UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences
Département de mathématiques

MEMOIRE DE MAGISTER

Option : Recherche opérationnelle
Intitulé : Modélisation mathématiques pour l'aide à la décision

SYSTÈMES EXPERTS ET THÉORIE DES GRAPHES APPLIQUÉE

Par

BENATALLAH Mohamed

Devant le jury composé de :

M. BLIDIA	Professeur, Université de Blida	Président
M. ABBAS	Professeur, USTHB, Alger	Examinateur
S. OUKID	Maitre de Conférences A, U. de Blida	Examinatrice
M. MAHIEDDINE	Maître de Conférences A, U. de Blida	. Promoteur
F. HANNANE	Professeur, Université de Blida	Co-promoteur
N. BENBLIDIA	Maitre de Conférences A, U. de Blida	Invitée

Blida, Janvier 2012

RÉSUME

Dans ce mémoire, nous nous intéressons principalement à la modélisation de règle de production d'une base de règles d'un système expert avec le raisonnement de chaînage avant pour représenter et exécuté.

Dans un premier temps, nous donnons une nouvelle représentation graphique des règles de production par un graphe à deux types de sommets. Tout d'abord, nous décrivons une procédure de transformation automatique entre un système expert à base de règles vers un nouveau modèle graphe que nous avons au préalable défini. Cette procédure de transformation permettra l'injection des résultats du deuxième modèle en entrée de l'autre.

Dans un second temps, nous représentent une base de règles d'un système expert par un graphe orienté qu'on appellera " Graphe de Relation ", la mise en ordre (décomposition en niveaux) de ce dernier, nous donnera l'ordre d'exécution des règles. Nous avons utilisé pour cela quelques notions de la théorie des graphes : la détection, l'énumération des circuits dans les graphes et le problème de feedback vertex set.

Afin d'effectuer une validation expérimentale de nos travaux, une implémentation dans les langages de programmation Java est décrite dans chacune des deux représentations précédentes.

En fin ; un test a été effectué pour comparer le temps d'exécution des règles, en utilisant un programme existant avec nos travaux.

Mots-clés: moteur d'inférence chaînage avant, règle de production, graphe proposé, graphe de relations, circuits, feedback vertex set.

ABSTRACT

In this thesis, we are interested principally in modeling of production rules of based rules in expert system with reasoning of forward chaining to represent and execution.

Initially, we give a new chart of production rules by a graph to two vertexes. First of all, we describe an automatic procedure of transformation between an expert system containing rules towards a new model graph which we have with the definite precondition. This procedure of transformation will allow the injection of the results of the second model in entry of the other.

In the second time, represent rules base of an expert system by a directed graph which one will call "Relations Graph", the ordering of this last (decomposition in levels), will give us the order of execution of rules. We used for that some notions of graph theory: detection, enumeration of circuits in directed graphs and problem of feedback vertex set.

In order to carry out an experimental validation of our work, an implementation in the programming languages Java is described in each of the two preceding representations.

End, a test was carried out to compare the execution time of the rules, by using an existing program with our work.

Key words: inference engine forward chaining, production rule, graph proposed, relations graph, circuits, feedback vertex set.

? ?????

????????? ?? ??? ??? ? ?T?? ??? ????? ???????? ? ?? ? ? ?? ? ??????T?? µ ?? ? ????? ? ?
????t ?? ?? ? ??? ? ???? ?? ??? d?d T??? ?? ?????? ? ? ?????? ? ???T ??

? ? ?????? ? ? ???T ??G ??? ????? ?? ??? ?? ??? ?T?? ? ?????? ? T????? ???????? ? ????? ? ???????? ?
? ????? ? ? ??? ?T?? ??? ???? ?? G ??? ????? ?? ?? ??? T????? ??? ??t ? ??? T??? ? ?T??T?? ? ?????
????T??? ???? ????????? ?????? ??T?? ???Td ?? ??? T??? ??t ? ???? ??? ???????? ?? ??

???T?????? ??? ? ? T?????? ?? ??? d?d T??? ?? ?????? ? ? ?????? ?? ??? ? ??? .????????Gt ? ??? ?
?? ? ??? ? ? ??????? ????? ???????? ? ??? ???T ??G ??? ????? ?? ??? ??? ?? ???????????? ???? ????T??
?????T? ? ???? ? ?? ??? ? ? ???? ????? ??????? ? ????d?? ? ??? ? ??? ????? ? ???????? ??G?? ? ????
? ?? T?? ??????? ???????? ??? ??? ?? ? ?? ? ? ???????T ?? ?????? ?? ??????? ?? ? ?? ??t ???? ?? ? ?
?? ?µ ??????? ??? ? T??????G?? ?? ? ? ??????? ? ? ??????? ???????? ? ??? ? ? ? ? ? ?????G????
????G ???? ????? ?

? ?S?T????? ?? ??????? ? ? ?????? ?? ?? ?????? ?? ?????T???? ???? ? ? ? ??? ? T??? ?? T???? ? ?

??????? ?????? ???T ??? ?? ?????? ?T??? T?? ? ????? ??? ?? ???? ????? ?

REMERCIEMENTS

Tout d'abord, je voudrais exprimer à Monsieur **Mohamed Mahieddine**, Maître de conférences à l'université Saad Dahlab de Blida mes vifs remerciements et ma profonde reconnaissance d'avoir accepté d'être rapporteur ainsi que pour sa disponibilité, ses orientations et ses conseils constants qu'il a pu émettre à la fois sur le fond et sur la forme de ce travail, et Monsieur **Farouk Hannane**, Professeur à l'université Saad Dahlab de Blida, , pour leur suivi attentif, leur réflexion pertinents et leur rigueur. Je suis très sensible à la confiance qu'ils m'ont témoignée à plusieurs reprises et à leur patience exemplaire qu'ils ont prouvée durant la période de la réalisation de ce travail.

Je tiens à remercier vivement Monsieur **Mostafa Blidia**, Professeur à l'université Saad Dahlab de Blida, pour l'honneur qu'il me fait en acceptant de présider le jury de ce mémoire.

Je tiens également à remercier: Monsieur **Moncef Abbas**, Professeur à l'USTHB d'Alger, Madame **Oukid S**., Maître de conférences à l'université Saad Dahlab de Blida et Mademoiselle **Benblidia N**., Maître de conférences à l'université Saad Dahlab de Blida, dont les précieuses remarques m'ont aidé à accomplir ce travail et dont la participation à ce jury m'honore.

Je saisis aussi cette opportunité pour exprimer mes remerciements et ma gratitude à tous ceux qui m'ont aidé de prés ou de loin et dont le soutien m'a été précieux durant la préparation de ce mémoire, Je ne peux pas oublier de mentionner la grande compétence et la justesse d'appréciation de messieurs **M.Chellali, A.Derballa** (Directeur de Labo) et **O.Tami** (Directeur de département Maths), **Mr.H.Bakouche(ing info)**,Melle **N.Djillali** (Doctorant) et **R.Mazari** (Magister).

Enfin et profitant de cette occasion, je voudrais exprimer mes remerciements et ma sympathie à ma famille pour le soutien qu'elle m'a pu apporter ainsi que pour ses encouragements permanents.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

Dans son travail sur la conception de systèmes multi-agents intelligents et mobile pour téléphones portables, Mazari R. [1] a modélisé le coté intelligence des agents par un framework qu'il a instancié par un système expert à base de règles de production, utilisant le chaînage avant. Dans son implémentation des règles de production Mazari R, a utilisé une structure de liste chaînée, et les bases de règles et faits comme des listes ou piles. Malgré la conception optimisée qu'il a adoptée, néanmoins un problème majeur en découlait, et qui est celui du temps important qui s'écoule dans l'inférence.

Le but de cette recherche est de modéliser les règles au moyen de graphes pour voir si on ne pourrait pas améliorer le coût en temps du fait de l'utilisation d'une algorithmique basée sur les opérateurs des graphes.

Les systèmes à règles de production connaissent un réel succès en tant que moyen de représentation de connaissances dans les systèmes experts [2]. Dans ce mémoire, nous nous intéresserons à la représentation des connaissances qui nous permettrait d'améliorer leur traitement par un moteur d'inférence. Il existe plusieurs modélisations qui ont fait l'objectif d'études afin d'améliorer à performance du dit moteur [4]. La base de notre réside dans le fait que, nous avons pensé à modéliser ce type de règles par la création d'un nouveau graphe en se référant à d'autre types de graphes préalablement utilisés dans le domaine de l'informatique en général.

Les recherches sur les systèmes experts portaient au début sur les moteurs d'inférence. De nombreux types de moteurs très performants ont été développés. Certains types de moteurs d'inférence sont très connus, comme les méthodes de chaînage avant et de chaînage arrière.

Dans les méthodes de chaînage avant, c'est à partir des faits que l'on désigne par conditions, qu'on arrive aux conclusions. Pour déduire un fait particulier, on déclenche les règles dont les conditions sont connues jusqu'à ce que le fait à déduire soit également connu ou qu'aucune règle ne puisse être déclenchée. Cependant, cet algorithme ne peut pas choisir une règle applicable sans effectuer un chaînage.

Dans ce travail, nous étudions et développons les mécanismes de recherche qui permettent de choisir la règle applicable qui doit être utilisée la première afin d'améliorer le temps d'inférence. Ce qui nous a conduits à réfléchir à la manière dont en doit modéliser les règles. En sachant que les graphes orientés ont connu un grand succès dans la représentation des connaissances dans différents domaines, nous avons de notre part essayé de représenter les règles par un formalisme de graphes, et d'utiliser ainsi la théorie des graphes pour essayer d'améliorer les performances résultants de cette représentation.

En général, notre travail consiste à modifier la modélisation d'un système expert existant réalisé à base de règles implémenté au moyen de listes chaînées qui présentait des performances pas très acceptables, par une conception au moyen de graphes, mais comme le problème réside beaucoup plus dans le choix de la meilleure applicable pour l'inférence. Nous avons dans un deuxième temps, utilisé une modélisation par les graphes orientés et proposé une méthode pour trier ou ordonner les règles de la base de règles pour la meilleur inférence au moyen de techniques des graphes.

1. Etat de l'art

Dans cette partie nous allons donner un aperçu sur les principaux travaux menés pour la modélisation de base de règles d'un système expert à base de graphe.

En 1965, une équipe de l'université de Stanford travaille sur la résolution d'un problème pour lequel on ne connaît pas de solution algorithmique : il s'agit de trouver la correspondance entre le spectre de masse et la formule développée d'un corps chimique. Le premier système expert est né [3].

Feigenbaum [4] a défini un système expert comme étant " un logiciel intelligent qui utilise des connaissances et un procédé d'inférence pour résoudre des problèmes. Ces problèmes sont assez difficiles à résoudre et requièrent une expertise humaine significative pour arriver à une solution. Les connaissances de système expert sont composées de faits et de heuristiques ".

Donc, les systèmes experts sont faits pour résoudre des problèmes dans un domaine spécifique d'expertise et ne sont pas des outils généraux de résolution de problèmes. Par exemple, un système expert ne va probablement jamais résoudre des problèmes d'un domaine aussi large que le domaine médical, mais peut toutefois s'adresser parfaitement à des sous-domaines spécifiques du domaine médical. Cela a été démontré par des systèmes comme MYCIN et DENDRAL [2], des systèmes experts très connus utilisés pour le diagnostic de maladies infectieuses sanguines et pour l'identification de structures moléculaires respectivement. Actuellement, les systèmes experts couvrent un large domaine d'application [8], dans lequel on peut distinguer deux principales applications : les systèmes d'aide à la décision et les systèmes d'aide au diagnostic. Ces systèmes doivent être conçus pour *aider* l'homme dans la réalisation de tâches difficiles et ne doivent jamais être utilisés pour le *remplacer* complètement.

Il existe plusieurs modèles de connaissances pour l'étude des applications. En particulier, Luger dans [12] en décrit quatre: les systèmes à base de règles, les systèmes de raisonnement à partir de cas, système expert basé sur un modèle et système expert hybride. Nous nous focalisé sur la réalisation d'un modèle de système expert basés sur des règles.

Les représentations de la connaissance sont peut-être les plus populaires en recherche. Pour des bases de connaissance avec des centaines ou des milliers de règles, le nombre de chemins possibles d'inférence est très grand.

Les travaux sur les systèmes experts s'intéressent aussi à la représentation des connaissances. Plusieurs formalismes ont été proposés pour la représentation des connaissances en machine. Les toutes premières bases de connaissances étaient organisées en règles de production. D'autres formalismes ont été aussi proposés et utilisés : les réseaux sémantiques, les graphes conceptuels [18, 19], les frames [15], les représentations par objet [9], etc.

Les réseaux sémantiques se trouvent bien adaptés à certains domaines tels que la compréhension des langages naturels et la reconnaissance des formes. Leurs avantages résident dans la structure graphique de regroupement des éléments d'une connaissance, leur inconvénient majeur est la complexité de représentation et de gestion qui croit d'une manière significative avec la taille de la connaissance [21]. Un exemple utilisent la structure d'un réseau sémantique est PROSPECTOR (1978) [18].

Cependant, il existe un modèle dérivé des réseaux sémantiques et dont la sémantique est moins floue et plus formelle. Ce sont les graphes conceptuels.

Le graphe conceptuel est un diagramme qui représente non pas la syntaxe mais la sémantique d'une phrase. Les graphes conceptuels ont été utilisés dans des outils de spécification et de modélisation de logiciels [21]. De plus, ils ont été aussi utilisés comme langage de représentation de connaissance pour la génération de langages naturels et l'extraction d'information [25].

Les réseaux bayésiens sont eux un outil de choix dans la représentation de connaissances et dans l'exploitation de celles-ci [31]. Les réseaux bayésiens dans ont été utilisés dans beaucoup d'applications, sans même le savoir par Microsoft par exemple, Google et Mozilla [32].

Tous les graphes cités précédemment, n'ont pas eu d'application, à notre connaissance, dans la modélisation de règles de productions qui contiennent plusieurs conditions et conclusions, donc nous n'avons pas rencontré, publiée dans la littérature, de structure de graphe pour la représentation de règles de production.

Dans les bases de règles s'écrire sous la forme de règle de productions, pose une importante question pour la vérification des systèmes à base de règles [35].

La théorie de graphes est un outil très intéressant pour représenter l'ordre conceptuel des dépendances. D'ailleurs la théorie de graphes fournit des techniques analytiques pour les propriétés comme la connexité et l'accessibilité de manière formelle, de plus l'analyse des formulations montre que les problèmes dans la base de règles mènent à l'existante de certaines propriété dans les graphes orientés [38].

Les méthodes à base de graphes d'induction [24] occupent une place intermédiaire, car la fonction de classement est exprimée par un graphe d'induction qui peut être re-écrit sous forme de règles de production. Cependant, le re-écriture d'un graphe comme une simple collection de règles, engendre un certain nombre de problèmes [25].

Parmi les outils pour modéliser un système expert, l'arbre a été activement utilisé [26]. L'arbre *et/ou* a été utilisé comme méthode de représentation de base pour comprendre le processus d'inférence d'un système expert qui représente une inférence en chaînage arrière [30].

Le réseau de Pétri a lui été utilisé pour simuler l'exécution de la base de connaissances [32]. Ces méthodes doivent vérifier le modèle sous tous les états initiaux possibles afin de garantir l'absence d'incohérences. Malheureusement, ce test peut être parfois de calcul très coûteux.

L'algorithme de Rete fournit la base pour une exécution plus efficace d'un système expert [37]. Un système expert basé sur Rete établit un réseau des nœuds, pour minimiser le temps de réévaluation des conditions des règles lors de changements de la mémoire de travail. Dans les systèmes experts très grands, cependant, l'algorithme de Rete original tend à rencontrer des problèmes de consommation de mémoire.

2. Buts visés par le projet

Les débuts de cette recherche consistent en l'étude et l'analyse du système de raisonnement à base de règles choisi actuellement, de comprendre comment il a été implémenté, de mettre le doigt sur son coté faible, et de chercher ensuite comment concevoir puis utiliser un modèle à base de graphes pour remédier à ses faiblesses.

La plupart des techniques de modélisation utilisées pour la tâche de vérification des règles avec l'inférence ont utilisé des structures d'arbre [40]. Cependant concernant mon travail, il fallait opter pour la structure de graphe dans le but d'arriver à une représentation aboutissant à un ordonnancement des règles de production permettant d'améliorer la performance de l'inférence par un moteur d'inférence chaînage avant.

3. Contexte du projet

Ce projet est entrepris au sein de l'équipe GLODOO, de Génie Logiciel et de Développement Orienté Objets faisant partie du LDRSI (Laboratoire de Développement et de Recherche en Systèmes Informatisés), situé à l'université de Blida.

On pourrait dire que c'est un projet de recherche fondamentale et appliquée.

Les buts supérieurs de GLODOO sont :

- aider à comprendre et de continuer à évaluer et à améliorer des procédés de travail en technologies récentes de la conception et de la programmation.
- fournir un environnement flexible de travail commun pour concevoir des applications sur les dispositifs mobiles.
- disséminer les résultats aux collègues, aux étudiants, aux compagnies, et à la communauté s'intéressant au domaine dans son ensemble.

4. Contributions

La contribution de ce projet s'articule principalement autour de :

- L'étude,
- la conception,
- l'implémentation et
- l'application

Des aspects avancés de la technologie des systèmes experts, avec une visée particulière de résolution des problèmes de gain de temps d'exécution.

Cette thèse vise à améliorer de manière significative des possibilités de développement des systèmes experts. La conception par les graphes est explorée, et leur utilisation dans la conception est pratiquée comme facteur significatif découlant d'un choix judicieux d'opérateurs de graphes permettant d'aboutir à une optimisation en temps de l'inférence qui est à la base du raisonnement des agents en intelligence artificielle.

L'application choisie a pour but principal de prouver la réalisabilité d'une application concrète de notre technique.

5. Structure du mémoire

Ce mémoire comporte quatre chapitres qui sont développés comme suit :

Dans le chapitre 1, on y introduit le domaine de systèmes experts. Nous y présentons aussi les principes et les concepts. Nous présentons la deuxième section une définition des différents types de graphes connus dans le domaine des systèmes experts.

Le chapitre 2 introduit un état de l'art sur les systèmes experts et le problème de la modélisation de règles de productions par les différents types de graphes. La deuxième partie du chapitre présente les définitions de base de la théorie des graphes et décrit les problèmes de la détection et de l'énumération des circuits eulériens [45] dans les graphes orientés ainsi que le problème de *feedback vertex set* [49].

Nous présentons ensuite dans le chapitre 3 la méthode de modélisation que nous avons proposée et implantée. Nous proposons une nouvelle structure de graphe qui comporte deux types de sommets ainsi que les critères de ce choix. Dans une deuxième partie, nous donnerons une autre modélisation par des graphes orientés permettant de trier et d'ordonner les règles de base pour optimiser l'ordonnancement des règles dans le moteur d'inférence d'un système expert, nous ferons ainsi la mise en ordre d'un graphe orienté sans circuit. Toutefois, le problème qui a fortement contraint notre travail réside dans l'existence de ces circuits eulériens que nous allons par conséquent chercher à éliminer, ce problème est appelé *feedback vertex set*.

Dans le chapitre 4, nous donnons dans la partie de réalisation, tous les programmes utilisés et un exemple de règles évoqué des exemples de la partie trois et nous effectuons une copie d'écrans de l'application à travers le système expert développé pour les agents sur téléphones portables [1].

Enfin, nous terminons par une conclusion avec une présentation de quelques unes des perspectives d'avenir envisageables.

CHAPITRE 1
CONCEPTS FONDAMENTAUX

Les recherches en intelligence artificielle découlent du besoin de développer des programmes capables de résoudre des problèmes d'une manière considérée comme intelligente, c'est-à-dire de manière semblable à celle utilisée par les hommes [5]. Les systèmes experts ont comme finalité de reproduire le comportement de l'être humain dans ses activités de raisonnement. Le but de notre thèse est de gagner en efficacité dans le traitement des tâches complexes de décision [7]. Pour rendre un programme intelligent, il est nécessaire de lui fournir une quantité importante d'informations spécifiques au domaine du problème à résoudre.

Nous présenterons tout d'abord quelques définitions relatives aux systèmes experts, aux composants logiciels, aux agents intelligents et modes de représentation des connaissances ainsi que les techniques de raisonnement. Puis nous introduirons différents types de graphes connus dans le domaine de représentation des connaissances.

1.1 Les systèmes experts

Il existe plusieurs définitions des systèmes experts dans la littérature [3,7]. Certains mettent l'accent sur les tâches que ces systèmes peuvent effectuer, d'autres accordent plus d'importance à la façon dont les tâches sont mises en œuvre. Un système expert est défini comme un programme informatique qui peut conseiller, analyser, classer, communiquer, consulter, concevoir, explorer, anticiper, former, identifier, interpréter, justifier, gérer, planifier, apprendre et tester [8]. Ces tâches constituent des problèmes complexes qui se réfèrent généralement à l'expertise humaine pour l'élaboration d'une solution.

Feigenbaum met en évidence les mécanismes de raisonnement dans sa définition: un système expert est un programme intelligent qui utilise des connaissances et des procédures d'inférences pour résoudre des problèmes [4]. L'inférence est une opération intellectuelle, par laquelle on passe d'une vérité à une autre vérité, jugée telle en raison de son lien avec la première. Ces définitions présentent un système expert comme un programme informatique capable de réagir à des questions d'un utilisateur selon certaines règles. Cependant, ces définitions restent toujours informelles, n'obéissant pas à des règles déterminées puisque généralement ces systèmes dépendent du modèle choisi pour la représentation des connaissances. Ces modèles de représentation des connaissances forment un élément clé pour identifier le mode de raisonnement pertinent.

1.1.2 Composants d'un système expert

L'architecture générale d'un système expert [8] est illustrée par la figure 1.1. Nous pouvons classifier ces éléments en deux parties :

- Les acteurs : l'expert et les utilisateurs
- Les composants logiciels : la base de connaissances et le moteur d'inférence

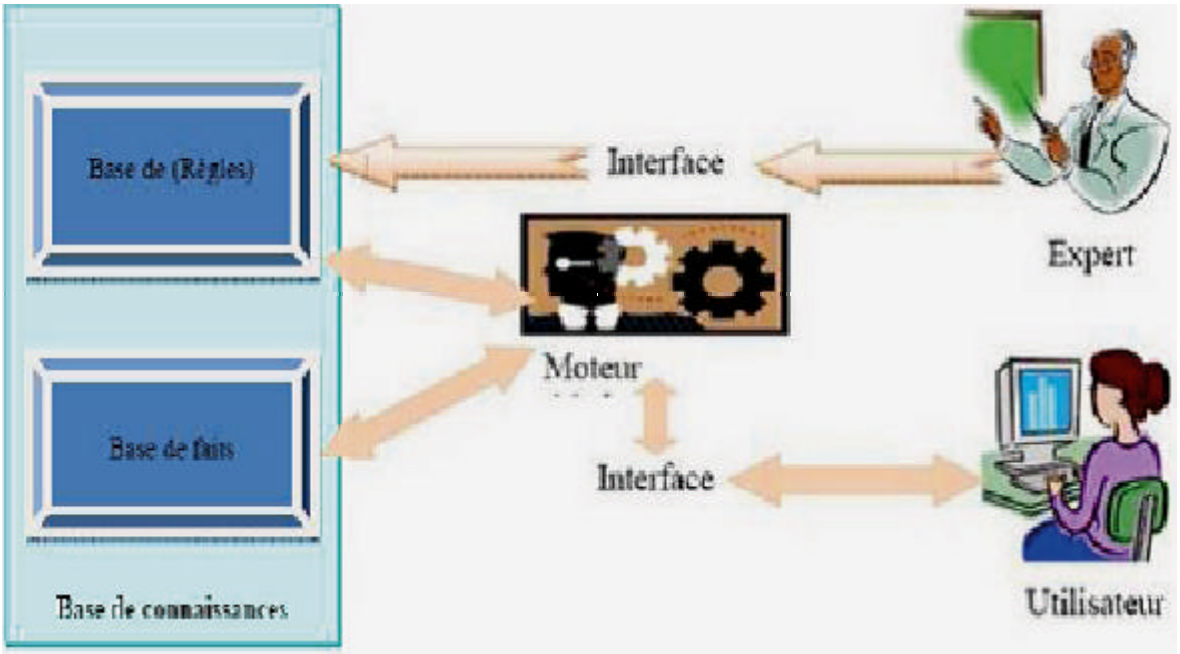


Figure 1.1 Architecture principale des systèmes experts.

