الجمـــهورية الجزائريـــة الديمقراطيـــة الشعبيـــة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT
SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE
UNIVERSITE 8 MAI 1945
GUELMA
Faculté de Mathématiques,
D'Informatique et de Sciences de la
Matière
Département : D'Informatique

وزارة التعليم العالـي
و البحث العلمـي
جامعـة 8 مـــاي 1945
قالمـــة
كليـة الرياضيـات، الإعـلام الآلـي
و علـوم المـادة
قسم: الإعلام الآلي

Systèmes Experts

1ère Année Master en Informatique

Support de cours réalisé par

Dr MEHENAOUI zohra

Mehenaoui.zohra@univ-guelma.dz
Zahra_mehnaoui@yahoo.fr

2020-2021

SYLLABUS

(Plan de cours)

Unité d'Enseignement : **UEF1**, Matière : Systèmes experts

Domaine/Filière : Informatique, Master1 Systèmes d'informations (SIQ)

Semestre : 1, **Année Universitaire** : 2019/2020

Crédits : 5, **Coefficient** : 3

Volume Horaire Hebdomadaire Total : 3H

- Cours Magistral (*Nombre d'heures par semaine*) : 1,5H.
- Travaux Dirigés (*Nombre d'heures par semaine*) : 1,5 H

Langue d'enseignement : Français

Enseignant responsable de la matière : MEHENAOUI. Z., **Grade** : MC « B »

Email : zahra_mehnaoui@yahoo.fr.

Objectifs :

Ce module permet à l'étudiant de s'initier aux techniques utilisées en intelligence artificielle

Programme du cours théorique :

Chapitre 1 :

Introduction à l'intelligence artificielle et domaines d'application

Chapitre 2 :

Formalismes de représentation des connaissances

Chapitre 3 :

Les systèmes inférentiels (Prolog, systèmes experts, ...)

Chapitre 4 :

Systèmes experts et application

Chapitre 5 :

Le raisonnement incertain

Chapitre 6 :

Méthodologie de construction des systèmes experts

Références bibliographiques :

Daniel Kayser, La représentation des connaissances, Hermes, 1997.

Thayse André, Approche logique de l'intelligence artificielle, Dunod informatique, 1990.

Avant-propos

Ce cours s'adresse aux étudiants de première année master informatique. Il doit permettre aux étudiants de s'initier aux techniques utilisées en intelligence artificielle (IA) notamment les systèmes experts. Ce cours permet aussi aux étudiants d'identifier quelques problèmes relevant du domaine de l'IA. Il permet également de maitriser les formalismes de représentation de connaissances.

Le support de cours est présenté d'une manière simple et directe. Il n'exige pas des connaissances approfondies particulières, sauf des connaissances en logique mathématique.

Le contenu présenté dans ce support est conforme au canevas de formation de master, option système informatique (SIQ). Le présent manuscrit est structuré en 6 chapitres. Dans le premier chapitre, nous présentons une **introduction à l'intelligence artificielle et ses domaines d'application.** Dans le deuxième chapitre**,** nous nous intéressons à un des domaines les plus importants de la recherche en intelligence artificielle, qui est **la représentation des connaissances.** Différents formalismes de représentation de la connaissance sont présentés. Le troisième et le quatrième chapitre sont consacrés à la présentation de deux systèmes inférentiels : **le Prolog et les Systèmes experts**. Le chapitre cinq introduit la notion du **raisonnement incertain**. Dans ce chapitre, les principales approches utilisées dans les systèmes experts pour le traitement de l'incertitude sont abordées. Le dernier chapitre de ce support présente **la méthodologie de construction des systèmes experts.**

Sommaire

Chapitre 3: Les systèmes Inférentiels: PROLOG

Chapitre 4: Systèmes Experts et Application

Liste des figures

Liste des tableaux

Chapitre 1

Introduction à l'Intelligence Artificielle et Domaines d'Application

1.1 Introduction

L'intelligence artificielle (IA) est depuis des années un point de focalisation des scientifiques, des médias, et du public. On s'interroge sur son existence en tant que discipline scientifique, sur ses objectifs et les difficultés rencontrées, sur ses champs d'application et sur sa situation actuelle et à venir.

Les premiers pas de l'intelligence artificielle remontent à 1950, où Alain Turing (1912-1954) a énoncé une vision de l'intelligence artificielle dans son article intitulé « *Computing Machinery and Intelligence* » dont lequel le mathématicien explore le problème de définir si une machine pouvait être considérée comme consciente ou non (Turing, 1950). Dans cet article, découlera *le test de Turing* qui permet d'évaluer la capacité d'une machine à tenir une conversation. En 1956, la discipline nommée *Intelligence artificielle* a pris naissance lors d'une conférence au États-Unis qui s'est tenue au Dartmouth Collège (McCarthy et al., 1955).

1.2 Définition de l'Intelligence Artificielle

Le terme intelligence artificielle a été défini par l'un de ses créateurs, Marvin Lee Minsky (1927-2016), comme étant « *la construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique* ».

Une autre définition prenant l'être humain comme référence est donnée par (Bellman, 1978) : « *l'intelligence artificielle est l'automatisation des activités associées au raisonnement humain, telles que la décision, la résolution de problèmes, l'apprentissage* ». Vers la fin des années quatre-vingt-dix, de nouvelles définitions ont été données à l'IA, entre autres : « l'intelligence artificielle est l'étude de la conception d'agents intelligents» (Poole et al., 1998).

Plusieurs définitions ont été données à l'IA, cependant Il n'y a pas vraiment de consensus sur la définition de cette discipline. Cependant, toutes les définitions s'accordent sur le fait que l'objectif de l'IA est de créer des systèmes intelligents, mais elles diffèrent significativement dans leur façon de définir l'intelligence. Quatre façons ont été déclinées pour définir l'intelligence :

a) **Agir comme des êtres humains** : c'est une définition opérationnelle de l'IA, selon laquelle une machine est considérée comme intelligente si elle peut converser de telle manière que les interrogateurs (humains) ne peuvent la distinguer d'un être humain (test de Turing).

b) **Penser comme des êtres humains** : selon cette approche, l'IA est une science expérimentale, car il faut comprendre comment fonctionne l'esprit humain pour le modéliser et ensuite évaluer les systèmes conçus par rapport à leurs similarités avec le raisonnement humain.

c) **Penser rationnellement** : selon cette définition, les systèmes doivent raisonner d'une manière rationnelle, c'est-à-dire en suivant les lois de la logique. Cette approche peut être critiquée car il semble que certaines capacités (la perception, par exemple) ne sont pas facilement exprimables en logique.

d) **Agir rationnellement** : cette définition concerne le développement des agents qui agissent pour mieux satisfaire leurs objectifs. Cette définition est plus générale, car une action rationnelle peut être une conséquence d'un raisonnement rationnel ou non. par exemple, le réflexe de retirer sa main d'un objet brulant est rationnel mais n'est pas le résultat d'une inférence logique

1.3 Historique

- **Les inspirateurs (Avant 1956)**

  - Calculabilité et Machine de Turing (Alain Turing, 1936) ;
  - Approche physiologique des neurones dans le cerveau, modèles ; mathématique abstrait composé de neurone en réseau (McCulloch & Pitts, 1943) ;
  - Architecture d'un calculateur (Von Neumann, 1945) ;
  - Cybernétique : décrit les contrôles et la stabilité de réseau électriques (Wiener, 1948) ;
  - Théorie de l'information : les signaux numériques, le codage informatique, la cryptographie (Shannon, 1949) ;
  - Premier ordinateur basé sur les réseaux de neurones (Minsky & Edwards, 1951)

- **Espoirs grandissants (1956-1969)**

  - **Naissance de L'IA** : Conférence de Dartmouth college (USA, 1956) : le terme IA est proposé par McCarthy (créateur du LISP) ;
  - Les programmes **Logical Theorist** (par Newell et Simon, 1956) et **Geometry Theorem** Prover (Gelernter) pour la démonstration des théorèmes mathématiques. Généraliser en 1958 pour arriver à **General Problem Solver,** dont le but est de construire un solveur de problèmes universel
  - Reconnaissance de caractère, la souris cybernétique, le perceptron (Rosenblatt, 58), dames anglaise (Samuel, 59).
  - **Les premiers défis** : Des étudiants de Minsky travaillèrent sur les petits problèmes ("microworlds") tels que les problèmes d'analogie donnant naissance au programme ANALOGY (Evans, 1963), ou encore les manipulations de cubes. Programme capables de jouer aux échecs (première idées en 1950 par Shannon, première victoire sur un champion du monde 1997 : deep blue bat Kasparov). Projet de traduction automatique (1966 : rapport ALPAC), et dialogue en langage naturel : ELIZA (Weizenbaum, 1965) et SHRDLU (Winograd, 1968). Ce fut aussi l'époque du Shakey (Stanford Research Institue, 1967) le premier robot à être capable de raisonner sur ses propres actions.

