UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRESENTÉ À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

YANNICK EURIN

UN MODÈLE D'ACTION BASÉ SUR LA LOGIQUE DE DESCRIPTION POUR LA RECONNAISSANCE DE PLAN

5 janvier 2006

Bibliothèque Paul-Émile-Boulet

UQAC Université du Québec à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

RÉSUMÉ

Une des problématiques dans le domaine des systèmes multi-agents est la reconnaissance de plan. Celle-ci peut être vue comme le fait de recevoir une série d'actions, en entrée, puis d'inférer le but poursuivi par l'acteur de cette série. Dès lors, les résultats obtenus peuvent être employés à de multiples fins. Par exemple, nous pouvons prendre le cas des domaines de la médecine (pour assister des personnes en perte d'autonomie) ou de l'aviation (pour le pilote automatique).

L'objectif général de ce travail de recherche est d'élaborer un ensemble d'outils servant de base à la conception d'un modèle de reconnaissance de plan fondé sur la logique de description. Cette logique est une extension des réseaux sémantiques pour la représentation des connaissances. Elle est basée sur la relation de subsomption qui est une méthode d'inférences pour la classification de concepts à travers une ontologie. Le but étant de ramener la problématique de la reconnaissance de plan à un problème de classification. En somme, nous proposons un modèle de classification de plan d'action, servant de support au processus de la reconnaissance de plan.

La démarche consiste, en premier lieu à clarifier la notion de reconnaissance de plan, ainsi, que les éléments essentiels, la constituant. Puis, lors d'une deuxieme etape de redéfinir ses éléments à travers la logique de description.

Ce mémoire se veut une première phase d'un projet de recherche beaucoup plus large, visant le développement d'un projet d'assistance aux personnes à mobilité réduite. Il doit donc être considéré comme un pas en avant vers la réalisation de ce projet d'envergure.

REMERCIEMENTS

Tout d'abord, je dédie cette thèse à la ma mémoire de feu mon père.

J'aimerais remercier mon directeur de recherche, M. Abdenour Bouzouane, qui grâce à ses idées, toujours judicieuses et son suivi rigoureux de chacune des étapes de réalisation du projet ont permis de mener à bien ce travail de recherche.

Je souhaiterais également remercier mon frère, Patrick Eurin, pour ses efforts, ainsi que son support, sans qui tout ceci n'aurait jamais abouti.

Dans un second temps, j'aimerais remercier M. Djamal Rebaine, pour m'avoir beaucoup apporté dans le domaine de l'enseignement, et surtout, pour sa répartie qui m'a toujours amené à me questionner. De même que Mme. Duygu Kocaefe pour avoir eu la chance de travailler avec elle.

Je remercie aussi mon collègue Bruno Bouchard, pour le temps passé mais aussi pour avoir été un bon compagnon de bureau. Jean François Michaud, dont ses opinions m'ont toujours apporté une autre vision des choses, de même que ses recettes de cuisines qui, pour certaines, sont trop américaines à mon goût. Carine Ruhlmann, pour le temps passé, ses remarques mais aussi sa rigueur dans les relectures de ce mémoire.

Pour finir, je souhaiterais remercier mes amis comme Sabrina, David, Benjamin, Audrey, Jaisheenah et tous les autres, pour le temps passé, pour leurs soutiens apportés, facilitant grandement ma vie au Canada.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

Longtemps, l'informatique en général et l'intelligence artificielle (IA) en particulier ont considéré les programmes comme des entités individualisées capables de rivaliser avec l'être humain. Au départ simple machine à calculer, l'ordinateur s'est ainsi vu consacrer à des tâches qui relevaient de domaines de plus en plus complexes comme la gestion d'une entreprise, la surveillance et le contrôle de processus industriels, l'aide au diagnostique médical ou la conception de nouvelles machines. Cette compétition entre l'être humain et la machine s'est accompagnée d'une identification de la machine à l'humain, un programme représentant directement un « expert » capable de résoudre un problème par lui-même [Ferber, 1995]. Ces programmes donnèrent, d'ailleurs, naissance à la notion d'agents artificiels capables de s'organiser pour accomplir collectivement les fonctionnalités qui leur sont demandées [Chaib-Draa *et al*, 1992] [Wooldridge, 2000]. Ainsi, un agent représente une entité informatique capable de prendre ses propres décisions, de se déplacer à travers un réseau informatique, d'échanger de l'information avec d'autres agents, d'apprendre, etc. Par exemple, dans le domaine de l'aéronautique, l'agent sera utile pour détecter des problèmes sur l'avion et les transmettre au pilote. De même, l'agent pourra détecter, en fonction de certains paramètres, une situation susceptible de dégénérer et fournir au pilote les informations les plus utiles pour la réussite de la tâche requise [Dehais *et al*, 2004]. La réalisation d'une tâche ne sera donc pas l'œuvre uniquement de l'utilisateur ou de l'agent, mais plutôt des deux et dans lequel ils auront les mêmes possibilités de proposer, de suspendre, de refuser et d'arrêter l'autre [Dautenhahn, 2001].

Néanmoins, afin que l'agent puisse devenir un véritable collaborateur, il devra être apte à appréhender les actions de l'usager. Dès lors, afin d'assurer une coopération usager-agent efficace et la moins dangereuse possible, particulièrement dans un contexte où la communication est défaillante à cause des moyens de communication ou imposée par la nature même de l'application (militaire), alors la compréhension du comportement de l'autre est utile pour assister et coopérer d'une manière la plus adéquate.

Dans ce contexte, une des problématiques majeures de cette compréhension est la reconnaissance de plan [Carberry, 2000]. Le problème de la reconnaissance de plan peut, notamment, se synthétiser par la capacité de comprendre ce à quoi s'efforce l'usager ; pour le cas échéant, anticiper son action future. Il est à noter que l'usager qui effectue différentes actions est considéré comme l'acteur et l'agent comme l'observateur. Plus précisément, il est possible de dire que la reconnaissance de plan s'attache à découvrir le but recherché par l'utilisateur, en partant d'une description incomplète des actions observées effectuées par celui-ci un processus de reconnaissance retrouve le plan qui sous-tend et relie ces actions. Cette reconnaissance permettra entre autres de prévoir les actions futures du sujet ou encore de donner à la machine les éléments suffisants pour compléter la tâche souhaitée [Villasenor-Pineda, 1999].

Ainsi, l'objectif général de notre travail de recherche est d'élaborer un ensemble d'outils servant de base à la conception d'un modèle de reconnaissance de plan basé sur la logique de description [Brachman et levesque, 1985] [McGregor, 1991], qui est adéquate au

problème de la reconnaissance de plan en le transformant en un problème de classification de plans. Plus précisément, le premier objectif visé est de clarifier la notion de reconnaissance de plan, ainsi que les éléments essentiels, la constituant. Le deuxième objectif est de redéfinir ses éléments à travers la logique de description. Cette logique est une extension des réseaux sémantiques pour la représentation des connaissances [Minsky, 1974] [Schmidt, 2000]. Elle est basée sur la relation de subsomption qui est une méthode d'inférences pour la classification de concepts à travers une ontologie. Une ontologie pouvant être perçue comme une structure organisée sous la forme d'une hiérarchie. Cette hiérarchie permet de représenter d'une façon explicite la sémantique de l'information structurée relative à un domaine [Davies *et al*, 2003]. Finalement, le troisième objectif, consiste à valider l'approche proposée à travers un exemple d'application provenant de la théorie de Kautz [Kautz, 1987] que nous présentons au chapitre 2.

Dès lors, notre démarche consiste dans un premier temps à entreprendre une étude détaillée de différents travaux sur la reconnaissance de plan, tel que par exemple ceux de Kautz [Kautz, 1987] qui décrivent les actions en plusieurs niveaux d'abstractions, donnant la possibilité de dégager les buts possibles de l'agent acteur, grâce à différentes hypothèses. Ou encore de ceux de Wobcke [Wobcke, 2002] qui décrivent la reconnaissance de plan comme une opération de conséquence, avec l'aide de la logique non monotone. Il est à noter que nous nous focaliserons sur les travaux qui sont basés sur la logique, car notre but est de pouvoir fournir un modèle formel en logique de description pouvant servir à la reconnaissance de plan. L'emploi de la logique évitant d'avoir à créer l'ensemble de tous

les plans possibles comme c'est le cas des approches probabilistes [Charniak *et al*, 1985] [Carberry, 2000] [Albrecht *et al*, 1998]. En effet, ces approches permettent de pouvoir calculer au mieux les probabilités entre les différents plans possibles. En revanche, une approche basée sur la logique n'a pas besoin de créer en premier lieu l'ensemble des plans possibles, car ceux-ci peuvent être représentés uniquement à partir d'une observation. L'étape suivante de notre démarche consiste à réaliser une recherche approfondie sur la logique de description afin de connaître ses capacités et ses limites. Ceci, dans le but de trouver le moyen d'employer celle-ci dans le cadre de la reconnaissance de plan. En effet, l'originalité de notre approche se situe au niveau de l'extension de cette logique de description qui va nous permettre de ramener la problématique de la reconnaissance de plan à un problème de classification. En somme, notre but consiste à fournir un modèle de classification de plan d'actions, servant de support au processus de la reconnaissance de plan. Notre contribution de modélisation d'actions et d'extraction de plans possibles s'insère dans le cadre d'un travail de Doctorat sur la reconnaissance de plan de Bruno Bouchard, sous la direction des professeurs Abdenour Bouzouane et Sylvain Giroux de (Université de Sherbrooke).

Ainsi, dans un premier temps au chapitre 2, nous définissons de manière détaillée la reconnaissance de plan et introduisons les différents éléments essentiels à celle-ci. L'illustration de ces éléments se fera à travers les travaux de Kautz [Kautz, 1987] et Wobcke [Wobcke, 2002].

Le chapitre 3 se veut être un portrait de la logique de description. Nous y abordons ses origines de même que celle des systèmes de représentation des connaissances terminologiques. Ensuite, nous verrons les concepts fondamentaux qui la composent, de même que les différents formalismes, ses avantages et ses limitations.

Finalement, le chapitre 4 présente notre contribution au domaine de la reconnaissance de plan. Dans un premier temps, nous présentons un modèle d'action basé sur la logique de description. Ensuite, dans une seconde étape, nous décrivons un langage permettant d'étendre les possibilités de cette logique, afin que celle-ci puisse répondre aux besoins de la reconnaissance de plan. Un exemple d'application est présenté pour servir de validation à notre modèle.

En conclusion générale, présentée au chapitre5, nous récapitulons le bilan de nos travaux. Nous y présentons également de nouvelles voies de recherches intéressantes qui pourraient en découler.

CHAPITRE 2

LA RECONNAISSANCE DE PLAN

2.1. Introduction

Lorsque l'on veut interagir avec quelqu'un, il est nécessaire de le comprendre, au sens large du terme. En effet, si par exemple une personne vous demande où est situé le magasin d'alimentation le plus proche, afin d'acheter des pâtes, vous en déduirez que cette personne cherche à se procurer des pâtes en l'achetant dans ce dit magasin pour en manger prochainement. Ainsi, cette capacité à comprendre, voir à anticiper les buts d'une personne, est souvent considérée comme un élément de la reconnaissance de plan. Celle-ci pouvant être vue comme une tâche de prédiction sur les buts d'une personne, basée sur les observations de cette dite personne, en train d'accomplir son but [Neal, 1998]. Ce travail fait intervenir un usager qui pose des actions, donc un agent acteur et un autre qui se trouve en position d'observation, donc un agent observateur. De surcroît, un élément intéressant de la reconnaissance de plan est que celle-ci permet par exemple de proposer une aide plus adéquate dans un système interactif [Carberry, 2000] ou lors d'une tâche coopérative de pouvoir effectuer des clarifications pouvant éviter les conflits [Vanbeek *et al*, 1991], etc. Dans un premier temps, nous définissons de manière informelle la reconnaissance de plan, puis nous en décrivons succinctement les approches théoriques telles que celles de Kautz [Kautz, 1987] (décrivant les actions en plusieurs niveaux d'abstractions, donnant la possibilité de dégager les buts possibles de l'agent acteur, grâce à différentes hypothèses) et celles de Wobcke [Wobcke, 2002] (présentant la reconnaissance de plan comme une opération de conséquence).

2.2. Définition

Le problème de la reconnaissance de plan peut être défini comme « prendre en entrée une séquence d'actions effectuées par un acteur et inférer le but poursuivi par celui-ci et aussi organiser la séquence d'actions en terme de structure de plan » [Schmidt *et al*, 1978]. Dès lors, les résultats obtenus peuvent être employés à de multiples fins, comme la construction d'un contexte qui permettra de clarifier les observations futures, l'aide ou l'entrave de l'acteur ou un résumé de la situation [Kautz, 1987]. Si nous considérons l'exemple de l'agent acteur qui sort la boîte de spaghetti, l'agent observateur en déduira éventuellement que celui-ci a pour but de s'alimenter et qu'avec des spaghettis, il est possible de faire des spaghettis aux fruits de mer ou des spaghettis pesto. Une des ambitions de la reconnaissance de plan est de pouvoir, justement, prédire ce que l'acteur souhaite faire, par exemple, des spaghettis aux fruits de mer.

2.3. Approches théoriques de la reconnaissance de plan

La reconnaissance de plan peut se regrouper sous deux grandes approches théoriques, c'est-à-dire la reconnaissance de plan d'intention communicative (« intended ») et la reconnaissance de plan à l'insu (« keyhole ») [Pasquier et Chaib-draa, 2004]. Dans le cadre de la reconnaissance de plan d'intention communicative, l'agent acteur (observé) agit pour que ses actions soient évidentes afin d'aider l'agent observateur. Ainsi, l'agent acteur structure délibérément son action afin de communiquer ses buts. En revanche, dans la reconnaissance de plan à l'insu, l'agent acteur ne possède aucune connaissance de la présence de l'agent observateur ou celui-ci n'y prête pas attention [Neal, 1998]. Il est à

noter qu'il existe une troisième approche théorique, peu courante, que nous ne détaillerons pas ici. Celle-ci concerne le fait que l'agent acteur peut effectuer des plans erronés afin de contrecarrer l'agent observateur [Pollack 1986b] [Azarewicz *et al* 1986].

Néanmoins, nous détaillerons plus particulièrement les recherches concernant la reconnaissance de plan à l'insu, étant donnés que nos travaux portent sur cette approche. En effet, nous souhaitons que l'agent ait une place d'assistant, donc celui-ci doit pouvoir intervenir, mais sa présence ne doit pas être la raison d'une modification du processus de résolution de la tâche, d'où notre choix pour la reconnaissance de plan à l'insu. De la même manière, il existe différents modèles pour formaliser les diverses approches de la reconnaissance de plan, nous nous focaliserons ici sur les modèles à base de la logique du premier ordre. En effet, parmi tous les modèles pour la reconnaissance de plan, il est possible de dégager ceux qui sont à base de la logique du premier ordre et ceux qui sont à base de méthodes probabilistes [Charniak *et al*, 1985], [Carberry, 2000], [Albrecht *et al*, 1998]. Ces modèles permettent, grâce à un ensemble de règle « probabiliste », de réviser les probabilités accordées aux hypothèses en fonction d'une observation. Le plan choisi étant, alors, le plan le plus probable. Un des éléments de cette approche est, d'ailleurs, l'obligation de créer toutes les possibilités de plan, afin de pouvoir calculer au mieux les probabilités en fonction de tous les plans possibles. Un modèle, à base de la logique du premier ordre, permet de représenter les plans possibles grâce à une observation. Il n'est donc pas obligatoire de créer tous les plans possibles. De plus, notre souhait est de fournir un modèle servant de fondation au processus de la reconnaissance de plan, basé sur

la logique description. Dans ces conditions, nous présenterons ici des modèles basés sur la logique du premier ordre, afin de mieux introduire le chapitre 4, portant sur nos travaux. En premier lieu, nous regarderons les travaux de Kautz [Kautz, 1987] qui fournissent un fondement théorique à l'inférence de plans, d'une manière générale et sans être liés à un domaine particulier, tout en s'étendant sur une suite d'observations, dont chacune permet d'affiner la conclusion tirée à partir des précédentes. Puis nous nous pencherons sur ceux de Wobcke [Wobcke, 2002], qui comme Kautz enrichit la conclusion à la suite d'une série d'observations, mais celui-ci place la reconnaissance de plan dans un sous-ensemble plus réaliste vis-à-vis de l'agent acteur.

2.4. Elément de la théorie de Kautz

Dans l'approche de Kautz [Kautz, 1987], l'agent observateur démarre avec une collection d'actions qui constituent sa base de connaissances. Les actions pouvant avoir un certain nombre de sous-types (une manière de spécialiser les types d'actions) ou des décompositions (une procédure à suivre pour exécuter l'action), l'ensemble formant une hiérarchie d'abstraction/décomposition [Wobcke, 2002]. Un exemple de hiérarchie sur la préparation de quelques recettes de cuisine est décrit à la Figure 1 ; nous l'utiliserons afin d'illustrer cette approche.

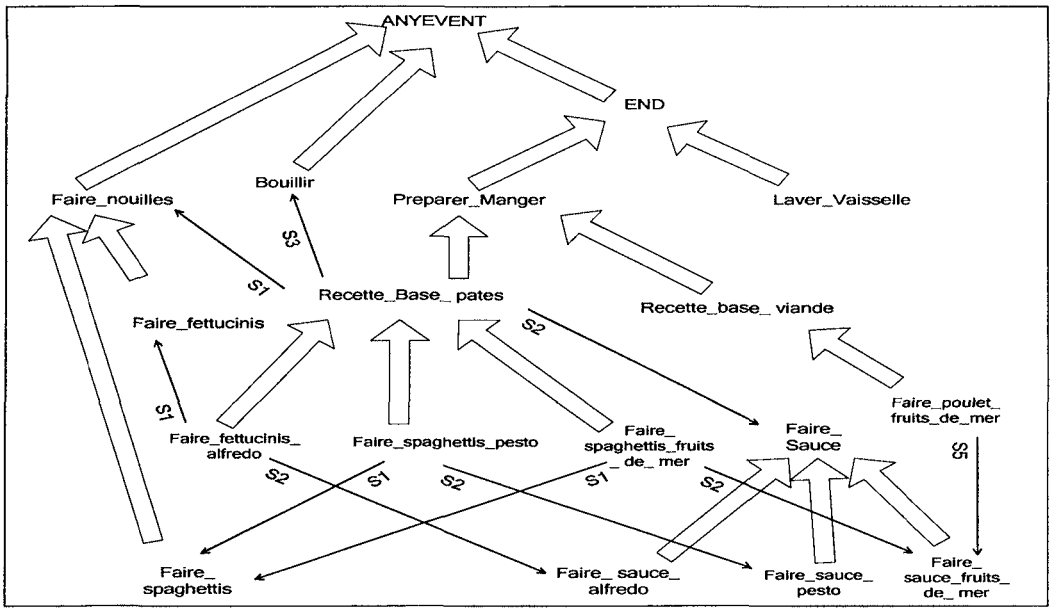


Figure 1 : Hiérarchie d'action de recette de cuisine.

Les flèches plus épaisses dénotent les relations d'abstraction (allant du spécifique vers le général) tandis que les flèches plus fines indiquent les relations de composition directe (allant d'un schéma de plan vers ses éléments le composant ; ceux-ci ordonnés par étapes indiquées dans la figure par S1 et S2). En outre, la hiérarchie possède deux types d'actions spéciales, c'est-à-dire les actions END et ANYEVENT. Les spécialisations de END sont les actions que l'on peut considérer comme motivées par elle-même et indique à quel moment la reconnaissance de plan va s'arrêter. ANYEVENT étant l'action la plus générale de la hiérarchie. END et ses spécialisations étant l'ensemble des schémas qui n'apparaissent pas dans la décomposition d'aucun type d'action. Dans un premier temps, nous définissons les éléments de la hiérarchie d'action. Ceci nous permettra de voir les différentes hypothèses proposées par Kautz, pour enrichir le processus de reconnaissance de plan afin d'arriver

aux conclusions sur les buts de l'acteur. Finalement, nous regarderons les propositions de celui-ci pour l'implémentation, telle que les graphes explicatifs ou les algorithmes de minimisation des modèles de couverture.

2.4.1. Hiérarchie d'actions

La hiérarchie d'actions, qui représente les connaissances des usagers, définit les abstractions, les spécialisations entre les actions, comme vu précédemment. Celle-ci peut être vue comme la forme logique des réseaux sémantiques tels que décrits par Hayes [Hayes 1985]. Il est à noter que l'agent acteur et l'agent observateur possèdent une connaissance identique, mais sans la partager, par exemple, chacun possède deux versions du même livre de cuisine. Par ailleurs, la logique utilisée dans l'approche de Kautz est la logique du premier ordre. En effet, Kautz utilise une représentation améliorée des actions tirées de Allen [Allen, 1983] dans laquelle les actions sont représentées par des termes et les types d'actions par des prédicats, ainsi une formule telle que *a(x)* dénote la proposition telle que l'action *x* observée est de type *a*. Ensuite, comme nous l'avons précisé ci-dessus, les actions possèdent des relations entre elles, comme les relations d'abstraction et de décomposition. La relation d'abstraction peut être définie informellement par le fait qu'un type d'action a_1 abstrait directement un type d'action a_2, s'il y a une simple flèche d'abstraction de a_2 vers a_1.

Une hiérarchie *H*, telle que vue à la Figure 1, peut se découper en différentes parties : H_E, H_A, H_{EB}, H_D et finalement H_G

- H_E est l'ensemble des prédicats atomiques, contenant non seulement les actions les plus spécifiques, mais aussi les actions END et ANYEVENT.
- H_A est l'ensemble des axiomes d'abstraction tel que :

$$\forall x, \exists (a_1, a_2) \in H_E^2 \mid a_1(x) \supset a_2(x), x \text{ étant l'action observée.}$$

Il est donc possible de dire que a_2 est une abstraction directe de a_1. Par exemple, dans la Figure 1, nous pouvons voir que *« Preparer_manger »* abstrait *« Recette_base_pates »* cela va donc se représenter comme suit :

$$\forall x.\ (Preparer\_manger\ (x) \supset\ Recette\_base\_pates(x))$$

- H_{EB} est l'ensemble des prédicats de type simple, ils n'abstraient aucun autre type d'action.
- H_D est l'ensemble des axiomes de décomposition, chacun étant de la forme :

$$\forall x.\ a_0(x) \supset (a_1(etape_1(x)) \wedge a_2(etape_2(x)) \wedge \ldots \wedge a_n(etape_n(x)) \wedge \kappa$$

Où $a_0, \ldots, a_n \in H_E$. a_1 à a_n sont appelées les composants directs de E_0. a_1 étant le composant direct de a_0; $etape_1, \ldots,$ $etape_n$ étant les fonctions identifiant précisément les étapes dans la décomposition ; et κ décrivant un regroupement de plusieurs contraintes sur e_0. Ces contraintes pouvant être des contraintes de temps ou d'utilisateur (c'est-à-dire qui doit faire la tâche). Cela pouvant décrire aussi les préconditions nécessaires ou finalement les effets de cette action [Kautz, 1987]. Voici un exemple illustrant un axiome de décomposition, basé sur la Figure 1.

$\forall x.Recette\_base\_pates(x) \supset$	*Faire_nouilles(Etape1(x))* $\wedge$
Composants	*Faire_sauce(Etape2(x))* $\wedge$ *Bouillir(Etape3(x))* $\wedge$
Contraintes d'égalités	*Agent(Etape1(x)=agent(x))* $\wedge$
	Resultat(Etape1(x))=Entrée(Etape3(x)) $\wedge$
Contraintes temporelles	*Durant (Temps(Etape1(x)),temps(x))* $\wedge$
	AvantRencontre(Temps(etape1(x)),Temps(Etape3(x))) $\wedge$
	Chevauchement(Temps(x),postTemps(x)) $\wedge$
Preconditions	*Lieu(Cuisine(agent(x)), Temps(x))* $\wedge$ *Agile(Agent(x))* $\wedge$
Effets	*Lieu(PresManger(result(x)), PostTemps(x))* $\wedge$
	Recette_base_pates(resultat(x))

En outre, le type END n'apparaît jamais comme le composant direct d'un autre type d'action ; de même qu'aucune autre action n'apparaît lorsque l'action END l'abstrait.

- H_G est l'ensemble des axiomes en général, ceux qui ne contiennent aucun membre de H_E. Essentiellement, H_G contient les axiomes pour les contraintes de temps.

2.4.2. Inférences de plans

Force est de constater que la hiérarchie d'action n'est pas suffisante pour effectuer la reconnaissance de plan, car pour pouvoir arriver à reconnaître au mieux les actions de l'agent acteur, Kautz a introduit une série d'hypothèses afin d'enrichir la hiérarchie d'action et un modèle de couverture. Un modèle permet d'interpréter un langage associant

par exemple, les termes aux objets, reliant des tuples d'éléments entre eux ou finalement faisant correspondre des prédicats à des tuples d'éléments. Ainsi, un modèle de couverture peut s'expliquer par le fait que chacune des actions contenues dans ce modèle est expliquée ou « couverte » par l'action END ; celle-ci étant aussi contenue dans ce dit modèle. Dès lors, une action qui n'est pas une spécialisation de l'action END intervient donc comme un composant d'une autre action. En somme, nous verrons, en premier lieu, l'hypothèse d'exhaustivité permettant d'obtenir les actions spécialisant une observation, puis l'hypothèse d'antinomie permettant de définir une relation disjonctive entre deux actions. Puis nous regarderons l'hypothèse de composant/utilisation permettant d'obtenir les causes d'une observation. Pour finir, nous étudierons deux points concernant les modèles de couvertures et leur minimisation, permettant respectivement de construire une explication pour une observation et de pouvoir rassembler ses explications pour tenter de prédire le plan envisagé par l'agent acteur.

2.4.2.1. Hypothèse d'exhaustivité

L'hypothèse d'exhaustivité permet d'obtenir à partir d'une observation, toutes les actions qui sont directement abstraites par celle-ci. Ainsi, pour une observation a_0 appartenant à H_A et $\{a_1,\ldots, a_n\}$, l'ensemble des prédicats décrivant les actions et qui est directement abstraits par a_0. Nous pouvons décrire l'ensemble EXA comme étant l'ensemble des actions qui abstrait directement a_0, de la forme :

$$\forall x.\ a_0(\mathrm{x}) \supset (a_1(\mathrm{x}) \vee \ldots \vee a_n(\mathrm{x}))$$

Si nous prenons le cas où nous avons l'information comme quoi l'agent acteur prépare une sauce, mais que ce n'est pas une sauce Alfredo ni une sauce pesto. En employant la Figure 1, nous pouvons dire par EXA que :

$$\forall x.\ Faire\_sauce(x) \supset Faire\_sauce\_alfredo(x) \vee Faire\_sauce\_fruits\_de\_mer\ (x) \vee Faire\_sauce\_pesto(x)$$

Avec les informations tirées des observations, nous en déduisons, toujours grâce à la Figure 1, que :

$$\forall x.\ Faire\_sauce(x) \wedge \neg Faire\_sauce\_alfredo(x) \wedge \neg Faire\_sauce\_pesto(x) \supset Faire\_sauce\_fruits\_de\_mer\ (x)$$

Finalement, il est possible de déduire, jusqu'à preuve du contraire, que l'agent acteur prépare une sauce aux fruits de mer.