Les composants logiciels sont compilés en deux parties indépendantes :

a) La base des connaissances

Elle se décompose en deux parties. *La base de faits* qui contient les faits spécifiques au domaine d'application et *la base de règles* contenant l'ensemble des règles qui vont permettre au système de raisonner à partir de la base des faits. Il existe différents modes possibles de représentation des données.

? La base des faits

Elle constitue la mémoire de travail du système. Elle contient les données initiales et les données recueillies par les hypothèses émises ainsi que les nouveaux faits prouvés. La base des faits a pour particularité de s'enrichir de manière dynamique au cours de la résolution. Les faits peuvent prendre des formes plus ou moins complexes. Les faits élémentaires peuvent avoir des valeurs booléennes, symboliques, ou réelles. Un système qui n'utilise que des faits booléens est dit *d'ordre 0*. Un système qui utilise des faits symboliques ou réels, sans utiliser de variables, est *d'ordre 0+*. Un système utilisant toute la puissance de la logique du premier ordre est *d'ordre 1*.

? La base des règles

La base de règles contient l'ensemble des règles de raisonnement du système. Elle peut être modélisée par différents modes de représentation des données (voir section 1.6). Cette base rassemble la connaissance et le savoir-faire de l'expert. Elle n'évolue donc pas au cours d'une session de travail et constitue la partie statique du système.

b) Le moteur d'inférence

Le moteur d'inférence est un programme qui, dans le système expert interprète les données de base de connaissances et assure, suivant des stratégies générales et particulières, l'enchaînement des étapes de la résolution d'un problème donné. Il met en œuvre le mécanisme de raisonnement chargé d'exploiter les connaissances. Il effectue les déductions et donc peut générer de nouveaux faits. D'une façon générale, le moteur d'inférence est capable de répondre à des questions, de

raisonner et de déduire des conséquences. C'est donc lui qui fournit les réponses aux utilisateurs.

1.2 Agents Intelligents

Les agents *intelligents* combinent les trois caractéristiques (autonomie, coopération, adaptation) à leur plus haut niveau. Ils sont donc en principe capables de planifier leurs actions, de négocier avec d'autres agents et d'acquérir ou de modifier leurs connaissances. Ils sont en général dotés de la capacité d'apprentissage [9]. On peut conclure de cette définition qu'un agent (logiciel) est un composant logiciel, autonome, situe, communicant, réactif, proactif, ayant des buts et des compétences. A partir de la définition précédente, on peut schématiser l'agent comme suit :

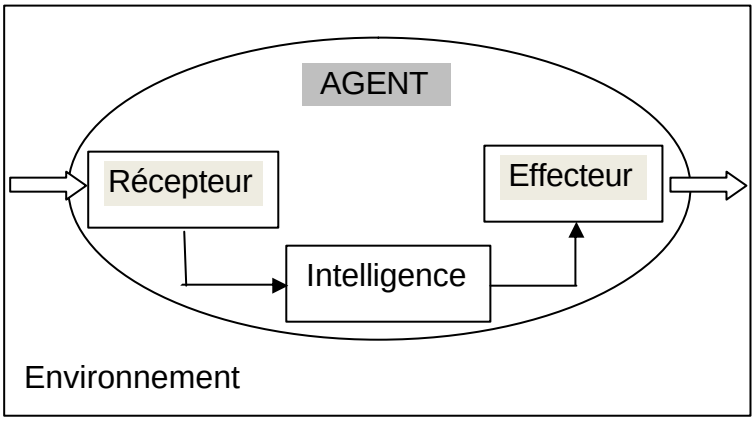


Figure 1.2 La structure simplifiée l'environnement d'un agent intelligent.

Un agent intelligent contient un ou plusieurs des éléments suivants [10] :

- Une base de connaissance prédéfinie,
- Un moteur d'inférence, lui permettant de tenir des raisonnements plus ou moins complexes,
- Un système d'acquisition de connaissances,
- Un mécanisme d'apprentissage.

1.2.1 Caractéristiques de l'agent intelligent

Il doit être essentiellement:

1. Autonome : Il s'exécute sans aucune intervention humaine.
2. Personnalisé : L'agent intelligent est configuré de manière à répondre selon les désirs de l'utilisateur.
3. Auto-adaptatif : L'agent intelligent sait ajuster son comportement en fonction de la situation selon les attentes de l'utilisateur.
4. Auto-apprenant : Il sait exploiter l'expérience passée pour mieux comprendre les souhaits actuels de l'utilisateur.
5. Coopérant : Les agents intelligents communiquent entre eux et interagissent.

1.2.2 Intelligence des Agents

Dans cette section, nous présenterons un certain nombre de concepts du domaine des systèmes de raisonnement et des agents intelligents [11].

Une approche intéressante pour distinguer les agents intelligents peut être basée sur le rapport entre

- les actions externes (comportement) et
- l'adaptation interne.

L'adaptation effectuée à l'intérieur de l'architecture du système évolue avec le temps (l'architecture d'évolution du système) donne:

- comment un agent est manœuvré pour atteindre ou restructurer l'information dans un autre agent,
- comment les agents communiquent les uns avec les autres pour changer l'état de la connaissance pour l'un d'entre eux,
- comment traiter des agents comme systèmes au niveau de la connaissance en leur assignant de la connaissance et des buts,
- comment décider pour chaque agent quelle tâche va-t-il effectuer, et ce qu'il faudrait apprendre sur les rapports entre les tâches (un agent accomplissant certaines tâches pourrait pouvoir accomplir d'autres tâches).

1.2.3 Agents et Raisonnement Basé sur les Règles

Le système (ou moteur) de raisonnement d'un *agent* "To" en général accepte n'importe quel système de raisonnement.

Pour l'instant, notre travail réside sur le raisonnement qu'avec un moteur d'inférence chaînage avant à base de règles.

L'agent "To" raisonne donc dans son domaine, concernant les décisions à prendre au sujet des actions qu'il pourrait entreprendre, en utilisant un système de déduction à base de règles de production.

Un système de déduction logique (ou de raisonnement) *basé sur les règles* est un système expert qui emploie un ensemble de règles en tant que base de modèle de son savoir-faire.

Les systèmes de raisonnement basés sur les règles se combinent bien avec les agents pour deux raisons :

- les règles rendent une définition compacte du comportement possible,
- sous sa forme la plus simple, un comportement est un ensemble d'actions et de conditions sous lesquelles ces actions devraient se produire.

Dans cette section, nous allons surtout explorer certaines des techniques employées pour représenter la connaissance dans un domaine dans les programmes d'intelligence artificielle. Nous allons principalement nous concentrer sur la façon dont les diverses représentations graphiques de la connaissance peuvent être employées pour implémenter le raisonnement pour les agents.

1.3 Organisation des connaissances

La représentation des connaissances est considérée comme la manière de mémoriser les connaissances dans un ordinateur. Il y a plusieurs modes de représentation des données qui peuvent être utilisés séparément ou en conjonction pour construire un système [8]. Chaque technique offre certains avantages de performance, de lisibilité ou de modularité. Les modes de représentation les plus utilisés dans les systèmes experts sont : les règles de production et la structure d'objets, la logique formelle.

1.3.1 Règles de Production

Les règles de production sont des modèles de représentation des connaissances très répandu [8]. Elles sont ainsi dénommées, car les règles produisent de nouveaux faits au cours de la déduction. L'ordre dans lequel les règles sont introduites dans la base de connaissance n'est pas important. Par contre, lorsque plusieurs règles sont déclenchées, l'ordre peut jouer un rôle pour le choix de la première règle à considérer. Ces règles sont représentées sous la forme suivante :

SI condition(s) **ALORS** conclusion(s).

La partie condition, appelée aussi prémisse, correspond à une expression qui doit être vérifiée pour que la règle se déclenche. La partie conclusion peut correspondre au déclenchement d'un nouveau fait.

La plupart des règles représentent des connaissances du domaine mais il existe également des règles qui matérialisent des métaconnaissances.

Les métaconnaissances sont des connaissances sur les connaissances. Elles peuvent porter sur l'intérêt d'une connaissance, sa priorité, sa véracité, sa précision etc. Les métaconnaissances permettent souvent de mettre en place une stratégie de choix lorsque plusieurs règles sont applicables.

Exemple 1.1.

- règle sans variable :

Si la machine 1 est en attente alors elle doit être chargée par l'opérateur 5.

- règle avec variable :

Toutes les machines de type 1 doivent être chargée par les opérateurs de type 5.

Pour s'affranchir de problèmes de cohérence le formalisme des règles retenu est souvent à base de propositions

Exemple de la règle 217 de MYCIN [18] :

SI L'organisme est une bactéroïde
ET Le site de la culture était stérile
ALORS
La thérapie recommandée doit être choisie parmi les suivantes : chloramphénicol, clindamycine, tétracycline, limdomycine, gentamycine.

a. Avantages de la méthodologie des règles

Les systèmes basés sur les règles de raisonnement sont puissants parce que les actions elles-mêmes peuvent déduire de nouveaux faits.

Parmi leurs avantages, on pourrait citer :

- La modularité des règles qui simplifie la tâche de mettre à jour la base de connaissance.
- Différentes règles peuvent être ajoutées, supprimées, ou modifiées sans affecter l'exécution globale du système.
- Documentation: Un système expert peut fournir une documentation permanente du procédé de décision.
- Largeur: La connaissance des experts humains multiples peut être combinée pour donner à un système plus de largeur qu'une personne simple est susceptible de réaliser.

b. Inconvénients de la méthodologie des règles

- Le programmeur devrait pouvoir fournir directement la connaissance au système.
- Pour mettre à jour la base de règle directement: le travail doit être entrepris par le programmeur.
- La difficulté d'acquisition des règles: la formulation de la connaissance par des règles.
- Les enchaînements infinis: les cycles entre les règles.
- Les contradictions par de nouvelles connaissances: les nouvelles connaissances prévues pour fixer un problème peuvent présenter des contradictions non désirées.

- La modification des règles existantes: en plus de l'enchaînement et des contradictions infinis, les règles additionnelles peuvent résulter en des modifications
- L'inefficacité: les algorithmes de recherche basée sur un modèle de règle ne sont souvent pas efficaces, et nécessitent des optimisations (*Algorithme de Rete*).
- La puissance d'expression de la connaissance par des règles ne s'adapte pas toujours à d'autres modèles de la connaissance.

1.3.2 La logique formelle

Les règles de production sont ensuite traitées par les moteurs d'inférence. Ceux-ci permettent d'appliquer des méthodes de raisonnement formel déductif en utilisant les règles disponibles. Le calcul propositionnel est une des méthodes (système formel) qui permet de raisonner sur des formules logiques [6]. Le calcul des prédicats à son tour est un système formel de raisonnement sur des prédicats (formules généralisées). Les données représentées sont déclaratives et se basent essentiellement sur la logique mathématique. Il convient de distinguer principalement : la logique des propositions et la logique des prédicats.

- La logique des propositions: est constituée d'un ensemble de propositions qui peuvent prendre la valeur "vrai" ou "faux". La détermination des propositions repose sur l'utilisation de connecteurs logiques (et, ou, non, implique, équivalent). La logique des propositions n'utilise pas les notions de variables, de fonctions, de prédicats ni les quantificateurs [8].
- La logique des prédicats: les principes de base du langage ne sont plus des propositions, mais des prédicats [8]. Un prédicat peut être vu comme une proposition qui contient une ou plusieurs variables. La logique des propositions est donc incluse dans la logique des prédicats.

Les techniques d'inférence permettent de guider le système expert pour prouver la vérité ou la fausseté d'une proposition, ou bien pour déduire des conclusions à partir des conclusions. C'est l'application des règles d'inférence (ou règles de déduction logique) qui guide le système expert dans ce sens.

- A et A ? B. Ce mode de raisonnement est utilisé lorsque les connaissances du système sont représentées par la logique du premier ordre. La règle d'inférence de Modus Ponens par exemple, les propositions suivantes : (1) aujourd'hui il pleut, (2) s'il pleut alors je prends mon parapluie, on peut déduire que je prends mon parapluie.
- Modus Tollens : c'est une règle presque similaire à la règle Modus Ponens par exemple, (1) je suis sorti, (2) s'il pleut je reste chez moi, on peut en déduire qu'il ne pleut pas.

La logique formelle utilisée de bases théoriques solides et uniformes et correspond souvent à la façon naturelle de s'exprimer. Elle constitue un compromis intéressant entre la puissance et la carté d'expression. Cependant, elle présente également des inconvénients, la rigueur propre à la logique formelle ne permet pas de prendre en compte des appréciations nuancées, car le résultat est toujours booléen (Vrai ou Faux). Cette limitation se manifeste par exemple lorsque le contexte d'interprétation d'une connaissance n'est pas connu avec précision ou certitude. D'autres logiques, comme la logique à plusieurs valeurs ou logique floue, ont été introduites pour contre carrer ces limitations.

1.3.3 Programmation Orientée Objets java :

Pour développer les algorithmes utilisés dans notre approche d'optimisation, nous avons utilisé le langage de programmation orienté objet JAVA. Parmi les avantages de cette programmation orientée: la compréhensibilité de la conception, l'encapsulation de l'information, l'extensibilité et la réutilisation du programme pour un futur développement, et la modularité de la conception. Ce qui rendra le programme facile à examiner, à tester et à retrouver d'éventuelles erreurs.

Les principales techniques utilisées pour réutiliser le comportement sont :

- Héritage
- composition

Dans l'héritage

- il est plus facile de redéfinir les méthodes héritées, et
- le code est statique (plus simple à suivre).

La composition

- comporte moins de programmation, et
- le changement du comportement s'effectue en temps d'exécution.

1.4 Le moteur d'inférence:

L'inférence est pour les ordinateurs ce qu'est le raisonnement pour les humains. La méthode de raisonnement employée dans un moteur d'inférence est le raisonnement qu'avec un moteur d'inférence chaînage avant.

1.4.1 Le chaînage avant :

Il permet de déduire les nouveaux faits à partir des faits initiaux. Le mécanisme consiste à appliquer toutes les règles possibles à l'ensemble des faits connus, en ajoutant chaque nouvelle conclusion à cet ensemble. Par application itérative du processus, chaque conclusion peut elle-même satisfaire les conditions d'une autre règle, ce qui conduit à un chaînage avant des règles [14]. Le processus s'arrête lorsqu'aucune nouvelle règle n'est applicable à l'ensemble des faits, ou quand une solution satisfaisante est trouvée. La plupart des moteurs d'inférence en chaînage avant utilisent l'algorithme de RETE [4] pour optimiser les performances.

Exemple 1.2. Soit une base de règles qui contient les règles suivantes :

- REGLE r1 : **SI** animal vole **ET** animal pond des œufs **ALORS** animal est un oiseau.
- REGLE r2 : **SI** animal a des plumes **ALORS** animal est un oiseau.
- REGLE r3 : **SI** animal est un oiseau **ET** animal a un long cou **ET** animal a de longues pattes **ALORS** animal est une autruche.

La base de faits :

- F1 : animal a des plumes.
- F2 : animal a un long cou.
- F3 : animal a de longues pattes.

Base de faits initiale : F1, F2, F3.

Examen de la règles r1 : les conditions ne sont pas satisfaisantes ;

Examen de r2 : la condition est satisfaisante

F4 : “animale est un oiseau“ est ajouté à la base de faits

Nouvelle base de faits : F1, F2, F3, F4.

Examen de r3 : les conditions sont satisfaisantes

F5 : “animale est une autruche“ est ajouté à la base de faits

Nouvelle base de faits : F1, F2, F3, F4, F5.

Le moteur s’arrête, la base de faits est saturée (aucun faits nouveau ne peut être déduit).

Graphiquement, le fonctionnement du chaînage avant sur les règles est représenté par un exemple à la figure 1.3.

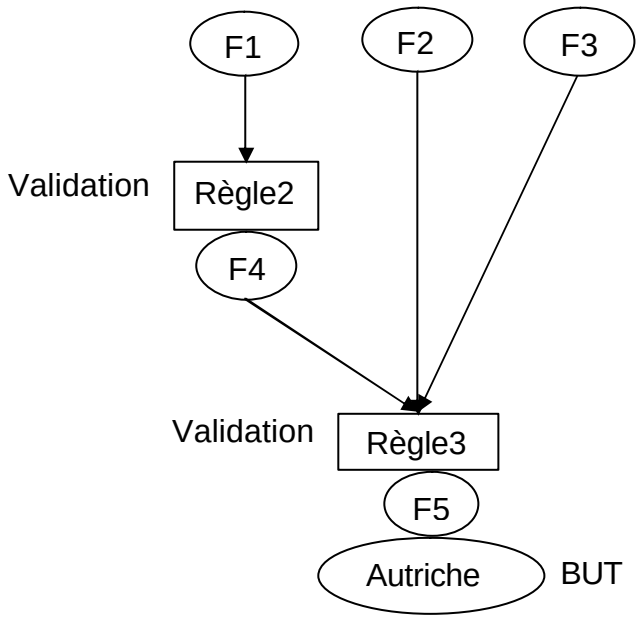


Figure.1.3 Le graphe du raisonnement en chaînage avant.

On remarque que cet algorithme n'indique pas comment choisir une règle applicable. C'est à ce niveau que la métaconnaissances du domaine intervient et permet de définir une stratégie de choix. Cet algorithme peut être très pénalisant pour certaines bases, l'algorithme de chaînage avant s'arrête toujours.

1.5 Représentation des connaissances

Nous allons présenter dans cette section les différents types de graphes, très connue dans le domaine de systèmes experts. Notre but est de mieux connaître les manipulations sur les graphes, leurs avantages ainsi que leurs problèmes particuliers.

1.5.1 Réseaux sémantiques

Issus des modèles psychologiques de Quillian et Raphael, (1968), les réseaux sémantiques sont des graphes dont les sommets représentent des objets, des événements ou des concepts et les arcs représentent des relations entre les sommets [2]. Si on reprend l'assertion précédente, elle sera représentée comme suit :

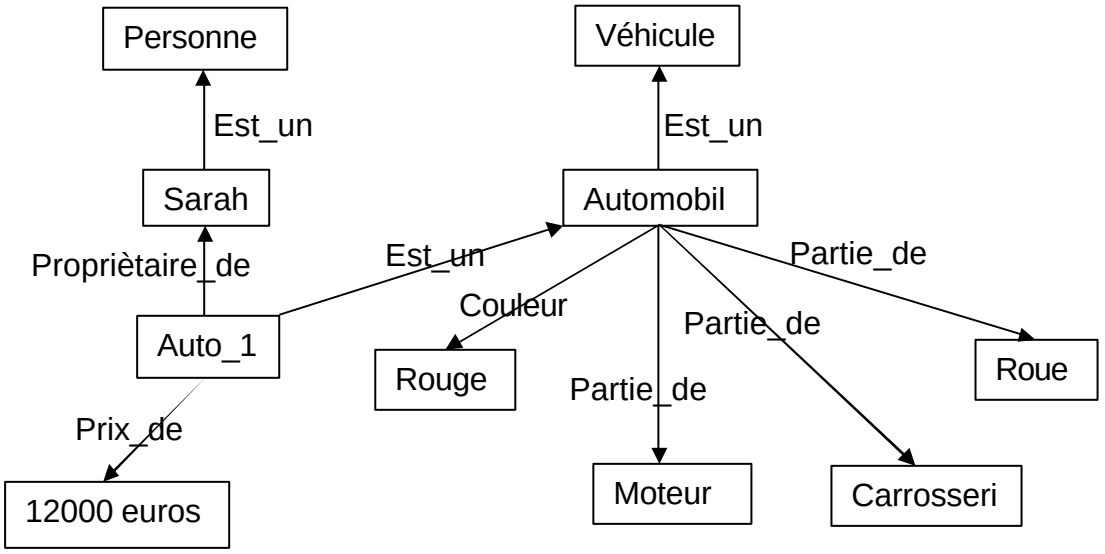


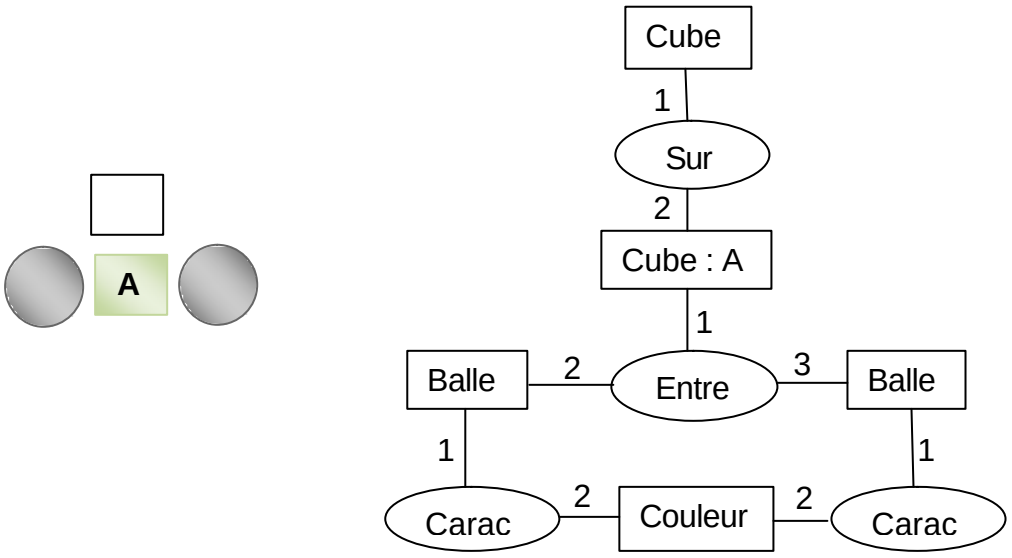
Figure 1.4 : Réseau sémantique.

Introduite pour représenter le sens des mots, cette technique a été utilisée dans le système expert Prospector (Recherche géologique) [2]. La représentation graphique représentée des connaissances statiques et dynamiques comme facilite la compréhension lorsqu'il s'agit pour l'utilisateur d'examiner le contenu de la base de connaissances. Son inconvénient majeur est la complexité de représentation et de gestion qui croit d'une manière significative avec la taille de la connaissance [18].

1.5.2 Graphes conceptuels

Le formalisme des graphes conceptuels (Sowa, 1984) est dérivé à la fois des modèles de représentation de connaissances généraux de type "réseau sémantique" et des graphes existentiels de C.S. Peirce [19].

Les graphes conceptuels sont constitués de deux types de sommets: les sommets concepts (aussi appelés *concepts*) et les sommets relations (aussi appelés *relations conceptuelles)*. Chaque sommet (ou sommet) est muni d'une étiquette.
L'étiquette d'une relation conceptuelle est un type de relation, tandis que celle d'un concept est constituée d'un type de concept et d'un référent [20].
Un graphe conceptuel peut représenter la signification littérale d'une phrase. Par exemple, « Il y a un cube sur le cube A, ce cube A étant entre des balles, qui ont la même couleur ».



La figure1.5. Représentation par un graphe conceptuel.