- **Premières déceptions (les années 1960-1970..**)

  - les prédictions faites par les chercheurs en lA avaient été beaucoup trop optimistes ;
  - le cas par exemple pour traduction automatique: approche purement syntaxique n'étaient pas suffisante ;
  - Annulation en 1966 de tout le financement du gouvernement américain pour les projets de traduction automatique ;
  - Grande déception lors de l'application d'algorithmes de l'IA aux problèmes de grande taille ;
  - 1973, l'arrêt du financement de la quasi-totalité des projets en IA en Grande Bretagne ;
  - Minsky et Papert prouvèrent dans leur livre « Perceptrons » de 1969 que les réseaux de neurones de l'époque ne pouvaient pas calculer certaines fonctions pourtant très simples, ce qui mit en cause toute la recherche en apprentissage automatique, entrainant une crise dans cette branche de l'IA.

- **Un tournant (Début des années 70)**

  - Beaucoup de micromondes possèdent un comportement « intelligent » mais les micromondes restent des micromondes et n'évoluent pas vers des applications réelles ;
  - Nouvelle conviction : Un comportement « intelligent » a besoin d'une connaissance approfondie du domaine étudie ;
  - Début des systèmes à base de connaissances et de l'ingénierie des connaissances ;

- **Systèmes experts (1969-1979)**

  - DENDRAL (en chimie) réalise l'analyse automatique de spectres de masse pour déterminer la structure moléculaire du corps chimique étudie ;
  - MYCIN (en médecine) diagnostique les maladies infectieuses du sang et propose un traitement adapté ;
  - Les générateurs de systèmes experts (NEXPERT, CLIPS, etc.).

- **L'IA dans l'industrie (1980-présent)**

  - L'entreprise DEC commença à utiliser un système expert d'aide à la configuration de systèmes informatiques, ce qui leur permit d'économiser des dizaines de millions de dollars chaque année ;
  - Systèmes experts : Digital Equipment (Xcon), Intellicorp
  - Lisp machines: LMI, Symbolics

- **Le retour des réseaux de neurones (1986- présent)**

  - La règle d'apprentissage « back-propagation » qui permit le développement des réseaux de neurones capables d'apprendre des fonctions très complexes (curieusement, cette règle avait déjà été proposée en 1969, mais n'avait eu aucun écho dans la communauté scientifique).

- **L'IA moderne** (**1987-présent**)

  - Depuis 1990, nouvelle vision : « les agents »
  - En 1997, la prédiction de H.Simon, qu'un ordinateur serait champion d'échec, s'est vérifiée (Deep Blue). Aujourd'hui les ordinateurs ont gagné les titres de champions du monde aux jeux de dames, d'Othello et d'échecs.

- Dans les années 1970, beaucoup croyait que les robots seraient partout de l'usine au domicile. Aujourd'hui quelques industries (automobile, électronique) sont robotisées mais les robots domestiques ne sont pas encore répandus. Aussi, des robots ont exploré Mars, ils réalisent des opérations du cerveau et du cœur
- L'intelligence artificielle est devenue au fil du temps une matière scientifique de plus en plus rigoureuse et formelle. La plupart des approches étudiées aujourd'hui sont basées sur des théories mathématiques ou des études expérimentales plutôt que sur l'intuition, et sont appliquées plus souvent aux problèmes issus du monde réel.

1.4 Les principaux langages de l'IA

- Lisp (1958, J. MacCarthy) : traitement des listes ;
- Prolog (1973, A. Colmerauer) : programmation logique ;
- SmallTalk (1972, A. Kay) : langage objet ;
- Les langages de Frame pour la représentation de connaissances (YAFOOL, KL-ONE) ;
- Tous les langages de la logique de description.

1.5 Les grands domaines de l'IA

L'IA s'est divisée en de nombreuses sous disciplines qui essaient chacune de traiter une partie du problème.

- **Les systèmes experts :** un système expert est un logiciel capable de simuler le comportement d'un expert humain effectuant une tâche précise. Le succès de l'intelligence artificielle dans ce domaine est certain, dû au caractère ciblé de l'activité qu'on lui demande de simuler (ex. tâche de diagnostic en médecine, configuration d'appareils,…etc).
- **La représentation de connaissance :** si l'on veut qu'un logiciel soit capable de manipuler des connaissances, il faut savoir les représenter symboliquement. C'est l'un des secteurs les plus importants de la recherche artificielle.
- **Le traitement du langage naturel :** qu'il s'agisse de traduire un texte dans une autre langue ou de le résumer, le problème crucial à résoudre est celui de sa compréhension (ex. Apparition des interfaces en langage naturel ⇒ interrogation de bases de données en français ou en anglais).

- **La résolution de problèmes :** l'objectif est de créer des algorithmes généraux pour résoudre des problèmes généraux. Les jeux fournissent une bonne illustration de ce domaine (ex. Echecs, Othello, BackGammon …etc.)
- **Reconnaissance des formes** : reconnaissance de la parole (ex. réservation d'hôtel, etc.), de l'image (ex. reconnaissance de visage, etc.) et de l'écriture (ex. reconnaissance de chèques, reconnaissance des codes postaux, etc).
- **La robotique** : cette discipline vise à réaliser des agents physiques qui peuvent agir dans le monde.
- **Apprentissage** : consiste à créer des programmes qui génèrent leurs propres connaissances en se modifiant à partir de leurs succès et leurs erreurs.

1.6 Quelques difficultés rencontrées en IA

A. Difficulté de modélisation :

- Les problèmes ne sont pas toujours parfaitement définis ;
- Certaines notions sont difficiles à exprimer : possibilité, probabilité, préférence,…etc.

B. Difficulté de résolution :

- Difficulté de conception des algorithmes ;
- Espace de recherche très vaste ;
- Problèmes de temps de réponse :

C. Difficultés technologique :

- Toute avancée rendre des méthodes opérationnelles ;

D. Difficultés de généralisation :

- Les méthodes sont souvent dédiées à un problème particulier
- Des problèmes très variés.

1.7 Conclusion

L'intelligence artificielle est présente sous plusieurs formes, parfois inattendues et dans des domaines variés mais toutes ses applications ont pour but de simplifier la vie de l'être humain et ses travaux. La représentation des connaissances fait partie des domaines de recherche de l'intelligence artificielle. Le prochain chapitre est consacré aux différents formalismes de représentation des connaissances.

Chapitre 2

Formalismes de Représentation des Connaissances

2.1 Introduction

La représentation des connaissances est un domaine de recherche de l'intelligence artificielle qui traite des modèles de représentation de l'information ou des connaissances dans le but de former des hypothèses et de produire des inférences. Comme vu précédemment, L'IA cherche à reproduire l'intelligence artificielle pour résoudre des problèmes complexes. Pour simuler sur un ordinateur un processus de résolution de problème ou prise de décision, il faut disposer d'une représentation de la connaissance à exploiter. Une mise en œuvre de raisonnement sur les connaissances représentées est déclenchée pour trouver des solutions aux problèmes posés.

