social est de la traiter comme étant la somme d'actes individuels de plusieurs agents. En réalité, le problème de l'action sociale est plus complexe. Etant donnéeVu l'existence de nombreux domaines (sciences) étudiant les comportements humains dans les situations et sous les aspects les plus variés possibles, l'introduction du concept d'action sociale dans les SMA oblige à étudier les résultats obtenus dans les différentes disciplines sociales correspondantes.

Castelfranchi et son groupe de recherche sont peut-être les plus actifs de ce point de vue, avertissant sur la complexité de cette démarche. Leur méthode de travail multidisciplinaire, à double objectif, considère l'IA comme une discipline dont le but est de comprendre les êtres intelligents (humains) en construisant des systèmes intelligents (artificiels). Ils s'inspirent donc des domaines sociaux comme la sociologie et la psychologie sociale pour créer des modèles applicables au monde des SMA, dont ils espèrent utiliser (par l'intermédiaire de la simulation) pour une meilleure compréhension du phénomène psychosocial. Parmi les recherches les plus intéressantes, on note des travaux liés à l'interaction sociale et à son aspect cognitif [Castelfranchi, 1995; Conte et Castelfranchi, 1995], aux normes dans les SMA [Conte *et al.*, 1999; Castelfranchi *et al.*, 1999], à la notion de confiance [Castelfranchi et Falcone, 1998] et à la délégation [Castelfranchi et Falcone, 1997]. Étant donné la richesse des sujets abordés, il est difficile de faire une synthèse exhaustive. Par conséquent, nous allons analyser quelques thèmes intéressants du point de vue de leur applicabilité au contexte de cette thèse.

Nous commençons par présenter les résultats obtenus à propos de l'*action sociale* (AS). En partant de la définition de l'AS telle qu'elle est utilisée dans l'IA et dans la philosophie, Castelfranchi montre que l'AS est le résultat émergent des actions individuelles et propose un modèle de l'action sociale individuelle [Castelfranchi, 1998].

On observe, dans les approches de Castelfranchi, la présence de l'hypothèse sur la rationalité des agents ainsi que d'autres paradigmes (i.e. réactivité, normativité). La condition de rationalité dans un contexte social est expliquée par le fait que l'agent doit être « conscient » qu'il se trouve dans un monde de relations causales et de dépendances : il dépend et il influence les actions des autres agents. Donc, dans un éventuel modèle de l'AS, un agent doit prendre en compte les croyances des autres agents ou leurs intentions. Dans le premier cas, il s'agit d'une AS faible, dans le deuxième, d'une AS forte. L'interférence et la dépendance d'un agent par rapport aux autres sont motivées, soit parce que l'agent veut bénéficier des actions des autres, soit parce qu'il veut éviter leur influence négative.

La manière dont un agent peut agir pour satisfaire ses buts suppose deux grands types de comportements :

- l'agent s'adapte aux autres (relation de dépendance) ;
- l'agent induit sur les autres un comportement désiré (relation d'influence ou de pouvoir).

Pour illustrer cela, Castelfranchi propose deux scénarios :

1. Un robot ou une personne s'arrête brusquement dans son déplacement pour éviter un obstacle sur son chemin. L'obstacle peut être soit une porte ouverte par le vent, soit une autre personne ou robot. Si, dans ce genre de cas, les autres agents sont traités comme s'ils étaient des objets, il n'y a rien de social dans leur attitude.
2. C'est le cas d'une action ordinaire qui devient sociale. On considère Adam, un agent dans le monde des cubes. Dans la situation initiale, on a le cube A sur le cube B. Si le but d'Adam est « le cube A sur la table » alors l'action correspondante n'est pas une AS. Supposons la présence d'un deuxième agent, Ève, qui a comme but « le petit cube a sur le cube B » et qui n'est pas capable de déplacer les gros cubes. Dans la situation initiale Ève est dépendante des actions d'Adam, car elle ne peut pas réaliser son but toute seule.

Dans le deuxième exemple, si on suppose qu'Adam connaît le but d'Ève et que son intention et d'aider Ève à réaliser son but, la même action exécutée quand il était seul devient AS. Cet exemple montre le cas d'une action faible. Si le but d'Ève est de faire induire à Adam le but de libérer le cube B pour qu'elle puisse ensuite mettre dessus le petit cube a, son action (par exemple communicative, où elle demande à Adam de déplacer le cube A) est aussi une AS, cette fois-ci de type forte.

Castelfranchi note très bien, par le biais de cet exemple, que la communication n'est pas une composante nécessaire de l'AS ou de l'interaction en général. Elle est seulement un instrument (utile). Les agents ne sont pas des agents (sociaux) parce qu'ils communiquent, ils communiquent parce qu'ils sont sociaux.

Une fois le concept de l'AS introduit, Castelfranchi l'utilise pour étudier les autres aspects de la vie sociale tels que la *coordination*. On se contente de rappeler au passage quelques aspects sur sa typologie :

– la coordination réactive, basée sur la perception d'un obstacle (ou d'une certaine situation) et la manifestation d'une réaction afférente ;
– la coordination par anticipation, liée à l'anticipation des interférences, basée sur un processus soit d'inférence, soit d'apprentissage.

Dans la coordination des agents, les principaux moyens employés à sa réalisation, sont la délégation et l'adoption de buts. On ne présente pas ici leurs principes et propriétés. Par contre, nous ferons quelques remarques sur les motivations qui conduisent les agents à se coordonner dans un contexte social avec des agents autonomes qui décident seuls du choix de leurs actions. Plus précisément, on essaie de répondre aux questions suivantes : comment modifie t-on l'opinion de l'autre agent, ou comment induit-on une croyance ou un comportement chez l'autre (dans le contexte où la communication ne suffit pas), en tenant compte de leurs liens de dépendance/influence et de pouvoir.

Les modèles utilisés pour expliquer ces motivations s'inscrivent dans la typologie de l'action rationnelle et normative présentée ci-dessus :

– un modèle d'agent rationnel qui aide une société à obtenir un maximum d'utilité globale par la maximisation de son utilité individuelle. Le moyen principal pour motiver un agent est de lui donner une prime dans le cas d'une action positive et/ou de le pénaliser dans le cas contraire ;
– un modèle d'agent normatif qui, en général, obéit aux normes émises par une autorité.

L'inconvénient du modèle rationnel est que si le système de pénalités fonctionne mal ou s'il est inexistant, il existe un risque de convergence rapide du comportement des agents vers une transgression des interdictions des actions considérées comme négatives. De ce point de vue, le modèle normatif semble plus attractif car on sait que les agents vont préférer, par construction, obéir aux normes.

Pour conclure, nous observons que la séparation stricte entre les deux types de modèles d'agents n'est pas obligatoire. Nous avons montré que la rationalité et la normativité ne sont pas mutuellement exclusives, en revanche, elles peuvent cohabiter au sein du même modèle. Ainsi, une solution intéressante serait de séparer l'ensemble des actions qu'un agent peut exécuter en actions pour lesquelles il peut décider (pour garder son autonomie) et actions auxquelles il doit se conformer (afin de rester qualifié comme étant normatif).

3.4 Comportement individuel vs. structures d'interaction

Concernant la conception des SMA, deux aspects sont fondamentaux. Le premier aspect concerne l'interaction entre agents, et plus précisément, la manière dont les agents obtiennent un résultat en commun ou affichent une propriété en tant que groupe et la nature du cadre ou des structures de cette interaction assurant un comportement perceptible seulement au niveau macro. Les concepts directement concernés et d'ailleurs les plus étudiés sont ceux de coordination et de coopération.

Le deuxième aspect se concentre plus sur la structure interne d'un agent : la manière dont un agent est construit pour qu'il soit capable d'afficher un comportement social, c.a.d. de collaborer et de vivre avec les autres.

En ce qui concerne l'interaction entre les agents, on peut envisager deux solutions majeures. La première solution met en avant l'individu ou l'agent ordinaire. Grâce à ses capacités d'observation, de compréhension, de décision, d'action et d'influence, il contraint les autres agents ou accepte les contraintes imposées par les autres afin d'obtenir le résultat global. C'est l'agent qui par son comportement aide à créer les structures de l'interaction. L'intelligence du système est la manifestation de l'intelligence individuelle multipliée ou renforcée grâce à ses membres. C'est la solution qui soulève le plus d'intérêt pour la technologie de construction de tels agents. La deuxième solution considère le contraire, c.a.d. que les structures de l'interaction précédent l'individu. Elles existent avant que l'agent soit créé et elles persistent après sa disparition. Si un agent veut entrer en interaction avec un autre, il doit tenir compte du fait que cette interaction est institutionnalisée et par conséquent, il doit se soumettre instantanément aux contraintes et aux conséquences que cela peut impliquer. Cette démarche accepte l'existence des lois sociales.

Malheureusement, les deux solutions qu'on a décrites ne donnent que des réponses partielles. La vraie réponse se trouve entre les deux. Cela veut dire qu'on ne peut pas dissocier le comportement d'un agent des structures que l'agent a créées par son comportement, et vice versa. Par exemple, dans le cas où le comportement individuel est plus important, les actions de l'agent vont influencer pour toujours le reste de l'interaction. C'est ainsi que les structures de l'interaction ont été créées, par une pratique et une acceptation sociale, par exemple, les comportements les plus efficaces ou les plus employés vont générer des standards et des règles de comportement, la signification de certains comportements ou des mots employés pour communiquer (la communication étant considerée comme une forme d'interaction) est de plus en plus la même pour plusieurs participants à l'interaction, etc. Les structures ainsi créées, même si elles n'existent pas physiquement, vont agir à leur tour de la même façon que si elles précédaient l'individu, avec la différence qu'elles ne sont pas figées pour toujours. Chaque interaction qui est ou non dans le contexte de la structure, augmente ou affaiblit sa valeur sociale. On arrive ainsi à désobéir aux normes et aux lois qu'on traite de révolues, par exemple à ignorer les protocoles ou les autorités, à « oublier » et changer le sens de certains mots, etc. Même s'il est difficile de parler des valeurs et des mécanismes associés, la conclusion est que les structures sociales qui contraignent le comportement d'un individu sont soumises à une validation dite sociale par le comportement individuel lui-même.

3.5 Structures sociales artificielles

Dans le domaine des SMA, plusieurs tentatives ont été faites pour introduire la notion de social et par conséquent de société (artificielle). Dans la suite, nous présentons quelques travaux qui font usage des termes à connotation sociale, tels que : groupe, équipe, société, agence, organisation ou institution.

3.5.1 Groupe

La notion de *groupe* a été déjà étudiée dans le domaine des systèmes distribués (SD). Le groupe est une abstraction permettant de réunir des entités (par exemple : processus, objets, machines) qui partagent un minimum de caractéristiques communes (propriétés ou fonctionnalités) sous une identification unique. Dans l'IAD (intelligence artificielle distribuée), la notion de groupe prend une autre forme, équivalente à un système multi-agents. Un groupe d'agents représente une collection d'agents définie et délimitée par le type d'interactions entre ses membres [Singh, 1991]. Les interactions entre les membres d'un groupe déterminent leurs rôles respectifs au sein du groupe. Ainsi, les membres d'un groupe peuvent avoir des comportements identiques, complémentaires ou différents et peuvent interagir et se coordonner pour résoudre un problème ou offrir un service. Tous les aspects liés à la coordination ou à la résolution des problèmes concernent aussi la problématique liée au groupe.

Ferber et Gutknecht [Ferber et Gutknecht, 1998] proposent une architecture basée sur le modèle agent - groupe - rôle (AALAADIN). Ils décrivent une organisation d'agents comme étant une relation structurelle entre des collections d'agents (ou groupes). La structure est définie de bas en haut par les rôles que les agents peuvent jouer au sein du groupe et par la façon dont ils forment les groupes. Pour définir cette structure organisationnelle, ils définissent d'abord la structure du groupe. La structure du groupe est définie par le triplet :

$$S = \{R,G,L\}$$

où R représente un ensemble de symboles de rôles, G un graphe orienté étiqueté qui spécifie les interactions valides entre deux rôles, la direction de l'arête correspondant au rôle qui commence l'interaction, et L signifie le langage d'interaction, c.a.d. le langage formel utilisé pour décrire la manière dont cette interaction est réalisée.

La structure organisationnelle est définie par le couple :

$$O = \{S,Rep\}$$

où S représente la structure du groupe et Rep exprime la structure représentative de l'organisation. La structure représentative est définie entre deux groupes comme étant l'ensemble d'agents jouant des rôles au sein de deux groupes en même temps, sachant qu'un rôle est défini à l'intérieur d'un groupe et qu'un agent peut jouer plusieurs rôles ou faire partie de plusieurs groupes à la fois.

L'acquisition d'un rôle par un agent est décrite par une fonction d'acceptation. Quelques exemples de telles fonctions :

- l'acceptation ou le refus systématique ;
- l'acceptation conditionnelle conformément au statut du groupe (par exemple, il existe un nombre limité de rôles dans un groupe) ;
- l'acceptation à base de compétences, etc.

Une observation intéressante à propos de l'application de la notion de groupe : on remarque l'utilisation des solutions proposées par les SMA dans l'organisation du groupe dans les systèmes distribués. Parmi ces travaux, on peut mentionner, par exemple, Malville qui dans [Malville, 1999] présente un mécanisme d'auto-organisation des groupes pour l'allocation des tâches et son application aux services de groupe CORBA [9].

9. CORBA [Object Management Group, 2001] est le standard d'interopérabilité dans les systèmes distribués.

3.5.2 Équipe

La notion d'*équipe* est souvent employée pour décrire un ensemble d'agents qui ont un but commun. En général, on considère que le but est imposé et on ignore donc si les agents décident eux-mêmes de participer ou pas à l'effort collectif. Les membres d'une équipe ont un comportement bénévole, ils coopèrent et aident les autres agents pour atteindre le but commun.

3.5.3 Société

Cavedon et Sonnenberg [Cavedon et Sonenberg, 1998] essaient de définir une société d'agents à l'aide des rôles, des relations et des types de relations. Ils proposent un modèle qui utilise un langage composé des ensembles suivants : $Agent$, $Role$, $Reln$ (pour décrire les relations entre agents) et $RelType$ (pour les types de relations). Un rôle est par défaut associé à un type de relation. Par exemple, la relation directeur de thèse - étudiant implique la présence de deux rôles, directeur de thèse et thésard. Une fois le rôle défini pour un type de relation, il ne peut plus être utilisé dans la définition d'une autre relation. Ce choix est dû au fait qu'on veut attribuer un rôle à une seule équipe formée pour résoudre un problème particulier. Par exemple, on ne peut pas avoir une relation de type directeur de thèse - rapporteur, car le rôle directeur de thèse a déjà été employé. On fait donc l'hypothèse suivante :

$$\forall \alpha, R_1, R_2, Role\_of(\alpha, R_1) \wedge Role\_of(\alpha, R_2) \Rightarrow R_1 = R_2$$

où α est un rôle, R_1 et R_2 sont des types de relations, et $Role\_of(\alpha, R)$ le prédicat qui associe un rôle à un type de relation.

Pour décrire le fait qu'un agent a joue un certain rôle α dans une relation de type r on utilise le prédicat $In(a, \alpha, r)$. Par exemple, le fait qu'Alice est la directrice de Bob et que Charles est le directeur de Daniel, peut être décrit de la manière suivante :

$$In(alice, directeur, r_1)$$
$$In(bob, thesard, r_1)$$
$$In(charles, directeur, r_2)$$
$$In(daniel, thesard, r_2)$$

L'interaction sociale est décrite par l'intermédiaire de l'engagement social [Castelfranchi, 1995] : on attribue des buts à un rôle et au moment où un agent accepte de jouer ce rôle il s'engage à réaliser tous les buts qui lui sont attribués. Le prédicat $RoleGoal(\alpha, g)$ décrit le but g attaché au rôle α et il intervient pour décrire la propriété ci-dessus exprimée par l'axiome :

$$RoleGoal(\alpha, g) \wedge In(a, \alpha, r) \Rightarrow GOAL(a, g)$$

Afin d'éviter les éventuels conflits entre les buts provenant de différents types de relations, les auteurs utilisent une relation d'ordre entre les buts provenant de rôles différents. On utilise pour cela le prédicat $Influence(a, < \alpha{,}r_1 > , < \beta{,}r_2 >)$ qui exprime que le rôle α dans la relation r_1 est plus influent que le rôle β dans la relation r_2, α et β étant des rôles que a peut jouer. Par exemple, supposons que Charles a un autre étudiant, Éric, qui est sur le point de publier un article. Le fait que, Charles préfère s'occuper plus d'Éric que de Daniel, s'écrit :

$$Influence(charles, < directeur{,}r_3 > , < directeur{,}r_2 >)$$

Dans [Panzarasa *et al.*, 1999], les auteurs étendent le modèle de Cavedon et Sonnenberg. Au moment de l'acquisition d'un rôle, l'agent n'adopte pas seulement les buts mais aussi les autres attitudes cognitives comme les croyances et les désirs. Par exemple, pour montrer comment un rôle influence l'attitude de l'agent qui le joue, les auteurs proposent l'utilisation du prédicat $Infl(Att(a{,}\phi){,}\alpha)$ qui exprime le fait que si l'agent a joue le rôle α, il adopte l'attitude mentale Att, qui peut être une croyance, une intention ou un désir par rapport à ϕ.

3.6 Rôles

La notion de rôle, comme on l'a déjà vu, est intimement liée au concept d'organisation. Il existe beaucoup de théories qui se sont développées autour de ces concepts. Les premiers travaux sont apparus dans le cadre des sciences sociales, sous le nom de la théorie des rôles. Selon cette théorie, dans une société, les individus sont supposés jouer certains rôles qui donnent des droits et imposent des devoirs dans la société [Biddle, 1979]. En plus de son aspect normatif, la théorie des rôles s'intéresse à la formalisation de ces concepts, à l'analyse des conflits et aux rapports avec la psychologie humaine.

D'autres travaux ont été réalisés dans le cadre de l'analyse de la gestion d'une entreprise, plus précisément ceux qui étudient les conditions de distribution des responsabilités dans une entreprise. Par exemple, Malone [Malone, 1987] décrit plusieurs types d'activités organisationnelles en termes de tâche et de processeurs (les processeurs exécutent des tâches). En fonction du mode de répartition des tâches entre processeurs, on parle d'une organisation ayant une hiérarchie de produit ou d'une hiérarchie fonctionnelle (voir la figure 3.1). Quand on s'intéresse à la décentralisation du processus de décision, on distingue les organisations de type marché de celles de type contrôle hiérarchique. Même si Malone n'utilise pas explicitement le terme de rôle, il met en évidence le besoin de structurer les activités des entités et d'identifier les rapports de dépendances entre celles-ci (par exemple : les relations de producteur/consommateur dans les structures de type marché ou contrôle hiérarchique).

Plus récemment, l'utilisation du concept de rôle s'est imposé assez naturellement en informatique. Souvent, les applications logicielles doivent être construites en tenant compte de la structure de l'or-

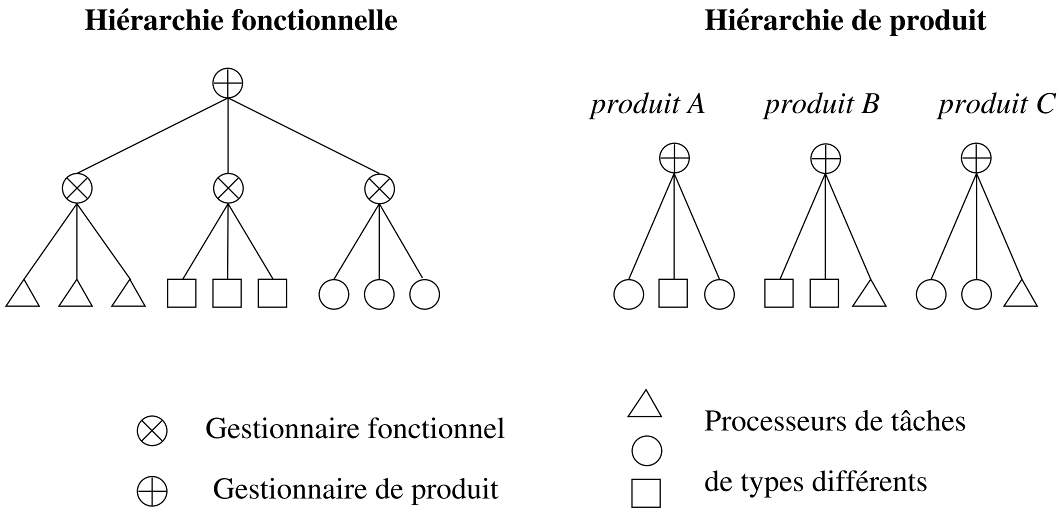


FIG. 3.1 – *Exemples de hiérarchies*

ganisation dans laquelle elles seront utilisées. Le rôle permet donc de tirer parti de l'existence de cette structure en offrant les atouts suivants :

– d'une part l'organisation elle-même fournit une base de départ et peut suggérer une structure et un certain nombre de règles ;
– d'autre part on peut décrire des patterns de comportement génériques en s'inspirant du fonctionnement de l'organisation.

Dans la suite, nous allons présenter quelques travaux qui appliquent la notion de rôle dans des domaines comme les systèmes distribués, la sécurité et les SMA.

3.6.1 Management des systèmes distribués à base de rôles

Dans le domaine de la gestion des systèmes distribués on remarque l'existence de plusieurs travaux qui utilisent les rôles. Lupu et Sloman [Lupu et Sloman, 1997; Lupu, 1998] proposent une telle approche pour le management des systèmes distribués. En respectant la théorie des rôles, ils définissent le rôle comme étant un ensemble de droits et de devoirs que les managers peuvent avoir dans un système. Les devoirs sont décrits par des politiques positives d'obligations et les droits par des politiques (positives ou négatives) d'autorisations. Les politiques négatives d'obligations spécifient les actions qu'un manager doit s'abstenir de faire. Les politiques établissent une relation entre les *sujets* exécutant des opérations au sein du système et les *objets cibles* sur lesquels on exécute ces opérations. Les sujets et les objets sont décrits par des *domaines*, une façon de regrouper les différents éléments du système sous un seul nom. L'avantage de l'utilisation des domaines en politiques est que les objets ou les sujets peuvent être ajoutés à un domaine ou en être retirés sans avoir à modifier la politique explicitement. Voici quelques exemples de politiques et leur description formelle :

- `A+ @/personnel/nurses {administer(analgesics)}`
  `x:@/patients/lung-diseases`
  `when (x.temperature > 37) && (x.temperature < 38.5)`
  Il s'agit d'une autorisation positive `A+` qui donne la permission aux infirmières `@/personnel/nurses` d'administrer des analgésiques `administer(analgesics)` aux patients d'un certain type `@/patients/lung-diseases` dont la température corporelle est entre 37°C et 38,5°C.
- `O+ on drugs_administered @/personnel/nurses {update} /drugs_db`
  Il s'agit d'une obligation positive `O+`, créée par l'événement `drugs_administered` indiquant que les médicaments ont été administrés, et qui oblige les infirmières à actualiser la base de données de médicaments `/drugs_db`.

L'approche proposée par Lupu et l'équipe de systèmes distribués à l'Imperial College trouve beaucoup d'applications dans le domaine de l'administration des réseaux et de la sécurité des systèmes (pour la description de politiques de sécurité).

3.6.2 Contrôle d'accès à base de rôles - RBAC

Sandhu utilise, lui aussi, la notion de rôle afin de spécifier les politiques de sécurité dans un système informatique, notamment pour contrôler l'accès d'un utilisateur humain aux ressources d'un système. Sandhu [Sandhu *et al.*, 1996] propose le modèle RBAC (Role-Based Acces Control) où, au lieu d'assigner des privilèges (permissions et interdictions) aux utilisateurs individuels, on les attribue aux rôles. Les utilisateurs peuvent acquérir les permissions par le rôle que leur est attribué.

Le modèle RBAC propose d'autres concepts tels que les sessions, la hiérarchie de rôles et les contraintes. Les *sessions* sont nécessaires pour permettre aux utilisateurs d'utiliser le système. On associe une session à un seul utilisateur qui reste constant pour toute la durée de vie de la session. Les rôles peuvent être organisés de manière à former une *hiérarchie de rôles*. Les hiérarchies de rôles permettent de raffiner progressivement les différentes permissions attribuées à chaque rôle en structurant ou héritant la description. Ainsi, si le rôle « chargé de portefeuille bancaire » est inclus dans le rôle « employé de banque », le rôle « chargé de portefeuille bancaire » héritera des privilèges associés au rôle « employé de banque » (comme la possibilité de réaliser des opérations courantes au guichet). Les privilèges issus de l'héritage pourront être complétés par des privilèges spécifiques. Par exemple, le rôle « chargé de portefeuille bancaire » pourra également disposer du privilège d'accorder des prêts à la consommation pour de faibles montants. Enfin, les *contraintes* permettent d'imposer sur ce modèle des propriétés nécessaires à la réalisation des objectifs de sécurité, comme l'exclusion mutuelle de deux rôles utilisée aux fins de la séparation des pouvoirs au sein d'une organisation.

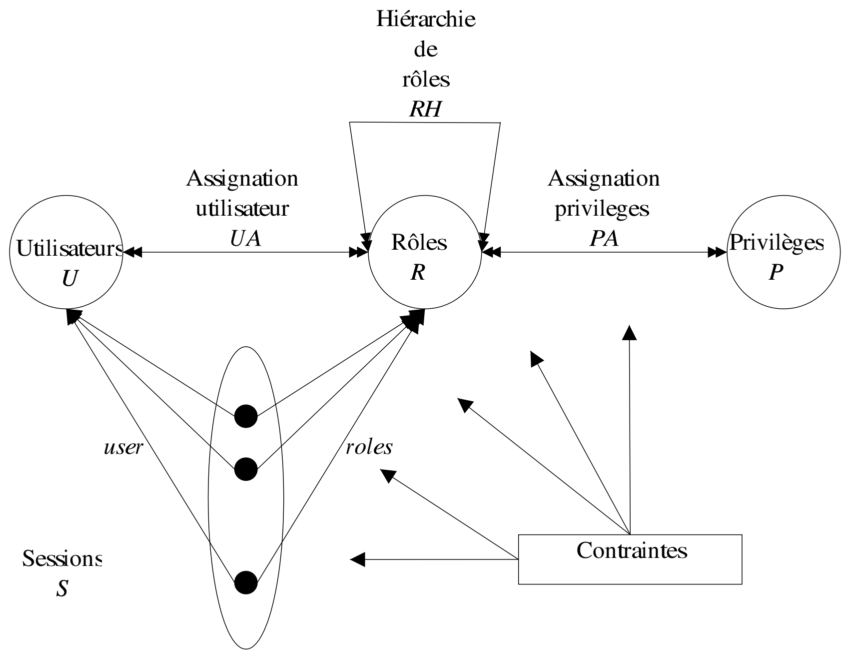


FIG. 3.2 – *Le modèle RBAC*

L'apparition de cette approche a généré de nombreuses études. Par exemple, on s'intéresse aux problèmes liés à l'assignation de permissions aux rôles, ou à la définition de hiérarchies et de rôles. On se contentera ici de décrire brièvement la famille des modèles proposée par Sandhu et ses caractéristiques :

- U,R,P et S, respectivement des ensembles d'utilisateurs, de rôles, de privilèges et de sessions.
- $PA \subseteq R \times P$, une relation associant un privilège à un rôle ;
- $UA \subseteq U \times R$, une relation associant un ou plusieurs rôles à un utilisateur ;
- $RH \subseteq R \times R$, une hiérarchie de rôles partiellement ordonnée (la relation d'ordre étant notée $\succeq$) ;
- $user : S \rightarrow U$, une fonction associant chaque session s_i à un seul utilisateur $user(s_i)$, qui reste constant pour toute la durée de vie de la session ;
- $roles : S \rightarrow 2^R$, une fonction associant chaque session s_i à un ensemble de rôles ;
- et une collection de contraintes qui détermine si certains éléments du modèle RBAC sont acceptables (seuls les éléments acceptables étant effectivement intégrés dans le modèle).

Nous présentons quatre modèles différents, de RBAC$_0$ à RBAC$_3$. Dans le modèle RBAC$_0$, on considère seulement les relations UA et PA. Le modèle RBAC$_1$ inclut RBAC$_0$, auquel on ajoute la capacité de définir des hiérarchies de rôles héritant des privilèges. Le modèle RBAC$_2$ inclut le modèle RBAC$_0$ plus la capacité de définir des contraintes. Le modèle RBAC$_3$ est le modèle qui regroupe toutes les caractéristiques des modèles RBAC$_0$-RBAC$_2$.

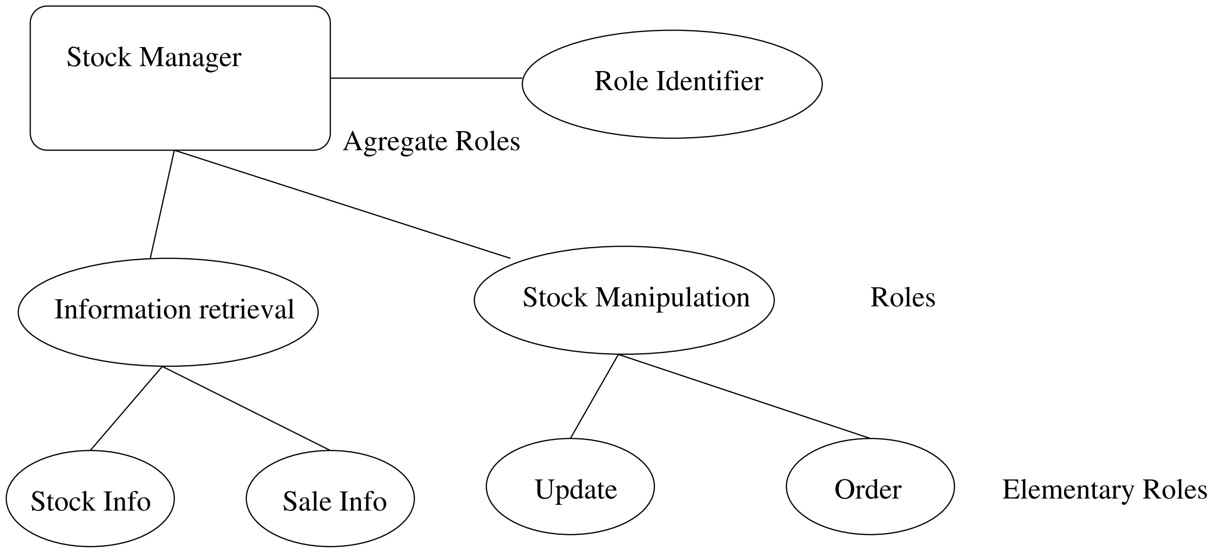


FIG. 3.3 – *APRIL : Agrégation et assignation des rôles*

L'inclusion dans le modèle RBAC des notions de privilège et de contrainte permet d'adapter cette approche à toutes les politiques de sécurité. En ce sens, le modèle RBAC est neutre, c.a.d. indépendant de la politique de sécurité.

3.6.3 Rôles dans les SMA - APRIL

La notion de rôle est déjà présente dans plusieurs théories et plates-formes multi-agents [Cavedon et Sonenberg, 1998; Ferber et Gutknecht, 1998]. Dans ce paragraphe nous présentons l'un des travaux liés aux rôles les plus complets [Skarmeas, 1995] qui traite des aspects concernant la modélisation, l'assignation, l'implémentation et les relations entre rôles. Ce modèle a été spécialement conçu pour la plate-forme APRIL [McCabe et Clark, 1995]. Les agents APRIL sont des objets actifs, capables d'exécuter des tâches et de communiquer par messages. Les rôles sont utilisés afin de structurer et de grouper les tâches dont les agents (humains ou logiciels) sont responsables. Les rôles peuvent être décrits par la hiérarchie suivante (voir également l'exemple de la figure 3.3) :

– élémentaires : composés de collections de tâches complexes ;
– rôles : collections de rôles élémentaires ;
– rôles agrégés : collections de rôles, qui correspondent aux différentes positions sociales au sein de l'organisation.

L'attribution des rôles agrégés à l'utilisateur humain est réalisée en attachant un identificateur de rôle à un rôle agrégé. L'identificateur de rôle représente un rôle ordinaire, par exemple, un agent représentant l'utilisateur dans le système.

Les relations de hiérarchie entre rôles sont spécifiées comme faisant partie de la définition du rôle. La spécification hiérarchique est définie par l'intermédiaire d'un contrat, qui décrit les tâches que les parties contractuelles doivent réaliser. Les contrats à leur tour peuvent être composés hiérarchiquement, par exemple, un contrat entre rôles agrégés peut se décomposer en sous-contrats entre sous-rôles.

L'implémentation des rôles est dépendante de la plate-forme APRIL. Le code concernant les rôles, les contrats et d'autres processus organisationnels est stocké dans une base de données *organisationnelle*. Ainsi, les rôles peuvent être ajoutés, effacés, lus et déplacés. Les rôles sont implémentés comme des processus APRIL communiquant par l'envoi de messages. L'exécution d'une tâche spécifique à un rôle est réalisée par l'envoi du message « execute » au rôle correspondant. Les capacités d'un rôle sont décrites dans un *Skill Server* qui assure la cohérence de la description en cas de changement (dans la spécification) d'un rôle, et qui propose un service de courtier afin de trouver les rôles offrant une certaine caractéristique.

Pour répondre à d'autres besoins organisationnels, il existe d'autres types de rôles tels que :

Virtual Roles - utilisés pour modéliser la délégation d'un rôle d'un niveau supérieur à un niveau inférieur, au sein d'une hiérarchie.

Group Roles - utilisés dans la modélisation des comités. Plusieurs rôles (correspondant aux diverses positions dans un comité : président, secrétaire, etc.) peuvent être ainsi regroupés dans un rôle agrégé. Ce type de rôle distribue la responsabilité au sein du groupe, car toute tâche attribuée au comité est redistribuée à l'un de ses membres. Il n'existe pas de notion de tâche commune.

Gateway Roles - contient des tâches pour filtrer l'information (par exemple, des rapports véhicules entre différents niveaux organisationnels) ;

Liaison Roles - sont utilisés pour assurer l'échange d'informations entre deux groupes qui collaborent.

Collective Roles - expriment des classes de rôles (agrégés) qui peuvent être instanciées en fonction d'un contexte spécifique. Par exemple, le rôle professeur assure la tâche d'enseigner, mais on ne précise pas quoi exactement et ni à qui.

3.6.4 Conclusions sur les rôles

Même si le concept de rôle est souvent utilisé de manière *ad-hoc*, on considère le rôle comme une abstraction utile pour grouper les fonctionnalités ou les propriétés génériques. Lupu dans [Lupu, 1998] insiste sur le fait que l'utilisation du terme *rôle*, et plus précisément l'expression *jouer un rôle*, peut avoir deux significations différentes en fonction du contexte dans lequel elle est utilisée :

- un ensemble de fonctionnalités ;
- un participant à une relation, association ou communauté.

Les deux définitions supposent que toute théorie ou modélisation des rôles prenne en compte :

- les obligations qui spécifient les devoirs des agents (ou managers, utilisateurs, processus, etc.) et les tâches qu'ils doivent réaliser ;
- les autorisations qui spécifient les droits que les agents vont acquérir grâce aux rôles qui leur sont assignés ;
- les relations entre les rôles exprimant les différents patterns organisationnels et types de collaboration (et les protocoles d'interaction) ;
- l'assignation des rôles aux agents ;
- l'agrégation des rôles ; le traitement des conflits et des redondances ;
- la spécification et l'utilisation des compétences des agents assignés à un rôle ;
- la cohérence interne de la spécification du rôle (par exemple : l'absence des conflits) ;
- la cohérence externe des rôles en terme de : séparation des devoirs entre rôles, rôles mutuellement exclusifs, contraintes sur l'assignation des rôles aux agents et sur la participation des rôles à une relation ;
- la modélisation des responsabilités communes ;
- l'implémentation et le déploiement des sessions (utilisateur) afin de donner la possibilité de combiner les rôles ou au contraire de séparer les contextes entre rôles.

3.7 Conclusions

Afin de décrire l'interaction entre plusieurs agents, on fait souvent appel à des expériences et à des connaissances déjà acquises dans des domaines tels que la philosophie, la sociologie, l'économie ou même dans des sous-domaines de l'informatique tels que les systèmes distribués, la sécurité, etc. Nous avons appris que l'interaction ne se réalise pas de manière complètement aléatoire, mais qu'elle possède une certaine structure. Les travaux que nous avons analysés dans ce chapitre utilisent des métaphores sociales pour décrire cette structure. Comme l'utilisation de normes est considerée comme un cas particulier de la métaphore sociale, nous reprendrons certains concepts qui nous aideront à préparer le terrain pour décrire l'interaction normative. Nous les énumérons dans la suite comme suit :

- l'organisation sociale : c'est une abstraction qui apporte une structure permettant une meilleure identification des types de relations (dépendance, pouvoir, autorité, etc.) existant entre agents et une explication plus précise de leurs positions dans ces relations. Comme nous l'avons montré dans la section 3.4, l'utilisation des normes suppose l'existence d'une structure organisationnelle. Une norme est en dernière instance l'expression d'une autorité dont la légitimité n'a de sens que par la présence de cette structure. Nous préférons le modèle où les rôles sont définis à partir de l'organisation et non l'inverse, c.a.d. où l'organisation est définie comme étant une collection de rôles (cf. section 3.5).

– le groupe : c'est une abstraction permettant de réunir des agents qui ont en commun certaines caractéristiques ;
– le rôle : c'est une abstraction qui permet d'une part de grouper des fonctionnalités et propriétés génériques et d'autre part de limiter la sphère d'influence d'un agent par rapport à l'autre. Comme les normes sont à leur tour génériques par rapport à leur cible (par exemple, dans le texte d'une loi, on ne spécifie pas le nom d'une personne concrètement, mais sa fonction juridique : le président de la République) la notion de rôle est utile (même nécessaire) afin de les décrire.

En ce qui concerne le type de modèle d'agent idéal utilisé pour la coordination, notre choix repose sur un modèle mixte, c.a.d. rationnel et normatif. Avant d'entrer dans les détails de construction des agents qui reflètent la métaphore normative, nous allons traiter les questions liées à la définition et à l'utilisation du concept de norme. C'est le sujet du chapitre suivant.

Chapitre 4

Aspects normatifs

La vertu, mon cher étudiant, ne se scinde pas :
elle est ou n'est pas. On nous parle de faire
pénitence de nos fautes. Encore un joli système que
celui en vertu duquel on est quitte d'un crime
avec un acte de contrition !
Balzac (*Le père Goriot*)

4.1 Introduction

Dans ce chapitre nous abordons les aspects liés à la *normativité* qui décrit « *un état habituel, conforme à la règle établie* »[10]. Le concept de normativité est très complexe et donc son utilisation dans l'informatique en général (et dans les SMA en particulier) reste un sujet controversé. Ainsi, on peut se demander à quoi servent les normes dans la conception des SMA puisqu'on peut créer des agents qui par construction ont un comportement conforme à la norme. Ce cas est un exemple typique d'*enrégimenter* [Jones et Sergot, 1993] où les normes sont décrites par des lois sociales qui gouvernent exhaustivement les agents comme dans [Shoham et Tennenholtz, 1995]. Dans cette thèse, nous abordons une autre approche, celle de la réglementation normative. Pour nous, les normes représentent seulement des indications données à un agent afin qu'il puisse atteindre un état désirable ou qu'il se comporte d'une certaine façon. Les agents sont autonomes et donc ont le pouvoir de prendre des décisions qui les concernent : s'ils acceptent la norme en principe et plus concrètement, s'ils y obéissent dans toutes les circonstances.

Afin de mieux comprendre le sens que nous donnons à la notion de norme, nous proposons de remonter à l'origine et de voir comment elle est définie et utilisée dans des domaines tels que les sciences sociales, le domaine juridique et même informatique. On remarque la diversité des concepts liés à cette notion et la difficulté de l'identifier de manière précise.

10. cf. [Larousse, 1995]

4.2 Normes et actions

La notion de norme est intimement liée à la notion d'action, car les effets des normes ont un impact direct sur le comportement. Ainsi, dans la philosophie ou plutôt dans la philosophie du droit, on essaie d'expliquer le rapport qui existe entre action et norme. Par exemple, von Wright considère en priorité les normes. Tout ordre (ou système) normatif suppose un but ou une valeur, les normes provenant en dernière instance d'une autorité. Selon lui [von Wright, 1993], il existe trois types de normes qui correspondent aux modalités de conditionnement relatif à :

– un ordre normatif quelconque (obéir aux règles ou lois de l'Etat, de l'église, de la prison, de l'école, etc.) ;
– un but d'un agent (avoir des amis, obtenir un diplôme, ouvrir une porte, arriver sur Mars, etc.) ;
– une catégorie d'individus dans laquelle un agent peut s'inscrire par ses activités (être à la mode, être un « pro », être « politiquement correct », etc.).

Cette typologie montre que toute action a une explication normative. On agit parce qu'il y a une « pression » normative aux origines diverses. Même si von Wright ne mentione pas le concept de rationalité, on remarque que la rationalité est « cachée » dans la normativité. Pourtant, cette catégorisation est un peu obsolète, car on la retrouve transfigurée dans une typologie d'actions rationnelles plus récente qu'on a évoquée dans la section 3.3.1 et qui explique les actions d'un agent comme étant des actions :

– normatives - l'agent agit parce qu'il accepte les valeurs communes d'un groupe social dont il est membre ;
– téléologiques - l'agent agit pour réaliser son but ;
– auto-réflexives - l'agent agit dans le sens de « travailler son image » qu'il veut montrer de façon délibérée aux autres.

On préfère cette catégorisation, parce qu'elle permet de faire une meilleure distinction entre les concepts de rationalité et de normativité lorsqu'on explique les comportements des agents.