Les graphes conceptuels ont été implémentés dans des outils de spécification et modélisation de logiciels. De plus, ils ont été aussi utilisés comme langage de

représentation de connaissance pour la génération de langage naturel et l'extraction d'information [21]. Ainsi les graphes conceptuels permettent de gérer un comportement dépendant de plusieurs situations sans avoir à utiliser toutes les ressources de la machine. Il est de plus, possible de gérer un groupe de situations simultanément liées ou non. Les graphes représentent donc une très bon modèle dès qu'il s'agit de faire des choix dépendants de plusieurs situations et acceptent aisément la complexité et l'abondance de situations.

1.5.3 Graphe contextuel

Un graphe contextuel est un graphe orienté acyclique avec une entrée unique et une sortie unique, et une organisation série-parallèle de sommets connectés entre eux par des arcs orientés [22]. Globalement, la forme générale d'un graphe contextuel est celle d'un ensemble de faisceaux. La Figure 3 présente un exemple de graphe contextuel. Les différents types de sommets sont: les actions Ai (représentées par des boîtes carrées), les éléments contextuels représentés par des paires {sommet contextuel (les gros cercles numérotés Cj), sommet de recombinaison (notés Ri)}, les sous-graphes et les groupements d'actions parallèles (les deux derniers items ne sont pas représentés dans la Figure 3). L'algorithme correspondant à un graphe contextuel se termine toujours [23].

Un chemin dans le graphe contextuel correspond à une séquence ordonnée d'éléments (les actions et les sommets contextuels et de recombinaison) dans le graphe contextuel depuis son entrée jusqu'à sa sortie. Chaque chemin correspond, au travers de la séquence d'actions correspondantes, à une pratique qui est donnée dans son contexte (les éléments contextuels sur le chemin). La notion d'activité considérée par les psychologues cogniticiens, trouve une expression directe dans les graphes contextuels (principalement basée sur la notion de sous-graphe). Par conséquent, les graphes contextuels donnent une représentation des connaissances et du raisonnement qui est directement interprétable par les opérateurs (dans leur langage).

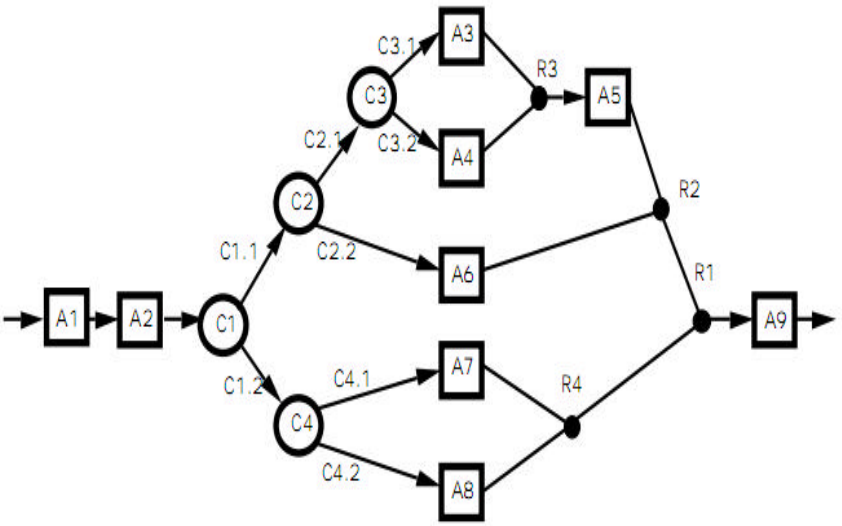


Figure 1.6 : Un exemple de graphe contextuel.

Le contexte du graphe contextuel dans la Figure 1.6 correspond aux éléments contextuels C1, C2, C3 et C4. Le contexte d'une action (par exemple l'action A3) est composé de deux parties, l'ensemble des éléments contextuels qui sont instanciés sur le chemin depuis l'entrée du graphe jusqu'à l'action (par exemple, le contexte de l'action A3 contient C1 avec la valeur C1.1, C2 avec la valeur C2.1 et C3 avec la valeur C3.1), et les éléments contextuels qui ne trouvent pas sur ce chemin (comme l'élément contextuel C4 dont l'instanciation est indifférente pour définir le contexte de l'action A3). Les éléments contextuels considérés par leur instanciation constituent le contexte procéduralisé, et le reste des éléments contextuels forme l'ensemble des connaissances contextuelles. Le contexte de l'action A3 se résume donc par :

- son contexte procéduralisé: {C1 avec la valeur C1.1, C2 avec la valeur C2.1, et C3 avec la valeur C3.1}, en supposant que les actions A1 et A2 soient exécutées;
- l'ensemble des connaissances contextuelles (ici réduit à l'élément contextuel C4).

Le contexte procéduralisé est une séquence ordonnée d'éléments contextuels instanciés. Nous le verrons par la suite, il existe un deuxième ordonnancement lié à la chronologie dans laquelle les éléments contextuels sont introduits dans le graphe contextuel par acquisition incrémentale de pratiques.

Les graphes contextuels permettent la représentation des pratiques, il n'est pas possible de développer à priori une représentation exhaustive de toutes les manières de résoudre un problème car le nombre de variantes contextuelles est très grand. Un système basé sur les graphes contextuels soit connaît une pratique utilisée par un opérateur pour résoudre un problème, soit va l'acquérir s'il ne la connaît pas.

1.5.4 Graphe d'induction

L'objectif des méthodes à base de graphe d'induction est de construire une fonction de classement représentable par un graphe : le graphe est construit en partant de la racine et en allant vers les feuilles. On parle de méthode d'induction descendante et on trouve, dans la littérature anglaise, le terme de TDIDT (Top Down Induction of Decision Trees) afin déqualifier les méthodes à base de graphe d'induction.

Un graphe est un ensemble dont les éléments sont des sommets reliés par des arcs et des chemins. Il y a deux types de sommets particuliers : sommet initial et sommet terminal ; un sommet initial, appelé aussi la racine, n'a pas de prédécesseur, un sommet terminal, appelé aussi une feuille, n'a pas de successeur. Dans les graphes d'induction il y a un sommet initial unique et plusieurs sommets terminaux. Ces graphes ne comportant pas de circuit on pourra parler d'arborescence (les arbres de décision sont un cas particulier des graphes d'induction). Par conséquent, l'ensemble des feuilles constitue une partition de l'échantillon d'apprentissage et le graphe tout entier représente la fonction de classement [24].

La dimension et la profondeur du graphe dépendent du choix des critères de partition et d'arrêt. Chaque feuille du graphe est étiquetée par la classe qu'ont les exemples sur ce sommet, les sommets intermédiaires sont associés aux attributs discriminants et chaque chemin correspond à une valeur possible de l'attribut. A partir d'un sommet intermédiaire, il y a autant de chemins que l'attribut a de valeurs. Si l'on veut classer un exemple, il faut partir de la racine, évaluer chaque attribut pour l'exemple considéré et choisir le chemin approprié. Le processus continu jusqu'à ce qu'une feuille soit rencontrée. Est attribuée à l'exemple, la classe associée à la feuille.

Les graphes d'induction sont un outil très satisfaisant pour l'extraction de connaissances à partir de données, une fonction de classement efficace et explicative [25].

1.5.5 Les arbres

Un arbre est un graphe orienté, sans cycles, dont les sommets portent une question, les arcs des réponses, et les feuilles des conclusions, ou des classes terminales [26].

Exemple d'arbre de décision :

Vais-je sortir mon chien ?

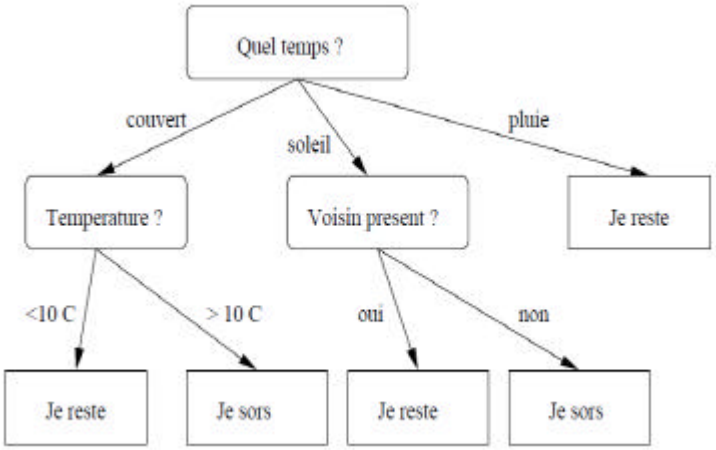


Figure 1.7 représente un arbre de décision.

Le chemin reliant une feuille à la racine de l'arbre peut être lu comme une règle de prédiction du type attribut-valeur « **SI** condition **ALORS** Conclusion».

Pour résoudre un problème de classification, on peut utiliser des règles de production. Les arbres sont une représentation commode de *fonctions de classifications*, moins puissantes que les règles de production mais plus faciles à utiliser.

Avantage des arbres :

1. Facile à comprendre et à utiliser.
2. Nombre de tests limités par le nombre d'attributs (de *question*).
3. Construction efficace (mais technique) à l'aide d'apprentissage par optimisation (pour obtenir un arbre petit et « correct »).

1.5.6 Les Réseaux Bayésiens

Un réseau bayésien est un système représentant la connaissance et permettant de calculer des probabilités conditionnelles apportant des solutions à différentes sortes de problématiques [27]. La structure de ce type de réseau est simple : un graphe dans lequel les sommets représentent des variables aléatoires, et les arcs (le graphe est donc orienté) reliant ces dernières sont rattachées à des probabilités conditionnelles. Notons que le graphe est acyclique : il ne contient pas de boucle. Les arcs représentent des relations entre variables qui sont soit déterministes, soit probabilistes. Ainsi, l'observation d'une ou plusieurs causes n'entraîne pas systématiquement l'effet ou les effets qui en dépendent, mais modifie seulement la probabilité de les observer. L'intérêt particulier des réseaux bayésiens est de tenir compte simultanément de connaissances a priori d'experts (dans le graphe) et de l'expérience contenue dans les données [28]. Domaines d'utilisation principaux par ce type de représentation : diagnostic (médical et industriel), analyse de risques, détection de spam, etc.… Donc le réseau bayésien est un modèle probabiliste graphique permettant d'acquérir, de capitaliser et d'exploiter des connaissances, né du besoin de créer des systèmes experts à base de probabilités. L'inférence sur les réseaux bayésiens est un problème NP-difficile, c'est pourquoi il était convenable de le voir de façon complète pour des instances réalisables et incomplète dans les autres cas [29].

1.5.7 Réseaux de Pétri

Les réseaux de Pétri sont nés de la thèse de Carl Adam Pétri [31], qui a proposé un graphe avec deux types de sommets : les places et les transitions. Les places représentées graphiquement par des cercles sont associées à des états et les transitions, représentées par des rectangles décrivent les changements d'états.

Une façon simple de modéliser une base de règles consiste à définir chaque règle par une transition et chaque fait par une place (un fait est vrai si un jeton se trouve dans la place). Une place additionnelle est ajoutée pour contrôler le tir de la règle et l'empêcher d'être tirée plus d'une fois (Figure 1.8).

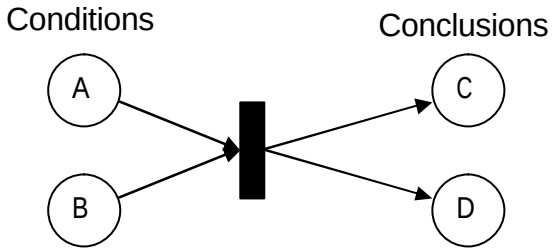


Figure.1.8. Réseau de Pétri représentant la règle : **SI** A **ET** B **ALORS** C **ET** D.

La règle de production d'un système expert modélisé par un réseau de Pétri pour simuler l'exécution de la base de connaissances. Ces méthodes doivent vérifier le modèle sous tous les états initiaux possibles afin de garantir l'absence d'incohérences. Malheureusement, ce test peut être parfois de calcul très coûteux [32].

1.5.8 Graphes orientés

La théorie de graphes est un outil très intéressant pour représenter dans l'ordre conceptuel des dépendances. D'ailleurs la théorie de graphes fournit des technique d'analytique pour les propriétés comme la connexité et l'accessibilité d'une manière régulariser et formelle, de plus l'analyse des formulations montre que les problèmes dans la base de règles mènent à l'existante de certaines propriété dans les graphes orientés. Ce développement est lié à la grande facilité de modéliser une situation concrète par un graphe [38].

Un graphe orienté G est donné par la définition d'un couple d'ensembles (V; U) où V est un ensemble non vide et U est une relation binaire portant sur V.
G se note G = (V; U) :

- Les éléments de V sont les sommets. Le cardinal de V est appelé l'ordre de G, noté n:
- Les éléments de U sont les arcs. Le cardinal de U est noté m:

Pour un arc u = (x; y), le sommet x est son origine ou son extrémité initiale, et le sommet y son extrémité terminale. Un arc de la forme u = (x; x) est appelé une boucle.

Soit une base de règles suivantes (BR) :

R1 : **SI** p **ET** q **ALORS** s

R2 : **SI** s **ET** t **ALORS** u

R3 : **SI** p **ET** q **ET** t **ALORS** u

R4 : **SI** s **ET** f **ALORS** p

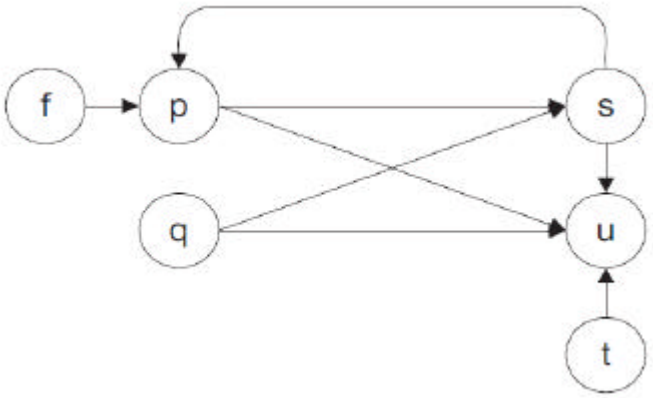


Figure 1.9 Graphe orienté représentant la base de règles(BR).

L'approche alternative à la vérification fondée sur des bases de règles, dans lequel le système est modélisé comme un graphe orienté, le contenu d'une base de règles, qui est codé, est exempt de problèmes de cohérence, l'exhaustivité, et la concision, est nécessaire pour éviter toute erreur de performance qui pourrait se produire pendant les séances de consultation avec la base de règles du système expert.

1.5.9 Les listes :

Une liste est une structure de données qui permet de stocker une séquence d'objets d'un même type. En cela, les listes ressemblent aux *tableaux*. Les opérations usuelles sur les listes (afficher, ajouter, supprimer, retourner, dupliquer...).

L'accès aux éléments d'une liste repose sur un principe simple : lorsqu'un pointeur contient l'adresse d'un élément il peut très facilement prendre pour valeur l'adresse de l'élément suivant. Les listes chaînées peuvent répondre à des besoins très divers. Dans de nombreuses situations de programmation trois types particuliers de listes chaînées me semblent mériter d'être signalés ici.

a. Listes ordonnées.
b. Listes circulaires.
c. Listes superposées.

Soit une règle de production : **SI** A **ET** B **ET** C **ALORS** D.
La figure 1.10 illustre cette représentation pour la liste exemple ci-dessus.

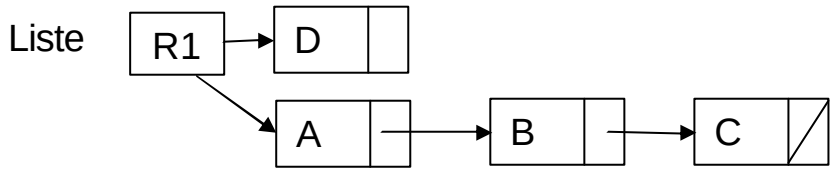


Figure.1.10 Représentation la règle de production à base de liste.

1.6 Conclusion

Dans ce chapitre, nous avons rappelé le principe général des systèmes experts. Nous avons décrit dans un premier temps les définitions et les modèles de représentation des connaissances des systèmes experts ainsi que les modes de raisonnement. Dans un second temps, nous avons donné les définitions des différents types de graphes, très connues dans le domaine de systèmes experts. L'avantage essentiel du système expert réside dans la capacité d'assimiler des connaissances considérables et de les exploiter exactement comme un expert, en

menant un dialogue pertinent avec ses utilisateurs. Les systèmes experts ont rapidement été mis à l’´ecart à cause des nombreux inconvénients qu’ils présentaient [15]. En effet, l’explosion rapide de la taille de la base de règles les rendaient difficilement maintenables, il devenait de plus en plus dur de vérifier la cohérence des règles entre elles, les erreurs dans les règles étaient indétectables et enfin, il était impossible d’assurer la complétude de l’expertise [16].

Notre objectif est de modéliser la règle de productions d’ordre 0, par un nouveau graphe à deux types de sommets. Ainsi l'algorithme de chaînage avant n’indique pas comment choisir la règle exécutée avant d’autre, donc la deuxième approche consiste à utiliser la théorie des graphes permettant de trier ou ordonner les règles de productions pour optimiser l'ordre d'exécution.

CHAPITRE 2
REPRESENTATIONS GRAPHIQUES DES BASES DE REGLES ET CONCEPTS DE LA THEORIE DES GRAPHES

Les règles de production permettent de formaliser un raisonnement suivant le schéma « **Si** conditions **alors** conclusions » [3,4]. Elles sont connues dans le domaine des systèmes experts, un succès qui est surtout dû à leur format très simple. Elles permettent par leur intégration du schéma d'implication, de simuler les déductions et les raisonnements logiques. Elles peuvent s'appliquer en « *chaînage avant* », « *chaînage arrière* » et « *chaînage mixte* ». Elles permettent ainsi d'exprimer des raisonnements à la fois complexes et naturels. Cependant, la sémantique des règles de production est moins claire que celle de la logique. La lenteur des systèmes qui les utilise et l'augmentation de leur nombre introduit une complication supplémentaire.

Dans la première partie de ce chapitre, nous citerons des travaux effectués pour modéliser une base de règles à l'aide de trois types de graphes.

La deuxième partie sera consacrée aux graphes orientés. Nous donnerons des concepts généraux et nous nous intéresserons au problème de la détection et de l'énumération des circuits de ces graphes ainsi qu'à celui du « *Feedback* ». Les algorithmes étudiés dans cette partie seront utilisés au chapitre 3 qui constitue l'essentiel de notre travail.

2.1 La base de règles à base de graphes

La représentation des connaissances revient à établir une correspondance entre le monde extérieur et un formalisme symbolique qui peut être traité par un ordinateur. Le domaine de la connaissance est trop vaste et varié pour être représenté et exploité par un formalisme unique. De ce fait plusieurs représentations sont utilisées, dont les avantages sont surtout techniques. Dans cette partie nous donnerons un aperçu sur les principaux travaux menés sur la modélisation des bases de règles à l'aide de graphes ainsi que les motivations et problématiques ayant poussé les chercheurs à investir ce domaine. Ces approches et méthodes proposées dans la littérature ont été détaillées selon différents aspects : les arbres, les réseaux de pétri, les graphes orientés. Nous présenterons enfin une discussion où nous montrerons les limites de ces approches en ce qui concerne nos techniques de modélisation.

2.1.1 Modélisation par un arbre *et/ou*

Un arbre *et/ou* est un graphe qui contient deux types de sommets à savoir les sommets ***ou*** et les sommets ***et***. Les sommets *ou* correspondent aux règles ayant une conclusion qui peut s'unifier avec le but cherché, chacune de ces règles est le début d'un chemin potentiel de résolution. Les sommets et correspondent aux conditions d'une règle donnée qui tente de résoudre un but qui s'est unifiée avec sa conclusion. Ces conditions deviendront à leurs tours des sous buts à résoudre.

L'arbre *et/ou* a été utilisé comme méthode de représentation de base pour comprendre le processus d'inférence d'un système expert [5]. Ses éléments fondamentaux sont des cercles et des arcs, où un cercle représente un sommet ou d'une variable et un arc indique une direction d'inférence. Un sommet utilisé dans un arbre *et/ou* est un sommet simple qui représente un élément unique de l'inférence, soit une condition lorsqu'il est utilisé dans la partie SI, soit d'une conclusion lorsqu'il est utilisé dans la partie ALORS. Dans le domaine de système expert, l'arbre *et/ou* a été utile en ce qui représente une inférence en chaînage arrière qui essaie de résoudre un problème en le divisant en sous problèmes et les résoudre individuellement. (Généralement, la partie ALORS une seule variable).

Par exemple, considérons une base de règle de production telle que:

R_1: SI C_1 et C_2 ALORS C_4.

R_2: Si C_2 et C_3 ALORS C_4.

L'arbre *et/ou* doit être plus robuste et efficace de sorte qu'il pourrait permettre une inférence conduite par une matrice de nombre entier. Pour ce faire, l'arbre *et/ou* se sont transformés par d'introduire un sommet composite. Que s'appelle un arbre prolongé *et/ou* [30]**.** Un sommet composite d'un arbre prolongé *et/ou*, ce qui représente une conjonction d'éléments de l'inférence. Une illustration d'arbre prolongé *et/ou* est représentée dans la figure 2.2, où C_5 et C_6 sont des sommets composite tout en C_1, C_2, C_3 et C_4 sont des sommets simples.

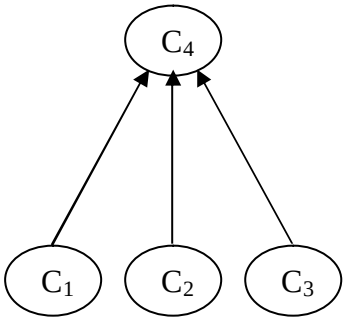


Figure.2.1 Un arbre *et/ou*.

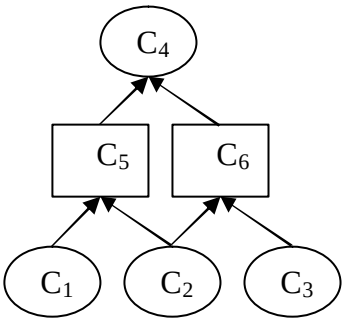


Figure.2.2 Un arbre prolongé *et/ou*.

L'arbre prolongé *et/ou* a plusieurs avantages, comme suit :

- Il peut favoriser une meilleure compréhension de l'architecture des règles, permettant de ce fait les pièges possibles cachés dans un ensemble de règles à clarifier, il peut permettre à des décideurs de fixer ces derniers avant de prendre des décisions spécifiques en utilisant les.
- Combiné avec quelques définitions et théorèmes mathématiques, il permet une rapide déduction.

L'inférence avec l'arbre prolongé *et/ou* peut être exécutée par un ensemble d'opérations simples de nombres entiers. Le but principal du deuxième avantage est de rendre possible la détermination de l'état de vérité [30].