2.2 La représentation des connaissances

2.2.1 Notion de la représentation

On définit la connaissance ou les connaissances comme ce qu'on a appris par l'étude ou par la pratique. La représentation de la connaissance consiste en la reformuler en une forme symbolique compréhensible et interprétable par un ordinateur. Cette représentation constitue le support préalable aux traitements ultérieurs que l'on souhaite effectuer sur cette connaissance.

Les connaissances concernent des faits, considérés vrais dans un certain monde. Pour représenter ces faits on a recours à un formalisme ou mode de représentation. Le langage naturel est un outil informel pour représenter les connaissances. L'arithmétique et la logique, plus formels, sont aussi des outils de représentation.

2.2.2 Type de représentation des connaissances

De même qu'il n'y a pas de langage universel de programmation, il n'existe pas de formalisme "idéal" pour représenter les connaissances. Néanmoins, on peut faire la distinction entre trois types de représentation des connaissances :

A. **La représentation procédurale :** (automates finis, programmes) qui rends explicitement les inter-relations entre fragments de connaissances mais qui est facilement modifiable. Dans ce type de représentation, le problème est décomposé facilement et le déclaratif pur n'existe pas.
B. **La représentation déclarative :** (calcul des prédicats, règle de production**,** réseaux sémantique) où les fragments de connaissances sont indépendants les uns des autres. Cette modularité dans la représentation des connaissances facilite la tâche de modification. Les connaissances sont combinées avec un mécanisme général de raisonnement et déduction.
C. **La représentation mixte :** (objets structurés, frames, schémas scripts, objets…etc) mélange les deux modes de représentation précédentes.

2.2.3 Langage ou formalisme de représentation

Définition 2.2.3. Un langage formel est un langage qui utilise un ensemble de termes et de règles syntaxiques pour permettre de communiquer sans aucune ambiguïté (par opposition au langage naturel).

Il s'agit naturellement :

- d'un alphabet, ensemble de symboles pas nécessairement réduit à des caractères;
- d'un procédé de formation des expressions, pas nécessairement la concaténation;
- d'un ensemble d'*axiomes,* c'est-à-dire d'expressions obéissant aux deux premiers points ci-dessus, et dont on décide arbitrairement qu'ils appartiennent au système ;
- des règles de dérivation qui, à partir des axiomes, permettent de produire des *théorèmes* (c'est-à-dire des expressions appartenant au système), et peuvent ensuite s'appliquer aux théorèmes pour en produire d'autres.

2.3 Les formalismes de représentation des connaissances

La représentation des connaissances est le support préalable aux traitements ultérieurs que l'on souhaite effectuer sur ces connaissances. Le choix du formalisme à utiliser dépend à la fois du domaine d'application, des opérations à mettre en œuvre sur ces. Dans ce qui suit, quelques formalismes de représentation des connaissances largement utilisés en intelligence artificielle sont présentés.

2.3.1 Les formalismes logiques

Les formalismes logiques et surtout la logique des prédicats jouent un rôle important dans tout ce qui relève de la représentation des connaissances. Ils sont les premiers (chronologiquement parlant) formalismes utilisés en IA notamment pour les applications de démonstration automatique de théorèmes. Les formalismes logiques sont purement syntaxiques dont la sémantique est rigoureusement définie. Ils bénéficient d'une base mathématique solide en termes de mécanismes de raisonnement qui procèdent uniquement par manipulation symbolique.

La logique est une approche déclarative pour représenter les connaissances. Deux principaux outils mathématiques sont utilisés :

- **Logique propositionnelle :**
- Méthode de preuve efficace ;
- Trop restreinte car il n'y a pas de quantificateurs ;
- **Logique des prédicats :**
- Dans le cas général, indécidable ;
- Méthodes qui s'arrêteront si c'est un théorème mais risquent de ne pas s'arrêter si on l'applique à une formule non valide (semi-décidable).

La logique est basée sur l'utilisation des méthodes d'inférence bien connues pour déduire de nouvelles connaissances.

2.3.1.1 Aspect de la logique comme formalisme de représentation des connaissances

- **L'aspect syntaxique** concerne la définition de la syntaxe correcte des formules (dites bien formées) qui vont nous permettre d'écrire les énoncés concernant les faits du monde réel ;
- **L'aspect sémantique** concerne le calcul des valeurs de vérité des formules qui représentent ces faits en se basant sur *l'interprétation* des différents éléments constituant ces formules ;
- **L'aspect raisonnement** s'occupe de l'établissement de preuves formelles de la validité des formules (et donc des faits qu'elles représentent) en se basant sur des règles d'inférence valides et des formules initiales valides.

2.3.1.2 La logique propositionnelle

Une proposition est un énoncé qui peut être vrai ou faux (**les constantes**). La proposition constitue l'élément essentiel de la logique propositionnelle. Par exemple :

- Mohamed mange une pomme ;
- Le facteur vient de dimanche au jeudi ;
- Tous les nombres impairs sont des nombres premiers.

Dans le langage formel de la logique ces énoncés seront représentés par des **variables** (p, q, r, etc.).

2.3.1.2.1 Syntaxe des formules logiques

A. Connecteurs :

En plus de la représentation des connaissances dans des propositions simples (p : Mohamed mange une pomme), d'autres moyens sont utilisés pour énoncer des problèmes et des propositions inter-reliées du monde réel. Pour former des énoncés plus complexes, deux ou plusieurs propositions peuvent être combinées en utilisant des **connecteurs logiques**. Ces connecteurs ou opérateurs sont désignés comme étant les opérateurs logiques ET, OU, NÉGATION, IMPLIQUE et ÉQUIVALENT.

1) Connecteur	2) Symbol
ET	$\wedge$
OU	$\vee$
NÉGATION	$\neg$
IMPLIQUE	$\Rightarrow$
ÉQUIVALENT.	$\Leftrightarrow$

Donc, une formule propositionnelle est définie à l'aide de : constantes (symbole vrai $\top$, symbole faux$\bot$), variables et connecteurs. A ces éléments syntaxiques, des objets mathématiques sont associés:

- Un ensemble B = {0, 1}, dont les deux états sont appelés valeurs de vérités. On peut les notés par « 0 et 1 » ou « v et f » ;
- $f_{\neg}: B \rightarrow B$ une fonction de l'ensemble B dans l'ensemble B appelé fonction d'interprétation du connecteur $\neg$ (la négation) ;

- $f_{\vee}, f_{\wedge}, f_{\rightarrow} : B \times B \rightarrow B$ des fonctions de l'ensemble $B \times B$ dans l'ensemble B appelés fonctions d'interprétation respectivement des connecteurs : $\vee$ (ou), $\wedge$ (et) et $\Rightarrow$ (implique).

B. Les formules propositionnelles

L'ensemble des formules bien formées (*fbf*) est défini par induction comme suit :

- Toute constante est une *fbf* ;
- Toure variable est une *fbf* ;
- Si A et B sont des *fbf* alors :

$$\begin{cases} \neg A \\ A \wedge B \\ A \vee B \\ A \rightarrow B \end{cases} \quad sont\ des\ fbf$$

Les règles d'accumulation de priorité entre opérateurs permettent d'éviter l'accumulation des parenthèses :

$$p \wedge q \vee r \rightarrow p \wedge r \quad est\ équivalente\ \text{à}:\ ((p \wedge q) \vee r) \rightarrow (p \wedge r))$$

C. Sémantique des connecteurs

A chaque connecteur est associée une fonction d'interprétation. Ces fonctions sont définies par les tables de vérités suivantes :

p	$\neg p$
0	1
1	0

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

$p \rightarrow q$ signifie que si la proposition p est vraie, alors la proposition q est vraie également. Alors que l'équivalence $p \leftrightarrow q$ s'interprète comme étant « si et seulement si ».

2.3.1.2.2 Approche Sémantique (Théorie des modèles)

L'approche sémantique fait appel à la notion d'interprétation.

A. Notion d'interprétation

Toute formule avec *n* variables admet 2^n interprétations. La valeur de vérité associée par une interprétation I à une formule A dépend des valeurs de vérité assignées par I à chacune des variables de A.