2.4.2.2. Hypothèse d'antinomie (« Disjointedness »)

Considérons deux actions de la hiérarchie qui ne s'abstraient pas l'une par rapport à l'autre. Nous pouvons normalement présumer que celles-ci sont disjointes. Ainsi, pour a_1 et a_2 deux actions non compatibles[1], DJA est l'ensemble des prédicats de la forme :

$$\forall x.\ \neg a_1(x) \vee \neg a_2(x)$$

Nous pouvons considérer l'exemple suivant :

$$\forall x.\ Recette\_base\_pates(x) \supset \neg Recette\_base\_viande(x)$$

$$\forall x.\ Recette\_base\_viande(x) \supset \neg Recette\_base\_pates\ (x)$$

[1] On dira que deux actions sont compatibles lorsque l'une abstrait l'autre, soit directement ou par transitivité. Par exemple, « Recette à base de pâtes » et « Faire des fettucinis Alfredo »

Dès lors, nous pouvons en déduire que :

$$\forall x.\ \neg Recette\_base\_pates(x) \vee \neg Recette\_base\_viande(x)$$

2.4.2.3. Hypothèse de composant/utilisation (« Component/use »)

Un élément clé de la reconnaissance de plan est de pouvoir retrouver les causes possibles d'une observation à travers la hiérarchie. De ce fait, si l'agent observateur reçoit l'information telle que l'agent acteur a effectué l'action *« Bouillir »*, nous pouvons en conclure que celui-ci effectue une recette à base de pâtes. L'hypothèse serait donc :

$$\forall x.\ Bouillir(x) \supset \exists y.Recette\_base\_pates(y) \wedge x = etape3(y)$$

Cela signifie que si l'on observe une action x de type *Bouillir*, alors on peut supposer qu'il existe un plan y de type *Recette_Base_pates* et que l'action x est la troisième étape du plan en question. Ainsi, nous pouvons définir l'ensemble CUA comme suit : $\forall a \in H_E$, soit *COM(E)* l'ensemble des prédicats avec lequel a est compatible. Ainsi, le $j^{ème}$ axiome de décomposition à la forme suivante, où a_{ji} est l'élément de *COM(E)* :

$$\forall x.\ a_{j0}(x) \supset a_{j1}(etape_{j1}(x)) \wedge \ldots \wedge a_{ji}(etape_{ji}(x)) \wedge \ldots \wedge a_{jn}(etape_{jn}(x)) \wedge \kappa$$

($etape_1,\ldots,\ etape_n$ étant les fonctions identifiant précisément les étapes dans la décomposition)

Concrètement l'ensemble CUA, représente l'ensemble des actions qui ont pour composant direct, l'observation elle-même. CUA représentant aussi les actions qui ont pour composant direct, une action compatible avec l'observation. On considéra aussi par CUA, toutes les actions compatibles avec une action déjà présente dans cet ensemble. Si nous

considérons l'exemple suivant où *« Faire_nouilles »* est observée. Nous obtenons par CUA les actions suivantes qui ont pour composant direct l'observation :

« Recette_base_pates »

Si nous considérons les actions compatibles, nous obtenons par abstraction les actions suivantes :

$$\textit{« Faire\_Fettucinis\_Alfredo »} \vee \textit{« Faire\_spaghettis\_Pesto »} \vee$$

$$\textit{« Faire\_spaghettis\_fruits\_de\_mer »}$$

Nous pouvons dire par CUA que :

$$\forall x.\ \textit{Faire\_nouilles}(x) \supset \exists y.\ \textit{Recette\_base\_pates}(y) \wedge x = \textit{etape1}(y) \vee$$

$$\exists y.\ \textit{Faire\_Fettucinis\_Alfredo}\ (y) \wedge x = \textit{etape1}(y) \vee$$

$$\exists y.\ \textit{Faire\_spaghettis\_Pesto}\ (y) \wedge x = \textit{etape1}(y) \vee$$

$$\exists y.\ \textit{Faire\_spaghettis\_fruits\_de\_mer}\ (y) \wedge x = \textit{etape1}(y)$$

En simplifiant les relations non présentes dans la hiérarchie, finalement nous obtenons ceci :

$$\forall x.\ \textit{Faire\_nouilles}(x) \supset \exists y.\ \textit{Recette\_base\_pates}(y) \wedge x = \textit{etape1}(y) \vee$$

$$\exists y.\ \textit{Faire\_Fettucinis\_Alfredo}\ (y) \wedge x = \textit{etape1}(y)$$

Il est à noter que nous voyons ici les prémisses de l'explication d'une observation. Ceci pouvant se faire en construisant un modèle de couverture contenant les actions permettant d'arriver à cette dite explication.

2.4.2.4. Modèle de couverture

Le modèle de couverture contient l'ensemble des actions montrant le plan possible permettant d'expliquer une observation. En effet, la tâche de reconnaissance, sous-entend que le plan exécuté par l'agent acteur est une instance d'un plan que l'agent observateur peut former à partir de sa hiérarchie [Wobcke, 2002]. Ainsi, en partant de l'action END et en parcourant la hiérarchie jusqu'à ladite observation, on peut former le modèle de couverture correspondant. Ceci explique le nom de cet ensemble, car chaque action à l'intérieur est abstraite, couverte par l'action END. Ainsi, nous pouvons définir un modèle de couverture *M* pour une observation donnée par : *M* est vrai pour l'ensemble défini par $H_E \cup CUA \cup EXA \cup DJA$. Par exemple, si nous avons l'observation *« Faire_Fettucinis »*, le modèle de couverture correspondant sera alors :

> $\exists a.\{END(a), Preparer\_Manger(a), recettes\_base\_pates(a), faire\_Fettucinis\_Alfredo(a), Faire\_Fettucinis(S_1(a)), faire\_sauce\_Alfredo(S_2(a))\}$, S_1 et S_2 étant les étapes symbolisant l'ordre dans lequel sont exécutées les actions correspondantes.

Cet ensemble peut ainsi expliquer une observation, mais lorsque plusieurs observations surviennent, il y a autant de modèle de couverture que d'observations. Il est donc nécessaire de réduire leur nombre si la situation le permet.

2.4.2.5. Minimisation du nombre de modèles de couverture

Jusqu'à présent, la possibilité de combiner différentes observations n'a pas encore été évaluée. Les algorithmes proposés par Kautz, cherchent en premier lieu à regrouper les

actions entre elles. Si cela n'est pas possible, c'est que l'agent acteur effectue plusieurs plans d'action distincts. De plus, Kautz suppose que l'agent acteur n'effectue pas d'actions venant contredire une précédente observation, ainsi chacune des actions surviennent dans une suite logique d'un plan. A cette fin, il est nécessaire de voir s'il est possible de combiner les actions END. Pour cela, il est nécessaire de considérer les cas possibles de combinaison de END définis par MA_i de 0 à n :

MA_0 $\quad \forall x.\ \neg \mathrm{End}(x)$

MA_1 $\quad \forall x, y.\ \mathrm{End}(x) \wedge \mathrm{End}(y) \supset x = y$

MA_2 $\quad \forall x, y, z.\ \mathrm{End}(x) \wedge \mathrm{End}(y) \wedge \mathrm{End}(z) \supset (x = y) \vee (x = z) \vee (y = z)$

.
.
.

Donc, soit Γ l'ensemble des formules représentant les observations. La minimisation du nombre de plans possibles pour H_E et Γ est la formule MA_i, où i est le plus petit entier tel que

$$\Gamma \cup H_E \cup CUA \cup EXA \cup MA_i$$

De surcroît, ceci se calcule en tenant compte des informations apportées par les observations. En effet, si une information précise qu'une action est explicitement exclue, cela peut empêcher de pouvoir minimiser les modèles de couvertures. Par exemple, si nous prenons le cas où l'agent acteur est en train de préparer des spaghettis et une sauce aux fruits de mer. La première observation nous permet de déduire, grâce à l'hypothèse de composant/utilisation, vue ci-dessus, que l'acteur effectue un plan faisant intervenir l'action

« faire_spaghettis », cela concerne donc les plans *« Faire_spaghettis _fruits_de_mer »* et *« Faire_spaghettis_pesto »*. De plus, la deuxième observation nous permettra de dire qu'il effectue un plan faisant intervenir l'action *« Faire_sauce_fruits_de_mer »*, ceci permettant de dégager les plans suivants : *« Faire_spaghettis_fruits_de_mer »* ou *« Faire_poulet_fruits_de_mer »*. Nous obtenons ainsi les modèles de couvertures suivants, que nous appellerons successivement M_1, M_2, M_3. S_1 et S_2 étant les étapes symbolisant l'ordre dans lequel sont exécutées les actions correspondantes.

M_1 : $\exists a$. *{END(a), Preparer_Manger (a), recettes_base_pates (a), Faire_spaghettis_Pesto(a), Faire_spaghettis(S_1(a)), Faire_pesto(S_2(a))}*

M_2 : $\exists a$. *{END(a), Preparer_Manger (a), recettes_base_pâtes (a), Faire_spaghettis_fruits_de_mer (a), Faire_spaghettis (S_1 (a)), Faire_sauce_fruits_de_mer(S_2(a))}*

M_3 : $\exists a$. *{END(a), Preparer_Manger (a), recettes_base_viande (a), Faire_poulet_fruits_de_mer (a), Faire_sauce_fruits_de_mer (S_1(a))}*

Mais nous n'avons pas de moyen pour tirer la conclusion que l'agent acteur effectue le plan *« Faire_spaghettis_fruits_de_mer »*. Hors, une action ne peut pas être deux actions à la fois, c'est-à-dire qu'elle ne peut pas être une viande de poulet et en même temps une viande de bœuf. Partant de cette idée, nous pouvons constater que M_3 est écartée, car les

observations ne contiennent pas de poulet, de même pour M_1, car les observations ne contiennent pas l'action *« Faire_pesto »*. Dès lors, le seul modèle de couverture qui contient à la fois des spaghettis et des fruits de mer est M_2. Il est noté qu'il n'y a pas d'information contredisant cette inférence. En effet si l'agent observateur sait que l'acteur n'effectue pas le plan *« Faire_spaghettis_fruits_de_mer »*, on obtient alors deux plans non reliés. La seule conclusion serait que celui-ci effectue le plan *« Faire_spaghettis »* puis effectue le plan *« Faire_sauce_fruits_de_mer »*.

2.4.2.6. Exemple d'application

Considérons l'exemple suivant, toujours inspiré par le monde de la cuisine, telle qu'à la Figure 1, pour illustrer les points ci-dessus. Les informations étant numérotées, nous pourrons ainsi montrer quel axiome qui nous permettent de les obtenir.

Connaissance initiale :

On suppose que l'agent observateur dispose de la connaissance initiale comme quoi l'acteur ne désire pas faire une sauce Alfredo et qui se traduit par :

[0] $\forall x.\ \neg\, Faire\_sauce\_alfredo(x)$

Première observation :

[1] $Faire\_nouilles\ (Obs1)$

Hypothèse de composant/utilisation :

[2] $\exists i_1.\ Recette\_base\_pâtes\ (i_1) \wedge etape(i_1) = Obs1$

Abstraction avec [2] :

[3] $\exists i_1.\ Preparer\_manger\ (i_1)$

Abstraction avec [3] :

[4] $\exists i_1.\ END(i_1)$

A ce point, nous avons reconstruit le plan d'action de l'agent acteur. Mais nous pouvons, dès lors, effectuer certaines prédictions afin d'anticiper les futures actions de l'agent.

Décomposition [2] :

[5] $\exists i_1.\ Bouillir\ (etape3\ (i_1)) \wedge Apres\ (temps(x),\ temps\ (etape3\ (i_1)))$

Nous pouvons constater que la simple information en [2], n'est pas suffisante pour tirer les conclusions qui seraient définitives. D'autres inférences sont dès lors nécessaires.

Hypothèse d'exhaustivité [2] :

[7] $\exists i_1.Faire\_spaghettis\_fruits\_de\_mer(i_1) \vee Faire\_spaghettis\_pesto(i_1) \vee Faire\_Fettuccinis\_Alfredo(i_1)$

Décomposition :

[8] $\exists i_1.\ Faire\_Fettuccinis\_Alfredo(i_1) \supset Faire\_sauce\_alfredo(etape2(i_1))$

Modus tollens [0,8] :

[9] $\exists i_1.\ \neg Faire\_Fettuccinis\_Alfredo(i_1)$

Réduction d'antinomie [7,9] :

[10] $\exists i_1.\ Faire\_spaghettis\_fruits\_de\_mer(i_1) \vee Faire\_spaghettis\_pesto(i_1)$

Décomposition :

[11] $\exists i_1.\ Faire\_spaghettis\_fruits\_de\_mer(i_1) \supset Faire\_spaghettis(etape1(i_1))$

[12] $\exists i_1.\ Faire\_spaghettis\_pesto(i_1) \supset Faire\_spaghettis(etape1\ (i_1))$

Raisonnement par cas [10,11,12] :

[13] $\exists i_1$. *Faire_spaghettis(etape1 (i_1))*

Supposons maintenant qu'une deuxième observation, apprend à l'agent observateur que l'agent acteur effectue une sauce aux fruits de mer. La minimisation du nombre de plans possible, va permettre à l'agent observateur d'effectuer l'intersection entre les explications possibles, comme la première observation et la deuxième. Ceci dans le but d'atteindre la conclusion que l'agent acteur effectue une sauce aux fruits de mer.

Deuxième Observation :

[14] *Faire_sauce_fruits_de_mer(Obs2)*

Hypothèse de composant/utilisation [14] :

[15] $\exists i_2$. *Faire_spaghettis_fruits_de_mer(i_2)* $\vee$ *Faire_poulet_fruits_de_mer(i_2)*

Abstraction [15] :

[16] $\exists i_2$. *Recette_base_pates(i_2)* $\vee$ *Recette_base_viande(i_2)*

Abstraction [16] :

[17] $\exists i_2$. *Preparer_manger(i_2)*

Abstraction [17] :

[18] $\exists i_2$. *END (i_2)*

Minimisation des plans possibles :

[19] $\forall x, y$. *End(x)* $\wedge$ *End(y)* $\supset$ $x=y$

Modus ponens [4, 17,19] :

[20] $i_1 = i_2$

Substitution des égalités [2,30] :

[21] $\exists i_2.\ Recette\_base\_pates(i_2)$

Suppositions disjointes [2,30] :

[22] $\forall x.\ \neg Recette\_base\_pâtes(x) \vee \neg Recette\_base\_viande(x)$

Réduction d'antinomie [21,22] :

[23] $\exists i_2.\ \neg Recette\_base\_viande(i_2)$

Abstraction :

[24] $\exists i_2.\ Faire\_poulet\_fruits\_de\_mer(i_2) \supset Recette\_base\_viande(i_2)$

Modus Tollens [23,24] :

[25] $\exists i_2.\ \neg Faire\_poulet\_fruits\_de\_mer(i_2)$

Réduction Disjonctions [15,25] :

[26] $\exists i_2.\ Faire\_spaghettis\_fruits\_de\_mer(i_2)$

L'agent observateur conclu, donc, que l'acteur a pour but de préparer des spaghettis aux fruits de mer, en partant des observations *Obs1* et *Obs2*.

2.4.3. Algorithme de reconnaissance des observations

L'algorithme proposé par Kautz reprend les points suivants : à chaque observation, appliquer l'hypothèse de composant/utilisation puis d'abstraction, vue ci-dessus, jusqu'à ce qu'une action de type END soit obtenue. Ensuite il y aura tentative de réduire les disjonctions. Afin de combiner plusieurs observations, il sera appliqué l'hypothèse de minimisation des modèles de couverture. Si à la fin de l'utilisation de cette hypothèse, il existe des observations pour lesquelles il n'a pas été possible de les rassembler, alors on

peut conclure que les observations appartiennent à des plans distincts. Par ailleurs, des propositions de plus en plus grandes sont générées par un emploi répétitif de l'hypothèse de composant/utilisation. En effet, la première application de l'hypothèse génère une disjonction, en appliquant celle-ci sur chacune d'entre elles, cela va générer d'autres disjonctions et ainsi de suite. Pour cela, il utilise une structure de graphe permettant de représenter un modèle de couverture correspondant.

La construction de cette structure s'effectue en deux étapes. Tout d'abord on place le nœud correspondant à l'observation. Ensuite on dispose récursivement les nœuds faisant abstraction de celui-ci jusqu'à obtenir l'action END. Il est possible de dire informellement, qu'un graphe explicatif contient 3 types de nœuds. Un nœud action, ordonnancé de N_1 à N_n, des nœuds symbolisant des noms propres (par exemple, « Jean ») ou encore des nœuds désignant des limites temporelles composées de quadruplets (ceux-ci représentant des contraintes temporelles $<d_i, d_e, f_i, f_e>$; d_i et d_e définissent respectivement la borne minimale et maximale du moment de début de l'action ; f_i et f_e représentant les bornes minimales et maximales du moment de fin de l'action).

Il y a aussi deux types d'arcs : les arcs paramétrés et les arcs alternatifs. Un arc paramétré peut être défini par le triplet suivant : $<n, f_r, v>$ c'est un arc du nœud action n, étiqueté avec la fonction temporelle f_r, pointant sur le nœud action v. Le deuxième type d'arc est l'arc alternatif. Celui-ci pointe d'un nœud d'un certain type vers un nœud d'un type plus spécialisé. Il peut être défini par le triplet suivant, $<n, =, m>$, indiquant que le

nœud m est une alternative au nœud n, dans la structure de graphe. A la Figure 2, nous pouvons voir un exemple illustrant la représentation du modèle de couverture pour l'observation de « Faire_sauce_fruits_de_mer »

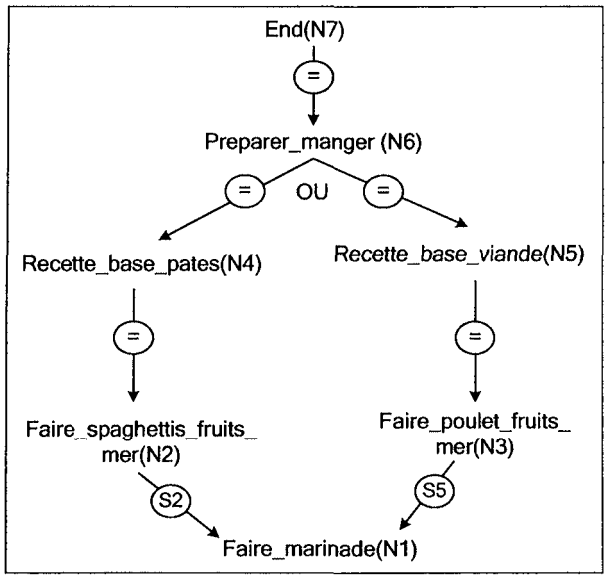


Figure 2 : Structure de graphe explicatif pour faire une sauce aux fruits de mer tirée de la section 2.4.2.6.

Il est à noter que la Figure 2 ne comporte que des nœuds représentants les actions. Elle contient par exemple les deux arcs alternatifs suivants : *<Recette_base_de_pâtes(N_4), =,Faire_spaghettis_fruits_de_mer(N_2)>* et *<Recette_base_viande(N_5), =, Faire_poulet _fruits_de_mer(N_3)>* et finalement les deux arcs paramétrés s_2 et s_5. Par exemple, *<Faire_poulet_fruits_de_mer(N_3), S_5, Faire_sauce_fruits_de_mer(N_1)>*.

2.4.3.1. Algorithme pour la minimisation des modèles de couverture

De manière informelle, l'algorithme de minimisation de Kautz consiste en premier lieu à construire un graphe explicatif correspondant à une observation. Lorsque l'on obtient une deuxième observation, on construit le nouveau graphe et on cherche à regrouper les deux graphes ensemble. S'il n'est pas possible de regrouper les graphes alors il est possible de conclure que les observations appartiennent à des plans disjoints. La figure suivante illustre ce processus.

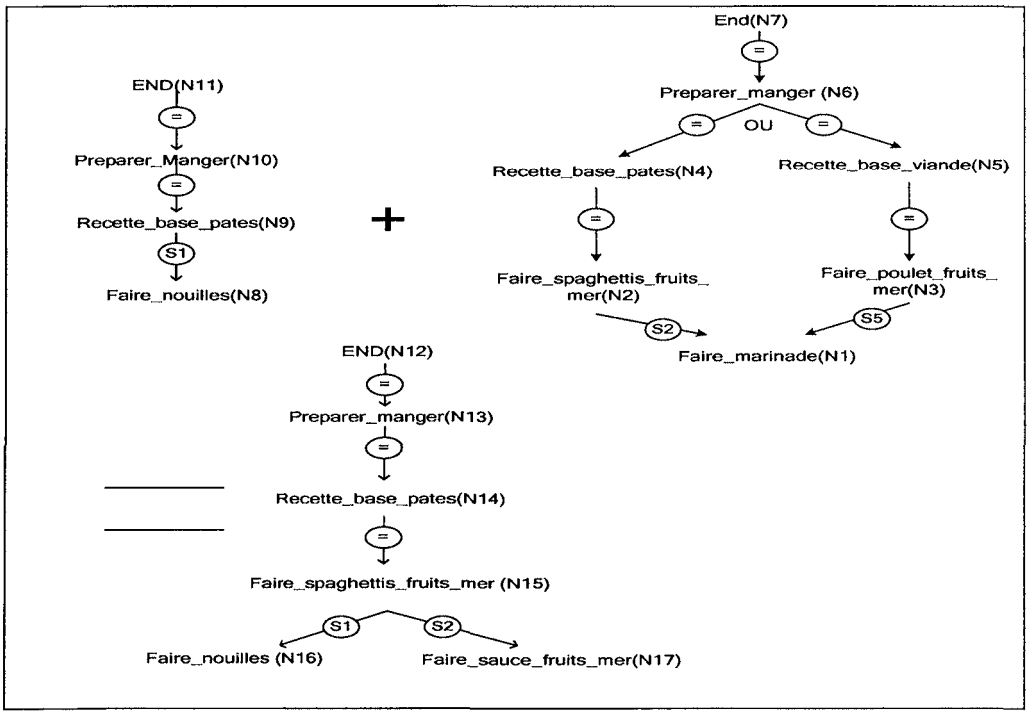


Figure 3 : Minimisation des graphes explicatifs tirée de la section 2.4.2.6.

Dans la Figure 3, nous pouvons voir les graphes explicatifs des observations décrites dans l'exemple d'application, cité précédemment. Le processus de minimisation débute par les deux actions END (à savoir *END(N11)* et *END(N7)*), comme il s'agit de la même action, on ne considéra qu'une seule des deux. Puis, comme les actions END possèdent des relations d'abstractions, alors on va ajouter récursivement l'action la plus spécifique. Lorsqu'il existe plusieurs possibilités de relations d'abstractions, donc d'alternatives, chacune d'entre elles sera parcourue afin de rechercher la plus spécifique. S'il existe une antinomie entre deux actions, la partie du graphe qui comporte l'action testée, est abandonnée. Comme c'est le cas pour « *Recette_base_pates*(N_9) » et « *Recette_base_viande*(N_5) », où la branche contenant cette dernière action est abandonnée du fait de l'antinomie entre les deux. Finalement, lorsque l'on arrive à un arc paramétré, de la même manière, on va chercher à insérer l'action la plus spécifique. S'il n'existe aucune relation d'abstraction entre les deux actions, celles-ci sont insérées au dernier nœud séparément. Comme cela peut être le cas pour « *Faire_sauce_fruits_de_mer*(N_{16}) » et « *Faire_nouilles* (N_{17}) ».

2.5. Bilan sur la théorie de Kautz

Kautz a été un des premiers à proposer un cadre formel pour la reconnaissance de plan basé sur la hiérarchie d'actions et la construction d'un certain nombre d'hypothèses formulées en logique du premier ordre telles que les hypothèses d'exhaustivité, de composant/utilisation, de minimisation des modèles de couverture, etc. La reconnaissance de plan est alors réduite à un processus d'inférence afin de prédire des actions futures suite

à une ou plusieurs observations. Un des avantages de la théorie de Kautz est qu'elle est basée sur la logique du premier ordre, qui est simple et bien établie, contrairement aux autres logiques (théorie des mondes possibles [Kripke 1991], etc...). La théorie de Kautz présente une limite concernant le fait que la structure est figée. L'ajout d'une action dans celle-ci provoquerait une remise en question d'une partie des éléments la constituant. Kautz ne propose pas d'outils pour gérer le cas où la structure de connaissances est différente, voire conflictuelle, entre l'agent observateur et l'agent acteur. En effet, il pose l'idée que l'agent observateur et l'acteur possèdent la même connaissance, celle-ci n'est pas partagée mais identique. Une observation peut venir contredire une action déjà observée, créant ainsi une situation d'antagonisme.

De plus, l'hypothèse de minimisation des modèles de couverture ne prend pas en compte ni l'héritage multiple, pour représenter un maximum de relations dans la hiérarchie ni le choix d'un seul plan plausible parmi un ensemble de plans candidats. Notons que contrairement à cette approche, nous en proposons une, qui se basant sur la logique de description, tente de remédier à cette contrainte d'héritage multiple par l'utilisation de la relation de subsomption de la logique de description. Précisons qu'une logique peut être définie de manière informelle comme monotone si, quels que soient les théorèmes d'une théorie, ceux-ci ne changent pas si on vient ajouter des axiomes à la théorie. Dans la logique non-monotone, un théorème dans une théorie formelle ne reste pas nécessairement un théorème lorsque cette théorie est augmentée: nos croyances précédentes peuvent

changer. Parmi les auteurs ayant repris les travaux de Kautz, Wobcke [Wobcke 2002] s'inspire de la théorie de Kautz pour proposer une approche non-monotonique.

2.6. Théorie de Wobcke

Wobcke propose une approche non monotonique basée sur les travaux de Kautz en utilisant la théorie des situations [Barwise et Perry, 1983], qui est un cas particulier de la théorie des mondes possibles [Kripke 1991], [Lewis, 1986]. En effet, dans la théorie des situations, une situation est représentée par un ensemble d'infons qui symbolisent des actions survenues à un moment donné au cours du temps. Une action est une expression de la forme $\pi(x_1 : \tau_1, \ldots, x_n : \tau_n)$, où π est un type d'action et chaque x_i est un objet manipulée par l'action de type de τ_i. Dès lors, un type d'action (ou schémas de plan) peut se définir comme une structure contenant un nom unique ; possiblement une action parent dont le schéma en serait le sous-type , un ensemble pouvant être vide contenant les actions composantes de ce schéma et un ensemble de contraintes sur les actions composantes de ce schéma. Ainsi, dans un premier temps, nous définirons les notions d'infons et de situations qui sont utilisées par Wobcke pour représenter les plans de reconnaissance, ainsi que la manière d'exprimer la reconnaissance de plan sous forme d'opération de conséquences monotonique $\vdash$. Puis nous verrons les différents éléments de la hiérarchie de plans. Ceci nous permettra de regarder les différentes hypothèses pour la reconnaissance de plan proposé par Wobcke, dont par exemple les hypothèses d'abstractions et de décompositions définissant les relations entre les actions. Finalement, nous regarderons comment Wobcke

définit les plans simples, ceux-ci correspondant au plan que l'on peut tirer de la hiérarchie et comment celui-ci, les ordonnent grâce à une approche non monotonique.

2.6.1. Notion infon

Un infon est une unité basique d'information, symbolisant une action durant une période de temps précise. On peut le représenter par un symbole de relation, de ses arguments et d'une polarité (positive ou négative ou non définis), qu'il est possible d'exprimer par : « P, $a_1,\ldots, a_n, i$ » où P représente la relation, a_1 à a_n étant les arguments en relation avec P, i étant la polarité (0 ou 1) permettant d'indiquer le fait que la relation est effectuée ou non selon une situation donnée. Un infon n'appartient pas à une situation précise, il peut porter sur une ou plusieurs situations, voir même être indéfinies pour certains d'entre elles. Nous pouvons illustrer cela par l'exemple suivant :

- *« Qu'est-ce qui s'est passé dans les bois cet après-midi ?*
- *Jackie s'est cassé la jambe »*

Si nous considérons que Jackie s'est cassé la jambe à un instant t, il est possible de définir deux états de la situation, à savoir Jackie s'est cassé la jambe et Jackie ne s'est pas cassé la Jambe. Ceci se représentant comme suit :

a : <<Jambe_cassé, t , Jackie, 1>>

a' : <<Jambe_cassé, t , Jackie, 0>>

Ainsi, soit σ la situation dans le bois cet après-midi dans laquelle Jackie s'est cassé la jambe, il est possible d'exprimer cela en utilisant l'opération de conséquence logique $\models$: $\sigma \models a$ cela signifiant que σ supporte a. Autrement dit, σ satisfait la formule a.

2.6.2. Notion de situation

Une situation est un cas particulier des mondes possibles, dans le sens qu'une situation peut être décrite par un ensemble d'infons qui correspondent à une partie de la réalité avec laquelle on peut interagir. La principale différence entre la théorie des situations et des mondes possibles est que dans cette première, l'on doit assigner à chacune des propositions, une valeur de vérité dans chacun des mondes possibles. En d'autres termes, une proposition doit être nécessairement ou possiblement vraie ou fausse. A contrario, dans ce modèle de situations, un infon peut être non défini dans une situation à un instant donné. Formellement, pour une proposition a, $a \vee \neg a$ existe dans chacun des mondes possibles, mais les situations où a n'est pas définis, alors $a \vee \neg a$ n'existe pas. Il est à noter que toutes inférences, existantes pour la classe des situations, sont aussi valides pour la classe des mondes possibles.

Une situation est cohérente, si elle ne contient pas un infon et son dual[2] en même temps. Par ailleurs, une situation σ supporte un infon a (représenter par $\sigma \models a$) sous les conditions suivantes :

---

[2] Le dual d'un infon a est défini par $a^{\perp}$, la polarité de celui-ci étant l'inverse de a

a) $\sigma \models a$ si $a \in \sigma$ pour un infon a

b) $\sigma \models \neg a$ si $\sigma \nexists a$ (signifiant que a est rejeté par σ)

c) $\sigma \models a \wedge b$ si $\sigma \models a$ et $\sigma \models b$

d) $\sigma \models a \vee b$ si $\sigma \models a$ ou $\sigma \models b$

e) $\sigma \nexists a$ si $a^{\perp} \in \sigma$ pour un infon a

f) $\sigma \nexists \neg a$ si $\sigma \models a$

g) $\sigma \nexists a \wedge b$ si $\sigma \nexists a$ ou $\sigma \nexists b$

h) $\sigma \nexists a \vee b$ si $\sigma \nexists a$ et $\sigma \nexists b$

2.6.3. Opération de conséquences

Les situations fournissent une base sémantique pour les plans, en effet les inférences sur les situations permettent de considérer la reconnaissance de plan comme une opération déductive. Ceci étant représenté par un opérateur de conséquences monotonique $\vdash$, permettant de relier l'ensemble des prémisses (correspondant aux observations), avec l'ensemble des conséquences (correspondant aux prédictions). Il est possible d'admettre que pour une observation a et une prédiction b, de dire que $a \vdash b$, cela signifiant que b est présent dans chacune des réalisations de chacun des plans simples supportant a. il est à noter que nous définirons la notion de plan simple à la section 2.6.6.Dès lors, b est validement inféré à partir de a. Ainsi une opération de conséquence est une relation monotonique $\vdash$ entre un infon et un ensemble d'infons, qui satisfont les conditions suivantes. Soient Γ, Δ deux ensembles d'infons et a un infon tel que $\Gamma \vdash \Delta$ Signifie que $\Gamma \vdash a$ pour chaque $a \in \Delta$

1. Si ($\Gamma \vdash a$ et $\Gamma \subseteq \Delta$) alors $\Delta \vdash a$ (« Monotony »)
2. Si ($\Gamma \vdash \Delta$ et $\Gamma \cup \Delta \vdash a$) alors $\Gamma \vdash a$ (« Cut »)
3. Si ($\Gamma \vdash a$ et $\Gamma \vdash b$) alors $\Gamma \vdash a \wedge b$ (« And »)

2.6.4. Hiérarchie d'actions

Wobcke organise les plans de reconnaissance dans une structure hiérarchique en utilisant les relations d'abstraction et de décomposition sur un ensemble de types d'action. Ainsi un type d'action peut avoir des sous-types, par exemple *pâte* est le sous-type de *nourriture*. Nous pouvons prendre pour exemple la hiérarchie suivante, inspirée par celle de Kautz [Kautz, 1987] qui nous servira à illustrer la théorie.