2.1.2 Modélisation par des réseaux de pétri :

Un réseau de Pétri est d'abord un *graphe orienté* ayant *deux types de sommets* : des *places* représentées par des *cercles* et des *transitions* représentées par des *rectangles*. Les arcs du graphe ne peuvent relier que des places vers des transitions, ou des transitions vers des places (par d'arcs entre places ni entre transitions). Un réseau de Pétri décrit un *système dynamique à événements discrets*. Les places permettent la description des états (qui sont donc discrets) et les transitions permettent la description des événements (changements d'état) [11]. Les réseaux de pétri ont des applications diverses. Ils permettent dans l'industrie de modéliser la supervision des systèmes discontinus. Ils permettent également de modéliser des activités parallèles ou en concurrence des protocoles de communications ou d'étudier des langages formels [11]. Des techniques efficaces de vérification existent pour les systèmes à base de règles [31]. La plupart des approches se basent sur une logique de premier ordre. Dans le cas classique général, lorsqu'un fait est inféré, celui-ci demeure vrai pendant tout le processus d'inférence. On dit alors que le système est monotone. Toute modification subséquente de la valeur de vérité de ce fait sera perçue comme une anomalie du système. Pour des fins de vérification, un système de règles de production peut efficacement être modélisé par un réseau de Pétri. Vachon, J et al dans [32] explique comment ce formalisme s'avère particulièrement approprié pour réaliser cette vérification. Ce modèle formel offre des possibilités de vérification aussi bien dynamique (simulation) que statique (analyse des états). Tel qu'illustré par la Figure 2.3, une façon simple de modéliser une base de règles consiste à définir chaque règle par une transition et chaque fait par une place (un fait est vrai si un jeton se trouve dans la place). Une place additionnelle est ajoutée pour contrôler le tir de la règle et l'empêcher d'être tirée plus d'une fois.

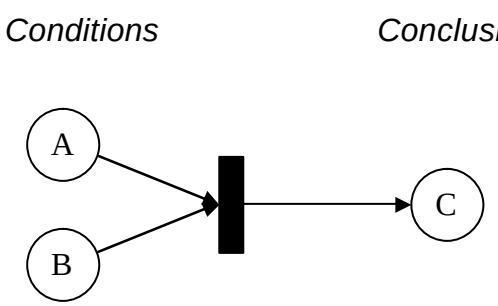


Figure.2.3. Réseau de Pétri représentant la règle : **SI** A **ET** B **ALORS** C.

Xudong, H. et al ont présenté dans [35] une technique spéciale accessibilité graphique basée sur les réseaux de Pétri (un type spécial de faible niveau de Pétri), la nouvelle modélisation est utilisée pour détecter et analyser les différents types d'erreurs structurelles. Le réseau de Pétri est utilisé pour simuler l'exécution de la base de connaissances. Ces méthodes doivent vérifier le modèle sous tous les états initiaux possibles afin de garantir l'absence d'incohérences. Malheureusement, ce test peut être parfois de calcul très coûteux [33]. Siminski .R dans [34] accorde une attention à l'analyse de l'utilisation possible des réseaux de Pétri et de matrice d'incidence que le mode de représentation des connaissances. Le document présente la relation entre les réseaux de Pétri et les réseaux des unités de décision et en même temps des points à l'utilisation éventuelle de réseaux de Pétri pour développer les caractéristiques des unités de décision.

2.1.2 Modélisation par les graphes orientés :

La théorie de graphes est un outil très intéressant pour représenter dans l'ordre conceptuel des dépendances. D'ailleurs la théorie de graphes fournit des techniques d'analyse pour les propriétés comme la connexité et l'accessibilité d'une manière régulière et formelle, de plus l'analyse des formulations montre que les problèmes dans la base de règles mènent à l'existence de certaines propriétés dans les graphes orientés.

Gursaran .G.S et al, présente dans [38] une approche alternative à la vérification fondée sur des bases de règles, dans lequel le système est modélisé comme un graphe orienté, le contenu d'une base de règles, qui est codé, est exempt de problèmes de cohérence, l'exhaustivité, et la concision, est nécessaire pour éviter toute erreur de performance qui pourrait se produire pendant les séances de consultation avec la base de règles du système expert. La vérification du contenu d'un type spécifique de base de règles.

Siminski .R dans [39] a proposé les différents types de représentation graphique à base de graphes orienté. L'objectif principal de ce travail est de modéliser la base de règles d'une unité de décision par graphe orienté et son utilisation pour la tâche de vérification.

Arman, N dans [40] a modélisé la base de règles par un graphe orienté pondéré. L'approche utilise des arbres de recouvrement minimaux pour vérifier une base de règles pour différents types de défauts.

2.1.3 Discussion et conclusion

Par rapport aux approches de la modélisation précédente de base de règles, certains systèmes de représentations sont plus appropriés et efficaces pour certains modèles. Toutefois, pour choisir la meilleure représentation il faut prendre aussi en compte les données fournies au système et la manière dont il s'en sert. Chaque représentation possède des avantages et des inconvénients, et en fonction de nos besoins, nous nous sommes particulièrement intéressé aux aspects suivants : la représentation des systèmes de raisonnement à base de règles par les graphes.

Notre objectif de recherche est la conception d'un modèle qui produit automatiquement des faits. Les entrées de ce modèle sont les informations déduites par le moteur de chaînage avant des bases de règles. Dans cette partie nous avons étudié l'aspect informatique de la représentation et de la prise de décision pour le choix d'un graphe. La plus part des techniques de modélisation précédemment utilisée pour la vérification des règles où avec le raisonnement de chaînage arrière comme l'utilisation des arbres. Mais notre but est de trouver une autre modélisation par graphe orienté qui utilise le principe ou raisonnement de chaînage avant. Dans le cadre de notre travail, nous allons essayer de proposer deux modèles, le premier représente la base de règles et le deuxième représente l'ordonnancement des règles, afin d'améliorer la pertinence des résultats. Ces propositions sont relatives à chacune des deux types d'approches (graphe deux types de sommets et graphe orienté simple).

2.2 Concepts de la théorie de graphes

Les graphes représentent un instrument puissant pour modéliser de nombreux problèmes combinatoires, qui seraient sans cela difficilement abordables par des techniques classiques [41, 42].

2.2.1 Définitions

Définition 1.1

Un graphe orienté $G=(X, U)$ est déterminé par les données suivantes :

- Un ensemble X dont les éléments sont appelés sommets. Si $N=card(X)$ est le nombre de sommets, on dit que le graphe G est d'ordre N.

- Un ensemble U dont les éléments sont des couples ordonnés de sommets appelés arcs. Si $u= (i, j)$ est un arc de G. i est l'extrémité initiale de u et j est l'extrémité terminale de u. on notera $card (U)=M$.

Définition 1.2

Un multigraphe est un couple $G=(X, U)$, dans lequel X est un ensemble de sommets, et U est une famille d'arcs $U= (u_1, u_2,\ldots, u_m)$ [43]. Cette définition permet de traiter des graphes dont plusieurs arcs auraient la même origine et la même extrémité, d'où le nom de multigraphes.

Définition 1.3

Un graphe orienté valué $G=(X, U, W)$. X désigne un ensemble de N sommets et U un ensemble de M arcs. $W (i, j)$, aussi noté W_{ij}, est la valuation (aussi appelée poids ou coût) de l'arc (i, j), par exemple une distance, un coût de transport, ou un temps de parcours.

Définition 1.4

Un chemin p_{ij} ou $p= (i, j)$ joignant deux sommets i et j d'un graphe est une suite d'arcs :

$$P (i, j) = ((i, j_1), (j_1, j_2)\ldots (j_{t-1}, j)).$$

Le chemin est souvent représenté par la suite associée de sommets :

$$(i, j_1, j_2\ldots j_{t-1}, j).$$

- Un chemin p_{ij} dont tous les sommets sont distincts est un chemin *élémentaire*.

- Le chemin p_{ij} est dit simple s'il ne passe pas deux fois par un même arc.
- Un circuit est un chemin fermé, i.e. un chemin dont l'extrémité terminale coïncide avec l'extrémité initiale.

Définition 1.5

Le degré de x_i noté par $d(x_i)$ est le nombre de sommets adjacents à x_i.

- Le demi-degré extérieur de x_i, est le nombre d'arcs ayant x_i comme extrémité initiale ; $d^+(x_i)$.
- Le demi-degré intérieur de x_i, est le nombre d'arcs ayant x_i comme extrémité finale ; $d^-(x_i)$.
- Le degré de x_i est $d(x_i)=d^+(x_i) + d^-(x_i)$. Le degré d'un sommet d'un graphe non orienté est le nombre d'arêtes qui lui sont incidentes.

Exemple 1.1

Un exemple de graphe orienté simple :

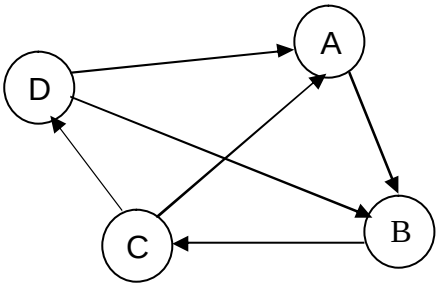


Figure 2.4 Graphe orienté simple.

On peut représenter un digraphe par une *matrice d'adjacences* [35]. Une matrice *n x n* est un tableau de *n* lignes et *n* colonnes. Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe.

Le graphe de la figure (2.5) peut être représenté par la matrice d'adjacence suivante :

	A	B	C	D
A	0	1	0	0
B	0	0	1	0
C	1	0	0	1
D	1	1	0	0

Figure 2.5 Matrice d'adjacences.

Cette matrice à plusieurs caractéristiques :

1. Elle est carrée : le nombre de lignes est égale au nombre de colonnes.
2. Il n'y a que des zéros sur la diagonale. Un *1* sur la diagonale indiquerait une boucle.
3. Contrairement au cas non orienté, elle n'est pas symétrique.

Propriétés 2.1

- la somme des éléments de la i^{ieme} ligne de la matrice d'adjacences est égale au degré sortant $d^+(x)$ du sommet x_i de G.
- la somme des éléments de la j^{ieme} colonne de la matrice d'adjacences est égale au degré entrant de $d^-(x)$ du sommet x_j de G.

2.2.2 Circuit

Un circuit est un chemin eulérien dont les extrémités sont confondues. Ce type particulier de circuits est particulièrement intéressant car on le retrouve dans les nombreux problèmes appliqués [45].

2.2.2.1 Méthode de détection des circuits

Les problèmes de détection et énumération des circuits dans les graphes orientés sont toujours d'importance fondamentale. Il existe des algorithmes qui déterminent l'existence de circuits, parmi les algorithmes de Marimont, [46] utilisés pour déterminer l'existence des circuits dans des graphes orientés.

Deux étapes très importantes sont utilisées dans l'algorithme de Marimont

- élimination des boucles qui sont des circuits élémentaires.
- élimination progressive de tous les sommets qui contiennent que des arcs entrants ou que des arcs sortants (aucun circuit ne contient ces sommets).

L'algorithme de détection de circuit est le suivant :

Tant que : s'il existe x dans G tel que $d^-(x)=0$ où $d^+(x)=0$
    Faire $G \leftarrow G/x$
    Si G= $\varnothing$
        Alors G n'a pas de circuit,
    Sinon
        G à un circuit.

Figure .2.6 Algorithme de détection des circuits.

On calculé les demi-degrés négatifs $d^-(x)$ et les demi-degrés positifs $d^+(x)$ de chaque sommet x du graphe. Cet algorithme est inspiré du parcours en largeur d'un graphe (file d'attente). La complexité de ce dernier est en $O(n+m)$ (calculs des demi-degrés négatifs inclus).

2.2.2.2 Méthode d'énumération des circuits d'un digraphe :

Divers algorithmes on été formulés pour énumérer les circuits dans les graphes orientés. Ces derniers sont déterminés en temps exponentiel. Un algorithme d'énumération des circuits pour les graphes orientés à été proposé en [47] ou il a utilisé l'idée de Jonson pour les multiple-arcs et les individu-arcs. Cet algorithme donne une exécution efficace et un rendement de mémoire élevé dans le langage de la programmation de D. L'algorithme original de Johnson a été donné dans un

Pascal comme un pseudo code et a utilisé un système entier 1… N étiquetage. Ce travail a utilisé une structure d'une matrice d'adjacente incapable de supporte des boucles et des arcs multiples. Hawick et al dans [47] ont utilisé un ensemble d'arcs, où chacun des arcs est spécifié précisément par sa source et les indices de sommet de destination.

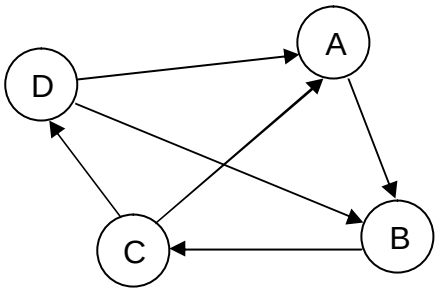


Figure.2.7 Recherche des circuits d'un Digraphe.

La figure 2.7 montre un petit test graphique de 4 sommets et 6 arcs, reliés de manière à donner seulement trois circuits (A-B-C-D, A-B-C et B-C-D). Notre structure de stockage pour la liste d'adjacence Ak [i] [j], où le i^{iemme} sommet a m = Ak [i] [0] arcs émanant de lui, avec leurs sommets destination stockées dans Ak [i] [1. m]. Les structures de la liste D peuvent être manipulées comme des tableaux dynamiques et une taille ajustée, même pour les applications qui peuvent avoir de multiples donc m =N. L'algorithme procédé par Johnson, stipule que les sommets qui sont impliqués dans certains circuits sont connectés. L'algorithme maintient une pile de sommets à partir des laquelle il a essayé de construire un arbre de recherche récursive pour les circuits. Le circuit de procédure est appelé de manière récursive à la recherche d'un circuit et la procédure de déblocage, qui gère les chemins élémentaires pour éviter la duplication des circuits. L'algorithme énumère donc les circuits élémentaires du graphe selon l'ordre des sommets.

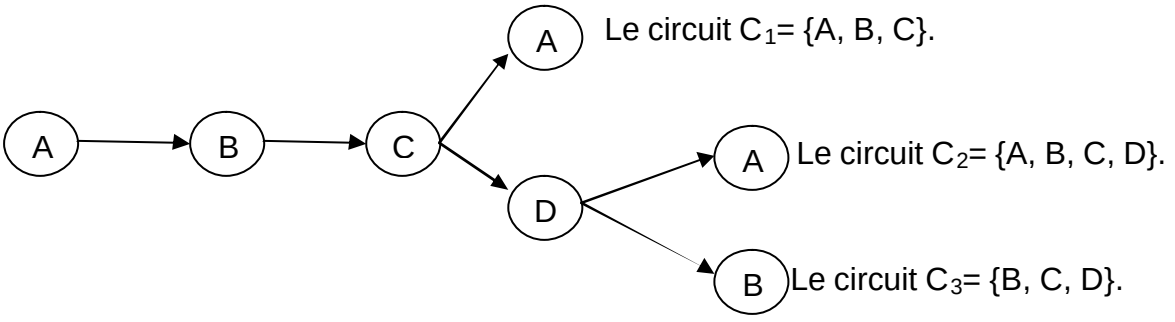


Figure.2.8 Un arbre de recherche récursive pour les circuits.

2.2.3 Mise en ordre d'un graphe sans circuit

Soit un graphe orienté *G(X, U)* sans circuit ou *graphe acyclique*. Ordonner un graphe orienté revient à disposer dans un certain ordre ses sommets, tel que les arcs soient dans le même sens. On définit pour cela les différents niveaux des sommets du graphe. Alors la mise en ordre d'un graphe orienté sans circuit est la détermination des niveaux $N_0, N_1, \ldots, N_k$ qui forme une partition de X [48].

L'algorithme de la mise en ordre d'un graphe sans circuit est le suivant :

A chaque itération, on recherche tous les sommets qui n'ont pas de précédent. Ils constituent un niveau. On efface ces sommets avant de rechercher le niveau suivant.

A partir de la matrice d'adjacence on peut faire la mise en ordre d'un graphe orienté sans circuit, nous supprimons à chaque itération, tous les sommets qui correspondent à une colonne nulle, donc qui n'ont pas de précédents**.**

Exemple 1.2

	A	B	C	D	E
A	0	0	1	0	1
B	0	0	1	0	1
C	0	0	0	1	1
D	0	0	0	0	1
E	0	0	0	0	0

Figure 2.9 : La matrice d'adjacence.

Nous appliquons le principe de l'algorithme de la mise en ordre à la matrice d'adjacence précédent, les sommets A et B qui reçoit des colonnes nulles donc on les mets dans le premier niveau d'ordre. Le même principe est suivi par les autres sommets.

La mise en ordre d'un graphe sans circuit par la matrice d'adjacence.

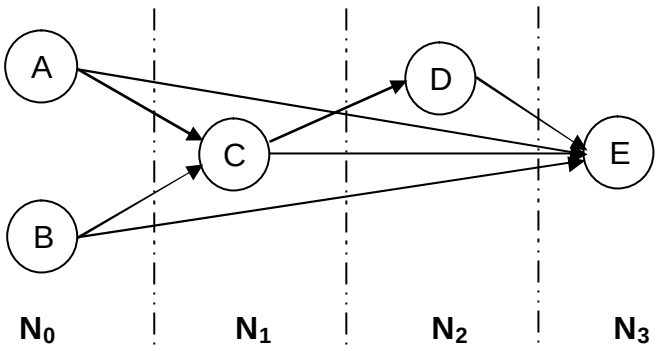


Figure 2.10 : Décomposition en niveaux.

Niveaux= $\mathbf{N_0}$= {A, B} ; $\mathbf{N_1}$={c} ; $\mathbf{N_2}$= {D} ; $\mathbf{N_3}$={E}.

2.2.4 Feedback vertex set:

Le *feedback vertex set* est un des premiers problèmes dont la *NP*-complétude à été prouvée (il fait parti des 21 problèmes *NP*-complet de Karp [49]). Le *feedback vertex set,* ou *FVS* d'un graphe orienté, est un sous-ensemble des sommets de *G* dont la suppression induit un sous-graphe *G'* de *G* qui ne contient pas de circuits, quand la cardinalité d'un *FVS* est minimale, celui-ci est appelé *minimum feedback vertex set,* ou *MFVS*. Le *MFVS,* comme nous l'avons déjà mentionné, trouve ses applications dans différents domaines de l'informatique [50]. Le problème consiste à déterminer un *MFVS* dans un graphe *G* qui est *NP*-complet en générale. Cependant, un grand nombre d'articles ont traite de ce problème, pour lequel il a été démontré que la détermination d'un *MFVS* devient polynomiale dans des topologies spécifiques [51]. De plus, dans des papiers plus récents on trouve des méthodes permettant de déterminer des bornes sur la cardinalité d'un MFVS dans des topologies précises [52].

2.2.4.1 Problèmes de Feedback :

Nous présentons dans cette section, plusieurs règles qui permettent très souvent de réduire un graphe orienté tout en préservant la solution cherchée (le MFVS). En effet, le but ultime étant d'identifier tous les circuits du graphe, il est beaucoup plus facile et rapide d'appliquer des heuristiques voire des algorithmes exacts sur des graphes préalablement réduits. On note *G (X, U)* le graphe orienté *G* avec *X* l'ensemble des sommets et *U* l'ensemble des arcs de *G*.

? Règles de réduction

a. La règle R1 [53]

Si un sommet du graphe n'a pas d'arc entrant ou sortant alors celui-ci ne peut pas faire partie d'un circuit. On supprime donc le sommet du graphe ainsi que tous ses arcs adjacents.

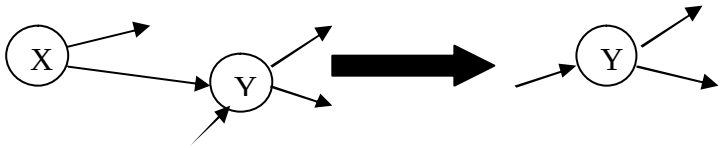


Figure 2.11 : Suppression du sommet du graphe qui reçoit des arcs sortants.

Figure 2.12 : Suppression du sommet du graphe qui reçoit des arcs entrants.

b. La règle d'absorption R2 [53]

Dans le graphe orienté, le sommet X n'a qu'un seul arc entrant provenant du sommet Y. Ainsi, tout circuit passant par X passera forcément par Y. Le sommet X peut donc être absorbé par le sommet Y. C'est-à-dire que X est supprimé et que tous les arcs de X deviennent des arcs de Y.

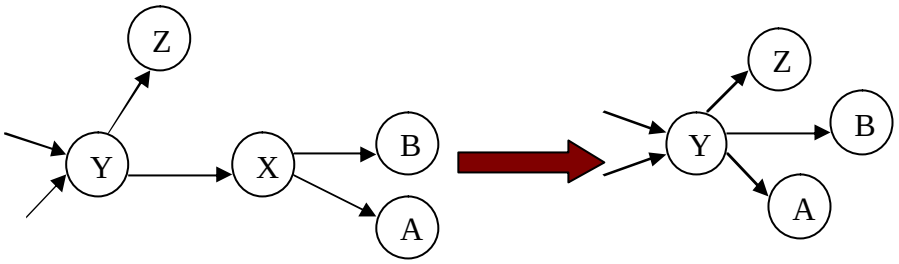


Figure 2.13 : Y absorbe X.

De même, si un sommet X n'a qu'un seul arc sortant vers un sommet Z alors Z peut absorber X, c'est-à-dire que X est supprimée et les arcs de X deviennent arcs de Z.

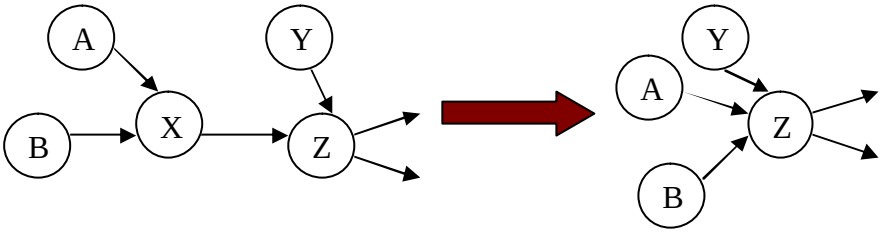


Figure 2.14 Z absorbe X.

c. La règle de Suppression des circuits unitaires R3 [53]

Un sommet possédant un arc rebouclé sur lui-même est un cas particulier, il forme un circuit unitaire. Supprimer ce sommet est donc la seule façon de couper ce circuit. Ainsi, tous les sommets d'un graphe ayant cette caractéristique devront être automatiquement supprimés et mis dans la liste minimale de sommet à supprimer pour couper tous les circuits (MFVS).

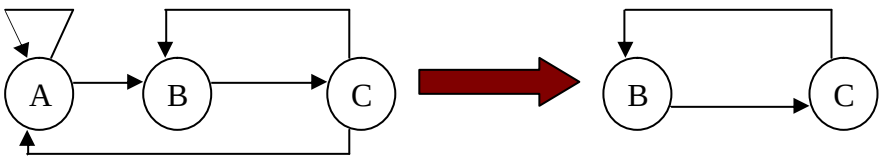


Figure 2.15 : R3 appliquée au sommet A.

Lin et Jou [53] utilisent les techniques de réduction (R1, R2, R3) et partitionnement en composantes fortement connexes. Ils ajoutent cependant deux nouvelles techniques : La première consiste à rechercher les cliques du graphe. Un graphe est une clique si tout sommet a un arc vers tous les autres sauf vers lui-même. Les auteurs isolent ces cliques et partitionnent ainsi encore plus le graphe initial. La deuxième technique consiste à rechercher les circuits essentiels (s'il ne contient pas d'autres circuits) ce qui permet également souvent de réduire le graphe. Finalement, ils utilisent encore une fois la technique classique du branch and bound afin d'obtenir une solution optimale. De manière générale, l'algorithme exposé précédemment est exact mais aussi de complexité exponentielle. Ils ne seront donc pas applicables à des circuits très complexes et fortement séquentiels. Néanmoins, plusieurs méthodes approximatives ont été développées, donnant en général d'assez bons résultats.

Une heuristique basée sur une énumération des circuits du graphe bornée dans le temps proposé par Lin, C et al [54]. L'ensemble des circuits obtenus doit être représentatif c'est à dire qu'ils ne doivent pas contenir eux-mêmes d'autres circuits, ce sont donc les circuits essentiels. Puis, ils sélectionnent le sommet apparu dans le plus grand nombre de circuits trouvés, ils le suppriment du graphe, l'ajoutent au MFVS et réitèrent l'énumération bornée jusqu'à avoir un graphe vide et donc un ensemble de sommets (non optimal en général) capable de couper tous les circuits. VOGEL, I dans [55] a donné quatre procédures de réduction de graphe, la première consiste à décomposer le graphe en plusieurs sous graphes (*Composante Fortement Connexe)* et les trois autres permettent de supprimer certains sommets et arcs (les règles de réduction) à chaque sous graphe.