4.3 Normes dans les sciences sociales

Dans les théories sociales, l'utilisation des normes se réfère aux cas où les membres d'une communauté attendent et demandent la manifestation d'un certain type de comportement. Tuomela [Tuomela, 1995] sépare les normes en : règles (r-normes) et normes sociales proprement dites (s-normes). Les règles sont des normes créées dans une société basée sur la notion d'autorité et d'acceptation (*agreement*). Elles sont souvent associées à des sanctions (formelles ou informelles). Les normes proprement dites sont basées sur des croyances et des acceptations mutuelles. Elles concernent les conventions qui s'installent entre les membres d'une société ou d'un groupe spécifique (ex. les lois sociales et économiques dans la société humaine). Les sanctions pour les deux types de normes sont prises par les autres membres et peuvent aller jusqu'à l'élimination du groupe. A part ces types de normes, Tuomela décrit l'existence de normes potentiellement sociales. Ces sont des normes auxquelles on obéit non pas pour répondre à une attente sociale, mais plutôt pour des raisons personnelles. Dans cette catégorie, on inclut les normes morales (m-normes) et les normes de prudence (p-normes). En résumé, les raisons pour lesquelles on obéit aux normes sont directement liées à leur type :

– les r-normes parce qu'on les accepte (il y a un accord) ;
– les s-normes parce que les autres attendent qu'on s'y conforme ;
– les m-normes à cause de la conscience individuelle ;
– les p-normes parce que c'est une chose rationnelle à faire.

L'influence des normes sur les actions d'un individu dépend de la façon dont elles sont prises en compte par l'individu en cause (acceptation).

4.4 Normes dans les sciences juridiques - positions normatives

Dans les sciences juridiques, la notion de norme décrit les droits et les obligations des individus, conformément aux rôles qu'ils jouent au sein d'une société. De ce point de vue, la définition d'une norme est similaire à celle d'une r-norme dans la terminologie de [Tuomela, 1995]. La différence consiste dans la façon d'utiliser les sanctions formelles. Si les principales théories dans le domaine juridique sont en général d'accord avec cette définition, elles divergent lorsqu'il s'agit d'expliquer leur rapport avec la rationalité, par exemple, pour déterminer pourquoi les agents acceptent et obéissent aux normes.

La formalisation des concepts composant une norme est basée en grande partie sur la logique déontique qui fournit également des mécanismes de raisonnement adaptés (voir le chapitre 6 pour une présentation plus détaillée). Le raisonnement déontique a comme objet les normes, c.a.d. qu'on essaie de déduire un ensemble de normes en partant d'un autre ensemble qu'on connaît (par exemple, les droits et les obligations valides à un certain moment) et des propriétés générales existant entre ces normes (par exemple, l'interdiction est le concept dual de la permission).

L'activité de formalisation des concepts juridiques connaît une longue tradition, qui a commencé au début du siècle dernier par les efforts de Hohfeld [Hohfeld, 1913] pour construire une théorie des droits. Cette théorie décrit ce qu'il appelle les notions juridiques fondamentales (*droits*) : droit d'exiger, devoir, sans droit, privilège ou liberté, pouvoir, responsabilité, incapacité et immunité. Le but de cette théorie était d'une part de désambiguïser les notions juridiques distinctes et d'autre part de proposer des solutions basées sur un raisonnement juridique précis. Ainsi, Hohfeld donne une vision unitaire de l'interprétation des propositions concernant les droits. Pour lui, les droits sont des relations juridiques entre le porteur et la contrepartie du droit. Par exemple, le droit d'exiger quelque chose d'autrui est décrit comme une relation entre deux individus i et j et un état de choses F : $Droit(i,j,F)$.

Les droits peuvent être classés en deux groupes : déontiques (droit, liberté, sans droit, devoir) et normatifs (pouvoir, immunité, incapacité, responsabilité). Le premier groupe décrit l'aspect factuel des droits. Afin d'illustrer l'utilisation de la relation $Droit(i,j,F)$, Hohfeld considère le cas d'une personne exigeant d'une autre de ne pas entrer sur son terrain. Le fait F, de rester en dehors du terrain de quelqu'un, est interprété comme un état de choses factuel. Quant au deuxième groupe il s'agit des relations concernant l'état normatif. Ainsi, dans la relation $Power(i,j,F)$, F est un état normatif. La relation peut s'utiliser pour exprimer des propositions de type : « i a le pouvoir de créer différentes obligations contractuelles sur j ».

Malheureusement, à cette époque, Hohfeld ne disposait pas des outils d'analyses dont disposent aujourd'hui les logiciens, pour formaliser sa théorie. Kanger [Kanger, 1971] et puis Lindhall dans [Lindahl, 1977] ont tenté de construire une formalisation logique de la théorie de Hohfeld, connue de nos jours sous le nom de la théorie des *positions normatives*. L'avantage de cette analyse des droits est qu'elle permet d'obtenir une spécification précise et exhaustive de toutes les relations juridiques existant entre deux personnes, à partir d'un ensemble incomplet et imprécis de besoins (voir le chapitre 6 pour plus de détails sur les formalismes existants).

En conclusion, l'utilisation des normes dans les SMA peut bénéficier des résultats obtenus dans l'analyse des droits, une activité du domaine juridique, mais qui s'avère utile aussi aux applications informatiques. On remarque qu'avec le développement de l'Internet, on se pose de plus en plus des problèmes d'ordre juridique ou de sécurité liés aux interactions « électroniques » entre individus. Imaginons des échanges dans ce monde, où les humains seraient représentés par leurs agents logiciels. On délèguerait des droits et on serait responsable pour les actions de nos agents. Donc il existe encore un bon nombre de concepts juridiques qui peuvent être transférés et adaptés à ce monde. Nous citons les travaux de Krogh [Krogh, 1996b; Krogh, 1996a] qui présentent un scénario suggestif d'application à l'Internet, où on peut décrire les relations normatives entre deux agents : un agent qui veut accéder aux ressources d'un site et qui paye pour cela, et l'agent administrateur du site, qui donne des droits d'accès limité aux ressources. Krogh propose deux analyses : une analyse *ad hoc*, qui montre combien il est difficile de décrire les relations normatives d'une façon intuitive et qui finalement s'avère incomplète ; et une analyse correcte et complète basée sur la théorie des positions normatives, qui consiste à choisir parmi tous les cas possibles ceux qui correspondent ou décrivent le mieux le problème.

4.5 Normes en informatique

Le concept de norme en informatique est plutôt utilisé sous son aspect téléologique : une norme indique les moyens pour atteindre un but. De plus, la conformité aux normes est obligatoire. On trouve souvent les avatars d'une norme dans des concepts comme ceux des règle de comportement, protocole, loi ou politique. Les normes sont utilisées dans la description des règles ou politiques de sécurité, dans le management des systèmes distribués, ou pour contrôler les activités de coordination des agents.

4.5.1 Politiques de sécurité

Assurer la sécurité d'un système d'informations consiste à garantir le maintien d'un certain nombre de propriétés telles que la confidentialité, l'intégrité et la disponibilité. Ceci implique d'empêcher la réalisation d'opérations illégales contribuant à mettre en défaut ces propriétés, et aussi de garantir la possibilité de réaliser des opérations légitimes dans le système. La description de ces propriétés en tant que spécification fait partie de la *politique de sécurité* du système. Assurer la sécurité du système, c'est donc assurer que les propriétés retenues sont vérifiées.

D'après ITSEC [11], la politique de sécurité représente « l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique ». Souvent, on peut considérer qu'une politique de sécurité définit à la fois les *objectifs* de sécurité (c.a.d. les propriétés de confidentialité, d'intégrité et de disponibilité) et les *règles* de sécurité imposées aux systèmes, comme moyens permettant de modifier l'état de sécurité du système.

En fonction du domaine qu'elles décrivent, on peut classer les politiques de sécurité de la manière suivante :

physique qui touche à la partie physique du système, en spécifiant les parties critiques, les objectifs (ex. la protection contre les agressions physiques, les catastrophes naturelles, le feu, etc.) et les mesures à prendre (ex. l'utilisation d'un coffre-fort, d'un détecteur de feu, etc.)

administrative qui traite tout ce qui est lié à la sécurité d'une entreprise d'un point de vue organisationnel : l'organigramme choisi, la répartition des tâches et des responsabilités, l'affectation des fonctions aux individus, etc. Les propriétés de sécurité recherchées visent, par exemple, à limiter les cumuls ou les délégations abusives de pouvoir ou à garantir une séparation des pouvoirs.

logique qui traite le contenu du système d'informations, plus précisément, qui décrit le contrôle d'accès logique, en spécifiant qui a le droit d'accéder, à quoi et dans quelles circonstances.

Un politique de sécurité logique peut être affinée en plusieurs branches correspondant aux différentes étapes de l'accès au système d'information :

identification : un individu qui veut utiliser le système doit d'abord s'identifier ;

authentification : ensuite, il doit apporter la preuve qu'il est bien la personne qu'il prétend être ;

11. ITSEC est le résultat de l'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni, concernant les critères pour évaluer la sécurité d'un système informatique [ITSEC, 1992].

autorisation : enfin, on doit définir les opérations que l'utilisateur est autorisé à réaliser.

Dans la suite, nous restreindrons notre intérêt seulement aux politiques d'autorisation. Les organisations qui doivent respecter des contraintes de sécurité, que ce soit par obligation légale ou pour des raisons déontologiques (par exemple dans le domaine bancaire, dans le domaine des assurances, dans le domaine médical, etc.) expriment les politiques de sécurité généralement en langage naturel sous la forme d'un *règlement de sécurité*. En raison de leur nature informelle, ces règlements sont souvent sujets à interprétation et peuvent contenir des ambiguïtés ou conduire à des contradictions (par exemple, vis-à-vis du secret médical [Saury, 1991]). Pourtant, ils constituent la grande majorité des politiques de sécurité en application et font l'objet des représentations formelles [Jones et Sergot, 1993; Royakkers, 1994; Cuppens et Saurel, 1996].

Dans le cas des systèmes informatiques, l'utilisation des politiques d'autorisation constitue la majorité des travaux possédant une base rigoureuse, c.a.d. spécifiés clairement dans un langage mathématique à partir d'un *modèle de sécurité*. Les différents modèles de sécurité qu'on trouve dans la littérature (même s'ils correspondent souvent étroitement à la définition d'une politique de sécurité particulière [12]), sont caractérisés par un certain niveau de généricité. Cette généricité concerne d'une part la spécification du système en termes de composants actifs (qui manipulent l'information et qu'on appelle *sujets*) et passifs (qui contiennent l'information et qu'on appelle *objets*), et d'autre part l'introduction des attributs de sécurité associés aux éléments du système (comme les privilèges ou les niveaux dans les politiques multi-niveaux [13]).

Considérons, par exemple, le cas des modèles basés sur la notion de matrice de contrôle d'accès [Lampson, 1971], dédiée à la représentation des droits d'accès. La structure de ce modèle est celle d'une machine à états où chaque état est un triplet (S,O,M), où S est un ensemble de sujets, O un ensemble d'objets et M une matrice de contrôle d'accès. La matrice M possède une ligne pour chaque sujet et une colonne pour chaque objet. Un élément de la matrice $M(s,o)$ exprime l'ensemble des droits d'accès que le sujet s possède sur l'objet o. Ces droits d'accès sont pris dans un ensemble fini A, défini par la politique de sécurité, et correspondent aux différentes opérations qu'un sujet peut réaliser sur un objet [14]. La matrice n'est pas figée, puisqu'elle peut évoluer dans le temps en fonction de la création de nouveaux sujets ou objets et des opérations effectuées par les utilisateurs. L'état de sécurité est ainsi modifié par toutes les actions qui modifient la matrice M. La vérification des propriétés du modèle (c.a.d. la détermination des états maximaux de la matrice de contrôle d'accès) impose plus ou moins certaines contraintes acceptables. Par exemple, pour un système réel, de telles vérifications sont généralement coûteuses et in-

12. Par exemple, les modèles basés sur la notion de treillis sont en liaison étroite avec la définition d'une politique multi-niveaux [Bell et LaPadula, 1975].

13. Le modèle le plus connu est celui de Bell-LaPadula [Bell et LaPadula, 1975] qui est basé sur une partition de l'ensemble des sujets et de l'ensemble des objets. A chaque partition, on associe un *niveau*. Les objectifs de sécurité, ainsi que les politiques associées s'appuient également sur ces niveaux.

14. On utilise ce modèle pour contrôler les droits d'accès aux fichiers sous Unix.

complètes : on ne peut pas vérifier, sans faire une énumération totale, qu'il n'est pas possible d'atteindre un état d'insécurité. Par ailleurs, l'extension de ces modèles et surtout de leur implémentation vers les systèmes distribués pose de nombreux problèmes. Par exemple, la gestion centralisée d'une matrice de contrôle d'accès couvrant l'ensemble des sites du système a un impact extrêmement négatif sur les performances. Dans la plupart des implémentations, la défaillance d'un des sites met en péril la sécurité de l'ensemble du système.

Les politiques de sécurité ont un caractère obligatoire, dans le sens qu'on suppose que leur implémentation demande la présence d'une infrastructure de réalisation. Par exemple, les modèles basés sur une matrice de contrôle d'accès demandent que le système correspondant soit muni de mécanismes qui d'une part permettent la construction et la manipulation de la matrice, et d'autre part doivent empêcher ou laisser les sujets exécuter des opérations conformément au contenu de la matrice. La vérification des propriétés de sécurité est un problème légitime seulement en présence de cette hypothèse, sinon elle n'a pas de sens.

En conclusion, nous considérons que la problématique de la sécurité est étroitement liée à la notion de normativité, car elle essaie d'assurer la présence d'un ensemble de propriétés en imposant un ensemble de contraintes sur la construction et le fonctionnement d'un système. Beaucoup de travaux dans ce domaine peuvent être utilisés par la recherche dans le domaine des systèmes multi-agents, car les modèles proposés sont génériques en ce qui concerne la nature des éléments qui composent un système et les types d'interactions entre ceux-ci. De plus, ces travaux utilisent des concepts similaires pour décrire les politiques de sécurité (tels que les droits ou l'obligation) et proposent des moyens d'implémentation et des méthodes de spécification et vérification.

4.5.2 Coordination réglementée dans les systèmes distribués

Dans [Minsky et Ungureanu, 1997], les auteurs proposent un modèle de coordination dans les Systèmes Ouverts par l'intermédiaire des politiques (ou lois) de coordination. Une *loi* est composée d'un ensemble de règles (spécifiées en Prolog) qui réglementent l'échange de messages entre les membres d'un groupe d'agents. Les règles s'appliquent au moment où un agent envoie ou reçoit un message. Ce mécanisme est réalisé par l'intermédiaire d'un agent *contrôleur* qui est créé pour chaque agent membre du groupe. Son rôle est de se placer entre l'agent et l'environnement de communication, et de s'assurer que les règles sont respectées :

– pour les événements `send`, avant la délivrance du message à l'environnement de communication ;
– pour les événements `receive`, avant la délivrance du message à l'agent.

Les règles sont maintenues par un agent *secrétaire* et elles s'appliquent à tous les membres du groupe, car ces règles sont principalement réactives de type condition/action. Minsky ajoute la possibilité de spécifier que certaines actions doivent être exécutées à un certain moment à l'aide des *obligations*. A titre

```
sent(X,M,s) :-
     buffer([])@CS,
     lastCall(Tlast)@CS, delay(DT)@CS, clock(T)@CS,
     T > (Tlast + DT) ->
          (do(lastCall(Tlast) <-lastCall(T)), do(forward)),
          |
          (do(+obligation(sendMessage,Tlast+DT)),
          do(buffer([])<-buffer([M]))).
```

FIG. 4.1 – *Exemple de règle pour la coordination réglementée*

d'exmple, considérons le cas du contrôle du flux d'appels des clients vers un serveur. La règle présentée dans la figure 4.5.2 assure que le message M envoyé vers le serveur s sera renvoyé (par le contrôleur de l'agent) s'il n'y a pas de message dans sa mémoire tampon (buffer) et si entre deux invocations un délai minimum DT est respecté. Sinon, le message est mis dans le buffer et une obligation d'envoyer le message au moment le plus proche possible satisfaisant la condition de délai minimum est construite.

4.5.3 Coordination par obligations - Barbuceanu

La plupart des travaux liés à la coordination se concentrent davantage sur l'aspect du travail en commun ou en équipe. Plus précisément, on considère que tous les membres d'une équipe ont principalement une seule obligation ou engagement par rapport à un seul but commun. Pourtant, Barbuceanu et ses collaborateurs [Barbuceanu *et al.*, 1998; Barbuceanu, 1998] proposent une théorie de la coordination sociale qui est basée sur l'échange [15] de plusieurs contraintes normatives ou sociales. Ces contraintes sont décrites à l'aide des concepts déontiques tels que l'obligation, l'interdiction et la permission (*OPI*). Leur fonction est de caractériser essentiellement les rôles définis dans un SMA qui, par défaut, est vu comme ayant une structure organisationnelle. Un agent qui joue un certain rôle dans l'organisation sera soumis à toutes les contraintes associées à ce rôle. Afin que les agents puissent inférer quelles sont les obligations qu'ils doivent prendre en compte ou choisir parmi les normes conflictuelles, chaque agent dispose d'un mécanisme de propagation des contraintes déontiques.

La modélisation des OPI utilise le modèle de réduction de la logique déontique à la logique dynamique, proposé par Meyer [Meyer, 1988] (voir également le chapitre 6). Pour obtenir des actions complexes on peut utiliser lors de la description des actions les opérateurs de composition séquentielle, choix non-déterministe et composition parallèle, symbolisés respectivement par « ; », « $\cup$ » et « $\&$ ». Le mécanisme de propagation des contraintes utilise les théorèmes (6.8) - (6.17) formulés dans le modèle de Meyer. Ces théorèmes ne sont utilisables que pour décrire le comportement des agents pour des ac-

15. Le langage utilisé COOL (COOrdination Language) est basé sur KQML.

tions composites. Ils ne permettent pas de comparer les obligations les unes aux autres lorsque celles-ci génèrent des situations conflictuelles. Par conséquent, Barbuceanu introduit la notion de coût de violation attaché à chaque contrainte déontique. Ce coût permet de résoudre un conflit en choisissant d'exécuter l'action qui génère la violation la moins coûteuse. Le mécanisme de propagation des contraintes déontiques utilise une représentation des buts sous la forme d'un réseau. Les noeuds représentent les buts (ou les actions) et les arcs les liens de composition. Lors du calcul du coût d'un but complexe, on applique le mécanisme de propagation étendu de deux nouvelles règles. La règle de propagation qui comprend le *coût minimum* s'applique pour calculer le coût d'un but complexe de type choix non-déterministe, composé par des sous-buts interdits (voir le théorème (6.9)) : le coût de l'action complexe est le coût minimum des sous-buts. La règle de propagation avec le *coût maximum* s'applique quand le but complexe est obtenu par une composition parallèle des sous-buts interdits (voir le théorème (6.10)) : intuitivement, le coût de l'action complexe est donné par le sous-but qui est le plus cher.

La coordination entre agents est réalisable grâce aux échanges (par voie communicative) des contraintes déontiques. Par exemple, un agent A décrit ce qu'il veut de la part d'un autre agent B, par l'intermédiaire d'un ensemble d'obligations et d'interdictions qui sera communiqué à B. Localement au niveau de l'agent B, il y aura un processus de propagation de cet ensemble et d'autres obligations ou interdictions concernant les autres engagements de B. C'est ainsi que l'agent B va décider ce qu'il pourra faire ou non. Dans le cas où il lui est impossible de donner un cours positif aux requêtes de A, l'agent A sera obligé de revoir les conditions de sa requête et de modifier ses paramètres : ajouter ou retirer des contraintes déontiques, augmenter ou baisser le coût de violation, etc.

4.5.4 BDI et normes

Le modèle BDI (voir 2.4.1) est basé sur l'aspect cognitif du comportement d'un agent. L'architecture d'un agent est composée de plusieurs modules spéciaux représentant des croyances, des désirs et des intentions. Plus précisément, les croyances de l'agent représentent ses croyances sur le monde, sur les autres agents et sur lui-même. Les désirs correspondent aux buts et aux préférences de l'agent ; ils font référence à ses motivations. Enfin, les intentions représentent les engagements de l'agent et les plans d'actions qu'il va exécuter pour satisfaire ses buts et désirs. L'agent cherche en permanence à évaluer et choisir ses désirs en fonction des croyances associées. Les intentions donnent les plans futurs que l'agent suivra. Notons que les intentions ne sont pas irrévocables ; elles changent car le monde dans lequel les agents évoluent change aussi.

Dans sa forme originale, le modèle BDI décrit les caractéristiques d'une architecture d'agent très générale qui peut être utilisée pour décrire le comportement dans un monde mono-agent ou multi-agents. Souvent, on sent le besoin de bien distinguer les actions ayant des motivations intrinsèques, nées à l'intérieur de l'agent et les actions conformes à des intérêts sociaux, venues de l'extérieur, générées par le fait qu'on met l'agent dans un contexte social. Si les intentions expriment des choix parmi les désirs d'un agent, en tant que motivations, c'est donc parmi les désirs qu'on doit faire cette séparation. Dans la littérature sur les SMA il existe plusieurs propositions que nous analysons dans la suite.

Dignum et al.

Dans [Dignum *et al.*, 2000] Dignum et al. proposent d'augmenter le modèle BDI et de lui adjoindre deux nouvelles structures, les obligations et les normes. Les deux notions représentent des mécanismes destinés à influencer les agents. Du point de vue de l'agent, ces mécanismes sont une source des motivations (dites *externes*) pour agir. Elles décrivent les comportements standards que les agents doivent manifester. Leur but est d'obtenir une coordination plus efficace. Techniquement, elles agissent comme un filtre sur les buts possibles d'un agent.

La différence conceptuelle entre les normes et les obligations, quant à elle, n'est pas facile à comprendre, les auteurs s'appuyant sur des travaux antérieurs [Conte *et al.*, 1999] qui introduisent ces concepts par rapport à leur impact sur un agent. Selon ces travaux, les normes doivent être *acceptées* consciemment par l'agent, alors que les obligations doivent être toujours respectées (sinon une punition sera appliquée dans le cas d'une violation). Cette différence est moins évidente au moment de la formalisation car les deux opérateurs expriment finalement des obligations. Pour décrire les normes, on utilise l'opérateur dyadique $N^z(\phi|\psi)$ qui exprime l'obligation générale (ou impersonnelle) pour tout agent qui accepte la norme, de réaliser l'apparition de la propriété ϕ à condition que ψ soit vraie. Pour exprimer les obligations, on utilise l'opérateur dyadique $O^z_{ab}(\phi|\psi)$, qui cette fois représente une obligation orientée (ou personnalisée) de l'agent a (son porteur) envers l'agent b (sa contrepartie) selon laquelle l'agent a est obligé envers l'agent b de faire en sorte que ϕ si ψ est vraie. Pour les deux opérateurs, z représente la société ou l'organisation à qui appartient ou qui a émis la norme et qui est responsable, dans le cas d'une obligation violée, d'assurer l'apparition des conséquences correspondantes (i.e. pénalités).

L'architecture BDI modifiée permet un raisonnement sur les normes et les obligations considérées comme un certain type de connaissances. Ce raisonnement est possible grâce à un nouveau type d'événements appelés déontiques (*deontic events*). Ils correspondent à l'application immédiate d'une obligation ou d'une norme. Par exemple, si l'agent a se trouve sous l'influence d'une obligation orientée $O^z_{ab}(\phi|\psi)$ et que la condition ψ devient vraie, alors un événement déontique, $\mathcal{O}^z_b(\phi)$, sera mis dans la file d'événements de a. Les événements déontiques sont provoqués par des modifications des normes, des obligations ou des croyances d'un agent. Pour répondre aux événements déontiques, les agents ont des plans spécifiques dont la condition de déclenchement est donnée par ces événements. Par exemple, l'agent a peut avoir un plan avec la condition de déclenchement $\mathcal{O}^z_b(\phi)$ et le corps composé d'une suite d'actions assurant la réalisation de ϕ.

Les auteurs de [Dignum *et al.*, 2000] proposent également de traiter des normes dites *introspectives*,

```
BDI-iterpreter
initialize-state() ;
do
    selected-events := event-selector(event-queue) ;
    augmentd-events := potential-event-closure(selected-events) ;
    options := option-generator(augmented-events) ;
    selected-options := deliberate(options) ;
    update-intentions(selected-options) ;
    execute() ;
    get-new-external-events() ;
    drop-successful-attitudes() ;
    drop-impossible-attitudes() ;
until quit
```

FIG. 4.2 – *L'interpréteur BDI modifié*

c.a.d. des normes ayant la condition de déclenchement constituée d'obligations ou d'autres éléments de type cognitif (intentions, croyances, désirs). Pour illustrer ce type de norme, ils donnent l'exemple d'une norme du genre « si un agent est obligé d'exécuter une tâche ϕ pour un autre agent x et qu'il n'a pas l'intention de le faire, alors il doit le dire à x ». Formellement, cela s'écrit : $N^z(dire(x)|\mathcal{O}_x^z(\phi) \wedge \neg I(\phi))$ [16]. Techniquement, pour permettre le traitement de ce genre de norme, l'interpréteur BDI doit parcourir dans un seul cycle deux étapes. D'abord, il s'agit de prendre en compte la présence de l'obligation, l'effet produit étant la mise à jour de la structure contenant les normes, et puis de traiter effectivement les normes. Comme l'interpréteur ne peut traiter, dans le processus de délibération, qu'une seule étape à la fois, les auteurs proposent de :

- modifier l'interpréteur d'origine en ajoutant un nouveau type d'événement appelé *événement déontique potentiel* ;
- traiter les événements déontiques potentiels dans une étape qui suit celle de la sélection des événements (voir la figure 4.2).

D'après nous, l'utilisation des normes introspectives pose des problèmes de vérification, car il nous est impossible de savoir de l'extérieur si l'agent est conforme à un tel type de norme. Pour ce faire, on doit connaître les états mentaux de l'agent, ce qui représente une hypothèse très forte. Dans l'exemple ci-dessus, il serait plus réaliste d'avoir une norme qui demande d'informer x au moment où on ne l'aide pas

16. I est le symbole de l'intention.

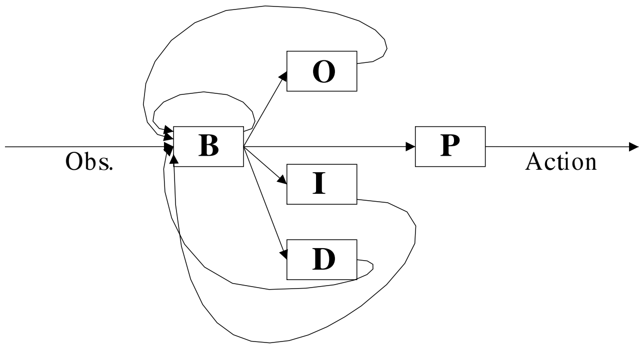


FIG. 4.3 – *BOID*

effectivement [17]. Cela permettrait de pénaliser une violation à base de faits et non pas à base d'intentions. On peut très bien échouer, même si on a l'intention d'agir.

BOID

Un autre effort d'intégrer le traitement des obligations dans une architecture BDI est celui présenté dans [Broersen *et al.*, 2001]. La différence avec le modèle de base consiste dans la représentation explicite de buts créés par des motivations externes à l'agent et qu'on appelle obligations. Dans l'architecture BOID [18] les quatre composantes élémentaires (une composante pour chaque attitude mentale) sont des processus de type entrée-sortie dont le comportement est décrit par un ensemble de règles de production. Ces processus communiquent entre eux par l'intermédiaire d'un ensemble de formules, appelées extensions [Thomason, 2000] - des ensembles maximaux, cohérents et fermés pour la déduction.

L'activité de délibération consiste à choisir une règle parmi les règles applicables (*conflict resolution*). Il existe deux types de conflits entre les règles : internes et externes. Les conflits internes peuvent apparaître entre les règles du même type d'attitude mentale, par exemple un conflit entre les intentions : j'ai l'intention de finir mon article pour dimanche ; j'ai aussi l'intention d'aller samedi à la plage ; si je vais samedi à la plage je ne peux pas finir l'article pour dimanche. Les conflits externes sont des conflits entre des règles de types différents. Par exemple, le conflit entre une obligation et une croyance (BO conflict) : je suis obligé de voir ma belle-mère ce week-end, mais je crois que je n'ai pas le temps de le faire.

En fonction de la manière dont les conflits sont résolus, on obtient plusieurs types d'architectures. Par exemple dans le cas de l'architecture *Single Extension* BOID on attribue des valeurs de priorité uniques à

17. Cette solution à son tour pose des problèmes, cette fois-ci, de formalisation, connue dans la logique déontique sous le nom du paradoxe de Chisholm. Pour plus de détails voir le chapitre 4.

18. Acronyme pour *Beliefs, Obligations, Intentions and Desires*.

chaque règle et on traite une seule extension (un seul ensemble de formules) à la fois. On obtient ainsi un système de production à base de priorités qui a un comportement décrit par la boucle suivante jusqu'au moment où, soit il ne reste plus aucune règle applicable, soit on arrive à un point fixe (le résultat est toujours le même, c.a.d. la même extension) :

1. chercher les règles applicables à l'extension courante : *conflict set* ;
2. sélectionner dans le *conflict set* la règle qui a la priorité la plus faible ;
3. calculer la nouvelle extension en appliquant la règle sélectionnée.

En fonction de la valeur de la priorité attachée à une règle, on peut privilégier une certaine attitude mentale. On obtient ainsi différents types de comportements pour un agent : réaliste (on privilégie les croyances), égoïste (les désirs sont prioritaires), social (les obligations sont prioritaires), etc.

Une autre architecture alternative consiste à traiter les conflits entre les règles du même type d'attitude mentale avant de continuer le traitement des autres. Par exemple, pour un agent de type BIDO, on commence d'abord par traiter la composante B, puis une règle de I qui va nourrir de nouveau la composante B. Ensuite, on applique les règles de D, qui vont influencer le contenu des I et par conséquent celui de B. Enfin, les règles de O seront appliquées, chaque fois le contenu de B, I et D est modifié de nouveau.

4.6 Discussions et conclusions

Dans ce chapitre, nous avons essayé de comprendre ce qui est derrière le concept de norme en remontant à l'origine de son emploi et de sa définition. On remarque que ce concept est utilisé sous une variété de formes dans des domaines divers comme la philosophie, les sciences sociales, le domaine juridique ou l'informatique, d'où la difficulté de donner une définition précise et d'indiquer une utilisation exacte. Dans la suite nous indiquons globalement les principales leçons qu'on peut tirer de cette investigation :

– le concept de norme est introduit par rapport à la notion d'autorité. Elle est au service d'une autorité et exprime ses buts. Une autorité est une sorte de repère, dont la valeur est reconnue, acceptée, et à laquelle on adhère.
– la norme est prise en compte si l'autorité se manifeste d'une façon ou d'une autre (elle existe, on l'a vu) ;
– être normatif représente le fait d'avoir un comportement conforme à la norme. En effet, il s'agit d'une caractérisation binaire, on l'est ou on ne l'est pas ;
– accepter d'être normatif impose des contraintes sur le comportement et le modifie pour pouvoir satisfaire le but pour lequel la norme a été conçue ;

– dans un groupe ou système social, l'autorité est légitimée par le système lui-même. Les buts d'une société sont prioritaires par rapport aux buts de l'individu.
– dans le cas d'un système dont les membres ont la possibilité de rejeter (violer) les valeurs communes tout en restant dans le système (par exemple, les agents autonomes), les normes sont doublées de sanctions formelles. Le rôle des sanctions est d'augmenter, de faire valoir l'autorité sociale.
– l'application du contrôle ou des sanctions nécessite la présence d'une infrastructure spéciale ;
– les systèmes enrégimentés sont les systèmes dans lesquels il est impossible (par construction) d'avoir un comportement non-conforme à la norme. Il n'est pas nécessaire de parler de violation dans un tel système, car elle ne se produit jamais.

Notre intérêt pour l'étude des normes vient du fait que les normes peuvent être considérées dans les SMA comme une source de motivations (ou de contraintes) pour l'agent qui le poussent à agir ou à s'abstenir de toute action. Dans la suite, nous faisons quelques remarques pour préciser quelles sont les propriétés des normes et les problèmes qu'on doit traiter lorsqu'on les utilise dans le cadre de la modélisation de l'interaction entre agents :

1. Les normes expriment un comportement ou une situation idéale du point de vue de l'agent lui-même ou des autres (agents ou autorités quelconques). L'un des problèmes liés aux normes est donc de savoir quelle est leur origine, interne ou externe, c.a.d. si elles correspondent aux désirs ou aux principes d'action d'un agent, d'un groupe d'agents ou d'une autorité.
2. Un agent est considéré comme normatif par rapport à un ensemble de normes, si son comportement est conforme au comportement décrit par les normes. Ce qui nous intéresse dans un premier temps dans l'utilisation des normes est une décision binaire, à savoir si on peut qualifier un agent comme étant normatif ou non. Le deuxième objectif lié à l'utilisation des normes est d'imposer à l'agent un certain type de comportement idéal, de l'influencer dans ce sens.

   On remarque que le premier objectif des normes n'impose aucune contrainte sur le type d'agent participant à l'interaction normative. Peu importe si l'agent est rationnel ou réactif, le but est d'obtenir de la part d'un agent un comportement conforme à la norme. Par contre, la complexité du contenu véhiculé par une norme nous incite à nous interroger beaucoup sur la pertinence du modèle réactif.
3. Un autre problème est celui de l'impact des normes sur un agent. Puisque les normes existent indépendamment de l'agent, on doit préciser si un agent a la possibilité de les refuser ou si elles sont imposées. Une fois une norme acceptée (de manière « consciente ») ou imposée, elle génère des contraintes souvent décrites par des concepts déontiques tels que l'obligation et l'interdiction. On doit aussi préciser si l'agent à la possibilité de violer ces contraintes. Cela nous permet d'avoir une plus grande diversité de degrés de liberté dans les comportements des agents. Par exemple,

on peut passer du niveau des agents enrégimentés [19] (dont le comportement est en toute circonstance conforme à la norme parce que, par construction, on contrôle complètement l'exécution d'un agent), à un niveau où on influence les agents par des mécanismes de primes ou de pénalités et en finissant sur un niveau de liberté totale, où on mise sur l'autonomie de chaque agent, responsable de son comportement.

4. La représentation des normes doit autoriser un certain degré de manipulation, où on peut les créer, stocker, détruire ou les modifier. Les normes peuvent être communiquées aux agents. De plus, on doit pouvoir raisonner sur les normes, par exemple pour inférer à qui elles s'appliquent, quelles sont les contraintes qu'elles imposent, sous quelles conditions, avec quelles exceptions, etc.
5. Un agent peut se trouver à la fois sous l'influence de plusieurs normes ou obligations qui imposent des comportements contradictoires. Par exemple, un agent peut avoir simultanément deux obligations d'exécuter α et de ne pas exécuter α. Pour éviter l'apparition des conflits, on introduit une relation d'ordre (ou de préférence) entre normes ou obligations. Même si, objectivement, il est impossible de comparer deux normes, cette relation subjective aide l'agent à choisir son propre chemin. La relation de préférence peut être considérée comme un trait de caractère, spécifique à chaque agent. De plus, elle peut être étendue pour résoudre des conflits avec d'autres concepts cognitifs tels que les désirs ou les intentions, lorsque la norme et les propriétés cognitives expriment des comportements contradictoires.
6. Le fait qu'une norme peut exister indépendamment des sujets à qui elle s'applique, implique une séparation entre les mécanismes utilisés dans la construction des normes et ceux qui assurent leur réalisation. Les derniers sont plus ou moins présents (nécessaires) dans le système normatif en fonction des dégrés de liberté qu'on donne aux agents. Dans le cas des interactions sans restrictions, chaque agent va adopter sa propre attitude face aux violations ou au respect des normes. A l'opposé, dans le cas des agents enrégimentés, c'est le rôle du système (comme environnement d'exécution) d'assurer que les agents ne se trouvent que dans des situations conformes aux normes. Entre les deux, on trouve le cas peut être le plus intéressant qui nécessite un partage de contrôle entre les agents autonomes et le système dans lequel ils évoluent. Les normes sont utilisées dans un double rôle d'influencer et de contrôler les agents.

Dans la littérature étudiée, on remarque que la frontière entre la notion de norme et celle d'obligation est souvent très floue. Nous essayons ici de montrer les différences que l'on peut faire entre les deux notions, conformément à notre niveau actuel de compréhension.

1. La première manière de voir l'obligation est en tant que concept déontique. Elle est utilisée pour qualifier une situation ou un comportement, comme propriété. Par exemple, on l'utilise pour exprimer des propriétés de type « il est obligatoire que α » ou plutôt « α est obligatoire » où α

19. Comme on l'a déjà précisé ce niveau correspond à la plupart des cas rencontrés en informatique.

représente un comportement ou une situation. Étant donné que les normes décrivent des situations et des comportements idéaux ou normaux, on peut remplacer les expressions de type « il est idéal que ... » ou « il est normal que ... » par une expression déontique (unique). Ainsi, au moment de l'application d'une norme, on génère des contraintes sur les comportements des agents qui sont des propriétés exprimables par des obligations. Une première conclusion est donc que les concepts déontiques sont utilisés dans la construction des normes pour mettre en évidence qu'il s'agit des comportements ou des situations idéaux.

2. La deuxième manière de voir l'obligation, et qui aide à la distinguer d'une norme, concerne la perspective sociale qu'elle exprime. Ainsi, la principale différence entre une norme et une obligation, comme moyen d'exprimer un but à poursuivre par les autres, est liée au contexte de leur application. Les normes expriment une perspective plus large, celle de toute une société d'agents qui doit atteindre un but dit social. Quant aux obligations, elles s'appliquent à un niveau inférieur que celui d'une société, notamment pour exprimer les besoins concrets (buts) d'un seul agent ou d'un groupe d'agents.
3. On remarque que les obligations ne sont pas forcément liées au concept de norme ; on peut générer des obligations autrement, par exemple en faisant une promesse, en prenant un engagement, en demandant ou ordonnant l'exécution d'une tâche[20].
4. La notion de violation est généralement liée à celle d'obligation. Comme les normes, les obligations existent indépendamment de l'agent. Si les normes peuvent être acceptées ou non, cela n'est pas le cas pour les obligations. La présence d'une obligation ne peut pas être ignorée, elle doit être prise en compte par l'agent. Par contre, on laisse à l'agent le choix de lui obéir ou non. La violation des obligations implique souvent l'existence de conséquences concrètes (i.e. des sanctions). C'est cette caractéristique qui permet une certaine flexibilité dans le comportement de l'agent, en sachant que les obligations réduisent son autonomie par la présence (facultative) des conséquences explicites si elles sont violées.

Toutes ces observations ne font que compléter l'image sur le paradigme social présenté dans le chapitre 3, préparant le terrain pour l'intégration des normes dans l'interaction entre agents. On rappelle que notre but est d'une part de préciser quelles sont les structures architecturales nécessaires à cet effort et d'autre part de proposer les modèles et les outils correspondants pour les construire. Dans le chapitre suivant nous présentons notre manieère d'aborder la partie liée aux structures de l'interaction normative.

20. On peut même introduire l'action d'obliger qui a comme effet principal la génération d'une obligation.

Chapitre 5

Systèmes d'agents normatifs

5.1 Introduction - systèmes d'informations ouverts

Au début de cette thèse, nous avons mis en évidence que l'une des applications des SMA est la modélisation et la conception des Systèmes d'Informations Ouverts (SIO) [Hewitt et de Jong, 1984]. L'intérêt croissant pour l'étude de ces systèmes est motivé par l'évolution rapide de l'Internet et de la puissance de calcul des ordinateurs qui exigent et permettent aussi le développement des applications distribuées (ex. le commerce électronique). Un SIO est en général un système :

1. gros - contient beaucoup de composants,
2. distribué - les composants se trouvent sur plusieurs machines,
3. hétérogène - les composants sont conçus de manière différente par diverses personnes,
4. dynamique - les composants sortent et entrent dans le système pendant son exécution,
5. avec des composants autonomes.

Parmi les caractéristiques d'un SMA, l'autonomie est la plus attractive pour la modélisation des SIO. Elle suppose que l'agent est libre de percevoir et d'agir dans son environnement d'exécution, sans l'intervention explicite du concepteur externe. C'est une propriété importante, car, dans la pratique, les agents sont utilisés pour désigner des composantes hétérogènes, conçues de manières diverses par différentes personnes et donc difficilement modifiables pour s'adapter aux situations nouvelles.

Un autre aspect important lié à la modélisation des SIO est le besoin de systèmes robustes dont la notion clé est celle du contrôle. On contrôle l'exécution d'un programme, l'allocation dans l'espace mémoire, l'accès aux ressources, etc. Appliqué aux agents, ce concept signifie qu'on contrôle (i) ce que les agents perçoivent et (ii) leur façon d'agir. Si, dans le premier cas, on peut construire des mécanismes de filtrage de l'information venue de l'environnement et perçue par les agents, dans le deuxième cas, il faut proposer une solution qui soit un compromis acceptable entre l'autonomie des agents et le contrôle de leurs comportements.

Selon nous, un bon candidat capable de réaliser le compromis entre l'autonomie et le contrôle des agents est celui de l'utilisation des normes. Une norme est une information qui décrit un comportement idéal (ce que l'agent doit faire ou ne pas faire). Son rôle principal est d'*influencer* le comportement d'un agent. On préserve ainsi l'autonomie des agents.En revanche, en ce qui concerne le contrôle, on doit proposer des structures spéciales qui nous assurent que les normes sont coercitives, c.a.d. qu'elles ont réellement une influence sur le comportement des agents. Dans la suite nous présentons notre approche de la structuration de l'interaction entre agents autonomes pour pouvoir utiliser les normes dans les deux sens, informatif et coercitif. Les structures architecturales qui en résultent sont appellées de façon générique Système d'Agents Normatifs (SAN).

Le rôle de ce chapitre est de montrer quels sont les éléments architecturaux et les principes pour réaliser des SAN. Nous allons intégrer des concepts et des idées qu'on a vus dans les chapitres précédents, concernant la perspective sociale de la conception des SMA et la notion de norme.