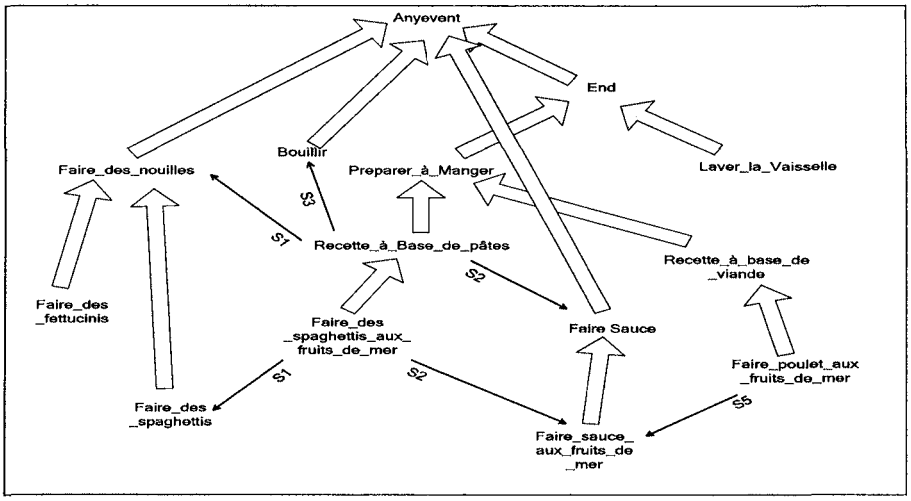


Figure 4 : Hiérarchie d'actions de cuisine modifiée.

Les flèches plus épaisses dénotent les relations d'abstraction (allant du spécifique vers le général) tandis que les flèches plus fines indiquent les relations de composition directe (allant d'un schéma de plan vers ses éléments le composant, ceux-ci ordonnés par étapes). Il est à noter la présence des actions END et ANYEVENT qui, chez Kautz, correspondent respectivement pour la première à l'action la plus générale de la hiérarchie et à l'action effectuant une abstraction des actions qui sont considérées, motivées par elle-même. Wobcke n'utilise pas ces deux actions, il définit un ensemble d'actions primaires et d'actions terminales.

2.6.4.1. Action primaire

Une action est dite une « action primaire » si elle est directement abstraite par l'événement END et qu'elle n'est pas le composant d'aucun schéma de plan. Dans la hiérarchie de la Figure 4, nous pouvons constater que seule l'action « *Preparer_manger* » est une action primaire.

2.6.4.2. Action terminale

Une action est dite une « action terminale » si elle ne possède aucun sous-type ou tous ses sous-types sont composants d'autres schémas de plan. Par exemple, dans la hiérarchie de la Figure 4, il est possible de dégager les actions terminales suivantes : *« Bouillir », «Faire_fettuccinis»,«Faire_spaghettis»,« Faire_spaghettis_fruits_de_mer»,« Faire_poulet _fruits_de_mer »,« Faire_sauce »,« Faire_Sauce_fruits_de _mer », « Laver_Vaisselle ».*

2.6.4.3. Hypothèse d'abstraction

Soit, *a'* et *a* deux actions, il définit la relation d'abstraction tel que :

(ABS) $a(i) \vdash a'(i)$

où a' est une abstraction directe de a, i étant un intervalle de temps quelconque. Pour représenter l'hypothèse d'abstraction, nous pouvons prendre l'exemple suivant grâce à la Figure 4 :

Faire_spaghettis_fruits_de_mer (i)⊢ Recette_base_pâtes (i)

2.6.4.4. Hypothèse de décomposition

La relation de décomposition signifie, que si une action *a(i)* est décomposable en ses éléments qui sont reliés par les contraintes de temps, elle peut être considéré par :

(DEC) $a(i) \vdash a_1(s_1(i)) \wedge \ldots \wedge a_n(s_n(i)) \wedge k$,

Où a_j correspond aux j^{eme} composant de a, s_1 ,...,s_j étant la fonction donnant l'étape j dans la décomposition, i étant un intervalle de temps quelconque, k étant la conjonction d'infons temporel relatif à i et $s_1(i),\ldots, s_n(i)$.

Ensuite, pour représenter l'hypothèse de décomposition, nous pouvons prendre l'exemple suivant :

Faire_spaghettis_fruits_de_mer (i) $\vdash$ *Faire_spaghettis*$(s_1(i)) \wedge$

Faire_sauce_fruits_de_mer$(s_2(i)) \wedge$ *Bouillir* $(s_3(i))$

2.6.4.5. Hypothèse de spécialisation

Pour deux actions a et b, si a est un rôle de spécialisation de b alors a est abstrait par b. Cette relation pouvant être perçue comme la relation inverse de la relation d'abstraction. Chaque rôle de spécialisation a_r d'une action a est une spécialisation de cette action, si une instance de a_r survient dans un plan, alors elle doit être aussi une instance de a.

(SPEC) $a_r(i) \vdash a(i)$,

pour chaque rôle de spécialisation a_r de a, i étant un intervalle de temps, a_r est abstrait par

b

Finalement, pour représenter l'hypothèse de spécification, nous pouvons prendre l'exemple suivant :

Recette_base_pâtes (i) ⊢ *Faire_spaghettis_fruits_de_mer (i)*

2.6.4.6. Hypothèse d'exhaustivité

Il est possible de définir l'hypothèse d'exhaustivité sur le fait que chaque plan simple contient exactement une action primaire sur un intervalle de temps précis. La notion de plan simple sera d'ailleurs abordée à la section 2.6.6.

(EXH) $\vdash XOR(a_1(i_0), a_n(i_0))$

où $a(i_0)$ est une action primaire et i_0 un intervalle de temps précis.

(EXH) $a_j(i_1) \wedge a_j(i_2) \vdash i_1 = i_2$ où a_j est une action primaire

Dans la Figure 4, nous pouvons considérer par exemple :

$\vdash XOR(Preparer\_manger(i_0), Laver\_vaisselle(i_0))$

2.6.4.7. Hypothèse de sous-types.

Un plan simple ne peut contenir qu'une instance d'un sous-type d'une action terminale ou alors de ses spécialisations.

(SUB) $a(i) \vdash XOR(a_1(i), \ldots, a_n(i), b_1^k(s_k(i)) \wedge \ldots \wedge b_{mk}^k(s_k(i))$,

pour une action non terminale a et chacun de ses composant b^k (k étant l'ordre du composant dans a), a_j étant les composants de a, b_i^k étant les composants de b^k qui ne sont composants d'aucun autre schéma de plan, i étant un intervalle de temps quelconque. Dans la Figure 4, nous pouvons considérer par exemple :

$$Faire\_nouilles(i) \vdash XOR(Faire\_spaghettis(i), Faire\_fettuccines(s1(i)))$$

2.6.4.8. Hypothèse de compositions

L'hypothèse de compositions est basée sur le principe que si une action est incluse dans un plan, alors au moins une instance du schéma de plan contenant cette action, doit être placée dans ce dit plan.

$$\text{(COMP) } a(i) \vdash \ldots \vee (a_j(p(i))=i)) \vee (s_{kj}(p(i))=i)) \vee \ldots$$

où l'action a_j est l'action la plus abstraite du schéma de plan, $p(i)$ donnant l'intervalle de l'occurrence de a, i étant un intervalle de temps quelconque. Dans la Figure 4, nous pouvons considérer par exemple :

$$Faire\_sauce(i) \vdash (Recette\_base\_pâtes(p(i)) \wedge (s_2(p(i)=i)) \vee Faire\_Poulet\_Marinara(p(i)) \wedge (s_5(p(i)=i))$$

2.6.5. Exemple d'application

Si nous considérons l'exemple employé par Kautz, les inférences suivantes peuvent être effectuées grâce aux hypothèses ci-dessus, soit i_1 un intervalle de temps quelconque :
L'agent observateur voit en i_1 l'action :

(1) $Faire\_sauce\_Marinara(i_1)$

Nous pouvons effectuer, à partir de cette action, les inférences suivantes :

(2) *Faire_spaghettis_fruits_de_mer* $(p(i_1)) \vee$ *Faire_Poulet_fruits_de_mer* $(p(i_1))$ Avec (1) et (COMP)

(3) *recette_base_Pates*$(p(i_1)) \vee$ *recette_base_viande*$(p(i_1))$ Avec (2) et (ABS)

(4) *Preparer_manger* $(p(i_1))$ Avec (3) et (ABS)

En outre de la manière que précédemment l'agent observateur voit en i_2 soit l'action fettucini ou spaghetti :

(5) *Faire_fettucinis*$(i_2) \vee$ *Faire_spaghettis*(i_2) par observation

(6) *Faire_nouilles*(i_2) avec (5) et (ABS).

(7) *Recette_base_pates*$(p(i_2))$ avec (6) et (COMP)

(8) *Preparer_manger*$(p(i_2))$ Avec (7) et (ABS)

(9) $p(i_1) = p(i_2)$ avec (4), (8) et (EXH)

(10) *Recette_base_pates*$(p(i_1))$ avec (7) et (9)

(11) *Preparer_manger* $(p(i_1))$ Avec (10) et (EXH)

(12) *Faire_spaghettis_fruits_de_mer* $(p(i_1)) \vee$ *Recette_base_viande*$(p(i_1))$ avec (2) et (ABS)

(13) *Faire_spaghettis_fruits_de_mer* $(p(i_1))$ avec (11) et (12)

(14) *Faire_spaghettis* $(s_1(p(i_1))$ avec (13) et (DEC)

2.6.6. Plan simple

Tout comme les modèles de couverture de la théorie de Kautz, un plan simple correspond à un plan que l'on peut former avec la hiérarchie et dont le plan observé en sera

une instance. Dès lors, un plan simple est un ensemble d'actions et d'infons temporel permettant d'expliquer une observation. Afin de le construire, il est nécessaire de partir d'une action primaire a et d'un intervalle de temps que l'on peut poser de manière arbitraire. Il est à noter que a' est l'action qui abstrait a_j, le $j^{ème}$ composant de a. Ainsi, l'infon $a(i_0)$ est tout d'abord ajouté, ensuite on placera $\neg\ a'(i_0)$ pour les autres actions primaires en respect de l'hypothèse d'exhaustivité et finalement $a'(s_j(i_0))$. Par répétition, on élaborera les actions non terminales, jusqu'à ce qu'il n'y a plus de relations d'abstraction. Pour élaborer une action non terminale t telle que $t(i)$ appartient au plan simple en construction, il est possible de soit spécialiser t ou alors de spécialiser chacun des composants non terminaux de t, en respect de l'hypothèse de spécification. Le processus s'arrêtant lorsqu'une action terminale est atteinte. Par ailleurs, pour qu'un plan simple soit bien formé celui-ci doit répondre à quatre critères suivants :

1) Si une action a possède des relations de décompositions vers des actions $a_1,\ldots, a_n$. a abstrait b, toute action b_i ajoutée au plan comme composant de b doivent être une spécialisation de a_i
2) Un composant d'un plan, par exemple que nous appellerons c ne peut pas être un composant d'un autre plan.
3) Uniquement les actions non terminales sont spécialisées.
4) Soit une action primaire est spécialisée ou alors tous ses composants non terminaux.

Avec la hiérarchie décrite par la Figure 4, nous pouvons considérer l'exemple de plan simple suivant :

{Preparer_manger, recette_base_pates, Faire_nouilles, Faire_fettucinis, Faire_sauce, Bouillir}

A contrario l'ensemble suivant n'est pas un plan simple, car l'action « *Faire_nouilles* » n'a pas été spécialisée :

{Preparer_manger, recette_base_pates, Faire_nouilles, Faire_sauce, Bouillir}

2.6.7. Relations d'ordre sur les plans simples

Afin d'exprimer la reconnaissance de plan comme un cas particulier de conséquences non-monotonique, Wobcke propose de travailler sur une structure ordonnée des plans qui permettra de remettre en question les prédictions. Tout d'abord, celui-ci construit l'ensemble des plans simples que l'on peut exprimer grâce à la hiérarchie. Puis il les ordonne en fonction de leur plausibilité afin de dégager le plan le plus plausible pour une observation. Ceci s'effectuant grâce à l'opérateur $\preceq$, qui peut être défini une relation d'ordre partiel presque connecté. Ce type de relation, sur un ensemble S est une relation binaire qui satisfait les conditions suivantes :

(Pour tout $\alpha, \beta, \gamma \in S$)

1. Réflexivité, tel que, $\alpha \preceq \alpha$
2. Antisymétrie, tel que $\alpha \preceq \beta$ et $\beta \preceq \alpha$ implique que $\alpha = \beta$
3. Transitivité, tel que $\alpha \preceq \beta$ et $\beta \preceq \gamma$ implique que $\alpha = \gamma$
4. Presque connecté tel que $\alpha \preceq \beta$ implique $\alpha = \beta$, $\alpha \preceq \gamma$ ou $\gamma \preceq \beta$

Finalement, les observations font que les prédictions sur le plan, le plus plausible, sont remises en question, afin de reconnaître le plan de l'agent acteur. Nous pouvons considérer l'exemple suivant, récapitulant les différentes étapes. Tout d'abord, il est nécessaire d'exprimer la liste des plans simples que l'on peut tirer de la hiérarchie. Puis de l'ordonner grâce à la relation d'ordre $\preceq$, pour ensuite effectuer les inférences nécessaires. A titre d'exemple, voici la liste des plans simples que l'on peut tirer et ordonner de la Figure 4.

« Preparer_manger », « Recette_base_viande », « Faire_poulet_fruits_de_mer »,

« Faire_sauce », « Faire_sauce_fruits_de_mer »

$\preceq$

« Preparer_manger », « Recette_base_pates », « Faire_spaghettis_fruits_de_mer », « Faire_nouilles », « Faire_spaghettis », « Faire_sauce », « Faire_sauce_fruits_de_mer », « Bouillir »

« Preparer_manger », « Recette_base_pates », « Faire_nouilles », « Faire_fettucinis »,

« Faire_sauce », « Bouillir »

$\preceq$

« Laver_la_vaisselle »

L'agent observateur croit initialement que l'agent acteur effectue *« Faire_poulet_fruits_de_mer »*. Ce plan étant le plus probable, de là on peut prédire les actions *« Faire_sauce_fruits_de_mer »*, *« Faire_sauce »*, *« Recette_base_viande »* et *« Preparer_manger »*. En observant l'action *« Faire_fettucinis »*, il lui sera possible de réviser ses croyances grâce à l'opérateur de révision. Ainsi en abandonnant les convictions en *« Faire_poulet_fruits_de_mer »*,*« Recette_base_viande »*,

« *Faire_sauce_fruits_de_mer* » et ne gardant que « *Faire_sauce* », « *Preparer_manger* », la prédiction la plus probable devient dès lors, le plan « *Faire_nouilles* ».

2.7. Bilan sur l'approche de Wobcke

Wobcke emploie la théorie des situations afin de proposer une théorie pour la reconnaissance de plan, basé sur la logique non monotone. En effet grâce aux hypothèses qu'il propose tel que les hypothèses d'exhaustivité, d'abstraction, de spécialisation, etc.. il est possible d'exprimer les plans de la hiérarchie sous forme d'un ensemble de plan simple. Donc, l'ordonnancement permet, grâce à la logique non monotone, d'effectuer des prédictions sur les actions futures de l'agent acteur. Un des avantages de la théorie de Wobcke est que celui-ci définit et caractérise les plans simples qui sont les plans reconnus dans le processus de la reconnaissance de plan. Par ailleurs, la structure hiérarchique définit par Wobcke devant être cohérente, mais n'est pas obligatoirement complète. En effet, un plan exécuté par l'agent acteur, comme « *Faire_fettucinis* » peut très bien être reconnu, alors que dans la théorie de Kautz [Kautz, 1987] celui-ci ne le serait pas, car il est nécessaire d'avoir une action dont « *Faire_fettucinis* » en serait le composant.

De plus, la théorie de Wobcke évite certaines difficultés de la théorie de Kautz, telle que la minimisation des modèles de couverture. Ceci grâce au fait que Wobcke pose que l'agent acteur ne va pas effectuer deux plans séparés en même temps. Il n'est donc plus nécessaire de chercher à minimiser les modèles de couvertures. Ceci formant d'ailleurs une limite de la théorie de Wobcke, car un agent acteur peut être amener à faire deux ou plusieurs plans

en même temps. Il est à noter que dans des modèles tel que ceux à base du raisonnement hypothétique, cette capacité à effectuer deux plans est prise en compte, tout en dégageant le plan le plus probable. Ce point étant un des éléments moteurs de la reconnaissance de plan à base de réseaux bayésiens [Pynadath *et al*, 1995] ou de l'utilisation de la théorie de Dempster-Shafter [Carberry, 1990]. Toutefois, une autre limite de la théorie de Wobcke est la complexité du formalisme proposé par le fait qu'il est difficile de le rendre opérationnel dans un cas concret.

2.8. Conclusion

Au cours de ce chapitre, nous avons défini informellement la notion de reconnaissance de plan et les paradigmes qui s'y rattachent. Ensuite nous avons passé en revue, quelques modèles de la reconnaissance de plan tel que celui de Kautz, qui à partir d'une hiérarchie d'événements et de quelques hypothèses, construit une explication des actions de l'acteur. Puis nous avons vu les propositions de Wobcke pour la reconnaissance de plan, qui grâce à la théorie des situations exprime une catégorie de plan particulière, les plans simples tirés de la hiérarchie. Des opérations non monotonique tel que la révision, permettent de remettre en question, la croyance que l'agent acteur effectue un plan simple plutôt qu'un autre.

Que ce soit pour expliquer les actions de l'acteur ou de prendre en entrée une séquence d'action, il est évident que la notion d'action est un des éléments essentiels de la reconnaissance de plan. Nous souhaitons employer la logique de description dans nos

travaux, qui est un formalisme de représentation de la connaissance beaucoup plus souple que celui de la théorie des situations. Dès lors une des problématiques est comment formaliser l'action dans cette logique, pour répondre aux limites décrites précédemment. Par ailleurs, formaliser l'action est une chose, mais il est nécessaire d'avoir un langage permettant d'utiliser cette notion d'action pour la reconnaissance de plan. Ce sont donc des points que nous tenterons de répondre dans le dernier chapitre.

Pour cela, nous avons proposé un chapitre concernant la logique de description pour, non seulement permettre la lecture de ce mémoire mais aussi, le fait qu'il n'existe pas de document à grand public, synthétisant ce domaine et facilitant par là même sa familiarisation.

CHAPITRE 3

LA LOGIQUE DE DESCRIPTION (LD)

3.1. Introduction

La recherche dans le domaine de la représentation de la connaissance et du raisonnement est généralement basée sur des méthodes pour fournir des descriptions de hauts niveaux du monde, que l'on peut employer pour construire des systèmes intelligents. Dans ce contexte, « intelligent » référa à la faculté d'un système de trouver des conséquences implicites à travers une représentation explicite de la connaissance [Nardi, Brachman, 2002]. Au cours de cette introduction, nous allons faire un bref survol des éléments précurseurs à la création de la logique de description (LD), pierre angulaire de nos travaux et pour certains chercheurs, de la représentation de la connaissance [Baader et Nutt, 2002]. En outre, les connaissances désignent l'ensemble des informations (savoir, savoir-faire, expériences, souvenirs, concepts, faits...) nécessaires à un être humain (ou à une machine) de manière à ce qu'il puisse accomplir une tâche considérée comme complexe. Par exemple, diagnostiquer une maladie, résoudre un problème de mathématique, sont des activités, qui, lorsqu'elles sont réalisées par des êtres humains, réclament une grande quantité de compétences et d'informations organisées pour un but précis [Ferber, 1995].

De plus, la représentation de la connaissance peut être perçue fondamentalement comme un substitut à l'objet représenté, qui sera employé, par une entité, afin de déterminer différentes conséquences grâce au raisonnement sur le monde, plutôt que d'agir sur celui-ci. En somme, si par exemple, nous avons les informations suivantes, qu'il pleut, la connaissance que la pluie est de l'eau, que l'eau mouille, cela nous permet, donc, par un

rapide raisonnement d'en déduire que si nous sortons, nous serons mouillés ; ceci sans avoir était dehors pour le vérifier [Davis *et al*, 1993]. Ainsi, il est possible de diviser la représentation de la connaissance en deux catégories, une ayant un formalisme basé sur la logique qui s'appuie sur l'intuition que la logique des prédicats peut représenter au mieux les faits constituant le monde ; et une autre, basée sur une autre approche non logique de la représentation. Dans une abstraction basée sur la logique, le langage de représentation est souvent utilisé comme une variante du calcul des prédicats du premier ordre, ses raisonnements étant surtout des vérifications sur les conséquences logiques entre ces prédicats. Dans une approche non logique, souvent basée sur l'emploi d'interfaces graphiques, la connaissance est représentée au moyen de structure de données, les plus adéquates possibles et les raisonnements sont accomplis par des procédures manipulant ses structures de manière, là aussi adéquates [Nardi, Brachman, 2002].

Parmi ses représentations, il y a les réseaux sémantiques [Quillian, 1967] et les cadres (frames) [Minsky, 1981]. Les réseaux sémantiques ayant pour but de caractériser au moyen de structure cognitive sous la forme de réseaux, la connaissance et le raisonnement du système. Les cadres reposent sur la notion de prototypes. Cette notion, tirée de la psychologie, consiste à créer une généralisation basée sur un objet précis et typique de la famille de celui-ci. Le manque de sémantique, précise dans ses systèmes, développa l'intérêt sur l'emploi d'une structure hiérarchique permettant une meilleure représentation, mais aussi une meilleure efficience. En outre, il est possible de donner une sémantique aux cadres en employant la logique du premier ordre [Hayes, 1980]. En effet, les éléments de

base de la représentation peuvent être caractérisés par des prédicats unaires, dénotant un ensemble d'individus. Les relations entre ces individus pouvant être représentées par des prédicats binaires. De plus, la logique est une base naturelle pour spécifier un sens pour ces structures, il est évident que pour la plupart des réseaux sémantiques et des cadres ne requiert pas l'application de toute la logique du premier ordre, mais seulement un fragment de celle-ci [Brachman et levesque, 1985].

KL-ONE fruit des travaux de Brachman et Levesque, en 1985, apporta la notion de rôle et de concept et comment ceux-ci peuvent être interreliés. Ce système se basant sur les réseaux sémantiques, se plaça comme l'ancêtre de la logique de description, logique que nous détaillerons plus particulièrement, car elle est au cœur de nos travaux. Par ailleurs, celle-ci, apporte une sémantique formelle, mais aussi des inférences comme la classification de concept et d'individu. La classification de concept détermine une relation entre les sous-concepts et les super-concepts, appelés relation de subsomption. Celle-ci permettant de structurer la terminologie dans une forme de hiérarchie ordonnée par la subsomption. La classification des individus permet que pour un individu précis, celui-ci soit toujours une instance d'un certain concept [Baader et Nutt, 2002].

Nous venons d'effectuer un bref survol de l'historique de la représentation de la connaissance qui permit à la logique de description de voir le jour. Dès lors, nous allons décrire celle-ci de manière informelle, en incluant les concepts de base sur lesquels repose ce formalisme ainsi que les différentes notations de celle-ci. Puis, nous présenterons

l'architecture d'un Système de représentation de la connaissance terminologique (SRCT), ainsi que l'un d'entre eux, LOOM.

3.2. Description de la logique de description

On peut définir la logique de description comme étant un formalisme de représentation des connaissances se divisant en deux parties bien distinctes : la partie terminologique, qui permet de définir un ensemble de concepts ainsi que leurs rôles et la partie assertionnelle, qui permet d'une part la création d'individus appartenant aux différents concepts de la partie terminologique et d'autre part aussi les opérations de classification et d'inférence. Par ailleurs, il est à noter que la classification, qui constitue la base du raisonnement en logique de description, s'appuie sur la relation de subsomption qui permet d'organiser les concepts et les rôles en une hiérarchie de termes.

3.2.1. Eléments de base de la logique de description

La logique de description peut être définie comme la conjonction d'un langage terminologique et d'un langage assertionnel. Un langage terminologique peut à son tour être défini comme étant un ensemble d'opérateurs (appelés des constructeurs) et de symboles qui permettent la définition de concepts et de rôles. Par exemple, le langage terminologique *TF*, que nous utiliserons pour les démonstrations tout au long du chapitre, peut être défini comme suit [Nebel, 1995]:

$$TF = < \top, \bot, \doteq, \leq \textit{ disjoint, and, all, atleast, atmost} >$$

où $\{T, \bot, \doteq, \leq\}$ sont des symboles et { *disjoint, and, all, atleast, atmost* } sont des opérateurs. Le langage *TF* sera décrit de façon plus explicite à la section 3.3.2.

Un langage assertionnel peut être défini comme étant un ensemble de symboles et d'opérateurs permettant la création d'instances issues de concepts et de rôles décrits par un langage terminologique. Par exemple, le langage terminologique *AF*, qui est le complément assertionnel de *TF*, se définie comme suit :

$$AF = < c, r, \geq, \leq \text{ atleast, atmost}>$$

où $\{\geq, \leq\}$ sont des symboles et *{c, r, atleast, atmost}* sont des opérateurs. Il est à noter que *c* et *r* représentent ici les noms de deux termes désignant respectivement un concept *c* et un rôle *r* décrit dans la terminologie. Les opérateurs et les symboles du langage *AF* seront décrits de façon plus explicite à la section 3.3.2.

3.2.1.1. Notion de concept

D'un point de vue ensembliste, on peut voir un concept comme étant une série de définitions qui permettent la création d'un ensemble en spécifiant les conditions nécessaires d'appartenance à celui-ci. Un concept[3] en logique de description peut être vu comme une conjonction de caractéristiques qui sont définies par ses rôles. En fait, il existe deux types de concepts en logique de description :

[3] Habituellement, chaque mot désignant le nom d'un concept commence par une lettre majuscule

- **Concept primitif** : les concepts primitifs servent de base à la construction de concepts définis. Ce type de concept représente le niveau le plus atomique de la connaissance en logique de description. Ces concepts peuvent posséder certains rôles qui les définissent, mais leurs descriptions sont incomplètes et représentent des conditions nécessaires mais insuffisantes pour déterminer le type de l'individu : il n'est pas possible de reconnaître un représentant d'un concept primitif à la vue de ses seuls rôles [Haton *et al*, 1991]. Ainsi, si un concept nommé « Personne » et un autre nommé « Animal » sont tous deux définis avec le rôle « date-de-naissance », il sera vrai de dire que toutes les instances du concept « Personne » auront une « date-de-naissance », mais l'inverse ne sera pas vrai.

- **Concept défini** : les concepts définis sont construits à partir des rôles qu'ils entretiennent avec des concepts primitifs ou avec d'autres concepts définis. Ce type de concept représente les niveaux de connaissances les plus complexes. Contrairement aux concepts primitifs, les concepts définis possèdent une définition complète et suffisante pour en déduire l'appartenance certaine d'une instance [Haton *et al*, 1991]. Si l'on prend, par exemple, un concept nommé « Parent » qui entretient un rôle nommé « enfant », il sera possible d'affirmer qu'une instance de type « Parent » aura automatiquement un ou des « enfant » et qu'une instance ayant un ou des « enfant », appartiendra automatiquement au concept « Parent ».

3.2.1.2. Notion de rôle

Un rôle[4] représente une relation binaire entre deux concepts. C'est une relation qui a pour fonction de caractériser le premier concept en précisant sa relation avec le second. Le concept non caractérisé par le rôle est appelé le co-domaine. Des restrictions de cardinalité peuvent être attribuées aux rôles. Par exemple, le rôle « enfant », caractérisant le concept « Parent », devrait être défini avec une cardinalité $n \geq 1$.

3.2.1.3. Relation entre les rôles

Une relation entre deux rôles est un lien servant à préciser un rôle à l'aide d'un autre. Par exemple, une relation pourrait être introduite entre un rôle « date-de-naissance » et un rôle « date-de-naissance-humain » afin de préciser que le deuxième est une spécialisation du premier. On peut également vouloir préciser que deux rôles sont complètement différents, même s'ils définissent le même concept et ont le même co-domaine. Ainsi, un rôle nommé « père-de » pourrait entretenir un lien avec un rôle « mère-de » afin de le différencier de celui-ci. Il est à noter qu'on ne peut pas attribuer de restrictions sur la cardinalité de ces relations.

3.2.2. Création de concepts et de rôles

Beaucoup de langages terminologiques ont été construits à partir des éléments de base vus précédemment. Nous allons, d'ailleurs, illustrer le fonctionnement de ceux-ci par l'un d'entre eux à savoir le langage terminologique TF défini par Nebel [Nebel, 1995]. Cette section du chapitre présentera, ainsi, une description sur la construction de concepts et de

[4] Habituellement, le nom d'un rôle est écrit entièrement en lettres minuscules

rôles en utilisant le formalisme de TF. Ce langage est très proche de celui développé par Brachman et Schmolze pour KL-ONE [Brachman et Schmolze, 1985].