2.3 Conclusion

La majorité des auteurs ont trouvé le nombre minimal de feedback qui une fois supprimés rendent le graphe sans circuit. Il n'existe donc pas de solution algorithmique dont la complexité croît de manière polynomiale avec le nombre de feedback dans le graphe. Nous avons donc, d'un côté des algorithmes exacts basés sur une recherche exhaustive pour les graphes de taille raisonnable et des

heuristiques pour les autres. Dans mon travail la solution nous permet de connaître le nombre minimal de sommets et le problème de choix ďun sommet le plus utilisé en effectuant un algorithme.

CHAPITRE 3

MODELISATION ET ORDONNANCEMENT DES BASES DE REGLES

Beaucoup de techniques de transformation des bases de règles ont été proposées dans la littérature. Réseaux de Pétri ont été décrites dans [34]. Dans cette approche, une base de règles sera modélisée comme un réseau de Pétri. Ensuite, la transition de relation modélisée dans un réseau de Pétri peut être résumée sous la forme d'une matrice d'incidence. Une approche graphe orienté présentée dans [35], où la base de règles est modélisée comme un graphe orientée et le processus de détection d'anomalie est réduit à accessibilité entre les sommets. Arman, N dans [40] a modélisé la base de règles par un graphe orienté pondéré. L'approche utilise des arbres de recouvrement minimaux pour vérifier une base de règles pour différents types de défauts. Dans cette thèse, nous utilisons la technique de transformation où la base de règles est modélisée comme graphe. Nous allons présenter dans ce chapitre les méthodes de modélisation que nous avons implémentées. Nous donnons dans une première section un modèle proposé. Dans une deuxième partie, nous proposons une autre modélisation par graphe orienté simple (Graphe de Relation) permettant de trier ou d'ordonner la base de règles pour optimiser l'ordonnancement de ces règles.

3.1 Le modèle proposé:

Les techniques de transformation des bases de règles ont été mentionnées précédent. La base de règles modélisées par les réseaux de Pétri ont été décrites dans [34, 35] et le graphe orienté dans [38, 39, 40].

Dans notre travail, nous utilisons la technique de transformation où la base de règles est modélisée comme un multigraphe orienté pondéré et composé de deux ensembles X et Y.

X : contient les sommets des conclusions et disjoints.

Y : contient les sommets des conditions.

Le modèle proposé comporte une architecture générale de fonctionnement et une représentation de la base de règles sous forme de graphe. Cette modélisation de graphes forme un réseau dans lequel circule un flux de pensées représentées par les simples chemins. Un graphe est lui-même représenté par des sommets et des liens étiquetés, à partir de la structure de graphe conceptuel de Sowa [29]. Nous représenterons notamment un nouveau graphe qui contient deux types de sommets. Le graphe proposé est un multigraphe fini et composé de deux types de sommets. Nous avons vu que dans la forme graphique, les conclusions sont représentées par des rectangles, les conditions par des cercles et les arcs correspondant à la réglementation (les trajets parcourus les règles). Chaque règle a un poids qui représente le coût de l'application de cette règle (la place des règles dans la base de règles) ; on parle alors d'arcs pondérés. Pour simplifier, nous mettons, dans la figure suivante, des pondérations simples (des nombres naturels).

- Les sommets conclusions, aussi appelés partie précédent. Un sommet conclusion contient des arcs sortant seulement étiqueté par un nombre qui représente la place de la règle dans la base de règles. Par exemple dans la figure 3.2, le sommet conclusion ‘’B’’ étiqueté par deux arcs “2“ et “3“ qui représente les places des règles.

- ✍ Les sommets conditions, aussi nommés prémisses ou simplement partie antécédente. Les sommets conditions spécifient les rapports entre les conclusions. Ils sont aussi reliés par d'arc étiquetés. Chaque sommet condition possède un arc ou plus.

Dans notre modélisation, nous utilisons la technique de transformation où la base de règle est modélisée comme un graphe à deux types de sommets. Les bases de règles sont souvent modélisées par des graphes. Les représentations par des graphes différents cependant d'un modèle à un autre.

Exemple 3.1.

La figure 3.4 donne l'exemple de la modélisation d'une base de règle, qui sera l'outil principal de notre méthode de modélisation en considérant quatre règles dans la figure 3.1.

- R1 : ***SI*** B ***et*** C ***et*** F ***ALORS*** A.
- R2 : ***SI*** A ***et*** D ***et*** H ***ALORS*** B.
- R3 : ***SI*** A ***et*** E ***et*** D ***et*** H ***ALORS*** B.
- R4 : ***SI*** E ***et*** D ***et*** F ***ALORS*** C.

Figure 3.1 : Base de règles.

La figure ci-dessous présente le graphe proposé. Les rectangles représentent les conclusions et les cercles représentent les conditions.

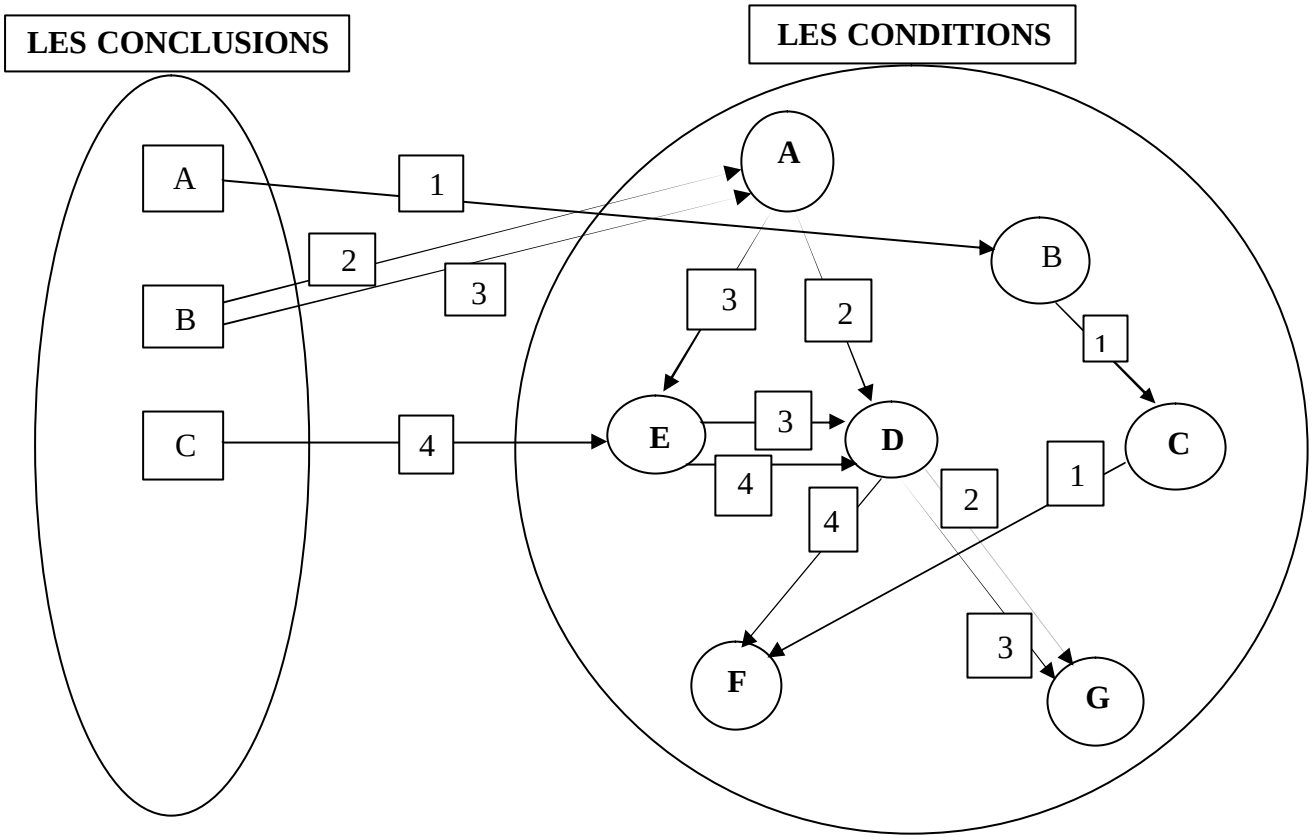


Figure 3.2 : Base de règles en modèle graphe proposé.

Il n'existe pas de cycles car si on a un cycle on prend une règle qui contient deux fois la même condition chose qui est impossible. S'il est possible par cette modélisation de faire la vérification de la règle sur les problèmes de cohérence, d'exhaustivité, et de concision pour évitée toute erreur de performance qui pourrait se produire pendant les séances de consultation avec la base de règles du système de raisonnement. Nous utilisons le parcours de graphe pour calculer les chemins simples couvrants les même poids d'arc, car chaque chemin représente une règle dans la base de règles du système de raisonnement et nous appliquons le même principe sur le moteur d'inférence de chaînage avant.

Les chemins calculent par le graphe précédent :

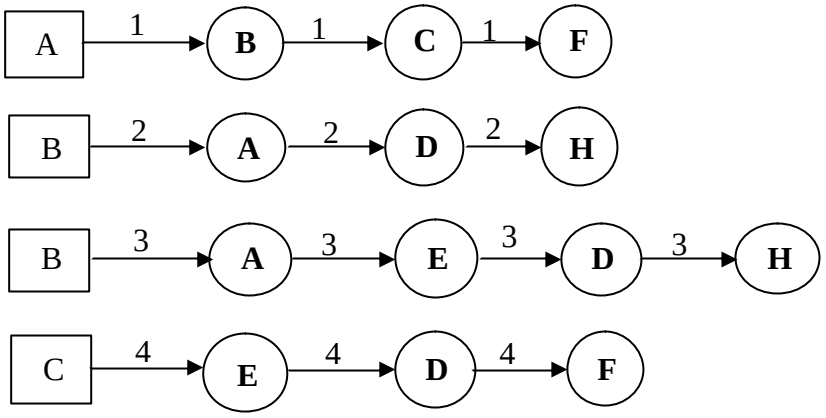


Figure 3.3 : Les chemins représente nt les règles.

Donc notre travail qui consiste à modéliser le système de raisonnement à base de règles implémentées à base de liste à été adopté, comme nous modifions l'implémentation pour la faire à base de graphe mais le problème reste sur la règle applicable avant d'autre. Donc nous passons à la deuxième modélisation par les graphes orientés pour trier ou ordonner la base de règles ou tour d'exécutions.

3.2 L'ordonnancement de la base de règles :

Dans la deuxième partie nous entamons la modélisation de base de règles d'un système de raisonnement à base de graphes. On fait la modélisation avec le graphe orienté simple (Digraphe) car le graphe permet de représenter simplement la structure, les connections, les cheminements possibles, et aussi une structure de données puissantes pour l'informatique. Les graphes orientés offrent un certain nombre d'avantages à la visualisation de l'information, avec le plus important :

- *compréhensibilité*: l'information qui a le graphe orienté peut être comprise facilement et exactement par les humains

- *l'expressivement*: la topologie de graphe orienté soutient l'information non triviale.

En outre, dans le cas de la représentation graphique des règles, les graphes orientés semblent être extrêmement appropriés. Le graphe orienté obtenu fourni des techniques d'analyse. Dans la première partie nous utilisons le graphe obtenu, pour

arranger les règles par ordre d'exécution. Afin de savoir quelle est la règle obtenue par l'autre, nous devons appliquer ce qui est connu dans la théorie des graphes, la mise en ordre d'un graphe orienté, il faut que le graphe obtenu soit un graphe orienté acyclique ou *DAG* (Directed Acyclic Graph), c'est-à-dire un graphe qui ne possède pas de circuit. Le problème qui contraint notre travail, réside dans l'existence de ces circuits, que nous allons par conséquent chercher à éliminer. Ce problème est appelé *feedback vertex set* [49]. Les bases de règles qui s'écrivent sous la forme de règle de productions, posent une importante question pour la vérification des systèmes de raisonnement à base de règles [38]. La théorie des graphes est un outil très intéressant pour représenter dans l'ordre conceptuel des dépendances. D'ailleurs la théorie des graphes fournit des techniques analytiques pour les propriétés comme la connexité et l'accessibilité d'une manière régulière et formelle, de plus l'analyse des formulations montre que les problèmes dans la base de règles mènent à l'existence de certaines propriétés dans les graphes orientés. Dans [39] ont proposé les différents types de représentations graphiques à base de graphes orientés. L'objectif principal de ce travail est de modéliser la base de règles d'une unité de décision par un graphe orienté et son utilisation pour la vérification. La base de règles est représentée dans l'article [40] par un graphe orienté pondéré. L'approche utilise des arbres de recouvrement minimaux, pour vérifier une base de règles pour différents types de défauts.

3.2.1 La partie modélisation

On appelle graphe orienté ou digraphe *G (X, U)* la donnée d'un ensemble *X* dont les éléments sont appelés sommets et d'une partie *U* de *X* × *X* dont les éléments sont appelés arcs. Nous pouvons présenter un exemple d'une base de règles comme un graphe orienté, les sommets de graphe représentent les règles, et Les arcs entre les sommets définissent la relation entre les règles.

Exemple 3.3.

Nous considérons la base de règle suivante qui contient des règles. Pour la simplification de la représentation, nous donnons un exemple d'une réunion d'amis suivante (décrivant des règles à respecter lors des invitations à une réunion d'amis) :

- R_1 : **SI** Benoît **ET** Djamel **ET** Emma **ALORS** Félix.
- R_2 : **SI** Gaelle **ET** Djamel **ALORS** Amandine.
- R_3 : **SI** Cloé **ET** Félix **ALORS** Amandine.
- R_4 : **SI** Benoît **ALORS** Xena.
- R_5 : **SI** Djamel **ALORS** Emma.
- R_6 : **SI** Xena **ET** Amandine **ALORS** Habiba.
- R_7 : **SI** Cloé **ALORS** Djamel.
- R_8 : **SI** Xena **ET** Cloé **ALORS** Amandine.
- R_9 : **SI** Xena **ET** Benoît **ALORS** Djamel.

Figure 3.4 : La base de règles d'une réunion d'amis.

Les sommets représentent les règles et les arcs représentent les relations entre les règles, par exemple la conclusion « Félix » de la règle R_1 appartient aux conditions de la règle R_3. Donc nous avons ajouté un arc de R_1 vers R_3. Par cette modélisation, nous trouvons la figure du graphe orienté suivant, qui représente la base de règles.

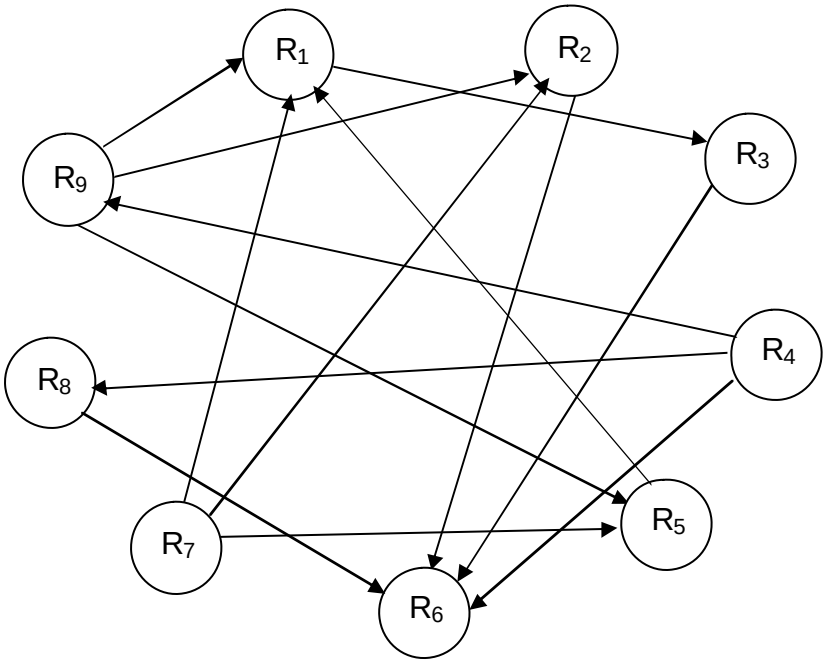


Figure 3.5 : Graphe de Relation des Règles.

Le graphe de relation de règle contient des sommets et des arcs. Les règles avec le terrain communal, les conditions dans leurs antécédents et la conclusion pourraient être naturellement connexes. Chaque sommet correspond à une règle. Les arcs entre les sommets, définissent la relation entre les règles. Dans le graphe précédent, il n'existe pas de boucles ni d'arcs parallèles, car s'il existe des boucles cela veux dire que la conclusion d'une règle est dans les mêmes conditions de cette règle : contradiction (ou impossible). De même, si nous avons des arcs parallèles, cela veux dire que la conclusion d'une règle, est deux fois dans la condition d'une autre règle, *i.e.* la règle qui contient deux fois la même condition, s'appelle "la redondance".

3.2.2 La matrice d'adjacence

Un certain nombre de représentations existent pour décrire un graphe. En particulier, elles ne sont pas équivalentes du point de vue efficacité des algorithmes. On distingue principalement la représentation par matrice d'adjacence, par matrice d'incidence sommets-arcs (ou sommets arêtes dans le cas non orienté) et par listes d'adjacence.

On considère un graphe orienté. La matrice d'adjacence qui fait correspondre les sommets (initiale) d'origine des arcs (placés en ligne dans la matrice) aux sommets destination (placés en colonne). Dans le formalisme *matrice booléenne*, l'existence d'un arc (*Ri, Rj*) se traduit par la présence d'un 1 à l'intersection de la ligne *Ri* et de la colonne *Rj* ; c'est-à-dire la conclusion de la règle Ri dans les conditions de la règle Rj ; l'absence d'arc par la présence d'un 0, signifie qu'il n'existe pas de relation entre les conditions, et les conclusions entre les règles Ri et Rj.

Donc nous avons représenté le graphe orienté de la figure 3.5 par la matrice d'adjacence suivante :

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
R_1	0	0	1	0	0	0	0	0	0
R_2	0	0	0	0	0	1	0	0	0
R_3	0	0	0	0	0	1	0	0	0
R_4	0	0	0	0	0	1	0	1	1
R_5	1	0	0	0	0	0	0	0	0
R_6	0	0	0	0	0	0	0	0	0
R_7	1	1	0	0	1	0	0	0	0
R_8	0	0	0	0	0	1	0	0	0
R_9	1	1	0	0	1	0	0	0	0

Figure 3.6 : La matrice d'adjacence des règles.

Afin de savoir quelle est la règle obtenue par l'autre, nous devons appliquer ce qui est connu dans la théorie des graphes, la mise en ordre d'un graphe orienté, il faut que le graphe obtenu soit un graphe orienté acyclique ou *DAG* (Directed Acyclic Graph), c'est-à-dire un graphe qui ne possède pas de circuit.

Il y a deux étapes pour détecter l'existence des circuits, qui sont très utiles pour simplifier les méthodes de résolution dans les graphes.

Premièrement, nous éliminons les boucles, c'est-à-dire un sommet possédant un arc rebouclé sur lui-même est un cas particulier, qui est des circuits mais dans le graphe il n'existe pas de boucles car le graphe obtenu est un graphe orienté simple (Digraphe).

Deuxièmement, nous éliminons progressivement tous les sommets qui contiennent les arcs entrants ou les arcs sortants car, si un sommet du graphe n'a pas d'arc entrant ou sortant alors celui-ci ne peut pas faire partie d'un circuit dans le graphe. Nous supprimons donc le sommet du graphe ainsi que tous ses arcs adjacents, principe de l'algorithme de MORIMANT [46]. Si par cette procédure, nous effaçons le graphe alors le graphe ne possède pas de circuits.

À partir de la matrice d'adjacence booléenne nous éliminons tous les sommets qui contiennent des lignes et des colonnes nulles, Il reste des sommets qui contiennent des lignes et colonnes non nulles et qui constituent un circuit. En appliquant cette procédure à la matrice précédente nous remarquons que le sommet R_6 reçoit des lignes nulles ($d^+(R_6)=0$) et les sommets R_4 et R_7 reçoivent des colonnes nulles donc en supprimant ces sommets de la matrice d'adjacence nous trouvons un graphe orienté réduit.

	R_1	R_2	R_3	R_5	R_8	R_9
R_1	0	0	1	0	0	0
R_2	0	0	0	0	0	0
R_3	0	0	0	0	0	0
R_5	1	0	0	0	0	0
R_8	0	0	0	0	0	0
R_9	1	1	0	1	0	0

Figure 3.7 : La matrice d'adjacence réduite des règles.

Nous avons appliqué la même procédure sur la matrice réduite, où nous avons pris le sommet R_8 qui reçoit une ligne nulle et le sommet R_9 qui reçoit une colonne nulle donc nous avons supprimé les sommets de la matrice d'adjacence réduite précédente, nous avons obtenu une nouvelle matrice réduite.

	R_1	R_2	R_3	R_5
R_1	0	0	1	0
R_2	0	0	0	0
R_3	0	0	0	0
R_5	1	0	0	0

Les sommets R_2 et R_3 reçoivent des lignes nulles donc nous obtenons :

	R_1	R_5
R_1	0	0
R_5	1	0

Le sommet R_1 reçoit une ligne nulle et R_5 une colonne nulle donc nous supprimons les deux sommets de la matrice d'adjacence nous obtenons un graphe vide. Cela veut dire que le graphe orienté ne contient pas de circuits alors c'est un graphe sans circuit ou DAG. Donc nous pouvons faire la mise en ordre de ce graphe.

A partir de la matrice d'adjacence nous pouvons faire la mise en ordre d'un graphe orienté sans circuit, nous supprimons à chaque itération, tous les sommets qui correspondent à une colonne nulle, donc qui n'ont pas de précédents.

En appliquant cette procédure on obtient :

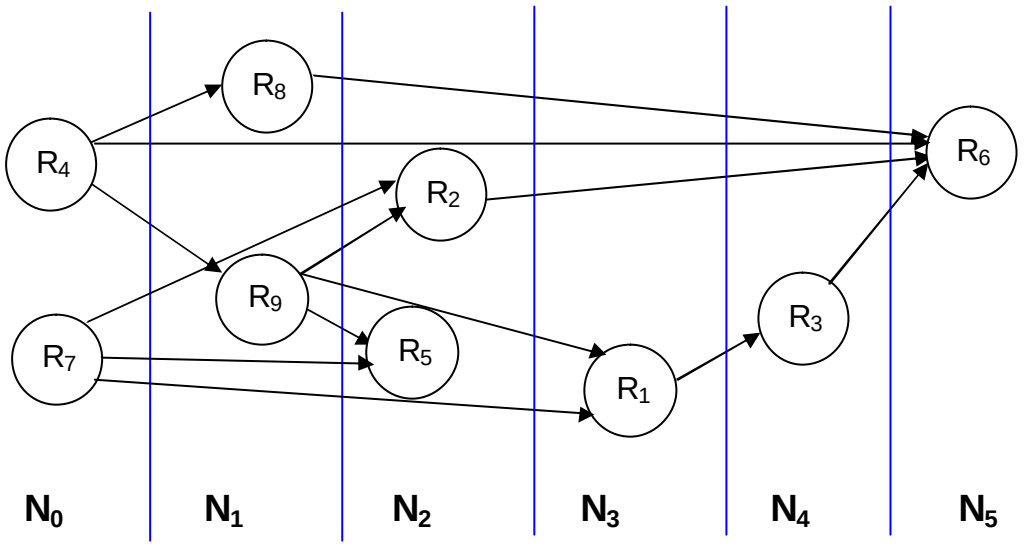


Figure 3.8 : La mise en ordre de la base de règles.