Exemple 2.3.1.2.2.

La formule $p \rightarrow \neg q$ admet 4 interprétations qui donnent à la formule les valeurs de vérité indiquées dans le tableau suivant :

Interprétation	p	q	$p \rightarrow \neg q$
I_1	0	0	1
I_2	0	1	1
I_3	1	0	1
I_4	1	1	0

B. Modèles et satisfiabilité

Définition 2.3.1.2.2.1. Un modèle d'une formule Φ est une interprétation *M* telle que :

$$M(\Phi) = vrai$$

Il peut exister, zéro, un ou plusieurs modèles pour une formule ou un ensemble de formules.

Définition 2.3.1.2.2.2. S'il existe au moins un modèle pour la formule F on dit que F est *satisfiable* (ou vérifiable)*.*

Définition 2.3.1.2.2.3. Une *tautologie* est une formule vraie dans toutes ses interprétations.

2.3.1.2.3 Approche déductive (Théorie des preuves)

Dans l'approche sémantique, pour démontrer que f est une conséquence logique de E ($et\ on\ note\ E \vDash f$) il faut :

- Tous les modèles de E ;
- Pour chaque modèle M de E, vérifier que M(f)=v ;

Si le nombre de variables de E est n, le nombre de modèles potentiels est2^n. Donc, le temps de vérifications de tous les modèles croit exponentiellement avec le nombre de variables.

Dans l'approche déductive, on s'intéresse à la notion de preuve formelle. A partir d'axiome et d'hypothèse, on déduit de nouveaux faits en appliquant un ensemble de règles d'inférence.

A. Axiome

Un axiome est une formule bien formée posée comme théorème sans démonstration et admise a priori. Les axiomes sont obtenus à partir des schémas d'axiomes suivant :

1. $\big(P \to (Q \to P)\big)$;
2. $\big(P \to (Q \to R)\big) \to \big((P \to Q) \to (P \to R)\big)$;
3. $(\neg P \to \neg Q) \to \big((\neg Q \to P) \to Q\big)$.

B. Théorème

Un théorème est une formule bien formée démontrable à partir d'axiomes en utilisant les règles d'inférences (la notation $\vdash Q$ signifie que Q est un théorème).

C. Règles d'inférence

Schéma minimal de raisonnement valide qui permet de produire de nouvelles propositions à partir de prémisses qui sont soient des théorèmes, soient des hypothèses.

- ✓ La règle d'inférence ***modus ponens :***

$$Si\ P\ alors\ Q$$
$$et\ P$$
$$Alors\ Q$$

- ✓ ***Notation*** *:* $P \to Q, P \vdash Q$

Exemple 2.3.1.2.3. La règle de *modus ponens* nous permet de déduire à partir des deux énoncés suivant :

S'il pleut alors il y a des nuages

Il peut

L'énoncé :

Il y a des nuages

2.3.1.2.4 Eléments de modélisation

La notion de modélisation consiste à traduire une proposition exprimée en langage naturel et la mettre sous la forme d'une formule propositionnelle.

Exemple 2.3.1.2.4. Prenons par exemple la proposition : ***S'il pleut alors il y a des nuages.***

Pour modéliser notre proposition, on a besoin d'identifier des variables associées à certains faits qui peuvent être vrais ou faux.
Prenons la variable **p** associé au « fait il pleut », et la variable **q** associé au fait « il y a des nuages ». On doit également donner un sens aux valeurs de vérité associées à ces variables (voir le tableau suivant). Par exemple p s'interprète à 1 s'il pleut et à 0 s'il ne pleut pas. De même, q s'interprète à 1 s'il y a des nuages et à 0 s'il n'y a pas de nuages.

	Oui	Non
Il pleut	I(p) = 1	I(p) = 0
Il y a des nuages	I(q) = 1	I(q) = 0

Notre but est de trouver une formule propositionnelle Φ tel que : pour toute interprétation I, I falsifié Φ (I(Φ) = 0) si et seulement si elle est en contradiction avec la proposition à modéliser. C'est le cas de la formule :

$$\Phi : p \rightarrow q$$

2.3.1.3 La logique des prédicats

Bien que la logique propositionnelle soit un formalisme de représentation de connaissance, elle n'est pas vraiment utilisée dans les applications d'intelligence artificielle. Etant donné que la logique propositionnelle se base sur des énoncés complets qui peuvent seulement être vrais ou faux, ses capacités de représenter des connaissances du monde réel sont limitées. La logique propositionnelle suppose que le monde contient des faits, alors que la logique des prédicats suppose que le monde contient des objets (mohamed, systèmes experts, IA …etc), des fonctions sur les objets (TD de systèmes experts, TD de l'IA), et des relations entre les objets (Mohamed assiste au cours de systèmes experts).

C'est pour cette raison que l'IA, a su en tirer profit en utilisant « la logique des prédicat » comme un formalisme de représentation des connaissances dans de nombreuses applications où la logique propositionnelle se déclarait insuffisante.

2.3.1.3.1 Syntaxe de la logique des prédicats

A. Vocabulaire

- Des symboles de variables: x, y, u, v, w,…,z ;
- Des symboles de foncteurs (n-aires): f, g, h. Par exemple : *pere(x), f(x), g(x,y)* (si l'arité est zéro, constante) ;
- Des symboles de prédicats (n-aires): A, B, C,…,Z. Par exemple : Humain(x), P(y). (si l'arité est zéro, symbole propositionnel ou proposition) ;
- Des connectives logiques: $\rightarrow, \leftrightarrow, \wedge, \vee, \neg$
- Des quantificateurs: $\forall$ (Tout), $\exists$ (Il existe)

B. Langage

- **Les termes**: variables et constantes. Si f est un foncteur d'arité n et $t_1, t_2,\ldots,t_n$ sont des termes, alors $f(t_1,t_2,\ldots,t_n)$ est un terme.
- **Les formules atomiques**: Si P est un prédicat à n arguments ($n\geq 0$), et $t_1, t_2,\ldots,t_n$ sont des termes, alors $P(t_1,t_2,\ldots,t_n)$ est une formule atomique. Une formule atomique, ou sa négation est un littéral.
- **Les formules bien formées**: une formule atomique est une fbf. Si P et Q sont des fbf, et x est une variable, alors:

$\neg P$, $(P \rightarrow Q)$, et $(\forall x)$ P, sont des fbf.

Exemple 2.3.1.3.1

TOUT être humain est mortel.

$\forall$ **x** Etre-humain(**x**) $\Rightarrow$ Mortel(**x**)

Socrate est un être humain.

Etre-humain (socrate)

C. Procédure d'inférence

Dans tout système formel, certaines phrases sont désignées comme étant vraies. On les appelle des axiomes. Par exemple, tous les faits d'une base de connaissances sont des axiomes. L'inférence est le processus permettant de dériver des conclusions à partir

d'hypothèses. Une règle d'inférence comprend donc un ensemble de prémisses (ou hypothèses) et un ensemble de conclusions.

- **Règles d'inférences :**

Modus Ponens: Si Φ et $(\Phi \rightarrow \Psi)$ Alors on déduit Ψ

Modus Tollens: Si $\neg\Psi$ et $\Phi \rightarrow \Psi$ Alors $\neg\Phi$

Transitivité: Si $\Phi \rightarrow \Psi$ **et** $\Psi \rightarrow \Gamma$ Alors $\Phi \rightarrow \Gamma$

2.3.1.3.2 Eléments de modélisation

Le **logique des prédicats**, ou **logique du premier ordre** est par nature plus expressive que la logique des propositions, et permet de représenter les types de connaissances relatifs à des environnements complexes. Dans cette section, nous présentons quelques éléments utilisés dans le passage du langage naturel vers la logique du premier ordre.
La logique du premier ordre est en relation très étroite avec la théorie des ensembles. En effet, un ensemble peut être vu comme une collection d'objets avec une certaine propriété. Ainsi, on pourra exprimer les caractéristiques d'un individu par une formule de la logique des prédicats. Par exemple : $\exists x(Femme\ (x) \wedge Enseignante\ (x))$ correspond à l'intersection des deux ensembles Femme et Enseignante. Alors que la formule : $\forall x(Enceinte(x) \rightarrow Femme(x))$ exprime que l'ensemble Enceinte est inclus dans l'ensemble Femme.