3.2.2.1. Construction de concepts avec le langage TF

En logique de description, un *terme* désigne le nom d'un concept ou d'un rôle. La terminologie du langage TF est principalement basée sur l'introduction de nouveaux termes et sur l'opération de disjonction. Avant d'aller plus loin dans les explications, il faut d'abord introduire quelques symboles qui serviront tout au long de la présentation.

C_i	=	*Symbole qui représente le terme d'un concept*
R_i	=	*Symbole qui représente le terme d'un rôle*
$\top$	=	*Symbole représentant le concept et le rôle maximal, le « tout »*
$\bot$	=	*Symbole représentant le concept et le rôle minimal, le « rien »*
$\leq$	=	*Symbole servant à définir un concept ou un rôle primitif*
$\doteq$	=	*Symbole servant à définir un concept ou un rôle complexe*

En langage TF, un concept peut être défini de trois façons différentes en fonction de sa nature. Premièrement, il est possible de définir un concept primitif qui ne possède aucune caractéristique et qui servira de base à la création de concepts plus complexes. Ce type de concept doit être défini comme un descendant du concept maximal $\top$. Un concept primitif peut être défini selon sa position par rapport aux autres concepts de la terminologie en spécifiant la combinaison de concepts qui le compose. Finalement, on peut définir un concept complexe à l'aide d'opérateurs qui seront appliqués sur des concepts déjà existants

afin d'en définir ses caractéristiques. Voici les trois différentes notations pour la définition de concepts en TF :

<Concept primitif> $\leq$ T

<Concept primitif> $\leq$ *<concept ou combinaison de concepts>*

<Concept complexe> $\doteq$ *<concept ou combinaison de concepts>*

3.2.2.2. Opérations sur les concepts

Les opérateurs servent spécifiquement à définir les concepts, les combinaisons de concepts et les rôles dans la terminologie. Plus les opérateurs d'un langage permettront de définir de façon précise les concepts et les rôles, plus ce langage pourra représenter un grand nombre de réalités. Il est possible de créer des concepts complexes à l'aide de certains opérateurs (qui sont également appelés des constructeurs). Il est à noter qu'il y a des différences marquées en ce qui concerne les opérateurs des différents langages terminologiques, notamment au niveau du nombre disponible. Par exemple, le langage TF ne possède pas d'opérateur de négation ni d'opérateur existentiel, ce qui limite la flexibilité du langage. La notation habituelle qui est utilisée pour la définition de ces opérateurs est très proche du langage Lisp (Common Lisp). En fait, il est possible de décrire les opérateurs sous forme concrète (Lisp) et sous forme abstraite (logique classique). Voici la liste des opérateurs disponibles en langage TF :

a) **Disjoint** : Cet opérateur particulier sert à introduire des disjonctions entre deux concepts. Il peut également être utilisé avec les rôles. Il a pour fonction de signaler

que certains concepts ou rôles sont opposés, par exemple les concepts « Homme » et « Femme ». L'utilisation de cet opérateur de disjonction se fait de la façon suivante :

Forme concrète : *(disjoint <Concept C_1> <Concept C_2>)*

Forme abstraite : $C_1 \sqcup C_2$

Exemple TF : *(disjoint Homme Femme)*

b) **AND** : Cet opérateur représente le « ET » logique. Il est utilisé pour créer une conjonction de plusieurs concepts. Par exemple, la création d'un concept « Dauphin » pourrait nécessiter la conjonction entre le concept « Animal-Marin » et « Mammifère ». La notation utilisée pour cet opérateur est la suivante :

Forme concrète : *(AND C_1 ... C_n)*

Forme abstraite : $C_1 \sqcap \ldots \sqcap Cn$

Exemple TF : *Dauphin $\doteq$ (AND Animal-Marin Mammifère)*

c) **ALL** : Cet opérateur exprime la restriction de valeur. En langage TF, il a pour fonction de restreindre la nature du co-domaine d'un rôle donné. Prenons, par exemple, un rôle R_1 dénoté $R_1(C_1, C_2)$. Il serait possible, grâce à l'opérateur *ALL*, de définir la nature exacte du concept C_2 de la relation. En d'autres termes, on pourrait utiliser cet opérateur dans la définition d'un concept « Dauphin » pour limiter le co-

domaine d'un rôle nommé « mange » aux instances de concepts « Nourriture ». La notation de cet opérateur est la suivante :

Forme concrète : *(ALL R C)*

Forme abstraite : $\forall R : C$

Exemple TF : *Dauphin ≐ (AND Animal-Marin Mammifère ALL mange Nourriture)*

d) **ATLEAST** : Cet opérateur permet d'effectuer une restriction de cardinalité minimum sur un rôle donné. Il sert à définir une borne inférieure sur un rôle. Par exemple, un rôle « enfant » d'un concept « Parent » pourrait se voir imposer une cardinalité minimum de 1. Cela signifie qu'une instance du concept « Parent » devrait au moins entretenir un rôle « enfant » avec une autre instance. Cet opérateur est noté de la manière suivante:

Forme concrète : *(ATLEAST n R)*, *n* est un entier supérieur à 0.

Forme abstraite : $\geq n\ R$

Exemple TF : *Parent ≐ (AND Humain ATLEAST 1 enfant)*

e) **ATMOST** : Cet opérateur permet d'effectuer une restriction de cardinalité maximale sur un rôle donné. Il sert à définir une borne supérieure sur un rôle. En utilisant cet opérateur et l'opérateur *ATLEAST*, il est possible de définir complètement le domaine des valeurs permises pour un rôle. Par exemple, un

concept « Personne » pourrait se voir imposer une cardinalité maximale de 2 sur un rôle nommé « géniteur ». Cet opérateur utilise la notation suivante :

Forme concrète : *(ATMOST n R)*, *n* est un entier supérieur à 0.

Forme abstraite : $\leq n\ R$

Exemple TF : *Personne* $\doteq$ *(AND Humain ATMOST 2 géniteur)*

3.2.2.3. Construction de rôles

Les rôles occupent une place particulièrement importante en logique de description. Ils servent à caractériser les concepts et permettent de définir les relations qu'ils entretiennent les uns avec les autres. Tout comme les concepts, il existe en langage TF trois façons de définir un rôle. La première consiste à définir un rôle de base qui n'est la spécialisation d'aucun autre rôle existant. Ce type de rôle devra être issu du rôle maximal *T*. Par exemple, on pourrait définir un rôle nommé « mère » comme étant issu de *T*. La seconde consiste à définir un rôle comme étant une spécialisation d'un autre rôle existant. Par conséquent, cette action définit également une relation de subsomption entre ces deux rôles. Par exemple, on pourrait définir un rôle « mère-célibataire » comme étant une spécialisation du rôle « mère ». La dernière façon consiste à créer un rôle qui est équivalent à un autre. En fait, il s'agit de donner un deuxième nom à un rôle qui existe déjà. On pourrait, par exemple, définir un rôle nommé « maman » et spécifier qu'il est l'équivalent du rôle « mère ». Une grande partie des langages terminologiques permet également la conception de rôles complexes qui peuvent être introduits à l'aide de conjonctions de plusieurs rôles ou via d'autres opérateurs. Cependant, le langage TF ne permet pas la création de rôles

complexes. Il est tout de même suffisamment complet pour modéliser un grand nombre de situations [Nebel, 1995]. Voici les trois notations disponibles en langage TF pour l'introduction de nouveaux rôles :

<Rôle atomique>	$\leq$	T
<Rôle atomique>	$\leq$	*<Rôle atomique existant>*
<Rôle atomique>	$\doteq$	*<Rôle atomique existant>*

Exemples TF :

mère	$\leq$	T
mère-célibataire	$\leq$	*mère*
maman	$\doteq$	*mère*

La Figure 5 nous montre la syntaxe complète du langage TF décrit par Nebel [Nebel, 1995]. Toutes les notions peuvent être décrites de façon compacte en utilisant le format BNF. Cette notation est semblable à celle utilisée pour la description des grammaires hors-contexte en théorie des automates. Il est à noter ici que le langage TF ne représente que la partie terminologique d'un formalisme de description ; la partie assertionnelle associée au langage TF s'appelle le langage AF. Cette partie assertionnelle sert à l'introduction et à la manipulation d'individus issus des concepts et des rôles décrits par la partie terminologique. La conjonction des deux langages constitue un formalisme terminologique à part entière.

Syntaxe de TF

⟨*terminologie*⟩	::=	{ ⟨*termeIntroduction*⟩ | ⟨*restriction*⟩ }
⟨*termeIntroduction*⟩	::=	⟨*conceptIntroduction*⟩ | ⟨*roleIntroduction*⟩
⟨*conceptIntroduction*⟩	::=	⟨*atomicConcept*⟩ ≐ ⟨*concept*⟩ | ⟨*atomicConcept*⟩ ≤ ⟨*concept*⟩ | ⟨*atomicConcept*⟩ ≤ *ANYTHING*
⟨*roleIntroduction*⟩	::=	⟨*atomicRole*⟩ ≐ ⟨*role*⟩ | ⟨*atomicRole*⟩ ≤ ⟨*role*⟩ | ⟨*atomicRole*⟩ ≤ *ANYTHING*
⟨*concept*⟩	::=	⟨*atomicConcept*⟩ | (*AND* ⟨*concept*⟩+) | (*ALL* ⟨*role*⟩ ⟨*concept*⟩) | (*ATLEAST* ⟨*nombre*⟩⟨*role*⟩) | (*ATMOST* ⟨*nombre*⟩⟨*role*⟩)
⟨*role*⟩	::=	⟨*atomicRole*⟩
⟨*restriction*⟩	::=	(*DISJOINT* ⟨*atomicConcept*⟩⟨*atomicConcept*⟩)
⟨*nombre*⟩	::=	⟨*entierNonNegatif*⟩
⟨*atomicRole*⟩	::=	⟨*identifieur*⟩
⟨*atomicConcept*⟩	::=	⟨*identifieur*⟩

Figure 5 : Syntaxe du langage TF (format BNF).

3.2.3. Sémantique en logique de description

A l'instar de la logique classique, une sémantique est associée aux descriptions de concepts et de rôle en logique de description. Les concepts sont interprétés comme des sous-ensembles d'un domaine d'interprétation et les rôles comme des sous-ensembles d'un produit entre deux domaines d'interprétations [Napoli, 1997].

3.2.3.1. Notion d'interprétation

Nous allons décrire la notion d'interprétation, symbolisée par I, en s'inspirant d'articles tels que ceux de Baader et Hollunder [Baader et Hollunder, 1991] [Baader *et al*, 1991]. Pour la définir, il faut se donner un *domaine sémantique* Δ^I, qui correspond à l'ensemble de

tous les individus de la terminologie étudiée (pas seulement ceux qui sont effectivement existants) et une fonction $(.)^I$ appelée *fonction d'interprétation*, dont le rôle est d'associer chacun des termes désignant un concept à un sous-ensemble de Δ^I et chaque nom de rôle à un sous-ensemble de $\Delta^I \times \Delta^I$. Il est à noter que la *fonction d'interprétation* s'utilise en respectant la terminologie introduite par le langage terminologique sous-jacent. Prenons, par exemple, les concepts suivants : {Chaise, Dossier, Pied} et le rôle « partie » qui relie une composante d'un meuble à celui-ci. Le domaine sémantique Δ^I sera l'ensemble de tous les individus issus des concepts « Chaise », « Dossier » et « Pied » ainsi que toutes les parties possibles (respectivement c_i, d_i, p_i pour représenter les concepts et des couples de ceux-ci pour représenter le rôle « partie », en respectant sa définition). Donc, en appliquant une fonction d'interprétation, on aurait, par exemple :

$$Chaise^I = \{ c_1, c_2, c_3, c_4 \}$$
$$Dossier^I = \{ d_1, d_2, d_3 \}$$
$$Pied^I = \{ p_1, p_2, p_3, p_4 \}$$
$$partie^I = \{ (c_1, d_1), (c_1, p_1), (c_1, p_2), (c_1, p_3), (c_1, p_4), (c_3, d_2) \}$$

La notion d'interprétation signifie littéralement que l'on interprète les concepts et les rôles donnés par une terminologie. Il peut donc y avoir plusieurs interprétations I possibles pour une terminologie donnée.

3.2.3.2. Illustration ensembliste de TF

Il est possible de réécrire tous les opérateurs d'un langage terminologique avec un symbolisme ensembliste. C'est ce qu'on appelle la sémantique d'un langage

terminologique. On distingue souvent une terminologie de son interprétation en utilisant les termes *définition par intention* et *définition par extension* pour l'interprétation. Voici, à titre d'exemple, la sémantique du langage TF, sous le format « *forme concrète* = *forme abstraite* = *sémantique* », où C et D désignent deux concepts :

$(le \ll tout \gg)^I$	=	T^I	=	Δ^I
$(le \ll rien \gg)^I$	=	$\bot^I$	=	$\varnothing$
$(AND)^I$	=	$(C \sqcap D)^I$	=	$C^I \cap D^I$
$(ALL)^I$	=	$(\forall R : C)^I$	=	$\{d \in \Delta^I \mid (d, e) \in R^I \rightarrow e \in C^I, \forall e\}$
$(ATLEAST)^I$	=	$(\geq n R)^I$	=	$\{d \in \Delta^I \mid \#^5 \{ e \mid (d, e) \in R^I\} \geq n\}$
$(ATMOST)^I$	=	$(\leq n R)^I$	=	$\{d \in \Delta^I \mid \# \{ e \mid (d, e) \in R^I\} \leq n\}$

La définition d'un concept A comme étant un sous-concept de B, c'est-à-dire $A \leq B$, s'écrit sémantiquement avec le symbole de l'inclusion des ensembles $A \subseteq B$. Quant à la définition d'un concept A à l'aide du symbole $\doteq$, elle est équivalente à la création d'un nouvel ensemble $A \subseteq \Delta^I$. Ceci s'applique également aux rôles, mais en sachant qu'un rôle est une fonction dans $\Delta^I \times \Delta^I$. À partir de là, il est tout à fait possible de présenter graphiquement la sémantique d'un langage terminologique, puisqu'elle est uniquement composée d'opérations sur des ensembles. Par conséquent, une autre façon de visualiser une logique de description est par l'utilisation des ensembles. Il suffit d'appliquer les opérations ensemblistes : l'intersection pour le *AND*, l'union pour le *OR* (non disponible en

[5] Dans le cas présent, le symbole # désigne la cardinalité d'un ensemble.

TF, mais il se retrouve dans plusieurs autres langages terminologiques), etc. Cette équivalence ensembliste s'applique également à la relation de subsomption. Par conséquent, une relation de subsomption, entre un concept *A* et un concept *B*, peut être présentée comme l'inclusion d'un ensemble *A* dans un ensemble *B* ou vice-versa.

3.2.4. Notion de subsomption

La relation de subsomption permet d'organiser les concepts et les rôles par niveaux de généralité. Intuitivement, un concept *A* subsume un concept *B* si *A* (le subsumant) est plus général que *B* (le subsumé). Cette réalité peut être exprimée de trois différentes façons :

a) **Définition ensembliste en extension :** un concept *A* subsume un concept *B* si et seulement si l'ensemble des représentants dénoté par *A* contient nécessairement l'ensemble des représentants dénoté par *B* [Levesque et Brachman, 1987]. Dans cette définition, chaque concept est représenté par un ensemble et chacune des relations de subsomption est représentée par une inclusion ensembliste. Par exemple, le concept qui dénote les mammifères subsume celui dénotant les chats.

b) **Définition ensembliste en intention** : Un concept *A* subsume un concept *B* si toute entité représentée par *B* l'est aussi par *A*. Ainsi, chaque entité dont la description est définie par *B* possède les caractéristiques qui sont spécifiées par *A* [Finin, 1986]. Par exemple, le concept dénotant les chats possède toutes les caractéristiques de celui qui dénote les mammifères en plus de celles qui sont propres aux chats. Un concept

se compose donc d'une description propre à lui-même et d'une description « partagée » ou « commune » avec ses subsumants.

c) **Définition logique** : Un concept A subsume un concept B si un élément qui est décrit par B implique logiquement que cet élément est décrit par A. Ainsi, l'implication $(Chat(x) \Rightarrow Mammifère(x))$ est vraie pour tout x [McGregor, 1988].

En utilisant la notion d'interprétation, on peut définir la relation de subsomption de la manière suivante : un concept B est subsumé par un concept A (respectivement A subsume B), ce qui se note $B \subseteq A$, si et seulement si $B^I \subseteq A^I$ pour toute interprétation I.

La subsomption est à la base de la classification dans la logique de description. C'est une relation d'ordre partiel qui permet d'organiser les concepts en une hiérarchie. Chaque concept possède une description qui permet de le comparer aux autres et ainsi de déterminer les relations de subsomption qu'il entretient avec les autres concepts. La subsomption est une relation réflexive ; un concept A est donc subsumé par lui-même. Cette relation est également transitive, ce qui signifie que si un concept A subsume un concept B et que ce concept B subsume un concept C, alors le concept A subsume automatiquement le concept C. La relation de subsomption est aussi antisymétrique, ce qui signifie que si un concept A subsume un concept B et que le concept B subsume à son tour le concept A, alors $A = B$. Les différentes relations de subsomption qu'entretient un concept permettront de déduire l'emplacement que ce concept devrait occuper dans la hiérarchie. Un concept

maximal, représentant la totalité du domaine d'interprétation, est présent au sommet de la hiérarchie. Ce concept représente l'ensemble le plus général et subsume tous les autres, permettant ainsi d'obtenir un point de départ pour la taxonomie. Inversement, un concept minimal est présent au bas de la hiérarchie. Ce concept est subsumé par tous les autres ; il représente en fait le vide ou le « rien ».

Prenons, par exemple, les concepts « Animal », « Mammifère », « Ovipare », « Reptile » et « Ornithorynque », tels que représentés de façon ensembliste sur la Figure 6. On peut voir que le concept « Ovipare » subsume le concept « Reptile », car le fait d'être un « Reptile » implique le fait d'être un « Ovipare ». On peut également noter, sur la même figure, que le concept « Ornithorynque » est subsumé par le concept « Mammifère » et par le concept « Ovipare », mais de façon plus explicite, par la conjonction des deux concepts. Le fait d'être un « Ornithorynque » implique obligatoirement le fait d'être un « Mammifère » et un « Ovipare ». Chacun des concepts est donc délimité par l'ensemble des individus qu'il représente par rapport aux autres concepts. On peut donc en conclure que la classification d'un nouveau concept se réduit à déterminer où devra se situer l'ensemble qu'il représente par rapport aux autres ensembles déjà existants.

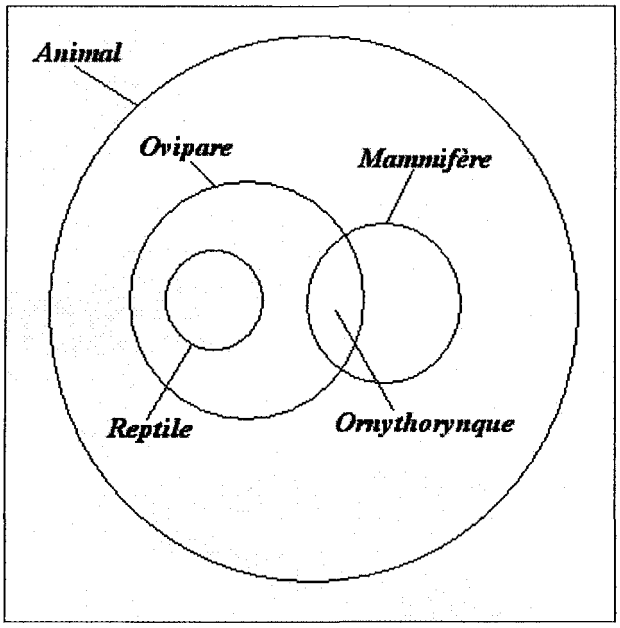


Figure 6 : Représentation ensembliste d'une série de concepts.

3.2.4.1. Subsomption versus Héritage

Il est très important de faire la distinction entre une relation d'héritage (telle que l'on connaît dans les langages objets) et une relation de subsomption. À première vue, les deux relations peuvent sembler identiques, mais même si elles s'apparentent l'une à l'autre, elles ne le sont pas. En vérité, on peut dire que la relation de subsomption subsume ou est plus générale que la relation d'héritage, car elle porte sur les concepts, sur les individus, mais aussi, de façon duale, sur des clauses : les concepts s'apparentent à des conjonctions de littéraux de la logique du premier ordre [Hamburger et Richards, 2002] alors que les clauses sont des disjonctions de littéraux [Borgida, 1996]. La relation de subsomption relève donc d'un niveau d'abstraction plus élevé que la relation d'héritage. On peut voir l'héritage comme un mécanisme de partage de la connaissance qui peut être considéré comme une

façon d'implanter la subsomption [Haton *et al*, 1991]. De plus, l'héritage est une relation tout ou rien, ce qui n'est pas le cas de la subsomption.

3.2.4.2. Détection des relations de subsomption

Afin de pouvoir identifier une relation de subsomption entre deux concepts, il faut les comparer « composante par composante ». Il faut se rappeler qu'un concept en logique de description peut être vu comme une conjonction de caractéristiques définies par des rôles et que des restrictions de cardinalités peuvent être attribuées à ces caractéristiques. Autrement dit, pour vérifier si un concept A subsume un concept B, il faut s'assurer que pour chaque rôle du concept B, il existe un rôle équivalent ou plus général défini par le concept A. Afin de mieux comprendre la méthode de détection, voici une version simplifiée de l'algorithme utilisé avec KL-ONE par Schmolze et Lipkis [Schmolze et Lipkis, 1983] pour déterminer si un concept A subsume un concept B :

Si tous les concepts qui sont subsumants de A sont aussi subsumants de B
Alors retourner vrai
Pour tous les rôles a_i *de A*
Pour tous les rôles b_j *de B*
Si l'un des rôles b_j *de B exprime le même rôle que* a_i *de A (si* $a_i = b_j$*) OU*
Si l'extension du rôle a_i *est incluse par l'extension du rôle* b_j *(si* a_i *subsume* b_j*)*
Alors passer à (a_{i+1} *)*
Sinon retourner échec ou inconnu
Retourner vrai

Il faut maintenant préciser comment il est possible de déterminer si un rôle a_i subsume un rôle b_j, il est nécessaire de répondre aux deux questions suivantes :

- Est-ce que le co-domaine du rôle a_i subsume le co-domaine du rôle b_j?
- Est-ce que la cardinalité du rôle a_i est égale ou supérieure à celle du rôle b_j?

Si les réponses aux questions précédentes sont vraies, alors on peut affirmer que le rôle a_i subsume le rôle b_j.

L'algorithme de vérification d'une relation de subsomption présenté ci-haut peut retourner deux réponses : vrai ou inconnu. Dans le cas où l'algorithme retournerait vrai, cela signifie que le concept A subsume avec certitude le concept B. Cependant, lorsque l'algorithme retourne inconnu, cela ne signifie pas nécessairement que le concept A ne subsume pas le concept B, mais plutôt qu'avec les informations connues sur les deux concepts, il est impossible de le prouver. Le problème de la détection des relations de subsomption est d'une importance capitale pour la classification dans une hiérarchie de concepts.

3.2.5. Différents formalismes terminologiques

Les principales différences entre les langages terminologiques, résident dans le nombre d'opérateurs disponibles pour la construction de concepts et de rôles, ainsi que dans la syntaxe utilisée par ceux-ci. Le premier langage à avoir vu le jour fut le langage FL^-, conçu par Brachman et Schmolze [Brachman et Schmolze, 1985] pour KL-ONE. Ce langage ne

comportait que trois opérateurs, soit *AND*, *ALL* et *SOME*. Il a donné naissance à la famille des langages FL. Tous les langages de cette famille utilisent la syntaxe issue de FL^- en y ajoutant certains opérateurs. Il est à noter que le langage TF utilisé précédemment est basé, lui aussi, sur le langage FL^-.

Dans la littérature, la famille de langages terminologiques de références est la famille AL. La particularité de cette famille est que chaque langage qui en fait partie comporte la syntaxe et les opérateurs de base AL, plus un certain nombre d'opérateurs. Une description des opérateurs les plus communs est disponible au Tableau 1.Chaque opérateur additionnel est représenté par une lettre qui est ajoutée au nom du langage. Par exemple, le langage ALUC est composé des opérateurs de base de AL, ainsi que de l'opérateur de disjonction, représenté par la lettre U et l'opérateur de négation, représenté par la lettre C. Les langages AL forment la plus grande famille de langages terminologiques. La formule générale pour la formation des langages AL est la suivante :

AL[E] [U] [C] [N] [O] [B]

Nom	Lettre (AL)	Symb. concret	Symb. abstrait	Symb. Sémantique	Commentaire
Tout	-	TOP	$\top$	Δ^I	-
Rien	-	BOTTOM	$\bot$	$\varnothing$	-
Conjonction	-	AND	$C \sqcap D$	$C^I \cap D^I$	-
Disjonction	U	OR	$C \sqcup D$	$C^I \cup D^I$	-
Complément	C	NOT	$\neg C$	$\Delta^I \setminus C^I$	-
Quantification universelle	-	ALL	$\forall R C$	-	-
Quantification existentielle non qualifiée	-	SOME	$\exists R$	-	Il existe un rôle R sans restriction sur le co-domaine
Quantification existentielle qualifiée	E	SOME	$\exists R C$	-	Il existe un rôle R qui doit relier un concept C
Restriction de nombre	N	ATLEAST ATMOST EXACT	$\geq n R$ $\leq n R$ $n R$	-	S'applique sur le nombre de rôles de type R
Egalité	-	EQ	$R = S$	$R^I = S^I$	S'applique sur les rôles et les concepts
Inégalité	-	NEQ	$R \neq S$	$R^I \neq S^I$	S'applique sur les rôles et les concepts
Sous-ensemble	-	SUBSET	$R \subseteq S$	$R^I \subseteq S^I$	S'applique sur les rôles et les concepts
Collection	O	ONEOF	$\{ a, b, \ldots \}$	$\{ a^I, b^I, \ldots \}$	Précise la liste des individus qui sont des instances d'un concept
Contrainte d'individus	B	IS	$R : a$	-	Restreint un rôle R à être relié à un individu a
Attributs	F	-	-	-	Permet l'utilisation d'attributs

Tableau 1 : Liste des opérateurs communs des langages terminologiques.

Légende

C et D	=	Désignent deux concepts
R et S	=	Désignent deux rôles
a et b	=	Désignent deux individus spécifiques
n	=	Désigne un entier positif

En plus des deux grandes familles que forment FL et AL, il existe certains langages qui n'appartiennent à aucune de ces familles. Plusieurs SRCT possèdent leur propre langage terminologique adapté à leurs besoins. Par exemple, les langages terminologiques utilisés par les SRCT CLASSIC [Resnick *et al*, 1995] et LOOM [LOOM, 1991]. D'autres langages ayant eu un impact moindre ont également été développés. Par exemple, le langage U, développé par Patel-Schneider en 1987 [Schmidt, 1991][Nebel, 1995], qui se voulait un langage terminologique universel.

3.2.6. Comparaison de la LD avec la logique du premier ordre

La logique de description et la logique du premier ordre (LPO) sont très proches l'une de l'autre. En effet, les concepts que l'on retrouve en logique de description peuvent être vus comme des prédicats unaires et les rôles comme des prédicats binaires. Il est toujours possible d'effectuer la conversion d'une réalité modélisée avec un langage terminologique en LPO, mais l'inverse n'est pas toujours vrai. On peut donc en conclure que la LPO subsume la logique de description. Cependant, la LPO est une logique mathématique difficilement utilisable de façon concrète pour la classification, contrairement à la logique de description.

Une comparaison entre le pouvoir d'expression de la logique de description et celui de la logique du premier ordre est donnée par Baader et Borgida [Baader, 1999] [Borgida, 1996]. Leurs comparaisons établissent que tous les langages terminologiques proposés

jusqu'à présent peuvent exprimer *toutes* les notions exprimables dans la LPO avec trois variables *au plus* et *seulement* celles-là [Borgida, 1996].

3.3. Systèmes de représentation de la connaissance terminologique (SRCT)

Jusqu'à maintenant, il a surtout été question de la partie terminologique de la logique de description, qui sert principalement à définir des concepts et des rôles de façon à pouvoir créer un ensemble où l'on peut classifier les concepts. Dans cette section du chapitre, nous allons nous attarder plus en détails sur la partie assertionnelle de la logique de description. Cette partie assertionnelle nous permet de créer des individus issus des concepts, de manière à peupler notre univers fini.

3.3.1. Composantes d'un SRCT

Un SRCT est un système terminologique permettant l'implantation d'une base de connaissances. Il constitue une application, dans un contexte réel, de la théorie sur la logique de description. Tous les SRCT sont composés de trois parties bien distinctes, comme nous le montre la Figure 7 : la partie terminologique (TBox) qui permet de définir un univers fini, la partie assertionnelle (ABox) qui permet l'introduction d'individus dans cet univers et un moteur d'inférence qui utilise la ABox et la TBox pour fournir différents services aux utilisateurs.