5.2 Scénario - protocole d'entrée dans un SAN

Afin de montrer quels sont les éléments nécessaires à la conception d'un SAN, on commence par présenter le scénario le plus simple dans un tel système (fig. 5.1). Supposons qu'il existe déjà un SAN qui est finalement une agence dans laquelle tous les agents obéissent à un ensemble de normes $\mathcal{N}$. Si un agent de l'extérieur de l'agence veut résoudre un problème qui nécessite la coopération avec les agents du SAN, alors il doit entrer dans le SAN et avoir un comportement « obéissant », conforme à l'ensemble $\mathcal{N}$. L'entrée dans le SAN est régie par un *protocole d'entrée* qui demande d'abord que l'agent fasse une demande d'entrée explicite auprès du SAN (1ère étape). Si le SAN ou son représentant décide (2ème étape) que l'agent a les capacités d'interagir avec les agents internes, il va lui envoyer $\mathcal{N}_{rôle}$, une instance du règlement $\mathcal{N}$, en fonction du *rôle* que l'agent va jouer par la suite (3ème étape). Une fois l'agent informé de son rôle et du règlement afférent, il peut être considéré comme faisant partie du SAN et donc il peut faire appel aux autres agents tout en respectant $\mathcal{N}_{rôle}$ (4ème étape).

5.3 Structures de l'interaction

A partir de ce scénario, on essaie de montrer comment structurer les interactions des agents génériques afin de réaliser des architectures de type SAN.

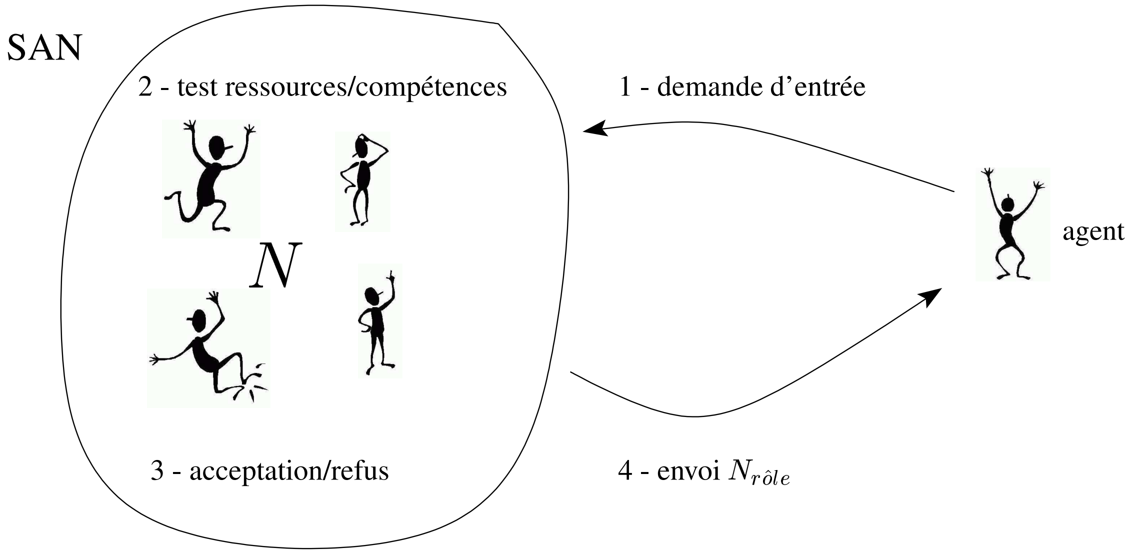


FIG. 5.1 – *Scénario - protocole d'entrée dans un SAN*

5.3.1 Niveau comportemental

Par définition, les agents sont situés, c.a.d. que l'on peut toujours préciser de l'extérieur quels sont les éléments de leur environnement : les actions possibles, les objets sur lesquels ils peuvent agir, les informations qu'ils reçoivent ou transmettent, etc. Cette propriété permet de décrire leur comportement dans un environnement. En considérant cette hypothèse, on peut faire appel aux normes, telles qu'elles sont utilisées dans les sociétés humaines : les normes sociales ont pour rôle d'influencer le comportement d'un individu dans la société. Adaptées au monde des SMA, les normes vont jouer le même rôle, d'influencer les comportements des agents.

Afin de préserver l'autonomie d'un agent, la norme est essentiellement indicative. Elle décrit un comportement idéal potentiellement différent du comportement effectif de l'agent. On remarque que cette perspective est à l'opposé du cas d'agents enrégimentés, où les agents sont construits d'une telle manière qu'ils obéissent à toutes les normes dans toutes les situations, sans exception. La souplesse du contexte normatif doit être une caractéristique d'un SAN et elle consiste dans la possibilité de modifier le degré d'obéissance à un ensemble de normes. Par exemple, un architecte SAN peut utiliser le coût de violation comme paramètre de contrôle du niveau de conformité à une norme.

Il est clair que l'existence des normes doit être prise en compte au niveau de l'architecture d'un agent. Un agent doit être capable de produire des plans pour agir premièrement en conformité aux raisons pour lesquelles il a été conçu (par exemple, le modèle BDI correspond à ce critère) et deuxièmement, pour qu'il soit conforme aux normes qui viennent de l'extérieur. Ainsi, dans l'architecture d'un agent, on précise la façon dont les deux aspects sont inter-connectés, pour répondre aux deux besoins (par exemple, on doit montrer comment on intègre les normes dans le modèle BDI).

5.3.2 Niveau organisationnel

Un autre type de structuration est de doter un SMA ordinaire d'une structure organisationnelle. L'agence sera vue comme une organisation possédant une structure hiérarchique à base de rôles. Au moment de la spécification du système, les interactions seront décrites entre rôles et non entre agents. De ce point de vue, le rôle dans les SMA est une abstraction qui a la même importance qu'une variable dans la programmation. Il permet une description plus générale d'un comportement. Par contre, au moment de l'exécution du système, on est obligé de raisonner sur le comportement d'un agent concret, tout en tenant compte de sa spécification et donc de son rôle. En attribuant un rôle à l'agent, on suppose qu'il y a un transfert vers l'agent de toutes les propriétés (et les contraintes) définies d'une façon générique pour le rôle.

Lorsqu'un agent joue un rôle, il peut avoir une vue partielle ou complète de la hiérarchie organisationnelle et de l'ensemble des normes. Le degré de connaissance qu'un agent a de l'ensemble du système peut être un paramètre modifiable de la politique d'implémentation utilisée par le concepteur du système. Une vision complète permet aux agents une initiative plus large et une vue partielle fait partie des moyens de contrôler (ou limiter) les comportements des agents.

L'utilisation d'une structure hiérarchique introduit un autre avantage, à savoir celui de pouvoir introduire la notion d'autorité. Une autorité est importante dans la mesure où un système normatif exprime non pas seulement ce qu'on doit faire, mais aussi ce qui se passe dans le cas contraire. La notion d'autorité a donc un double rôle : elle représente l'élément qui motive les agents à obéir aux normes et celui qui est chargé de mettre en place les pénalités correspondant à leur violation. Dans une architecture SAN, les deux rôles peuvent être joués par des entités différentes ou par une seule entité (par exemple, le même agent).

Par conséquent, la notion d'organisation apporte deux éléments à la structuration de l'interaction :

- une composante de type informationnel permettant à l'agent d'acquérir des connaissances sur l'identité des autres agents dans le système, leurs compétences, leurs droits et obligations, et de se positionner par rapport à eux d'un point de vue institutionnel (s'il s'agit d'une relation de dépendance ou de pouvoir).
- une composante de type infrastructure qui assure la mise en place de la partie physique de l'organisation. Par exemple, une structure qui contrôle l'échange des messages entre agents, en ne laissant un agent communiquer qu'avec les agents qu'il connaît et avec les supérieurs hiérarchiques, ou qui assure qu'un agent ne peut exécuter que les actions autorisées, etc.

5.3.3 Niveau communicationnel

La structuration de l'interaction peut s'appliquer aussi au niveau de la communication entre agents. L'approche la plus répandue est de considérer les messages exprimés dans un langage de communication de haut niveau (FIPA ACL, KQML) basé sur la théorie des actes de langage de Searle [Searle, 1969].

La norme est une contrainte imposée localement à un agent afin d'obtenir un but social global. La manière dont la contrainte normative se manifeste diffère d'un système à l'autre, l'agent étant plus ou moins capable de savoir quel est son contenu ou ses effets. Par exemple, on peut enrégimenter les agents par une implémentation des normes qui utilise soit des contraintes physiques, soit des solutions de type logiciel en modifiant le programme de l'agent pour qu'il soit conforme. Une autre solution, plus libérale, consiste à informer l'agent du contenu d'une norme, comme dans le scénario présenté dans la section 5.2. Et enfin, le dernier cas est celui où l'agent découvre tout seul la norme ([Conte *et al.*, 1999]) par un processus d'observation ou d'apprentissage par découverte.

Dans ce paragraphe nous étudions la situation dans laquelle les normes se transmettent par voie communicative. L'avantage de ce cas est double : d'une part il est directement lié au modèle SAN et d'autre part, lorsqu'on veut modifier une norme il est plus simple de le faire par envoi de message que de modifier physiquement l'environnement ou l'agent (reprogrammation).

Dans un premier temps, il s'agit d'assurer la transmission des normes entre l'autorité (ou son représentant) et l'agent. Si les agents communiquent par l'intermédiaire d'un langage de communication dédié (ex. FIPA-ACL [FIPA, 2000]), le performatif de type `inform` avec la norme déposée dans son contenu, en général suffit. L'effet de cet acte sur l'agent est de mettre à jour les croyances normatives.

Une autre situation où la communication est importante est liée au cas où l'état normatif est créé par un acte de langage. Cela implique que la sémantique du langage est étendue afin de prendre en compte les aspects déontiques. De plus, la nouvelle sémantique permet de différencier entre plusieurs expressions d'une même requête réalisée dans des contextes différents qui dépendent finalement des rôles que les agents jouent, des leurs droits, des autorités reconnues, etc. Par exemple, une même requête peut être une commande qui doit être exécutée tout de suite, une obligation ayant une force moins importante que celle d'une commande et qu'on peut ajourner moyennant un délai ou finalement une simple demande qui n'implique pas de conséquences graves si on l'ignore. On peut ainsi introduire dans la communication entre deux agents de nouveaux actes de type impératif pour exprimer les obligations, les interdictions ou les permissions, leur effet illocutoire étant la création de propriétés déontiques.

Traditionnellement, les protocoles de communication dont le rôle est de décrire des échanges structurés de communication sont implémentés par les concepteurs des agents logiciels comme des machines à états finis. Pourtant, cette démarche est très rigide, car si les protocoles changent alors il faut modifier (reprogrammer) la structure interne des tous les agents. Une alternative serait d'utiliser la description normative de la communication, c.a.d. de reconsidérer les protocoles et les dialogues comme étant des conventions ou des normes décrivant un comportement communicationnel idéal. Dans le cas où le protocole, cette fois-ci indicatif, ne correspond pas aux besoins de la communication, il peut être soit modifié d'une manière *ad hoc*, soit complètement abandonné. Cette approche peut être utile aussi aux théories de la communication qui considèrent le dialogue construit d'une manière dynamique. On peut utiliser des morceaux de dialogue qui localement au niveau d'une partie du discours sont des conventions (voir des règles).

5.4 Normes

Les normes proviennent en dernière instance d'une autorité et décrivent des comportements (ou des situations) idéaux ou standards. Le problème normatif consiste à se conformer à ce qu'on considère être idéal. Dans les SMA, la norme peut devenir le principal moyen de décrire les objectifs poursuivis par un système et les modalités de les satisfaire par le biais des activités de ses agents. Ces objectifs peuvent être l'expression du but du concepteur du système, d'un groupe d'agents ou d'un seul agent. On considère donc une norme comme étant seulement une description syntaxique d'un comportement standard.

Pour préserver leur autonomie, on permet aux agents d'ignorer ou de violer les normes, c.a.d. de choisir s'ils sont conformes aux comportements standards. Cette caractéristique permet d'aller de la simple détection d'un comportement « anormal » jusqu'à l'application des pénalités, voir des punitions. Outre le cas général, où une norme n'implique pas la création de pénalités, on s'intéresse aussi au cas plus strict où la notion de norme est remplacée par celle de règle. La règle est la norme « enrégimentée » à laquelle on doit forcément obéir et dont la violation suppose des conséquences précises. Par conséquent, on utilise les normes dans les SMA pour influencer les comportements des agents avec des degrés qui varient en fonction de la nature du problème qu'on veut résoudre.

5.4.1 Représentation

Le rôle d'une description normative est de permettre aux différents acteurs d'un SAN, i.e. les agents, les éléments de contrôle, l'autorité, de déterminer quel est le comportement idéal d'un agent dans une situation donnée. Ainsi, la norme est porteuse d'information et les éléments qui la composent en tant que description sont comme suit :

- type : obligations ou droits
- objet : une action ou une propriété
- condition sur l'objet : par exemple, un intervalle de temps
- sujet : un ensemble non-vide de rôles ou d'agents
- autorité

– auteur
– validité
– contexte d'application : une condition logique
– coût de la violation
– exceptions au contexte de l'application
– exceptions au sujet

Pour décrire un comportement (ou une situation) idéal, plus précisément un comportement souhaitable ou interdit, nous introduisons la notion de *propriété déontique*. Une propriété déontique indique si un agent est obligé, autorisé ou n'a pas le droit d'exécuter une action ou de réaliser l'apparition d'un état α. Les autres composants d'une norme ont le rôle de spécifier le contexte dans lequel la propriété déontique est créée. Par exemple, si le langage choisi pour décrire les normes est celui de la logique (voir les chapitres suivants), pour représenter les trois types de propriétés déontiques, on utilise les prédicats $O(Agent,\alpha)$, $P(Agent,\alpha)$ et $I(Agent,\alpha)$. Dans le même langage logique, une norme peut avoir la forme (réduite) suivante :

$$O^{\mathcal{N}}(Agent,\alpha,i) \longleftarrow condition$$

avec $\mathcal{N}$ - le représentant de l'autorité, i - un intervalle de temps, α - une action et $condition$ - la condition sur le contexte d'application.

5.4.2 Raisonnement

Il est important de noter que, dans un SAN, il existe au moins trois types d'acteurs qui manipulent les normes :

1. le concepteur des normes ;
2. les agents ordinaires ;
3. les éléments de contrôle (ou de monitoring).

Dans le cas du concepteur des normes, il s'agit d'un processus itératif *off-line* ([Shoham et Tennenholtz, 1995; Moses et Tennenholtz, 1995]) très complexe, i.e. NP-complet, dont le résultat est un ensemble cohérent de lois sociales ou normes garantissant l'obtention d'un but général si les normes sont respectées. Les solutions pour implémenter les normes peuvent être soit de type logiciel, soit de type physique. Pour vérifier la cohérence de l'ensemble des normes, par exemple la détection des conflits, on peut utiliser divers outils et techniques de raisonnement (inductif, déontique, temporel, etc.). On note que le format d'une norme est en interrelation directe avec la méthode ou le type de raisonnement employé, qui, à son tour, reflète le contexte et le type de problème qu'on veut résoudre.

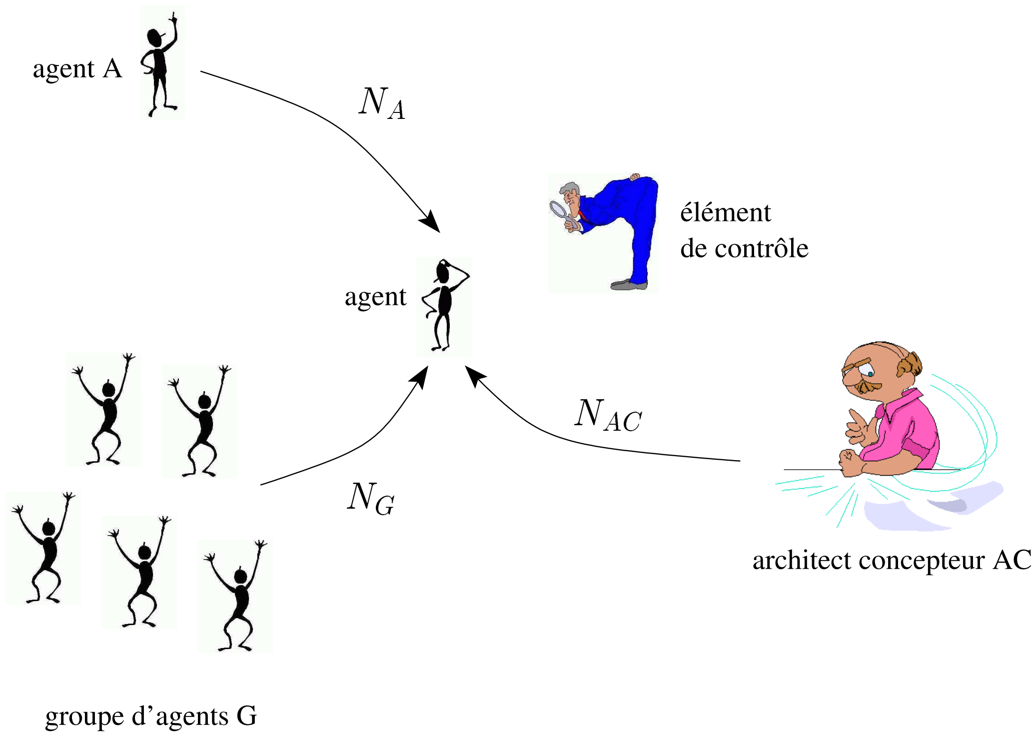


FIG. 5.2 – *Un SAN et ses composants*

Dans le cas des agents ordinaires, il s'agit d'intégrer les informations normatives dans les modules de planification et d'ordonnancement afin d'obtenir un comportement standard ou normatif. Cela implique que les agents soient équipés d'outils de raisonnement et de prise de décisions permettant de :

– détecter les propriétés déontiques ;
– repérer les conflits entre les normes d'origines différentes ou entre les normes et les buts de l'agent ;
– calculer le coût des violations ;
– calculer la situation ou l'action la plus favorable à exécuter, en tenant compte des points précédents ;
– mettre à jour les normes dans le cas d'un changement.

Les éléments de contrôle sont chargés de la détection des comportements anormaux et du calcul de leurs conséquences. En général, la notion de contrôle prend deux formes principales : le contrôle *préventif* et le contrôle *ultérieur*. A titre d'exemple, prenons le cas du contrôle sur le paiement de transport. Le métro parisien applique un contrôle préventif exigeant l'achat d'un billet avant d'entrer dans son réseau. Quant aux bus caenais, il s'agit d'un contrôle ultérieur, où on peut monter dans un bus sans payer, mais si un contrôleur arrive on doit faire la preuve du payment. Dans le cas d'un ordre normatif pur, où on s'intéresse seulement aux comportements normatifs, l'élément de contrôle est presque inutile. Son rôle est réduit à celui d'observateur, son activité principale étant équivalente à celle de monitoring. Donc, il ne s'agit pas d'un véritable contrôle. Par contre, dans la situation où on cherche à imposer un comportement standard par contrainte, i.e. par l'utilisation des pénalités, il s'agit bien d'un contrôle de type ultérieur qui exige que les éléments de contrôle aient des capacités (similaires à celles des agents ordinaires) de :

– détecter les propriétés déontiques ;
– repérer les cas de violation ;
– calculer les pénalités correspondantes.

On remarque que la distinction entre les agents ordinaires et les éléments de contrôle tient compte de leur rôle fonctionnel dans une organisation de type normatif. Dans la pratique, un agent joue souvent le rôle de l'élément de contrôle.

Les formalismes utilisés pour décrire le raisonnement spécifique à chaque type d'acteur dans les SAN sont très divers. Le plus adapté pour le raisonnement à base de concepts déontiques tels que les obligations, les permissions et les interdictions est celui proposé par la logique déontique (voir les chapitres 4 et 6). Le raisonnement normatif ne se réduit pas à la construction d'un ensemble de normes à partir d'un autre ensemble (contrairement au raisonnement dans la logique déontique). Il est plus complexe, car il existe en plus des concepts déontiques d'autres notions pour décrire une norme, telles que l'agence, l'action et le temps. Donc, l'un des problèmes qu'on doit résoudre est de savoir comment on intègre ces concepts pour faciliter la description des normes et le raisonnement en présence de ces dernières. Nous insistons sur cette question dans la deuxième partie de cette thèse.

5.4.3 Interprétation et exécution

Une fois une norme créée par une autorité et envoyée à l'agent concerné, on se pose le problème de savoir comment l'agent va interpréter et accepter la norme. Par définition, un agent normatif est obéissant, c.a.d. qu'il cherche toujours à avoir un comportement conforme au standard. Un ensemble cohérent de normes peut garantir qu'un agent peut avoir toujours un comportement conforme. En réalité les choses ne sont pas aussi simples. Dans le cas des agents enrégimentés, on obtient le comportement normatif par le fait que les agents sont normatifs par construction. Dans le cas des agents autonomes, souvent les agents peuvent avoir des buts personnels qui entrent en conflit avec les contraintes normatives. Ou encore, un agent peut se trouver sous l'influence de plusieurs systèmes normatifs qui ont des buts différents ou contradictoires. Pour résoudre les conflits qui apparaissent entre les normes d'origines différentes, l'agent est forcé de choisir la norme (ou les normes) à laquelle il va obéir et les normes qu'il va violer.

Afin de conserver le caractère normatif d'un agent, même dans le cas de violation, une solution possible est de réparer l'effet d'une norme violée par une autre norme. Dans la logique déontique, ce type de norme est connue sous le nom de Contrary-to-Duty [Prakken et Sergot, 1996] (voir également le chapitre 6) et il est souvent formalisé par une relation de préférence introduite entre plusieurs états idéaux. Une autre solution qui va dans le même sens, c.a.d. qui attribue à chaque norme une valeur ou une préférence, est celle où la valeur est interprétée comme étant le coût à payer en cas de violation. On remarque que la représentation des normes proposée ci-dessus permet de spécifier cette valeur ou coût.

5.4.4 Hypothèses sur l'exécution des normes

Au moment de l'exécution d'un SAN, afin d'obtenir les caractéristiques qu'on a présentées sur la représentation des normes, la communication, le contrôle, etc., on est obligé d'introduire certaines conditions nécessaires.

Premièrement, nous faisons une remarque sur l'utilisation du concept de violation. Cette notion, comme on l'a déjà montré, est liée à la préservation de l'autonomie de l'agent. Celle-ci implique des hypothèses sur l'existence des mécanismes de *monitoring* et de *pénalités*. Si le système de pénalité fonctionne mal ou s'il est inexistant, il existe un risque que les comportements des agents convergent rapidement vers une transgression des interdictions des actions indésirables.

Deuxièmement, afin de permettre la communication entre les agents et l'entité qui transmet les normes, on a besoin de l'hypothèse d'une *syntaxe* et *sémantique* communes. De plus, on ajoute à celle-ci l'hypothèse d'un *langage de communication*, avec un nombre fixe de performatifs, qui présente l'avantage de faciliter le suivi des interactions.

Et enfin, l'utilisation de rôles implique un *test de ressources* (capacités) des agents au moment de leur distribution. La contrainte est plus importante dans les applications liées à la sécurité, où il s'agit de donner des droits (permissions et interdictions) aux agents qui vont jouer un certain rôle « sensible ».

5.5 Conclusions

Les systèmes d'agents normatifs étendent la méthodologie qui utilise la métaphore sociale pour modéliser et concevoir des systèmes d'informations ouverts. Dans ce chapitre nous avons présenté les structures nécessaires qui permettent l'utilisation des normes dans ce sens. Étant donné que la notion de norme est le concept clé dans cette affaire, on énumère les avantages qu'elle peut apporter. Ainsi, la norme :

- permet une description homogène des comportements des agents hétérogènes ;
- permet un contrôle flexible du comportement de l'agent avec divers degrés de liberté, qui peuvent varier d'un comportement complètement déterminé à un comportement totalement autonome ;
- permet d'obtenir une propriété ou un comportement global du système (par émergence) en décrivant des contraintes locales aux agents ;
- permet de décrire un contrôle global et de le décentraliser, c.a.d. de le réaliser localement ;

– assure une modification dynamique des contraintes locales, sans arrêter le système ou les agents (pour les reprogrammer). On peut considérer cette modification dynamique comme étant une façon de « régler » différents paramètres du système afin qu'il parvienne à un état désiré.
– informe l'agent du comportement ou de la situation qu'on veut obtenir de lui. Cette information peut être utilisée par l'agent pour planifier les actions futures.
– peut être intégrée dans l'architecture de l'agent et traitée au même niveau que les autres composantes cognitives[21].
– aide l'agent à résoudre des conflits. Un agent peut préférer une norme à une autre, ou une action ayant une motivation interne (désirée) à une action imposée par une norme.
– est facile à construire si le système à modéliser présente déjà une structure organisationnelle et un règlement de fonctionnement. Cette condition facilite aussi la construction de l'architecture du SAN.
– peut être confrontée à l'exécution courante du système et permet de détecter les déviations éventuelles par rapport au comportement ou à la situation idéale qu'elle décrit. Ce processus peut être réalisé au moment de l'exécution du système (pour assurer un contrôle préventif), ou *a posteriori* (pour effectuer un contrôle ultérieur ou une analyse de type audit).

Évidemment, l'utilisation des normes présente certains inconvénients que nous décrivons dans la suite :

– la représentation des normes n'est pas une chose simple. Les choix qu'on fait dépendent du domaine auquel elles s'appliquent, de qui les utilise et dans quel but. Les problèmes liés au raisonnement influencent aussi ce choix (par exemple, le choix d'utiliser ou non la logique déontique pour décrire et raisonner sur les comportements obligatoires).
– la gestion des normes et leur maintenance augmentent en difficulté avec leur nombre ;
– si plusieurs normes réglementent divers comportements pour assurer l'apparition d'une même propriété, il est difficile de savoir quel est le « réglage » optimal, c.a.d. quelle norme on doit modifier, à quel degré, afin d'obtenir la propriété globale désirée.
– les préférences utilisées ou le coût afférent aux normes, en tant que valeurs quantitatives ou qualitatives, expriment un choix subjectif de celui qui les conçoit. Ce n'est pas un inconvénient majeur, c'est plutôt une remarque, car on rencontre le même problème dans les systèmes experts.

21. On trouve dans la littérature des travaux qui étudient l'utilisation des normes seulement dans le cas des agents rationnels qui doivent d'abord les « accepter consciemment » [Dignum *et al.*, 2000]. En revanche, l'utilisation des normes dans un contexte d'agents réactifs reste pour l'instant un sujet controversé. Dans le chapitre 4, nous avons montré que la normativité et la rationalité sont deux concepts orthogonaux. Le modèle architectural d'un SAN utilise ainsi un modèle d'agent très général, qui n'implique pas la présence des attributs cognitifs, finalement l'intérêt étant d'étudier l'impact des normes dans l'interaction multi-agents.

– dans le cas des agents hétérogènes, les normes doivent manipuler des concepts qui sont compréhensibles et représentables côté agent. Pourtant, ce n'est pas un problème spécifique des SAN, car il concerne plus généralement tout le domaine des SMA.

Dans les chapitres suivants, nous présentons les modèles et les outils permettant de construire les différents composants présents dans un SAN en tenant compte aussi des aspects qu'on a vus jusqu'ici.

Deuxième partie

Formalismes et outils pour l’intéraction normative

Chapitre 6

Formalismes déontiques

6.1 Introduction

Les normes sont les éléments principaux manipulés par les composants d'un système normatif. Leur représentation fait intervenir des expressions qu'on appelle *déontiques*, telles que les mots obligation, devoir, permission, pouvoir, droit, etc. Dans cette partie nous présentons des aspects liés à la formalisation de ces concepts, qu'on trouve dans la littérature groupés sous le nom de *logique déontique*.

6.2 Problématique

On peut dire que la logique déontique est née avec l'apparition de l'article [von Wright, 1951] écrit par von Wright au début des années cinquante. Tous les travaux qui ont suivi dans le domaine de la logique déontique sont plus ou moins marqués par les idées présentées dans cet article.

La logique déontique essaie d'étudier les propriétés des notions déontiques. Deux caractéristiques des obligations sont considérées comme acceptables par la plupart des chercheurs dans ce domaine. La première concerne le principe selon lequel tout ce qui est obligatoire est aussi permis. La justification de ce principe est le fait qu'on ne peut pas obliger quelqu'un à faire quelque chose, si on ne lui donne pas en même temps la permission de le faire. Dans un langage formel, ce principe est exprimé par $O\alpha \Rightarrow P\alpha$ et se lit « si α est obligatoire, alors α est permit ».

La deuxième caractéristique décrit le lien qui existe entre la notion d'obligation et celle de possibilité. On se demande si toute obligation utilise comme hypothèse la possibilité matérielle de la réaliser. Formellement, si on note avec $\diamond$ l'opérateur modal de possibilité, ce principe est exprimé par la formule $O\alpha \Rightarrow \diamond\alpha$. Un axiome dérivé de ce principe est celui de $\neg O\bot$, qui dit que l'impossible n'est pas obligatoire. Le symbole $\bot$ représente la contradiction $\alpha \wedge \neg\alpha$.

Une troisième caractéristique a été proposée, l'hypothèse de l'absence de conflit, mais qui n'est pas assumée par tous les systèmes de logique déontique. Cette propriété exprime le fait que si une propriété est obligatoire, alors elle ne peut pas être en même temps interdite. Formellement, on exprime ce principe par la formule : $\neg(O\alpha \wedge \neg O\alpha)$.

Sinon, pour ce qui reste du domaine de la logique déontique on rencontre beaucoup de divergences autour des autres propriétés des concepts déontiques et de la manière de les représenter. Même si ces problèmes sont plus d'ordre philosophique, on trouve intéressant de les présenter (même brièvement) pour montrer les difficultés que les démarches de conceptualisation et de formalisation de ces concepts impliquent.

Le premier problème concerne la question de savoir si une norme est porteuse ou non de valeur de vérité. Pour donner une réponse à cette question, on doit faire la distinction entre les normes et les propositions normatives. D'abord, les normes ne sont pas dépendantes d'un langage quelconque. Par contre, elles peuvent être exprimées par des moyens linguistiques, mais tout en restant indépendantes de toute expression linguistique. En ce qui concerne les propositions normatives, elles décrivent le résultat de l'application d'une norme, par exemple, une proposition qui exprime le fait qu'il existe une obligation selon une norme ou un ensemble de normes non-specifiées. Pour montrer la différence entre l'aspect prescriptif (les normes) et l'aspect déclaratif (les propositions normatives) d'une expression normative, Alchourrón [Alchourrón, 1993] propose l'exemple suivant. Selon lui, on peut se représenter les caractéristiques déontiques (par exemple l'obligation ou l'interdiction) avec des boîtes vides qu'on peut remplir. Quand une autorité utilise les propositions déontiques pour créer (ou prescrire) une norme, on peut considérer cette activité comme identique à celle qui consiste à mettre quelque chose dans les boîtes. C'est une façon de construire une partie de la réalité. Quand quelqu'un utilise les propositions déontiques d'une façon descriptive, on peut considérer cette activité comme similaire à celle qui consiste à dessiner celui qui met des choses dans les boîtes ou à faire une photo de cette partie de la réalité. Ainsi, seulement les propositions normatives sont porteuses de valeur de vérité.

Le deuxième problème concerne la manière de formaliser les notions déontiques. Par exemple, le débat porte sur le choix entre une démarche de premier ordre et une démarche modale. Le système présenté initialement par von Wright est construit dans une logique de premier ordre, mais la plupart des travaux qui ont suivi ont développé la logique déontique comme une branche de la logique modale. Cela s'explique par le fait qu'il y a une forte similitude entre les notions déontiques d'obligation, de permission et d'interdiction d'une part, et les notions de nécessité, de possibilité et d'impossibilité d'autre part. Von Wright remarque que, malgré toutes ces similitudes, il existe quelques différences. Par exemple, il n'existe pas dans la logique déontique un principe similaire à celui de nécessité $\Box\alpha \Rightarrow \alpha$, car intuitivement, ce qui est obligatoire ne l'est pas forcément toujours.

Le troisième problème de la logique déontique est de préciser quels sont les éléments auxquels s'appliquent les notions déontiques[22]. Initialement, von Wright a utilisé les opérateurs déontiques pour caractériser les propriétés normatives des actions. Par exemple, $O\alpha$ se lit « quelqu'un doit exécuter α », O étant un prédicat et α, un terme qui exprime le type d'acte. Ensuite, von Wright a changé d'avis et il a donné une autre interprétation de l'expression $O\alpha$, dans une approche modale, O étant un opérateur modal et α une proposition, qui se lit « quelqu'un doit assurer l'apparition de α ». Le principal avantage de cette écriture est qu'on peut utiliser dans une formule des propositions et des opérateurs sur les propositions, par exemple pour décrire la violation[23] : $O\alpha \wedge \neg\alpha$. Son inconvénient est qu'elle permet d'imbriquer plusieurs opérateurs pour construire une formule du type $O(O(O(...(O\alpha)...)))$, qui n'a pas de signification intuitive : « il est obligatoire qu'il est obligatoire qu'il est obligatoire ... que α ».

Le dernier problème concerne les paradoxes de la logique déontique. Avant de les présenter, on doit introduire les éléments principaux qui composent la logique déontique modale.

6.3 Logique déontique modale

La logique déontique modale est construite comme une extension de la logique propositionnelle classique à laquelle on ajoute un ou plusieurs opérateurs déontiques. Plus précisément, le langage de logique déontique modale est formé par un ensemble Φ de variables propositionnelles et des connecteurs booléens habituels $\neg$ et $\Rightarrow$. De plus, si O, P et I désignent les opérateurs déontiques et si α est une formule, alors $O\alpha$, $P\alpha$ et $I\alpha$ sont des formules, qui ont la traduction habituelle dans un langage naturel : « il est obligatoire que α », « il est permis que α » et « il est interdit que α ». Entre les différents opérateurs on peut décrire les relations suivantes :

1. D'abord, on pourrait considérer que l'opérateur I peut être défini à partir de l'opérateur O, car entre les deux il existe la relation :
$$I\alpha =_{def} O\neg\alpha \tag{6.1}$$
2. L'opérateur P est le *dual* de l'opérateur O. On peut également considérer la relation suivante entre les opérateurs O et P qui relie les notions d'obligation et de permission de la même manière que les opérateurs de nécessité et de possibilité sont reliés dans la logique modale classique :
$$P\alpha =_{def} \neg O\neg\alpha \tag{6.2}$$

Le système de logique modale contient tous les axiomes et toutes les règles d'inférences de la logique propositionnelle. Il contient donc l'ensemble des tautologies et il est fermé sous la règle d'inférence du

22. Ce problème est appelé dans la littérature déontique *Tunsollen* versus *Seinsollen*, ou en anglais *ought-to-do* versus *ought-to-be*.

23. Ce qui n'est pas le cas pour le modèle initial de von Wright parce que α dans $O\alpha$ n'est pas une proposition mais un type d'acte.

Modus Ponens **MP**

$$\frac{\alpha \Rightarrow \beta \qquad \alpha}{\beta} \tag{6.3}$$

de nécessité **NR**

$$\frac{\alpha}{O\alpha} \tag{6.4}$$

et l'axiome **K**

$$O(\alpha \Rightarrow \beta) \Rightarrow (O\alpha \Rightarrow O\beta) \tag{6.5}$$

Jusqu'ici, la logique déontique que nous avons définie a toutes les caractéristiques d'un système de logique modale *normal*. La logique déontique *standard* (SDL) contient en plus l'axiome **D** qui relie les notions d'obligation et de permission. Cet axiome correspond, comme on l'a montré auparavant, à la volonté d'inclure dans la logique le fait qu'une obligation doive impliquer la permission correspondante :

$$O\alpha \Rightarrow P\alpha \tag{6.6}$$

La sémantique de la logique déontique est celle qu'on utilise couramment pour la logique modale, c.a.d. la sémantique de Kripke, ou encore la sémantique des mondes possibles [Kripke, 1963]. Un modèle de Kripke M pour un système de logique modale normal est un triplet $< W,R,V >$, où W est un ensemble de mondes possibles w, R est une relation binaire sur W appelée la relation d'accessibilité, et $V : W \times \Phi \rightarrow \{vrai,faux\}$ est une fonction de valuation qui donne, pour chaque monde $w \in W$, la valeur de vérité $V(W,\alpha)$ de la proposition atomique α. Quand un monde w est en relation avec w', on dit que w' est directement accessible depuis w, ou que w' est un monde possible pour w. On note avec $M,w \models \alpha$ le fait que la proposition α soit vraie dans un monde w dans le modèle M. Cette valeur de vérité est définie de la manière suivante, qui étend le calcul propositionnel habituel :

- si $\alpha \in \Phi$ alors $M,w \models \alpha$ ssi $V(W,\alpha) = vrai$;
- $M,w \models \neg\alpha$ ssi on n'a pas $M,w \models \alpha$;
- $M,w \models \alpha \vee \beta$ ssi $M,w \models \alpha$ ou $M,w \models \beta$;
- $M,w \models O\alpha$ ssi pour tous les mondes accessibles w' avec $R(w,w')$, il est vrai que $M,w' \models \alpha$.

Si on considère la définition de P (6.2), on peut construire facilement la définition de sa valeur de vérité :

- $M,w \models P\alpha$ ssi il existe $w' \in W$ avec $R(w,w')$ et qu'il est vrai que $M,w' \models \alpha$.

Dans le cas de SDL incluant l'axiome **D**, on peut également remarquer que la relation R entre les mondes du modèle de Kripke est une relation *sérielle*, c.a.d. qui a la propriété :

$$\forall w \in W, \exists w' \in W \text{ tel que } R(w,w')$$

qui exprime que tout monde peut accéder à un autre monde.

6.4 Paradoxes

Une fois le formalisme logique afférent introduit on peut parler des paradoxes de la logique déontique. L'apparition de ces paradoxes s'explique par le fait que les propriétés logiques des systèmes formels utilisés pour formaliser les concepts déontiques induisent des caractéristiques qui ne sont pas intuitives.

6.4.1 Affaiblissement déontique

Les paradoxes les plus connus sont ceux qui sont provoqués par la propriété d'*affaiblissement déontique*. Par exemple, on peut dériver des obligations moins fortes avec $O(\alpha) \Rightarrow O(\alpha \vee \beta)$ ou avec son équivalent $O(\alpha \wedge \beta) \Rightarrow O\alpha$ qui sont des théorèmes pour SDL[24]. Pour la première règle, Ross donne un exemple (connu sous le nom de paradoxe de Ross) qui montre qu'elle n'est pas naturelle : si on doit envoyer une lettre, alors on doit l'envoyer ou la brûler. Dans le même registre, on connaît le paradoxe du libre arbitre, généré par $P(\alpha) \Rightarrow P(\alpha \vee \beta)$: si quelqu'un a la permission de fumer, alors il a la permission de fumer ou de tuer.

D'autres paradoxes sont apparus à cause de l'introduction des obligations conditionnelles formalisées par la formule $\beta \Rightarrow O\alpha$ où β est la condition et α est la conclusion déontique[25]. Les obligations conditionnelles permettent l'utilisation d'un type spécial d'obligation appelée en anglais *Contrary-To-Duty* (CTD). Les obligations CTD (ou *secondaires*) introduisent une description complémentaire à une obligation *primaire* ou *prima facie*. Par exemple, on peut avoir l'obligation primaire $O\alpha$ et son obligation CTD $\beta \rightarrow O\alpha'$, où β et en contradiction avec α. En d'autres termes, l'obligation primaire décrit le cas le plus fréquent, et l'obligation CTD s'applique quand l'obligation primaire n'est pas respectée. Comme exemple de paradoxe, on présente le paradoxe de Forrester.

6.4.2 Le paradoxe de Forrester

Considérons les propositions suivantes dans une théorie SDL :

1. $O\neg t$: Alice ne doit pas tuer Bob ;
2. $t \Rightarrow O(t \wedge g)$: Si Alice tue Bob, alors elle doit le faire avec gentillesse ;
3. t : Alice tue Bob.

La deuxième obligation est une obligation CTD de la première obligation, parce que $\neg t$ et t sont contradictoires. De plus, la logique SDL permet ce qu'on appelle le *détachement factuel*, exprimé par la

24. On peut les obtenir à partir de l'affaiblissement propositionnel $\alpha \Rightarrow \alpha \vee \beta$ et l'application de **NR** et **K**.
25. On trouve dans la littérature d'autre notations comme $O(\beta \rightarrow \alpha)$ ou celles qui utilisent un opérateur dyadique du type $O(\alpha|\beta)$.

formule :

$$(\beta \wedge (\beta \Rightarrow O\alpha)) \Rightarrow O\alpha \tag{6.7}$$

En utilisant cette formule et les deux dernières propositions, on obtient que $O(t \wedge g)$, qui nous permet de dériver que Ot. Mais ce résultat dans SDL est contradictoire avec la première proposition $O\neg t$. A partir de l'axiome **D** et de la définition de $P\alpha$ (6.2), on obtient que dans SDL $\neg(O\alpha \wedge O\neg\alpha)$ est un théorème. Le paradoxe consiste donc en la présence d'une contradiction dans un ensemble de propositions qui selon notre intuition est cohérent.

6.4.3 Le paradoxe de Chisholm

Le plus celébre paradoxe de SDL est le paradoxe de Chisholm [Chisholm, 1963]. Il comporte trois obligations conditionnelles et une observation factuelle :

1. Oa : Il est obligatoire que Jean aide ses voisins ;
2. $O(a \Rightarrow t)$: Il est obligatoire que si Jean va aider ses voisins, il leur téléphone pour leur dire qu'il vient ;
3. $\neg a \Rightarrow O\neg t$: Si Jean ne va pas aider ses voisins, alors il ne doit pas leur téléphoner pour leur dire qu'il vient ;
4. $\neg a$: Jean ne va pas aider ses voisins.