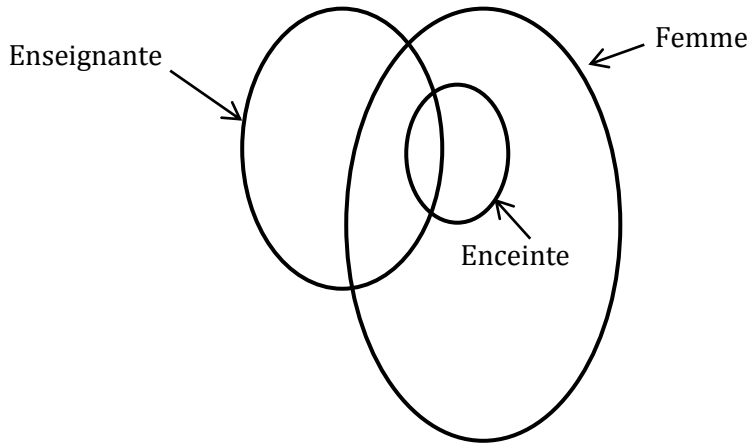


Figure 2.1 : La logique des prédicats et la théorie des ensembles.

- **Liens entre ∀ et ∃** : On a les lois de Morgan pour les quantificateurs :

$\neg(\forall x\ F) \equiv \exists x\ \neg F$

$\neg(\exists x\ F) \equiv \forall x\ \neg F$

$\forall x\ F \equiv \neg\exists x \neg F$

$\exists x\ F \equiv \neg\forall x \neg F$

Exemple 2.3.1.3.2.1.

« Tout le monde déteste les brocolis » revient au même que « Il n'existe personne qui aime les brocolis » :

$\forall x \neg Aime(x, brocolis) \equiv \neg\exists\ x\ Aime(x, brocolis)$

« Tout le monde aime les glaces » et « Il n'y a personne qui n'aime pas les glaces » sont équivalents :

$\forall x\ Aime(x, glaces) \equiv \neg\exists x \neg Aime(x, glaces)$

Exemple 2.3.1.3.2.2. Modélisation d'expression :

Tous les étudiants sont algériens : $\forall x\ Etu(x) \rightarrow Alg(x)$

Quelques étudiants ne sont pas algériens : $\exists x\ Etu(x) \wedge \neg Alg(x)$

Aucun étudiant n'est un algérien : $\forall x\ Etu(x) \rightarrow \neg\ Alg(x)$

Remarque : On traduit couramment certaines expressions en logique du premier ordre:

- « Tous les A sont B » : $\forall x\ (A(x) \rightarrow B(x))$;
- « Seuls les A sont B. » : $\forall x\ (A(x) \rightarrow B(x))$;
- « Aucun A n'est B » : $\forall x\ (A(x) \rightarrow \neg B(x))$;
- « Quelques A sont B» : $\exists x\ (A(x) \wedge B(x))$.

2.3.1.4 Autres types de logique

2.3.1.3.1 La logique modale

La logique propositionnelle comme la logique des prédicats décrivent des raisonnements sur des formules en décidant si elles sont vraies ou fausses. Mais c'est tout ce qu'elle peut en dire dans un modèle, on a soit p, soit ¬p. Dans la réflexion quotidienne, il y a des jugements plus nuancés sur la validité d'une proposition. Par exemple les phrases

- Il peut pleuvoir cet après-midi (éventualité) ;
- Je dois assister au TD (obligation) ;
- Je sais que tu es là (connaissance) ;
- …etc.

La logique modale permet d'introduire de nouveaux opérateurs appelés les opérateurs modaux, pour créer de nouveaux types de propositions. Parmi ces opérateurs modaux, on peut citer : (□ : nécessaire), (◊possible), (K : sait que), (B : croit que), (O : obligatoire), (A : permis).

Par exemple, si p est une proposition, □p est une nouvelle proposition (nécessaire p), de même que K.p (sait que p) ou A.p (autorisé p).

2.3.1.3.2 La logique floue

La logique floue est une extension de la logique booléenne créée par Lotfi Zadah en 1965. La logique floue permet de représenter des connaissances imparfaitement définies. Pour raisonner sur de telles connaissances, la logique classique ne suffit pas et on utilise une logique floue, lorsque les connaissances sont imprécises, vagues et éventuellement incertaines. Dans la logique floue, la valeur de vérité d'une proposition *p* n'est plus booléenne (0 (faux) ou 1 (vrai)) mais réelle : $p(v) \in [0 \quad 1]$, ce qui reflète l'appartenance à un ensemble floue.

Plusieurs techniques (Zadeh, Bandler,…) de représentations des opérateurs sur les ensembles flous ont été proposées. Dans le tableau suivant, nous présentons deux techniques de calcul de valeur de vérité pour les connecteurs : $\wedge$, $\vee$,$\neg$. p et q sont deux valeurs de vérités (des valeurs réelles) proposées à deux propositions déférentes :

	ET ($p \wedge q$)	OU ($p \vee q$)	NON($\neg p$)
Opérateurs de Zadeh	Min(p,q)	Max(p,q)	1-p
Opérateurs de Bandler	p*q	(p+q)-(p*q)	1-p

Tableau 2.1 : Calcul des valeurs de vérité dans la logique floue

2.3.1.3.2 La logique de description

La logique de description, appelée aussi la logique descriptive (LD) est une famille de formalismes de représentation de connaissances basés sur la logique. La logique de description permet de représenter la connaissance terminologique d'un domaine d'application d'une façon structurée et formelle en terme de concepts (classe), rôles (propriétés, relations) et individus (objets). La logique de description a été introduite

dans le but de rendre la représentation de connaissances plus naturelle qu'en logique du premier ordre.

2.3.2 Les réseaux sémantiques

L'utilisation des réseaux sémantiques comme formalisme de représentation de connaissances remonte aux travaux du linguiste Quillian (en 1968) sur la mémoire sémantique. En effet, les réseaux sémantiques sont très utilisés dans les travaux sur le traitement et la compréhension des langages naturels.

Définition 2.3.2.

Un réseau sémantique est un graphe orienté et étiqueté, composé d'un :

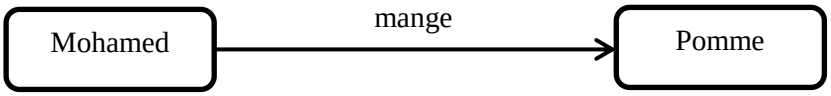
- Ensemble de nœuds représentant les concepts ;
- Ensemble de liens étiquetés entre les nœuds ;

De plus :

- Un ensemble d'opérations d'exploitation de ce graphe, constituant les mécanismes de raisonnement

2.3.2.1 Représentation d'un réseau sémantique

- **Représentation graphique** : facilite la lecture, ne correspond généralement pas au formalisme d'implémentation.



Ce réseau sémantique est représenté sous forme graphique. Cette représentation

- **Représentation non graphique** :

(alice, manger, pomme)

2.3.2.2 Concepts de base d'un réseau sémantique

- **Les nœuds :**

- Atomiques: entités élémentaires (valeurs, individu);
- Complexes: entités complexes (propositions, phrases);
- Ils doivent être typés: concept, individu, action, proposition, etc.

- **les liens :**

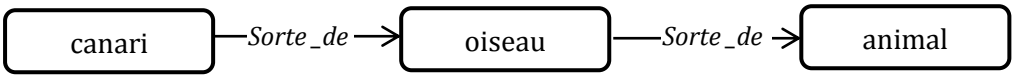
- Structuraux: indépendant de la sémantique du domaine ;
- Spécifiques: dépendant de la sémantique du domaine.