Donc nous avons exécuté la base de règles précédente par cette ordre dans les moteurs de chaînage avant d'un système expert est ne peut pas faire un cycle d'exécution, c'est-à-dire quelque soit les fait dans la base de faits d'un système expert, le moteur d'inférence exécute les règles dans l'ordre une seule fois donc en ne faisant pas la deuxième exécution, si une règle n'est pas exécutée on a jamais exécuté.

1ière exécuté ; R_4: **SI** Benoît **ALORS** Xena.
2 ième exécuté ; R_7: **SI** Cloé **ALORS** Djamel.
3 ième exécuté ; R_8: **SI** Xena **ET** Cloé **ALORS** Amandine.
4 ième exécuté ; R_9: **SI** Xena **ET** Benoît **ALORS** mousse
5 ième exécuté ; R_2: **SI** Gaelle **ET** Djamel **ALORS** Amandine.
6 ième exécuté ; R_5: **SI** Djamel **ALORS** Emma.
7 ième exécuté ; R_1: **SI** Benoît **ET** Djamel **ET** Emma **ALORS** Félix.
8 ième exécuté ; R_3: **SI** Cloé **ET** Félix **ALORS** Amandine.
9 ième exécuté ; R_6: **SI** Xena **ET** Amandine **ALORS** Habiba.

Figure 3.9 : L'ordonnancement de la base de règles.

Cette procédure reste vraie si le graphe de relation ne contient pas des circuits. Le problème posé est : s'il y a des circuits comment faire pour cette modélisation. Alors nous prenons un deuxième exemple de base de règles modélisé par un graphe de relation des règles qui contient des circuits.

Exemple 3.5.

Soit la base de règles suivante :

R_1 : **SI** A **ET** B **ALORS** C.
R_2 : **SI** C **ET** D **ALORS** F.
R_3 : **SI** F **ET** B **ALORS** E.
R_4 : **SI** F **ET** A **ALORS** G.
R_5 : **SI** G **ET** F **ALORS** B.

Figure 3.10 : Règles de productions de la base de règles.

Les sommets représentent les règles Ri et les arcs représentent les relations entre les règles, par exemple la conclusion C de la règle R_1 appartient aux conditions de la règle R_2. Donc nous ajoutons un arc de R_1 vers R_2.

Nous pouvons présenter une base de règles comme un graphe orienté. La modélisation de cette base de règles montre la figure suivante :

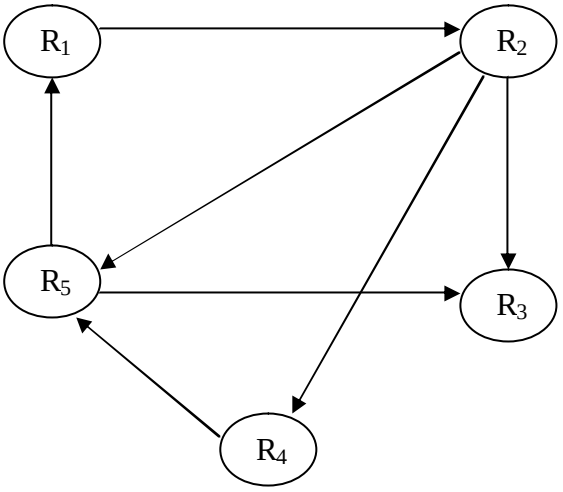


Figure 3.11 : Graphe de Relation des règles avec circuits.

Le graphe de relation précédent est représenté par sa matrice d'adjacence:

	R_1	R_2	R_3	R_4	R_5
R_1	0	1	0	0	0
R_2	0	0	1	1	1
R_3	0	0	0	0	0
R_4	0	0	0	0	1
R_5	1	0	1	0	0

Figure 3.12 : La matrice d'adjacence des règles.

On utilise la matrice d'adjacence précédente pour calculer les demi- degrés ($d^+(x)$, $d^-(x)$) de sommet d'un digraphe.

Nous calculons dans la matrice d'adjacence les demi-degrés positifs $d^+(x)$ et négatifs $d^-(x)$, la somme des lignes dans la matrice représente les demi-degrés positifs et la somme des colonnes de la matrice représente les demi-degrés négatifs. En théories des graphes on dit les demi-degrés des arcs sortants et entrants. On calcule les demi-degrés dans le problème de choix de sommets.

	R_1	R_2	R_3	R_4	R_5	$d^+(x)$
R_1	0	1	0	0	0	1
R_2	0	0	1	1	1	3
R_3	0	0	0	0	0	0
R_4	0	0	0	0	1	1
R_5	1	0	1	0	0	2
$d^+(x)$	1	1	2	1	2	/

Figure 3.13 : La matrice d'adjacence des règles et demi-degrés.

Nous allons présenter dans cette partie la méthode d'initialisation que nous avons implantée. Nous donnons dans une première section une définition générale de la classe d'heuristique que nous avons décidé d'utiliser, ainsi que les motivations de ce choix. Dans une deuxième partie, nous donnerons l'algorithme complet permettant de trouver le MFVS d'un graphe. Cet algorithme sera composé d'une phase de prétraitement de graphe suivi de la méthode proposée. Celle-ci doit bien sûr être adaptée à notre problématique. Nous développerons tous les détails.

3.3. La méthode générale de la mise en ordre :

3.3.1 Définition :

Une méthode approchée ou heuristique (heuristic, approximation method), pour un problème d'optimisation combinatoire, est un algorithme qui a pour but de trouver une solution réalisable, tenant compte de critères d'optimisation et des contraintes, mais sans garantie d'optimalité. On oppose les méthodes approchées aux méthodes exactes, qui trouvent toujours l'optimum, mais leur inconvénient est le temps de résolution (CPU). Un algorithme qui donne rapidement une bonne solution, bien que pas forcément optimal s'appelle une heuristique. Les heuristiques forment une classe de méthodes réputées pour fournir des solutions de bonne qualité en un temps raisonnable à des problèmes combinatoires difficiles. L'heuristique posée est une méthode multi-départs (c'est-à-dire utilisant plusieurs itérations) pour la recherche d'une solution approchée de problèmes difficiles d'optimisation combinatoire.

La méthode proposée démarre par la question suivante : est-ce que le graphe orienté modélisé contient des circuits où non ? Si non, on fait la mise en ordre d'un graphe orienté sans circuit. Si oui on cherche le nombre minimal de sommets pour désactiver tous les circuits de digraphes. La première étape est de chercher tous les circuits de digraphes et de construire une matrice cyclomatique (sommet/circuits) où on prend les sommets de choix d'un problème de feedback.

3.3.2. La méthode appliquée au problème du MFVS

Le problème consiste à trouver le nombre minimum de sommets dans un graphe orienté permettant d'en couper tous les circuits. Une solution à ce problème est donc un ensemble de sommets, la solution optimale correspond au plus petit ensemble possible. Dans cette section, nous allons définir chaque partie de la méthode implantée.

3.3.2.1 Motivations du choix

Notre choix s'est porté sur une heuristique pour deux raisons principales : D'abord, cette classe d'heuristiques est réputée pour être efficace sur des problèmes de grande taille. Or, nous avons pour objectif de traiter la base de règles et donc comportant souvent un très grand nombre de règles. Enfin, comme nous le verrons dans la suite de ce mémoire, cette heuristique a la particularité d'être paramétrable. En effet, l'objectif d'utiliser l'heuristique. Selon la taille du problème qu'il considère, il peut par exemple privilégier la qualité de la solution au détriment du temps CPU nécessaire ou inversement obtenir une solution seulement "acceptable" en un temps CPU le plus court possible.

3.3.2.2 Prétraitement des graphes

Avant de développer la méthode permettant de trouver le MFVS d'un graphe orienté simple ou digraphe, plusieurs phases de prétraitement de graphe peuvent être réalisées. En effet, il est en général possible de simplifier un graphe c'est-à-dire supprimer certains sommets et certains arcs sans pour autant modifier son MFVS. Ces méthodes sont de faible complexité donc rapides. L'algorithme appliqué ensuite sera d'autant plus performant que les simplifications auront été nombreuses.

a) Réductions

La première méthode de prétraitement consiste à appliquer la règle élémentaire de réduction de graphe. Le graphe initial est réduit et donne un réseau par le principe de détecter les circuits et on obtient un graphe réduit. Cette première phase permet d'éliminer un certain nombre de sommets et d'arcs, de plus le nombre minimum de feedback du graphe initial étant même de graphe réduit.

? L'existence des circuits dans un graphe orienté

Il y a une méthode pour détecter l'existence des circuits qui sont très utiles pour simplifier les méthodes de résolution dans les graphes. Donc on élimine progressivement tous les sommets qui continent les arcs entrants ou sortants car, si un sommet du graphe n'a pas d'arc entrant ou sortant alors celui-ci ne peut pas faire partie d'un circuit dans le graphe. On supprime donc le sommet du graphe ainsi que tous ses arcs adjacents, principe de l'algorithme de MORIMANT [46]. Si, par cette procédure, on efface le graphe alors à partir de la matrice d'adjacence booléenne on élimine tout les sommets qui contiennent des lignes et des colonnes nulle, Il reste des sommets qui contiennent des lignes et colonnes non nulles et qui constituent un circuit. C'est en appliquant cette procédure à la matrice précédente qu'on remarque que le sommet R3 reçoit des lignes nulles ou $d^+(R_3)=0$ donc en supprimant ce sommet de la matrice d'adjacence on trouve un graphe réduit. En appliquant le principe de détecter les circuits on obtient un graphe réduit, donc la matrice d'adjacence correspondante au graphe réduit est :

? La matrice réduite :

	R_1	R_2	R_4	R_5
R_1	0	1	0	0
R_2	0	0	1	1
R_4	0	0	0	1
R_5	1	0	0	0

Figure 3.14 : La matrice d'adjacence réduit.

Il reste la matrice d'adjacence réduite qui présente des sommets et qui contient des lignes et des colonnes non nulles et qui constituent un circuit. Le problème qui reste est de trouver tous les circuits et comment les désactiver.

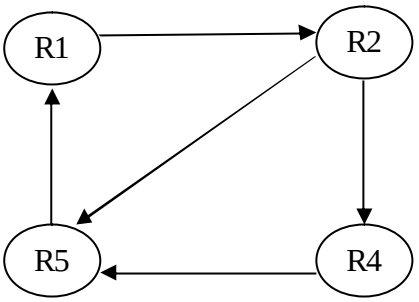


Figure 3.15 : Graphe de Relation réduit.

? Énumération des circuits dans un graphe orienté

Pour faire la mise en ordre d'un digraphe, il faut que le graphe obtenu soit un graphe orienté acyclique ou *DAG* (Directed Acyclic Graph), c'est-à-dire un graphe qui ne possède pas de circuit. Dans le cadre de ce travail, le graphe orienté contient des circuits. La deuxième phase de prétraitement consiste à une énumération des circuits. Chercher les circuits d'un graphe G revient en fait à chercher les circuits de graphe réduit de G. Dans l'étape suivante on obtient un graphe orienté réduit, on applique la stratégie d'énumérer tous les circuits de ce graphe, on utilise la matrice d'adjacence, le principe de l'algorithme d'énumération des chemins et on calcule tous les circuits élémentaires du graphe réduit. La méthode suivante qui semble relativement simple : repose sur la méthode définie sur le chapitre 2, qui présente l'algorithme d'énumération de circuit de Johnson pour des graphes orientés et présenter une exécution efficace et à rendement élevé de mémoire dans le langage de programmation de JAVA. Nous allons utiliser cette méthode sur le graphe orienté réduit précédent (figure. 3.15). L'ensemble des circuits est : $\{\{R_1R_2R_5\}, \{R_1R_2R_4R_5\}\}$. Par la méthode d'arbre de recherche récursive pour les circuits.

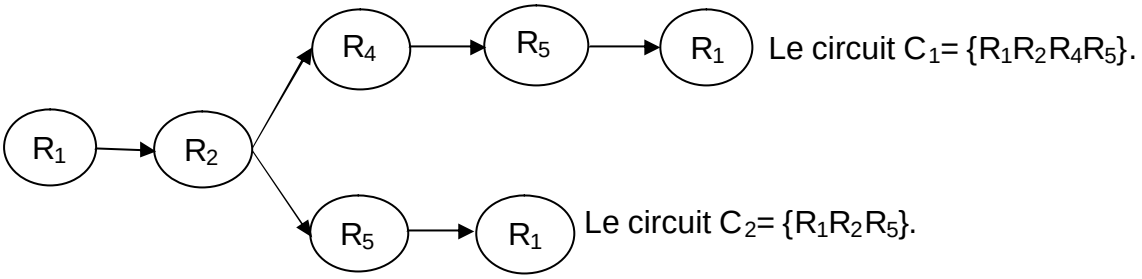


Figure 3.16 : Un arbre de recherche récursive pour les circuits des règles.

3.3.2.3 Construction de la matrice cyclomatique

La matrice cyclomatique contient des circuits et des sommets, les colonnes de la matrice contiennent les circuits « Cj» et les lignes contiennent les sommets Ri alors C : *p x n*. La matrice cyclomatique est une matrice booléenne, l'existence d'un sommet dans le circuit se traduit par la présence d'un 1 à l'intersection de la ligne Ri et de la colonne Cj ; c'est-à-dire le sommet Ri dans le circuit Cj ; l'absence d'un sommet dans le circuit par la présence d'un 0, i.e. il n'existe pas de sommet Ri dans le circuit Cj.

	C1	C2
R_1	1	1
R_2	1	1
R_3	0	0
R_4	0	1
R_5	1	1

Figure 3.17 : La matrice cyclomatique.

On utilise la matrice cyclomatique précédente pour calculer la somme des lignes et prendre le maximum. La somme des lignes dans la matrice cyclomatique représente le nombre de circuits qui constituent ce sommet.

On prend le maximum pour utiliser dans le problème de choix de sommets d'un *feedback*.

	C_1	C_2	? R_{ij}	MAX(Ri)
R_1	1	1	2	2
R_2	1	1	2	2
R_3	0	0	0	
R_4	1	0	1	
R_5	1	1	2	2

Figure 3.18 : La matrice cyclomatique et le maximum.

Si on prend le sommet qui reçoit le maximum puis on les met dans l'ensemble de feedback puis, on supprime les circuits qui constituent ce sommet et on applique la même procédure pour la matrice réduite. Le problème reste sur le choix de sommet car s'il existe des sommets qui ont le même maximum, on passe à la deuxième étape du choix des sommets triés par les demi-degrés d'un sommet.

Nous effectuions une procédure de choix comme suit :

Si $d^+(R_1) < d^+(R_2) < \ldots < d^+(R_n)$

On prend le sommet qui contienne le demi-degré minimum.

Sinon $d^-(R_n) < \ldots < d^-(R_2) < d^-(R_1)$

On prend le sommet qui contienne demi-degré maximum.

Si les deux cas n'existent pas, on prend alors un sommet au hasard.

.

3.3.2.4 Le choix de sommets :

Dans la matrice cyclomatique on prend le maximum de la somme des lignes, le maximum dans la matrice précédente est égale à 2, les sommets correspondants ce maximum sont les sommets R_1, R_2 et R_5.

On a $d^+(R_1)$=1, $d^+(R_2)$=3 et $d^+(R_5)$=2, on choisit le sommet R_1 car leurs degrés est plus petit que celui de R_2 et R_5.

3.3.2.5 Comment désactiver le circuit

On prend le sommet de choix par exemple « R » contient des arcs entrants et des arcs sortants, on devisé le sommet « R » en deux sommets; le premier sommet contient seulement des arcs sortants et l'autre sommet contient des arcs entrants de sommet « R» pour éliminer les circuits (Problème de Feedback).

D'prés l'exemple précèdent le sommet de choix R_1 contient des arcs entrants et des arcs sortants, on devisé le sommet R_1 en deux sommets :

- R_{11}**:** contient les arcs sortants de sommet « R_1 » (la colonne de « R_{11} » reçoit des zéros et la ligne de « R_{11} » reçoit la ligne de « R_1 »).

- R_{12} **:** contient les arcs entrants de sommet « R_1 » (La colonne de « R_{12}» reçoit la colonne de « R_1 » et la ligne de « R_{12} » reçoit des zéros).

Dans notre problème ne peut pas supprimé et dupliqué les sommets car les sommets représentent les règles. Les sommets qui contiennent les arcs sortants en mettent le premier niveau de la mise en ordre. Dans l'exemple, le sommet R_{11} ne donne aucun effet dans notre problème alors en supprime. Nous effectuons des zéros sur la ligne de sommet de choix R_1 en prend une matrice d'adjacence modifié sans circuits.

3.3.2.6 La matrice d'adjacence modifié sans circuits

A partir de la matrice d'adjacence on fait toutes les procédures précédentes et on prend une nouvelle matrice d'adjacence modifié.

	R_1	R_2	R_3	R_4	R_5
R_1	**0**	**0**	**0**	**0**	**0**
R_2	0	0	1	1	1
R_3	0	0	0	0	0
R_4	0	0	0	0	1
R_5	1	0	1	0	0

Figure 3.19 : La matrice d'adjacence modifié sans circuits.

A partir de la matrice d'adjacence modifiée on peut faire la mise en ordre d'un graphe orienté sans circuit, on supprime à chaque itération, tous les sommets qui correspondent à une colonne nulle, donc qui n'ont pas de précédents.

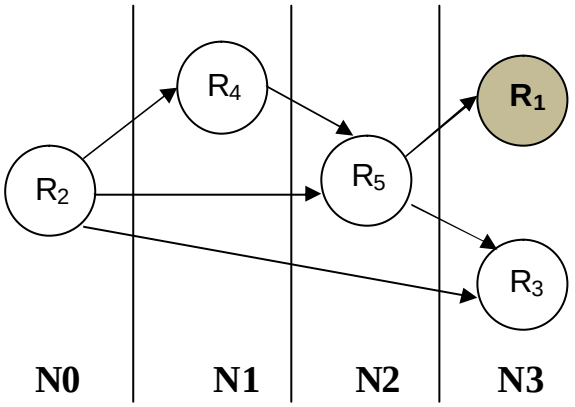


Figure 3.20 : La mise en ordre d'un graphe de relation sans circuit.

Cela va nous faire aboutir à une base de règles ordonnée sur laquelle nous allons faire appliquer le moteur d'inférence chaînage avant d'un système à base de règles. Nous pouvons aussi faire un cycle d'exécution en deux fois, c'est-à-dire nous allons parcourir les règles dans cet ordre. Dans le cas où il y'ait des règles non exécutées nous allons faire un deuxième parcours. Donc nous avons applique le moteur d'inférence chaînage avant sur la base de règles une seule fois dans le cas ou le graphe de relation sans circuits, ou deux fois s'il continent des circuits (problème de feedback).

1ière exécuté ; R_2 :	**SI**	C	**ET**	D	**ALORS**	F.
2 ième exécuté ; R_4 :	**SI**	F	**ET**	A	**ALORS**	G.
3 ième exécuté ; R_5 :	**SI**	G	**ET**	F	**ALORS**	B.
4ième exécuté ; R_1 :	**SI**	A	**ET**	B	**ALORS**	C.
5 ième exécuté ; R_3 :	**SI**	F	**ET**	B	**ALORS**	E.

Figure 3.21 : L'ordonnancement de la base de règles.

Résumons toutes les étapes précédentes, ou donnons l'algorithme suivant qui détermine le nombre minimal de sommet à supprimé dans un graphe :

3.3.2.7 Algorithme:

1. Trouver tous les circuits dans le graphe
2. construire la matrice cyclomatique ***C*** (les colonnes contiennent les circuits « *p* » et les lignes contiennent les sommets x_i alors *C* : *p x n.*
3. calculer la somme des lignes et prendre le maximum

   ? s'il y a plus de deux sommets on fait le choix :

   a) premièrement par les plus petits degrés sortant de sommets $d^+(x_i)$.
   b) deuxièmement par les plus grands degrés entrant de sommets $d^-(x_i)$
   c) si les deux choix sont égaux, on prend un sommet au hasard.

   ? à chaque étape on élimine les circuits qui contiennent le sommet de choix.

3.3.2.8. Synoptique de l'algorithme complet :

Le synoptique complet de l'heuristique mis en place est donné sur la figure 3.22. Nous savons que la complexité de l'algorithme de recherche des énumérations des circuits est *O ((N+m) x(c+1))* et que la complexité de règle de réduction appliquée successivement est *O (n+m)* [46]. Donc pour tous les graphes dont le nombre de sommets |X| est un réseau, il est plus judicieux d'appliquer d'abord la procédure d'énumération des circuits, puis celle de réduction sur le graphe. Une fois le graphe initial réduit (phase de prétraitement), l'algorithme proposé est appliqué en vue de résoudre le problème particulier du MFVS. Comme défini au début du chapitre, la méthode est constituée de deux phases principales : D'abord la phase de prétraitement. Pour cette procédure il est nécessaire de définir dans un premier temps la règle de réduction utilisée pour réduire le graphe initial. Ensuite vient la phase de recherche du nombre minimale de sommets permettant d'en couper tous les circuits, appliquer après la phase de réduction qui améliore souvent la solution construite.

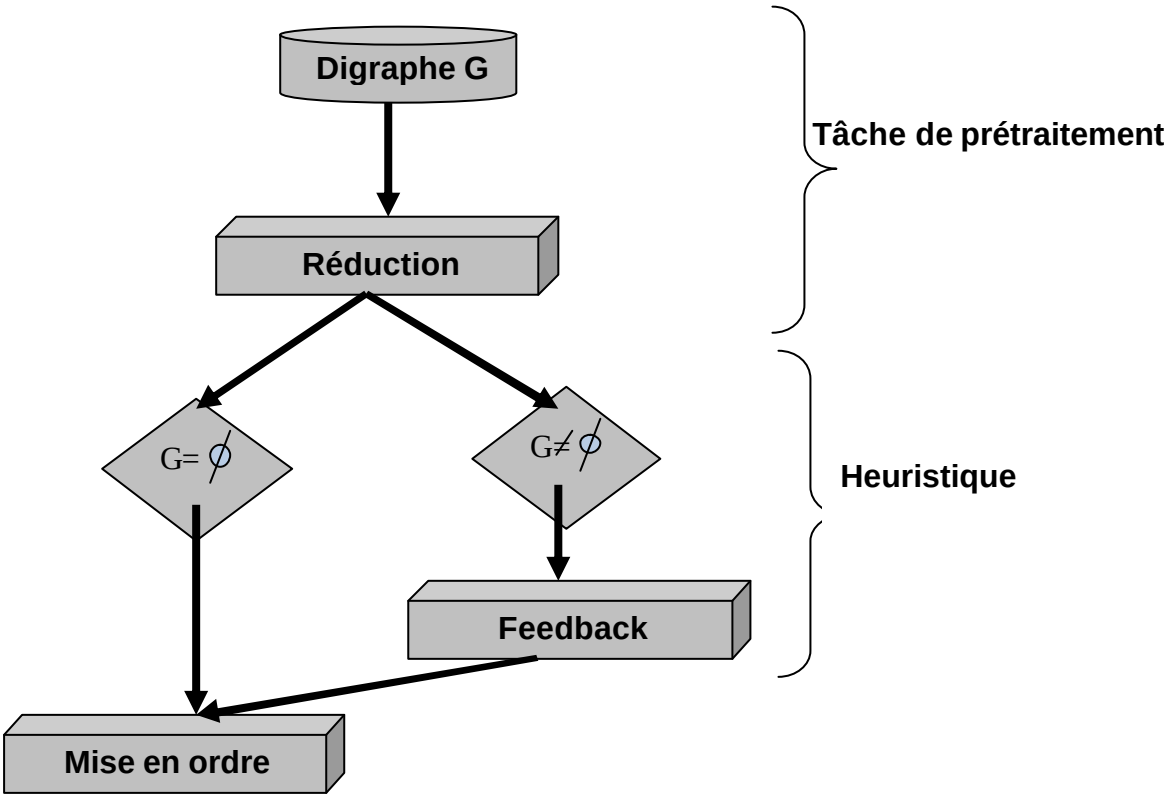


Figure 3.22 : Synoptique de l'algorithme global.