Si les quatre propositions semblent être cohérentes, leur formalisation en SDL ne l'est pas. En utilisant l'axiome **K** et les propositions 1 et 2, on obtient Ot. Si on applique la règle du *modus ponens* et les propositions 3 et 4, on obtient que $O\neg t$, ce qui est en contradiction avec Ot dans SDL, comme on l'a montré ci-dessus. Ce paradoxe reflète les problèmes posés par le fait qu'on peut obtenir une obligation non-conditionnelle à partir d'une obligation conditionnelle de deux façons différentes. La première utilise le détachement factuel de $O\alpha$ à partir de β et $\beta \Rightarrow O\alpha$. L'autre utilise le *détachement déontique* de $O\alpha$ à partir de $O\beta$ et $O(\beta \Rightarrow \alpha)$. Dans SDL, les deux types de détachement sont valides et provoquent l'inconsistance de l'ensemble de propositions 1-4.

On peut se demander si d'autres formalisations possibles des propositions 2 et 3, par exemple, celle où on utilise seulement l'implication matérielle de la forme $\beta \Rightarrow O\alpha$ (qui remplace donc $O(\beta \Rightarrow \alpha)$), peuvent apporter une solution. La réponse est négative car les propositions ne sont plus indépendantes du point de vue logique : la proposition 4 implique, par affaiblissement propositionnel, que $a \Rightarrow Ot$. Idem pour la formalisation avec seulement l'implication du type $O(\beta \Rightarrow \alpha)$: la proposition 1 implique par affaiblissement déontique que $O(\neg a \Rightarrow \neg t)$.

Les solutions trouvées pour éliminer ces paradoxes sont multiples et en présenter une synthèse dépasserait largement le sujet de cette thèse. On se contente donc de remarquer ici que pour éliminer la propriété d'affaiblissement déontique ou de détachement factuel dans les obligations conditionnelle les différents travaux proposent en général des solutions qui modifient la structure des mondes possibles. Par exemple, on ajoute un ordre de préférence ou une hiérarchie entre ces mondes qui permettent une plus grande flexibilité pour définir les situations où une obligation est vraie [van Eck, 1982; Hansson, 1990; Alchourrón, 1993; Horty et Belnap, 1995].

6.5 Actions dans la logique déontique

Nous avons montré que, dans la logique déontique, l'un des problèmes est de préciser à quoi s'appliquent les opérateurs déontiques, à des actions ou à des propositions, problème qui a donné naissance aux logiques des obligations *ought-to-do* (qui expriment l'implication d'un agent), et respectivement *ought-to-be* (qui décrivent une situation).

Dans la littérature déontique, on trouve deux types d'approches pour exprimer l'obligation « l'agent i doit faire α ». Le premier type, développé au sein des théories de *stit*[26] interprète l'obligation $O_i\alpha$ par « l'agent i doit s'assurer que α » [Kanger, 1971; Pörn, 1977; Horty et Belnap, 1995]. Formellement, on a comme définition :

$$O_i\alpha = OE_i\alpha$$

Ici α represente une proposition et l'opérateur O_i s'applique à toute proposition. En effet, l'opérateur O_i qui exprime une obligation personalisée, l'agent i étant le porteur de cette obligation, est obtenu par l'application d'une obligation générale (et impersonnelle) à la proposition $E_i\alpha$. Cette proposition se lit « l'agent i assure l'apparition de α ». Les propriétés de l'opérateur modal stit E_i sont exprimées par l'intermediaire des axiomes suivants :

E.RE

$$\frac{\alpha \equiv \beta}{E_i\alpha \equiv E_i\beta}$$

E.T

$$E_i\alpha \Rightarrow \alpha$$

L'axiome E.T indique qu'il s'agit d'une action reussie, c.a.d. que si l'agent i assure l'apparition de α alors on obtient α. Une autre façon d'interpréter cet axiome est de dire que x est responsable de l'apparition de α seulement si α est vraie.

L'axiome E.RE dit que si un agent est responsable de la réalisation d'une certaine propriété, alors il est aussi responsable de l'apparition de toute propriété logiquement equivalente.

Dans certaines autres théories, l'opérateur peut être doté avec les proprietés :

no$\top$

$$\neg E_i\top$$

26. Acronyme en anglais pour *sees to it that*.

E.C

$$(E_i\alpha \wedge E_i\beta) \Rightarrow E_i(\alpha \wedge \beta)$$

L'axiome no$\top$ explique qu'aucun agent ne peut réaliser ce qui est logiquement vrai [27].

Le deuxième type d'approche est celui où les actions ont un caractère autonome et sont exprimées par des termes de premier ordre. Pour illustrer cela, nous présentons la réduction de la logique déontique à la logique dynamique, proposée par Meyer [Meyer, 1988]. Meyer utilise une logique propositionnelle dynamique, c.a.d. une logique propositionnelle étendue avec l'opérateur modal de nécessité $[\alpha]$ pour toute action α définie dans le langage et de son dual, de possibilité, $< \alpha >$. Dans cette logique, la formule $[\alpha]\phi$ se lit « l'exécution de l'action α assure nécessairement l'apparition de la propriété ϕ ». La sémantique du langage utilise des structures Kripke et des relations d'accessibilité R_α associées à toute action α. Les définitions des concepts déontiques sont comme suit, où V représente un état indiquant si une violation s'est produite par rapport à l'action (ou au but) α et les contraintes déontiques :

- $F\alpha \equiv [\alpha]V$: l'exécution d'une action est interdite si dans tous les états qui suivent son exécution, le prédicat V est vrai ;
- $P\alpha \equiv \neg F\alpha$: la permission est identique à la non-interdiction ;
- $O\alpha \equiv F{-}\alpha$: l'obligation est l'interdiction de la négation d'une action (notée avec $-\alpha$) [28].

Dans la description des actions, on peut utiliser des opérateurs pour obtenir des actions complexes. Par exemple, « ; » représente la composition séquentielle, « $\cup$ » le choix non-déterministe et « $\&$ » la composition parallèle des actions. La description utilisant la réduction ci-dessus pour les notions déontiques et celle pour les actions complexes, permet d'obtenir les formules valides suivantes :

$$F(\alpha;\beta) \equiv [\alpha]F\beta \tag{6.8}$$

$$F(\alpha \cup \beta) \equiv (F\alpha \wedge F\beta) \tag{6.9}$$

$$(F\alpha \vee F\beta) \Rightarrow F(\alpha\&\beta) \tag{6.10}$$

$$O(\alpha;\beta) \equiv (O\alpha \wedge [\alpha]O\beta) \tag{6.11}$$

$$(O\alpha \vee O\beta) \Rightarrow O(\alpha \cup \beta) \tag{6.12}$$

$$O(\alpha\&\beta) \equiv (O\alpha \wedge O\beta) \tag{6.13}$$

$$P(\alpha;\beta) \equiv < \alpha > P\beta \tag{6.14}$$

27. On remarque que c'est un résultat discutable (voir [Horty et Belnap, 1995; Sergot, 1999a]).

28. La notion de négation d'une action pose des problèmes de formalisation. Ici, on l'utilise plutôt pour exprimer le concept de non-exécution de l'action α. C'est un choix discutable, car il est difficile de s'imaginer l'action de ne pas exécuter une autre action. Le symbole $-$ est diffèrent de celui de la négation logique $\neg$ pour montrer qu'il s'applique seulement à des termes représentant des actions.

$$P(\alpha \cup \beta) \equiv (P\alpha \vee P\beta) \tag{6.15}$$

$$P(\alpha \& \beta) \Rightarrow (P\alpha \wedge P\beta) \tag{6.16}$$

$$O(\alpha \cup \beta) \wedge F\alpha \wedge P\beta \Rightarrow O\beta \tag{6.17}$$

L'introduction d'une logique dynamique à la place d'une logique propositionnelle n'élimine que certains des paradoxes de la logique déontique (parce qu'on ne peut plus les exprimer, par exemple, $O\alpha \Rightarrow OO\alpha$). Les paradoxes concernant l'affaiblissement déontique subsistent car cette propriété est valable dans la logique de Meyer. Par exemple, on peut faire une analogie entre le choix non-déterministe et la disjonction propositionnelle $\alpha \Rightarrow \alpha \vee \beta$, et entre l'exécution parallèle et la conjonction $\alpha \wedge \beta \Rightarrow \alpha$, respectivement :

$$O\alpha \Rightarrow O(\alpha \cup \beta)$$
$$O(\alpha \& \beta) \Rightarrow O(\alpha)$$

6.6 Formalismes pour la théorie des positions normatives

Dans le chapitre 4, nous avons évoqué les efforts de formalisation des concepts juridiques de Hohfeld, datant du début du siècle dernier [Hohfeld, 1913]. Nous continuons ici la description des formalismes utilisés, qui sont basés sur la notion déontique d'obligation et des résultats obtenus dans la logique déontique[29].

Le but des travaux de Hohfeld était de construire le raisonnement dans la jurisprudence sur des bases plus précises et rigoureuse, et donc plus correctes. Malheureusement, à cette époque, Hohfeld ne disposait pas de méthodes qui sont aujourd'hui disponibles aux logiciens pour offrir un outil à l'analyse des droits. Kanger [Kanger, 1971] a été le premier qui a tenté de construire une formalisation logique de la théorie de Hohfeld. Pour définir les concepts juridiques fondamentaux de Hohfeld, il a utilisé deux opérateurs logiques : un pour l'obligation (comme celui de la logique déontique O) et l'autre pour l'action (l'opérateur *stit* E_i).

Tout d'abord, Kanger propose les utilisations suivantes du terme « droit » :

1. i a le droit de récupérer l'argent emprunté à j.
2. i a le droit de se déplacer dans le magasin de j quand il est ouvert.
3. i a le droit de donner tout son argent à j.
4. i a le droit de se déplacer dans la rue à l'extérieur du magasin de j.

29. Nous ne faisons qu'explorer un domaine très vaste. Pour une description plus complète, nous suggérons [Krogh, 1996a] dont nous nous sommes inspirés dans cette partie.

pour illustrer les concepts de droit d'exiger, la liberté, le pouvoir, et respectivement l'immunité. En utilisant les deux opérateurs O et E_i, ces relations peuvent être définies par :

$$Droit(i,j,\alpha) \equiv O(E_j\alpha) \tag{6.18}$$
$$Liberte(i,j,\alpha) \equiv \neg O(E_i\neg\alpha) \tag{6.19}$$
$$Pouvoir(i,j,\alpha) \equiv \neg O\neg(E_i\alpha) \tag{6.20}$$
$$Immunite(i,j,\alpha) \equiv O\neg(E_j\alpha) \tag{6.21}$$

Pour les autres droits définis par Hohfeld, le sans droit, le devoir, l'incapacité et la responsabilité, on utilise les définitions suivantes :

$$SansDroit(i,j,\alpha) \equiv \neg O(E_j\neg\alpha) \tag{6.22}$$
$$Devoir(i,j,\alpha) \equiv O(E_i\alpha) \tag{6.23}$$
$$Incapacite(i,j,\alpha) \equiv O\neg(E_i\neg\alpha) \tag{6.24}$$
$$Responsabilite(i,j,\alpha) \equiv \neg O\neg(E_j\alpha) \tag{6.25}$$

Notons que la proposition de Kanger d'utiliser l'opérateur modal O qui exprime une obligation générale ne décrit pas exactement la relation juridique initialement introduite par Hohfeld, qui est définie entre le porteur du droit et sa contrepartie. Par exemple, le droit de i d'exiger de j l'apparition d'un état de choses α est expliqué comme étant l'équivalent du devoir de j envers i de réaliser α. Ce devoir est appelé *obligation orientée* de son porteur j envers i, sa contrepartie. Par conséquent, la démarche de Kanger a été critiquée sous cet angle et d'autres solutions ont été proposées, comme celle présentée dans [Krogh, 1996a] qui introduit l'opérateur ${}_iO_j(\alpha)$ pour l'obligation orientée. Cet opérateur se lit « i a l'obligation envers j d'assurer que α soit vrai ». Sa définition est donnée par :

$${}_iO_j(\alpha) \stackrel{def}{=} {}_iO(E_i\alpha) \wedge O_j(E_i\alpha)$$

où ${}_iO(\alpha)$ est l'opérateur pour exprimer une obligation personnelle qui se lit « i est obligé que α » et $O_i(\alpha)$ est l'opérateur pour exprimer ce qui est idéal pour un agent (du point de vue de la loi).

Kanger, puis Lindhall, dans [Lindahl, 1977] ont continué l'analyse de Hohfeld concernant les relations entre les droits. Par exemple, Kanger a observé qu'il était possible de combiner plusieurs explications des droits et il a ainsi défini ce qu'il appelle les *types atomiques de droits*. Ils sont décrits [Makinson, 1986] par des expressions appartenant à l'ensemble :

$$[\![\pm O \pm \begin{pmatrix} E_i \\ E_j \end{pmatrix} \pm \alpha]\!] \tag{6.26}$$

Les expressions qui se trouvent entre crochets expriment 16 propositions de la forme : $\pm O \pm E_i \pm \alpha$ ou $\pm O \pm E_j \pm \alpha$, où $\pm$ exprime les deux possibilités pour l'affirmation ou la négation et les parenthèses décrivent les deux alternatives E_i et E_j connues sous le nom de *schéma-choix*. $[\![\Phi]\!]$ exprime l'ensemble de *maxi-conjonctions* d'un ensemble d'expressions Φ, c.a.d. toutes les conjonctions possibles formées par des expressions de Φ dont on élimine les membres inconsistants[30] et redondants (sans répétition). On remarque que par construction, les maxi-conjonctions (6.26) sont consistantes, mutuellement exclusives et que leur disjonction est une tautologie. On conclut donc que dans une situation donnée une seule maxi-conjonction est vraie (conformément aux principes logiques considérés).

L'ensemble défini par (6.26) est appelé dans la terminologie de Kanger l'ensemble de *positions normatives pour deux agents*. Il peut s'écrire d'une façon équivalente [Sergot, 1999b] comme une conjonction d'expressions plus simples de type *position normative pour un seul agent* (dans la terminologie de [Jones et Sergot, 1993]) :

$$[\![\pm O \pm \begin{pmatrix} E_i \\ E_j \end{pmatrix} \pm \alpha]\!] = [\![\pm O \pm E_i \pm \alpha]\!] \cdot [\![\pm O \pm E_j \pm \alpha]\!] \tag{6.27}$$

La notation $\cdot$ dans (6.27) exprime l'ensemble de toutes les conjonctions consistantes qui peuvent se former en mettant ensemble les expressions décrites par ses arguments.

La maxi-conjonction qui correspond aux positions normatives pour un seul agent

$$[\![\pm O \pm E_i \pm \alpha]\!] \tag{6.28}$$

est composée de 6 éléments[31] (conformément aux logiques correspondant à O et E_i)

$$PE_i\alpha \wedge PE_i\neg\alpha \tag{6.29}$$
$$O\neg E_i\alpha \wedge O\neg E_i\neg\alpha \tag{6.30}$$
$$OE_i\alpha \tag{6.31}$$
$$PE_i\alpha \wedge P\neg E_i\alpha \wedge O\neg E_i\neg\alpha \tag{6.32}$$
$$OE_i\neg\alpha \tag{6.33}$$
$$O\neg E_i\alpha \wedge PE_i\neg\alpha \wedge P\neg E_i\neg\alpha \tag{6.34}$$

qui sont, comme on l'a déjà vu, consistants, mutuellement exclusifs et dont la disjonction est une tautologie. Dans n'importe quelle situation une de ces expressions est vraie.

La maxi-conjonction qui décrit les types atomiques de Kanger (6.26) contient par conséquence $6 \times 6 = 36$ conjonctions, dont 10 sont inconsistantes. En conclusion, les 26 éléments dans l'analyse de Kanger décrivent exhaustivement les droits entre deux agents par rapport à un état de choses, dans toutes les circonstances possibles.

30. Conformément aux propriétés des opérateurs modaux O et E_i.
31. On rappelle que dans la logique déontique, P est le dual de O : $P\alpha =_{def} \neg O\neg\alpha$.

Dans [Sergot, 1999b], Sergot essaie de généraliser la description des positions normatives pour m agents et n états. Il propose de construire l'ensemble :

$$\left\langle \pm O \pm \begin{pmatrix} E_i \\ \vdots \\ E_j \end{pmatrix} \pm \begin{pmatrix} \alpha \\ \vdots \\ \beta \end{pmatrix} \right\rangle \tag{6.35}$$

où

$$\langle \pm O \pm \binom{E_i}{E_j} \pm \alpha \rangle \stackrel{def}{=} \left[\!\!\left[\pm O \pm [\![\pm \binom{E_i}{E_j} \pm \alpha]\!] \cdot [\![\pm\alpha]\!] \right]\!\!\right] \tag{6.36}$$

Dans le cas où il s'agit de deux agents et d'un seul état, l'expression (6.36) génère 255 positions normatives. Par conséquent, pour m agents et n états l'expression (6.35) génère $2^{2^{(m+n)n}} - 1$ positions normatives. Pour réduire cette complexité, Sergot propose une méthode de génération mixte qui nécessite la présence d'un utilisateur humain. Le processus de génération est réalisé en plusieurs étapes ; à chaque étape l'utilisateur humain peut affiner l'ensemble des positions normatives générées. Il existe une implémentation logicielle de cette méthode, appelée Norman-G [Sergot, 1999a].

6.7 Discussions et conclusions

Dans ce chapitre nous avons présenté la façon dont on essaie de formaliser les concepts déontiques qui expriment l'obligation, l'interdiction et la permission. Leurs propriétés, qui sont basées sur une pratique dans les domaines juridique, éthique ou de la philosophie, ne sont pas toujours facile à exprimer dans un formalisme logique. Les paradoxes, comme les autres problèmes que nous avons évoqués concernant la logique déontique, montrent les difficultés dont cette démarche fait preuve.

Les notions déontiques peuvent s'appliquer très naturellement pour décrire les comportements idéaux des agents, d'où l'intérêt dans cette thèse pour ces concepts. Les problèmes que nous allons essayer de traiter plus tard concernent d'une part la manière d'adapter ces concepts pour exprimer les normes dans un contexte multi-agents, et d'autre part, la manière de les intégrer dans un formalisme temporel, car dans le modèle que nous proposons dans la suite, l'exécution d'une action par un agent, ainsi que la description normative d'un comportement doivent prendre en compte le passage du temps.

On remarque aussi que le raisonnement normatif n'est pas nécessairement basé sur la logique déontique. La logique déontique peut nous indiquer seulement quelles sont les obligations qu'on peut dériver à partir d'un ensemble d'autres obligations. Plus précisément, on peut montrer quelles sont les relations logiques entre obligations et ensuite préciser quelles sont les effets directs des normes. Par exemple, pour qu'un agent planifie ses actions il doit anticiper les actions des autres. Pour des raisons d'efficacité, on présume que les autres agents vont obéir aux normes. On peut considérer ainsi, qu'il existe une relation causale entre les actions qui vont se réaliser et les actions qui doivent se réaliser. Sans cette hypothèse, la logique déontique ne peut pas être utilisée dans le raisonnement dans la planification. Lorsqu'on étudie les obligations dans le monde des agents autonomes, cette hypothèse change : on permet aux agents de désobéir aux normes pour préserver leur autonomie. Par conséquent, on doit revoir quels principes logiques déontiques on conserve et comment on peut les enrichir avec d'autres concepts afin de les utiliser dans un contexte plus riche.

Certains choix concernant le formalisme que nous allons utiliser peuvent être annoncés dés à présent, car ils correspondent aux principes présentés dans la première partie.

– Les normes que nous utilisons règlemenent le comportement des agents. Ainsi, les obligations, en tant que concepts déontiques, vont s'appliquer à des actions.
– Ce choix facilite le type de démarche logique, qui sera exprimé par un formalisme de premier ordre. Donc, les obligations, les permissions et les interdictions seront représentées par des prédicats de premier ordre.
– Pour l'instant, pour des raisons d'économie logique, les relations exprimées par ces prédicats seront non-interprétées, c.a.d. qu'on ne fait aucun lien entre les définitions de deux concepts déontiques, chacun ayant un statut autonome (pas de logique déontique). La motivation principale de ce choix est qu'on peut avoir un agent qui est en même temps obligé d'exécuter une action et à qui on a interdit d'exécuter la même action[32].
– Les normes doivent permettre de décrire les comportements dans le temps. Ce choix doit être compatible avec les choix précédents pour pouvoir intégrer dans le même formalisme, les obligations, les actions et le temps.
– On s'intéresse plus à l'aspect opérationnel du formalisme, le but étant de produire des outils logiques pour l'automatisation du raisonnement dans un contexte normatif (planification, détection de violations).

Avant de présenter les résultats de nos efforts de formalisation, nous décrivons dans le chapitre suivant la façon de formaliser un autre grand participant à la description des normes, le temps.

32. Ce choix n'est plus pertinent lorsqu'on veut décrire un ensemble cohérent de normes. Par exemple, c'est le cas d'une autorité (l'architecte-concepteur) qui doit s'assurer que les lois proposées ne sont pas contradictoires. Dans ce cas, on doit revoir comment on réintègre les propriétés étudiées par la logique déontique.

Chapitre 7

Formalismes temporels

Falstaff : *Let time shape*

Shakespeare (*Henri IV*, parte II, acte III, scène 2)

7.1 Introduction

L'un des objectifs de cette thèse est de proposer une méthode unitaire pour la conception des agents normatifs en utilisant le temps comme élément unificateur. Cette approche est directement liée à l'effort déjà annoncé par l'IA classique de simuler, du point de vue calculatoire, le raisonnement humain qui pour être complet doit prendre en compte le changement du monde et le passage du temps. Dans ce chapitre, nous présentons les principaux points de vue trouvés dans la littérature sur le traitement du temps que nous considérons importants pour cette thèse.

7.2 Notions de base et problématique du raisonnement temporel

Plusieurs domaines tels que la physique, les bases de données, les systèmes experts, le traitement du langage, la planification, les recherches opérationnelles, etc. étudient et intègrent le temps dans les produits qu'ils proposent, en employant différentes méthodes de représentation. Comme en général dans l'IA classique et dans l'informatique, on insiste davantage sur la rigueur de représentation des modèles, les approches sont souvent très formelles et utilisent des langages de description à base de logique. Un des avantages est le fait que ces langages proposent à la fois un grand pouvoir d'expressivité et des mécanismes de raisonnement. L'introduction du temps dans ces langages a donné naissance aux langages de logique temporelle. Leur utilisation pour représenter les aspects temporels des connaissances suppose des extensions avec des concepts adjacents issus d'une théorie qui tente d'expliquer ce qu'est le temps ou le changement (voir plus bas les notions d'événements, états, processus, etc.).

Les propriétés et le pouvoir d'expression des langages de logique temporelle dépendent en grande partie des propriétés de la structure temporelle sous-jacente souvent adaptée aux besoins poursuivis. Par exemple, on utilise une structure de temps discret dans la modélisation des systèmes de calcul car on s'intéresse aux modifications d'états du système. Ou bien, on peut avoir une structure dense et linéaire dans le traitement des langues naturelles pour représenter, par exemple, des expressions de type « *Il a plu hier en Normandie* », etc. Dans la suite, nous présentons les différents éléments qui caractérisent la structure d'un langage de logique temporelle.

7.2.1 Points et intervalles

La caractéristique la plus simple est l'unité temporelle qu'un formalisme utilise. Il existe deux approches classiques, basées sur des *instants* (ou points) et sur des *périodes* (ou intervalles). Certaines théories utilisent exclusivement soit les points [van Benthem, 1991], soit les intervalles [Allen et Hayes, 1985]. D'autres utilisent les deux en même temps ou représentent l'une par l'intermédiaire de l'autre [Allen et Hayes, 1989].

7.2.2 Relations temporelles

En fonction de l'unité temporelle choisie, on peut introduire des relations *qualitatives* entre les éléments primitifs. Par exemple, s'il s'agit des instants, on utilise trois relations entre deux points sur une ligne : $<$, $=$ et $>$. McDermott [McDermott, 1982] utilise une théorie du temps définie sur un ensemble infini de points et une relation d'ordre $\leq$ qui est *réflexive*, *antisymétrique* et *transitive*.

Les relations sont plus complexes lorsqu'il s'agit des intervalles. Par exemple, Allen [Allen, 1984] propose une algèbre d'intervalles avec treize relations correspondant à toutes les relations mutuellement exclusives qui peuvent exister entre deux intervalles (voir la figure 7.1). Ces relations peuvent être définies par l'intermédiaire d'une relation primitive unique $meets$ dont les propriétés sont décrites par un ensemble d'axiomes [Allen et Ferguson, 1994]. Par exemple, un intervalle est avant un autre s'il existe un intervalle entre eux :

$$before(i,j) \equiv \exists m.meets(i,m) \wedge meets(m,j)$$

Allen introduit le prédicat in pour décrire exhaustivement la relation où un intervalle est contenu dans un autre :

$$in(i,j) \equiv before(i,j) \vee starts(i,j) \vee finishes(i,j) \tag{7.1}$$

Dans le cas où une information numérique est disponible dans les relations temporelles (par exemple, une heure « 20h34 » ou une date « le 10/10/2001 »), on peut introduire la notion de *durée*. La durée est la distance entre deux points temporels. Le fait d'utiliser une représentation numérique absolue nous permet

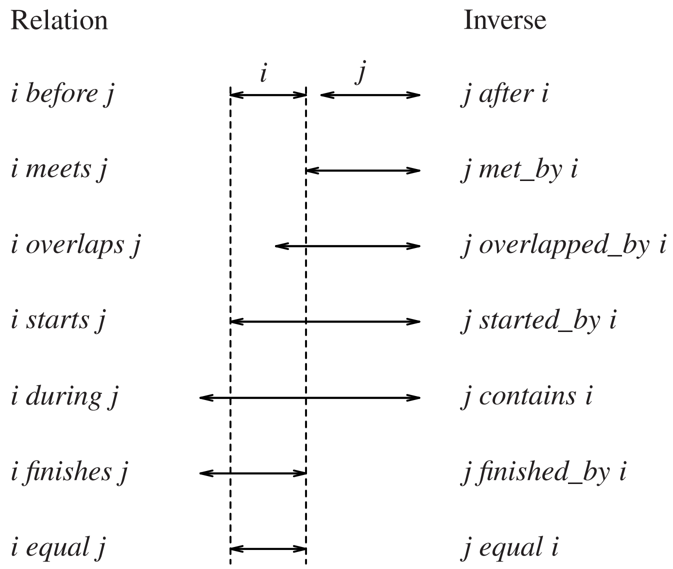


Figure 7.1: Les 13 relations entre les intervalles temporels

de calculer les durées très facilement et de proposer des algorithmes simples pour détecter l'apparition d'un événement dans un intervalle. Dans ce genre de situation, on parle de relations *métriques*.

7.2.3 Granularité

La granularité est une autre caractéristique directement liée à l'opposition entre la représentation du temps par des points et la représentation par des intervalles. Elle décrit la façon dont on change d'échelle dans une représentation. Des contextes différents peuvent demander des granularités différentes. Comme, dans un système, on peut avoir plusieurs contextes, on s'intéresse à des mécanismes qui permettent de basculer entre plusieurs granularités au sein du même système. Par exemple, on considère la foudre normalement comme un événement ponctuel, instantané, mais on peut aussi changer de contexte et l'analyser comme toute une scène qui se déroule avec un commencement et une fin. La granularité est souvent utilisée dans la présence de métriques (par exemples, dates, jours, semaines, mois, etc.), mais elle est aussi étudiée d'un point de vue qualitatif avec des applications, par exemple, dans le traitement du langage naturel [Becher *et al.*, 2000].

7.2.4 Structure du temps

Une fois que l'on a défini à quoi se réfère le temps (unités temporelles et relations), on veut introduire une structure qui permet de définir le comportement (par exemple, par l'intermédiaire des axiomes) de relations temporelles dans une théorie du temps. Cette structure donne forme aux différentes propriétés du temps. On parle ainsi :

– du temps *discret*, comme étant une collection discrète d'éléments temporels, par opposition à un temps *dense* où pour toute paire d'éléments il existe un troisième élément entre eux ;
– du temps *borné* et *non-borné* qui traite le problème si le temps est infini (dans un seul sens ou dans les deux sens) ;
– de la précédence dans le temps, qui peut donner naissance à plusieurs modèles du temps : *linéaire*, *ramifié*, *circulaire*, *parallèle*.

7.2.5 États, événements, processus

Le temps est un élément très important pour décrire ou raisonner sur les changements qui ont lieu dans le monde et sur leurs propriétés. Chaque fois que l'on parle de changement on doit comparer les différentes propriétés ou conditions (qu'on appelle *états*) caractérisant un même objet ou une même partie du monde. Cette comparaison due au changement établit en effet une relation qui est explicitement ou implicitement de type temporel. Les états représentent donc les aspects statiques du monde qu'on décrit dans un langage logique par des assertions qui peuvent être vraies ou fausses à un moment donné, par exemple, « *la porte est ouverte* ». De plus, le changement est supposé comme être le résultat d'une *action*. Souvent l'*événement* associé à cette action est définie comme une *transition* entre deux états. Les événements représentent donc l'aspect dynamique du monde comme dans l'exemple « *fermer la porte* ».

Il y a plusieurs perspectives sur le rapport entre le temps d'une part et les événements et les états de l'autre part. D'après Lin [Lin, 1991], on peut les regrouper en théories absolues et théories relationnelles du temps. Dans l'approche *absolue*, le temps a un statut autonome, défini indépendamment de toute autre notion. Les autres concepts se définissent en fonction de leur rapport au temps : les événements passent et les faits ou les états subsitent (ou tiennent). Dans l'approche *relationnelle* les événements sont les entités importantes que nous percevons et qui donnent forme au monde. Elles sont reliées entre elles par une relation qui définit le temps. Lin propose l'introduction d'une approche intermédiaire, qualifiée d'*absolue modérée*, qui conserve l'importance du statut autonome du temps mais qui explique les relations temporelles par l'intermédiaire des relations qui existent entre les événements.

Dans l'IA, les approches absolues sont préférées aux approches relationnelles. Comme Villa le remarque [Villa, 1994], ce phénomène s'expliqué par le fait que la plupart des personnes habituellement pensent qu'ils ont été placés dans un espace où le temps exprime une dimension d'origine indépendante.

On trouve une troisième catégorie pour décrire les changements du monde, qui, dans la littérature, est traitée différemment des événements ou des états. Il s'agit des processus. Les processus, comme les états, ont une durée et comme les événements, ont une dynamique. Les processus ont été introduits pour rendre compte de l'aspect de déroulement d'un événement, par exemple, pour formaliser des expressions de type « *Alice est en train de manger* »[33]. Le concept fait encore l'objet d'un débat philosophique. En IA, Allen [Allen, 1984] essaye de donner une définition axiomatique des processus qui a été très critiquée : un processus est apparu dans un intervalle de temps s'il est apparu dans quelques sous-intervalles (voir l'axiome 7.4). Comme la plupart des applications du temps sont dans le domaine de la planification, la notion de processus semble être moins importante et par conséquent, elle est souvent assimilée aux événements [Allen et Ferguson, 1994].

7.3 Qualification temporelle

L'utilisation de la logique comme langage de représentation impose une certaine rigueur à propos de sa syntaxe ou de la sémantique des symboles utilisés. L'extension d'un langage logique doit donc se faire avec beaucoup de soin pour préserver les propriétés du langage d'origine et pour offrir de nouvelles caractéristiques (temporelles) qui augmentent son expressivité. Dans la littérature, il existe trois grands types d'approches pour intégrer le temps dans un langage logique, problème connu sous le nom de la *qualification temporelle* : la méthode des arguments temporels dans une logique de premier ordre, la logique modale temporelle et la réification.

7.3.1 Arguments temporels

La méthode la plus simple est de considérer le temps comme argument (parmi d'autres) d'un prédicat de premier ordre. L'argument temporel change l'interprétation du prédicat ou de la fonction dans lesquels il apparaît comme paramètre. Par exemple, on peut représenter le fait qu'il pleut à Caen à 10h par le prédicat : $pleut(Caen,10h)$ ou que la bibliothèque est ouverte de 8h à midi par le prédicat $ouvert(Bibliotheque,8h,12h)$.

Cette méthode a été largement utilisée dans la représentation du temps en mathématique et physique et plus récemment en IA (voir plus bas *le calcul des situations* [McCarthy et Hayes, 1969]). Outre sa simplicité et en dehors du fait qu'elle propose une représentation naturelle du temps, cette méthode présente l'avantage majeur de rester dans un formalisme de premier ordre pour lequel on dispose de résultats et de techniques de preuves bien établis. Ses inconvénients sont liés au statut du temps et au pouvoir d'expressivité. En ce qui concerne le premier inconvénient, le temps n'a pas un statut autonome. Dans l'exemple ci-dessus, il n'existe aucune contrainte pour nous empêcher de considérer une formule du type : $pleut(Caen,Alice)$, où l'objet $Alice$ peut avoir une origine différente d'un objet appartenant au domaine temporel. Pourtant, il semble que cet aspect n'est pas incontournable, car en proposant une logique de type multi-sorte de premier ordre [Walther, 1988], on peut séparer les objets temporels des

33. En anglais, on utilise la forme continue (ou progressive) du verbe : « *Alice is eating* ».

autres objets du domaine. En ce qui concerne le deuxième inconvénient, la méthode des arguments temporels est moins lisible que d'autres. Par exemple, pour décrire qu'Alice a travaillé toute la journée on utilise :

$$\forall t.9h00 \leq t \leq 18h00.travailler(Alice,t)$$

La même expression peut avoir une description plus compacte dans une logique temporelle modale métrique :

$$Holds_{[9h00,18h00]}travailler(Alice)$$

De plus, la méthode des arguments temporels ne permet de décrire que des relations temporelles de type prédicatif, étant de ce point de vue plus limitative que, par exemple, les logiques modales (voir plus bas).

7.3.2 Logique modale temporelle

Les logiques modales temporelles sont construites à base de logiques modales où l'interprétation d'une formule tient compte du contexte temporel. Ces logiques sont obtenues par l'extension du langage de prédicats de premier ordre avec des opérateurs modaux temporels issus de l'interprétation des opérateurs modaux standard de possibilité et nécessité, par rapport aux éléments de temps passés ou futurs :

$F\phi$	-	ϕ est vrai quelque part dans le futur ;
$P\phi$	-	ϕ est vrai quelque part dans le passé ;
$G\phi$	-	ϕ est toujours vrai dans le futur ;
$H\phi$	-	ϕ est toujours vrai dans le passé ;
$Next\phi$	-	ϕ est vrai pour l'élément de temps suivant ;
$Previous\phi$	-	ϕ est vrai pour l'élément de temps précédent.

On utilise la même sémantique basée sur des mondes possibles où chaque élément du temps, instant ou intervalle, représente un monde possible. Les éléments du temps sont liés par une relation de précédence, similaire à la relation d'accessibilité dans les logiques modales. L'interprétation d'une formule, si elle est vraie ou fausse, doit se faire par rapport à un modèle du monde M et un élément du temps t, souvent noté : $M,t \models \phi$. En fonction des propriétés de la relation de précédence, décrites par des ensemble d'axiomes, on obtient différentes logiques.

Il existe des variations des opérateurs temporels qui utilisent le temps comme indice. Par exemple, dans les logiques modales temporelles métriques [Koymans, 1989; Brzoska, 1998], on peut utiliser des intervalles comme indice temporel pour restreindre le champ d'application des opérateurs temporels de nécessité ou de possibilité :

$\Box_{[t_1,t_2]}\phi$	-	ϕ est vrai sur tout l'intervalle $[t_1,t_2]$
$\Diamond_{[t_1,t_2]}\phi$	-	ϕ est vrai quelque part dans l'intervalle $[t_1,t_2]$

Par exemple, pour décrire qu'on va recevoir un message d'Alice dans le quart d'heure qui suit, on utilise : $\Diamond_{[0,15min]} recevoir(message,Alice)$.

On remarque que ces opérateurs sont *relatifs* par rapport au moment de leur interprétation. Cependant, il existe aussi des opérateurs *absolus* interprétés toujours par rapport à un moment d'origine t_0 fixe. Par exemple : $Holds_{[9h00,18h00]} travailler(Alice)$.

Les langages modaux temporels présentent l'avantage de permettre une écriture compacte des formules et offrent plus d'expressivité. Le fait qu'un opérateur s'applique à une formule, le résultat étant une formule dans le même langage logique, donne la possibilité de combiner plusieurs opérateurs modaux d'origine différente (par exemple, opérateurs déontiques pour l'obligation O ou l'interdiction I, opérateurs épistémologiques Bel, etc.).

7.3.3 Réification et *tokens*

La *réification* [Allen, 1984; McDermott, 1982] consiste à transformer une proposition temporelle dans un *terme propositionnel* qui devient ainsi un objet ordinaire du langage ayant le même statut que les autres termes. L'avantage principal de la réification est qu l'on peut décrire la vérité d'une proposition tout en restant dans un formalisme de premier ordre. Ensuite, la qualification temporelle, l'association du temps à cet objet propositionnel, se fait par l'intermédiaire d'un prédicat spécial, par exemple $holds$, qui a comme arguments le terme et le temps (par exemple, $holds(travailler(Alice),[9h00,18h00])$). Ces prédicats ne sont pas seulement un simple lien entre la vérité d'une proposition et le temps, mais ils décrivent aussi la façon dont les deux sont liés. Par exemple, pour le prédicat $holds$, on considère qu'en général il manifeste la propriété d'*homogénéité* : si une proposition est vraie sur un intervalle, alors elle est vraie sur tous ses sous-intervalles. On décrit cette propriété par un axiome de type :

$$holds(p,i) \equiv \forall i'.in(i',i) \rightarrow holds(p,i') \tag{7.2}$$

La plupart des formalismes proposés en IA [Allen, 1984; McDermott, 1982; Kowalski et Sergot, 1986; Shoham, 1988] font usage de la réification en raison de sa capacité de généralisation qui permet de décrire des relations entre des termes propositionnels d'une part et des éléments temporels d'autre part, tout en restant dans un formalisme de premier ordre. Ces formalismes sont présentés avec des logiques multi-sortes et réservent un statut spécial au temps (sans présenter une sémantique distincte qui explique les aspects temporels).

Pourtant, la réification a ses critiques, par exemple Galton qui considère qu'elle n'est pas techniquement nécessaire [Galton, 1991]. Il propose une technique pour éviter l'utilisation de la réification en s'inspirant de l'analyse existentielle de Davidson [Davidson, 1967]. Davidson fait une analyse du calcul des événements, selon laquelle, chaque verbe d'action est associé à une variable existentielle, variable qui exprime des *tokens* d'événements. Par exemple, la proposition « *Jean a vu Alice jeudi à Paris* » a la forme logique suivante :

$$\exists e.saw(Jean,Alice,e) \land in(Paris,e) \land on(Jeudi,e)$$

Pour réfuter l'aspect artificiel des termes ayant un contenu propositionnel, Galton fait appel à des raisons philosophiques. De plus, il montre qu'on peut introduire les types d'événements en quantifiant sur les tokens.

Même Allen reprend ses travaux dans la même direction, et propose avec Ferguson [Allen et Ferguson, 1994] une solution qui considère les variables événements comme la composante centrale pour organiser les connaissances sur le changement.

7.4 Modèles du temps - exemples

Dans cette partie nous présentons quelques exemples de formalismes temporels, qui ont marqué par leur apparition les théories et les techniques pour représenter le temps et les actions en IA.

7.4.1 Action et temps - Allen

Allen propose dans [Allen, 1984] une théorie de l'action et du temps, basée sur des intervalles. Il introduit une algèbre des intervalles et décrit exhaustivement les 13 relations (voir la figure 7.1) qui peuvent exister entre deux intervalles. Selon lui, les points ne sont pas nécessaires parce qu'on peut les représenter par des intervalles très petits. Le modèle du temps est linéaire. La qualification temporelle se fait par l'intermédiaire de la réification et le langage de base est celui des prédicats de premier ordre.

Les concepts de base pour décrire le changement du monde sont les événements, les états (ou *fluents*) et les processus. Pour chaque concept, il introduit des prédicats spéciaux, à savoir $occurs$, $holds$ et $occurring$. Leurs propriétés sont décrites par l'intermédiaire des axiomes. Pour les fluents, on a l'axiome d'homogénéité (7.2) et pour les événements, l'axiome de *solidité* : un événement ne peut se produire que dans un seul intervalle de temps.

$$occurs(e,i) \land in(i',i) \rightarrow \neg occurs(e,i') \tag{7.3}$$

Pour les processus, on a l'axiome selon lequel un processus doit se produire sur au moins un de ses sous-intervalles :

$$occurring(e,i) \rightarrow \exists t'.in(i',i) \land occurring(e,i') \tag{7.4}$$

7.4.2 Calcul des événements

Le Calcul des Événements (CE) a été inventé par Kowalski et Sergot pour permettre la gestion d'une base de données temporelle décrite en Prolog [Kowalski et Sergot, 1986]. Le CE considère comme des notions primitives les événements, les propriétés (états, relations, situations, etc.), les points et les intervalles temporels. Plus précisément, les données temporelles sont utilisées pour décrire un scénario caractérisé par un ensemble d'événements dont l'apparition est définie à base de points temporels et a comme effet le commencement ou la terminaison de la validité d'une propriété (définie à son tour dans un intervalle). Par exemple, les événements pour entrer et sortir d'un parking à 10h et respectivement à 11h, commencent et terminent la propriété qui veut qu'une voiture se trouve dans le parking.

Pour représenter les propriétés, on utilise la réification et pour les événements, les tokens. Le CE permet de dériver diverses propriétés et de calculer les intervalles de temps de leur validité (par exemple, la voiture se trouve dans le parking à 10h30). Pour cela, deux types d'informations doivent être fournis :

– des propositions qui expriment l'apparition (concrète) de certaines instances d'événements ; on utilise pour cela le prédicat `happens_at(E,T)`. Dans l'exemple ci-dessus, en utilisant la syntaxe Prolog, on obtient :

  ```
  happens_at(9716-WM-14 entre-dans-parking, 10h).
  happens_at(9716-WM-14 sort-de-parking, 11h).
  ```

– des règles générales qui indiquent quel type d'événement génère quel type d'état de choses.