- **Les opérations :**

- Souvent représentées par des programmes ;
- Doivent être définies clairement.

Exemple 2.3.2.2.1. Nœud concept :

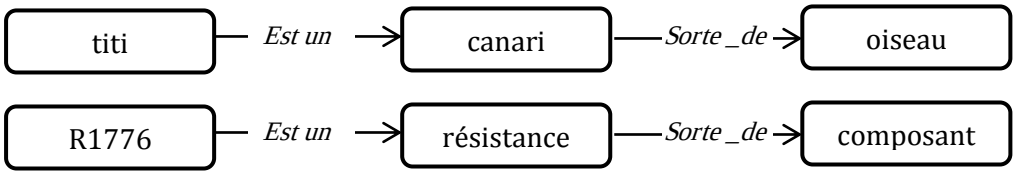
« Les canaris /sont de/ oiseaux »



- canaris et oiseaux = **concept** (nom commun) **-->** classes
- Sorte de = **relation -->** inclusion de classes
- La relation « **sorte de »** est un **lien structurel** indépendant du domaine, elle représente une inclusion.

Exemple 2.3.2.2.2. Nœud individu :

« Titi / est un / canari »



- Titi = **individu (nom propre) -->** élément d'un ensemble
- Est un (instance) = **relation -->** appartenance d'un élément à une classe
- La relation « **est un »** est un lien **structurel** indépendant du domaine.

Exemple 2.3.2.2.3. Un réseau sémantique :

Un réseau sémantique avec des nœuds concepts, des nœuds individus, des liens d'inclusions (sorte de) et des liens d'instanciation (instance ou est un).

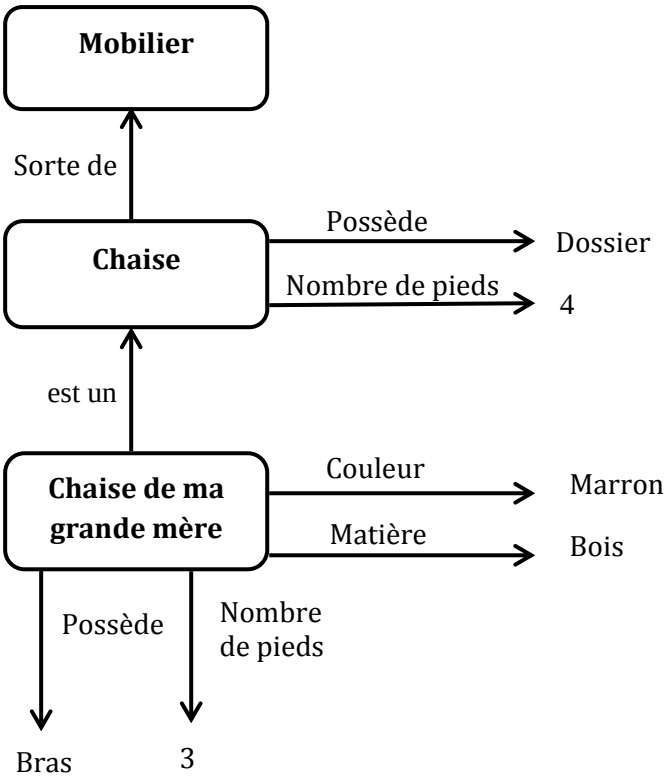


Figure 2.2 : Taxonomie d'un réseau sémantique.

Le nœud « mobilier » représente une classe d'objet dont le nœud « chaise » est un élément particulier.

De même pour les nœuds « chaise » et « chaise de ma grande mère ».

Cette structure hiérarchique permet aux nœuds des niveaux inférieurs d'hériter des propriétés des nœuds de niveaux supérieurs.

Du schéma de la figure précédente, nous pouvons déduire :

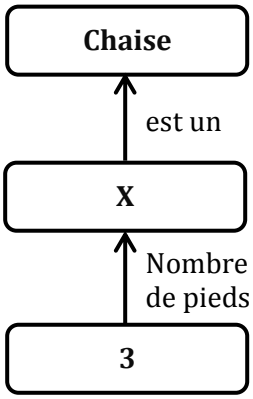
- La chaise de ma grande mère possède un dossier (héritage de propriété);
- La chaise de ma grande mère a 3 pieds en particulier, alors qu'une chaise en général (la classe d'objet) à 4 pieds. De là, nous pouvons dire que les propriétés du niveau inférieur (les exceptions) corrigent les propriétés du niveau supérieur (valeurs par défauts).

2.3.2.3 Fonctionnement d'un réseau sémantique

Les réseaux sémantiques peuvent exprimer n'importe quelle phrase. Ils peuvent également modéliser les quantificateurs, les implications…etc.

Cependant, toutes ces représentations ne sont opérationnelles que s'il existe des procédures qui les manipulent. Ces procédures fonctionnent comme un algorithme d'unification. Les variables inconnues et recherchées sont remplacées par des constantes en comparant les requêtes et l'ensemble des prédicats.

Par exemple la requête « quelle chaise a un nombre de pieds égal à 3 ? » sera représentée par :



L'étude des diagrammes trouve facilement que : « X = chaise de ma grande mère ». Cependant souvent, la réponse à cette question n'est pas immédiate et nécessite le parcours du réseau en vérifiant les attributs et les relations des entités impliquées.

Parmi les inconvénients de cette représentation est l'explosion combinatoire de la recherche d'information quand le nombre de nœuds et d'arcs devient important.

2.3.3 Les règles de production

Les règles de production permettent de représenter des connaissances opératoires. Chaque règle est un morceau indépendant de connaissance (on dit granule), elle contient toutes les conditions à son application. Les règles de productions sont représentées sous la forme de couples conditions-conclusions. La syntaxe de représentation d'une règle est la suivante :

Si	**<** Prémisse 1>	Conditions de déclanchement d'une règle
	< Prémisse 2>	
	…	
	< Prémisse n>	
Alors	< Conséquence 1>	des actions à envisager ou des conclusions qui s'appliquent si les prémisses sont vraies
	< Conséquence 2 >	
	< Conséquence p >	

Exemple 2.3.3. Une règle de production :

Si (le voyant rouge à gauche du tableau de bord est allumé)

Et (la température de l'eau est élevée)

Alors vérifier le niveau de l'huile

Un ensemble de règle de production forme une base de connaissance qui constitue une composante d'un système de production.
Un système de production comprend : une base de connaissances, une mémoire de travail et un mécanisme d'inférence.
Les connaissances initiales concernant les informations connues sur le problème ou les cas en cours sui sont fournies à la mémoire de travail. Le mécanisme d'inférence recherche dans la base de connaissances, des règles pouvant s'associer aux connaissances ou conditions initiales. L'exemple le plus connu d'un système de production est un système expert.
Parmi les systèmes experts qui utilisent cette méthode de représentation de connaissance, on peut citer : MYCIN (diagnostic médical), DENDRAL (détermination des structures des molécules), PUFF (maladies pulmonaires), DART (pannes d'ordinateurs),…etc.

2.3.4 Les objets structurés

Les différents formalismes de représentation de connaissances présentés se caractérisent par l'aspect déclaratif de la connaissance ainsi que sa nature inférentielle (les formalismes logiques et les règles de production).

Puisque les connaissances qui ne sont pas inférentielles sont difficilement représentables, des outils de représentation des connaissances mixte (déclaratives et procédurales) ont été développés afin de représenter les connaissances de nature structurée. Le terme objet structuré recouvre diverses applications telles que : frames (schémas), scripts et les langages objets. Ce formalisme de représentation de connaissance peut être considéré comme une extension des réseaux sémantiques qui introduise une représentation procédurale de la connaissance.

2.3.4.1 Les frames

On appelle schéma (en anglais frame) le sous-réseau sémantique identifié par un nœud accompagné de ses attributs et de leurs valeurs (éventuellement déterminées par des

attachements procéduraux). Le réseau devient un système de schémas lorsque seuls les liens d'héritage (relations est_un et sorte_de) sont exprimés et que les attributs et leurs valeurs sont encapsulés au niveau des nœuds.