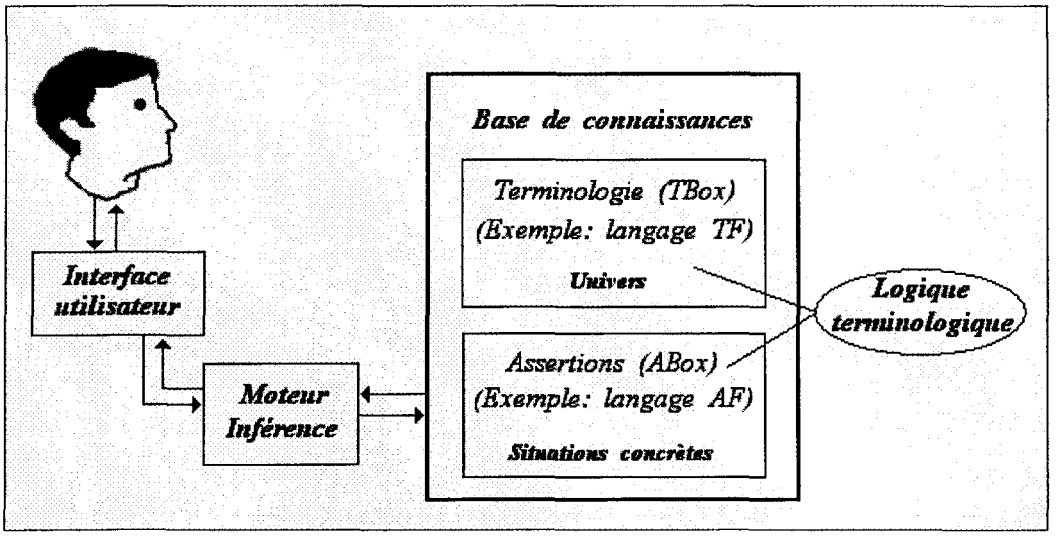


Figure 7 : Schéma général d'un SRCT.

La TBox est essentiellement composée d'une série de définitions terminologiques. Elle est décrite par un langage terminologique précis, par exemple le langage TF. Les définitions introduites dans une TBox sont habituellement non cycliques. Cependant, il est possible de créer un SRCT qui supporte certaines définitions cycliques, comme la récursion et les structures d'objets infinis [Nebel, 1995].

En ce qui concerne la ABox, elle est constituée d'une série d'assertions d'individus, écrites à l'aide de la syntaxe assertionnelle correspondant au langage terminologique utilisé par la TBox. Par exemple, si la TBox utilise la syntaxe du langage TF, la ABox devra utiliser celle du langage AF, qui est son complément assertionnel. La section suivante présentera plus en détails le langage assertionnel AF.

Finalement, le moteur d'inférence est constitué de fonctions et d'algorithmes permettant de fournir des services aux utilisateurs. Il permet de déduire de nouveaux faits en fonction de ceux qui sont définis dans la ABox, en se fiant aux règles de l'univers contenues dans la TBox. Les services d'inférence fournis par les différents SRCT seront décrits dans section 3.3.4 du chapitre.

3.3.2. Langage assertionnel AF

Le langage assertionnel AF sert à introduire des individus réels représentant des instances de concepts ou de rôles, définis préalablement avec le langage terminologique TF. Chaque langage terminologique possède son complément assertionnel qui permet la création d'individus. Il est impossible d'utiliser un langage assertionnel seul, il doit toujours être utilisé avec son équivalent terminologique. La Figure 8 nous présente la syntaxe complète du langage AF ; l'utilisation de cette syntaxe sera décrite dans les prochains paragraphes.

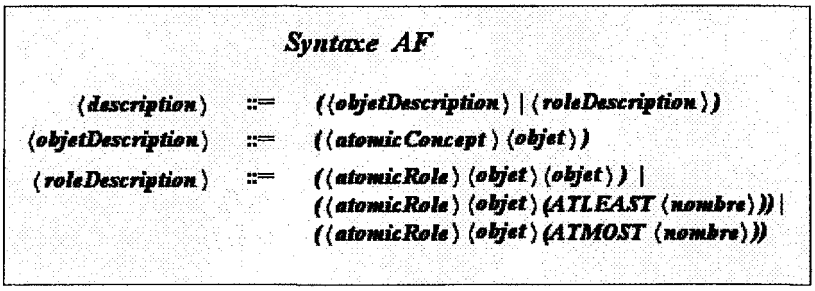


Figure 8 : Syntaxe du langage assertionnel AF.

Le langage assertionnel AF permet principalement trois choses : l'introduction d'individus issus de concepts, l'introduction d'instances de rôles et l'attribution de

restrictions propres aux individus. Il est à noter que dans les exemples qui suivront, le symbole « # » précédera chaque nom d'individus afin de les différencier des noms de concepts. Les notations suivantes nous permettront d'introduire des individus à l'aide du langage AF. Pour chacune des définitions, nous considérerons *C* et *D* comme étant deux concepts, *a* et *b* comme des instances de ces concepts et *R* comme étant un rôle :

a) **Introduction d'un individu issu d'un concept :** cette opération permet de définir des individus spécifiques issus de concepts existants. On peut, par exemple, introduire l'individu nommé « #John » qui est issu du concept « Humain ».

Notation concrète : *(C a)*
Notation concrète détaillée : *(Concept nomConcept)*
Notation abstraite : *C(a)*
Exemple en AF : *(Humain #John)*

b) **Introduction d'une instance de rôle :** cette opération permet de définir qu'un rôle donné existe effectivement entre deux individus. Par exemple, si un rôle nommé « marié » existe dans la terminologie comme étant une relation entre deux « Humains », on peut définir que l'individu « #John » est marié à l'individu « #Claire ».

Notation concrète : *(R a b)*
Notation concrète détaillée : *(Rôle nomIndividu1 nomIndividu2)*
Notation abstraite : *R(a, b)*
Exemple en AF : *(marié #John #Claire)*

c) **Introduction de restrictions sur le rôle d'un individu :** cette opération permet de définir des restrictions de cardinalité spécifique à un individu. Par exemple, supposons qu'un concept nommé « Equipe » soit défini comme ayant au plus 4 « membre », mais qu'un individu spécifique, par exemple « #EquipeA », ne puisse être composé que de trois membres au plus. Il sera possible de définir cette particularité à l'aide du langage AF.

Notation concrète : *(R a (ATLEAST n)), où $n \geq 1$*
: *(R a (ATMOST n))*

Notation concrète détaillée : *(Rôle nomIndividu (ATLEAST n))*
: *(Rôle nomIndividu (ATMOST n))*

Notation abstraite : *R(a, ($\geq$ n))*
: *R(a,($\leq$ n))*

Exemple en AF : *(membre #EquipeA (ATMOST 3))*

3.3.3. Notion de modèle dans un SRCT

Afin d'être en mesure de bien comprendre les mécanismes et les services offerts par les SRCT, il faut revenir sur la notion d'interprétation que nous avons introduite à la section 3.2.3.1, car cette notion s'applique également sur les langages assertionnels (par exemple AF). Par conséquent, l'interprétation d'une instance d'un concept (soit a^I) ou d'un rôle (soit $(a, b)^I$) peut être vue comme étant un individu précis de l'ensemble déterminé par le

concept ou le rôle en question. L'exemple de la Figure 9 illustre, à l'aide du domaine sémantique des êtres humains, cette notion d'interprétation des instances.

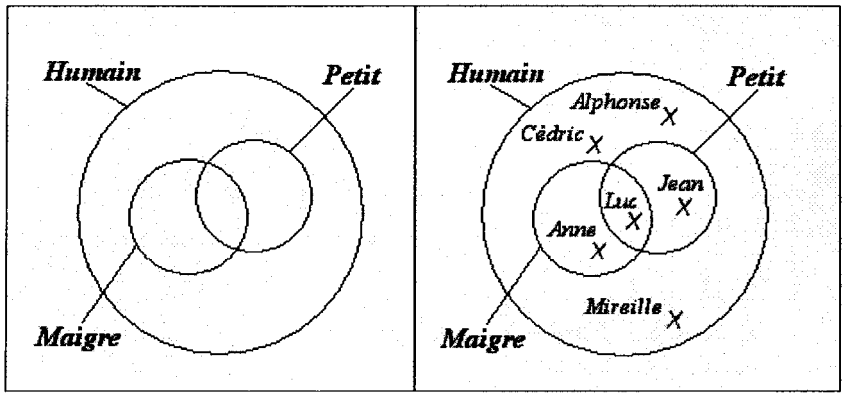


Figure 9 : Introduction d'individus dans une terminologie de concepts.

Pour les prochaines définitions, nous allons considérer les variables suivantes :

I	:	*une interprétation.*
C et D	:	*deux concepts.*
R	:	*un rôle.*
a et b	:	*deux individus (instances) de C et D.*

Une interprétation *I satisfera une TBox T* si et seulement si, pour toutes les définitions terminologiques incluses dans *T*, les deux conditions suivantes sont vérifiées et valides :

a) Pour toutes les définitions de concepts avec le symbole $\doteq$, c'est-à-dire $C \doteq D$ (où D peut être complexe), il est possible de vérifier que $C^I = D^I$.

b) Pour toutes les définitions de concepts avec le symbole $\leq$, c'est-à-dire $C \leq D$, il est possible de vérifier que $C^I \subseteq D^I$.

Ces conditions signifient qu'il existe un ou plusieurs sous-ensembles de Δ^I satisfaisant toutes les définitions terminologiques de *T*. Lorsque cela se produit, l'interprétation *I* devient un *modèle* de la TBox en question. Dans ce cas, la TBox est dite *cohérente.*

La même définition s'applique pour une ABox. Par conséquent, une interprétation *I* *satisfera une ABox A* si et seulement si, pour toutes les définitions assertionnelles incluses dans *A*, les deux conditions suivantes sont remplies :

a) Pour toutes les instanciations de concepts de la forme *C(a)*, il est possible de vérifier que $a^I \in C^I$. Autrement dit, cela signifie que les individus créés par le langage assertionnel existent vraiment dans les ensembles représentant les concepts (atomiques ou complexes) en question.

b) Pour toutes les instanciations de rôles de la forme *R(a, b)*, il est possible de vérifier que $R(a^I, b^I) \in R^I$. Autrement dit, les relations créées par le langage assertionnel existent vraiment dans les ensembles représentant les rôles (atomiques ou complexes).

Dans le cas présent, la signification des conditions est basée sur l'existence réelle des individus dans l'univers représenté. Par exemple, si les instances créées par le langage assertionnel ne sont pas contenues dans le domaine sémantique Δ^I, alors, ces conditions seront automatiquement invalides. Comme dans le cas de la TBox, une interprétation qui satisfait une ABox va s'appeler un *modèle* de la ABox.

Si une interprétation I est à la fois un modèle de la TBox et de la ABox d'un SRCT donné, alors cette interprétation est considérée comme un *modèle* du SRCT. Sachant cela, il sera plus facile de comprendre les services offerts par les SRCT ainsi que les langages assertionnels.

3.3.4. Raisonnement dans un SRCT

Afin de pouvoir raisonner et exploiter les connaissances de la TBox et de la ABox, un SRCT offre un certain nombre de services comme l'héritage, la restriction et la propagation de restrictions, la détection d'incohérences sur les descriptions de concepts et la classification. Une description des différents services et mécanismes qui sont supportés par les SRCT est présentée dans les articles de Baader [Baader et Hollunder, 1991], Valancia [Valancia, 2000] et Nebel [Nebel, 1995]. A titre d'illustration, nous décrirons le mécanisme de classification qui nous intéresse pour la compréhension du chapitre 4.

3.3.4.1. Classification

La classification permet de déterminer la position d'un concept ou d'un rôle dans la hiérarchie. L'emplacement de ceux-ci est d'une importance capitale lors des opérations

d'inférence. Ce service utilise les relations de subsomption décrites à la section 3.2.4, afin de déterminer le positionnement des différents concepts et rôles. Il entre en fonction chaque fois qu'un nouveau concept ou un nouveau rôle est défini dans la terminologie (TBox). Le processus de classification d'un nouveau concept se décompose en deux étapes : la recherche des subsumants les plus spécifiques (SPS) et la recherche des subsumés les plus généraux (SPG). Une fois ces deux étapes complétées, il suffit d'ajouter les relations nécessaires (par exemple une relation d'héritage) entre le nouveau concept et les différents SPS et SPG trouvés pour situer ce concept dans la hiérarchie.

L'exemple suivant nous montre un algorithme de classification inspiré de celui de KL-ONE [Lipkis, 1982]. La recherche des subsumants d'un concept à classer X commence au sommet de la hiérarchie et se poursuit en profondeur. La comparaison du concept courant O^* avec le concept à classer X s'établit de la façon suivante :

COMPARER(O^*, X)
  Si O^* ne subsume pas X
  Alors ***(la hiérarchie est bien formée : aucun descendant de O^* ne peut subsumer X)***
    Retourner nil
  Sinon ***(O^* est temporairement le subsumant le plus spécifique)***
    Si O^* est une feuille dans la hiérarchie
    Alors
      retourner { O^* }

Sinon ***(SPS1 est une variable locale)***

SPS1 = nil

Pour chaque descendant D^* de O^* Faire

SPS1 = SPS1 ∪ COMPARER(D^*, X)

(Si SPS1 a une valeur,

alors un des descendants de O^* est le subsumant le plus spécifique)

Si SPS1 = nil

Alors

retourner { O^* }

Sinon

retourner SPS1

Seule une partie de la hiérarchie est explorée : si O^* ne subsume pas X, alors le sous-arbre d'héritage de la racine O^* est élagué. L'exploration se poursuit à partir du premier frère de O^* faisant encore partie de l'ensemble des concepts à explorer. Si O^* subsume X, alors O^* est temporairement le subsumant le plus spécifique du sous-arbre courant. L'exploration se poursuit en profondeur à partir de O^*. Si aucun des descendants de O^* ne subsume X, alors O^* est déclaré être le subsumant le plus spécifique du sous-arbre courant. Une nouvelle relation, un lien d'héritage par exemple, peut-être mise en place entre le concept X et ses subsumants les plus spécifiques contenus dans SPS.

La seconde partie de l'algorithme concerne la recherche des subsumés les plus généraux. L'obtention des subsumants les plus spécifiques permet de focaliser cette recherche : puisque la relation de subsomption est transitive, il suffit de n'explorer que les descendants des SPS qui possèdent les mêmes propriétés que X. Si { D } désigne l'ensemble des descendants des SPS et D^* le descendant courant, l'exploration s'effectue sur { D } de la

façon suivante : si X subsume D^*, alors ce dernier est un SPG, sinon l'exploration se poursuit sur les descendants de D^* jusqu'à ce qu'un SPG ait été trouvé ou bien que D^* n'ait pas de descendant.

3.3.5. LOOM

Le SRCT appelé LOOM, conçu par l'Information Science Institute (ISI) de l'*University of Southern California*, se veut un langage très riche et complet, ce qui en fait un langage très complexe intégrant une multitude de fonctionnalités [McGregor, 1991][McGregor et Burstein, 1991]. LOOM est l'un des SRCT possédant le plus d'opérateurs de définitions pour les concepts et les rôles. Une grammaire simplifiée de LOOM est présentée à la Figure 10.

Les constructeurs qui sont inclus dans la majorité des langages (à l'exception du constructeur *TEST* qui est unique à CLASSIC) sont supportés par la grammaire de LOOM. Il faut aussi noter que les constructeurs *OR* et *NOT*, qui ont été écartés en CLASSIC, ainsi que dans plusieurs autres langages, sont également supportés. Soulignons également l'existence du constructeur « exactly », qui est propre à LOOM. Ce constructeur particulier permet de définir une restriction de cardinalité sur un rôle de façon exacte. Par exemple, on pourrait définir que le rôle « soutenu-par », qui implique un concept « Chaise » et un concept « Pied », doit être de cardinalité 4. De cette façon, on peut spécifier qu'une « Chaise » a exactement 4 « Pied ». Il est à noter qu'il est possible de représenter cette même situation, dans les autres formalismes terminologiques, avec la conjonction d'une

restriction *AT-MOST* avec une restriction *AT-LEAST*. Par exemple, on peut définir le rôle « soutenu-par » comme étant de cardinalité minimum 4 (avec *AT-LEAST*) et maximum 4 (avec *AT-MOST*). Finalement, les constructeurs « domain » et « range » permettent de spécifier le domaine et le co-domaine d'un rôle, ce qui ajoute à la richesse du langage.

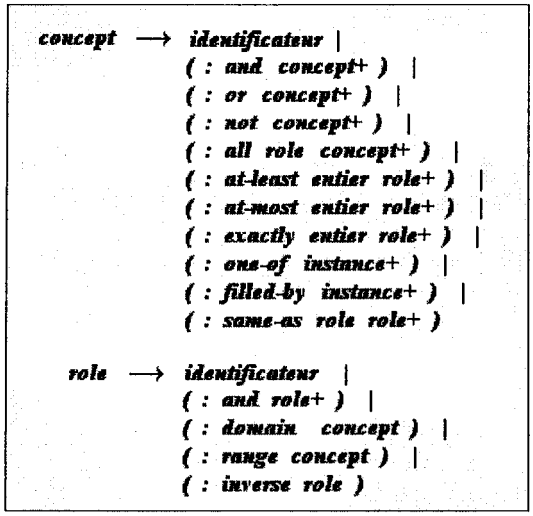


Figure 10 : Une grammaire simplifiée du langage de LOOM [LOOM, 1991].

3.3.5.1. Comparaison des formalismes LOOM et TF

LOOM possède un formalisme propre à lui-même. Pour illustrer cette particularité, l'exemple de la Figure 11 nous montre la représentation d'une situation réelle en langage TF et la compare avec la même modélisation effectuée avec le formalisme de LOOM. Il faut bien remarquer le niveau de détails que l'on peut obtenir avec LOOM, grâce aux définitions de concepts et de rôles. On peut noter, en regardant la Figure 11, que les

déclarations de concepts et de rôles, qui s'effectuaient en TF à l'aide des symboles $\doteq$ et $\preceq$, s'effectuent ici grâce aux commandes « defconcept » et « defrelation ».

TF	*LOOM*
Être ≤ *T* *Ensemble* ≤ *T* *Humain* ≤ *Être* *Homme* ≤ *Humain* *Femme* ≤ *Humain* *(disjoint Homme Femme)* *Robot* ≤ *Être* *(disjoint Robot Humain)* *membre* ≤ *T* *chef* ≤ *membre* *Équipe* ≐ *(AND Ensemble (ALL membre Être) (ATLEAST 2 membre))* *Petite-Équipe* ≐ *(AND Équipe (ATMOST 5 membre))* *Équipe-Moderne* ≐ *(AND Équipe (ATMOST 4 membre) (ATLEAST 1 chef) (ALL chef ROBOT))*	*(defconcept ETRE :is :primitive)* *(defconcept ENSEMBLE :is :primitive)* *(defconcept HUMAIN :is (:and ETRE :primitive))* *(defconcept HOMME :is (:and HUMAIN :primitive))* *(defconcept FEMME :is (:and HUMAIN (:not HOMME) :primitive))* *(defconcept ROBOT :is (:and ETRE (:not HUMAIN) :primitive))* *(defrelation membre :is :primitive))* *(defrelation chef :is (:and membre :primitive))* *(defconcept EQUIPE :is (:and ENSEMBLE (:all membre ETRE) (:at-least 2 membre)))* *(defconcept PETITE-EQUIPE :is (:and EQUIPE (:at-most 5 membre)))* *(defconcept EQUIPE-MODERNE :is (:and EQUIPE (:at-most 4 membre) (:at-least 1 chef) (:all chef ROBOT))*

Figure 11 : Comparaison du langage TF avec le formalisme de LOOM.

Une autre particularité vient du fait que les notions de concepts définis et primitifs apparaissent de façon explicite dans la définition des concepts, via la commande « :primitive », utilisée lors de la définition des concepts « ETRE », « ENSEMBLE », « HUMAIN », « HOMME », « FEMME » et « ROBOT ». L'absence de la commande « :primitive » lors de la définition des concepts « EQUIPE », « PETITE-EQUIPE » et « EQUIPE-MODERNE » spécifie implicitement que ces concepts sont bien définis.

Finalement, il faut noter que les disjonctions, définies en TF par l'opérateur « disjoint », sont effectuées ici à l'aide de l'opérateur de négation « not ».

3.3.5.2. Bilan sur LOOM

LOOM se veut, à l'heure actuelle, l'un des formalismes de représentation terminologique les plus complets et les plus polyvalents. L'environnement de LOOM a été enrichi, notamment avec un système à base de règles de production [Yen *et al*, 1991a][Yen *et al*, 1991b]. De plus, le système de raisonnement de LOOM utilise un moteur d'inférence très complet. C'est pour toutes ces raisons que LOOM est considéré comme l'un des SRCT les plus expressifs et les plus riches.

Plusieurs systèmes qui utilisent LOOM comme système de représentation de la connaissance ont été construits. Notamment, un système de gestion d'informations hétérogènes [Arens *et al*, 1993], un système de raisonnement à partir de cas dans le domaine légal [Ashley et Aleven, 1994] ainsi qu'un système de bases de données hiérarchiques réparties [Chu *et al*, 1993]. Il faut aussi mentionner la création d'une nouvelle version de LOOM appelée PowerLoom [PowerLoom, 1997] par ISI, que nous avons d'ailleurs utilisée comme plate-forme pour le développement de notre système de comparaison de points de vue. Il sera question de cette nouvelle version de LOOM dans le chapitre 4.

3.4. Conclusion

Au cours de ce chapitre, nous avons décrit informellement la logique de description et les notions qui s'y rattachent, telles les notions de concepts, rôles, etc. Nous avons décrit aussi la relation de subsomption permettant de classifier les concepts par leur généralité. Puis nous avons effectué une présentation de système de représentation de la connaissance terminologique (SRCT) et d'éléments comme la ABox et la TBox. Par ailleurs, le langage terminologique TF de même que le langage assertionnelle AF, nous a permis d'illustrer les différents exemples de ce chapitre.

La logique de description possède plusieurs avantages en tant que formalisme de représentation de la connaissance. Elle est reconnue pour être particulièrement efficace pour le raisonnement par classification. Cette logique possède une sémantique bien définie. Ceci implique que l'existence des instances n'est pas donnée de façon opérationnelle ou par l'implémentation elle-même. Ce serait l'inverse si on affirmait que l'assertion « John est un père » était valide uniquement parce que le système a répondu « père » à la question « Qu'est-ce que John ? ». L'existence des instances est plutôt donnée par des modèles et des descriptions qui permettent à la logique de description de s'articuler dans un univers bien défini et non d'être une simple suite de définitions statiques de références. Il faut également souligner que la logique de description est une logique formelle qui est beaucoup simple et intuitive d'utilisation que la logique du premier ordre, entre autres, parce qu'il n'y a pas de quantificateur.

De plus, il a été prouvé que la logique de description est complète et correcte [Baader *et al*, 1991], ce qui permet de l'utiliser dans plusieurs domaines connexes à l'intelligence artificielle. Par exemple, les travaux de Schmidt [Schmidt, 1992] [Schmidt, 2000] ont permis de voir la logique de description comme étant une solution possible pour la compréhension du langage naturel. On peut également voir les applications de la logique de description dans le domaine des systèmes multi-agents, pour tenter de résoudre le problème de l'hétérogénéité sémantique [Valancia, 2000] concernant l'utilisation de terme différent pour décrire une même réalité. La logique de description a aussi été proposée pour la modélisation de bases de données [Borgida, 1996]. Récemment, le consortium W3C a émis un guide pour tenter de standardiser les ontologies utilisées par les services Web avec le Web Ontology Language [McGuinness *et al*, 2002].

Malgré tous les avantages de la logique de description, son principal problème est sa dualité expressivité versus complexité [Nebel, 1995]. Autrement dit, plus un langage terminologique permet de modéliser un grand nombre de réalités, plus sa complexité théorique augmente au niveau des opérations d'inférences, en particulier pour la détection des relations de subsomption [Schmidt, 1991].

Finalement, bien que ce chapitre constitue une introduction détaillée à la logique de description, il faut savoir qu'il existe plusieurs points qui n'ont pas été présentés en profondeur, notamment les mécanismes pour supporter les définitions cycliques et récursives, les différents problèmes reliés à la complexité des SRCT et le fonctionnement

de certains services. Ce chapitre se veut cependant une introduction suffisamment complète pour comprendre la suite de ce document.

CHAPITRE 4

MODÈLE D'ACTION BASÉ SUR LA LOGIQUE DE DESCRIPTION POUR LA RECONNAISSANCE DE PLAN

4.1. Introduction

Nous avons défini la reconnaissance de plan comme « prendre en entrée une séquence d'actions effectuées par un acteur puis inférer le but poursuivi par celui-ci et finalement d'organiser la séquence d'actions en termes de structure de plan » [Schmidt *et al*, 1978]. Nous avons admis, par ailleurs, qu'un plan exécuté par un agent acteur serait une instance d'un plan que l'on peut former grâce à la librairie. Dès lors, à la vue de ces quelques définitions, il est évident que les notions de plan et d'action sont les cœurs principaux de cette tâche. Ainsi comme nous l'avons précisé, précédemment au cours du chapitre 1, nous allons présenter ici une modélisation formelle de l'action à travers la logique de description. Le but de nos travaux est de dégager les éléments nécessaires à la reconnaissance de plan afin de pouvoir les formaliser au mieux. En prévision de préparer, ultérieurement, un modèle, fondé sur la logique de description, qui serait basé sur ces formalisations. Ainsi, ma contribution à travers ce modèle d'action et les outils de subsomption que nous allons décrire dans ce chapitre s'inscrivent dans le cadre d'un travail de doctorat de Monsieur Bruno Bouchard sur la reconnaissance de plan.

En effet, plusieurs représentations de plans et d'actions ont été développées telles que STRIPS [Fikes, Nilsson, 1971], PDDL [Ghallab *et al*, 1998] qui emploient des formules de la logique du premier ordre pour décrire les états du monde. Néanmoins, ceux-ci ne sont pas suffisamment expressifs pour formuler des structures hiérarchiques d'actions permettant des raisonnements taxonomiques (type de raisonnements visant à classifier les connaissances) ou des abstractions de tâches de planifications dans une situation de

résolution de problèmes. La logique de description peut être utile dans les problèmes de planifications dans le sens que les entités dynamiques (concepts d'actions) ont à être intégrées dans ces langages de planifications (STRIPS, PDDL, etc.). Il en résulte, alors, des systèmes de planifications qui seront ainsi mieux équipés pour répondre plus efficacement aux applications de planification du monde réel.

Notre contribution se base sur le fait que le modèle « état transition de l'action » peut être considéré comme un modèle pour les logiques de descriptions dynamiques en présentant une interprétation du changement de concept, quand une action est effectuée.

4.1.1. Notion d'état

D'une manière générale, en intelligence artificielle, une action est perçue à travers une approche d'état et de transformation d'état. On suppose, dès lors, qu'il est possible de caractériser l'ensemble Σ des états possibles du monde. Une action *op* est alors définie comme une transition d'état, c'est-à-dire comme un opérateur dont l'exécution produit un nouvel état. Ainsi, à partir d'un état σ_1 *de* Σ, l'exécution de l'action *op* produit un nouvel état σ_2 de Σ. Par exemple, si à l'état σ_1, un agent A se trouve au lieu L_1 et que l'on applique l'action $op=aller(A,L_1,L_2)$, l'état σ_2 qui en résulte verra l'agent déplacé au lieu L_2. Il est possible de faire un parallèle entre cette approche et celle de la pellicule d'un film. Chaque image représente un état du monde et les actions décrivent le passage d'une image à une autre. Le déroulement du film donne alors l'illusion d'une continuité du mouvement.