En ce qui concerne le deuxième type d'information, deux prédicats sont définis pour décrire les effets générés sur une propriété par l'apparition d'un événement :

```
initiates(Evenement, Propriete)
terminates(Evenement, Propriete)
```

Dans l'exemple ci-dessus, on a la règle générale :

```
initiates(X entre-dans-parking, X est-dans-parking).
```

On remarque que les règles générales s'appliquent à des types d'événements (ex : `X entre-dans-parking`) qui contiennent des variables libres. On fait, ainsi, la distinction entre les types d'événements et les instances d'événements contenant seulement des constantes (ex. `9716-WM-14 entre-dans-parking`).

Pour dériver la validité d'une propriété, on utilise le prédicat `holds_at` qui montre la capacité du CE de traiter la *persistance*. Basé sur la négation par échec [Clark, 1978], ce prédicat décrit d'une façon axiomatique l'« inertie » d'un état par la règle : une propriété est valide à point s'il existe un événement qui l'a produite dans le passé et si elle n'a pas été modifiée[34] entre temps par un autre événement.

34. *clips* est le terme d'origine en anglais pour nommer ce changement.

Formellement cela s'écrit :

```
holds_at(P, T) :-
    happens_at(Ev, Ts), Ts =< T,
    initiates(Ev, P),
    not broken(S, Ts, T).

broken(S, Ts, Te) :-
    happens_at(E, T),
    T1 =< T, T < T2,
    terminates(E, S).
```

En utilisant ce prédicat on peut dériver dans notre exemple que :

```
?- holds_at(9716-WM-14 est-dans-parking, 10h30).
```

est vrai.

Dans sa forme d'origine, le CE présente certains inconvénients pour être utilisé tel quel. Par conséquent, plusieurs extensions ont été proposées telles que (voir [Cervesato *et al.*, 2000] pour un modèle unitaire) : l'introduction des opérateurs modaux, des primitives pour traiter les changements continus, les processus discrets et les actions parallèles, l'introduction des pré-conditions. Conçu comme un programme Prolog, le CE n'a pas de sémantique spéciale dans sa forme d'origine. Shanahan propose, dans [Shanahan, 1997], *Full Event Calculus* qui redonne au CE un statut autonome avec une sémantique basée sur la notion de préférence. Ainsi, selon Brandano [Brandano, 2001], le CE présenté sous cette forme, peut être utilisé pour résoudre un certain nombre de problèmes liés au raisonnement non-monotone sur les actions et le changement avec les caractéristiques suivantes : l'information sur les actions est complètement spécifiée, les actions se succèdent seulement si leurs pré-conditions sont satisfaites, les actions qui réussissent peuvent avoir des effets non-déterministes, les variables sur les états sont de type booléen, l'état initial du monde est complètement et rigoureusement spécifié et enfin, il n'y a pas d'autre information sur les autre états, sauf pour l'état initial.

7.4.3 Calcul des situations

Le Calcul des Situations (CS) a été proposé initialement par McCarthy et Hayes [McCarthy et Hayes, 1969] comme un langage de premier ordre adapté au raisonnement sur des actions. Les ingrédients principaux du CS sont les *situations*, qui correspondent aux prises de vue sur le monde à un instant de temps, et les *actions* ou les événements qui changent le monde d'un état à l'autre. En effet, le CS est basé sur une logique temporelle qui utilise les points et un modèle arborescent pour représenter le temps. Le langage est typé avec des sortes pour les situations et les actions. Certains prédicats, appelés *fluents*, utilisent les situations comme arguments. Une situation indique la séquence d'actions qui ont été exécutées pour arriver de l'état initial à l'état courant. On remarque la présence d'une fonction spéciale $do(a,s)$ qui étant donnée une situation s et une action a calcule la situation qui résulte après l'exécution de a dans la situation s. La situation initiale est notée par S_0. Par exemple, si $met(x,y)$ représente l'action de poser l'objet x sur l'objet y alors $do(met(A,B),s)$ dénote la situation qui résulte de l'action qui fait déplacer A sur B dans la situation s[35].

La formalisation des actions se fait par l'intermédiaire de deux types d'axiomes. Le premier type concerne les *pré-conditions* nécessaires qui doivent être satisfaites chaque fois que l'action peut être exécutée dans la situation courante. On utilise pour cela le prédicat $Poss(a,s)$ qui indique si c'est possible d'exécuter a dans la situation s. Par exemple, s'il est possible qu'un robot r prenne un objet x dans la situation s alors le robot ne tient aucun objet, il est suffisamment proche de x et x n'est pas un objet très lourd :

$$Poss(prendre(r,s),s) \rightarrow [\forall z. \neg tient(r,z,s)] \wedge procheDe(r,x,s) \wedge \neg lourd(x)$$

Le deuxième type concerne la description des lois causales, c.a.d. la manière dont les actions modifient les propriétés des fluents. La spécification de celles-ci se réalise par les axiomes sur les effets ou l'*état successeur*. Par exemple, dans le cas d'un robot qui laisse tomber un objet fragile, cette action va causer son éclat. Dans le CS, on dirait que, si dans la situation courante s, x est fragile, alors dans la situation $do(tomber(r,x)s)$ qui résulte de l'exécution de l'action $tomber(r,x)$, l'objet x va se casser :

$$fragile(x,s) \rightarrow casse(x,do(tomber(r,x)s))$$

L'une des applications du CS est liée au traitement du fameux *problème du cadre*, formulé pour la première fois dans [McCarthy et Hayes, 1969]. Ce problème consiste à spécifier les *invariants* du domaine, c.a.d. les fluents qui ne sont pas affectés par l'exécution de l'action en cause. Dans l'exemple ci-dessus, le fait qu'un robot laisse tomber un objet n'a pas d'influence sur sa couleur. Pour exprimer cela on utilise l'axiome suivant :

$$couleur(x,s) = c \rightarrow couleur(x,do(tomber(r,y),s)) = c$$

Le problème est qu'on peut avoir une infinité de tels axiomes pour représenter tous les invariants. Cependant, certaines solutions ont été proposées pour résoudre ce problème : voir [Shoham, 1988; Reiter, 1991] et [Shanahan, 1997] pour une présentation plus récente de théories et de problèmes connexes.

Grâce à sa capacité de construire une succession d'actions qui explique la transition du monde de l'état initial à l'état actuel, le CS devient un formalisme intéressant pour la planification. Par exemple, pour construire un plan pour atteindre le but G, on doit prouver qu'il existe une situation s où G est vraie.

35. Notons que dans le CS les actions sont représentées par des symboles de fonctions et les situations par des termes de premier ordre.

La démonstration consiste à construire la situation s par une composition d'actions qui donne en effet la séquence désirée, c.a.d. le plan.

En restant dans ce contexte, constructif, le CS a été souvent critiqué parce qu'il n'est pas suffisamment général et expressif pour répondre aux problèmes du monde réel. Par exemple, les actions sont considérées comme étant instantanées et le changement du monde comme étant le résultat d'une seule action exécutée par un agent unique. Pour contrecarrer, ces critiques le CS connaît plusieurs extensions : Gelfond, Lifschitz et Rabinov [Gelfond *et al.*, 1991] montrent comment on peut étendre le langage avec une notion plus générale d'action afin de permettre la représentation des actions simultanées, duratives et conditionnelles ; Lin et Shoham [Lin et Shoham, 1992] éétudient les actions concurrentes et l'indépendance entre actions ; Levesque, Lin et Reiter [Levesque *et al.*, 1995] proposent une extension qui permet au CS de travailler avec des actions complexes, construites à partir des actions élémentaires, d'une manière similaire à la programmation structurée ; Pinto [Pinto, 1994] ajoute au CS des capacités explicites pour le traitement du temps ; le groupe de l'université de Toronto propose GOLOG (et ses variants), une implémentation du CE en Prolog. Pour une description assez complète des problèmes et des travaux liés au CS nous suggérons l'ouvrage récemment écrit par Reiter [Reiter, 2001].

Avant de terminer la présentation du CS, nous faisons une dernière remarque sur l'utilisation du temps et des actions duratives dans le CS. Par défaut, le CS n'utilise que des actions instantanées. Pour introduire dans le CS les actions qui ont une durée d'exécution, Pinto [Pinto, 1994] les représente par l'intermédiaire d'un fluent relationnel appelé *processus* et deux actions instantanées qui commencent et terminent ce processus. Par exemple, pour représenter le déplacement de x à y qui nécessite un intervalle de temps pour l'exécuter, au lieu d'utiliser une représentation compacte de cette action avec $deplacer(x,y)$, on a deux actions instantanées $startDeplacer(x,y)$ et $endDeplacer(x,y)$ et le processus de se déplacer de x à y représenté par le fluent $deplacement(x,y)$. L'action $startDeplacer(x,y)$ a comme effet que le fluent $deplacement(x,y)$ devient vrai, et $startDeplacer(x,y)$ qu'il devient faux. Les axiomes pour définir les pré-conditions et l'état successeur de l'action de se déplacer sont les suivants :

$$
\begin{aligned}
&Poss(startDeplacer(x,y),s) \equiv \\
&\qquad \neg(\exists u,v)deplacement(u,v,s) \wedge position(s) = x \\
&Poss(endDeplacer(x,y),s) \equiv deplacement(x,y,s) \\
&deplacement(x,y,do(a,s)) \equiv \\
&\qquad a = startDeplacer(x,y) \vee deplacement(s,y,s) \wedge a \neq endDeplacer(x,y) \\
&position(do(a,s)) = y \equiv (\exists x)a = endDeplacer(x,y) \vee \\
&\qquad position(s) = y \wedge \neg(\exists x,y')a = endDeplacer(x,y')
\end{aligned}
$$

Avec ce genre de représentation, on peut décrire plusieurs actions qui s'exécutent en parallèle. Par exemple, on peut commencer en même temps à se déplacer et à mâcher de la gomme, suivi par l'arrêt de la mastication et le début de l'action de chanter, et enfin d'arrêter le déplacement. Ce scénario est

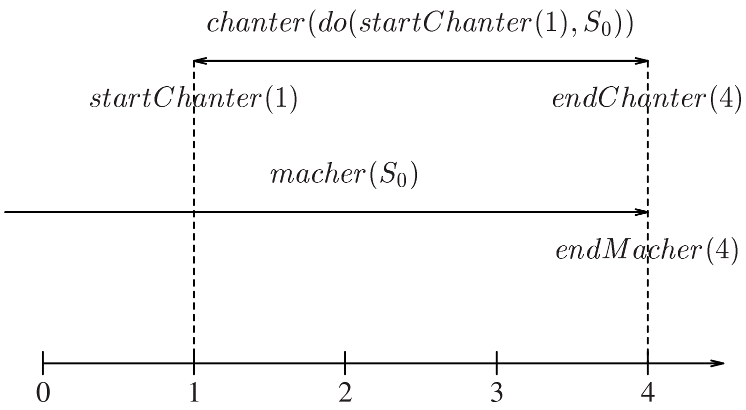


Figure 7.2: Processus temporels dans le Calcul des Situations

représenté par la séquence suivante des actions :

$$\{startDeplacer(A{,}B){,}startMacher\}{,}\{endMacher{,}startChanter\}{,}$$
$$\{endDeplacer(A{,}B)\}$$

L'introduction du temps se fait en ajoutant un argument temporel à toutes les actions instantanées qui représente leur temps concret d'apparition, ex : $startChanter(1)$ exprime le fait qu'on commence à chanter au moment 1 sur l'axe du temps (voir la figure 7.2). La situation correspondante est donnée par $do(startChanter(1){,}S_0)$. Alors, le fluent *chanter(do(startChanter* $(1){,}S_0))$ est vrai entre la situation décrite par l'action $startChanter(1)$ et la situation décrite par l'exécution de l'action $endChanter(4)$. De plus, on peut représenter l'action parallèle de mâcher de la gomme qui, au moment initial S_0, est en train de se dérouler $macher(S_0)$, mais qui se termine en même temps avec le processus de chanter. On a donc que le processus $macher(S_0)$ est vrai jusqu'à l'apparition de l'action $endMacher(4)$.

7.5 Conclusions

Dans ce chapitre, nous avons présenté les principaux problèmes et formalismes utilisés en IA pour représenter le temps et le changement du monde. Le but est de disposer de concepts et de méthodes pour les appliquer à la description de l'interaction normative.

Les problèmes qui nous intéressent sont d'abord liés à la façon dont on décrit la dynamique de l'interaction entre agents. La solution la plus acceptée, et à laquelle on adhère, consiste à utiliser le concept de fluent, pour décrire les propriétés du monde à un certain moment, et celui d'événement pour expliquer la façon dont ce monde change.

Ensuite nous avons besoin d'introduire la notion d'action. Nous le faisons par l'intermédiaire du concept d'événement « personnalisé », c.a.d. un événement pour lequel on connaît l'acteur (ou les acteurs) qui l'a produit. Une autre propriété de la notion d'action est qu'elle doit pouvoir être utilisée pour exprimer des situations réelles. Plus précisément, nous considérons que l'exécution d'une action prend du temps. Ainsi les modèles de temps qui seront utilisés doivent permettre la description et le raisonnement sur des actions duratives.

Un autre concept important lié à la conception des agents normatifs est celui d'observabilité. Comme nous l'avons montré dans le chapitre 5, nous avons besoin d'outils qui offrent aux agents ordinaires et aux éléments de contrôle la possibilité d'observer le comportement des autres agents dans le but de détecter les éventuels comportements déviants par rapport à la norme. Par conséquent, on se propose d'une part de pouvoir intégrer dans la description des normes des informations sur la façon d'exécuter une action dans le temps. Par exemple, on veut décrire l'obligation pour les employés d'une entreprise de faire une pause entre 12h et 14h, ou l'interdiction d'entrer dans les locaux de l'entreprise après 22h. D'autre part, on a besoin de pouvoir comparer cette information normative et temporelle avec le comportement réel des agents. Par exemple, s'ils ont vraiment pris une pause à midi ou s'ils ont essaie d'entrer dans les locaux pendant la nuit. La comparaison entre le comportement idéal et le comportement effectif peut se faire pendant l'exécution du système ou ultérieurement. On doit donc permettre deux types de raisonnement, l'un très réactif qui est basé sur une observation instantanée, et un autre qui utilise des traces d'exécution. Une trace d'exécution représente l'historique de l'exécution du système. Elle contient des informations concernant les événements (et leur moment d'apparition) qui ont eu lieu dans le système ainsi que les propriétés (et les intervalles de leur validité) qui ont changé durant cette exécution.

Les modèles présentés dans ce chapitre ne peuvent pas être utilisés dans leur forme d'origine pour répondre à nos besoins. Par exemple, nous avons montré que le CS doit être étendu pour qu'il puisse prendre en compte le temps, les actions duratives et les actions simultanées. Et pour le CE on doit apporter des modifications afin de représenter le changement continu ou les actions parallèles. Par conséquent nous préférons reprendre certains concepts et les adapter afin de proposer un modèle qui répond mieux aux problèmes liés à l'interaction normative, évoqués ci-dessus. Dans les chapitres suivants nous présentons les efforts de formalisation dans ce sens, ainsi que les outils issus de ce modèle.

Chapitre 8

Un modèle temporel pour l'interaction normative

Le principe réside dans les mots,
mais le principal est dans l'action
Proverbe oriental

8.1 Introduction

L'idée principale utilisée dans la conception d'un système normatif, comme nous l'avons montré auparavant, est d'employer les normes comme moyen (i) pour influencer et (ii) pour contrôler les comportements des agents [36]. Dans le premier cas, pour assurer un comportement conforme aux normes, on les injecte dans les modules de planification et d'ordonnancement des agents. Quant au deuxième cas, en tenant compte des descriptions normatives et des comportements concrets des agents (en cours ou passés), le système doit détecter les comportements déviants. Dans ce chapitre, nous proposons un modèle temporel de premier ordre, comme base formelle pour les SAN, qui permet la description de l'exécution d'une action dans le temps et l'utilisation de normes dynamiques (normes qui ont une durée de vie). Nous montrons comment on construit les normes par l'intermédiaire des concepts déontiques qui expriment l'obligation, la permission ou l'interdiction et qui caractérisent l'exécution d'une action par un agent dans un intervalle de temps. Ensuite, nous proposons une extension du modèle avec des mécanismes de *model checking* utilisés dans le calcul des violations. Les travaux présentés dans ce chapitre ont fait l'objet de plusieurs publications [Stratulat *et al.*, 2001a; Stratulat *et al.*, 2001b; Stratulat *et al.*, 2001c].

36. On rappelle que la notion de contrôle est utilisée ici pour décrire l'observation ou le monitoring d'un comportement. Ensuite, à partir de cette observation on peut prendre des décisions qui affectent directement l'agent en cause ou son comportement, d'où le terme de contrôle.

8.2 Syntaxe et sémantique

Le modèle que nous utilisons est décrit par l'intermédiaire d'un langage logique de premier ordre. Dans cette section nous présentons brièvement les éléments syntaxiques du langage et leur sémantique.

On commence par construire une signature typée Σ [Smolka et Ait-Kaci, 1989; Walther, 1988]. Pour cela on suppose la présence de $\mathbf{V}$ un *vocabulaire* dénombrable qui contient :

1. V un ensemble dénombrable de *variables*,
2. F, un ensemble fini de *symboles de fonctions* munis d'un *profil* (ou typage des arguments et du résultat),
3. P, un ensemble fini de *symboles de prédicats* munis d'un profil.

Avec ce vocabulaire, on construit un langage $\mathcal{L}_p$, qui est un ensemble récursif sur le vocabulaire $\mathbf{V}$ dont les éléments s'appellent des *formules*. Le langage est considéré typé, c.a.d. qu'on diveise l'univers des objets en sous-univers disjoints qui dénotent chacun un type qu'on appelle *sorte*. On utilise peu de sortes principales (prédéfinies) :

– $Agent$: sorte qui représente l'univers des agents ;
– $Event$ et $EventType$ pour représenter l'univers des événements et leurs types respectifs. On ajoute deux sous-sortes $Action \subset EventType$ et $Act \subset Event$ pour faire la distinction entre actes, événements et leurs classes (pour une discussion plus détaillée voir plus bas la section 8.5) ;
– $Time$ pour modéliser le temps, par des instants représentés par des valeurs entières $\mathbb{Z}$ ou rationnelles $\mathbb{Q}$;
– $Interval$ pour représenter les intervalles de temps ;
– $Boolean$ pour représenter le domaine de prédicats et de fonctions logiques.

On note $\mathcal{S}$ l'ensemble des sortes pré-définies ou définies par l'utilisateur. On va supposer donc qu'un symbole de sorte unique est associé à chaque élément du vocabulaire. Cette structuration à base de sortes ou de types contraint le domaine des variables, des constantes et les prédicats, ainsi que le domaine et le type des fonctions. Par exemple, pour modéliser les agents, on utilise la sorte *Agents*. Les agents spécifiques, ex. *Alice*, sont modélisés avec des constantes (fonctions d'arité 0) appartenant à la sorte *Agents*. Une remarque sur la notation : les variables sont notées par des mots en minuscules (ex : e, $agent$, α) et les constantes et les fonctions par des mots commençant par une majuscule (ex : $Alice$).

Plus formellement, on introduit la notion de signature pour les fonctions, qui a la signification suivante :

Définition 8.1 (signature) *Une signature $\Sigma = (\mathcal{S},F)$ est représentée par l'ensemble de symboles de sortes $\mathcal{S}$ et de fonctions F. Tout symbole de fonction $f \in F$ est de profil :*

$$f : s_1 \times \cdots \times s_n \to s_{n+1}$$

où n est l'arité de f et $s_i \in \mathcal{S}$ pour tout $i \in [1..n+1]$.

On considère les définitions usuelles pour les termes et les formules bien formées. Étant donnés une signature $(\mathcal{S}, F)$ et un ensemble de variables V, l'ensemble $\mathcal{T}(F, V)$ est l'ensemble de *termes* construit à partir de F et V. Un symbole de sorte unique est associé à chaque terme. $\mathcal{T}(F, V)$ peut être divisé en sous-ensembles disjoints suivant les éléments de $\mathcal{S}$.

$$\mathcal{T}(F, V) = \biguplus_{s \in \mathcal{S}} \mathcal{T}(F, V)_s$$

Définition 8.2 (terme) *Soit Σ une signature et V un ensemble de variables. L'ensemble $\mathcal{T}(F,V)$ est le plus petit ensemble tel que :*

- *toute variable v à laquelle est associé le symbole de sorte $s \in \mathcal{S}$ est un terme appartenant à $\mathcal{T}(F, V)_s$.*
- *pour tout symbole de fonction f de F de profil $f : s_1 \times \ldots \times s_n \to s$ et pour tout n-uplet de termes $(t_1, \ldots, t_n) \in \mathcal{T}(F, V)_{s_1} \times \cdots \times \mathcal{T}(F, V)_{s_n}$, $f(t_1, \ldots, t_n)$ est un terme de $\mathcal{T}(F, V)_s$.*

L'ensemble $\mathcal{T}(F, V)_s$ est aussi nommé la *sorte* s. Chaque sorte est ensuite supposée admettre au moins un élément. Par exemple, la sorte $Boolean$ est constituée par les constantes logiques $\{True, False\}$. Par $=$, on note la relation d'égalité syntaxique entre deux termes.

Définition 8.3 (atome) *Une* formule atomique *de $\mathcal{L}_p$ est une expression de la forme*

$$p(t_1, \ldots, t_n)$$

où $p \in P$ est un symbole de prédicat d'arité n et de profil $s_1 \times \cdots \times s_n \to Boolean$, tel que pour tout $i \in [1..n]$, t_i est un terme de sorte s_i.

Définition 8.4 (formule) *Une* formule *de $\mathcal{L}_p$ est obtenue à partir de l'application, un nombre fini de fois, des règles suivantes :*

1. *tout atome est une formule ;*
2. *si ϕ est une formule, alors $\neg\phi$ est une formule ;*
3. *si ϕ et ψ sont des formules alors $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \Rightarrow \psi)$, $(\phi \Leftrightarrow \psi)$ sont des formules ;*
4. *si ϕ est une formule et x une variable quelconque de type s, alors $\forall x\ \phi$ et $\exists x\ \phi$ sont des formules.*

En ce qui concerne la sémantique du langage, on construit d'une manière classique une Σ-structure sur la signature Σ, avec des définitions standards pour l'assignation de variables, la consistance et la validité d'une formule. Donc, on veut préciser quand une formule ϕ est vraie dans une Σ-structure $\mathcal{M}$, notée par $\mathcal{M} \models \phi$. Comme dans la définition d'une formule on fait intervenir des prédicats, des termes et des variables, on doit préciser par l'intermédiaire d'une fonction d'*interprétation* $\mathcal{I}$, à quoi se réfèrent les symboles respectifs. Dans la suite, l'univers des objets délimité par une sorte s est noté avec $|s|$ et l'ensemble de ces univers est noté avec $|S|$:

1. Pour les symboles de variables, on suppose qu'il existe une fonction $w : V \to |S|$ qu'on appelle *valuation des variables* qui pour toute variable $v \in V$ de sorte s décrit l'objet auquel elle fait référence dans l'univers délimité par s.
2. Pour les termes, on introduit une fonction $\bar{w}$ qui étend la fonction w aux termes ; on a pour toute variable $v \in V$, $\bar{w}(v) = w(v)$.
3. A tout symbole de prédicat p de type $s_1 \times \cdots \times s_n$, l'interprétation $\mathcal{I}$ attribue la relation $p^{\mathcal{M}} \subseteq |s_1| \times \cdots \times |s_n|$.
4. A tout symbole de fonction f de type $s_1 \times \cdots \times s_n \to s$, l'interprétation $\mathcal{I}$ attribue la fonction $f^{\mathcal{M}} : |s_1| \times \cdots \times |s_n| \to |s|$.

Avec ces précisions on peut définir la vérité d'une formule.

Définition 8.5 (vérité) *Une structure $\mathcal{M}$* satisfait *une formule ϕ, ou ϕ est vrai dans $\mathcal{M}$ ssi $\mathcal{M} \models \phi$. Étant donné $\bar{w}$ une valuation de variables et $\mathcal{I}$ une interprétation, la vérité d'une formule est définie par :*

- *pour deux termes t_1 et t_2 $\mathcal{M} \models t_1 = t_2$ ssi $\bar{w}(t_1) = \bar{w}(t_2)$;*
- *pour tout symbole de prédicat p d'arité n, $\mathcal{M} \models p(t_1,...,t_n)$ ssi $p^{\mathcal{M}}(\bar{w}(t_1),...,\bar{w}(t_n))$ est dans $\mathcal{M}$;*
- *$\mathcal{M} \models \neg\phi$ ssi on n'a pas que $\mathcal{M} \models \phi$;*
- *$\mathcal{M} \models \phi \vee \psi$ ssi $\mathcal{M} \models \phi$ ou $\mathcal{M} \models \psi$;*
- *$\mathcal{M} \models \forall x\ \phi$ ssi en attribuant à x toute valeur $o \in |S|$ on a $\mathcal{M} \models \phi$;*
- *pour les cas qui restent $(\phi \wedge \psi)$, $(\phi \Rightarrow \psi)$, $(\phi \Leftrightarrow \psi)$ et $\exists x\ \phi$ on utilise des définitions similaires.*

Définition 8.6 (modèle, validité, consistance, conséquence logique) *En ce qui concerne la validité d'une formule, on a les définitions suivantes :*

1. ***Modèle :*** *Une structure $\mathcal{M}$ est un* modèle *pour une formule ϕ ssi $\mathcal{M}$ satisfait ϕ. $\mathcal{M}$ est un modèle pour un ensemble de formules ssi il est un modèle pour chaque formule.*
2. ***Validité :*** *Une formule ϕ est* valide*, noté $\models \phi$, ssi toute structure $\mathcal{M}$ est un modèle pour ϕ.*
3. ***Consistance :*** *Une formule est* consistante *ssi elle a un modèle.*
4. ***Conséquence logique :*** *Considérons Γ un ensemble de formules. ϕ est une* conséquence logique *de Γ ssi tout modèle de Γ est aussi modèle pour ϕ.*

Dans la suite nous allons introduire les fonctions et les prédicats principaux du modèle.

8.3 Temps - instants et intervalles

On commence par la présentation des éléments liés au temps. Le modèle qu'on construit utilise une représentation linéaire du temps. Nous avons donc une seule perspective sur le passé et sur le futur. En ce qui concerne le choix des unités temporelles primitives on remarque le besoin (voir plus loin la notion d'observation et les normes dynamiques) d'utiliser en même temps les points et les intervalles. Le langage $\mathcal{L}_p$ est ainsi étendu avec les prédicats et les fonctions suivantes :

– prédicats pour définir les relations entre instants : $=$, $<$, et $\leq$.
– fonctions pour additionner deux durées : $+$.
– prédicats pour définir les relations entre intervalles, du type $Interval \times Interval$. Leur utilisation et définition sont celles proposées par Allen dans [Allen, 1984; Allen et Ferguson, 1994] avec les notations suivantes (voir également la figure 7.1) :

$$
\begin{array}{ll}
starts(i,j) & \\
finishes(i,j) & \\
meets(i,j) & i : j \\
before(i,j) & i \prec j \\
during(i,j) & i \sqsubset j \\
before(i,j) \vee meets(i,j) & i \prec: j \\
during(i,j) \vee i = j & i \sqsubseteq j \\
disjoint(i,j) & i \bowtie j \equiv i \prec: j \vee j \prec: i
\end{array}
$$

– d'autres prédicats et fonctions définis en même temps sur les instants et les intervalles :
  1. $in(t,i)$ - prédicat qui est vrai si l'instant t est dans l'intervalle i ;
  2. $[t_1,t_2]$ - fonction qui génère un intervalle à partir de deux instants ;
  3. $\cap$ - fonction qui calcule l'intersection de deux intervalles - on note avec $\varnothing$ l'intervalle vide ;
  4. $min(i)$ et $max(i)$ - fonctions qui retournent l'instant minimal, respectivement maximal, de l'intervalle i.
  5. les intervalles non-bornés :
     (a) $]\leftarrow,t] = \{x|x \leq t\}$
     (b) $[t,\rightarrow[= \{x|x \geq t\}$ et
     (c) $]\leftarrow,\rightarrow[=]\leftarrow,t] \cup [t,\rightarrow[$.

8.4 Prédicats temporels et *fluents*

Le modèle formel sous-jacent à un système d'agents normatifs demande une représentation explicite du monde et de la façon dont ce monde change. Suivant la tendance générale dans l'IA classique

$holds(4h00,pleut)$
$holds([9h,12h],ouvert(Bibliotheque))$

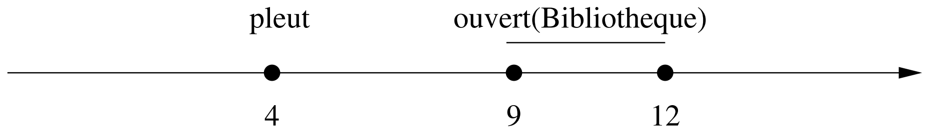


FIG. 8.1 – *Fluents*

[McCarthy et Hayes, 1969; Allen, 1984; Shoham, 1988], nous proposons d'utiliser dans notre approche comme principaux ingrédients, d'une part les $fluents$, qui décrivent l'état du monde à un instant donnée ou sur tout un intervalle et d'autre part, les événements qui expriment les changements qui ont lieu dans ce monde.

Traditionnellement, les fluents sont représentés par l'intermédiaire de la réification, une méthode qui transforme un prédicat dans un terme du langage. L'avantage principal de cette méthode est qu'elle nous permet de décrire les propriétés d'un prédicat tout en restant dans un formalisme de premier ordre. Sa critique majeure est qu'elle n'est pas naturelle [Galton, 1991]. Dans la suite nous utilisons des prédicats de premier ordre pour représenter les fluents et nous suggérons une notation qui permet de les différencier des autres prédicats.

Notons donc que pour représenter les deux éléments principaux du modèle, les fluents et les événements, on utilise des prédicats qui décrivent leurs relations avec le temps et qu'on appelle prédicats temporels. De ce point de vue, on peut considérer qu'on est dans une approche similaire à la méthode des arguments temporels présentée au paragraphe 7.3.1, à laquelle on ajoute la notion de fluent.

8.4.1 Notation *holds*

Nous faisons ici deux remarques qui nous aident à présenter cette notation. D'abord, on note que les propriétés des prédicats temporels peuvent être différentes suivant leur comportement sur un intervalle. Par exemple, dans le chapitre 7, nous avons montré que les fluents décrivent des relations *homogènes* : un prédicat p est homogène ssi quand il est vrai sur un intervalle i, il est aussi vrai sur tout sous-intervalle contenu dans i. Quant aux événements, il n'y a aucune information concernant leurs propriétés sur les sous-intervalles de leur intervalle d'apparition. On appelle ce comportement, par opposition avec la propriété d'homogénéité, *anti-homogène*.

Exemple 8.7 A titre d'exemple, on veut décrire le fait que la bibliothèque est ouverte le matin. On peut définir pour cela le prédicat $ouvert(Bibliotheque,[9h00,12h00])$ qui a comme argument temporel un intervalle. Pour montrer que le prédicat est un fluent et donc homogène, on doit définir sa propriété d'homogénéité à l'aide d'un axiome correspondant :

$$ouvert(Bibliotheque,i) \equiv \forall j.j \sqsubseteq i \Rightarrow ouvert(Bibliotheque,j)$$

Avec cet axiome, on peut ainsi inférer que la bibliothèque est ouverte par exemple entre 10h et 10h30, et donc que le prédicat $ouvert(Bibliotheque,[10h00,10h30])$ lui aussi est vrai.

Exemple 8.8 Comme exemple de prédicat anti-homogène, prenons le prédicat $horaire(Bibliotheque,[9h00,18h00])$ qui exprime que l'horaire affiché sur la porte de la bibliothèque est de 9h à 18h. Ce prédicat décrit une relation unique entre l'objet horaire affiché sur la porte de la bibliothèque et un intervalle de temps. Le prédicat ne doit pas nous permettre de faire des inférences incorrectes, par exemple, que sur l'affiche de la porte de la bibliothèque il est écrit que l'horaire est de 10h à 10h30.

Avec ces deux exemples, nous montrons le besoin de différencier entre les prédicats temporels qui ont la propriété d'homogénéité et ceux qui ne l'ont pas.

La deuxième remarque est liée au fait que la signification qu'on donne intuitivement à l'argument temporel n'est pas la même dans les deux exemples ci-dessus. Le problème vient du fait que dans l'approche d'arguments temporels le temps n'a pas un statut autonome (voir le chapitre 7). Dans l'exemple 8.7, on a un fluent dont l'intervalle est utilisé pour capter les propriétés du monde sur une période. Dans l'exemple 8.8, on s'intéresse plus à la notion d'intervalle généralisé, et donc on peut interpréter l'argument comme appartenant à une sorte différente de $Interval$ (comme temps). Donc, il ne s'agit pas du même contexte. On peut rencontrer par exemple l'utilisation de l'intervalle $[9h00,18h00]$ pour spécifier les conditions de travail dans un contrat, entre 9h et 18h travailleés par semaine (durée). Par conséquence, au moment de la définition d'un prédicat on doit faire attention aux types d'arguments. Le fait qu'on utilise une logique multi-sorte nous permet cette gestion correcte.

Exemple 8.9 Pour illustrer, on peut avoir dans le même prédicat deux arguments temporels représentés par des intervalles, l'un qui aide à exprimer une propriété du monde et l'autre qui décrit un intervalle général : l'année dernière il y avait affiché sur la porte de la bibliothèque l'horaire de 9h à 18h.

$$horaire(Bibliotheque,[9h00,18h00],[01/01/2000,31/12/2000])$$

Pourtant l'introduction d'une nouvelle sorte, chaque fois que l'on veut décrire un certain type d'intervalle temporel ayant les mêmes propriétés que les sortes $Time$ et $Interval$, nous semble artificielle et elle introduit beaucoup de redondances. De plus, les liens qui existent entre des sortes différentes sont difficiles à gérer.

A partir de ces deux remarques nous proposons d'introduire une *notation* spéciale pour les prédicats fluents homogènes. Si $p(t_1,...,t_n,t)$ est un prédicat fluent de type $s_1 \times \cdots \times s_n \times s$ avec s étant soit la sorte $Interval$ ou $Time$, et s'il est muni de la propriété d'homogénéité par rapport à l'argument t alors on peut utiliser la notation équivalente :

$$holds(t,p(t_1,...,t_n)) =_{notation} p(t_1,...,t_n,t) \tag{8.1}$$

Même si l'homogénéité est définie sur un intervalle, on peut utiliser la notation de $holds$ pour un instant t et un prédicat p. Elle est donnée par :

$$holds(t,p) \equiv holds([t,t],p)$$

Cette notation facilite la lecture parce qu'elle précise qu'il s'agit d'un prédicat fluent homogène. On peut réécrire dans les deux premiers exemples 8.7 et 8.8, que la bibliothèque a été ouverte seulement le matin, même si son horaire d'ouverture est pour toute la journée :

$$holds([9h00,12h00],ouvert(Bibliotheque)) \wedge horaire(Bibliotheque,[9h00,18h00])$$

Pour l'exemple 8.9 nous obtenons avec cette notation que :

$$holds([01/01/2000,31/12/2000],horaire(Bibliotheque,[9h00,18h00]))$$

8.5 Évènements, types d'événements, actes et actions

On introduit la notion d'*événement* pour décrire la manière dont le monde change. Nous n'utilisons que des événements non-déterministes, c.a.d. des événements pour lesquels on ne connaît pas (ou il est très difficile de savoir) quels sont les effets produits. Par conséquent, nous utilisons une représentation dans laquelle on précise le temps de son apparition. De plus, on considère que chaque événement a une durée : il commence à se manifester et il se termine à certains instants (différents ou identiques). L'apparition d'un événement est décrite par deux prédicats non-fluents $starts(e,t)$ et $finishes(e,t)$. Ils expriment que l'événement e a commencé/fini à l'instant t. Cette écriture est équivalente à $occurs(e,i)$ où i est un intervalle. Par exemple, le vol d'un avion Air France numéro 330 qui a décollé de Paris à 9h00 et qui a atterri à Nice à 10h00 est considéré comme un événement dont l'apparition est représentée par (voir la figure 8.2) :

$$occurs(Vol330,[9h00,10h00])$$

Par rapport au moment de l'observation t_{obs} que nous allons décrire plus loin, les prédicats $starts$, $finishes$ ou $occurs$ seront utilisés pour définir d'autres prédicats fluents, tels que ceux qui permettent de décrire l'exécution d'une action ou les cas de violation.

$occurs(Vol330,[9h00,10h00])$
$occurs(Presentation,[11h,12h])$

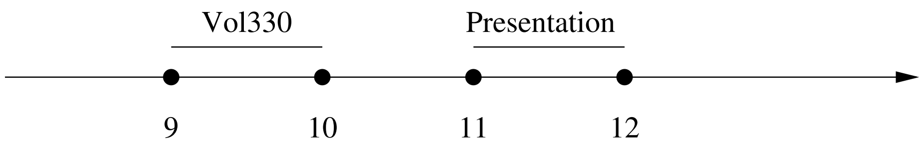


FIG. 8.2 – *Événements*

Les événements caractérisent, en général, des changements qui n'ont pas un acteur qui les a produits. Quand un événement a un acteur on l'appelle $acte$. Si, par exemple, Alice fait une présentation dans un séminaire, on a l'événement $Presentation$ décrit par le prédicat $occurs(Presentation,[10h00,11h00])$ et l'information qu'Alice est son acteur $agent\_of(Alice,Presentation)$. Ou plus court, dans le même langage logique (voir aussi la figure 8.2) :

$$does(Alice,Presentation,[10h00,11h00])$$

Nous introduisons la notion de classe d'événements, ou *type d'événements*, comme étant l'abstraction d'une certaine collection d'événements [Horty et Belnap, 1995]. Elle est très utile dans l'interprétation de l'apparition d'un événement qui représente plusieurs choses à la fois, et qui est donc la même instance de plusieurs classes. Par exemple, lorsque quelqu'un signe un chèque, l'événement peut être interprété soit comme une écriture sur du papier, soit comme une identification, soit comme un paiement. De la même façon, on introduit les $actions$ qui sont des classes d'actes ou d'événements produits par un acteur. Pour conclure, on considère qu'un événement est l'instance d'un type d'événement et qu'un acte est l'instance d'une action.

Dans la suite, nous présentons les principaux prédicats qui nous permettent de raisonner sur l'apparition d'un événement, voire sur l'exécution d'un acte.

8.5.1 Événements

Avant d'introduire les prédicats afférents aux événements, notons que les événements et les actes peuvent être primitifs ou composés. Pour permettre le traitement des événements composés, on utilise une algèbre d'événements avec des opérations de composition séquentielle « ; », parallèle « $||$ » et choix non-déterministe « ? ». On surcharge ces opérations pour qu'on puisse les utiliser avec des termes de type $Event$, $EventType$, $Acte$ et $Action$.

Les prédicats qui opèrent sur les événements sont :

– $starts(e,t)$ et $finishes(e,t)$, prédicats de type $Event \times Time$. Ce sont les prédicats de base avec lesquels on décrit l'apparition d'un événement. De plus on a que chaque événement commence et se termine à un instant unique, qui s'exprime dans le langage logique par les axiomes :

$$\forall e,t,t' \neq t \; starts(e,t) \Rightarrow \neg starts(e,t')$$
$$\forall e,t,t' \neq t \; finishes(e,t) \Rightarrow \neg finishes(e,t')$$

– $occurs(e,i)$, prédicat de type $Event \times Interval$ qui exprime si un événement e est apparu sur l'intervalle i. Ce prédicat va être utilisé dans les définitions de tous les autres prédicats du modèle. Sa définition dépend du type de l'événement : primitif ou composé.

  Pour les événements primitifs on a :

$$occurs(e,[t_1,t_2]) \; \equiv t_1 \leq t_2 \wedge starts(e,t_1) \wedge finishes(e,t_2)$$

  Pour les événements composés :

  1. $occurs(e_1 ? \, e_2,i) \; \equiv occurs(e_1,i) \vee occurs(e_2,i)$
  2. $occurs(e_1 || \, e_2,i) \; \equiv \exists i' \; i' \sqsubseteq i \wedge$
     $(occurs(e_1,i) \wedge occurs(e_2,i') \vee$
     $occurs(e_1,i') \wedge occurs(e_2,i))$
  3. $occurs(e_1 ; e_2,i) \; \equiv \exists i_1,i_2 \; i_1 \prec: i_2 \wedge$
     $starts(i_1,i) \; \wedge finishes(i_2,i) \wedge$
     $occurs(e_1,i_1) \wedge occurs(e_2,i_2)$

Une fois $occurs(e,i)$ défini, on se demande si en général l'événement est apparu, sans préciser l'intervalle. Pour cela, on définit $occurs(e)$. On rappelle qu'on est dans une logique typée qui permet l'utilisation du même nom pour deux prédicats ayant des arguments différents (type et nombre d'arguments) :

$$occurs(e) \equiv \exists i \; occurs(e,i)$$

Dans ce modèle, on dispose d'un temps linéaire qui permet l'utilisation du passé relatif à un instant. Pour décrire les apparitions passées d'événements, nous allons construire de nouveaux prédicats qui sont des versions fluents d'$occurs$.