Chaque schéma représente une **classe** (un ensemble) ou **un objet** (instance d'une classe). Comme dans les réseaux sémantiques, la relation *est_un* entre un schéma représentant un objet et un schéma représentant une classe exprime l'appartenance de l'objet à la classe. De même, la relation *sorte_de* entre un schéma représentant une classe et un schéma représentant une autre classe exprime l'inclusion de la première classe dans la seconde. Dans ce cas, on dit que la seconde classe est une superclasse de la première et que celle-ci est une sous-classe de la seconde.

Deux types de frames existent :

- Les frames **« prototypes »** représentent **une classe d'objets,** décrivant le contexte attaché à cette classe ;
- Les frames **« instances » :** réalisations particulières d'une classe donnée, décrivant des **individus** d'une classe.

On peut rattacher deux types d'informations à un Frame :

- des informations relatives à la description des objets : ***partie déclarative*** (propriétés relations d'états relations entre les objets, … etc)
- des informations relatives à la manipulation des objets : ***partie procédurale*** c'est-à-dire les opérations (procédures & méthodes) connues par l'objet (définition du contexte d'activation de l'objet, comment calculer une propriété, action à exécuter dans un contexte donné).

Un frame contient un certain nombre d'attributs (ou slots) qui sont les noms des propriétés caractérisant l'objet. Ses attributs sont caractérisés par des aspects. Plusieurs aspects peuvent être rencontrés, on peut citer :

- Aspects de typage : permet de définir le type de valeur des attributs d'un frame (les valeurs permises : entier, booléen, chaîne de caractère restriction de typage : intervalle, liste de…etc.) ;
- Aspect valeur : introduit la valeur d'un attribut ;
- Aspect défaut : introduit une valeur par défaut.

Exemple 2.3.4.1. Forme générale d'un frame :

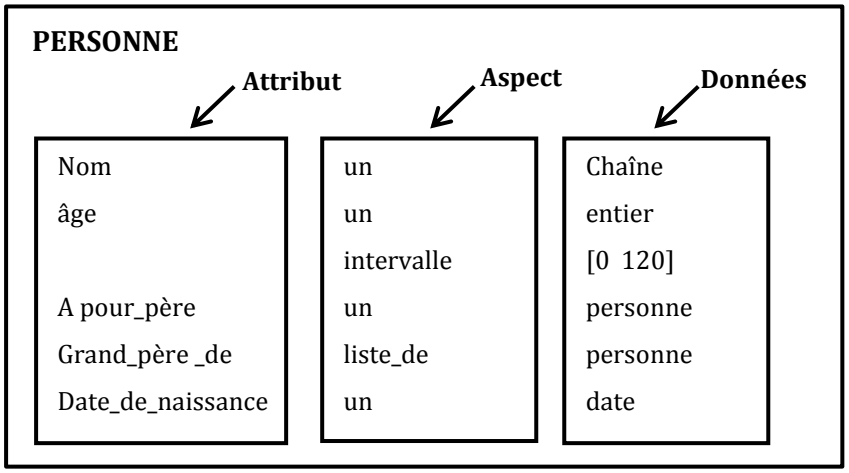


Figure2.3 : Un frame prototype.

2.3.4.2 Les scripts

Un script est une représentation structurée d'une suite d'événements, dans un contexte particulier. Un script est une forme de connaissance déclarative qui peut servir à :

- Communiquer, comprendre et raisonner sur une histoire ;
- Comprendre ou raisonner sur une situation ou une séquence d'évènements ;
- Raisonner sur les actions ;
- Planifier les actions ;

Un script est composé de :

- Les conditions d'entrée : l'ensemble des conditions nécessaires pour que le script puisse être appelé (exemple : le restaurant est ouvert, le client a faim) ;
- Le résultat : l'ensemble des faits après la réalisation du script (exemple : le client s'est appauvri, le client n'a plus faim, le restaurateur a plus d'argent) ;
- Les objets spécifiques (props) : l'ensemble des entités pour lesquels sont affirmés des faits (exemple : tables, chaises, garçon, client, chef, argent) ;
- Les rôles : l'ensemble des actions que chaque entité doit effectuer ou subir (exemple : le client sert à table et présente le menu, le client mange et paye) ;
- Les scènes : l'ensemble des suites d'évènement concernant les objets et les rôles.

Le script est divisé en une suite de scènes (exemple : l'entrée au restaurant, la commande, la consommation et la sortie du restaurant). Dans chaque scène, les acteurs

font les actions. Les acteurs agissent dans les lieux de la scène à travers les props. Les situations peuvent être organisées en séquence, arbre ou réseau.

2.4 Conclusion

L'intelligence artificielle cherche à reproduire l'intelligence humaine pour résoudre des problèmes complexes. Pour se faire, une représentation explicite des connaissances s'impose. Dans ce chapitre, nous avons présenté les formalismes largement utilisés pour la représentation des connaissances à savoir, les formalismes logiques, les réseaux sémantiques, les règles de production et les objets structurés.

Dans le prochain chapitre, nous allons présenter le langage Prolog comme étant un système inférentienl.

2.5 Exercices

Exercice N° 1

Soit p : « il fait froid » et q : « il pleut »

Enoncer des phrases simples qui traduisent chacun des énoncés suivants :

(1) $\neg p$; (2) $p \wedge q$, (3) $p \vee q$, (4) $q \vee \neg p$, (5) $\neg p \vee \neg q$, (6) $\neg\neg p$.

Exercice N° 2

Etablir la contraposition de chacun des énoncés suivant :
« S'il a du courage, il vaincra » ;
« Il ne vaincra, que s'il ne se fatigue pas »
$P \Rightarrow \neg q$
$\neg p \Rightarrow q$
De la réciproque de $\neg p \Rightarrow \neg q$

Exercice N° 3

Représenter les propositions suivantes dans la logique propositionnelle.

1. La musique n'est ni triste ni rythmée.
2. Quand il écoute de la musique rythmée, il est joyeux et il danse.
3. Il danse, sauf s'il n'est pas joyeux
4. S'il danse en baillant, c'est qu'il n'est pas joyeux.

5. Il écoute de la musique triste sans bailler
6. Personne n'a ri, ou même souri.
7. Georges ne viendra que si Albert et Lucienne ne viennent pas.
8. Il n'y a pas de fumée sans feu.
9. Si tu ne m'aides pas quand j'en ai besoin, je ne t'aiderai pas quand tu en auras besoin.

Exercice N° 4

Ecrire sous forme symbolique les propositions suivantes :

(1) « il suffit qu'il pleuve pour que l'escargot sorte »
(2) « pour que Mohamed aille au cinéma, il faut et il suffit qu'on l'y ait invité. »
(3) Il faut qu'une porte soit ouverte ou fermée »

Exercice N° 5

Représenter les propositions suivantes dans la logique des prédicats

1. Jean est plus grand que Marie
2. Si Jean est un homme, alors il est mortel
3. Un chat est entré
4. Tous les éléphants ont une trompe
5. Tout le monde est loyal à quelqu'un
6. Tout le monde aime une chanson de Johnny
7. Pour tout crime il y a quelqu'un qui l'a commis
8. Seuls les gens malhonnêtes commettent des crimes
9. Les gens malhonnêtes arrêtés ne commettent pas de crimes
10. Chaque étudiant fait quelques déductions, mais aucun ne les fait tous
11. Chaque personne aime quelqu'un et personne n'aime tout le monde, ou bien quelqu'un aime tout le monde et quelqu'un n'aime personne.

Exercice N° 6

Traduire en français la formule suivante :

1) $\forall x \, \forall y \, \forall z \, (T(x) \wedge C(x, y) \wedge C(x, w) \wedge C(x,z) \wedge D(y, z) \wedge D(y,w)) \Rightarrow G(f(g(y), g(z)), g(w))$,

Avec T (x) : x est un triangle, C(x, y) : y est le côté de x, D(x, y) : x est différent de y, G(x, y) : x est plus grand que y, f(x, y) : somme de x et de y, g(x) : longueur de x.