4.1.2. Interprétation d'un état

Il est possible de définir l'ensemble des états possibles et des actions par une paire $<W, A>$, où W est un ensemble d'états possible du monde et A est l'ensemble d'action applicable sur ces états. Une action a pouvant être définie sur un ensemble d'états W comme une relation binaire telle que $<w, e> \in a$ (w et e représentent respectivement à l'état courant et l'état suivant) si et seulement si $a(w) = \{ e / \langle w, e \rangle \in W \times W \}$. Dans ce cadre, les actions sont déterministes et opèrent sur les formules d'assertion, lesquelles sont des cas particuliers de formules de la logique du premier ordre [Borgida, 1996]. Si l'expression conceptuelle et l'assertion de la logique de description sont utilisées pour décrire les faits concernant un état du monde, ils peuvent être satisfiables ou non dépendamment de l'état. Par conséquent, les états du monde peuvent correspondent à une structure sémantique de la forme $w = \langle Dom(w), (.)^{I_w} \rangle$. Le domaine $Dom(w)$ étant le domaine d'interprétation, c'est-à-dire un ensemble non nul d'objets appelés individus qui existent dans le monde lorsque celui-ci est dans un état w à un moment précis. La fonction $(.)^{I_w}$ étant la fonction d'interprétation associée à w, elle assigne à chaque symbole conceptuel C, un sous-ensemble du domaine $Dom(w)$, tel que $(C)^{I_w} \subseteq Dom(w)$ et pour chaque rôle un sous-ensemble du domaine $Dom(w)$ x $Dom(w)$, de manière à ce que les équations suivantes soient satisfaites :

1) $(C \sqcap D)^{I_w} = C^{I_w} \cap D^{I_w}$
2) $(C \sqcup D)^{I_w} = C^{I_w} \cup D^{I_w}$
3) $(\neg C)^{I_w} = C^{I_w} - Dom(w)$
4) $(and\, r.C)^{I_w} = \left\{ i \in Dom(w) \mid \forall j : (i,j) \in r^{I_w} \rightarrow j \in C^{I_w} \right\}$
5) $(some\, r.C)^{I_w} = \left\{ i \in Dom(w) \mid \exists j : (i,j) \in r^{I_w} \wedge j \in C^{I_w} \right\}$
6) $(\geq n\, r)^{I_w} = \left\{ i \in Dom(w) \mid \# (j \in Dom(w) : (i,j) \in r^{I_w}) \geq n \right\}$
7) $(\leq n\, r)^{I_w} = \left\{ i \in Dom(w) \mid \# (j \in Dom(w) : (i,j) \in r^{I_w}) \leq n \right\}$

Les symboles C et D désignent deux noms de concepts et r le nom d'un rôle dans le sens de la Logique de Description. La relation de subsomption entre les concepts C, D est dénotée par $D \prec C$; ce qui signifie que $D^{I_w} \subseteq C^{I_w}$ dans l'état w. L'interprétation d'une conjonction (respectivement d'une disjonction) de concepts, désignée par les équations 1 et 2, se ramène à l'intersection (respectivement la réunion) de concepts. De même, pour l'équation 3, l'interprétation de la négation de concept est définie comme le complémentaire du concept. Les interprétations concernant les équations 4 et 5 signifiant qu'il existe un rôle *r*, dont la valeur est le concept *C*, affirme l'existence d'un couple d'individu *(i,j)* en relation par l'intermédiaire du rôle *r*, où j est de type *C*. L'interprétation de l'équation 6 précise le nombre minimum d'individus associés à *r* et devant être supérieure à *n*. De même pour l'équation 7, où l'interprétation précise le nombre maximum d'individus associés à *r* et devant être inférieur à *n*. Les actions interviennent au niveau factuel relatif à l'extension des concepts. L'assertion *C(i)* stipule qu'un individu *i* est une instance de concept *C* et l'assertion *r(i,j)* indique que le couple d'individus est une extension de *r*. Dans le but d'associer une interprétation à ces assertions, la fonction $(.)^{I_w}$

est étendu aux individus tels que $C(i)$ est satisfait par w, ce qui peut être noté par $w \vDash C(i)$ si et seulement si $i \in C^{I_w}$

La notion de variable n'est pas utilisée en logique de description étant donné qu'un concept dénote l'ensemble des individus qui le constitue, il y a d'ailleurs une équivalence logique entre la sémantique d'un concept (et d'un rôle) et un prédicat unaire (binaire). Dans le cas de l'action, il est nécessaire d'introduire la notion de variable libre ou quantifiée pour exprimer les contraintes à propos des états. Ces contraintes ne constituant pas nécessairement une description dans le sens de la logique de description. A ce niveau, la notion d'état est utilisée pour attribuer une interprétation aux variables en complément avec les autres symboles utilisés en logique de description. Ainsi, dire qu'une assertion est valide pour un état s'exprimera par, $w \vDash \varphi$, indiquant que l'assertion φ est satisfiable ou que l'état w satisfait φ. En somme, la satisfiabilité de φ se définit récursivement de la manière suivante :

$$w \vDash (x_1 = x_2) \in \text{ iff } x_1^{I_w} = x_2^{I_w}$$
$$w \vDash \mathrm{C}(x) \text{ iff } x^{I_w} \in \mathrm{C}^{I_w}$$
$$w \vDash \mathrm{r}(x, y) \text{ iff } \langle x^{I_w}, y^{I_w} \rangle \in r^{I_w}$$
$$w \vDash \forall x \varphi \text{ iff } w[i/x] \vDash \varphi, \forall i \in Dom(w)$$
$$w \vDash \exists x \varphi \text{ iff } w[i/x] \vDash \varphi, \exists i \in Dom(w)$$
$$w \vDash \neg\varphi \text{ iff } w \nvDash \varphi$$
$$w \vDash \varphi \sqcap \psi \text{ iff } w \vDash \varphi \text{ and } w \vDash \psi$$
$$w \vDash \varphi \sqcup \psi \text{ iff } w \vDash \varphi \text{ or } w \vDash \psi$$

où $w[i/x]$ désigne l'état obtenu à partir de w en substituant i à x. Les actions peuvent ne pas altérer l'ensemble des objets existant dans le monde, ce qui, pour une action $\langle w,e\rangle \in a$, se traduit par $Dom(w) = Dom(e)$.

4.1.3. Interprétation d'une action

Une action $a(w)$ est une structure utilisant la formulation traditionnelle proposée par le langage STRIPS [Fikes, Nilsson, 1971]. Chaque précondition $pre(a)$ étant une conjonction d'assertion concernant les objets conceptuels de même que les rôles qui sont en relation avec ses objets. L'ensemble des états dans lequel l'action $a(w)$ peut être exécuté, est donnée par le domaine :

$$Dom(w) = \{w \in W / \ w \vDash pre(a)\}$$

Ainsi, chaque assertion qui compose $pre(a)$ doit satisfaire chaque état w tel que $a(w) \neq 0$. Les effets de l'action $pos(a)$ peuvent être exprimé par l'ajout de conditions d'assertions décrites par les formules d'assertions $pos^{+}(a)$, qui signifient l'ajout à l'interprétation du concept ou du rôle impliqué dans une action a(w) et la suppression de l'interprétation du concept ou du rôle étant dénoté par $pos^{-}(a)$. Pour chacun des symboles conceptuels C de l'assertion $C(x)$, il est possible de construire une fonction f_C^a qui va décrire comment le concept C va être interprété lors de l'interprétation de l'action $a(w)$. De plus pour chaque action $<w, e> \in a$, la nouvelle interprétation du concept C est exprimée par :

$$C^{I_e} = f_C^a(w) = \{i \in Dom(w) \mid e \vDash pos_C(a)\},$$

tel que pour chaque concept C, il doit exister une formule d'assertion $pos_C(a)$ qui satisfait l'état suivant e. De la même manière, pour chaque symbole de rôle r, il est possible de construire une fonction d'interprétation f_r^a pour un rôle r impliqué dans une action $a(w)$ tel que :

$$r^{I_e} = f_r^a(w) = \{< i, j > \in Dom(w)^2 \mid e \vDash pos_r(a)\},$$

Cet ensemble de fonction d'interprétation, pour les concepts et de symboles de rôles, fournit un cadre approprié pour caractériser une action en logique de description, à travers les transitions d'état.

Les effets d'une action sont définis comme la différence entre l'ensemble des interprétations des symboles conceptuelles et les rôles. Chaque formule d'assertion $pos_C(a)$ d'une transition d'état pour un concept C, pouvant être exprimée en deux formules d'assertions suivantes : $pos_C^+(a)$ et $pos_C^-(a)$ qui respectivement décrivent les conditions d'ajouts et suppression de l'interprétation de chaque concept C. De plus, si une action $a(w)$ est effectuée dans un état w, l'interprétation de C dans l'état suivant e est donnée par la formulation suivante :

$$C^{I_e} = \begin{cases} C^{I_w} - (pos_C^-(a))^{I_e} & Si \quad ((pos_c^+(a))^{I_e} \cap (pos_c^-(a))^{I_e}) = 0 \\ C^{I_w} & sinon \end{cases}$$

La formule d'assertion $pos_C^+(a)$ dénote l'ensemble des individus qui sont ajoutés à l'interprétation de C dans un nouvel état e quand l'action $a(w)$ est exécutée et qui est décrite par :

$$(pos_C^+(a))^{I_e} = \{i \in Dom(w) \mid e \vDash pos_C^+(a)\}$$

La formule d'assertion $pos_C^-(a)$ désigne l'ensemble des individus qui peuvent être supprimé de l'interprétation de C dans un état e quand l'action $a(w)$ est exécutée. Il est possible de l'exprimer comme suit :

$$(pos_C^-(a))^{I_e} = \{i \in Dom(w) \mid e \vDash pos_C^-(a)\}$$

De là pour chacune des actions exécutées, les individus vont changer de type à travers l'exécution de celles-ci. Donc, la sémantique de l'action $a(w)$ en logique de description peut être formulée comme suit :

$$a(w) = \left\{ \begin{array}{l} f_c^a(w) = \left\{ \begin{array}{l} i \in Dom(w) \mid w \vDash pre(a) \wedge \\ \qquad\qquad e \vDash (pos_c^+(a) \sqcup \exists y\ C(y)) \sqcap \neg pos_c^-(a) \end{array} \right\} \\ f_r^a(w) = \left\{ \begin{array}{l} \langle i,j \rangle \in Dom(w)^2 \mid w \vDash pre(a) \wedge \\ \qquad\qquad e \vDash (pos_r^+(a) \sqcup \exists x,y\ r(x,y)) \sqcap \neg pos_r^-(a) \end{array} \right\} \end{array} \right.$$

Il est à noter que, $(pos_C^+(a))^{Ie} \cap (pos_C^-(a))^{Ie} = 0$ implique qu'il n'existe pas d'individus qui peuvent satisfaire simultanément les assertions d'addition et de suppression. De là, il est

possible de dire que, $\{pre_C(a)\} \not\models pos_C^+(a) \sqcap pos_C^-(a)$. Dès lors, l'expression $pos_C(a)$ concernant une transition d'état peut être réécrite comme suit :

$$(pos_c^+(a) \sqcup \exists y\ C(y)) \sqcap \neg pos_c^-(a))$$

Ceci signifiant que le concept C reste satisfait après l'exécution de l'action $a(w)$ si et seulement si, l'action a été satisfaite soit grâce à $pos_C^+(a)$ ou bien si C a été satisfait parce qu'il existe au moins un individu y de concept C, que l'action $a(w)$ ne rend pas satisfiable grâce à $\neg pos_C^-(a)$. Ceci étant identique pour un rôle r concerné par l'action $a(w)$.

4.1.4. Subsomption d'actions

Nous allons définir maintenant la relation de subsomption pour les actions afin de les organiser en structure hiérarchique. Soient a et b, deux actions quelconques, il est possible de dire informellement que a subsume b, indiqué par $b \prec_a a$, si une action a est satisfiable dans tous les états où b est satisfiable :

$$\begin{aligned} b \prec_a a \Rightarrow \forall \langle w, e \rangle \in b : (w \vDash pre(b) \Rightarrow w \vDash pre(a)) \wedge \\ (e \vDash pos^+(b) \Rightarrow e \vDash pos^+(a)) \vee \\ (e \vDash pos^-(b) \Rightarrow e \vDash pos^-(a)) \end{aligned}$$

si $b \prec_a a$ alors $Dom(b) \in Dom(a)$, il doit exister un état $w_b \in Dom(b)$, tel que $w_b \in Dom(a)$, à partir de là $w_b \vDash pre(a)$. Le même principe est vrai pour les postconditions de cette action en utilisant le co-domaine donné comme suit :

$$CoDom(a) = \{w \in W \mid w \vDash pos(a)\}.$$

4.2. Démarche pour l'implémentation du modèle proposée

Nous avons proposé jusqu'ici un modèle pour la représentation des actions. Ainsi à partir de maintenant, nous allons décrire un langage informatique pour ce modèle, appelé PDL (« Plan Description Langage ») qui se veut un enrichissement et une spécialisation de la logique de description pour répondre aux besoins particuliers inhérents au domaine de la reconnaissance de plan. Nous proposerons une application qui sera l'implémentation de celui-ci. Dans PDL, la gestion des actions s'inspire de celle, décrite dans le modèle de la première partie de ce chapitre. En effet, celles-ci sont, ici, définies avec une liste de fait décrivant les états correspondants. En effet, la liste de faits initiaux décrit l'état de départ, tandis que les listes de faits ajoutés et supprimés permettent de construire l'état d'arrivée tel que représenté dans STRIPS [Fikes, Nilsson, 1971]. En outre, dans le modèle de la première partie, les plans sont considérés comme un ensemble de suite d'action. Les actions seront ainsi ordonnancées séquentiellement avec un opérateur appelé *SEQ*, de manière antinomique avec l'opérateur *CHC* ou avec l'opérateur *COL* exprimant qu'il n'y a aucun ordre entre les actions mais celles-ci devant être effectuées.

L'application que nous proposons a été basée sur un système de représentation de la connaissance terminologique (SRCT), nommé PowerLoom [PowerLoom, 1997], conçu par l'Information Science Institute (ISI) de l'University of Southern California. Pour réaliser ce projet, nous avons encapsulé le SRCT PowerLoom en ajoutant une couche logicielle supérieure venant gérer les particularités relatives au langage PDL.

Dans PowerLoom, un usager peut créer une structure hiérarchique de concepts à l'aide d'une interface en ligne de commande. Cette interface permet d'interroger la base de connaissances terminologique et d'introduire des termes et de nouveaux concepts à l'intérieur de la structure hiérarchique. Le moteur d'inférence de PowerLoom classifie automatiquement les concepts et les organise en hiérarchie à l'aide de la relation de subsomption. De cette façon, l'usager peut se construire une base de connaissances terminologique peuplée de concepts, la gérer et l'interroger.

Dans l'application proposée, toute cette mécanique a été récupérée et encapsulée. En effet, lorsqu'un utilisateur introduit un nouveau concept décrivant un fait, cette définition est envoyée au moteur d'inférence de PowerLoom qui se chargera de la classifier, de la stocker et d'en gérer son introduction dans la structure de concepts. Lorsque la couche logicielle de l'application a besoin d'inférer sur ce type de concept, il interroge simplement le moteur d'inférence de PowerLoom.

La couche logicielle de l'application s'occupe, quant à elle, de gérer l'introduction, le stockage et la gestion des concepts d'actions et de plans. Ces nouveaux types de concepts ont une structure et une interprétation différente de celle qui est utilisée par les concepts traditionnels de la logique de description, ils ne peuvent donc pas être pris en charge par PowerLoom. En effet, les plans de par, leur gestion des actions, ne peuvent être traité par PowerLoom. Nous employons dès lors un algorithme inspiré de la théorie des automates,

pour implanter notre modèle etat-transition de l'action décrit de manière conceptuelle dans la section précédente.

4.2.1. Alphabet du langage

L'alphabet du langage PDL, dénotée par Σ, représente l'ensemble des symboles terminaux permettant de construire des expressions correctes appartenant à ce langage. Si Σ^* définit l'ensemble de toutes les expressions possibles formées à l'aide de l'alphabet Σ, alors il est possible de dire que PDL$\subset \Sigma^*$. L'alphabet Σ se définit formellement de la manière suivante :

Σ = { (,), *deffact*, *defplan*, *defaction*, *list*, *supp*, *SEQ*, *COL*, *CHC*, *User*, *Agent*, *terme*}

(,) : Ces symboles (parenthèses ouvrantes et fermantes) servent à spécifier le début et la fin d'une expression dans le langage PDL.

Deffact : Il désigne la commande pour définir un fait, la commande étant envoyée à PowerLoom.

Defaction : Il désigne la commande pour définir une action.

Defplan : Il désigne la commande pour définir un plan.

List : Il désigne la commande pour afficher la définition d'un élément (fait, action, plan)

Supp : Il désigne la commande pour supprimer un élément (fait, action, plan)

SEQ : Il désigne l'opérateur d'ordonnancement séquentiel vu au point précédent.

CHC : Il désigne l'opérateur de choix entre les actions, celui-ci devant être exclusif.

COL : Il désigne l'opérateur de collection d'action, où l'ordre n'est pas important, seul le fait que les actions devant être effectuées.

4.2.2. Grammaire du langage au format BNF (Backus Norm Form)

Nous allons décrire la syntaxe de PDL avec une grammaire au format BNF. En effet, cette grammaire facilite le développement rigoureux d'outil de manipulation des structures du langage (analyses, interprétation, compilation, etc.). La notation *BNF* est d'ailleurs le seul formalisme de description des syntaxes de langages actuellement normalisé [Mariano, 1997].

Symbole	Signification
::=	est définie par
|	Ou (disjonction)
*	Aucune ou plusieurs
< >	Expression du langage
[exp]	Expression optionnelle
+	Un ou plusieurs
« »	chaînes de caractères

Tableau 2 : Description des symboles employés

Declaration ::= (<Fait_definition> | <Action_definition> | <Plan_definition> |
<Affichage> | <Suppression>)

<Fait_definition> ::= deffact <espace>$^+$ <nom_fait> (?x | ?x <espace>$^+$ <nom_fait>)

<Action_definition>::= defaction <espace>$^+$ <nom_action>
*(<acteur><liste_fait><liste_*
fait><liste_fait>)

<Affichage> ::= list <espace>$^+$ <nom_fait>|<nom_action> | <nom_plan>

<Suppression> ::= supp <espace>$^+$ <nom_fait>|<nom_action> | <nom_plan>

<Acteur> ::= « user » | « agent »

<Plan_definition>::= defplan <espace>$^{+}$ <nom_plan>(<acteur><liste_action><liste_action><liste_action>)

<liste_fait> ::= (<ens_fait>)

<ens_fait> ::= <nom_fait> | <nom_fait> <espace>$^{+}$ <ens_fait>

<liste_action> ::= (seq|chc|col <liste_action>|<liste_action><espace>$^{+}$<liste_action>)

<Espace> ::= Caractère ascii numéro 32, symbolisant l'espace

<nom_fait>|<nom_action> | <nom_plan> ::= Chaîne de caractères non – alphanumérique représentant le nom du concept correspondant.

4.2.3. Représentation des Faits, Actions et Plans

4.2.3.1. Construction d'un Fait

Un Fait est un concept au sens de la logique de description, gérable par PowerLoom. De plus, la structure d'un Fait en PDL est identique à celle d'un concept de PowerLoom. Ainsi, un Fait pourra contenir des rôles, des parents, etc. Il sera déclaré de la même manière qu'un concept de PowerLoom sauf qu'au lieu d'utiliser *defconcept*, on emploie *deffact*. Ceci nous donne la possibilité de pouvoir vérifier la syntaxe de l'utilisateur avant de la transformer en déclaration de concept classique sous PowerLoom. Une syntaxe possible de la déclaration des faits ETRE et HUMAIN sera de la forme :

(deffact ETRE(?x))

(deffact HUMAIN(?x ETRE))

4.2.3.2. Construction d'une action

Une action est une structure qui peut être représentée par un quadruplet de la forme : <*Acteur*, *Initiale*, *Ajouts*, *Suppressions*>. Un tel concept comme défini dans le modèle de la première partie de ce chapitre représente un changement, une transformation qui peut être apportée par une entité à son environnement. Il comprend dans sa description le type d'acteur qui doit effectuer ce changement, la description de l'état initial nécessaire afin de pouvoir exécuter une telle action (sous forme d'un ensemble de concepts décrit par des faits déclarés directement dans Powerloom), une description des éléments que cette action ajoutera à l'état et finalement une spécification des éléments que celle-ci retirera de cet état. Ainsi, la syntaxe en PDL sera de la forme :

(*defaction nom_action*((*agent*)(*Fait_1 Fait_2*) (*Fait_1 Fait_2*) (*Fait_1 Fait_2*)))

Notons que *defaction* sert à identifier la commande pour la création des actions.

4.2.3.3. Construction d'un plan

Un plan regroupe un ensemble d'action. Dès lors, il est nécessaire de pouvoir les ordonner afin de voir dans quel ordre les exécuter. Dans PDL, nous considérons trois cas d'ordonnancement des actions, tout d'abord séquentiellement avec l'opérateur « *SEQ* », de manière antinomique avec l'opérateur « *CHC* » ou encore de manière aléatoire avec l'opérateur « *COL* ». Les exemples suivants illustrant les interprétations des opérateurs listés ci-dessus :

- *SEQ a b c* : Il est nécessaire d'effectuer d'abord l'action *a* puis l'action *b* et finalement l'action *c*.
- *CHC* a b c : Il est nécessaire d'exécuter au choix seulement une et une seule action, soit l'action a ou l'action b ou l'action c.
- *COL* a b c : Il est nécessaire d'exécuter les actions a, b, c, quel que soit l'ordre, mais il faut toutes les exécuter.

Ainsi un plan comprendra dans sa description, le nom du plan et la liste d'action ordonnée à l'aide des opérateurs que nous venons de voir. La syntaxe PDL est donc comme suit :

(*defplan nom_action*(*seq action1 action2*)))

(*defplan nom_action*(*chc action1 action2*)))

(*defplan nom_action*(*col action1 action2*)))

De la même manière que, pour les actions, *defplan* sert à identifier la commande pour la création des plans. Il est à noter que les plans que l'on peut définir comme composés sont des plans qui comportent plusieurs combinaisons d'opérateur d'ordonnancement des actions telles que *SEQ*, *COL*, *CHC*. L'ordonnancement des sous plans s'effectuant de la même manière que pour les actions.

4.2.4. Subsomption d'actions

Comme nous l'avons précisé auparavant dans le modèle de la première partie de ce chapitre, pour identifier une relation de subsomption entre deux concepts, il faut les comparer « composante par composante ». Ainsi pour étudier la subsomption entre deux actions, il est nécessaire de vérifier s'il existe un subsumant entre les deux listes de faits

initiaux, de même pour la liste de faits ajoutés et pour la liste de faits supprimés. Autrement dit, pour vérifier si une action *a* subsume une action *b*, il faut s'assurer que pour chaque fait, d'une liste de faits de l'action *b*, il existe, dans la liste correspondante de fait, de l'action *a*, au moins un fait équivalent ou plus général. Pour mieux cerner la méthode de subsomption d'actions, l'algorithme ci-dessous est appliqué pour les faits initiaux :

Pour tous les faits initiaux f_i de a
    Pour tous les faits initiaux f_j de b
        Si l'un des faits f_j de b exprime le même fait que f_i de a (si $f_i == f_j$) OU
        Si l'extension du fait f_i est incluse par l'extension du fait f_j (si f_i subsume f_j)
            Alors retourner vrai
    Fpour
Fpour
Retourner faux

Bien entendu, le même algorithme est appliqué pour les listes de faits ajoutés et les faits supprimés.

4.2.5. Représentation des plans à base d'automate

Un plan peut être perçu comme un ordonnancement déterministe des actions. Ainsi, en nous inspirant de la théorie des automates, il possible de dire qu'un plan peut correspondre à des expressions régulières. En outre, un langage est dit *régulier* si et seulement si celui-ci est reconnu par un automate [Sipser 1997]. Par conséquent, nous pourrons considérer le postulat tel que les plans exprimés en PDL puissent être représentés par des automates finis déterministes dont les actions seront les transitions entre des états fictifs. Une transition étant un passage d'un état de l'automate vers un autre en fonction du test d'un élément du

langage. Donc, dans une première étape, nous allons décrire tout d'abord l'équivalence entre un plan et un automate, pour les opérateurs *SEQ*, *COL* et *CHC*. Puis nous décrirons le calcul de la subsomption entre les plans en se basant sur les équivalences en automates.

4.2.5.1. Equivalence « *SEQ* » / automate

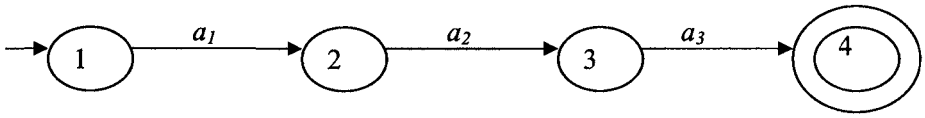
Dans la théorie des automates, un langage reconnu par un automate est un langage régulier, il existe donc une expression régulière qui le décrit [Sipser 1997]. Nous avons posé le postulat suivant : « un plan exprimé en PDL peut être représenté par des automates finis déterministes (modèle état-transition) ». Dès lors, pour un plan composé uniquement d'éléments ordonnés en séquence, P_1 : ($SEQ\ a_1\ a_2\ a_3$), il est possible de former l'expression régulière suivante : $(a_1).\ (a_2).\ (a_3)$. Nous pouvons donc former l'automate correspond à P_1 :

$$\{(1, 2, 3, 4)\ ;\ (a_1, a_2, a_3),\ \delta\ ,\ 1\ ,\ (4)\}$$

- L'ensemble Q des états de l'automate étant : (1, 2, 3, 4)
- L'ensemble Σ fini des actions étant : (a_1, a_2, a_3)
- q_0 symbolisant l'état de départ et valant 1
- l'état final : (4), étant donné que les actions sont ordonnées séquentiellement, a_3 ne peut être que la dernière.
- δ : Q x $\Sigma \longrightarrow$ Q, la fonction de transition de l'automate permettant de déterminer le prochain état en utilisant l'état actuel et l'action courante, celle-ci peut être représentée graphiquement par la matrice suivante :

	a_1	a_2	a_3
1	2		
2		3	
3			4

Sous une forme graphique, cela donne l'automate suivant :



On peut constater que les états sont des états fictifs, uniquement dans le but d'avoir le squelette de l'automate, les transitions étant les actions définies dans la séquence.

4.2.5.2. Equivalence « *CHC* » / automate

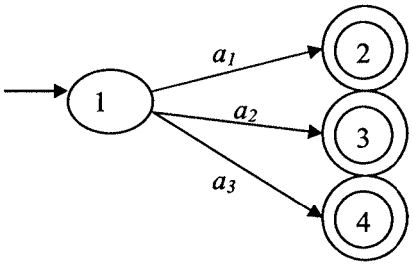
Pour un plan composé uniquement d'éléments ordonnés en choix, tel que P_2 : (*CHC* a_1 a_2 a_3), il est possible de former l'expression régulière suivante : (a_1) / (a_2) / (a_3). De la même manière que pour l'opérateur « *SEQ* », il est possible de former l'automate suivant, correspondant à cette expression régulière :

$\{(1, 2, 3) ; (a_1, a_2, a_3), \delta, 1, (3)\}$

- L'ensemble des états de l'automate étant : (1, 2, 3, 4)
- L'ensemble fini des actions étant : (a_1, a_2, a_3)
- q_0 symbolisant l'état de départ et valant 1
- l'ensemble des états finaux : (2, 3, 4) étant donné que a_1, a_2, a_3 sont disjoints, elles sont, dès lors, toutes respectivement finales.
- δ la fonction de transition de l'automate, représenté graphiquement par la matrice suivante :

	a_1	a_2	a_3
1	2	3	4

Sous une forme graphique, cela donne l'automate suivant :



4.2.5.3. Equivalence « *COL* » / automate

Pour un plan composé uniquement d'éléments ordonnés en collection, tel que P_3 : (*COL* a_1 a_2 a_3), il est possible de former l'expression régulière suivante : $\{((a_1).(a_2).(a_3)) / ((a_1).(a_3).(a_2)) / ((a_2).(a_1).(a_3)) / ((a_2).(a_3).(a_1)) / ((a_3).(a_2).(a_1)) / ((a_3).(a_1).(a_2))\}$. Dès lors, l'automate correspondant à cette expression régulière est le suivant :

$\{(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16) ; (a_1, a_2, a_3), \delta , 1 , (3)\}$

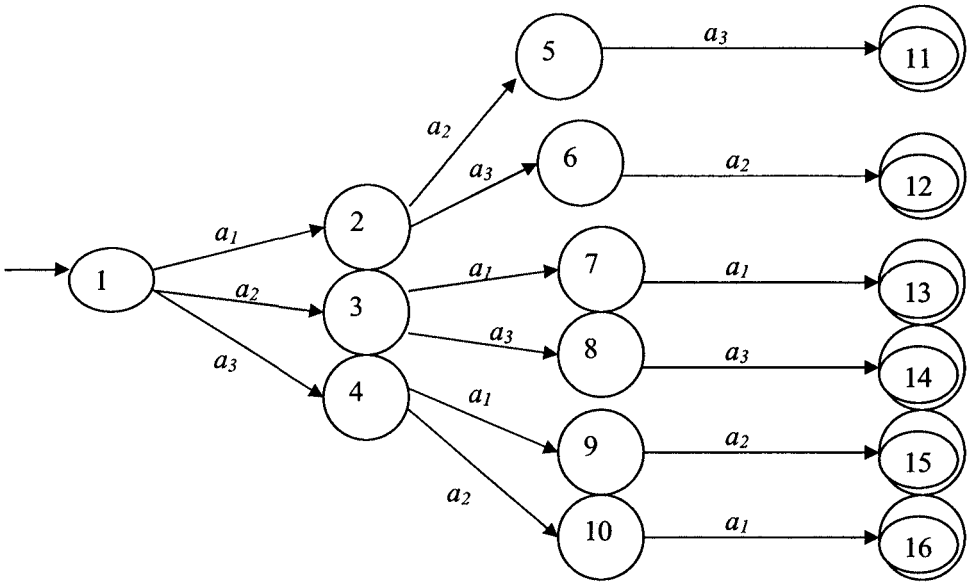
- L'ensemble des états de l'automate étant :

  (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)

- L'ensemble fini des actions étant : (a_1, a_2, a_3)
- q_0 symbolisant l'état de départ et valant 1
- l'ensemble des états finaux : (11, 12, 13, 14, 15, 16)
- δ la fonction de transition de l'automate, représenté graphiquement par la matrice suivante :

Liste des transitions restantes	Etats	a_1	a_2	a_3
$\{a_1, a_2, a_3\}$	1	2	3	4
$\{a_2, a_3\}$	2		5	6
$\{a_1, a_3\}$	3	7		8
$\{a_1, a_2\}$	4	9	10	
$\{a_3\}$	5			11
$\{a_2\}$	6		12	
$\{a_1\}$	7	13		
$\{a_3\}$	8			14
$\{a_2\}$	9		15	
$\{a_1\}$	10	16		

Sous une forme graphique, cela donne l'automate suivant :



4.2.5.4. Equivalence plan composé / automate

Faire la correspondance entre un plan composé et un automate, revient à considérer que l'automate correspondant aura certains états considérés comme des « sous-automates ». En s'inspirant de l'exemple d'application présent dans la théorie de Kautz, nous pouvons considérer l'exemple suivant. Soit A_1 l'automate représentant le choix pour faire à manger dont son regroupement d'action est (*chc Recette_Base_Pates Recette_Base_Viande*) et A_2 l'automate concernant la partie pour faire une sauce aux fruits de mer dont son regroupement est (*seq Faire_sauce_fruits_de_mer*). Le plan *Faire_Manger_sauce_fruits_de_mer* est alors défini en PDL comme suit :

(*defplan Faire_Manger_sauce_fruits_de_mer* ((*chc Recette_Base_Pates Recette_Base_Viande*) (*seq Faire_sauce_fruits_de_mer*)))

Le convertir en automate reviendra à convertir le plan suivant :

$$(seq\ A_1\ A_2)$$

Ainsi, l'expression régulière de cet automate que l'on peut former serait la suivante :

(Recette_Base_Pates / Recette_Base_Viande). Faire_sauce_fruits_de_mer

De même, l'automate correspondant à cette expression sera le suivant :

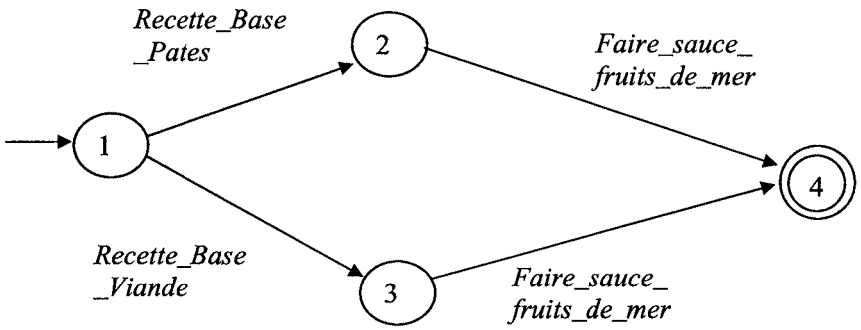
{(1,2,3,4) ; (*Recette_Base_Pates*, *Recette_Base_Viande*, *Faire_sauce_fruits_de_mer*), δ, 1 , (4)}

L'ensemble des états de l'automate étant : (1,2,3,4)

- L'ensemble fini des actions étant : (*Recette_Base_Pates*, *Recette_Base_Viande*, *Faire_sauce_fruits_de_mer*)
- q_0 symbolisant l'état de départ et valant 1
- l'état final : (4)
- δ la fonction de transition de l'automate, représenté graphiquement par la matrice suivante :

	Recette_Base_Pates	*Recette_Base_Viande*	*Faire_sauce_fruits_de_mer*
1	2	3	
2			4
3			4

Sous une forme graphique, cela donne l'automate suivant :



4.2.5.5. Subsomption

La relation de subsomption entre deux plans, consiste à s'assurer qu'il existe une inclusion entre les deux langages réguliers représentant les plans correspondants. En effet, pour voir si un plan P_1 subsume un plan P_2, il est nécessaire de regarder si L_2 le langage régulier représentant P_2 est inclus dans L_1 le langage régulier représentant P_1, comme nous le montre la figure suivante :

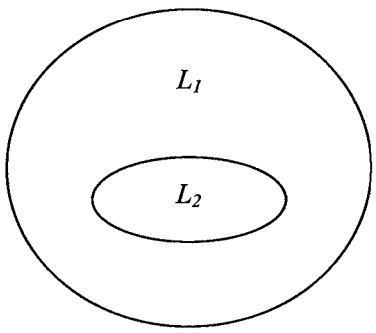


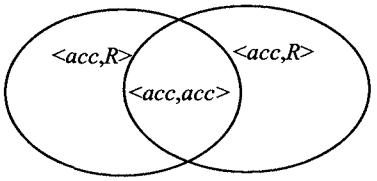
Figure 12 : Subsomption entre L_1 et L_2

Dès lors, la relation de subsomption peut se ramener à tester l'union des langages L_1 et L_2, dans les deux automates respectifs A_1 et A_2. Ces deux automates représentant respectivement les plans P_1 et P_2. Soit L le langage union de L_1 et L_2. Si nous testons L dans A_1 puis dans A_2, nous aurons trois cas possibles, A_1 et A_2 acceptent tous les deux l'action de L testée ou alors A_1 rejette mais A_2 accepte l'action ou encore A_2 rejette mais A_1 l'accepte. Il est possible de symboliser la situation par le schéma suivant où *acc* signifie que l'automate correspondant accepte et *R* qu'il rejette :

<*acc*,*R*> signifie que A_1 accepte et A_2 rejette l'action.