– on commence par $occurred(e)$ qui est vrai à l'instant t si e est apparu avant t :

$$holds(t,occurred(e)) \equiv \exists i \; occurs(e,i) \wedge i \prec: [t,\infty)$$

– $occurred(e,i)$ est la version qui indique si e est apparu dans l'intervalle i, où i est soit son intervalle d'apparition, soit un intervalle plus large qui le contient. Les deux variantes sont :

  1. $holds(t,occurred^*(e,i)) \equiv occurs(e,i) \wedge i \prec: [t,\infty)$
  2. $holds(t,occurred(e,i)) \equiv \exists i' \; i' \sqsubseteq i \wedge occurs(e,i') \wedge i' \prec: [t,\infty)$

$$holds(t_{obs}, occurred(Vol330, [9h00, 10h00]))$$
$$holds(t_{obs}, occurred(Vol330,]\leftarrow, t_{obs}]))$$
$$holds(t_{obs}, occurred(Vol330))$$
$$holds([10h, t_{obs}], occurred(Vol330, [9h00, 10h00]))$$

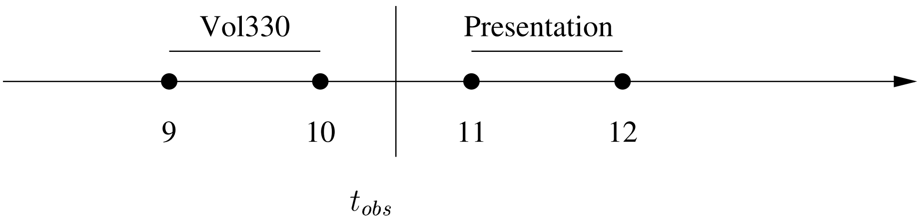


FIG. 8.3 – *Le fluent* $occurred$ *et ses variantes*

$$holds(t_{obs}, occurring(Presentation))$$

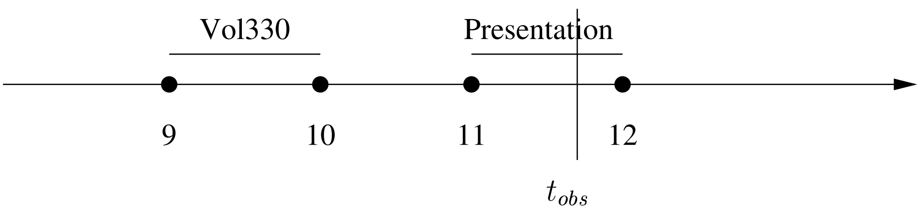


FIG. 8.4 – *Le fluent* $occurring$

Dans les définitions qui vont suivre, nous allons utiliser la deuxième variante.

– $occurring(e)$ est vrai à l'instant t si l'événement est en train de se produire :

$$holds(t,occurring(e)) \equiv \exists i \; in(t,i) \wedge occurs(e,i)$$

Comme exemple d'utilisation (voir la figure 8.3), nous avons introduit un temps d'observation t_{obs} qui peut se déplacer sur l'axe de temps. En fonction de sa position, on peut dire quels sont les événements qui sont apparus dans le passé, par l'intermédiaire du prédicat $occured$ et de ses variantes. Dans la figure 8.4, on a la même chose mais pour $occurring$ qui montre quels sont les événements en train de se produire.

– $subclass(\varepsilon,\varepsilon')$ un prédicat du type $EventType \times EventType$ qui aide la description des hiérarchies de classes d'événements. Par exemple, on peut considérer que signer est une façon d'écrire : $subclass(Signer,Ecrire)$.

– $instance\_of(e,\varepsilon)$ un prédicat du type $Event \times EventType$. Il fait le lien entre les événements et leurs classes. Pour exprimer que la signature du chèque no. 4526 est un paiement, on utilise $instance\_of(signe\_cheque(4526), Payer)$. Le prédicat $instance\_of(e,\varepsilon)$ doit s'appliquer aux deux types d'événements et classes d'événements : primitifs et composites. Pour cela, on introduit des axiomes pour *instance_of*, où on note avec « $\circ$ » l'un des symboles de fonctions pour la composition : « ? », « ; » et « || ».

$$instance\_of(e_1 \circ e_2,\varepsilon_1 \circ \varepsilon_2) \equiv instance\_of(e_1,\varepsilon_1) \wedge instance\_of(e_2,\varepsilon_2)$$

8.5.2 Actes

Un acte étant considéré comme un événement produit par un acteur, les prédicats sur les actes sont similaires aux prédicats sur les événements. Avant de les présenter, on remarque que dans les cas où il existe plusieurs acteurs pour une action composite, il est difficile de savoir dans le formalisme présent, qui est le responsable de chaque action individuelle. Par conséquent, on utilise l'hypothèse que l'action composite est réalisée soit par plusieurs agents qui participent tous à la réalisation de toutes les actions composantes, soit par un seul agent, mais pas par les deux. Pour décrire cette propriété, on a besoin d'introduire la notion de groupe d'agents, pour laquelle on introduit une nouvelle sorte $Agents$. Les agents et les groupes d'agents sont liés alors par l'intermédiaire des opérations simples sur les ensembles. Une solution élégante proposée dans [Panzarasa *et al.*, 1999] est d'introduire le prédicat $singleton(gr,agent)$ qui exprime que le groupe gr est formé par le seul agent $agent$:

$$singleton(gr,agent_1) \equiv \forall agent_2 \in gr \Rightarrow agent_1 = agent_2$$

Ainsi, on fait la différence entre l'exécution d'un acte par un seul agent, pour laquelle on introduit le prédicat $agent\_of(agent,act)$ et l'exécution d'un acte par un groupe d'agents, pour laquelle on utilise le prédicat $agents\_of(gr,act)$. Pour les actes composites nous avons que :

$$agent\_of(agent,act_1 \circ act_2) \equiv agent\_of(agent,act_1) \wedge agent\_of(agent,act_2)$$
$$agents\_of(gr,act_1 \circ act_2) \equiv agents\_of(gr,act_1) \wedge agents\_of(gr,act_2)$$

Maintenant, on peut introduire les prédicats qui expriment l'exécution des actes par un groupe d'agents ou par un agent seul :

– $do(gr,act,i)$, un prédicat du type $Agents \times Act \times Interval$ qui montre que le groupe gr a exécuté act dans l'intervalle i. C'est le prédicat correspondant à $occurs(e,i)$ pour des événements. Sa définition est :
$$do(agent,act,i) \equiv occurs(act,i) \wedge agent\_of(agent,act)$$
Dans le cas d'un seul agent, on a le prédicat $does(agent,act)$, défini par :
$$does(agent,act) \equiv \forall gr\ do(gr,act) \Rightarrow singleton(gr,agent)$$
Pour des raisons de simplicité, nous allons utiliser seulement les prédicats qui prennent en compte un seul agent pour exprimer l'acteur de l'action composite, le cas des plusieurs agents étant similaire à celui présenté ci-dessus.
– on peut définir d'une façon similaire d'autres prédicats sur les actes, respectivement, *done(agent, act)*, *done(agent, act, i)* et *doing(agent, act)*, plus leurs variantes correspondant aux groupes d'agents.

Nous introduisons un autre prédicat fluent $failed(agent,act,i)$, de type $Agent \times Act \times Interval$ pour représenter le fait que l'agent n'a pas exécuté l'acte dans l'intervalle considéré :

$$holds(t,failed(agent,act,i)) \equiv holds(t,\neg done(agent,act,i))$$

L'introduction de ce prédicat est motivée par le fait qu'on ne permet pas d'appliquer la négation à un symbole d'action ou d'utiliser le concept d'action négative qui en général pose des problèmes de conceptualisation et d'interprétation (cf. section 6.5). La solution que nous adoptons ici est de prendre en compte l'hypothèse d'un monde clos et de considérer la négation par échec de la même façon qu'elle est utilisée dans la programmation logique [Clark, 1978]. Ainsi l'échec d'exécuter un acte dans un intervalle est défini par l'échec de dériver si l'acte a été exécuté dans l'intervalle donné.

8.6 Propriétés déontiques et normes dynamiques

Pour décrire une norme, on utilise les concepts déontiques d'obligation, d'interdiction et de permission. Ces notions expriment des propriétés qui caractérisent l'état d'un agent, voire du monde. En ce qui concerne la dualité normes-violations, on introduit d'une part, de façon explicite, des prédicats fluent dits déontiques qui caractérisent l'aspect déontique et temporel des actions, et d'autre part des prédicats de violation qui se calculent en utilisant les descriptions normatives et le comportement réel du système.

8.6.1 Représentation

Les prédicats déontiques qui décrivent le statut déontique d'une action sont comme suit : $O(agent,\alpha,i)$, $P(agent,\alpha,i)$, $I(agent,\alpha,i)$. Ils mettent en relation un $agent$, une *action* et un *intervalle* de temps, exprimant qu'il est *obligatoire* / *permis* / *interdit* pour l'agent d'exécuter certains actes de type α dans l'intervalle i.

Étant donné le statut déontique d'une action, on peut construire les normes. Leur forme générale est de type conditionnel :

$$OPI \Leftarrow \phi \tag{8.2}$$

avec OPI l'un des prédicats déontiques, et ϕ, toute formule logique appartenant au langage présenté auparavant. Par exemple, on peut avoir une norme dans le code de la route qui oblige l'auteur ou la personne impliquée dans un accident de faire une déclaration dans les 24 heures qui suivent :

$$O(agent,declarer(accident),[t,t+24h]) \Leftarrow does(agent,accident,t)$$

Notre objectif est aussi de pouvoir exprimer que ces normes décrivent des propriétés qui changent avec le temps, comme toute loi qui dans la réalité a une durée de vie. On propose donc de qualifier temporellement les normes en précisant l'intervalle de temps dans lequel elles sont valides. Ce type de norme s'appelle *dynamique*. Pour l'instant, dans le langage $\mathcal{L}_p$, on ne peut qualifier que les propriétés déontiques, par exemple :

$$holds(i,O(agent,\alpha,j))$$

où i exprime la *durée de vie* et j est appelé l'*intervalle de référence*.

On se concentre donc sur l'étude des normes dont la condition est vide. Dans la section 8.8 nous allons montrer comment on peut qualifier temporellement les normes plus génériques du type (8.2).

8.6.2 Motivations sur la représentation

Dans la suite, nous faisons quelques remarques sur les raisons qui nous ont conduits à choisir cette représentation pour exprimer les normes et les propriétés déontiques :

1. Comme notre intérêt est plutôt lié à l'application des normes dans le monde des SMA, où un agent est par définition *celui qui agit*, on s'intéresse plus à l'aspect normatif du comportement et donc de l'action. C'est une raison suffisante pour choisir l'action comme domaine d'application

de la caractérisation déontique. Notons que cette approche est différente de la technique générale employée dans le domaine de la logique déontique standard, où les propriétés déontiques sont décrites avec des opérateurs modaux du type $O\alpha$ avec α une formule logique (voir le chapitre 6).

2. L'utilisation d'un intervalle de temps dans les prédicats déontiques est motivée par la manière dont les agents et les éléments de contrôle interprètent les normes. Par exemple, supposons qu'une norme oblige un agent à réaliser α sans préciser quel est l'intervalle d'exécution : soit l'agent va essayer d'exécuter α immédiatement, mais il aura besoin de temps[37], soit il va ajourner son exécution jusqu'au moment où il sera disponible (conformément au processus de planification et d'ordonnancement), soit il ne va jamais l'exécuter. L'élément de contrôle, lui, vérifie si l'agent a violé la norme et quand. Comme il n'y a pas de période de temps spécifiée, l'action doit être exécutée instantanément, au moment même où la norme a été énoncée ; on est donc dans un état de violation dès le début. Si on peut se permettre d'attendre *quelque temps*, alors cette période doit être spécifiée explicitement et introduite dans la norme comme argument temporel.
3. Les prédicats déontiques expriment des propriétés du monde et par conséquent, ce sont des fluents. C'est-à-dire que leur vérité change avec le temps. L'avantage de cette représentation est double. D'une part, en utilisant des fluents homogènes on peut représenter la notion de persistance d'une norme. D'autre part, on peut distinguer deux aspects temporels d'une norme : l'intervalle de temps auquel une norme fait référence et sa durée de vie. La notion de durée vie est très intuitive et naturelle, parce qu'on peut avoir dans le monde réel des lois qui ont été votées à un certain moment et puis abrogées. La durée de vie d'une loi peut être différente de l'intervalle de temps auquel la loi fait référence. Par exemple, l'obligation d'Alice de payer les taxes en janvier, s'écrit :

$$O(Alice{,}Payer\_taxes{,}[01/01{,}31/01])$$

Pourtant, ce n'est pas suffisant. Il faut préciser quand l'obligation est valable. Supposons qu'une loi plus générale sur les impôts, votée l'année dernière et toujours valable, nous impose cette obligation. Intuitivement, l'obligation spécifique pour Alice a la même durée de vie que celle de la loi mère. Avec cette hypothèse, ce qu'on peut dériver de la loi générale, pour Alice devient :

$$holds([01/01/2000,\ \rightarrow[{,}O(Alice{,}Payer\_taxes{,}[01/01{,}31/01]))$$

Si par hasard, la loi fiscale est abrogée ou modifiée en 2005, on a :

$$holds([01/01/2000{,}01/01/2005]{,}O(Alice{,}Payer\_taxes{,}[01/01{,}31/01]))$$

4. Un autre exemple qui met en évidence les deux aspects temporels utilisés dans la représentation des normes concerne les lois rétroactives. Dans ce cas, la période décrite dans la loi peut faire

37. Rappelons que dans notre approche on considère les actions comme ayant une durée d'exécution et on les représente comme des objets (termes) dans un langage de premier ordre.

référence à des sous-intervalles de temps antérieurs au moment de sa publication. En conséquence, il est possible d'avoir des actes qui sont complètement légaux au moment de leur apparition, mais qui sous les auspices de la nouvelle loi génèrent à présent des cas de violation.

5. Les propriétés des prédicats déontiques par rapport à l'argument temporel qui exprime l'intervalle de référence, sont différentes. La permission et l'interdiction sont homogènes et l'obligation est anti-homogène. Par exemple, le fait qu'il est interdit d'utiliser l'imprimante entre 12h et 14h implique aussi que l'interdiction s'applique à l'intervalle de 13h à 13h30. Idem pour la permission. En revanche, pour l'obligation, le fait d'être obligé de faire une réunion aujourd'hui n'implique pas que la réunion doit avoir lieu à midi.

6. Les prédicats déontiques s'appliquent à des actions et non à des actes. Une action est un concept abstrait qui est représenté dans notre modèle par la notion de classe. Les actes sont des réalisations concrètes, ou des instances d'actions. Il nous semble donc étrange de dire dans une norme que l'acte de « signer le chèque no. 4526 » est obligatoire. Ce qui est obligatoire, c'est l'action ou la classe à laquelle il appartient, c'est-à-dire « payer ».

7. Dans l'approche que nous proposons on s'intéresse moins au raisonnement sur les normes, comme c'est le cas de la logique déontique, qui propose d'une part des mécanismes pour détecter les conflits ou les incohérences entre plusieurs normes, et d'autre part des mécanismes d'inférence des nouvelles normes à partir d'un ensemble initial de normes. En ce qui concerne le premier aspect, nous avons besoin de tels mécanismes seulement au moment de la construction des normes et donc on doit munir les créateurs des normes des outils appropriés. Quant aux agents ordinaires et aux éléments de contrôle, la détection explicite des incohérences est moins importante. Ce qui compte est la résolution des conflits entre plusieurs obligations qui ne sont pas nécessairement originaires de la même source. Par exemple, un agent peut être obligé par deux autres agents en même temps, un qui lui demande de fermer une porte et un autre qui lui demande de l'ouvrir. L'agent ne peut satisfaire qu'une des deux obligations et forcément de violer l'autre. Le choix entre les deux obligations est la base du mécanisme interne de l'agent pour gérer les conflits.

   En ce qui concerne le deuxième aspect, lié à l'inférence de nouvelles normes, on simplifie le problème en considérant l'hypothèse selon laquelle les propriétés déontiques sont données telles quelles et ont un statut autonome (par exemple, on ne considère pas que l'obligation est le dual de la permission), et par conséquent que les agents sont capables de recevoir les informations normatives sous le format présenté ci-dessus.

   Pourtant, l'extension du modèle avec des mécanismes de raisonnement sur les normes est un problème important qui reste un défi pour les améliorations à proposer à l'avenir.

8. Faire l'hypothèse du monde clos pour représenter les interdictions et les permissions, s'avère être une chose utile. Avec cette hypothèse, on peut avoir deux cas. Soit on postule que tout est par défaut interdit et que les permissions doivent être explicites. Soit, l'inverse, c.a.d. que tout est permis par

défaut sauf les cas d'interdiction qui sont explicites. Du point de vue de l'implémentation, cette hypothèse est conforme à la façon dont Prolog représente les prédicats et la négation (par échec). Dans la suite nous optons pour le deuxième cas. Ce choix est expliqué par le fait qu'au moment où on doit définir la violation d'une action interdite, il est plus naturel de parler de l'exécution d'un acte en présence de l'interdiction que en l'absence de la permission.

8.7 Violations

Comme nous l'avons précisé avant, l'utilisation des normes implique aussi de prendre en compte la possibilité de les transgresser. En général on calcule les violations à partir de la description d'un comportement idéal et des observations sur l'exécution passée ou courante du système. Dans cette thèse, on utilise des techniques de *model checking* où la violation est définie comme la non-exécution d'une action obligatoire ou l'exécution d'une action interdite.

8.7.1 Violations et actions duratives

L'utilisation des actions duratives pose des problèmes de signification de la violation. D'abord on cherche àà repérer les violations des normes qui expriment un règlement sur le comportement d'un agent dans un intervalle de temps. La violation sera donc définie en fonction du rapport entre l'intervalle d'exécution effective d'un acte, l'intervalle de référence et le type déontique exprimé dans la norme. Ensuite, les effets d'une action sont considérés traditionnellement dans l'IA [Nilsson, 1980], seulement si l'action a été exécutée dans sa totalité. On retrouve le même principe dans le jugement juridique. Pour la violation des obligations, de type $O(agent,\alpha,[t_1,t_2])$ l'utilisation du temps est plus simple à intégrer. Elle dépend du moment de l'observation. Plus précisément, on est dans une situation de violation, si par rapport à ce moment d'observation t, l'intervalle de référence $[t_1,t_2]$ est complètement dans son passé et qu'il n'y a aucun acte de type α exécuté par $agent$ entièrement dans cet intervalle. Si on introduit le prédicat $V(agent,\alpha,i)\ :\ Agent \times Action \times Interval$ pour formaliser la violation à l'instant t d'une norme qui décrit un comportement idéal par rapport à l'exécution d'une action α sur un intervalle i, une première partie de sa définition est donnée par :

$$\begin{aligned} holds(t,V(agent,\alpha,[t_1,t_2]) \quad \Leftarrow \quad & holds(t,O(agent,\alpha,[t_1,t_2])) \wedge [t_1,t_2] \prec [t,\infty) \ \wedge \\ & \forall act\ (instance\_of(act,\alpha) \Rightarrow \\ & holds(t,failed(agent,act,[t_1,t_2]))) \end{aligned} \tag{8.3}$$

Les problèmes deviennent plus complexes quand il s'agit de l'interdiction. Nous montrons deux exemples qui suggèrent deux situations différentes pour la violation. Une de type *stricte* qui caractérise l'exécution complète d'une action interdite, et une autre, dite *faible*, qui exprime la tentative d'exécution et pénalise les effets pouvant apparaître en cours d'exécution d'une action.

Dans le domaine juridique, une personne est considérée coupable si par exemple elle a bien planté son couteau dans le coeur de la victime (conformément à une expertise médico-légale *a posteriori*). De plus, dans le même contexte juridique, on pénalise aussi la tentative d'agression. Par exemple si le criminel ne touche pas une partie vitale, consciemment ou par hasard, ou s'il s'arrête au dernier moment dans son action et la victime reste en vie, l'acte est considéré comme tentative de crime, encore pénalisé mais avec indulgence. On remarque que finalement, l'action de tuer et la tentative de tuer, comme acte commencé mais inachevé, appartiennent à deux types différents d'action, qualifiés ainsi ultérieurement conformément à leurs effets. Ces actions doivent être réglementées séparément, encore en fonction de leurs effets finaux.

On trouve le deuxième exemple dans le contrôle d'utilisation d'une ressource. On veut interdire l'impression des fichiers trop volumineux ou qui occupent l'imprimante plus de temps. Disons que le système de contrôle ne peut pas savoir à l'avance combien de pages contient le fichier ou combien de temps dure l'impression. On peut seulement réaliser un contrôle ultérieur ou pendant l'impression (en utilisant un compteur ou un horloge). En imposant une limite supérieure en nombre de pages ou temps occupé, tout dépassement de ces limités représente une violation. On veut donc permettre la capacité de déduire qu'une violation a eu lieu, même si l'acte n'a pas été consommé dans sa totalité, par exemple, pour arrêter l'impression immédiatement, sans attendre la fin, ou dans le cas précédent pour empêcher le tueur d'achever sa crime. Dans le cas des normes dynamiques, l'effet intermédiaire suffisant pour générer la violation est le simple fait d'exécuter l'acte dans l'intervalle interdit.

La séparation en deux types de violation est peut être plus évidente si on précise que l'interdiction stricte caractérise les actes, considérés pour leurs effets finaux, et que l'interdiction faible s'applique aux activités considérées pour leurs effets intermédiaires, en déroulement. Comme, dans ce formalisme, on ne fait pas de différence entre l'exécution d'un acte et le déroulement d'une activité, on est obligé de le faire au niveau de la caractérisation déontique, et par conséquent de la violation. Ainsi, nous utilisons deux versions pour le prédicat d'interdiction, I_s pour l'interdiction stricte et I_f pour l'interdiction faible.

8.7.2 Violation stricte

Nous commençons par la définition de la violation d'une interdiction stricte $I_s(agent,\alpha,i)$. En fonction de la position du moment de l'observation par rapport au moment de référence nous avons détecté deux cas possibles :

1. i est dans le passé de t et il y a un acte de type α exécuté par $agent$ à l'intérieur de i (voir la figure 8.5a) ;

(a) $[t_1, t_2]$ dans le passé de t (b) t dans l'intervalle $[t_1, t_2]$

Figure 8.5: Violations strictes de $I_s(agent, \alpha, [t_1, t_2])$ à l'instant t

2. t est contenu dans l'intervalle i et il y a un acte de type α exécuté par $agent$ dans un intervalle commencé par i et terminé par t (voir la figure 8.5b).

Formellement cela s'écrit :

$$\begin{aligned}
holds(t,V(agent,\alpha,[t_1,t_2])) \Leftarrow\ & holds(t,I_s(agent,\alpha,[t_1,t_2])) \wedge \exists act,instance_of(act,\alpha) \wedge \\
& ([t_1,t_2] \prec [t,\infty) \wedge holds(t,done(agent,act,[t_1,t_2]))) \vee \\
& (in(t,[t_1,t_2]) \wedge holds(t,done(agent,act,[t_1,t)))
\end{aligned}$$

8.7.3 Violation faible

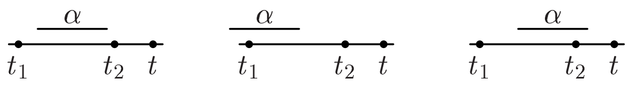
Dans le cas d'une interdiction faible $I_f(agent,\alpha,i)$ la situation est plus complexe par rapport au moment d'observation t. Ainsi nous avons détecté deux cas possibles de violation :

1. i est dans le passé de t et il y a un acte de type α exécuté par $agent$ dans un intervalle d'intersection non-vide avec l'intervalle i (voir la figure 8.6a) ;
2. t est contenu dans l'intervalle i et il y a un acte de type α exécuté par $agent$ dans un intervalle commencé par i et terminé par t (voir la figure 8.6b).

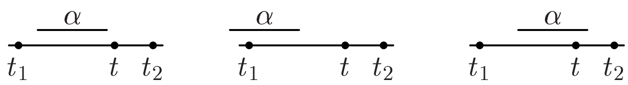
Formellement cela s'écrit :

$$\begin{aligned}
holds(t,V(agent,\alpha,[t_1,t_2])) \Leftarrow\ & holds(t,I_f(agent,\alpha,[t_1,t_2])) \wedge \exists act,instance_of(act,\alpha) \wedge \\
& ([t_1,t_2] \prec [t,\infty) \wedge (holds(t,done(agent,act,[t_1,t_2])) \vee \\
& holds(t_1,doing(agent,act)) \vee holds(t_2,doing(agent,act))) \vee \\
& (in(t,[t_1,t_2]) \wedge (holds(t,done(agent,act,[t_1,t)) \vee \\
& holds(t_1,doing(agent,act)) \vee holds(t,doing(agent,act)))
\end{aligned}$$

Exemple 8.10 Considérons comme exemple d'application le scénario suivant. Il existe dans une entreprise l'obligation pour tous les employés de produire des rapports annuels d'activité. Cette obligation est



(a) $[t_1, t_2]$ dans le passé de t



(b) t dans l'intervalle $[t_1, t_2]$

Figure 8.6: Violations faibles de $I_f(agent, \alpha, [t_1, t_2])$ à l'instant t

valable depuis 1980. Alice, une employée de cette entreprise, elle aussi est obligée cette année (2001) d'écrire son rapport annuel. Nous montrons deux cas possibles :

1. Alice a écrit son rapport pendant le mois de novembre 2001. Ainsi, il n'y a pas de violation (voir la figure 8.7). Le scénario est décrit par les propositions suivantes :

   $holds$([01/2001, $\rightarrow$[,
   $\quad O(X$,$Ecrire\_rapport$,[01/2001,12/2001]))

   $occurs$($Rapport$50,[10/2001,11/2001])

   $agent\_of$($Alice$,$Rapport$50)

   $instance\_of$($Rapport$50,$Ecrire_rapport$)

   Le modèle permet d'inférer les propriétés suivantes :

   $holds$([11/2001, $\rightarrow$[,$ocurred$($Rapport$50))

   $holds$([11/2001, $\rightarrow$[,$ocurred$($Rapport$50,[10/2001,11/2001]))

   $holds$([11/2001, $\rightarrow$[,$done$($Alice$,$Rapport$50))

   $holds$([11/2001, $\rightarrow$[,$done$($Alice$,$Rapport$50,[10/2001,11/2001]))

2. Alice n'a pas écrit son rapport. Le scénario est décrit par la proposition suivante :

   $holds$([01/2001, $\rightarrow$[,O($Alice$,$Ecrire\_rapport$,[01/2001,12/2001]))

   Le modèle doit permettre de montrer qu'il s'agit d'un cas de violation pour tout moment d'observation se trouvant après l'intervalle de référence (voir la figure 8.8) :

   $holds$([12/2001, $\rightarrow$[,V($Alice$,$Ecrire\_rapport$,[01/2001,12/2001]))

   Pour cela on va utiliser la formule (8.3). Certains lecteurs peuvent remarquer que cette formule pose de problèmes, notamment parce que, pour montrer la validité de la violation ci-dessus, on doit démontrer la formule : $\forall act\,(instance\_of(act,\alpha) \Rightarrow holds(t,failed(agent,act,[t_1,t_2])))$, qui

$$holds([01/2001, \rightarrow[, O(Alice, Ecrire\_rapport, [01/2001 - 12/2001]))$$
$$holds([11/2001, \rightarrow[, done(Alice, Rapport50, [10/2001 - 11/2001]))$$

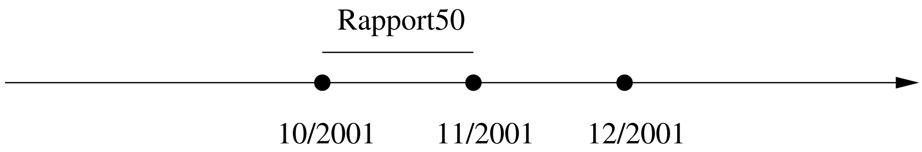


FIG. 8.7 – *Exemple d'obligation satisfaite*

$$holds([01/2001, \rightarrow[, O(Alice, Ecrire\_rapport, [01/2001 - 12/2001]))$$
$$holds([12/2001, \rightarrow[, V(Alice, Ecrire\_rapport, [10/2001 - 11/2001]))$$

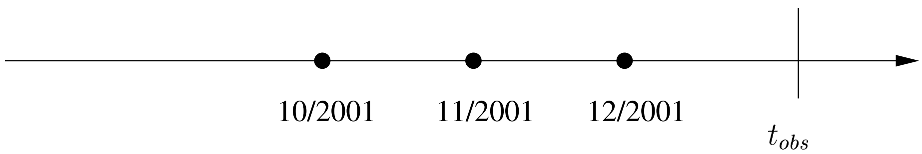


FIG. 8.8 – *Exemple de violation*

nécessite le parcours de tout l'ensemble d'actes définis dans le modèle. Pour eviter cet inconvénient, nous proposons dans le chapitre suivant, de doter le modèle temporel d'un mécanisme de preuve basé principalement sur Prolog et de tirer avantage des principes sur lesquels il a été conçu, celui du monde clos et de la négation par échec. Plus précisément, nous considérons que l'ensemble d'actes est un ensemble fini, construit seulement à partir des événements chronologiques visibles , qu'on peut connaître par une observation limitée. De plus, on ne garde que l'information positive sur ces événements. En reprenant la discussion sur la définition de l'échec d'un acte (voir le prédicat $failed$ et la section 8.5.2) la notion d'échec d'un acte dans un intervalle de temps est définie par l'échec de dériver si l'acte a été exécuté dans l'intervalle respectif. Ainsi la formule (8.3) sera adaptée et transformée pour que l'on puisse utiliser dans cette perspective. Pour plus détails sur la façon dont nous procédons, nous suggérons de voir le chapitre suivant et la définition du prédicat qui décrit la violation d'une action obligatoire, presentée dans l'annexe A.

8.8 Extension modale temporelle

L'utilisation d'un formalisme de premier ordre semble être suffisante pour répondre aux besoins de modélisation de propriétés déontiques et d'exécution des actions. En revanche, le fait qu'on utilise des normes de type conditionnel qui ont, en plus, une durée de vie, comme nous l'avons montré précédemment, pose des problèmes de représentation. Le principal problème vient du fait que pour décrire la durée de vie d'une norme qui est représentée par une formule conditionnelle (8.2), la qualification temporelle s'applique non pas seulement à un prédicat mais à toute une formule (non-atomique). Dans ce contexte, la notion de fluent, comme prédicat homogène sur un argument temporel est insuffisante et par conséquent on a besoin d'une extension du formalisme présent avec des opérateurs modaux qui s'appliquent à toute formule.

8.8.1 Syntaxe et sémantique

Nous proposons d'introduire dans le langage de premier ordre $\mathcal{L}_p$ présenté auparavant, l'opérateur temporel métrique[38] $\Box_{[t_1,t_2]}\phi$ avec ϕ une formule de $\mathcal{L}_p$ qui se lit ϕ est vraie sur tout l'intervalle $[t_1,t_2]$. Dans le langage ainsi obtenu $\mathcal{L}_m$ on peut exprimer maintenant les normes dynamiques :

$$\Box_{[t_1,t_2]}(OPI \Leftarrow \phi) \tag{8.4}$$

où OPI est l'un des prédicats déontiques et ϕ la condition d'application de la norme.

L'introduction de cet opérateur nous fait sortir du cadre de la logique de premier ordre et nous oblige à préciser sa sémantique, qui traditionnellement est donnée en termes de mondes possibles, ou encore de structures Kripke, simulées dans notre cas par des points temporels. Ainsi, la validité d'une formule dans le langage $\mathcal{L}_m$ est définie par rapport à un modèle $\mathcal{M}$ et un point temporel $t \in Time$.

Définition 8.11 (validité - version modale) *La validité d'une formule ϕ dans une $\mathcal{L}_m - \Sigma$-structure $\mathcal{M}$ au moment t sous la valuation de variables w, notée $\mathcal{M},t \models \phi$ est définie par :*

1. *Pour tout symbole de prédicat p d'arité n, $\mathcal{M},t \models p(t_1,\ldots,t_n)$ ssi $p_t^{\mathcal{M}}(\bar{w}(t_1),\ldots,\bar{w}(t_n))$ est dans $\mathcal{M}$;*
2. *$\mathcal{M},t \models \Box_{[t_1,t_2]}\phi$ ssi pour tout $t' \in [t+t_1,t+t_2]$ tel que $\mathcal{M},t' \models \phi$*
3. *$\mathcal{M},t \models \Diamond_{[t_1,t_2]}\phi$ ssi pour quelques $t' \in [t+t_1,t+t_2]$ tel que $\mathcal{M},t' \models \phi$*
4. *Pour les cas qui restent : $\neg\phi$, $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \Rightarrow \psi)$, $(\phi \Leftrightarrow \psi)$, $\exists x\ \phi$ et $\forall x\ \phi$ on utilise des définitions standards.*

38. La logique temporelle métrique a été proposée pour la première fois dans [Koymans *et al.*, 1983] afin de spécifier les systèmes temps-réel. Ces systèmes posedent des caracteristiques temporelles qui présentent un intérêt non seulement pour leurs propriétés qualitatives mais aussi quantitatives. On peut trouver une description des travaux plus récents dans [Koymans, 1989; Harel *et al.*, 1990; Brzoska, 1998].

Une formule ϕ de $\mathcal{L}_m$ est *valide (sous la valuation w) dans* $\mathcal{M}$ ssi $\mathcal{M},t_0 \models \phi$ avec t_0 le point d'origine de $Time$. La formule ϕ est *valide (dans le modèle* $\mathcal{M}$*)* ssi $\mathcal{M},t_0 \models \phi$ pour toute valuation de variables w. Cette définition est très importante, car elle sous-entend que toutes les formules que nous allons utiliser seront interprétées par rapport à un seul monde, à savoir celui qui correspond à l'origine sur l'axe du temps. Ce moment est différent de celui où on fait une observation t_{obs}, comme nous l'avons introduit pour décrire les propriétés qui changent en raison de l'apparition d'un événement, qui est à son tour défini (ou interprété) relativement à t_0 (en effet il s'agit du point absolu $t_0 + t_{obs}$).

La consistance et la conséquence logique sont définies de manière habituelle (voir la définition 8.6). Pour faciliter l'écriture, nous introduisons les variantes suivantes pour les opérateurs temporels métriques :

$$\Box\phi \equiv \Box_{(-\infty,\infty)}\phi \qquad \Box_t\phi \equiv \Box_{[t,t]}\phi$$
$$\Diamond\phi \equiv \Diamond_{(-\infty,\infty)}\phi \qquad \Diamond_t\phi \equiv \Diamond_{[t,t]}\phi$$

Notons que toute formule du langage $\mathcal{L}_p$ est désormais interprétée (et donc qualifiée) temporellement. Par exemple, on peut déduire facilement (à partir de la définition de l'opérateur $\Box$) la propriété d'homogénéité pour toute formule ϕ :

$$i \sqsubseteq j \wedge \Box_i\phi \Rightarrow \Box_j\phi$$

Pourtant, on veut utiliser dans le langage logique étendu $\mathcal{L}_m$ des formules qui sont indépendantes du temps (par exemple, qui correspondent aux prédicats non-fluents dans le modèle d'origine). La solution est de les considérer comme étant toujours valides $\Box\phi$ et, pour simplifier la notation, on propose de les décrire simplement par ϕ s'il n'y a pas d'ambiguïté.

Avec ces opérateurs temporels nous pouvons réécrire toutes les relations que nous avons introduites pour décrire l'apparition des événements, l'exécution des actions, les propriétés déontiques et déduire les informations correspondantes.

Ainsi pour les événements nous avons :

$$\begin{aligned}
occurs(e,[t_1,t_2]) &\Rightarrow \Box_{[t_2,\infty]}occurred(e,[t_1,t_2]) \\
\Box_k occurred(e,i) \wedge i \sqsubseteq j &\Rightarrow \Box_k occurred(e,j) \\
occurs(e,[t_1,t_2]) &\Rightarrow \Box_{[t_2,\infty]}occurred(e) \\
occurs(e,[t_1,t_2]) &\Rightarrow \Box_{[t_1,t_2]}occurring(e)
\end{aligned}$$

Pour les actes :

$$\Box_i occurred(act,j) \wedge agent\_of(agent,act) \equiv \Box_i done(agent,act,j)$$

$$\begin{aligned}
\Box_i occurred(act) \wedge agent_of(agent,act) &\equiv \Box_i done(agent,act)\\
\Box_i occurring(act) \wedge agent_of(agent,act) &\equiv \Box_i doing(agent,act)\\
\neg\Box_i done(agent,act,j) &\equiv \Box_i failed(agent,act,j)
\end{aligned}$$

Pour la violation, nous avons une définition similaire à celle introduite dans la partie précédente : $\Box_t V(agent,\alpha,[t_1,t_2]) \Leftarrow$

$$\begin{aligned}
&\Box_t O(agent,\alpha,[t_1,t_2]) \wedge [t_1,t_2] \prec [t,\infty) \wedge\\
&\quad \forall act\ (instance_of(act,\alpha) \Rightarrow \Box_t failed(agent,act,[t_1,t_2])))\vee\\
&\Box_t I(agent,\alpha,[t_1,t_2]) \wedge [t_1,t_2] \prec [t,\infty)\wedge\\
&\quad \exists act,instance_of(act,\alpha)\wedge\\
&\quad (\Box_t done(agent,act,[t_1,t_2]) \vee \Box_{t_1} doing(agent,act)\vee\\
&\quad \Box_{t_2} doing(agent,act))\vee\\
&\Box_t I(agent,\alpha,[t_1,t_2]) \wedge in(t,[t_1,t_2])\wedge\\
&\quad \exists act,instance_of(act,\alpha)\wedge\\
&\quad (\Box_t done(agent,act,[t_1,t]) \vee \Box_{t_1} doing(agent,act)\vee\\
&\quad \Box_t doing(agent,act))
\end{aligned}$$

8.8.2 Traduction dans une logique de premier ordre

L'introduction des opérateurs modaux pose des problèmes d'implémentation. Parmi les solutions proposées dans la littérature, nous retenons la méthode générale de traduction d'une formule modale dans une logique de premier ordre [Ohlbach, 1988; Auffray et Enjalbert, 1989; Debart *et al.*, 1992] . Plus précisément, nous allons utiliser les résultats présentés dans [Brzoska, 1998] qui s'appliquent au raisonnement temporel dans la programmation logique. La méthode décrite dans cet article consiste à transformer une formule modale temporelle dans une logique de premier ordre. L'idée de base est d'ajouter à chaque prédicat un argument temporel et d'exprimer les relations temporelles, décrites par les opérateurs temporels, par des formules dans une logique classique. Cette méthode ne s'applique qu'à un fragment de la logique temporelle métrique que nous avons présentée plus haut, plus précisément, qui est formée seulement par des formules Horn universelles bornées (sur un intervalle)[39] et présentées sous le format décrit par la grammaire BNF suivante :

$$F \ ::= \ A \mid \Box_i F \mid F \Leftarrow G$$

où F exprime une formule, A une formule atomique, i un intervalle et G un but universel borné, défini par

$$G \ ::= \ \varepsilon \mid A \mid \Diamond_i G \mid G \wedge G \mid \Box_{i'} G$$

39. Dans l'article [Brzoska, 1998] le terme utilisé est celui de *bounded universal (modality) Horn formulae*.

où ε est le but vide et i' un intervalle borné.

La fonction de traduction des formules temporelles en des formules appartenant à $\mathcal{L}_p$ est décrite par :

$$\begin{array}{rcl} \Pi(A) & = & \pi(A,0,\varnothing) \\ \pi(p(\mathbf{r}),t,C) & = & p(t,\mathbf{r}) \\ \pi(\Box_i A,t,C) & = & \forall x(\{x \in i\} \Rightarrow \pi(A,t+x,in(x,i) \ \cup \ C)) \\ \pi(\Diamond_i A,t,C) & = & \exists x(\{x \in i\} \wedge \pi(A,t+x,in(x,i) \ \cup \ C)) \\ \pi(A \Leftarrow B,t,C) & = & \pi(A,t,C) \Leftarrow \pi(B,t,C) \\ \pi(A \wedge B,t,C) & = & \pi(A,t,C) \wedge \pi(B,t,C) \end{array}$$

où **r** est un vecteur de termes $t_1,\ldots,t_n$ et C exprime un ensemble de contraintes temporelles.

Ainsi, la traduction des formules Horn universelles bornées génére des formules avec contraintes du type

$$F \ ::= \ A \mid \forall x(C \Rightarrow F) \mid F \Leftarrow G$$

et des buts

$$G \ ::= \ C \mid A \mid G \wedge G \mid \exists x G \mid \forall y(in(y,[l^-,l^+]) \Rightarrow G)$$

Brozska remarque que grâce aux équivalences logiques $((B \Rightarrow A) \Leftarrow C) \equiv (A \Leftarrow B \wedge C)$ et $((A \Leftarrow B) \Leftarrow C) \equiv (A \Leftarrow (B \wedge C))$ on peut se concentrer seulement sur les formules du type $A \Leftarrow C \wedge G$.

On note que cette traduction dans une logique classique transforme les opérateurs temporels en des restrictions sur le temps. C'est une caractéristique importante car on peut séparer ces restrictions du reste de la formule obtenue et utiliser des méthodes de raisonnement (par contraintes) spécialisées [Bürckert, 1991; Baumgartner et Stolzenburg, 1995; Stratulat, 1997].

8.9 Discussions et conclusions

Dans ce chapitre, nous avons introduit un modèle formel permettant la spécification et la construction des outils qui vont équiper les divers éléments composant un SAN. Avant de passer à leur présentation nous faisons dans la suite quelques remarques concernant les choix que nous avons faits dans la construction du modèle.

L'objectif majeur de ce modèle est d'offrir la possibilité de confronter les comportements effectifs des agents et les comportements désirables ou idéaux, et tout cela dans un contexte temporel, car ces comportements ont besoin de temps pour se manifester. Pour décrire l'évolution du système, ou plus généralement du monde dans le temps, nous avons fait appel à des concepts déjà étudiés dans l'IA, comme les événements ou les fluents. Pourtant, comme nous l'avons montré dans le chapitre 7, on ne peut pas utiliser directement les modèles traditionnels pour répondre à notre objectif. Par exemple, on veut représenter dans le même modèle des actions duratives et des propriétés temporelles qui nous permettent de savoir quel est l'état d'avancement dans l'apparition de ces actions, si elles sont en train de se produire ou si elles se sont déjà produites. Ainsi, il a fallu adapter et reprendre certains concepts et propriétés àcet effet. Une autre amélioration vient de l'intérêt d'avoir une implémentation directe du modèle. Nous avons introduit la notation *holds* qui sera utilisée dans le chapitre suivant pour implémenter un interpréteur qui répond à des requêtes concernant la validité d'une propriété (ou fluent) dans un intervalle de temps.