3.4. Conclusion

La méthode que nous avons mise en place est constituée de deux parties distinctes : La première concerne la phase de prétraitement, que permet de supprimer certains sommets du graphe et on prend un graphe réduit puis on réduit la règle de réduction en préservant le MFVS. Cette première partie est une méthode exacte. Nous verrons par la suite que pour certains graphes, elle suffit à trouver le MFVS. C'est le cas lorsque le graphe peut être totalement réduit par les règles de réduction, le MFVS trouvé sera alors optimal. La deuxième partie est l'heuristique : on propose une heuristique basée sur une énumération des circuits du graphe borné. Cette dernière contient plusieurs paramètres car le choix de sommet est exigé puisque nous utilisons le MFVS pour choisie le sommet de règles car le sommet représente des règles qui ne peuvent pas être supprimées.

CHAPITRE 4

IMPLEMENTATION, TESTS ET RESULTATS

Dans ce chapitre, nous allons présenter les différents résultats des simulations que nous avons réalisées à partir d'une implémentation que nous avons réalisée. Afin de valider nos méthodes de résolution, nous commençons par présenter l'application de notre approche pour la résolution du problème posé par le côté « raisonnement » de l'agent "To". A cet effet nous nous sommes intéressé à la base de règles faisant partie de ce système de raisonnement, qui est un point dans la conception de l'agent "To", sur laquelle nous avons appliqué notre approche d'ordonnancement.

4.1 Outil informatique utilisé

Pour développer les algorithmes utilisés dans notre approche d'optimisation, nous avons utilisé le langage de programmation orienté objet JAVA. Ce langage est conçu avec l'approche orientée objet, de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

Notre choix est basé sur les caractéristiques de ce langage comme, entre autres, la portabilité et l'indépendance vis-à-vis des plateformes. De plus, Java offre une bibliothèque très riche de classes comme celles pour l'utilisation du réseau ou de l'interface utilisateur graphique. Le langage Java offre aussi le support pour l'utilisation des processus légers (*threads*).

4.2 Implémentation du système expert à base de graphes :

Le diagramme de système expert, illustre par la figure suivante, donne un aperçu sur les classes implémentées dans notre système.

Nous avons implémenté un système d'inférence à base de règles au moyen de multigraphe pondéré qui utilise le parcours de graphes par le principe de calculons les chemins simples (même poids d'arcs) et utilisons le chaînage en avant comme une méthode d'inférence.

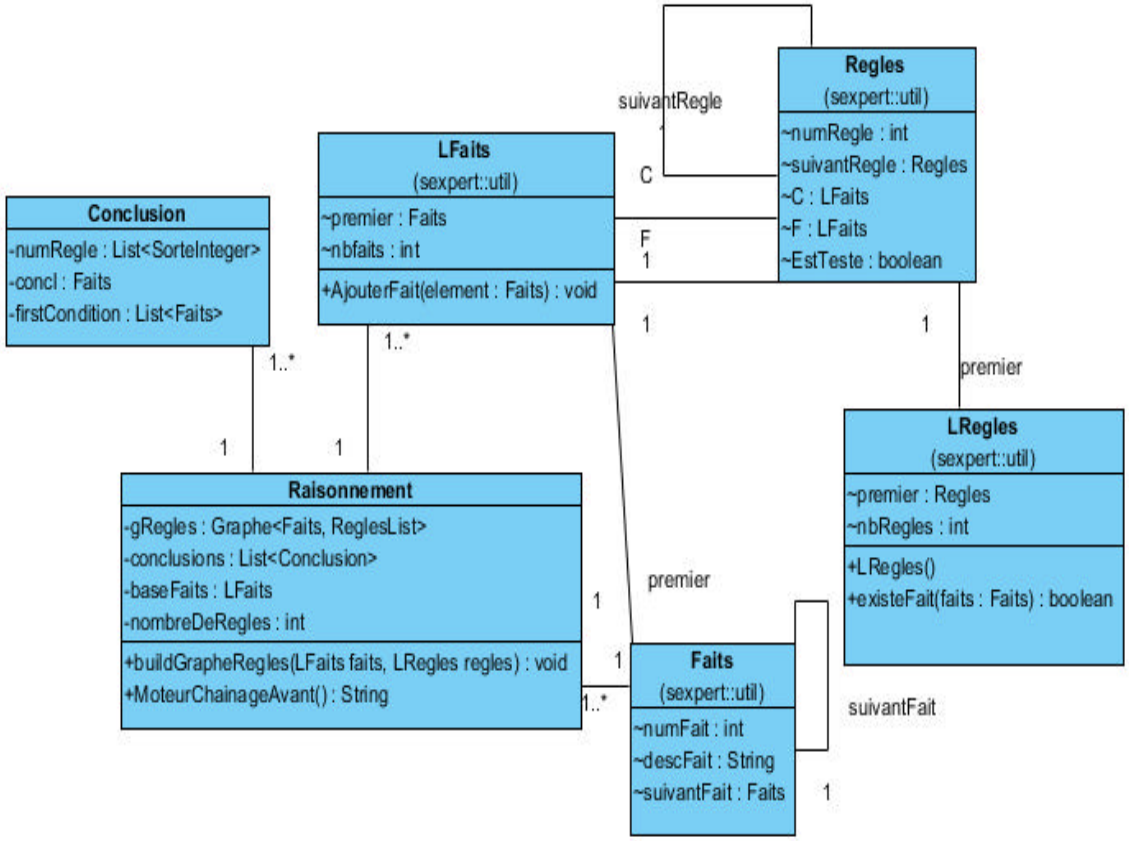


Figure 4.1. Diagramme de classe du système expert à base de graphes.

4.3. Implémentation du système expert avec la mise en ordre

4.3.1. Système expert :

Le diagramme de classe du système expert, illustré par la Figure 4.2, donne un aperçu sur les classes implémentées dans notre système.

Nous avons implémenté un système d'inférence à base de règles, c'est un système expert, qui utilise le chaînage en avant comme une méthode d'inférence.

Le système expert comporte la base des règles (LRègles) qui est implémenté par une liste des règles (Règles) et une base de faits (LFaits) implémentée aussi par une liste de faits (Fait).

Le diagramme de classe de ce système expert soit :

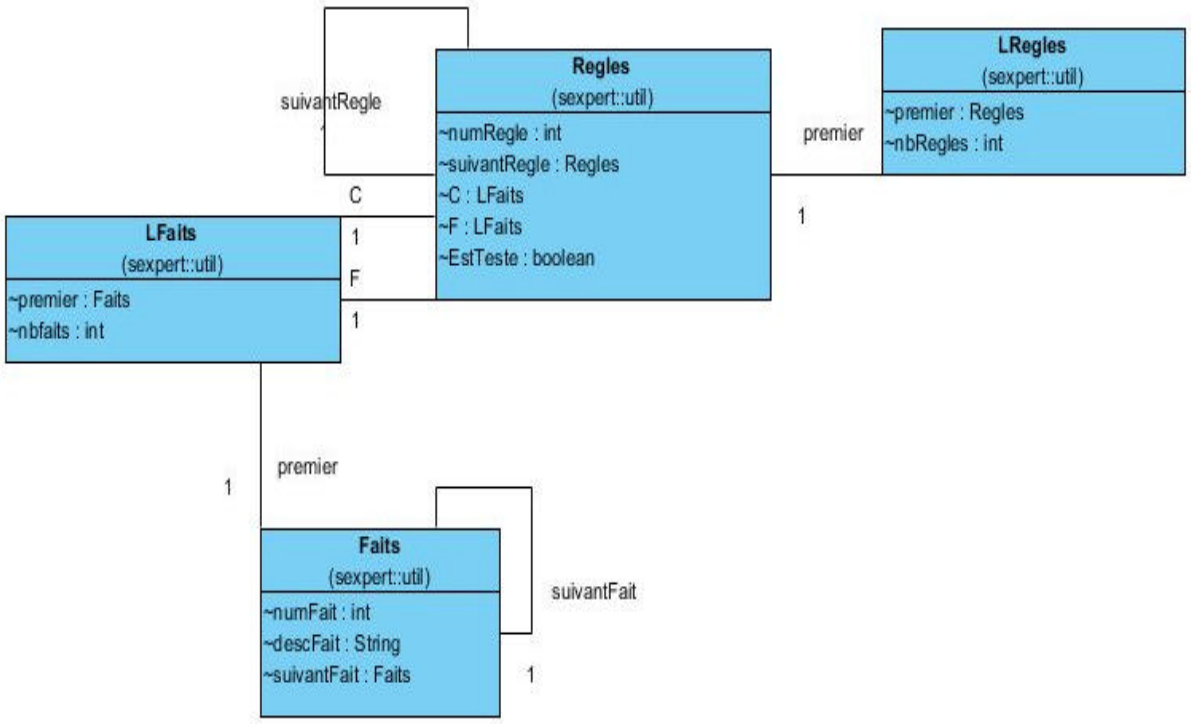


Figure 4.2. Diagramme de classes du système expert.

4.3.2. La mise en ordre

L'algorithme général de la mise en ordre que nous avons implémenté est représenté par le diagramme de la figure :

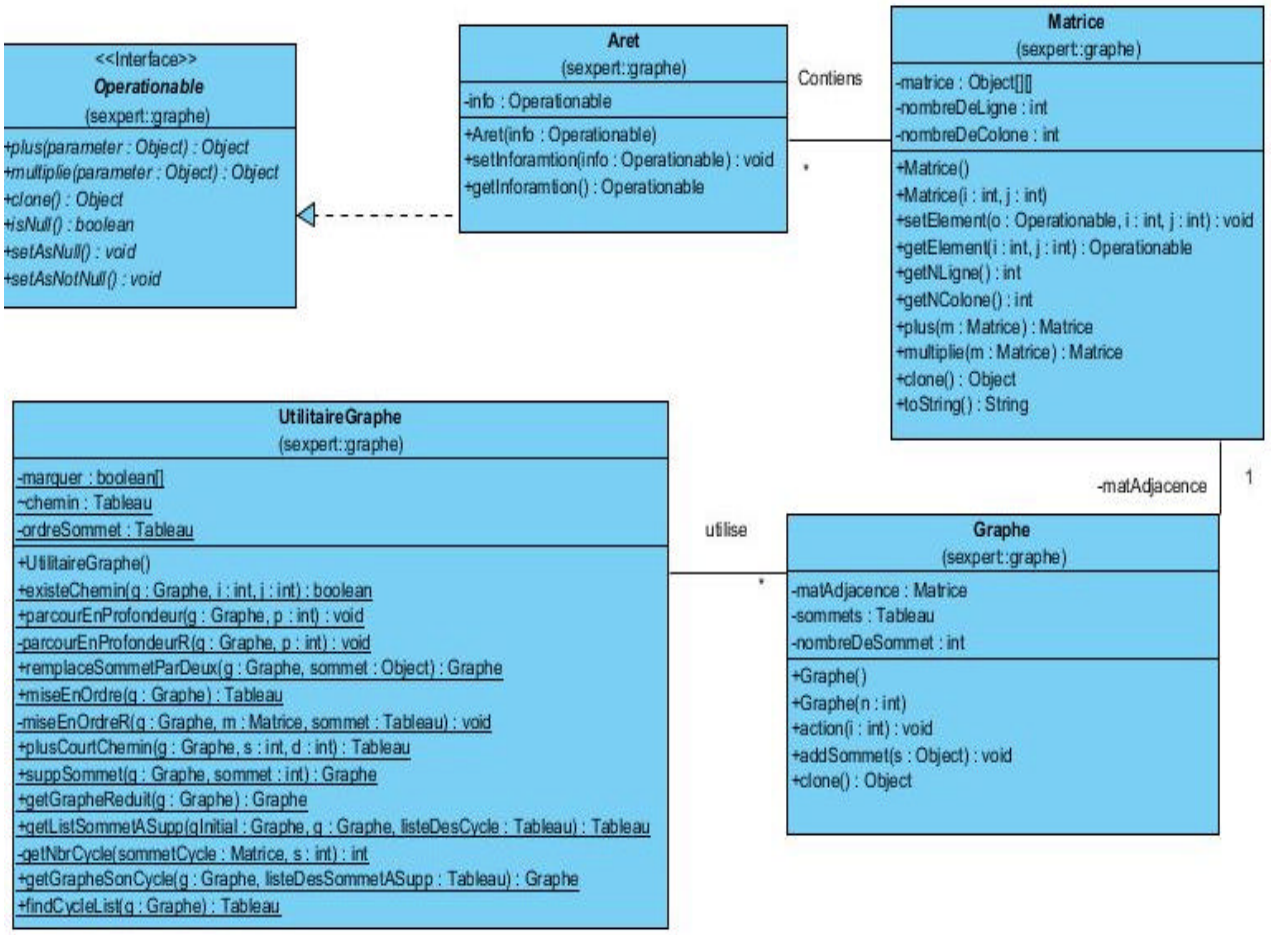


Figure 4.3. Diagramme de la mise en ordre.

4.3.3. Système expert avec la mise en ordre :

Le diagramme de système expert avec la mise en ordre, illustre par la figure suivante, donne un aperçu sur les classes implémentées dans notre système. C'est un élément important dans la phase de conception, il permet une compréhension générale de la façon dont le système fonctionne.

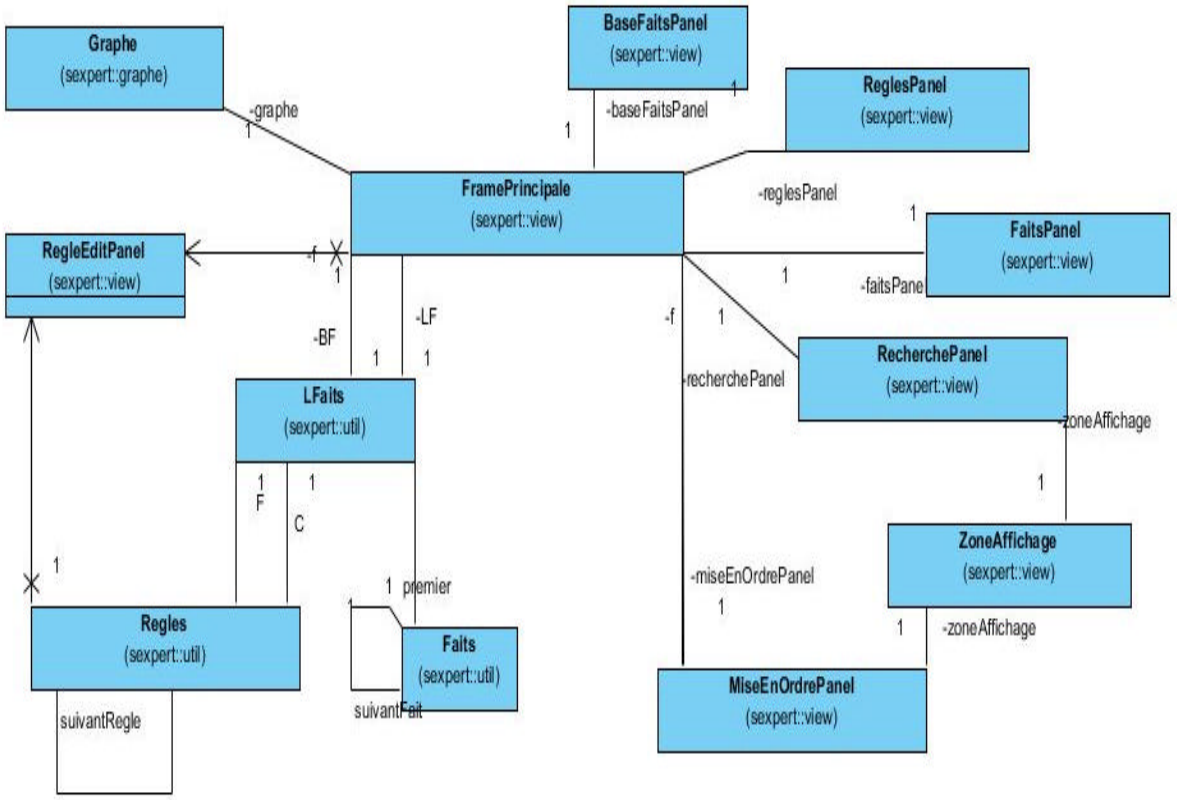


Figure 4.4. Diagramme de classe pour Système expert avec la mise en ordre.

4.4. Validation des algorithmes d'optimisation :

Suite à l'étude des modélisations existantes, nous avons choisi l'utilisation d'une modélisation graphique (graphe orienté "Graphe de Relation"). Cette dernière repose sur l'utilisation des algorithmes existants dans les graphes orientés. Nous avons effectué des tests afin de comparer les résultats entre la méthode que nous proposons et ceux découlant d'une base de règles quelconque.

Nous avons effectué plusieurs tests d'optimisation en changeant le nombre de règles à optimiser.

On compare le temps d'exécution entre les règles ordonnées et non ordonnées de chaque base de règles du système expert.

Suite à l'étude des modélisations existantes, nous avons choisi l'utilisation d'une modélisation à base de graphes orientés que nous avons appelé *graphes de relations*. Ce choix nous a été dicté par la considération de vouloir bénéficier des algorithmes existants dans le domaine des graphes orientés.

Pour automatiser le processus de construction de notre modèle ainsi que pour son utilisation dans les tests, nous avons réalisé une implémentation en orienté objets en utilisant le langage java.

L'architecture de notre logiciel est axée sur trois principales composantes :

1. La composante *système à base de règles* comportant les classes en vue de la réalisation de l'inférence en chaînage avant à partir d'une base de faits et d'une base de règles.

2. Le composant *graphe de relation* comportant les classes produisant le comportement pour la réalisation de la mise en ordre des règles de la base de règles, à partir d'un modèle de graphe orienté.

3. d'une *interface graphique*, offrant des possibilités de manipulation et d'affichage expressives et conviviales.

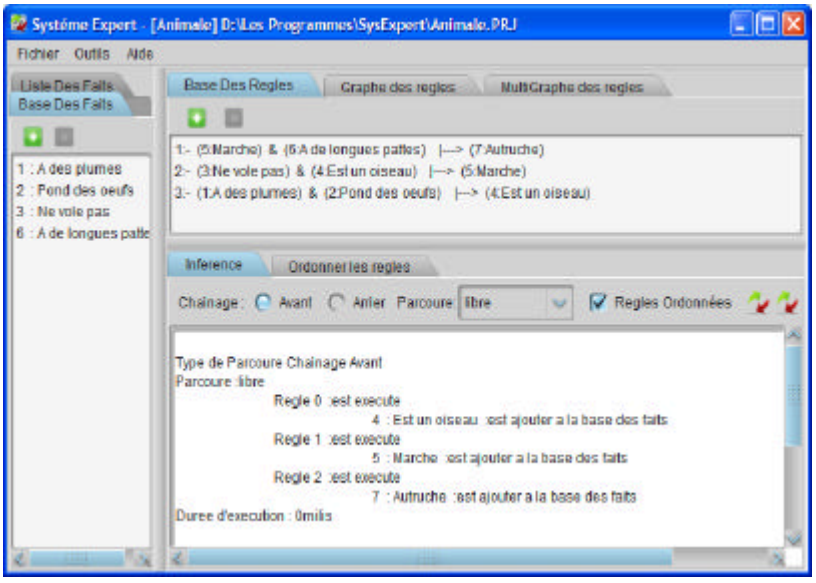


Figure 4.5. Logiciel de système expert.

On distingue deux cas de figures de comparaisons : la première concerne la recherche du temps d'exécution des règles de chaque base de règles et la deuxième concerne la détermination du nombre de parcours d'une base de règles.
Pour simplifier nous allons tester les deux cas suivants :
cas1 : sur le temps d'exécution des règles de chaque base de règles.
cas2 : sur le nombre de parcours.

? **Cas 1**

Après la phase de réalisation du modèle, nous validons pratiquement ce modèle, en effectuant à chaque fois une batterie de tests, afin de comparer entre le modèle proposé et la méthode utilisant des règles non-ordonnées.

Nous avons effectué plusieurs tests d'optimisation en jouant sur le changement du nombre de règles à optimiser en vue de comparer le temps d'exécution de l'inférence. Nous avons utilisé, à cet effet, une base de règles ordonnées au moyen de notre modèle et une autre non ordonnée.

Le tableau de la figure 4.6 représente une copie d'écran du fonctionnement du logiciel réalisé, affichant les résultats produits par des moyennes calculées sur 4 tests. Nous avons testé notre logiciel sur plusieurs bases de règles avec un nombre variable de règles de production.

La base de règles	**B_1**	**B_2**	**B_3**	**B_4**
Nombres des règles	50	100	150	200
Temps d'excusions des Règles Ordonnée (m/secs)	31	62	94	125
Temps d'excusions des Règles Non Ordonnée (m/secs)	141	156	172	203

Figure 4.6. Tableau du temps d'exécution des bases de règles.

D'après le tableau donné en figure 4.6, Nous voyons que la solution optimisée est obtenue par le modèle des règles ordonné, permettant d'atteindre ainsi un temps d'exécution minimum.

Afin de tester l'efficacité des résultats obtenus par nos approches, nous avons effectué des comparaisons avec d'autres systèmes experts.

Pour effectuer une comparaison, on doit les appliquer au même problème et avec des conditions équivalentes. Ce qui nous a permis de faire une première comparaison des deux méthodes.

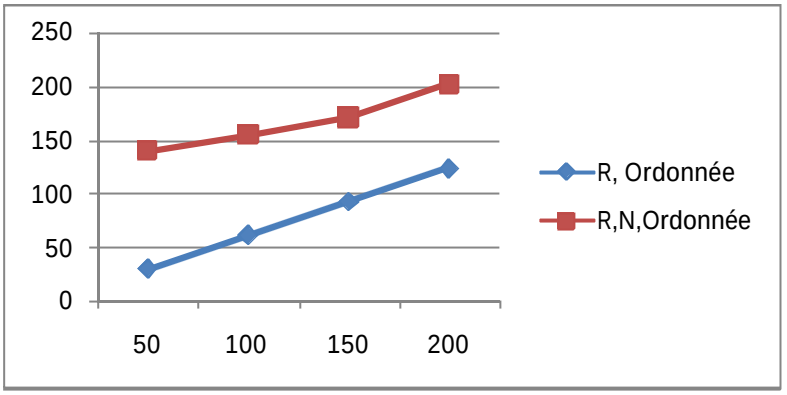


Figure 4.7. Comparaison entre les règles ordonnée et non ordonnée.

En comparant les temps d'exécution entre les deux méthodes, nous avons remarqué qu'il y a une différence, alors que les nombres de règles sont très différents entre les deux bases de règles.