<*acc*,*acc*> signifie que A_1 accepte et A_2 accepte l'action.

<*R*,*acc*> signifie que A_1 rejette et A_2 accepte l'action.



En testant L dans les deux automates, si on obtient des valeurs <*acc*,*R*> et <*acc*,*acc*> mais pas de valeur <*R*,*acc*> alors on peut dire que L_2 est inclus dans L_1. Au contraire, si on obtient des couples <*acc*,*acc*> et <*R*,*acc*> alors L_1 est inclus dans L_2.

Par ailleurs, les matrices de transitions sont très utiles pour étudier le comportement d'un automate sur son langage. Néanmoins, cela n'est pas suffisant pour nos besoins. Or, nous cherchons à voir quand un automate accepte un élément de L et quand l'autre automate le refuse. Ainsi, il est nécessaire de fusionner les matrices correspondantes dont chaque élément du tableau est un couple d'état d'arrivée pour les deux automates en fonction d'une action de L. Les transitions étant calculées en fonction des deux automates,

en plaçant le couple formé par les états cibles, des deux automates, $< E_{A_1}, E_{A_2} >$, où E_{A_1} et E_{A_2} forment les états cibles pour deux automates A_1 et A_2 quelconque. Ce couple est construit comme suit : pour l'automate A_1, $\delta(D,x)= E_{A_1}$ et pour l'automate A_2, $\delta'(D,x)= E_{A_2}$; D est un état non-terminal quelconque, x un élément quelconque du langage et δ et δ' étant respectivement les fonctions de transitions des automates A_1 et A_2. S'il n'existe pas d'état d'arrivée pour une transition alors nous prenons R pour symboliser le rejet.

Ainsi, comme nous l'avons précisé au préalable, la subsomption entre deux plans peut se calculer en testant l'union des deux langages réguliers correspondant dans chacun des automates. Dès lors, il est possible de dégager deux situations. La première concerne des plans comportant respectivement les mêmes actions, qui se traduit par une évaluation au niveau de l'organisation des actions dans le plan. Et la deuxième situation, lorsque les plans n'ont pas les mêmes actions, le calcul consiste non seulement à considérer les actions, mais aussi leurs organisations.

4.2.5.5.1. Cas des plans similaires

Considérons l'exemple suivant. Soit P_1 le plan défini à la section 4.2.5.1 et P_2 le plan défini à la section 4.2.5.2. Nous allons calculer la subsomption entre P_1 ($SEQ\ a_1\ a_2\ a_3$) et P_2 ($CHC\ a_1\ a_2\ a_3$)

Tout d'abord, voici les matrices de transition correspondantes :

P_1 :

	a_1	a_2	a_3
1	2		
2		3	
3			4

P_2 :

	a_1	a_2	a_3
1	2	3	4

Nous pouvons donc former la matrice de fusion suivante :

	a_1	a_2	a_3
1	<2,2>	<R,3>	<R,4>
2	<R,R>	<3,R>	<R,R>
3	<R,R>	<R,R>	<4,R>

En étudiant les résultats, il est possible de voir que nous obtenons des couples de la forme : <*R*,*acc*>, <*acc*,*R*>, où *acc* correspond à un état acceptant de l'automate correspondant et *R* un état rejetant. Dès lors, comme nous l'avons précisé précédemment, il est possible d'en déduire que P_1 ne subsume pas P_2. Il est même possible de constater que les deux langages sont disjoints.

4.2.5.5.2. Cas des plans possédant des actions différentes

Lorsque le langage est identique, pour une même transition dans deux automates différents, seuls les états d'arrivée vont changer. Or, pour un langage différent, une

transition à partir d'un même état risque de provoquer un rejet lorsque l'on teste deux actions différentes. En somme, s'il existe une subsomption entre deux actions, cela doit être pris en compte dans la transition. Dès lors, juste calculer la matrice de fusion comme il a été décrit à la section 4.2.5.5.1 n'est pas suffisant. Pour illustrer ceci, considérons l'exemple suivant, soit deux plans, P_1 (*defplan P_1(seq a b c)*) et P_2 (*defplan P_1(seq a d d)*), *a.b.c* étant l'expression régulière de P_1. L'automate correspondant à cette expression régulière serait le suivant :

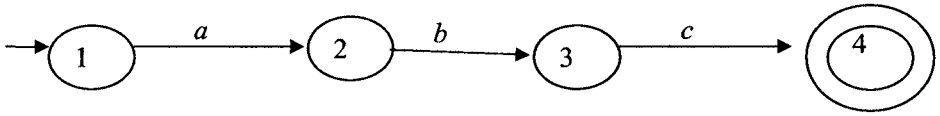
$$\{(1,2,3,4) ; (a,b,c), \delta , 1 , (4)\}$$

- L'ensemble des états de l'automate étant : (1,2,3,4)
- L'ensemble fini des actions étant : (*a*,*b*,*c*)
- q_0 symbolisant l'état de départ et valant 1
- l'ensemble des états finaux : (4)
- δ la fonction de transition de l'automate, représenté graphiquement

	a	*b*	*c*
1	2		
2		3	
3			4

par la matrice suivante :

Sous une forme graphique, cela donne l'automate suivant :



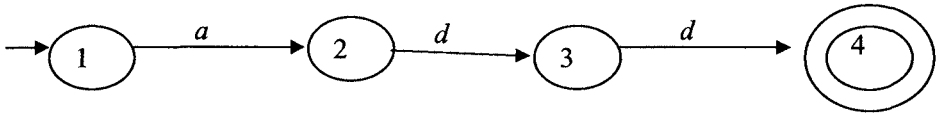
Pour *P2* l'expression régulière est la suivante : *a.d.d* .L'automate correspondant à cette expression régulière serait le suivant :

$$\{(1,2,3,4) ; (a,d),\ \delta , 1 , (4)\}$$

- L'ensemble des états de l'automate étant : (1,2,3,4)
- L'ensemble fini des actions étant : (*a*,*d*)
- q_0 symbolisant l'état de départ et valant 1
- l'ensemble des états finaux : (4)
- δ la fonction de transition de l'automate, représenté graphiquement par la matrice suivante :

	a	*d*
1	2	
2		3
3		4

Sous une forme graphique, cela donne l'automate suivant :



Donc pour étudier la subsomption entre P_1 et P_2, il est nécessaire de construire la matrice de fusion. Tout d'abord, voici les matrices de transition correspondantes :

Pour *P1* :

	a	*b*	*c*
1	2		
2		3	
3			4

Pour *P2* :

	a	*d*
1	2	
2		3
3		4

Nous pouvons à partir de ces deux matrices appliquées former la matrice de fusion suivante :

	a	*b*	*c*	*d*
1	<2,2>			
2		<2,R>	<3,R>	<R,3>
3			<4,R>	<R,4>

En regardant les couples obtenus, on constate que ceux-ci sont de la forme, *<acc,R>* *<R,acc>* donc la conclusion logique reviendrait à dire que P_1 est disjoints de P_2, mais on ne peut rien déduire sur leur inclusion respective.

Considérons le cas où l'action *c* subsume l'action *d*. Dès lors, il est possible d'admettre le postulat suivant : Si pour deux actions quelconques *c* et *d*, que *c* subsume *d* et pour deux automates A_1 et A_2 représentants les plans contenant *c* et *d*, le plan de A_1 ne contenant pas *d* et le plan de A_2 ne contient pas *c*. Alors il est possible d'admettre qu'une transition de l'automate qui accepte *d*, acceptera aussi *c*. Dans la matrice de fusion, pour la transition *d*, le couple sera formé par *<x,y>*, *x* étant l'état d'arrivée par la transition *c* et *y* l'état d'arrivée par la transition *d*. En effet, comme nous l'avons précisé dans le cas de la subsomption d'action, le $dom(d) \subseteq dom(c)$ et le $CoDom(d) \subseteq CoDom(c)$, ceci nous permettra d'admettre que ce qui sera accepté par la transition *d* sera aussi accepté par la transition *c*.

Si nous reprenons la matrice de fusion précédente, mais que nous effectuons les changements en tenant compte de l'information que *c* subsume *d*, ces changements seront encadrés :

	a	*b*	*c*	*d*
1	<2,2>			
2		<2,R>	<3,R>	<2,3>
3			<4,R>	<2,4>

(Les modifications étant encadrées).

Dès lors, si nous regardons la forme des couples obtenus : <*acc*,*acc*> et <*acc*,*R*> nous pouvons en déduire que P_1 subsume P_2

4.2.6. Architecture de PDL

Le langage PDL est équipé d'algorithmes permettant la classification des nouveaux concepts d'actions et de plans. Ceux-ci suivent les règles présentées dans le modèle formel de la première partie de ce chapitre pour déterminer les relations de subsomption entre les concepts. La Figure 13 nous montre le schéma général de l'architecture du langage développé. Comme nous l'avons précisé précédemment, le langage encapsule PowerLoom celui-ci possédant ainsi deux moteurs d'inférences. Le moteur d'inférence du langage s'occupe de tout ce qui a trait à la gestion, le stockage et l'interrogation des concepts d'actions et de plans. De plus, il se chargera d'invoquer le moteur d'inférence de PowerLoom lorsqu'il aura affaire à des concepts au sens de la logique de description, afin d'en laisser la gestion à celui-ci. Cette façon de procéder comporte beaucoup d'avantages, dont celui de pouvoir réutiliser l'algorithme de vérification de relations de subsomption de

PowerLoom pour comparer les concepts standard. Le SRCT PowerLoom nous fournit une base solide et nous permet d'implémenter ce langage d'une façon beaucoup plus simple et efficace que si nous avions démarré l'implémentation à zéro. De plus, l'application est munie d'un interpréteur de langage PDL qui permet de valider les expressions correctes de celui-ci et de les traduire en structures de données correspondantes afin que celles-ci puissent être traitées par son moteur d'inférence. Finalement, une interface est disponible à l'utilisateur afin que celui-ci puisse créer des plans et en vérifier la subsomption avec ceux qui sont contenus dans la taxonomie de plan. Si un plan de l'utilisateur est valide alors il est ajouté dans la taxonomie.

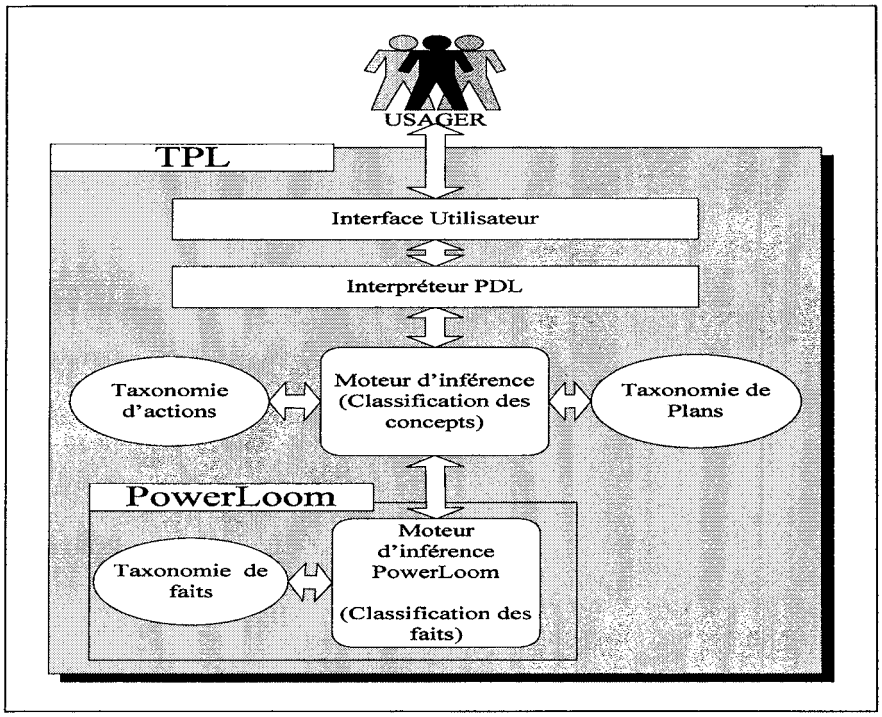


Figure 13 : Description générale de PDL.

On peut voir à la Figure 14 une présentation de l'application servant d'interface avec l'utilisateur. Notons que nous nous sommes inspirés de l'exemple d'application présent dans les travaux de Kautz. Cette interface comporte une partie à droite contenant toutes les actions de la librairie. Une partie en bas à gauche, symbolisant le plan de l'utilisateur. La partie en haut à gauche symbolise la liste des plans subsumant le plan de l'utilisateur. Lorsque ce dernier a fini de rentrer son plan, il peut cliquer sur le bouton « *Verification subsomption* » afin de voir si son plan subsume un plan de l'agent. Si l'utilisateur souhaite modifier la librairie, il peut le faire à l'aide du bouton « *Verification subsomption* » lui donnant la possibilité de saisir une commande PDL venant modifier la librairie.

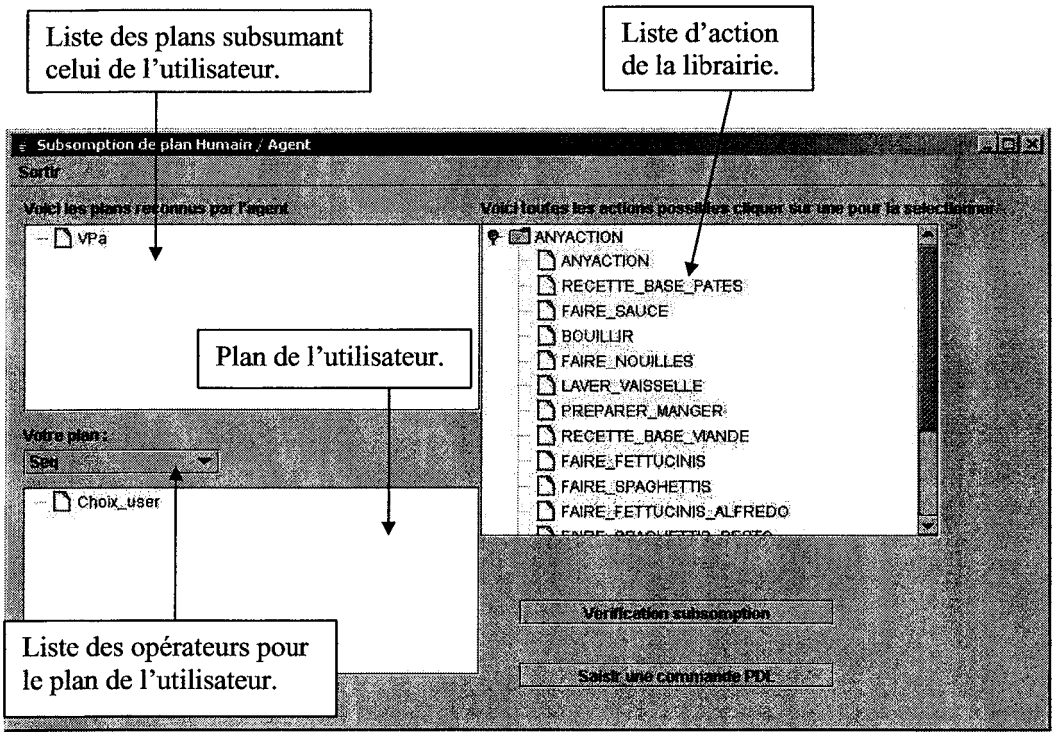


Figure 14 : Interface de l'application.

4.2.7. Powerloom

PowerLoom est le successeur du système de représentation de la connaissance Loom [Loom, 1991], disponible en version Java ou C++. Il fournit un langage et un environnement permettant de construire des applications de classification, dotées de base de connaissances terminologique [PowerLoom, 1997]. PowerLoom emploie un formalisme dérivé de celui de Loom, tel qu'on peut le voir sur la Figure 15. Son moteur d'inférence utilise un modèle de déduction par chaînage avant [Farreny et Ghallab, 1987] qui peut manipuler des règles et des relations complexes, telles les relations de subsomption. PowerLoom est également muni d'un système de classification avancé, permettant de situer avec beaucoup de précision un nouveau concept dans une hiérarchie déjà existante. Il s'intègre facilement à une autre application, comme un composant permettant la création de structure hiérarchique de concepts. Cette intégration s'effectue grâce à une API (Application Programming Interface) permettant de créer et de manipuler la base de connaissances PowerLoom via un autre logiciel. PowerLoom devient ensuite une partie intégrante de l'application.

LOOM	*PowerLoom*
(defconcept ETRE :is :primitive) *(defconcept ENSEMBLE :is :primitive)* *(defconcept HUMAIN :is (:and ETRE :primitive))* *(defconcept HOMME :is (:and HUMAIN :primitive))*	*(defconcept ETRE (?x))* *(defconcept ENSEMBLE (?x))* *(defconcept HUMAIN (?x ETRE))* *(defconcept HOMME (?x HUMAIN))*
(defconcept FEMME :is (:and HUMAIN *(:not HOMME) :primitive))*	*(defconcept FEMME (?x HUMAIN):<=>* *(and (not (HOMME ?x))))*
(defconcept ROBOT :is (:and ETRE *(:not HUMAIN) :primitive))*	*(defconcept ROBOT (?x ETRE):<=>* *(and (not (HUMAIN ?x))))*
(defrelation membre :is :primitive)) *(defrelation chef :is (:and membre :primitive))*	*(defrelation membre ((?x) (?y)))* *(defrelation chef ((?x) (?y)):<=>* *(and (membre ?x ?y)))*
(defconcept EQUIPE :is (:and ENSEMBLE *(:all membre ETRE)* *(:at-least 2 membre)))*	*(defconcept EQUIPE (?x):<=>* *(and (ENSEMBLE ?x)* *(all (membre ?x ?y) (ETRE ?x))* *(at-least 2 (membre ?x ?y))))*
(defconcept PETITE-EQUIPE :is (:and EQUIPE *(:at-most 5 membre)))*	*(defconcept PETITE-EQUIPE (?x):<=>* *(and (EQUIPE ?x)* *(at-most 5 (membre ?x ?y))))*
(defconcept EQUIPE-MODERNE :is (:and EQUIPE *(:at-most 4 membre)* *(:at-least 1 chef)* *(:all chef ROBOT))*	*(defconcept EQUIPE-MODERNE (?x):<=>* *(and (EQUIPE ?x)* *(at-most 4 (membre ?x ?y))* *(at-least 1 (chef ?x ?w))* *(ROBOT ?w)))*

Figure 15 : Comparaison des formalismes de LOOM et PowerLoom.

On peut noter sur la Figure 15, les différences entre le formalisme de LOOM et celui de son successeur PowerLoom. On peut voir que l'introduction de la définition du concept se fait grâce au symbole «<=>», contrairement au constructeur *is* de LOOM. D'autre part, on note qu'à chaque introduction d'un concept ou d'un rôle, on doit inclure des variables représentant les instances possibles de ceux-ci. Ces variables servent de références lors de la définition des concepts et sont désignées par le symbole « ? ». Outre ces quelques différences, on peut voir que les deux formalismes sont très similaires.

4.2.8. Reconnaissance de plan

Dans l'application, les plans sont formés à l'aide de la librairie. En effet, l'utilisateur en sélectionnant une action sur la liste proposée a ainsi à la possibilité de choisir les actions faisant partie de son plan. En outre, une liste est définie au-dessus de la partie symbolisant son plan afin de choisir l'opérateur d'ordonnancement comme le montre la Figure 16

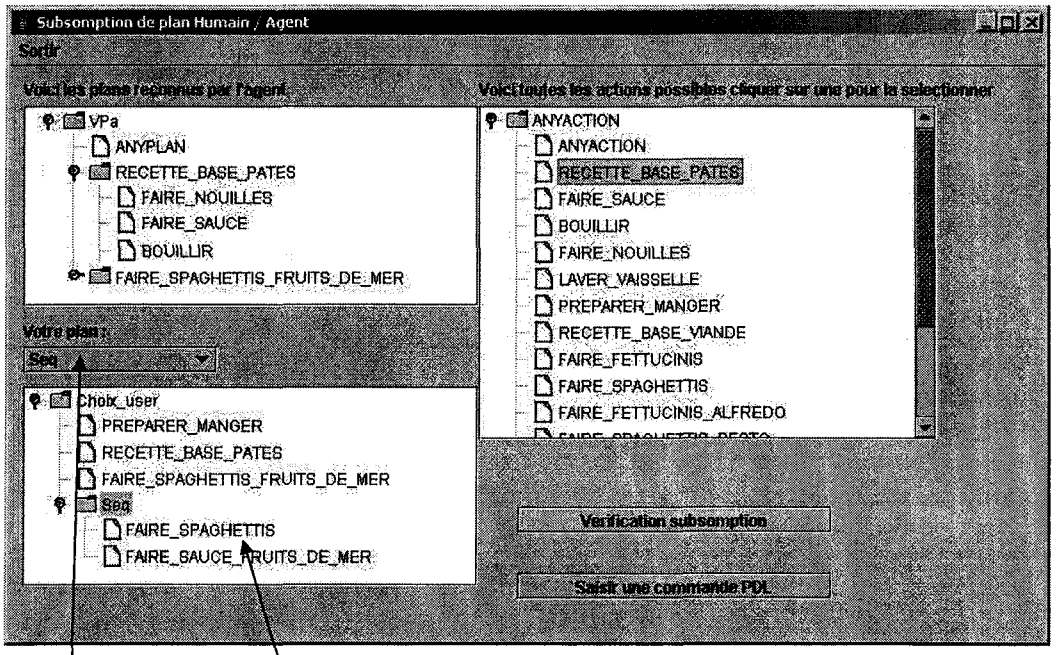


Figure 16 : Exemple de construction de plan.

Ainsi, les plans sont représentés sous forme d'arbre permettant ainsi à l'utilisateur de visualiser le plan qu'il construit. L'opérateur au-dessus de la fenêtre, symbolisant le plan en

cour de construction, porte sur l'ensemble de celui-ci. Il est bien entendu possible de placer des sous plans comme nous le montre le Tableau 3. Il est à noter que lors de la construction, le plan de l'utilisateur est nommé par défaut *temp* mais si celui-ci comporte une syntaxe correcte, confirmé par l'interpréteur de PDL, l'utilisateur est invité à changer son nom.

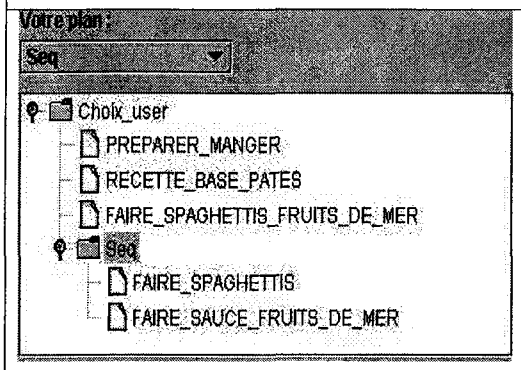
Structure en JAVA	*Définition PDL*
	*(deffplan temp(Seq PREPARER_MANGER RECETTE_BASE_PATES FAIRE_SPAGHETTIS_FRUITS_DE_MER (Seq FAIRE_SPAGHETTIS FAIRE_SAUCE_FRUITS_DE_MER))*

Tableau 3 : Equivalence d'un plan en langage PDL.

Concernant la formation des actions, celle-ci s'effectue en appuyant sur le bouton « Saisir une commande PDL » faisant apparaître une fenêtre comme nous le montre la Figure 17 permettant de taper une commande PDL, soit pour créer une nouvelle action, un nouveau fait etc. Bien entendu, les syntaxes étant vérifiées par l'interpréteur de PDL.

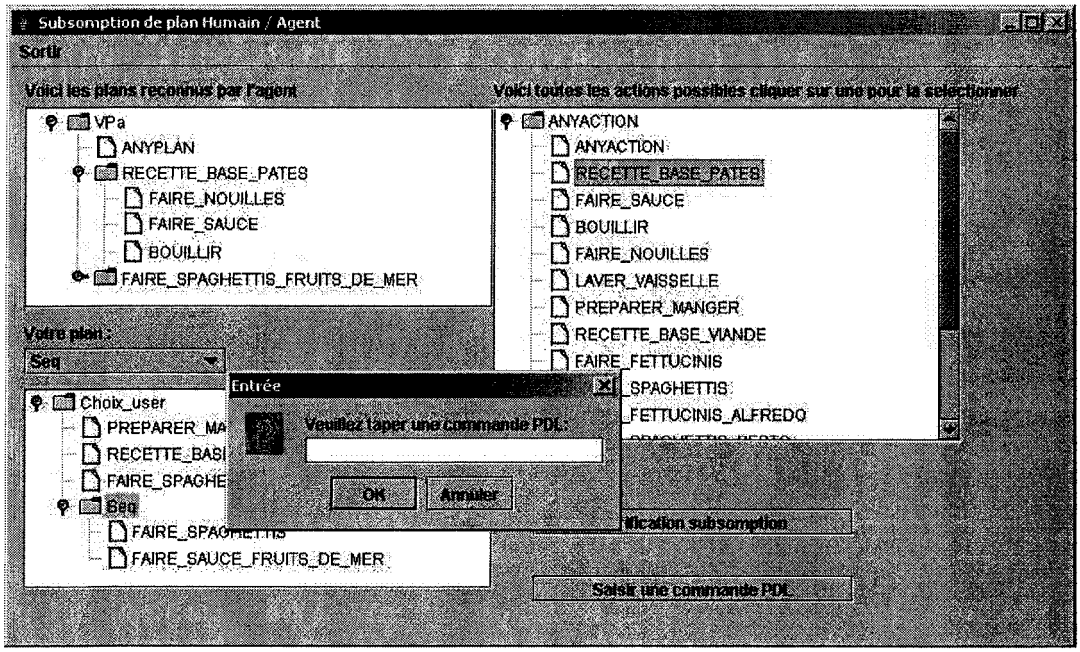


Figure 17 : Commande PDL.