En ce qui concerne la représentation des normes, notre option pour des normes dynamiques est tout à fait légitime, car les actions prennent du temps à être exécutées. Ainsi, une norme représente pour un agent une source de deux types de contraintes qu'il doit prendre en compte : déontiques et temporelles.

En ce qui concerne la représentation des concepts déontiques, nous avons utilisé pour le moment des prédicats ordinaires, sans aucune interprétation spéciale. Cela signifie qu'on ne prend pas en considération les diverses relations qui peuvent exister entre eux, par exemple le fait qu'on peut définir un concept déontique par l'intermédiaire de l'autre (la permission et le dual de l'obligation), comme c'est le cas dans la logique déontique standard. Ce choix est motivé d'une part par le fait que nous avons voulu rester dans un cadre de premier ordre pour des raisons d'implémentation, la logique déontique standard étant construite avec des techniques de logique modale. D'autre part, les diverses propriétés des concepts déontiques génèrent des paradoxes quand on les formalise à la manière de la démarche modale. Pourtant, l'utilisation de ces propriétés est un point clé dans la spécification correcte des normes et elle va faire l'objet d'améliorations futures.

L'extension modale du modèle temporel est un travail complexe. Elle est nécessaire seulement si on veut attribuer explicitement une durée de vie à une norme qui, pour être décrite, nécessite l'utilisation d'une formule logique complexe. Par exemple, si on fait un audit sur l'exécution passée du système, on doit traiter les normes qui étaient en place à ce moment là, même si actuellement, lorsqu'on fait l'audit, ces normes ne sont plus en viguer. Bien sûr que le modèle de premier ordre est préférable car on dispose déjà d'une implémentation le concernant. Par contre, le modèle modal devient caduc dans la situation où on ne prend pas en compte la durée de vie d'une norme, c.a.d. quand les normes ne dépendent pas du temps ou quand leur vie est donnée par l'intervalle $(-\infty,\infty)$. Ce cas correspond à un compromis ou à une simplification où on fait référence seulement aux normes qui sont en place au moment présent. Si cette approximation ne pose pas de problème, on peut utiliser le modèle de premier ordre pour représenter des normes nécessitant des formules complexes. On remarque encore que dans ce cas, les normes ne sont plus descriptibles avec des fluents et ne peuvent pas être des « arguments » pour *holds*.

Chapitre 9

Implémentation et outils

Tout cela est bien,
mais quand irons-nous dans la Lune ?
Jules Renard, *Journal*

9.1 Implémentation

Dans cette partie nous décrivons l'implémentation en PROLOG ECLIPSE [eclipse, 2001] du langage du premier ordre $\mathcal{L}_p$ présenté ci-dessus. La source complète de cette implémentation est décrite dans l'annexe A. Le traitement temporel est présenté plus en détail à l'annexe B qui porte sur l'implémentation des relations temporelles sur les intervalles, et à l'annexe D qui décrit l'utilisation de la bibliothèque des contraintes sur les domaines finis proposée par PROLOG ECLIPSE. Nous avons utilisé toutes les capacités de démonstration automatique offertes par l'interpréteur Prolog. De plus, on a employé des techniques de méta-programmation pour implémenter l'élément principal du langage, la notation *holds*.

On rappelle que cette notation a été introduite pour permettre le raisonnement d'une façon unitaire sur les propriétés homogènes temporelles du monde. Par exemple, on veut savoir quelles sont les propriétés valides à un certain moment ou sur un intervalle de temps. Ou l'inverse, quand une propriété est valide.

Plus précisément, nous avons défini le méta-prédicat `holds/2` qui comporte deux arguments : une valeur temporelle (instant ou intervalle) et une propriété relationnelle de type *fluent*. Techniquement, ce prédicat se comporte comme un interpréteur qui implémente comme fonction générale un démonstrateur pour une logique de deuxième ordre. Comme Prolog, on utilise cet interpréteur pour ses effets secondaires, c.a.d. pour obtenir des instances de variables qui apparaissent dans la théorème à démontrer, par rapport à un ensemble d'axiomes. Ainsi, une requête a la forme générale `holds(Time, Fluent)` et le résultat consiste en l'instanciation des deux variables `Time` et `Fluent`. On peut forcer l'exécution de l'interpréteur pour nous offrir tous les instances possibles, c.a.d. pour montrer quels sont les fluents valides, sur quels intervalles ou à quel moment.

9.1.1 Hypothèse sur l'implémentation

L'implémentation de cet interpréteur est basée sur certaines hypothèses. La première se réfère à l'hypothèse du monde clos qui suppose qu'au moment de l'« exécution » d'une requête nous disposons d'une description complète du monde. Une conséquence directe est que toutes les informations contenues dans la base sont figées jusqu'à l'obtention d'une réponse, ou que l'exécution de la requête est exécutée *off-line* sans que le monde change entre temps. La modification de ces informations peut se faire seulement entre deux requêtes. Ces informations concernent tout ce qui touche à la définition du monde par les prédicats non-fluents :

- les événements - leurs apparitions, noms, types, éventuellement leur acteur ;
- les actions ou les types d'événements - les noms, les hiérarchies ;
- les agents - les noms, l'appartenance aux groupes, les rôles attribués, les capacités ;
- les hiérarchies des rôles ;
- etc.

Une autre utilisation de l'hypothèse du monde clos est de pouvoir représenter des valeurs implicites pour les propriétés déontiques. Plus précisément, cette hypothèse stipule que si une propriété P n'est pas dérivable, alors $\neg P$ est dérivable. Notre interpréteur, et l'interpréteur Prolog en général, ne modélise que partiellement la négation de premier ordre, car les clauses Horn ne peuvent pas être utilisées pour dériver des propriétés négatives. En échange, ils utilisent un autre type de négation, appelée négation par échec [Clark, 1978], qui dit que la propriété $\neg P$ est dérivable si on prouve que P ne peut pas être prouvé. La conséquence directe est la suivante : si le résultat d'une requête est négatif, cela signifie soit que la propriété questionnée n'est pas valide, soit qu'on ne peut pas montrer (calculer) sa vérité. Ce résultat nous convient car il facilite la description des propriétés déontiques. Par exemple, on peut considérer que tout est autorisé implicitement, mais que toute interdiction doit être explicite.

Enfin, une dernière hypothèse concerne la définition des fluents homogènes. Elle doit se faire de façon axiomatique pour capter d'abord la propriété d'homogénéité, puis pour permettre le raisonnement sur leur comportement. De plus, pour permettre l'exploitation d'un système Prolog, on doit restreindre notre intérêt seulement aux fluents relationnels, par contraste avec les fluents fonctionnels qui posent certains problèmes au moment de l'évaluation dans la programmation logique.

Ainsi, nous avons introduit pour tout prédicat non-fluent utilisé dans le langage $\mathcal{L}_p$, son correspondant en Prolog. Par exemple : `start/2`, `finish/2`, `occ/2`, `does/3`, `instance_of/2`, `subclass/2`, `agent/2`, etc. Idem pour les prédicats décrivant les relations temporelles à la Allen (voir l'annexe B).

9.1.2 Implémentation de *holds*

En ce qui concerne les prédicats fluents, leur définition est, comme on l'a déjà précisé, donnée d'une façon axiomatique par l'intermédiaire du prédicat `holds/2` :

1. Pour des fluents qui sont utilisés seulement pour leur propriété d'homogénéité, on utilise des axiomes du type :

   ```
   holds(Time, fluent(Arg1, ..., Argn)) :-
             fluent(Interval, Arg1, ..., Argn), Time :: Interval.
   ```

   où le symbole `::` représente le prédicat d'appartenance à un ensemble de domaines finis (voir l'annexe D).

   Avec ce genre d'axiome on peut raisonner d'une manière uniforme sur une spécification initiale du monde, donnée en terme des fluents de type :

   ```
   fluent(Interval, Arg1, ..., Argn).
   ```

2. Pour les fluents qui ont un comportement spécifique, comme la plupart des propriétés étudiées dans ce chapitre, leur description axiomatique dans le langage $\mathcal{L}_p$ se transpose facilement en Prolog. Par exemple, pour tester si on est en train d'assister à l'apparition d'un événement, on utilise le fluent `occuring` qui conformément à sa définition (voir la section 8.5) s'écrit en Prolog sous la forme :

   ```
   holds(Time, occurring(Event)) :-
              occ(Event, Interval), in(Time, Interval).
   ```

9.1.3 Exemples d'utilisations

Pour illustrer ce qui précède, considérons les exemples des figures 8.3 et 8.4. La description du monde peut se faire en Prolog de la manière suivante :

```
start(vol330, 9).
start(presentation, 11).
finish(vol330, 10).
finish(presentation, 12).
agent(alice, presentation).
```

Maintenant, on peut déterminer par l'utilisation du `holds` quelles propriétés sont vraies et à quel moment ou intervalle de temps. L'implémentation du prédicat `holds` est décrite dans l'annexe A.

Par exemple, on veut savoir dans quel intervalle de temps on peut parler du `vol330` comme étant un événement passé.

```
[eclipse 2]: holds(When, occurred(vol330)).

When = When{[10..10000000]}
yes.
```

Le résultat obtenu consiste en l'instanciation de la variable `When` avec le domaine fini `[10..10000000]` qui nous indique que, pour tout moment supérieur à 10, on peut parler du `vol330` au passé. On remarque que Prolog nous impose une restriction sur la valeur maximale de la borne supérieure d'un domaine qui est limitée à 10000000, voire à -10000000 pour la borne inférieure. Ainsi, pour représenter tout intervalle non-borné défini dans le modèle formel, on est obligé de remplacer les symboles $]\leftarrow$ et $\rightarrow[$ par ces valeurs maximales.

Pour instancier les variables domaine, la bibliothèque utilisée fait appel à des mécanismes de propagation de contraintes. Dans l'exemple précédent, on peut demander en plus quel est l'intervalle dans lequel l'événement s'est produit, en utilisant le prédicat `occured/2` qui correspond à la deuxième définition de $occurred(e,i)$ présentée dans le chapitre 8 :

```
[eclipse 4]: holds(When, occurred(vol330, Int)).

When = When{[10..10000000]}
Int = T3{[-10000000..9]}, T4{[10..10000000]}

Delayed goals:
        When{[10..10000000]} - T4{[10..10000000]}#>=0
yes.
```

Pour la variable `When`, on obtient le même résultat. La variable `Int` représente l'intervalle de référence dans lequel peut se trouver (qui inclut) l'intervalle effectif d'apparition du `vol330`. La représentation de cet intervalle est de la forme `(T3, T4)`, où `T3` et `T4` sont à leur tour des variables domaine qui restent à être instanciées. Ainsi on obtient que `T3`, la borne inférieure de `Int`, peut prendre une valeur dans l'intervalle `[-10000000..9]`, voire `[10..10000000]` pour `T4`.

L'expression `When{[10..10000000]} - T4{[10..10000000]}#>=0` exprime une contrainte qui peut être propagée pour être prise en compte par d'autres traitements. Elle est représentée par une inégalité qui montre que `T4`, la borne supérieure de `Int` qui est l'intervalle de référence d'un événement, doit être dans le passé de `When`, le moment d'observation. Cette inégalité est donnée par la façon dont le prédicat `occurred/2` est défini (implémenté).

Avec ces précisions, on peut continuer et demander une réponse exhaustive à la question de savoir quelles sont toutes les propriétés valides (`What`) et à quels moments (`When`) :

```
[eclipse 2]: holds(When, What).

When = When{[12..10000000]}
What = occurred(presentation)     More? (;)

When = When{[10..10000000]}
What = occurred(vol330)     More? (;)
```

```
When = When{[12..10000000]}
What = occurred(presentation, (T3{[-10000000..11]}, T4{[12..10000000]}))

Delayed goals:
        When{[12..10000000]} - T4{[12..10000000]}#>=0     More? (;)

When = When{[10..10000000]}
What = occurred(vol330, (T3{[-10000000..9]}, T4{[10..10000000]}))

Delayed goals:
        When{[10..10000000]} - T4{[10..10000000]}#>=0     More? (;)

When = When{[11, 12]}
What = occurring(presentation)     More? (;)

When = When{[9, 10]}
What = occurring(vol330)     More? (;)

When = When{[12..10000000]}
What = done(alice, presentation)     More? (;)

When = When{[12..10000000]}
What = done(alice, presentation, (T3{[-10000000..11]}, T4{[12..10000000]}))

Delayed goals:
        When{[12..10000000]} - T4{[12..10000000]}#>=0     More? (;)

When = When{[11, 12]}
What = doing(alice, presentation)     More? (;)

no (more) solution.
```

Les 9 réponses obtenues représentent toutes les solutions possibles pour qu'un fluent soit vrai dans un intervalle de temps. Ces résultats sont obtenus à partir des informations contenues dans la base de faits décrivant le monde et des définitions des fluents.

Si on ajoute dans la base de faits d'origine l'interdiction pour tout le monde de s'exprimer en public dans l'intervalle `(9, 14)` et qui a une durée de vie dans l'intervalle `[0..24]`, exprimable en Prolog par :

```
instance_of(presentation, parler_en_public).
i(0..24, Tout_le_monde, parler_en_public, (9, 14)).
```

on obtient l'exécution suivante pour la requête qui cherche à obtenir toutes les violations existantes :

```
[eclipse 3]: holds(When, v(Who, What, Interval)).

When = When{[14..24]}
Who = alice
What = parler_en_public
Interval = 9, 14

Delayed goals:
        _214{[14..10000000]} - When{[14..24]}#>=0     More? (;)

When = When{[12..14]}
Who = alice
What = parler_en_public
Interval = 9, 14     More? (;)

When = When{[11, 12]}
Who = alice
What = parler_en_public
Interval = 9, 14     More? (;)

no (more) solution.
```

On observe que l'introduction de cette interdiction génère plusieurs réponses. Elles indiquent qu'il s'agit d'une violation de l'interdiction de parler en public si le moment d'observation est dans l'un des intervalles `[14..24]`, `[12..14]` ou `[11..12]`. En effet, la violation existe parce que la présentation a eu lieu dans l'intervalle `[11..12]` qui est donc en conflit avec l'intervalle de référence interdit `(9,14)`. Ensuite, l'intervalle de validité de cette violation est borné inférieurement par 11 qui est le moment de début de la présentation et supérieurement par 24 qui est l'instant où l'interdiction prend fin, comme durée de vie. On obtient plusieurs réponses, car ces résultats correspondent en fait aux différents cas traités par la définition du prédicat de violation. On rappelle que ce prédicat est défini par la disjonction de plusieurs situations possibles. Par conséquent, le résultat est tout à fait correct. Par contre, cette remarque suggère en même temps la possibilité d'améliorer ce résultat, en regroupant plusieurs réponses correctes à la même requête. Par exemple on peut enrichir l'interpréteur par un mécanisme qui collecte des résultats multiples, et qui pourrait donner comme réponse à la requête dans l'exemple ci-dessus un seul résultat pour les deux dernières réponses, c.a.d. quand la variable `Time` est dans l'intervalle `[11, 14]` au lieu de deux intervalles `[11, 12]`, respectivement `[12, 14]`.

9.2 Ordonnancement

9.2.1 Introduction

Une autre application du modèle présenté dans le chapitre précédent concerne la construction des outils pour les agents normatifs qui doivent planifier ou organiser leurs actions afin d'éviter les cas de violation. Par conséquent, nous proposons un algorithme d'ordonnancement qu'on appelle *déontique* qui prend en compte la présence de deux types de contraintes, temporelles et déontiques.

9.2.2 Problème général d'ordonnancement

Le problème d'ordonnancement est un problème plus général étudié principalement par le domaine des Recherches Opérationnelles et utilisé dans plusieurs domaines comme l'économie et l'IA. Ce problème consiste à trouver une organisation ou un ordonnancement d'un ensemble d'activités ou de tâches dans un système. Le processus d'organisation doit tenir compte de certaines variables comme la nature des tâches, les ressources disponibles dans le système, les contraintes et les objectifs. Par exemple, on doit trouver une organisation de l'utilisation des ressources qui respecte certaines contraintes temporelles sur la date de début de chaque activité, problème connu comme étant difficile, voir NP-complet [Blazewicz *et al.*, 1996].

Les contraintes temporelles sont de deux types :

1. contraintes d'antériorité : une tache T1 ne peut commencer qu'après la fin d'une tâche T2.
2. contraintes de localisation temporelle : la date d'exécution d'une tâche est bornée soit inférieurement, soit supérieurement soit les deux à la fois par des dates limites.

Les objectifs de l'ordonnancement consistent à définir des critères pour mesurer la qualité de la solution obtenue, c.a.d. quand une solution est optimale ou satisfaisante. Comme critères les plus souvent utilisés nous énumérons, par exemple dans un processus de fabrication, la minimisation de la durée totale de fabrication, des temps d'inactivité des machines, du coût total induit par une solution d'ordonnancement ou simplement le respect des délais pour chaque activité.

9.2.3 Problème d'ordonnancement déontique

Dans le cas des agents normatifs, qui se trouvent sous l'influence d'un ensemble de normes à respecter, nous voulons les doter d'un module d'ordonnancement qui les aide à trouver une organisation de leurs actions pour qu'ils puissent éviter les violations des normes. Ainsi nous proposons d'adapter le problème général d'ordonnancement à un ordonnancement déontique. Dans ce contexte, les tâches à organiser sont les actions à exécuter par un agent et les contraintes sont les contraintes déontiques temporelles qui expriment des contraintes de localisation temporelle plus s'il s'agit d'une action interdite ou obligatoire. On rappelle que les actions ont une durée d'exécution et qu'une norme prescrit si une action doit être exécutée ou évitée dans un intervalle de temps de référence.

En ce qui concerne l'objectif du problème d'ordonnancement déontique, nous avons deux cas possibles suivant le type d'utilisation des normes dans un SAN.

1. Dans le cas où on tente seulement d'éviter la violation des normes, l'objectif du problème d'ordonnancement déontique est de trouver une solution qui respecte seulement les contraintes temporelles et déontiques : exécuter les actions obligatoires et éviter les actions interdites dans l'intervalle de référence.
2. Dans le cas où l'ensemble de normes est renforcé par un mécanisme de pénalités [40], représenté dans la norme sous la forme d'un coût à payer en cas de violation, l'objectif du problème d'ordonnancement déontique est de trouver la solution optimale, c.a.d. celle qui permet à l'agent d'exécuter ses actions futures pour qu'il paie le coût minimum.

Dans la suite, on s'intéresse au deuxième cas pour lequel nous proposons un algorithme. L'algorithme est composé de deux sous-parties : une première partie (qui est en effet l'algorithme d'ordonnancement déontique) qui construit une solution possible avec le coût correspondant et une deuxième partie qui cherche la solution optimale parmi les solutions proposées par la première partie.

La méthode retenue pour construire l'algorithme d'ordonnancement déontique est celle où on traite le problème d'ordonnancement comme un problème de satisfaction des contraintes temporelles (*Temporal Constraint Satisfaction Problem* ou TCSP). Dans un problème TCSP, on sépare l'information temporelle du reste du problème et on la traite séparément comme un ensemble des contraintes pour lesquelles on utilise des mécanismes de raisonnement spécialisé. Par conséquent, on considère les contraintes des localisations temporelles comme les contraintes de TCSP et pour les contraintes déontiques, on essaie de proposer une méthode par construction progressive. Cette méthode procède à la construction d'une solution globale à partir d'une solution partielle complétée au fur et à mesure de la résolution. Plus précisément, pour chaque action α_k prescrite par une norme, on génère une instance planifiée (possiblement vide) et on collecte les contraintes temporelles ainsi créées dans l'ensemble global des contraintes $\mathcal{C}$. La création d'un acte et la propagation des contraintes utilisent le mécanisme de *backtracking* qui permet la recherche d'une autre solution en cas d'échec ou la génération des solutions multiples (à condition qu'elles existent).

Dans la notation utilisée pour décrire l'algorithme d'ordonnancement déontique présenté dans la figure 9.2.3, nous avons indexé les contraintes sur les intervalles de temps avec $\mathcal{C}$ pour montrer qu'elles seront ajoutées à l'ensemble général des contraintes seulement si elles préservent la consistance globale de l'ensemble. D'autres notations sont comme suit :

– $\mathcal{L} = \{(\alpha_k, [s_{\alpha_k}, f_{\alpha_k}])\}_k$ représente la solution trouvée par l'algorithme déontique donnée sous la

40. L'agent a la possibilité de décider de violer une norme mais la violation est pénalisée.

forme d'une liste d'actions futures α_k avec les temps de commencement s_{α_k} et de fin f_{α_k} ;

- $\mathcal{D} = \{duration(\alpha_k,d_k)\}_k$ liste qui contient les estimations des durées d'exécution de chaque action ;
- $\mathcal{O}_{\mathcal{F},t}$ et $\mathcal{I}_{\mathcal{F},t}$ sont les contraintes déontiques qui obligent, respectivement interdisent, l'exécution de l'action α dans un intervalle $\mathcal{F}$:

$$\mathcal{O}_{\mathcal{F},t} = \{O(\alpha,i,cost)|holds(t,O(\alpha,i,cost) \wedge i \sqsubset \mathcal{F}\}$$
$$\mathcal{I}_{\mathcal{F},t} = \{I(\alpha,i,cost)|holds(t,I(\alpha,i,cost) \wedge i \cap \mathcal{F} \neq \varnothing\}$$

- $\mathcal{C}$ représente les contraintes temporelles.

On note que l'algorithme d'ordonnancement déontique propose une solution donnée par la liste des actes à exécuter et de leur temps d'exécution, plus le coût global afférent. L'algorithme peut être étendu pour prendre en compte des contraintes temporelles d'antériorité, dans le cas où il existe des relations d'ordre d'exécution parmi les actions d'un agent. Ces contraintes seront traitées au même niveau que les autres contraintes temporelles.

Une fois une solution obtenue, on veut trouver celle qui coûte le moins cher. C'est le rôle de la deuxième partie de l'algorithme qui, pour la réaliser, utilise la méthode *branch and bound* (voir l'annexe E).

Dans l'implémentation de cet algorithme, nous avons utilisé pour représenter les intervalles de temps la bibliothèque des domaines finis proposée par Prolog Eclipse (voir également l'annexe D). Plus précisément, un intervalle est un domaine. Le choix est motivé par le fait que cette bibliothèque propose d'une part un mécanisme de propagation des contraintes qui a été utilisé pour représenter le mécanisme des contraintes temporelles. Par exemple, nous avons représenté les relations d'Allen par l'intermédiaire des contraintes temporelles. D'autre part, cette bibliothèque propose le prédicat `minimize(+predicat,cost)` qui implémente la méthode *branch and bound* pour trouver une solution (instance) de la variable `predicat` qui minimise `cost`.

A partir du modèle déjà implémenté en Prolog, il a été plus simple de construire un algorithme qui génère pour chaque obligation et interdiction une instance planifiée de l'action et qui récupère toutes les contraintes qui apparaissent dans la suite. On a utilisé la même bibliothèque de contraintes pour les domaines finis et le prédicat `minimize/2`.

En utilisant la méthode *branch and bound*, ce prédicat nous donne une solution optimale.

La complexité de cet algorithme reste très élevée $O(exp(n))$ à cause de l'utilisation du *backtracking* et de la méthode *branch and bound*.

procedure $schedule$

  input : $\mathcal{O}_{\mathcal{F},t}, \mathcal{I}_{\mathcal{F},t}, \mathcal{D}$

  output : $\mathcal{L}, Cost, \mathcal{C}$

  $\mathcal{L} = \{\}; Cost =_{\mathcal{C}} 0$;

  for $O(\alpha,i,cost) \in \mathcal{O}_{\mathcal{F},t}$

    $generate\_act((\alpha,i,cost),s_\alpha,f_\alpha,cost_\alpha)$;

    $\mathcal{L} = \mathcal{L} \cup \{(\alpha,[s_\alpha,f_\alpha])\}$;

    $Cost =_{\mathcal{C}} Cost + cost_\alpha$;

end

procedure $generate\_act$

  input : $(\alpha,i,cost)$

  output : $s_\alpha,f_\alpha,cost_\alpha$

  $generated_\alpha =_{\mathcal{C}} \{0,1\}$;

  if $generated_\alpha$

    $cost_\alpha =_{\mathcal{C}} 0$;

    $[s_\alpha,f_\alpha] \sqsubseteq_{\mathcal{C}} i$;

    $d_\alpha = duration(\alpha)$;

    $f_\alpha =_{\mathcal{C}} s_\alpha + d_\alpha$;

    $\mathcal{I}_\alpha = \{(i_k,cost_k) | I(\alpha,i_k,cost_k) \in \mathcal{I}_{\mathcal{F},t}\}$;

    for $(i_k,cost_k) \in \mathcal{I}_\alpha$

      if $i_k \cap [s_\alpha,f_\alpha] \neq \varnothing$

        /* interdiction violée */

        $cost_\alpha =_{\mathcal{C}} cost_\alpha + cost_k$;

  else

    /* obligation violée */

    $cost_\alpha =_{\mathcal{C}} cost$;

    $s_\alpha =_{\mathcal{C}} null$; $f_\alpha =_{\mathcal{C}} null$;

end

FIG. 9.1 – *Ordonnancement déontique*

9.3 Communication

9.3.1 Introduction

Dans cette partie nous décrivons des aspects liés à la communication dans un SAN. D'abord, on montre le caractère normatif de la communication, puis on essaye d'introduire de nouveaux actes spéciaux de communication dits déontiques, modélisés par l'intermédiaire du formalisme proposé dans le chapitre précédent.

9.3.2 Communication par actes

La démarche standard pour décrire l'interaction dans les SMA est celle qui considère la communication entre agents comme étant formée par l'exécution des actions spéciales appelées actes de communication ou *performatifs* [Searle, 1983]. Elle a été introduite pour répondre au caractère hétérogène de l'interaction, expliqué par l'origine différente des agents qui y participent. Pour expliquer le sens qu'on attribue aux performatifs la plupart des travaux [Sadek, 1991; Labrou et Finin, 1998] utilisent un modèle cognitif basé sur les notions de croyances, désirs et intentions (BDI). Quand quelqu'un envoie un message, c'est parce que il y a une certaine raison, ou une intention explicite. Par conséquent, celui qui reçoit le message doit faire la même chose, d'attribuer au message l'intention correspondante. La transmission d'un message est donc vue comme étant un acte qui a un certain effet (appelé *illocutoire*) sur celui qui le reçoit. En pratique, il est possible d'attacher au message l'intention de celui qui l'envoie en spécifiant le type de message. Dans la plupart des cas, les messages échangés entre agents peuvent être classés [Singh, 1998] dans l'une des catégories suivantes :

- *assertifs*, qui informent. Le destinataire du message doit en principe mettre à jour ses connaissances en fonction du contenu du message qui est une simple information. Ce type de message n'implique (directement) aucune réaction de la part du destinataire. Ex : « *La porte est ouverte.* »
- *directifs* qui demandent. Ils impliquent un changement dans le comportement de celui qui reçoit le message. Ex : « *Ouvre la porte !* » ou « *Est-ce que la porte est ouverte ?* »
- *commissifs*, qui promettent. Ex : « *Je vais ouvrir la porte !* »
- *permissifs*, qui autorisent l'exécution d'une action. Ex : « *Tu peux ouvrir la porte.* »
- *prohibitifs*, qui interdisent l'exécution d'une action. Ex : « *Tu ne peux pas ouvrir la porte.* »
- *déclaratifs*, qui changent les propriétés du monde par leur simple réalisation. Ex : « *Je vous déclare mariés !* »
- *expressifs*, qui expriment des évaluations sur le monde ou sur l'agent lui-même (émotions). Ex :« *Ce n'est pas bien ce que tu as fait !* »

Pour illustrer, considérons le cas de *i informs j* que « *Il est midi.* ». Le contenu du message est la proposition « *Il est midi.* » et l'acte est celui d'informer. L'interprétation du message va avoir des effets sur les deux agents en même temps et elle est réalisée individuellement par chacun d'entre eux. En fonction de la manière dont on fait l'interprétation du message, on parle de deux types de perspectives : individuelle et publique. Dans la perspective individuelle, la sémantique d'un acte est donnée du point de vue de l'agent qui le produit. C'est le cas des langages comme KQML [Finin *et al.*, 1995] et FIPA-ACL. Dans la perspective publique, on donne une interprétation de l'acte vue de l'extérieur. Par exemple, Pitt et Mamdani [Pitt et Mamdani, 2000] proposent une spécification basée sur une séparation entre l'interprétation donnée par le récepteur et celle donnée par celui qui produit le message à propos des motivations pour exécuter le performatif (appelées *triggers*) et des effets qu'il génère (appelés *tropismes*)[41]. Les triggers et les tropismes sont des spécifications locales à chaque agent, et elles peuvent être différentes. Par exemple, l'agent qui envoie le message sait clairement claire pourquoi il l'a fait, mais il ne fait que des hypothèses sur tous les effets que celui-ci produit sur le récepteur. Inversement, le récepteur est sûr des effets qu'il a subis par la réception du message, et il infère des croyances sur les causes de son exécution. Si les spécifications locales aux agents deviennent publiques, alors les deux agents peuvent faire des inférences correctes, sinon ce sont des croyances qui doivent être révisées.

Singh [Singh, 1998] remarque que la perspective publique est plus correcte, car elle va dans le sens dans lequel on traite le discours humain, c.a.d. où les participants sont traités comme égaux. De plus, il critique le modèle cognitif comme étant insuffisant pour vérifier la conformité d'un acte à sa sémantique standard. Par exemple, si l'agent Alice exécute un $inform$ avec le contenu « *Il pleut* », on se pose la question de savoir comment on vérifie si Alice croit qu'il pleut. Comme il n'y a aucun moyen de le faire les langages comme FIPA-ACL imposent des contraintes qui réduisent l'autonomie des agents, comme le fait de dire toujours ce qu'on croit, de croire les autres et de faire toujours ce que les autres demandent. Par conséquent, Singh va plus loin et suggère d'utiliser un modèle social de la communication qui considère l'acte communicatif comme faisant partie intégrante d'une interaction sociale plus générale comme on l'a présenté dans le paragraphe 5.3.3. Dans le modèle social, on ne s'intéresse pas seulement à l'état cognitif de l'agent mais aussi à la façon dont il se comporte, par exemple s'il est conforme aux règles de communication. Ainsi, les actes de langage sont satisfaits quand pour :

- les assertifs - le monde est identique à ce qu'ils décrivent ;
- les directifs - le récepteur du message agit pour assurer la réalisation de leur contenu ;
- les commissifs - celui qui envoie les messages agit pour assurer la réalisation de leur contenu ;
- etc.

41. En effet, les triggers sont une combinaison des croyances et désirs qui produisent les intentions pour réaliser l'acte communicatif. Les tropismes sont les effets sur les croyances et les désirs produits par l'exécution de l'acte.

9.3.3 Communication dans un contexte normatif

Dans le chapitre 5, nous avons montré que la communication joue un rôle important dans la structuration des interactions entre les agents de type normatif, notamment par le fait que les agents peuvent créer des états déontiques par la simple production de certains actes communicatifs. Dans la suite, nous montrons que le modèle temporel proposé dans le chapitre 8 se prête très bien à la déscription de l'activité communicationnelle dans un contexte normatif.

Notons que nous faisons une différence claire entre les propriétés déontiques comme l'obligation, l'interdiction ou la permission d'exécuter une action et les actions qui obligent, autorisent et interdisent[42]. Dans l'approche formelle que nous avons présentée dans le chapitre 8, nous utilisons pour représenter les états déontiques les prédicats[43] : $O(a,b,\alpha,i)$, etc. Pour représenter les actions on va utiliser les symboles $oblige(a,b,\alpha,i)$ avec la signification que l'agent a oblige l'agent b d'effectuer l'action α dans l'intervalle de temps i. Avec cette notation, on peut décrire des situations telles que :

$holds(i,O(Alice,oblige(Bob,inform(Bob,Charles,\text{“}il\ pleut''')),j))$

qui décrit qu'il y a une obligation pour Alice, et qui est valable sur l'intervalle i, d'obliger Bob de dire à Charles, dans l'intervalle j, qu'il pleut.

Dans un contexte simple, sans structure organisationnelle ou relations de pouvoir et autorité, on peut considérer que les effets produits par l'exécution de ces actions sont les états déontiques correspondants. On remarque que cette interprétation est valable pour tous les agents participant et elle a donc un caractère public. Formellement, cela peut s'exprimer par les axiomes suivants :

$$does(a,oblige(b,\alpha,i),j) \Rightarrow holds([max(j), \rightarrow[,O(b,\alpha,i))$$
$$does(a,authorize(a,b,\alpha,i),j) \Rightarrow holds([max(j), \rightarrow[,P(b,\alpha,i))$$
$$does(a,forbid(a,b,\alpha,i),j) \Rightarrow holds([max(j), \rightarrow[,I(b,\alpha,i))$$

Pourtant, en réalité, les choses ne sont pas aussi simples que cela. Le résultat d'un acte dépend du contexte dans lequel il s'est produit, par exemple de la position de l'agent dans l'organisation (si le SAN a une structure organisationnelle hiérarchique), de ses capacités, etc. Une autre remarque concerne le fait que les états déontiques peuvent être générés non seulement par les actes ci-dessus mentionnés, mais aussi par l'intermédiaire d'autres actes de communication comme les engagements ou les déclaratifs [Traum et Allen, 1994; Dignum et Weigand, 1995]. Par exemple, Dignum et Weigand [Dignum et Weigand, 1995] utilisent seulement les directifs et les commissifs[44] pour générer les obligations, et les

42. Ces actions ne sont pas forcément des actes de langage. On peut par exemple bloquer l'accès à une ressource sans informer les agents de l'interdiction qui s'est ainsi créée.

43. Dans le chapitre 8, pour simplifier la présentation, la signature de ces prédicats ne contenaie que trois arguments. Ici, on l'adapte pour répondre à d'autres besoins. En effet, une norme peut contenir plus d'informations (voir le chapitre 5) et cela est valable aussi pour la description d'états déontiques qu'elle génère.

44. Représentés dans notre formalisme par les termes $dir(a,b,\alpha,i)$, respectivement $commit(a,b,\alpha,i)$.

déclaratifs pour les permissions et les interdictions. Dans le cas d'un contexte institutionnalisé régi par des relations de *pouvoir* de type $sup(a,b)$ une obligation est générée si un agent supérieur a demande à son subalterne b l'exécution d'une action :

$$does(a,dir(a,b,\alpha,i),j) \wedge holds(j,sup(a,b)) \Rightarrow \\ holds([max(j), \rightarrow[,O(a,b,\alpha,i))$$

Dans un contexte basé sur des relations d'*autorité*, l'obligation est créée si l'agent qui demande l'exécution de l'action a la permission de la demander :

$$does(a,dir(a,b,\alpha,i),j) \wedge holds(j,P(a,dir(a,b,\alpha,i),j)) \Rightarrow \\ holds([max(j), \rightarrow[,O(a,b,\alpha,i))$$

Dans le cas d'un contexte où les relations sont de type *charitable*, la demande d'un agent génère automatiquement une obligation :

$$does(a,dir(a,b,\alpha,i),j) \wedge holds(j,charity(b)) \Rightarrow \\ holds([max(j), \rightarrow[,O(a,b,\alpha,i))$$

Un dernier cas étudié est celui où il s'agit d'un engagement $commit(a,b,\alpha,i)$ de l'agent a pour l'agent b d'exécuter α dans l'intervalle i :

$$does(a,commit(a,b,\alpha,i),j) \Rightarrow holds([max(j), \rightarrow[,O(a,b,\alpha,i))$$

9.4 Obligations répétitives

Une autre extension possible du modèle serait de pouvoir décrire et de raisonner sur des obligations répétitives, par exemple, du type : « l'obligation de voir deux filmes tous les dimanches ». La notion de répétitivité a été déjà étudiée en IA et elle s'applique en général aux événements [Ligozat, 1991; Morris *et al.*, 1995; Terenziani, 2000]. L'approche la plus connue consiste à voir un événement répétitif comme étant une collection d'intervalles, dotée d'une structure temporelle. La structure temporelle établit le cardinal et l'ordre d'apparition de l'événement. Les caractéristiques d'un événement répétitif sont ainsi données en termes d'ensemble des propriétés telles que : le nombre d'apparitions, la distance entre deux apparitions successives, l'intervalle dans lequel toutes ces apparitions doivent se produire.

Le raisonnement avec événements répétitifs est plus complexe que le raisonnement avec événements singuliers. En effet, dans le cas des événements répétitifs on doit représenter explicitement l'incomplétude dans l'information sur le nombre d'apparitions d'un événement et sur la *période* d'apparition. La période peut être exprimée de plusieurs façons. Le plus fréquemment on l'attribue à d'événements du type calendrier, par exemple, les dimanches.

L'utilisation des obligations sur les événements répétitifs a la même applicabilité que celle des obligations sur les événements singuliers :

– vérifier la consistance entre les apparitions d'un événement et sa description normative ;
– planifier l'exécution des actions satisfaisant les contraintes normatives.

La description d'une obligation répétitive est en effet une obligation sur un événement répétitif. Elle doit contenir des informations sur :

– l'intervalle dans lequel toute les apparitions de l'événement doivent se produire ;
– la période ;
– le nombre d'apparitions de l'événement dans une période.

Rappelons que les obligations sur les événements singuliers sont représentées par les fluents et que la durée de vie d'une telle obligation est donnée par son intervalle de validité. Nous optons aussi pour la représentation des obligations répétitives par des fluents dont la durée de vie exprime l'intervalle d'apparition de toutes les instances de l'événement répétitif. Ainsi, nous proposons la syntaxe suivante pour les obligations répétitives :

$$holds(i,O^*(agent,\alpha,n,\pi))$$

où i est la durée de vie ou l'intervalle d'apparition de tous les événements de type α et n représente le nombre d'apparitions dans la période π.

Si l'obligation ne suppose pas de période, c.a.d. qu'on veut exécuter une action plusieurs fois dans un intervalle de référence (singulier), la période π peut être remplacée par cet intervalle.

A titre d'exemple, l'obligation de Jean de voir deux films tous les dimanches dans l'intervalle $[15/05/2002 - 15/09/2002]$ s'écrit

$$holds([15/05/2002 - 15/09/2002],O^*(Jean,aller\_cinema,2,Dimanches^*)) \tag{9.1}$$

et l'obligation de voir deux filmes le 19/05/2002, valide dans le même intervalle, s'écrit

$$holds([15/05/2002 - 15/09/2002],O^*(Jean,aller\_cinema,2,[19/05/2002,19/05/2002])) \tag{9.2}$$

Afin de pouvoir exprimer les périodes, nous considérons la notation introduite dans [Leban *et al.*, 1986] qui propose un langage pour décrire les périodes définies par l'utilisateur. Dans la suite, nous présentons brièvement cette notation. Leban se propose de construire un calendrier à partir d'un ensemble de périodes de base. Il utilise ainsi un moment de référence MR (par exemple, considérons qu'un utilisateur veut choisir 01/01/2000) et deux opérateurs (*:during :* et *n/C*) pour construire des nouvelles collections d'intervalles (par exemple, *Semaines**, *Mois**, *Années**, *Dimanches**, *Janvier**, etc.) à partir des collections de base (e.g. *Jours**). Pour illustrer, on peut définir les semaines, les mois et les années de la manière suivante :

(d1) *Semaines** $\equiv$ $<$2 ; *Jours** ; 7$>$

(d2) *Mois** $\equiv$ $<$0 ; *Jours** ; 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31$>$

(d3) *Années** $\equiv$ $<$0 ; *Mois** ; 12$>$.

Le symbole $\equiv$ fait le lien entre le nom d'une collection et sa définition. Le premier terme de la définition représente la distance entre le MR et le commencement de la collection d'intervalles. Comme le MR, c.a.d. le 01/01/2000, tombe le samedi, la première semaine complète commence deux jours après. Le deuxième terme décrit la collection utilisée pour construire la nouvelle collection d'intervalles. Le dernier terme représente le nombre d'éléments qui doivent être regroupés pour former un élément de la nouvelle collection. Pour illustrer cela, essayons de voir le résultat obtenu par la définition des *Années** dans l'intervalle $[01/01/2001,31/05/2003]$. En considérant que MR = 0 et qu'une date D est représentée comme étant le nombre de jours entre MR et D (par exemple, le 01/01/2001 est représenté par 365 ; pour des raisons de simplicité nous ne traitons pas les années bissextiles), alors les *Années** vont générer la collection suivante d'intervalles :

(c1) {[365,729], [730, 1095], [1096, 1246]}

Notons que les intervalles dans une collection sont représentés par leurs bornes inférieures, respectivement, supérieures.

L'opérateur *:during :* permet la génération de nouvelles collections à partir des intervalles ou des collections. Plus précisément, il divise un intervalle ou une collection en parties conformément à une autre collection. Par exemple :

(d4) *Mois** :during : *Années**

démembre la collection des années (c1) dans des mois, permettant d'obtenir la collection (de profondeur 2) suivante :

(c2) {{[365,395], [396,423], [424,454], [455,484], [485,515], [516,545], [546,576], [577,607], [608,637], [638,668], [669,698], [699,729]}, {[730,760], [761,789], [790,820], [821,850], [851,881], [882,911], [912,942], [943,973], [974,1003], [1004,1034], [1035,1064], [1065,1095]}, {[1096,1126], [1127,1154], [1155,1185], [1186,1215], [1216,1246]}}

L'opérateur n/C s'applique à toute collection C, sélectionnant le n-ième intervalle de chaque collection C de profondeur 1. Par exemple, l'application de l'opérateur $2/$ à (d4) va produire la collection formée par tous les deuxièmes mois des années :

(c3) {[396,423], [761,789], [1127,1154]}

On peut combiner les deux opérateurs pour obtenir de façon incrémentale de nouvelles collections, par exemple, tous les dimanches, ou le premier dimanche des mois de février :

(d5) *Dimanche** $\equiv 7/$*Jours** :during : *Semaines**

(d6) *Fevrier** $\equiv 2/$*Mois** :during : *Années**

(d7) *Premier-Dimanche-de-Fevrier** $\equiv 1/$*Dimanche** :during : *Fevrier**

En ce qui concerne le raisonnement avec des obligations répétitives, le type de représentation choisi permet d'une part une traduction facile en termes d'obligations pour des événements singuliers et d'autre part d'utiliser après le modèle temporel et l'algorithme d'ordonnancement déontique. La traduction consiste dans le fait que :

– l'ensemble composé par la période et la durée de vie est utilisé pour démembrer l'obligation répétitive dans des obligations répétitives sur des intervalles de référence absolus. Les intervalles de référence sont calculés avec la méthode générative proposée dans [Leban *et al.*, 1986]. Dans l'exemple ci-dessus, l'obligation d'aller au cinéma 2 fois tous les dimanches dans l'intervalle [15/05/2002,15/09/2002], est remplacée par l'ensemble :

$$holds([15/05/2002,15/09/2002],O^*(Jean,aller\_cinema,2,[19/05/2002,19/05/2002])) \wedge$$
$$holds([15/05/2002,15/09/2002],O^*(Jean,aller\_cinema,2,[26/05/2002,26/05/2002])) \wedge$$
$$\ldots$$
$$holds([15/05/2002,15/09/2002],O^*(Jean,aller\_cinema,2,[15/09/2002,15/09/2002]))$$

– le nombre n d'apparitions de l'événement dans une période est conservé dans les nouvelles obligations et il peut être utilisé pour générer les contraintes de précédence entre les événements produits (ou à réaliser) dans le même intervalle de référence (par exemple, $i_1 \prec i_2$ où i_1 et i_2 sont les intervalles d'apparitions de deux instances du même type d'événement).