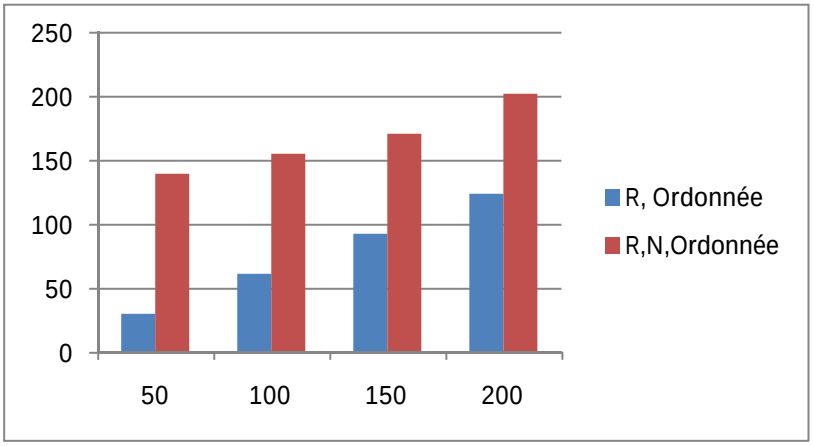


Figure 4.8. Histogramme entre les règles ordonnée et non ordonnée.

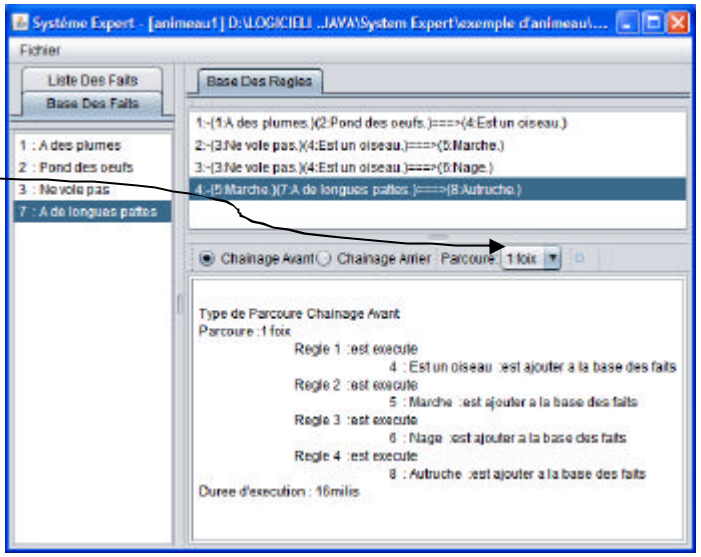
L'histogramme de la figure 4.8 montre clairement le degré d'optimisation atteint, et la différence du temps d'exécution entre les deux méthodes, et cela même dans le cas où le nombre de règles est très différent entre les bases de règles.

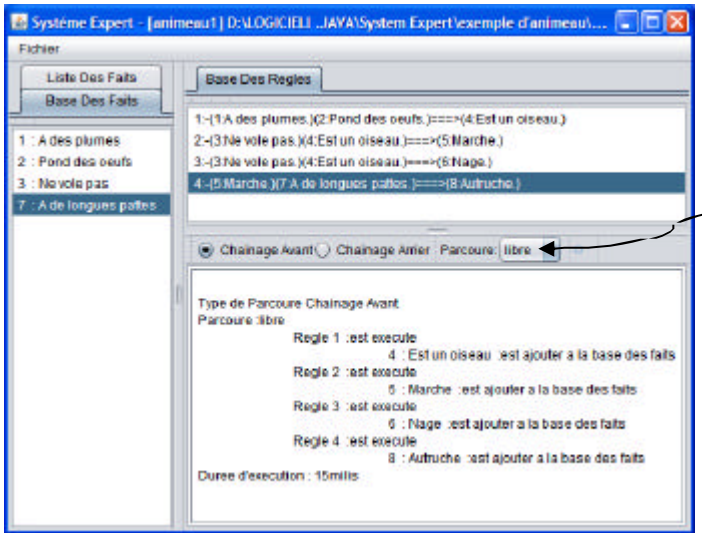
Cas 2 :

? La base de règles ordonnée:

On fait la comparaison entre le cycle d'exécution da la base de règles ordonnée par le moteur de chaînage avant d'un système expert.

Le moteur de chaînage avant parcours la base de règles une seule fois



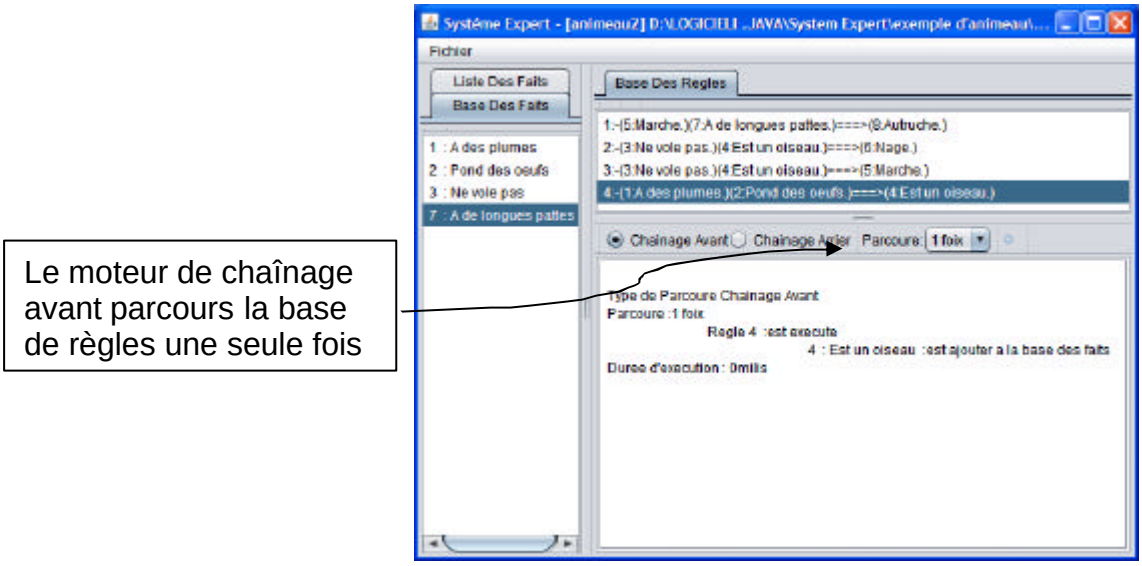


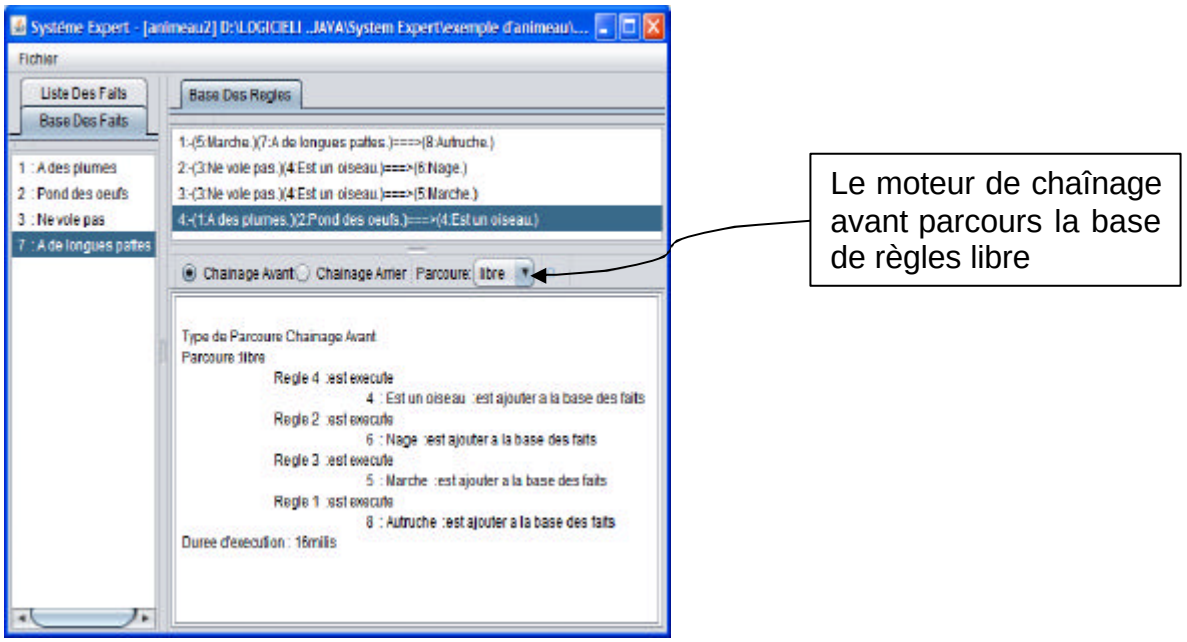
Le moteur de chaînage avant parcours la base de règles libre

On remarque, le parcours de la base de règles une seule fois si le *Graphe de Relation* est sans circuits, où deux fois s'il comporte des circuits (problème de feedback).

? La base de règles non ordonnée:

On opère de la même manière dans le cas d'une base de règles non ordonnée, on compare le cycle d'exécution, on remarque que si le moteur de chaînage avant s'exécute en une seule fois, il ne donne pas toutes les règles exécutées.





En fait un tableau pour simplifie :

	La base de règles ordonnée		La base de règles Non ordonnée	La base de règles Mal ordonnée
	Sans Circuits	Avec Circuits		
Nombre de parcours « **P** »	**P**=1	**P** = 2	1 = **P** = le nombre de règles d'une base de règles	**P** =le nombre des niveaux de la mise en ordre

Figure 4.8. Tableau comparatif pour les règles ordonnées et non ordonnées.

4.5. Conclusion

Dans ce chapitre nous avons testé notre approche d'ordonnancement par l'implémentation des bases de règles d'un système de raisonnement par les graphes, pour l'optimisation du temps d'inférence. Dans ce chapitre, nous avons présenté quelques résultats de simulation. Vues les difficultés d'accéder à des bases de règles d'un système de raisonnement. Ces simulations ont montré l'intérêt des approches que nous avons proposées pour l'optimisation des bases de règles. Pour le système de raisonnement à base de règles, nous avons testé les résultats sur l'agent "To" de l'équipe de recherche informatique GLOODO. Il reste à vérifier le comportement du système global dans des cas de figure de grandeur réelle, ainsi que l'intégration de ce travail avec d'autres applications en cours de réalisation dans notre équipe.

CONCLUSIONS ET PERSPECTIVES D'AVENIR

Le travail présenté dans cette thèse consiste à utiliser le formalisme de graphe pour dans un modèle de représentation des règles dans la base de règles d'un système de déduction. Nous nous sommes dans un premier temps intéressés à la modélisation de notre problème au moyen des graphes, cela nous a menés dans un deuxième temps à la proposition d'un graphe à deux types de sommets. Nous avons montré comment modéliser la base de règles par des graphes que nous ensuite utilisés pour résoudre le problème d'ordonnancement, en fournissant une stratégie de modélisation.

Pour mener à bien cette stratégie, nous avons élaboré un algorithme de transformation de la base de règles en graphe à deux types de sommets, qui un domaine qui, à notre connaissance, n'a pas été étudié jusqu'à présent.

Cela a abouti en particulier à un nouveau formalisme qui permet de modéliser à l'aide de graphe des problèmes initialement exprimés dans un système à base de règles. Nous avons proposé deux approches pour la base de règles. La première stratégie consiste à modéliser par un graphe à deux types de sommets. La seconde est caractérisée par graphe orienté (graphe de relation).

Ensuite, nous nous sommes intéressés au développement d'un mécanisme de représentation d'une base de règles. Cette contribution se traduit par la définition d'une approche d'une règle de production. Les principaux avantages de l'approche de la visualisation et la manipulation. La première proposition de l'approche s'appuie sur un algorithme de chemin. Cet algorithme utilise un chemin d'un graphe. Un des principaux intérêts de cet algorithme est qu'il bénéficie des algorithmes connus sur

les graphes. On a montré qu'on est arrivé à minimiser le temps d'exécution. Dans notre travail nous avons proposé deux modèles pour la modélisation des bases de règles.

En perspectives, ces travaux ouvrent trois nouvelles pistes de recherche. La première piste consiste à la modélisation de base de règles par des graphes. A l'heure actuelle, seule la modélisation de la règle de production d'une base de règles par un nouveau graphe a été implémentée. Ces résultats doivent être complétés par l'implémentation des autres graphes ou en créer des nouveaux. Les jeux d'essais testés devront également être étendus à d'autres problèmes plus variés.

La deuxième piste de recherche consiste à valoriser le nouveau formalisme proposé. La méthode de résolution proposée dans ce document consiste simplement à produire un algorithme de chemin de graphe. Par la suite, de nouveaux algorithmes spécifiques plus efficaces devront être conçus.

La troisième piste de recherche consiste à évaluer expérimentalement de nouvelles variantes de l'approche à base de graphes, notamment en appliquant les techniques de modélisation, sur différents jeux de tests et de les comparer ensuite.

L'objectif de cette thèse était de proposer un système expert à base de graphe pour l'optimisation du temps d'inférence. Les tests effectués montrent que cet objectif est bel et bien atteint.

Pour le côté réalisation, toutes les briques de l'architecture ciblée ont été conçues et développées. Il ne va plus rester à effectuer que la phase d'intégration, c'est ce qui va être donc l'objet de nos travaux futurs.

BIBLIOGRAPHIE

1. Mazari. R., “étude et conception d’un framework à base de patrons, pour la construction d’un agent intelligent et mobile sous J2ME“, mémoire de magister en informatiques, université de Blida 2011.

2. Osório, F.S., “Un système hybride neuro-symbolique pour l'apprentissage automatique Constructif“, Thèse doctorat en informatique, L’Institut National Polytechnique de Grenoble - I.N.P.G. Laboratoire LEIBNIZ– IMAG 30, le 3 février 1998.

3. Hemmer, M.C., “Expert systems in chemistry research“, ISBN 978-1-4200-5323-4, CRC Press, Taylor & Francis Group, Boca Raton London New York, 2008.

4. Fortin, N., “ Conception d’outils logiciels d’aide à la décision appliqués au de vie des anodes d’une aluminerie“, université de Québec à Chicoutimi, Juillet 2008.

5. Chachoua, M., “Travaux d’Etude et de Recherche en Informatique“, Département d’Enseignement et de Recherche Informatique, Mai 2007.

6. Denis, F., “Intelligence artificielle : les systèmes experts“, Laboratoire d’Informatique fondamentale, Marseille, Laurent Miclet, ENSSAT-IRISA, Lannion, Janvier 2006.

7. Kazaz, A., “Application of an Expert System on the Fracture Mechanics of Concrete“. Artificial Intelligence, pp. 177-190, 2003.

8. Belaïd, S., “Intégration des problèmes de satisfaction de contraintes distribués et sécurisés dans les systèmes d'aide à la décision à base de connaissances“, Thèse doctorat, université de Paul Verlaine-Metz, école doctorale IAEM Lorraine, 10 décembre 2010.

9. Zbigniew. S., and Arciszewski. T., “Intelligent Agents in Design”, 15th international Conference on Design Theory and Methodology Chicago Illinois, September 2003.

10. Kuflik. T., and Shoval. P., “User Profile Generation for Intelligent Information Agents –Research in Progress“, Proceedings of the Conference on Advanced Information Systems Engineering, CaiSE’00, Stockolm, Suède, 2000.

11. Bakam. T., I, “Des systèmes multi-agent aux réseaux de pétri pour la gestion des ressources naturelles : Le cas de la faune à l’Est-Cameroun“, Département d’informatique, université de YAOUNDE I, thèse doctorat, 28 mars 2003.

12. Gruselin, F., et Vilz, J., “ Vérification et validation d’un système expert pour la mesure fonctionnel (cosmicxpert) “, Faculté Universitaire Notre Dame de la Paix, Maîtrise en Informatique, 2002-2003.

13. Cordier, A., “ Gestion des Connaissances pour des Systèmes à Base de Connaissances hybrides“, Mémoire de DEA DISIC, Université Claude Bernard Lyon I, Juin 2004.

14. Schreiber, G., and al., “Knowledge Engineering and Management: The CommonKADS Methodology“. The MIT Press, Cambridge, Massachusetts, 2000.

15. Grove, R.F., “Design and development of knowledge-based systems on the web“. In : Proceedings of ISCA 2000 : Ninth International Conference on Intelligence Systems, Louisville, KY, USA, International Society of Computer Applications (USCA), pp. 147-150, 2000.

16. Cowan, R., “Expert systems: aspects of and limitations to the codifiability of knowledge“. Research Policy, Volume 30, Issue 9, pp. 1355-1372, December 2001,

17. Bessiere, C., Chmeiss, A., and Sais, L., “Neighborhood-based variable ordering heuristics for the constraint satisfaction problem“. In Proceedings CP'01, Paphos, Cyprus, pp. 565-569, 2001.

18. Charbonnaud, P., “ Traitement Symbolique “, ch. 4, Représentation des connaissances, pp. 21-33, http://www.enit.fr/~charbonnaud/se:Consulté le 01 janvier 2011.

19. Gerbé, O., Guy, W.M., Rudolf, K. K., “Un métamodèle des graphes Conceptuels“. RSTI - RIA. Volume 21 – n° 2/2007, pp. 257-286. 2007.

20. Barthélemy, J.D., Haemmerlé, O., and Salvat, E., “A semantic validation of conceptual graphs“, Knowledge-Based Systems 19 (2006), pp. 498–510, 2006.

21. GOHMANN, C., “Application dans les modèles de représentation des connaissances à la prise en compte des émotions, de la personnalité et de l'expérience pour l'interaction PJ-PNJ“. 01/09/2006. Consulté le 01 janvier 2011.

22. Brézillon, P. “Contextual graphs: A context-based formalism for knowledge and reasoning in representation“, T_0 appear in Research Report, LIP6, University Paris 6, France, 2003.

23. Brézillon, P., Pasquier, L., and Pomerol, J.C., “Reasoning with contextual graphs“, European Journal of Operational Research, 136 (2002), pp. 290-298, 2002.

24. Rakotomalala, M.R., “graphe d’induction“, thèse de doctorat, université Claude Bernard-Lyon1, 13 Décembre 1997.

25. Loudcher, R.S., “ Contributions à l’extraction automatique de connaissances : application à l’analyse clinique de la marche“, thèse de doctorat, université Claude Bernard-Lyon1, 16 Décembre 1996.

26. Rakotomalala, R., “Arbres de Décision“, Laboratoire ERIC Université Lumière Lyon2 5, av. Mendés France, 69676 BRON cedex, © Revue MODULAD, 2005.

27. Parent, O., et Eustache, J., “Les Réseaux Bayésiens“, Master 2 Recherche Connaissance et Raisonnement, 2006 – 2007. University Claude Bernard Lyon, 2007.

28. Naïm, P., Wuillemin, P.H., Leray, P., Pourret, and O., Becker, A., “Réseaux Bayésiens“, Paris, Eyrolles, 2002.

29. Dima, M.A., “Modélisation et représentation de la connaissance pour la conception d'un système décisionnel dans un environnement informatique d'apprentissage en chirurgie“, Thèse doctorat en informatique, université Joseph Fourier-Grenoble I, le 30 Septembre 2008.

30. Lee, K C., Cho, H R., and Kim, J S., “An expert system using an extended AND–OR graph“, Knowledge-Based Systems 21 (2008) 38–51.

31. Ramirez, J., and Angélica, A., “Checking the consistency of a hybrid knowledge base system“, Knowledge-Based Systems 20 (2007) 225–237.

32. Vachon, J., Sahraoui, H A. and Essalih, M, Mili. H ¸“Vérification par model-checking de systèmes hybrides objets-règles“, L’Objet. Volume 10 – 2004, pp.1-15, 2004.

33. Siminski, R., “Extending Decision Units Conception Using Petri Nets“. Advances in Soft Computing **5**, 413–420(2006).www.springerlink.com. ©Springer-Verlag Berlin Heidelberg 2006.

34. Siminski .R. “Petri Net and Matrix Representation of Rule Knowledge Base for Verification Task“. University of Silesia, Institute of Computer Science Poland, 41-200 Sosnowiec, Bedzinska 39, Phone (+48 32) 2 918 381 ext. 768, Fax (+4832) 2 918 283, siminski@us.edu.pl.

35. Xudong, H., William C. Chu., and Hongji, Y., “A new approaches to verify rule-based systems using Petri nets“. Information and Software Technology 45 (2003) 663–669.

36. Nigro, J.M., Barloy, Y., “The meta-inferences engine: A new tool to manipulate metaknowledge“, Knowledge-Based Systems 21 (2008) 588–598.

37. Deveze, B., et Fouquin, M., “ Quelques algorithmes des sciences cognitives“, SCIA, Janvier 2004.

38. Gursaran .G.S, Kanungo .S, and Sinha .A.K. “Rule-base content verification using a digraph-based modelling approach“. Artificial Intelligence in Engineering 13 (3) (1999) 321–336.

39. Siminski, R., “Graph-Based Knowledge Representations for Decision Support Systems“. M. Kryszkiewicz et al. (Eds.): RSEISP 2007, LNAI 4585, pp. 436–444, 2007. © Springer-Verlag Berlin Heidelberg 2007.

40. Arman, N. “Generating Minimum-Cost Fault-Free Rule Bases Using Minimum Spanning Trees“. Pages 116-120. *Vol. 4, No. 3, December 2006.*

41. Berge. C. “Graphs et hypergraphes“. 2ème éd. Dunod, Paris, 1973.

42. P. Lacomme., C. Prins., and M. Sevaux., “Algorithmes de graphes“. Eyrolles 2e édition 2003. ISBN 2-212-11385-4.

43. Zidi. K., “Système Interactif d’Aide au Déplacement Multimodal (SIADM) “, école centrale de Lille, université des Sciences et Technologies de Lille, Thèse doctorat en informatique, le 13 décembre 2006.

44. Lopez, P., “Cours de GRAPHES“, LAAS–CNRS, le 30 novembre 2005, http://www.laas.fr/~lopez/cours/GRAPHES/graphes.html

45. Rabache, A., “Circuits dans le graphe de DeBruijn Recherche de Formules“, Mémoire de Stage de Master, Soutenue le 01/09/2006.

46. Hélary, J.M., “Algorithmique des Graphes“, IFSIC, Cours C66, Juin 2004.

47. Hawick, K.A., et James, H.A., “Enumerating Circuits and Loops in Graphs with Self-Arcs and Multiple-Arcs“, Computer Science, Institute for Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand.
k.a.hawick@massey.ac.nz;heath.james@sapac.edu.au.
Consulté le 12.01.2010.

48. Ramoul, A., “ Contribution à l’étude des noyaux dans les graphes orientés“, Mémoire de magister, université de Blida, le 30 Juin 2008.

49. Schwikowskia, B., and Speckenmeyerb, E., “On enumerating all minimal solutions of feedback problems“, Discrete Applied Mathematics. 117 (2002) pp.253–265, 2002.

50. SIKORA, F., “ sur La Comparaison de Réseaux d’interactions Protéiques“, Mémoire de master recherche, Université Paris-Est, le 05/09/2008.

51. Kratsch, D., Müller, H., et Todinca, L., “Feedback vertex set on AT-free graphs“, Discrete Applied Mathematics 156 (2008), pp.1936 –1947, 2008.

52. Fomin, F.V., et al., “On the minimum feedback vertex set problem Exact and Enumeration algorithms", © Springer Science + Business Media, Algorithmica (2008) 52: 293–307.

53. Lin, H.M., and Jou, J.Y., “On Computing the Minimum Feedback Vertex Set of a Directed Graph by Contraction Operations“, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 19, n°3, March 2000.

54. Lin, C., Zorian, Y. and Bhawmik, S., “Integration of Partial Scan and Built-In Self-Test, Journal of Electronic Testing Theory and Application, pp. 125-137, 1995.

55. VOGEL, I., “Utilisation pratique du reset partiel : initialisation pour le test intègre de circuits fortement séquentiels“, Thèse doctorat, Université de Montpellier, Le 20 Décembre 2002.

56. Cordier, A., “ Gestion des Connaissances pour des Systèmes à Base de Connaissances hybrides“, Mémoire de DEA, Université Claude Bernard Lyon I, Laboratoire LIRIS, Lyon, Juin 2004.