La subsomption de plan s'effectuant de manière très transparente. En effet, à chaque action sélectionnée, la liste des plans subsumant est formée. Ceci permettant de voir les plans possibles pour une action. La Figure 18, nous montre la liste de plans subsumant pour le plan *(deffplan temp(Seq PREPARER_MANGER RECETTE_BASE_PATES FAIRE_ SPAGHETTIS_FRUITS_DE_MER (Seq FAIRE_SPAGHETTIS FAIRE_SAUCE_FRUITS_ DE_ MER))*

Figure 18 : Exemple de plans possibles.

La subsomption de plan que nous avons introduite permettra de classifier l'ensemble des plans possibles interprétant une action observée dans une structure d'ordre facilitant le choix d'un plan plausible parmi les plans possibles. Ce travail de raffinement en plans plausibles est en cours de développement et qui s'inscrit dans le cadre d'un travail de doctorat.

4.3. Conclusion

Au cours de ce chapitre, nous avons présenté un modèle théorique pour tenter de résoudre le problème de la formalisation de l'action en logique de description. Puis nous avons vu comment exprimer la subsomption d'action. Nous avons expliqué le langage PDL et décrit son implantation, inspirée de ce modèle. Cette étude est placée comme une extension de la logique de description, utilisant le SRCT Powerloom. L'application utilise la subsomption d'action, mais aussi une extension au modèle décrit dans la première partie de ce chapitre, c'est-à-dire la subsomption de plan, librement inspiré de la théorie des automates finis.

Les différents éléments présentés ici permettent de faciliter la reconnaissance de plan. En effet, nous avons défini cette tâche comme un processus de compréhension d'une série d'actions. Pour atteindre notre but de définir les fondements de la reconnaissance de plan, il était donc nécessaire d'avoir une définition formelle d'une action en logique de description. Cette logique permet justement de prévenir une grande partie des problèmes de compréhension liés aux conflits terminologiques. Ainsi, les travaux présentés dans ce mémoire, se veulent comme une étape importante avant la reconnaissance de plan.

Le langage développé ici peut se voir comme le précurseur au développement d'agents capable de saisir le comportement des entités en action en les observants. Ceci, afin de faire d'eux, de véritables collaborateurs mais aussi d'augmenter le niveau de confiance entre les deux dans un contexte de coopération, tel le pilotage automatique d'avions ou l'assistance aux personnes à mobilité réduite. Le langage PDL peut être l'initiateur à un modèle de reconnaissance de plan basé sur la logique de description, pour répondre à la problématique énoncée en introduction. La subsomption de plan donnera la possibilité de classifier les plans possibles afin de construire, une structure de plan permettant de choisir le plan plausible, qui a son tour entraînera la compréhension du comportement de l'agent observé.

CHAPITRE 5

CONCLUSION GENERALE

Ce travail s'est focalisé sur un aspect de la problématique de la coopération humain agent, c'est-à-dire celui où l'agent est dans une situation d'observateur et où il doit comprendre les actions de l'usager lors de la réalisation d'une tâche, en d'autre termes, la reconnaissance de plan. Cette problématique consiste à ramener la tâche de reconnaissance à un problème de classification utilisant la logique de description.

Ainsi dans ce mémoire, nous avons défini de manière informelle la reconnaissance de plan. Celle-ci peut se résumer à inférer le but poursuivi par un acteur à travers un ensemble d'actions qu'il effectue. Ensuite il s'agit de construire une séquence d'action faisant office de plan expliquant ce dit but. Cette construction peut se faire par le biais de différents moyens, dont par exemple la construction d'hypothèses formulées en logique du premier ordre [Kautz, 1987] ou bien l'utilisation de la théorie des situations à base d'opérations non monotoniques pour choisir un plan plausible parmi les plans possibles interprétant une action [Wobcke, 2002]. Nous avons, ensuite, effectué un portrait de la logique de description, celle-ci comportant de nombreux avantages pour la représentation de la connaissance ou le raisonnement par classification. Par la suite, nous avons présenté un modèle de l'action en logique de description. De même, nous avons décrit le langage PDL, comme support à la reconnaissance de plan à base de la logique de description.

Dans ce contexte, notre premier objectif était de comprendre la problématique de la reconnaissance de plan, c'est-à-dire de dégager une définition de celle-ci, de la compréhension à la fois des éléments, mais aussi des opérations essentielles à ce processus,

ceci dans le but de formaliser les éléments fondateurs de cette tâche et d'ouvrir la voie à d'autres recherches sur les modalités de la reconnaissance de plan. Dès lors, la réalisation de ce but a permis de démontrer qu'à travers la librairie de plans, le concept d'actions est au cœur même de cette problématique. Notre second objectif fut de créer un modèle formel d'actions en logique de description, permettant la classification des actions, afin de pouvoir employer les capacités de la logique de description pour la reconnaissance de plan. Ce but a été accompli après une étude de la logique de description. Notre dernier objectif fut de valider notre approche théorique, par la réalisation d'un exemple d'application montrant la faisabilité de celle-ci en s'appuyant sur le même exemple traité par Kautz.

La compréhension de la reconnaissance de plan a été menée à bien grâce à une revue de littérature sur des travaux portant sur différents domaines d'applications. Tout d'abord, dans le domaine des systèmes multi-agents [Wooldridge et Jennings, 1995], puis avec les travaux portant sur les ontologies et la classification de concept de Gruber [TR Gruber, 1993]. Par la suite, nous nous sommes focalisé, sur les travaux de Kautz [Kautz, 1987], Wobcke [Wobcke, 2002], Carberry [Carberry, 2000] et en passant par l'étude de l'initiative mixte de Cohen [Cohen *et al*, 1998]. Nous avons d'ailleurs choisi de nous concentrer sur les approches non probabilistes, car celles-ci s'impliquent au mieux dans nos objectifs sur le long terme. En effet, comme nous l'avons souligné en introduction, l'approche probabiliste soulève un certain nombre de limites telles que le calcul de la probabilité d'un plan vis-à-vis d'un autre, la nécessité de disposer de tous les plans possibles, quelle que soit l'observation effectuée [Carberry, 2000], alors que dans les travaux basés sur la logique,

ces dits plans sont formés à partir d'une observation, évitant ainsi d'avoir à les créer tous. Par ailleurs, ce choix est conforme à notre ambition qui est de fournir un modèle servant de fondation au processus de la reconnaissance de plan, basé sur la logique description. En outre, dans un futur proche, ce prototype permettra de formaliser le processus d'inférence de la reconnaissance de plan en un modèle de raisonnement par classification. C'est pourquoi nous avons fait le choix de nous concentrer essentiellement sur les modèles de reconnaissance de plan basés sur la logique de description.

L'étude détaillée de la logique de description a été faite en passant en revue la notion de connaissance par les travaux de Davis [Davis *et al*, 1993], suivie d'une recherche concernant l'aspect théorique et les éléments de base de la logique de description grâce aux travaux de Baader et Nutt [Baader et Nutt, 2002], pour finalement terminer avec une comparaison des SRCT disponibles à l'heure actuelle [PowerLoom, 1997]. Nous avons conclu qu'il était plus efficace d'utiliser la logique de description, qui se prête mieux au contexte de cohabitation, afin de ramener le problème de la reconnaissance de plan à un problème de classification. De même, le fait de pouvoir classifier les concepts dans une structure hiérarchique permet de diminuer le temps nécessaire à cette opération. Un dernier élément intéressant étant que la logique de description ne supporte pas la notion d'action qui est un concept dynamique. Par conséquent, il a été indispensable d'introduire et d'étendre cette logique pour répondre à notre problématique de reconnaissance, en proposant un modèle d'action qui sera utilisé comme support à cette reconnaissance.

Ainsi, en étudiant la reconnaissance de plan, nous avons conclu que cela peut se définir informellement comme une tâche de compréhension des actions effectuées par l'usager. Il est donc clair que le concept d'action est au cœur même de ce processus. Dès lors, nous avons proposé un modèle formel d'actions en logique de description, en utilisant ses outils algébriques. Une telle démarche nous a permis d'ailleurs de définir un langage basé sur ce modèle. Par là même, nous y avons redéfini la relation de subsomption pour les actions, mais aussi pour les plans. Pour valider cette approche, nous avons créé une application permettant à l'usager, à travers l'interface de l'application, de proposer son plan afin de résoudre une tâche commune. Puis à chaque action saisie par l'usager, l'agent rassemble les plans possibles parmi l'ensemble de ceux qui sont contenus dans sa librairie. Les plans possibles étant ceux qui possèdent une action qui subsume l'action saisie par l'usager. Cette étape, indispensable à la reconnaissance de plan, permettra d'inférer tous les plans possibles, susceptibles, d'interpréter l'action observée. L'ensemble des plans possibles, ordonnés par la relation de subsomption (relation d'ordre), sera utilisé dans une structure d'entrée à une étape de raffinement, afin d'extraire un plan plausible parmi ces plans possibles. Cette dernière est en cours de réalisation et s'inscrit dans un travail de doctorat comme une suite naturelle à ce travail.

Par conséquent, ce mémoire ne se veut pas une réponse ferme et définitive à la problématique énoncée en introduction. Cependant l'approche proposée doit être considérée comme un pas en avant contribuant à l'évolution de la compréhension de la reconnaissance de plan et par là même de la coopération dans les systèmes multi-agents.

Notre approche théorique, axée sur la logique de description, ouvre la voie au développement de différentes « techniques » de reconnaissance de plan dans un vaste champ de domaine, comme l'assistance aux personnes à mobilité réduite, dans un contexte d'habitat intelligent. Ce projet d'assistance s'inscrit dans le cadre de la préparation du doctorat de Bruno Bouchard.

D'un point de vue personnel, ce travail m'a permis de m'initier à la recherche scientifique. Plus exactement, je me suis familiarisé au domaine de la coopération dans les systèmes multi-agents ainsi qu'à la logique terminologique. Tout au long des différentes étapes de réalisation, j'ai pu acquérir des connaissances qui me seront sûrement utiles durant ma future carrière. Je dois également souligner que cette expérience enrichissante me donne envie de continuer dans cette voie. Je trouve très gratifiant le fait d'avoir le privilège de pouvoir contribuer à l'amélioration des technologies de pointe dans le domaine des systèmes multi-agents.

Pour conclure nous terminerons par une citation de Ferber [Ferber, 1995] : « Simples outils d'aujourd'hui, collaborateurs de demain, les agents informatiques nous seront aussi indispensables qu'une calculette ou, pour certains, qu'un ordinateur portable. Ces agents, en voyant circuler des informations, en s'adaptant au fonctionnement des êtres humains (qui, en retour, s'accommoderont eux aussi des agents informatiques) se réorganiseront et se restructureront dynamiquement pour s'améliorer et s'adapter aux conditions de leur

environnement. Ce type de système verra alors l'émergence d'un écosystème de nature complètement nouvelle, formé d'une population hybride d'agents humains et informatiques, qui seront amenés à cohabiter pour, tout simplement, accomplir leurs destins et vivre. »

BIBLIOGRAPHIE

[Allen, 1983] Maintaining Knowledge about Temporal Intervals, communications of the ACM, no.26, pp 832-843.

[Albrecht *et al*, 1998] Albrecht D.W, Zukerman I., Nicholson A. : Bayesian models for Keyhole Plan Recognition in an Adventure Game, User modelling and User-Adapted Interaction, Vol. 8., (1998), 5-47.

[Arens *et al*, 1993] Arens Y., Chee C.Y., Hsu C.N. et Knoblock C.A., Retrieving and integrating data from multiple information sources, in International Journal of Intelligent and Cooperative Information Systems, 2(2), 1993, pages 127-158.

[Ashley et Aleven, 1994] Ashley K.D. et Aleven V., A logical representation for relevance criteria, in Proc. of the 1st European WorkShop on Topics in Case-Based Reasoning, Kaiserslautern, Germany, Lecture Notes in Artificial Intelligence 837, S. Wess, K.D. Althoff et M.M. Richter éditeurs, Springer-Verlag, Berlin, 1994, pages 338-352.

[Azarewicz *et al*, 1986] Azarewicz, J., Fala, G., Fink, R. and Heithecker, C., Plan recognition for airborne tactical decision making. In: Proceedings of the Fifth National Conference on Artificial Intelligence. Philadelphia, Pennsylvania, pp. 805-811, 1986.

[Ankolendar *et al*, 2002] A Ankolendar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, D. Mcdermott, S.A McIlraith, S. Narayanan. M. Paolucci, T.R Payne et K. Sycara, DAML-S : Semantic Markup for Web services. In the First Internationasl Semantique Web Conference, (ISWC), Italy, 2002

[Artale, Franconi, 1998] A. Artale and E. Franconi, A temporal Description Logic for Reasoning about Actions and Plans, Journal of Artificial Intelligence Research, Vol. 9, 1998, 463-506

[Baader, 1999] Baader F., Artificial Intelligence Today: Recent Trend and Developments, chapter Logic Based Knowledge Representation, in *Lecture Notes in Computer Science.* Springer Verlag, M.J. wooldridge et M. veloso éditeurs, 1999, pages 13-41.

[Baader *et al*, 1991] Baader F., Bürckert H.J., Heinsohn J., Hollunder B., Müller J. Nebel B., Nutt W. et Profitlich H.J., Terminological Knowledge Representation : A proposal for Terminological Logic. Rapport technique TM-90-04 de la DFKI (Deutshes Forshungszentrum für Künstliche Intelligenz), Allemagne, 1991.

[Baader *et al*, 2003] F. Baader, D. Calvanese, D. McGuiness, D. Nardi and P. Patel-Schneider, The description Logic Handbook: Theories, Implementation and applications, Cambridge University Press, United Kingdom, 2003

[Baader et Hollunder, 1991] Baader F., Hollunder B., A Terminological Knowledge Representation System with Complete Inference Algorithms, in Proc. of the first

International Workshop on Processing Declarative Knowledge (PDK-91), Lecture Notes in Artificial Intelligence, éditions Springer-Verlag, vol.567, 1991, pages 67-86

[Baader et Nutt, 2002] Franz Baader, Werner Nutt, Basic description logics, In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.

[Barwise et Perry, 1983] J. Barwise et J. Perry. Situations and attitudes. MIT press, Cambridge, MA, 1983.

[Borgida, 1992] A. Borgida, « Towards the systematic development of description logic reasoners : CLASP reconstructed », In Proc. Of the Third International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, 1992, 259-269

[Borgida, 1996] Borgida A., On the relative expressiveness of description logics and predicate logics, Technical Report 9600004-5, Rutgers University, New Brunswick, USA, janvier 1996.

[Brachman et levesque, 1985] Ronald J. Brachman and Hector J.Levesque, editors, Readings in knowledge representation. Morgan Kauffmann, Los altos, 1985.

[Brachman *et al*, 1991] R.J Brachman, D. McGuinness, P.P Schneider, L.A. Resnick and A. Borgida, "Living with CLASSIC : When and How to Use a KL-ONE-like Language", In principles of Semantic Networks: explorations in the Representations of Knowledge, John F. sowa (Ed), Morgan Kauffmann , 1991

[Brachman et Schmolze, 1985] R.J. Brachman and J.G. Schmolze, « An Overview of the KL-ONE Knowledge Representation System », Cognitive science, Vol. 9:2, 171-216, 1985

[Carberry, 1990] S.Carberry, Incorporating default inferences into plan recognition. In Proceedings of the eight national conference on artificial intelligence (AAAI-90), pp. 471-478, 1990.

[Carberry, 2000] Sandra Carberry, Techniques for plan recognition, User modeling and User-adapted interaction, 11:31-48, kluwer academic publishers, 2001

[Chaib-Draa *et al*, 1992] Chaib-Draa B., Moulin B., Mandiau R. et P. Millot, Trends in distributed artificial intelligence, in artificial intelligence review, 6(1), 1992, pages 35-66.

[Charniak *et al*, 1985] Charniak Eugene, Drew Mcdermott, Introduction to Artificial Intelligence, Addison Wesley, Reading, MA.

[Cohen *et al*, 1981] Cohen, P. R. Perrault & J. Allen, beyond Question-Answering, Report No. 4644, BBN Inc., Cambridge, M.A, 1981

[Cohen 1984] Cohen phillip, Refering as requesting, proceedings of COLING-84, pp. 207, Stanford university, 1984.

[Cohen *et al*, 1998] Cohen R., C. Allaby, C. Cumbaa, M. Fitzgerald, K. Ho, B. Hui, C. Latulipe, F. Lu, N. Moussa, D. Pooley, A. Qian and S. Siddiqi, What is initiative? In User Modeling and User Adapated Interaction, 8(3-4):171-214, 1998

[Chu *et al*, 1993] Chu W.W., Merzbacher M.A., et Berkovich L., The design and implementation of CoBase, in Proc. of the ACM/SIGMOD International Conference on the Management of Data, Washinton (DC), USA, ACM SIGMOD Record, 22(2), 1993, pages 517-522.

[Dautenhahn, 2001] Dautenhahn K., socially intelligent Agents : the human in the loop, in IEEE trans on systems, 31(5), 2001, pages 345-348

[Davis *et al*, 1993] Davis R., Shrobe H. et Szolovits P. What is a knowledge Representation ? AI magazine, 14(1):17-33, 1993.

[Davies *et al*, 2003] Davies J., Van Harmelen F. et Fensel Dieter, Towards the semantic web : ontology – driven knowledge managements, editions Chichester, England, Hoboken, 2003, pages 1-288.

[Dehais *et al*, 2004] Frédéric Dehais, Charles Lesire, Catherine Tessier, Laurent Chaudron, Conflits et contre-mesures dans l'activité de pilotage RFIA 2004, Toulouse, France, janvier 2004.

[Devambu, Litman, 1996] P.T Devambu and D.J. Litman, « Taxonomic Plan Reasoning », Artificial Intelligence, Vol. 84, 1996, 1-35

[Falcone et Castelfranchi, 2001] Falcone R., et Castelfranchi C, The human in the loop of Delegated Agent : The Theory of adjustable Autonomy, in IEEE Trans, 33(5), 2001, pages 406-418.

[Farreny et Ghallab, 1987] Farreny H. et Ghallab Malik, Éléments d'intelligence artificielle, Traité des Nouvelles Technologies série Intelligence Artificielle, éditions Hermès, Paris, 1987, pages 31-69.

[Ferber, 1995] Ferber J., Les systèmes multi-agents : vers une intelligence collective, éditions InterEditions, Paris, 1995.

[Fikes, Nilsson, 1971] R.E. Fikes, N.J. Nilsson, « STRIPS : a new approach to the application of theorem proving to problem solving", Artificial Intelligence Journal, Vol.2., 1971, 189-208

[Finin, 1986] Finin T.W., Interactive Classification: A technique for acquiring and maintaining Knowledge Bases, in Proc. IEEE, vol. 74, 1986, pages 1414-1421.

[Gil, 2005] Y. Gill, "Description Logics and Planning", to appear in AI magazine, 2005, 1-22

[Ghallab *et al*, 1998] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso and D. Wilkins: "PDDL-the planning domain definition language", In: AIPS-98, Planning Committee, 1998.

[Goldmann et Charniak, 1990] R. P. Goldmann and E. Charniak. Dynamic construction of belief networks. In Proc. 6th Conf. Uncertainty in artificial Intelligence, 1990.

[Heinsohn *et al*, 1992] J. Heinsohn, D. Kudenko, B. Nebel and H.J. Profitlich, "RAT: Representation of Action Using Terminological Logics", DFKI Technical Report, 1992

[Hamburger et Richards, 2002] Hamburger H. et Richards D., Logic and Language Models for Computer Science, éditions Prentice-Hall, Upper Saddle River, 2002.

[Haton *et al*, 1991] Haton J-P., Bouzid N., Charpillet F., Haton M., Lâasri B., Lâarsi H., Marquis P., Mondot T. et Napoli A., Le raisonnement en intelligence artificielle, éditions InterEditions, Paris, 1991.

[Hayes, 1980] Hayes P., The Logic of Frames, in Frame Conceptions and Text Understanding, éditions Gruyter, 1980, pages 46-61.

[Hayes 1985] Hayes, P.J, the logic of Frames, in Readings in Knowledge Representation, eds. R.J Brachman & H.J. Levesque, Morgan Kaufman, Los Altos, CA.

[Huber *et al*, 1994] J. Huber, H. Edmund and M. Wellman. The automated mapping of plans for plan recognition. In Proc. 10th Conf. Uncertainty in Artificial Intelligence, pages 344-51, July 1994.

[Kautz, 1987] Henry A. Kautz, "A Formal Theory of Plan Recognition" Thesis - TR 215, 1987.

[Kemke, 2003] C. Kemke, « A formal Approach to describing Action concept in Taxonomical Knowledge bases », In : N. Zhong, Z.W. Ras, S. Tsumoto, E. Suzuku (eds.), Foundations of Intelligent Systems, Springer-Verlag, 2003, 657-662

[Kripke 1991] Kripke saul, Semantical Considerations on Modal Logic, in Reference and Modality, Linsky (ed), p 63-72, Oxford University Press.

[Levesque et Brachman, 1987] Levesque H. et Brachman R.J., Expressiveness an Tractability, in Knowledge Representation and Reasoning, Computational Intelligence, vol. 3, 1987, pages 78-93.

[Lewis, 1986] D.K. Lewis On the plurality of worlds. Blackwell, Oxford, 1986.

[Lipkis, 1982] Lipkis T., A KL-ONE Classifier, in Proc. of KL-ONE Workshop, Jackson, Mississippi, 1982, pages 126-143.

[LOOM, 1991] LOOM users guide (version 1.4), Information Science Institute, University of Southern California, Marina de Rey (CA), USA, 1991.

[Mariano, 1997] Georges Mariano, évaluation de logiciels critiques développés par la méthode B : une approche quantitative, thèse de doctorat, université de valenciennes et du Hainaut-Cambresis,1997

[McDermott, 2002] D. McDermott, «Estimated-Regression Planning for interactions with Web Services », In Proc. Of the Sixth International Conference in AI Planning, 2002

[McGregor, 1988] McGregor R., A deductive Pattern Matcher, in Proc. 7th AAAI, Saint-Paul, Minnesota, 1988, pages 403-408.

[McGregor, 1991] McGregor R., The evolving technology of classification-based knowledge representation systems, in Principles of Semantic Networks : Explorations in the Representation of Knowledge, Morgan Kaufmann Publishers, San Mateo, USA, 1991, pages 385-400.

[McGregor et Burstein, 1991] MacGregor R. et Burstein M.H., Using a description classifier to enhance knowledge representation, in IEEE Expert, 6(3), 1991, pages 41-46.

[McGuinness *et al*, 2002] McGuinness D., Smith M., Volz R., et Welty C., Web Ontology Language (OWL) Guide version 1.0, Document préliminaire du consortium W3C, http://www.w3.org/tr/2002/WD-owl-guide-20021104/, 4 novembre 2002.

[Minsky, 1974] Minsky M., A framework for representing Knowledge, Mémo technique 306 du laboratoire artificielle du MIT, juin 1974.

[Minsky, 1981] Marvin Minsky, A framework for representing knowledge. In J. Haugeland, editor, Mind Design. The MIT Press, 1981.

[Napoli, 1997] Napoli A., Une introduction aux logiques de descriptions, in Projet SYSCO, Rapport de recherche no.3314, 1997, pages 7-50.

[Nardi, Brachman, 2002] D. Nardi, R. J. Brachman. An Introduction to Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.

[Neal, 1998] Neal Lesh. Scalable and Adaptive Goal Recognition. PhD thesis, University of Washington, 1998

[Nebel, 1995] Nebel B., Reasoning and Revision in Hybrid Representation Systems, in Lecture Notes in Artificial Intelligence, éditions Springer-Verlag, vol. 422, 1995.

[Neufeld, 1989] E. Neufeld. Defaults and Probabilities; extension and coherence. In Proceedings of the first Int. Conference on Knowledge Representation and reasoning, pages 312-323, May 1989.

[Pasquier et Chaib-draa, 2004] Philippe Pasquier, Brahim Chaib-draa, Modèles des dialogues entre agents, un état de l'art, Cahiers Romans de Sciences Cognitives, Cuadernos Romances de Ciencias Cognitivas, Cadernos Românicos em Ciências Cognitivas, Quaderni Romanzi di Scienze Cognitive In Cognito (2004), Vol. 1 (n°4), pp-pp

[Pollack 1986a] Pollack Martha, A model of plan inference that distinguishes between beliefs of actors and observers, proceedings of the ACL-86, New York, 1986.

[Pollack 1986b] Pollack Martha, Inferring domain plans in question-answering. Ph. D. thesis, University of Pensylvania, Philadelphia, Pensylvania.

[PowerLoom, 1997] PowerLoom Manual (version 1.0), Information Science Institute, University of Southern California, Marina de Rey (CA), USA, 1991.

[Pylyshyn, 1989] Pylyshyn Z., Computing in Cognitive Science, in Foundations of Cognitive Science, Postner M.I. éditeurs, Cambridge, MIT press, 1989, pages 51-91.

[Pynadath *et al*, 1995] D. V. Pynadath and M. P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence, pp. 472-481, 1995.

[Quillian, 1967] M. Ross Quillian. Word concepts : a theory and simulation of some basic capabilities. Behavorial Science, 12:410-430, 1967.

[Resnick *et al*, 1995] Resnick L.A., Borgida A., Brachman R.J., McGuinness D.L, Patel-Schneider P. et Zalondek K.C., CLASSIC. Description and Reference Manual for Common

Lisp Implementation (Version 2.3), AT&T Bell Laboratories, Murray Hill (NJ), USA, 1995.

[Schmidt *et al*, 1978] Schmidt C. F., N.S. Sridharan and J.L. Goodson. The plan recognition problem: an intersection of psychology and artificial intelligence, Artificial intelligence, PP. 206-214, 1975.

[Schmidt, 1991] Schmidt R., Algebraic Terminological Representation, Rapport technique MPI-I-91-216 de MPI (Max-Planck-Institut für informatik), Allemagne, 1991.

[Schmidt, 1992] Schmidt R., Terminological Representation, Natural Language and Relation Algebra. Rapport technique MPI-I-92-246 de MPI (Max-Planck-Institut für informatik), Allemagne, 1992.

[Schmidt, 2000] Schmidt R., Relational Grammars for Knowledge Representation, in Variable-Free Semantics, collection Artikulation und Sprache, éditions Secolo Verlag, vol. 3, 2000, pages 162-180.

[Schmolze et Lipkis, 1983] Schmolze J.G. et Lipkis T.A., Classification in the KL-ONE Knowledge Representation System, in Proc. of the 8th IJCAI, Karlsruhe, 1983, pages 330-332.

[Sipser 1997] M. Sipser, Introduction to the theory of computation, PWS, 1997.

[Tessier *et al*, 2001] Tessier C. Chaudron L., Müller H-J., Conflicting agents : Conflict Management In multi-agent systems, Kluwer Academic Publishers, 2001, pages 1-30.

[TR Gruber, 1993] T.R. Gruber A translation approach to portable ontology specifications, Knowledge Acquisition, 199-200, 1993

[Valancia, 2000] Valentia E., Outils de topologie algébrique pour la gestion de l'hétérogénéité sémantique entre agents dialogiques, Thèse de Doctorat, Université paris-XI, 2000, pages 37-72.

[Vanbeek *et al*,1991] Peter van beek, Robin Cohen, Resolving plan ambiguity for cooperative response generation. Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, 938-944, August, 1991.

[Villasenor-Pineda, 1999] Villasenor-Pineda, Luis. Contribution à l'apprentissage dans le dialogue homme-machine. Thesis / PhD (08 February 1999), CLIPS / IMAG, UNIVERSITE JOSEPH-FOURIER - GRENOBLE I.

[Weida, 1993] Weida R., Terminological Constraint Network Reasoning and its Application to plan Recognition, Thèse de doctorat, University of Columbia, 1993, pages 9-34

[Weida, Litman, 1992] R.A. Weida and D.J. Litman, « terminological Reasoning with Constraints Networks and an Application to Plan Recognition », In Proc. Of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, MA, 1992

[Wobcke, 2002] Two logical theories of plan recognition, J. logic. Vol 12 no 3, pp. 371-412, 2002.

[Wooldridge, 2000] Wooldridge M., Reasoning about rational agents, MIT Press, Cambridge, Massachussets London, England, 2000.

[Wooldridge et Jennings, 1995] Wooldridge M. et Jennings N.R., Intelligent agents, theory and practice, in Knowledge Engineering Review, 10(2), 1995

[Yen *et al*, 1991a] Yen J., Juang H.-L., et MacGregor R., Using polymorphism to improve expert system maintainability, in IEEE expert, 6(2), 1991, pages 48-55.

[Yen *et al*, 1991b] Yen J., Neches R., et MacGregor R., CLASP : integrating term subsumption systems and production systems, in IEEE Transactions on Knowledge and Data Engineering, 3(1), 1991, pages 25-32.