9.5 Conclusions

Dans ce chapitre nous avons montré l'applicabilité du modèle temporel introduit dans le chapitre précédent pour produire des outils qui seront utilisés pour construire divers composants d'un SAN. Par exemple l'implémentation du modèle par l'intermédiaire de l'interpréteur pour $holds$ peut être utilisée par les agents et les éléments de contrôle pour détecter les déviations de la norme dans leur comportement ou dans le comportement des autres. D'ailleurs, l'interpréteur est construit de façon générale et donc il peut être utilisé pour vérifier la validité de toute propriété qualifiée temporellement.

L'algorithme d'ordonnancement déontique peut être intégré dans l'architecture d'un agent ordinaire, par exemple de type BDI, dans la partie délibération (voir les chapitres 2 et 4). Conçu sur des principes de programmation par contraintes, cet algorithme peut être étendu pour intégrer d'autres types de contraintes pour d'autres ressources.

Le choix que nous avons fait pour l'implémentation dans un langage de programmation logique a une double motivation. D'abord, l'aspect déclaratif d'un programme Prolog continue à séduire, car il y a une correspondance immédiate entre les prédicats (avec leurs propriétés) définis dans le modèle formel et leur expression Prolog. En d'autres termes, l'implémentation est évidente (même si ce n'est pas une chose simple). Notons au passage que les travaux que nous avons présentés dans cette thèse s'inscrivent dans un effort plus général réalisé par l'équipe Logique du GREYC, visant notamment à décrire l'interaction entre agents logiques (qui utilise la logique pour décrire leur comportement et même pour communiquer) [Beyssade *et al.*, 1995; Lefèvre, 1995]. Par exemple, la plate-forme Prolix [Clérin, 1999] qui consiste à faire communiquer plusieurs interpréteurs Eclipse Prolog offre des outils pour expérimenter la conception des systèmes d'agents à base de logique. D'ature part, si nous avons choisi Prolog, c'est aussi parce que les bibliothèques proposées par Eclipse Prolog contiennent des outils puissants pour la propagation des contraintes qui facilite à son tour l'écriture des algorithmes que nous avons proposés. Pour l'instant, nous avons traité seulement des exemples simples sur des architectures comprenant très peu d'agents. Par conséquent, la montée en charge des systèmes que nous avons proposés reste un problème à étudier et à tester.

En ce qui concerne la communication dans un contexte normatif, un domaine possible d'expérimentation est celui du dialogue à base de conventions. En fait, les conventions ont une forte connotation normative et donc on peut facilement les traiter comme des normes. Comme nous l'avons précisé dans le paragraphe 5.3.3, on veut étudier le remplacement éventuel des protocoles traditionnels par des conventions normatives, étant donné que nous disposons d'une infrastructure qui nous permet de traiter à la fois des aspects liés aux normes et à la communication.

Chapitre 10

Conclusions

10.1 Introduction

L'objectif principal de cette de thèse est de montrer la façon dont on utilise les normes dans les SMA. Un autre objectif est d'expliquer la motivation de l'utilisation des normes dans l'interaction inter-agents et de proposer les outils nécessaires à sa réalisation. Nous avons ainsi décrit dans la première partie les Systèmes d'Agents Normatifs, qui sont des structures architecturales reflétant le paradigme social de la conception multi-agents intégrant le concept de norme. Dans la deuxième partie de cette thèse, nous avons décrit un modèle qui utilise le temps comme élément unificateur pour décrire l'interaction à base de normes. Nous avons mis en évidence son aspect opérationnel qui permet l'implémentation facile des outils nécessaires à la construction des composants des SAN.

Dans ce chapitre, nous discutons des principaux résultats qui ont été présentés dans cette thèse et nous mettons en évidence nos contributions.

10.2 Choix et contributions

On peut classer les contributions de notre thèse sur trois plans : conceptuel, architectural et formel. Une première contribution sur le plan conceptuel est la mise en évidence de l'antagonisme existant entre le concept d'autonomie et celui de contrôle et de proposer l'utilisation des normes comme une solution possible de compromis pour accepter ce paradoxe dans les SMA. Ainsi, les principaux messages de cette thèse à ce sujet sont :

1. Dans le cas des agents autonomes, on doit accepter que :
   (a) le comportement des agents peut être différent de ce que nous voulons obtenir d'eux. Le modèle classique du contrôle de l'exécution d'un programme ne s'applique plus.

   (b) le problème du contrôle n'est pas une question binaire à la « to be or not to be ». Le contrôle se manifeste dans un registre plus large, entre des agents totalement libres et des agents enrégimentés.

2. Il existe deux solutions pour traiter l'antagonisme entre le concept d'autonomie et celui de contrôle. Elles sont basées sur l'idée de séparer la partie qu'on peut contrôler de la partie autonome :
   (a) On travaille seulement au niveau architectural de l'agent. La solution consiste à séparer le comportement en deux parties : une partie autonome et une partie enrégimentée par programmation.
   (b) On laisse les agents autonomes mais on s'assure de la présence des structures d'interaction (physiques, logicielles) qui influencent leur comportement pour qu'il soit correct. C'est ici qu'intervient la notion de norme.

   La première solution est moins acceptable, car le fait d'intervenir dans la construction d'un agent ne correspond pas à l'idée d'hétérogénéité. L'agent n'est plus une boîte noire. La partie enrégimentée est, dans ce cas, prioritaire, car elle doit garantir que le comportement est correct. De plus, elle est fixée pendant l'exécution d'un agent. Par conséquent, notre option est d'utiliser la deuxième solution, c.a.d. les normes.

3. La norme est une expression syntaxique. Elle représente l'autorité. L'autorité est légitime seulement par la présence des structures d'interaction. Dans ce contexte, la norme peut être vue sous deux aspects :
   (a) informatif - la norme indique, par l'intermédiaire d'une autorité, le comportement désirable. Cela se fait dans le sens de suggérer, de prescrire (comme un médecin à son patient) ou d'influencer le comportement afin d'obtenir celui qu'on souhaite.
   (b) coercitif - la norme n'informe pas seulement sur le comportement désirable, mais aussi sur les conséquences dans le cas où l'agent ne se conforme pas. Par exemple, l'interdiction d'utiliser l'imprimante peut être accompagnée de pénalités (amende, élimination du système, etc.) ou de l'impossibilité matérielle d'y accéder (on la met dans un coffre fort).

   La norme en tant que source d'information peut être utilisée pendant l'exécution du système pour spécifier quel est le comportement désirable. Cela permet une grande flexibilité dans le contrôle du système.

Une autre contribution à la fois sur plan conceptuel et architectural est la présentation de l'utilisation des normes dans le cadre du paradigme social de la conception des SMA. Nous avons présenté, d'une part, un état de l'art sur les aspects sociaux qui semblent importants pour notre démarche et d'autre part nous sommes remontés à l'origine des concepts, à savoir dans des domaines très divers comme la philosophie, la philosophie du droit, les sciences sociales et l'informatique, pour comprendre le concept de norme. La variété des formes sous lesquelles une norme est utilisée nous a conduits à lui donner notre propre signification et utilisation. Les choix que nous avons faits au cours de cette démarche, ainsi que nos contributions concernant la manière d'intégrer la norme à un système s'appliquent à trois niveaux :

1. niveau système - A ce niveau, le choix concerne le type de structure idéale pour l'interaction. Nous considérons que l'utilisation des normes doit se faire dans le cadre du paradigme social. Ainsi les concepts qu'on utilise pour structurer l'interaction sont ceux d'organisation avec une structure hiérarchique, de rôle et de groupe. Le concept de dépendance sociale n'est pas explicite dans notre thèse. Par contre, il devient implicite lorsqu'on parle des normes. La norme est vue considérée un moyen d'exprimer la présence des autres par l'intermédiaire d'une autorité que l'agent ne peut pas ignorer. A cette structure sociale, on ajoute la structure normative, qui manipule les normes. Elle est représentée par trois types d'acteurs :
   (a) autorité (le concepteur des normes)
   (b) agents ordinaires (les destinataires des normes)
   (c) structures de contrôle (l'élément coercitif par l'intermédiaire duquel on assure le contrôle et la manifestation de l'autorité).
2. niveau agent - A ce niveau, il faut choisir le type d'agent idéal participant à l'interaction normative. Même si la normativité ne demande pas la présence explicite de la rationalité (le but final de la normativité est de savoir si on est conforme à la norme et non pas comment on y arrive), nous préférons considérer le cas des agents rationnels. On part du principe que la norme est avant tout une information venue de l'extérieur. La pertinence du modèle BDI concernant la façon dont cette information doit être traitée explique notre choix. Par contre, on doit montrer comment l'information normative s'intègre dans le processus de décision d'un agent. Pour l'instant nous avons fait référence à d'autres travaux comme [Dignum *et al.*, 2000; Broersen *et al.*, 2001].
3. niveau outils - Les outils que nous avons proposés dans la deuxième partie sont destinés faciliter l'utilisation effective des normes dans un SMA. Ainsi, ils peuvent être utilisés pour la :
   (a) description des normes. Les normes décrivent le comportement idéal d'un agent. Elles font intervenir les notions déontiques, telles que l'obligation, l'interdiction et la permission, les actions, le temps et l'agence. Le langage retenu pour exprimer les normes est celui de la logique.
   (b) application au raisonnement normatif :
      i. observation du comportement,
      ii. détection des violations,
      iii. planification et ordonnancement déontique.
   (c) communication des normes et description des normes de communication.

La construction des outils que nous avons détaillés plus haut est basée sur l'aspect opérationnel du même modèle formel. Cette thèse a notamment contribué à la représentation dans le même langage logique de l'information normative décrivant le comportement idéal et de l'exécution du système dans le temps correspondant au comportement effectif. L'objectif est de permettre à tous les composants des SAN de savoir si un agent est en légalité ou non. Ainsi, nous avons choisi de prendre en compte des normes dynamiques, dotées d'une certaine durée de vie et réglementant l'exécution d'une action dans le temps. Le modèle permet l'observabilité du système, c.a.d. de préciser à un moment donné si on est en train d'exécuter une action ou si elle a été terminée, ou de savoir si une propriété est valide ou non dans un intervalle de temps donné.

L'introduction des concepts déontiques dans la planification est une autre contribution de cette thèse. La solution consiste à transformer les contraintes déontiques en contraintes temporelles et de les intégrer à un module d'ordonnancement sous la forme d'un problème de TCSP. Même si l'algorithme est présenté sous une forme non-optimale, l'avantage de cette transformation est qu'on peut récupérer des résultats performants obtenus dans le domaine TCSP [Schwalb et Dechter, 1997] et les appliquer à l'ordonnancement déontique.

Un autre principe qui se dégage sur le plan formel est celui de l'économie logique. Nous avons opté pour le modèle qui simplifie le plus la description du monde et du comportement des agents. Ainsi, dans le cas des agents ordinaires, l'introduction des concepts déontiques dans les normes n'implique pas forcément l'utilisation de la logique déontique (modale) pour aider le raisonnement normatif. C'est l'une des raisons qui nous permet d'introduire les obligations sous la forme de prédicats non-interprétés. Par contre, la logique déontique est essentielle pour les concepteurs de normes qui doivent vérifier leur cohérence (par exemple, pour s'assurer qu'il n'y a pas d'action obligatoire et interdite en même temps). Un autre exemple d'économie logique concerne le remplacement des fluents par des opérateurs temporels modaux pour qualifier temporellement les propriétés représentables par des formules complexes. Un modèle de premier ordre suffit pour l'usage des fluents, quant à l'introduction des formules complexes, la logique temporale métrique (modale) est nécessaire. On remarque que le principe d'économie logique vient aussi du fait que les modèles de premier ordre sont plus faciles à implémenter (par exemple en Prolog, un langage de programmation basé sur la logique de prédicats de premier ordre) que les modèles formalisés dans un langage de logique modale (pour lequel on dispose de moins d'outils, pour l'instant).

10.3 Limites et perspectives

Une première perspective sera d'approfondir au niveau conceptuel les divers points qu'on n'a pas bien su expliquer dans cette thèse. Par exemple, en ce qui concerne l'utilisation des normes, nous n'avons pas traité le probléme de determiner comment on s'en sert pour maintenir ou faire amener un système à un état d'équilibre. Autrement dit, quels sont les outils dont on peut s'en servir pour définir et mesurer l'état d'équilibre, afin de donner une indication sur le type de contrainte normative qu'on doit utiliser. Pour l'instant, on sait que certains domaines comme la physique et la théorie des systèmes étudient des problèmes similaires liés à la dynamique des systèmes mais sans pouvoir pour le moment envisager une piste concrète de recherche.

Ensuite, dans le même registre, une autre piste de recherche envisageable est de mieux préciser comment les normes peuvent être utilisées pour remplacer les protocoles de communication afin de permettre une plus grande flexibilité dans le dialogue.

D'un point de vue applicatif, une autre remarque concerne la validation concrète de ce modèle dans une application réelle. La majorité des exemples sont imaginaires et simples, leur but étant d'expliquer les concepts. La même remarque est valable pour les outils qui sont conçus selon le principe de la nécessité immédiate sans se poser de questions sur la robustesse, la montée en charge du système, la convivialité du système. On se propose donc de trouver une application qui pourrait nous permettre de nous investir dans l'amélioration de ces points (et sans doute encore d'autres).

Étant donné que les événements jouent un rôle important dans l'observabilité du système, on se propose d'étudier la possibilité de construire des SAN sur des mécanismes événementiels performants. Le système actuel étant implémenté en Prolog, on doit également étudier la manière dont on se connecte à une base Prolog. La dernière version de PROLOG ECLIPSE 5.3 [eclipse, 2001] permet de réaliser une interface avec le langage Java, qui est plus adapté à l'implémentation de ce type de mécanisme. On va utiliser le langage Prolog pour ses capacités de représenter les connaissances, de raisonnement logique et de raisonnement par contraintes.

Bien d'autres choses peuvent être améliorées, car, en fin de compte, l'originalité de notre approche est sans doute d'avoir tenté, dans le cadre d'une thèse, de réunir plusieurs domaines : la conception des SMA, les logiques temporelle, de l'action et déontique, le raisonnement avec contraintes, l'ordonnancement et la planification, la programmation logique. C'est un travail complexe qui est basé sur la lecture de nombreux articles (dont nous ne citons qu'une partie), sur l'état de l'art, des livres ou des thèses de doctorats. Nous avons compris et assimilé certains concepts et théories nous. D'autres nous échappent encore. Le présent manuscrit décrit en effet le niveau actuel de notre compréhension des choses que nous espérons à son tour éclaircir.

Troisième partie

Bibliographie et annexes

Bibliographie

[Agre et Chapman, 1987] P. Agre et D. Chapman. Pengi : An implementation of a theory of activity. Dans *Proceedings of IJCAI'87*, pages 268–272, Morgan Kaufmann, 1987.

[Alchourrón, 1993] C. E. Alchourrón. Philosophical foundations of deontic logic and the logic of defeasible conditionals. Dans *Deontic Logic in Computer Science : Normative System Specification*, rédacteurs J. J. Ch. Meyer et R. J. Wieringa. John Wiley & Sons, 1993.

[Allen et Ferguson, 1994] J. F. Allen et G. Ferguson. Actions and events in interval temporal logic. Rapport technique, Computer Science Department, University of Rochester, 1994.

[Allen et Hayes, 1985] J. F. Allen et P. Hayes. A common-sense theory of time. Dans *Proceedings of IJCAI'85*, pages 528–531, 1985.

[Allen et Hayes, 1989] J. F. Allen et P. Hayes. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5 :225–238, 1989.

[Allen, 1984] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23 :123–154, 1984.

[Auffray et Enjalbert, 1989] Y. Auffray et P. Enjalbert. Modal theorem proving : an equational viewpoint. Dans *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, volume 1, Detroit, 1989.

[Barbuceanu *et al.*, 1998] M. Barbuceanu, T. Gray et S. Mankovski. Coordinating with obligations. Dans *Proceedings of Autonomous Agents'98*, Minneapolis, MI, 1998.

[Barbuceanu, 1998] M. Barbuceanu. Agents that work in harmony by knowing and fulfiling their obligations. Dans *Proceedings of AAAI'98*, 1998.

[Baumgartner et Stolzenburg, 1995] P. Baumgartner et F. Stolzenburg. Constraint model elimination and a PTTP implementation. Dans *Proceedings of the 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, volume LNAI 918, pages 201–216. Springer-Verlag, 1995.

[Becher *et al.*, 2000] G. Becher, F. Clérin-Debart et P. Enjalbert. A qualitative model for time granularity. *Computational Intelligence*, 16(2) :137–168, 2000.

[Bell et LaPadula, 1975] D. E. Bell et L. J. LaPadula. Secure computer systems : Unified exposition and multics interpretation. Rapport Technique ESD-TR-73-306, The MITRE Corporation, 1975.

[Beyssade *et al.*, 1995] C. Beyssade, P. Enjalbert et C. Léfevre. Communicating logical agents : Model and applications. Dans *Proceedings of ATAL'95*, 1995.

[Biddle, 1979] B. J. Biddle. *Role Theory : Expectations, Identities and Behaviors*. Academic Press, 1979.

[Blazewicz *et al.*, 1996] J. Blazewicz, W. Domshcke et E. Pesch. The job shop scheduling problem : Conventional and new solution techniques. *European Journal of Operation Research*, 93(1) :1–33, 1996.

[Brandano, 2001] S. Brandano. The event calculus assessed. Dans *Proceedings of TIME'01*, Italy, 2001.

[Bratman, 1987] M. E. Bratman. *Intentions, plans, and practical reason*. Harvard University Press, 1987.

[Bürckert, 1991] H. J. Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*. LNAI 568. Springer-Verlag, 1991.

[Bretier et Sadek, 1996] P. Bretier et M. D. Sadek. A rational agent as a kernel of a cooperative dialogue system : Implementing a logical theory of interaction. Dans *Proceedings of ATAL'96*, 1996.

[Broersen *et al.*, 2001] J. Broersen, M. Dastani, Z. Huang, J. Hulstijn et L. van der Torre. The BOID Architecture. Dans *Proceedings of Agents'2001*, USA, 2001.

[Brooks, 1986] R. Brooks. Achieving artificial intelligence through building robots. Rapport Technique AI Memo 899, MIT, 1986.

[Brzoska, 1998] C. Brzoska. Programming in metric temporal logic. *Theoretical Computer Science*, 1-2(202), 1998.

[Castelfranchi *et al.*, 1999] C. Castelfranchi, F. Dignum, C. Jonker et J. Treur. Deliberative normative agents : Principles and architectures. Dans *Proceedings of ATAL'99*, Orlando, 1999. AAAI Press.

[Castelfranchi et Falcone, 1997] C. Castelfranchi et R. Falcone. Delegation conflicts. Dans *Multi-Agent Rationality*, rédacteurs M. Boman et W. Van de Velde. LNAI-1237, Springer Verlag, Berlin, 1997.

[Castelfranchi et Falcone, 1998] C. Castelfranchi et R. Falcone. Principles of trust for MAS : Cognitive anatomy, social importance, and quantification. Dans *Proceedings of ICMAS'98*, Paris, 1998.

[Castelfranchi, 1995] C. Castelfranchi. Commitements : From individual intentions to groups and organizations. Dans *Proceedings of ICMAS'95*. AAAI Press, 1995.

[Castelfranchi, 1998] C. Castelfranchi. Modeling social action for AI agents. *Artificial Intelligence*, 103 :157–182, 1998.

[Cavedon et Sonenberg, 1998] L. Cavedon et L. Sonenberg. On social commitment, roles and preferred goals. Dans *Proceedings of ICMAS'98*, Paris, France, 1998.

[Cervesato *et al.*, 2000] I. Cervesato, M. Franceschet et A. Montanari. A guided tour through some extentions of the event calculus. *Computational Intelligence*, 16(2) :307–347, 2000.

[Chaib-draa *et al.*, 1992] B. Chaib-draa, B. Moulin, R. Mandiau et P. Millot. Trends in distributed artificial intelligence. *Artificial Intelligence Review*, 6 :35–66, 1992.

[Chisholm, 1963] R. Chisholm. Contrary-to-duty imperatives and deontic logic. *Analysis*, 24 :33–36, 1963.

[Clark, 1978] K. Clark. *Negation as Failure*. Logic and Databases. Plenum Press, New York, 1978.

[Clérin, 1999] F. Clérin. Une maquette de SMA pour des agents logiques réalisée en Prolog. Rapport technique, Université de Caen, 1999.

[Cockburn et Jennings, 1996] D. Cockburn et N. Jennings. Archon - a distributed artificial intelligence system for industrial applications. Dans *Foundations of Distributed Artificial Intelligence*, rédacteurs G. O'Hare et N. Jennings, pages 319–344. Willey & Sons, 1996.

[Cohen et Levesque, 1990] P. R. Cohen et H. J. Levesque. Intetion is choice with commitment. *Artificial Intelligence*, 42(2-3) :213–262, 1990.

[Conte *et al.*, 1999] R. Conte, C. Castelfranchi et F. Dignum. Autonomous norm-acceptance. Dans *Intelligent Agents V*, rédacteurs J. P. Mueller, M. P. Singh et A. S. Rao. LNAI-1555, Springer Verlag, Berlin, 1999.

[Conte et Castelfranchi, 1995] R. Conte et C. Castelfranchi. *Cognitive and social action*. UCL Press, London, 1995.

[Cuppens et Saurel, 1996] F. Cuppens et C. Saurel. Specifying a security policy : A case study. Dans *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, Kenmare, Ireland, 1996.

[Davidson, 1967] D. Davidson. The logical form of action sentences. Dans *The Logic of Decision and Action*, rédacteur N. Rescher. University of Pitsburgh Press, Reprinted as [Davidson, 1980], 1967.

[Davidson, 1980] D. Davidson. The logical form of action sentences. Dans *Essays on Actions and Events*. Clarendon Press, Oxford, 1980.

[Debart *et al.*, 1992] F. Debart, P. Enjalbert et M. Lescot. Multimodal logic programming using equational and order-sorted logic. *Theoretical Computer Science*, 105 :141–166, 1992.

[Demazeau et Müller, 1990] rédacteurs Y. Demazeau et J.-P. Müller. *Decentralized Artificial Intelligence (1)*. Elsevier Science Publisher, 1990.

[Dennett, 1978] D. C. Dennett. *Mindstorms*. Bradford Books, Vermont, 1978.

[Dignum *et al.*, 2000] F. Dignum, D. Morley, L. Sonenberg et L. Cavedon. Towards socially sophisticated BDI agents. Dans *Proceedings of ICMAS'2000*, Boston, USA, 2000.

[Dignum et Weigand, 1995] F. Dignum et H. Weigand. Communication and deontic logic. Dans *Information Systems, Correctness and Reusability*, pages 242–260. World Scientific, Singapore, 1995.

[d'Iverno *et al.*, 1997] M. d'Iverno, D. Kinny, M. Luck et M. Wooldridge. A formal specification of dMARS. Dans *Intelligent Agents IV : Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL'97)*, volume LNAI 1365. Springer-Verlag, 1997.

[Drogoul, 1993] A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisations dans les systèmes multi-agents*. Thèse de l'université de Paris 6, Laforia, 1993.

[Durfee *et al.*, 1987] E. H. Durfee, V. R. Lesser et D. D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11) :1275–1291, 1987.

[Durfee, 1998] E. Durfee. Distributed problem solving and planning. Dans *A Modern Approach to Distributed Artificial Intelligence*, rédacteurs G. Weiß et S. Sen, pages 121–164. AAAI/MIT Press, 1998.

[eclipse, 2001] http ://www-icparc.doc.ic.ac.uk/eclipse/, 2001.

[Ferber et Drogoul, 1992] J. Ferber et A. Drogoul. Using reactive multi-agent systems in simulation and problem solving. Dans *DAI : Theory and Praxis*, rédacteurs N. M. Avouris et L. Gasser. Kluwer Academic Publishers, London, 1992.

[Ferber et Gutknecht, 1998] J. Ferber et O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. Dans *Proceedings of ICMAS'98*, Paris, 1998.

[Ferber, 1995] J. Ferber. *Les systèmes multi-agents, vers une intelligence collective*. InterEditions, Paris, 1995.

[Ferguson, 1992] I. A. Ferguson. *TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents*. Thèse de doctorat, University of Cambridge, UK, 1992.

[Finin *et al.*, 1995] T. Finin, Y. Labrou et J. Mayfield. Kqml as an agent communication language. Dans *Software Agents*, rédacteur J. Bradshaw. MIT Press, 1995.

[FIPA, 2000] http ://www.fipa.org/, 2000.

[Fox, 1981] M. Fox. An organisational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11 :70–80, 1981.

[Galton, 1991] A. Galton. Reified temporal theories and how to unreify them. Dans *Proceedings of IJCAI'91*, pages 1177–1182, 1991.

[Gasser, 1991] L. Gasser. Social conceptions of knowledge and action : DAI foundations and open systems semantics. *Artificial Intelligence*, 47 :107–138, 1991.

[Gasser, 1992] L. Gasser. Dai approaches to coordination. Dans *DAI : Theory and Praxis*, rédacteurs N. M. Avouris et L. Gasser. Kluwer Academic Publishers, 1992.

[Gelfond *et al.*, 1991] M. Gelfond, V. Lifschitz et A. Rabinov. What are the limitations of the situation calculus ? Dans *Working Notes, AAAI Spring Symposium Series on the Logical Formalization of Commonsense Reasoning*, pages 59–69, 1991.

[Georgeff et Lansky, 1987] M. P. Georgeff et A. L. Lansky. Reactive reasoning and planning. Dans *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI'87)*, Seattle, WA, 1987.

[Georgeff et Rao, 1996] M. P. Georgeff et A. S. Rao. A profile of the Australian AI Institute. *IEEE Expert*, 11(6) :89–92, 1996.

[Haddadi et Sundermeyer, 1996] A. Haddadi et K. Sundermeyer. Belief-desire-intention agent architectures. Dans *Foundations of Distributed Artificial Intelligence*, rédacteurs G. O'Hare et N. Jennings, pages 169–186. Willey & Sons, 1996.

[Haddadi, 1996] A. Haddadi. *Communication and Cooperation in Agent Systems : A Pragmatic Theory*. Springer-Verlag, Heidelberg, 1996.

[Hansson, 1990] S. O. Hansson. Preference-based deontic logic (PDL). *Journal of Philosophical Logic*, 19 :75–93, 1990.

[Harel *et al.*, 1990] E. Harel, O. Lichtenstein et A. Pnueli. Explicit clock temporal logic. Dans *Proceedings of the 5th Symposium on Logic in Computer Science*, Philadelphia, IEEE Computer Society Press, 1990.

[Hewitt et de Jong, 1984] C. Hewitt et P. de Jong. Open systems. Dans *On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases, and Programming*, rédacteur M. Brodie et al., pages 147–164. Springer, 1984.

[Hewitt, 1991] C. Hewitt. Open information systems semantics for DAI. *Artificial Intelligence*, 47 :79–106, 1991.

[Hohfeld, 1913] W. N. Hohfeld. Fundamental legal conceptions as applied in judicial reasoning. *Yale Law Journal*, 1913.

[Horty et Belnap, 1995] J. Horty et N. Belnap. The deliberative stit : A study of action, omission, ability, and obligation. *Journal of Philosophical Logic*, 24 :583–644, 1995.

[ITSEC, 1992] ITSEC. *Critères d'évaluation de la sécurité des systèmes informatiques (v 1.2)*. Office des publications officielles des Communautés Européennes, Luxembourg, 1992.

[Jennings, 1996] N. Jennings. Co-ordination techniques for distributed artificial intelligence. Dans *Foundations of Distributed Artificial Intelligence*, rédacteurs G. O'Hare et N. Jennings, pages 187–210. Willey & Sons, 1996.

[Jones et Sergot, 1993] A. J. I. Jones et M. Sergot. On the characterisation of law and computer systems : The normative systems perspective. Dans *Deontic Logic in Computer Science : Normative System Specification*, rédacteurs J. J. Ch. Meyer et R. J. Wieringa. John Wiley & Sons, 1993.

[Kanger, 1971] S. Kanger. New foundations for ethical theory. Dans *Deontic Logic*, rédacteur R. Hilpinen, pages 36–58. D. Reidel Publishing Company, Dordrecht - Holland, 1971.

[Kowalski et Sergot, 1986] R. A. Kowalski et M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4 :67–95, 1986.

[Koymans *et al.*, 1983] R. Koymans, J. Vytopil et W. P. de Roever. Real-time programming and asyncronous message passing. Dans *Proceedings of the 2nd ACM Symposium on Principles of Distributed Computing*, Montreal, Canada, 1983.

[Koymans, 1989] R. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic*. Thèse de doctorat, Technical University of Eindhoven, 1989.

[Kripke, 1963] S. A. Kripke. Semantical considerations in modal logic. *Acta Philosophica Fennica*, 16 :83–94, 1963.

[Krogh, 1996a] C. Krogh. *Normative Structures in Natural and Artificial Systems*. Ph.D. Disertation, University of Oslo, Norway, 1996.

[Krogh, 1996b] C. Krogh. The rights of agents. *Intelligent Agents II*, LNCS :1–16, 1996.

[Labrou et Finin, 1998] Y. Labrou et T. Finin. Semantics and conversations for an agent communication language. Dans *Readings in Agents*, rédacteurs M. Huhns et M. P. Singh, pages 235–242. Morgan Kaufmann, 1998.

[Lampson, 1971] B. Lampson. Protection. Dans *5th Princeton Symposium on Information Sciences and Systems*, 1971.

[Larousse, 1995] Le petit larousse. Larousse, Paris, 1995.

[Leban *et al.*, 1986] B. Leban, D. D. McDonald et D. R. Forster. A representation for collections of temporal intervals. Dans *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 367–371, Philadelphia, 1986.

[Lefèvre, 1995] C. Lefèvre. *Agents Logiques Communicants*. Thèse de doctorat, Université de Caen, 1995.

[Levesque *et al.*, 1995] H. J. Levesque, F. Lin et R. Reiter. Defining complex actions in the situation calculus. Rapport technique, Departement of Computer Science, University of Toronto, 1995.

[Ligozat, 1991] G. Ligozat. On generalized interval calculi. Dans *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 234–240, 1991.

[Lin et Shoham, 1992] F. Lin et Y. Shoham. Concurrent actions in the situation calculus. Dans *Proceedings of AAAI'92*, pages 590–595, 1992.

[Lin, 1991] Y. Lin. Two theories of time. *Journal of Applied Non-Classical Logics*, 1(1) :37–63, 1991.

[Lindahl, 1977] L. Lindahl. *Position and Change*. D. Reidel Publishing Company, Dordrecht - Holland, 1977.

[Lupu et Sloman, 1997] E. C. Lupu et M. S. Sloman. Towards a role-base framework for distributed systems management. *Journal of Network and Systems Management*, 5(1) :5–30, 1997.

[Lupu, 1998] E. Lupu. *A Role-Based Framework for Distributed Systems Management*. Thèse de doctorat, Imperial College, London, UK, 1998.

[Makinson, 1986] D. Makinson. On the formal representation of rights relations. *Journal of Philosophical Logic*, 15 :403–425, 1986.

[Malone, 1987] T. Malone. Modelling co-ordination in organisations and markets. *Management Sciences*, 33(10) :1317–1332, 1987.

[Malville, 1999] E. Malville. *L'auto-organisation de groupes pour l'allocation de tâches dans les SMA : application à CORBA*. Thèse de Doctorat en Informatique, Université de Savoie, France, 1999.

[McCabe et Clark, 1995] F. G. McCabe et K. L. Clark. April - agent process interaction language. Dans *Intelligent Agents, ECAI 1994*, rédacteurs M. Wooldridge et N. Jennings, volume LNAI 890. Springer Verlag, 1995.

[McCarthy et Hayes, 1969] J. M. McCarthy et P. J. Hayes. Some philosophical problems from the standpoint of artificial intellingence. *Machine Intelligence*, 4 :463–502, 1969.

[McCarthy, 1979] J. McCarthy. Ascribing mental qualities to machines. Dans *Philosophical Perspectives in Artificial Intelligence*, rédacteur M. Ringle. Harvester Press, 1979.

[McDermott, 1982] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitivie Science*, 6 :101–155, 1982.

[Meyer, 1988] J. J. Ch. Meyer. A different approach to deontic logic : Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29 :109–136, 1988.

[Microsoft, 1999] Microsoft agent sdk. Microsoft Press, 1999.

[Minsky et Ungureanu, 1997] N. H. Minsky et V. Ungureanu. Regulated coordination in open distributed systems. Dans *Proceedings of the 2nd International Conference on Coordination Models and Languages*, Berlin, Germany, 1997.

[Müller, 1996] J. Müller. *An Architecture for Dynamically Interacting Agents*. Thèse de doctorat, DFKI, University of Saarlandes, Saarbrücken, 1996.

[Morris *et al.*, 1995] R. A. Morris, L. Khatib et G. Ligozat. Generating scenarios from specifications of repeating events. Dans *Proceedings of the 2nd International Workshop on Temporal Representation and Reasoning (TIME'95)*, pages 41–48, Melbourne, FL, 1995.

[Moses et Tennenholtz, 1995] Y. Moses et M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(6) :533–562, 1995.

[Nilsson, 1980] N. J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980.

[Object Management Group, 2001] Object Management Group. The common object request broker : Architecture and specification. Document September, 2001.

[Ohlbach, 1988] H. J. Ohlbach. A resolution calculus for modal logics. Dans *Proceedings of the 9th International Conference on Automated Deduction*, volume LNCS 310. Springer-Verlag, 1988.

[Panzarasa *et al.*, 1999] P. Panzarasa, T. Norman et N. Jennings. Modelling sociality in the bdi framework. *World Scientific*, 1999.

[Pinto, 1994] J. A. Pinto. *Temporal Reasoning in the Situation Calculus*. Thèse de doctorat, University of Toronto, Departement of Computer Science, Canada, 1994.

[Pitt et Mamdani, 2000] J. Pitt et A. Mamdani. Communication protocols in multi-agent systems : a development method and reference architecture. Dans *Issues in Agent Communication*, volume 1916 de *LNCS*. Springer, 2000.

[Pnueli, 1986] A. Pnueli. Specification and development of reactive systems. *Information Processing 86*, 1986.

[Pörn, 1977] I. Pörn. *Action Theory and Social Science : Some Formal Models*. Sythese Lybrary 120, D. Reidel, Dordrecht, 1977.

[Prakken et Sergot, 1996] H. Prakken et M. Sergot. Contrary-to-duty obligations. *Studia Logica*, 1996.

[Rao et Georgeff, 1995] A. S. Rao et M. P. Georgeff. BDI agents : From theory to practice. Dans *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*, San Francisco, 1995.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus : a simple solution (sometimes) and a completeness result for goal regression. Dans *Artificial Intelligence and Mathematical Theory of Computation : Papers in Honor of John McCarthy*, rédacteur V. Lifschitz, pages 359–380. Academic Press, 1991.

[Reiter, 2001] R. Reiter. *Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[Royakkers, 1994] L. M. M. Royakkers. Towards a deontic logic approach to legal rules. Dans *Proceedings of the Second International Workshop on Deontic Logic in Computer Science*, Amsterdam, The Netherlands, 1994.

[Russell et Norvig, 1995] S. Russell et P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice-Hall, 1995.

[Sadek, 1991] D. Sadek. *Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication*. Thèse de Doctorat en Informatique, Université de Rennes I, France, 1991.

[Sandhu *et al.*, 1996] R. S. Sandhu, E. J. Coyne, H. L. Feinstein et C. E. Youman. Role-based acces control models. *IEEE Computer*, 29(2) :38–47, 1996.

[Saury, 1991] R. Saury. Le secret médical. *Gestions Hospitalières*, 303 :120–125, 1991.

[Schwalb et Dechter, 1997] E. Schwalb et R. Dechter. Processing temporal constraint networks. *Artificial Intelligence*, 93 :29–61, 1997.

[Searle, 1969] J. R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, UK, 1969.

[Searle, 1983] J. R. Searle. *Intentionality : an essay of philosophy of mind*. Cambridge University Press, Cambridge, UK, 1983.

[Sergot, 1999a] M. Sergot. Normative positions. Dans *Proceedings of POLICY'99*, Bristol, UK, 1999.

[Sergot, 1999b] M. Sergot. Normative positions. Dans *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, rédacteurs Henry Prakken et Paul McNamara, pages 289–308. IOS Press, Amsterdam, 1999.

[Shanahan, 1997] M. P. Shanahan. *Solving the Frame Problem : a mathematical investigation of the common sense law of inertia*. MIT Press, Cambridge, Massachusetts, 1997.

[Shoham et Tennenholtz, 1995] Y. Shoham et M. Tennenholtz. On social laws for artificial agent societies : off-line design. *Artificial Intelligence*, 73 :231–252, 1995.

[Shoham, 1988] Y. Shoham. *Reasoning about change*. MIT Press, 1988.

[Shoham, 1993] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60 :51–92, 1993.

[Singh, 1991] M. P. Singh. Group ability and structure. Dans *Decentralized Artificial Intelligence (2)*, rédacteurs Y. Demazeau et J.-P. Müller. Elsevier/North-Holland, 1991.

[Singh, 1998] M. P. Singh. Agent communication languages : Rethinking the principles. *Computer*, December, 1998.

[Skarmeas, 1995] N. Skarmeas. Organisations through roles and agents. Dans *Proceedings of the International Workshop on the Design of Co-operative Systems*, pages 385–404, Nice, France, 1995.

[Smith, 1980] R. G. Smith. The contract net protocol : High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12) :1104–1113, 1980.

[Smolka et Ait-Kaci, 1989] G. Smolka et H. Ait-Kaci. Inheritance hierarchies : Semantics and unification. *JSC*, 7(3/4) :343–370, 1989.

[Steels, 1995] L. Steels. When are robots intelligent autonomous agents ? *Robotics and Autonomous Systems*, 15 :3–9, 1995.

[Stratulat *et al.*, 2001a] T. Stratulat, F. Clérin-Debart et P. Enjalbert. Normes, violations et temps. Dans *Modèles Formels de l'Interaction (MFI'01)*, Toulouse, France, 2001.

[Stratulat *et al.*, 2001b] T. Stratulat, F. Clérin-Debart et P. Enjalbert. Norms and time in agent-based systems. Dans *Proceedings of the 8th International Conference on Artificial Intelligence and Law (ICAIL'01)*, St. Louis, MO, 2001.

[Stratulat *et al.*, 2001c] T. Stratulat, F. Clérin-Debart et P. Enjalbert. Temporal reasoning : An application to normative systems. Dans *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME'01)*, Italy, 2001.

[Stratulat, 1997] T. Stratulat. Raisonnement automatique avec contraintes. Applications au raisonnement temporel. Rapport D.E.A. de l'Université de Caen, 1997.

[Terenziani, 2000] P. Terenziani. Integrated temporal reasoning with periodic events. *Computational Intelligence*, 16(2) :210–256, 2000.

[Thomason, 2000] R. Thomason. Desires and defaults. Dans *Proceedings of KR'2000*. Morgan Kaufmann, 2000.

[Traum et Allen, 1994] D. R. Traum et J. F. Allen. Discourse obligations in dialogue processing. Dans *Proceedings of ACL'94*, 1994.

[Tuomela, 1995] R. Tuomela. *The Importance of Us : A Philosophical Study of Basic Social Norms*. Standford University Press, 1995.

[Turing, 1950] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236) :433–460, 1950.

[van Benthem, 1991] J. van Benthem. *The Logic of Time*. Kluwer Academic, Dordrecht, 1991.

[van Eck, 1982] J. van Eck. A system of temporally relative modal and deontic predicate logic and its philosophical applications. *Logique et Analyse*, 99,100, 1982.

[Villa, 1994] L. Villa. A survey of temporal reasoning in artificial intelligence. *AI Communications*, March, 1994.

[von Wright, 1951] G. H. von Wright. Deontic logic. *Mind*, 60 :1–15, 1951.

[von Wright, 1993] G. H. von Wright. On the logic and ontology of norms. Dans *Philosophical Logic*, rédacteurs J. W. Davis, D. J. Hockney et W. K. Wilson, pages 89–107. D. Reidel Publishing Company, Dordrecht - Holland, 1993.

[Walther, 1988] C. Walther. Many-sorted unification. *JACM*, 35(1) :1–17, 1988.

[Wooldridge et Jennings, 1995] M. Wooldridge et N. Jennings. Agent theories, architectures and languages : A survey. Dans *Intelligent Agents, ECAI 1994*, rédacteurs M. Wooldridge et N. Jennings, volume LNAI 890, pages 1–32. Springer Verlag, 1995.