Exercice N° 7

Représenter la suite des propositions suivantes par un réseau sémantique :
« Un félin est un carnivore. Un carnivore est un animal qui a les yeux dirigés vers l'avant et qui mange de la viande. Les pattes d'un félin ont des griffes à leurs extrémités. Un félin est un mammifère. Grisou et Garfield sont des chats. Grisou est un chat mâle. Léo est un lion. Les chats et les lions sont des félins. Un cheval est un équidé. Un équidé est un mammifère ».

Exercice N° 8

Soit l'énoncé suivant :

Ken Hill est un lanceur droitier pour les Expos de Montréal. La moyenne au bâton d'un lanceur est de .180. De façon générale, un joueur de base-ball frappe la balle selon sa dextérité. Habituellement mesurant 1,85m, il a une moyenne sur les buts de .250. Les joueurs de base-ball sont des adultes mâles. Les adultes mâles sont des personnes qui mesurent, en général, 1,75m. Une personne est habituellement de dextérité droite. Les Expos de Montréal sont des joueurs de base-ball ; ils constituent une équipe du base-ball majeur, dont l'uniforme est blanc à rayures bleues. Le gérant des Expos est Félipe Alou. Les équipes de base-ball majeur sont des équipes sportives. La terre comprend environ 6 milliards de personnes dont 2 milliards d'adultes mâles. Le base-ball majeur engage environ 672 joueurs par an dont 280 lanceurs. Il existe 28 équipes de base-ball de 24 joueurs chacune. Toute équipe sportive a un gérant et un uniforme distinctif.

Questions :

Après avoir analysé le texte, Donner une représentation ensemblistes permettant d'identifier les objets, les classes et les relations sorte-de et est-un.
Compléter la description du système de schéma avec ses données (les attributs)

2.6 Correction

Exercice N° 5

1. Grand (jean, marie)
2. Homme(j) $\Rightarrow$ Mortel(j)
3. $\exists$x (Chat(x) $\wedge$ Entrée(x))
4. $\forall$x (Elé(x) $\Rightarrow$Trop(x))
5. $\forall x \, \exists y$ loyal(x,y)
6. Cette phrase est ambiguë, on peut la représenter par :

$\forall x\ \exists y$ (Chanson_j(y) $\wedge$ Aime(x,y)) (Tout le monde aime au moins une chanson de Johnny, pas forcément la même) ou :

$\exists x$ (Chanson_j(x) $\wedge\ \forall y$ Aime(y,x)) (Il y a au moins une chanson de Johnny que tout le monde aime).

7. $\forall x$ crime(x) $\Rightarrow \exists$ y (personne(y) $\wedge$ commis_par(x, y))
8. $\forall x \forall y$ (commis_par(x,y) $\wedge$crime(x) $\wedge$personne(y) $\Rightarrow$ malhonnête(y))
9. $\forall x$ ((personne(x) $\wedge$ arrêté(x) $\wedge$ malhonnête(x) $\Rightarrow \neg \exists y$ (crime(y) $\wedge$ commis par(y, x))
10. $\forall x$ (Etudiant(x) $\rightarrow \exists y$ (Déduction(y) $\wedge$ Faire(x,y))) $\wedge\ \neg \exists x$ (Etudiant(x) $\wedge\ \forall y$ (Déduction(y) $\rightarrow$Faire(x,y))).

11. $(\forall x \exists y$ aime(x, y)$\wedge \neg(\exists x \forall y$ aime(x, y)))$\vee(\exists x \forall y$ aime(x, y)$\wedge \exists x \forall y \neg$aime(x, y))

Exercice N° 6

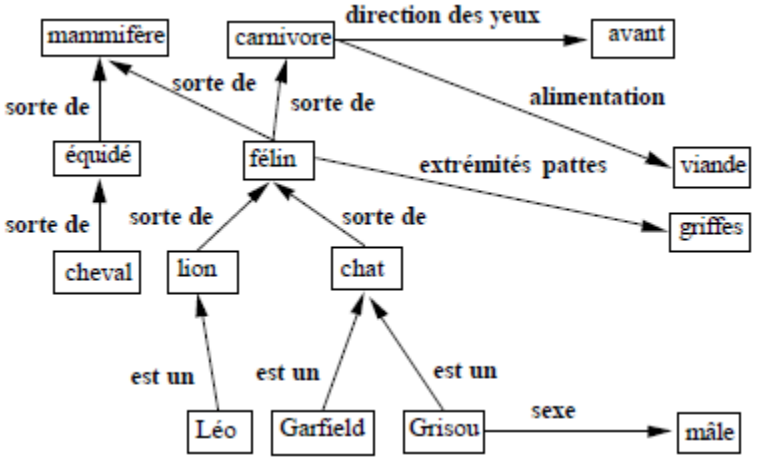
$\forall x\ \forall y\ \forall z$ (T (x)$\wedge$C(y, x)$\wedge$C(w, x)$\wedge$D(y, z)$\wedge$D(y,w)) $\Rightarrow$ G(f(g(y), g(z)), g(w)),

Avec T (x) : x est un triangle, C(x, y) : y est le côté de x, D(x, y) : x est différent de y, G(x, y) : x est plus grand que y, f(x, y) : somme de x et de y, g(x) : longueur de x.

La traduction de la formule est la suivante :

Pour tout triangle dont les côtés sont distincts, la somme des longueurs de deux de ses côtés est supérieure à la longueur du troisième.

Exercice N° 7



Chapitre 3

Les Systèmes Inférentiels : PROLOG

3.1 Introduction

La pratique de l'intelligence artificielle fait en sorte que la connaissance doit être utilisée pour faire des inférences, effectuer des raisonnements, répondre à des questions, ou tirer de nouvelles connaissances**.**

En logique, l'approche purement sémantique est basée sur la recherche des modèles, chose qui n'est en générale pas pratique. Par exemple, pour démontrer qu'une formule E, contenant n variables atomiques, est une tautologie, il faut vérifier tous les modèles potentiels qui sont au nombre de 2^n. Le temps de vérification croit exponentiellement avec le nombre de variables. L'approche syntaxique (déductive) a pour objet de calculer des conséquences logiques par l'application des règles d'inférences. Pour cela, des systèmes formels d'inférence, composés d'axiomes et de règles d'inférences, sont construits. La règle d'inférence la plus simple est « *le modus ponens* » qui permet d'affirmer à partir de la proposition « Si A alors B » et de l'assertion A, d'en déduire B.

Dans ce module, on va voir deux systèmes inférentiels. Le premier est le PROLOG, qui sera présenté dans ce chapitre. Le deuxième sera « les systèmes experts », qui font l'objet du prochain chapitre.

PROLOG est un langage qui, comme son nom l'indique (PROgrammation LOGique), utilise un mode de programmation dit « *logique* ». Ce mode de programmation a vu le jour grâce à John Robinson (1930-2016) qui a posé en 1965 les bases de la logique.

Les langages procéduraux traditionnels sont de nature « impérative », c'est à dire, pour résoudre un problème donné, le programmeur doit préciser dans un algorithme étape par étape la méthode de résolution de ce problème. Prolog est un langage où la programmation est de nature déclarative. Dans ce type de programmation, il revient au programmeur « de quoi » est fait le programme, plutôt que « comment » le résoudre (Boisard, 2014).

3.2 Symboles logique du Prolog

Prolog est fondamentalement un démonstrateur de théorèmes reposant sur le principe de résolution de Robinson. Prolog est basé sur la logique du premier ordre (logique des prédicats).

Langage naturel	Calcul prédicats	Prolog
Et	∧	,
Ou	∨	;
Seulement si	→	:-
négation	¬	**not**

Exemple 3.2. Soit les axiomes suivants :

Tous les hommes sont mortels

- **En logique des prédicats :** (∀x) (Homme(x) → Mortel(x)) ;
- **En Prolog :** mortel(X) :- homme(X).

Socrate est un homme

- En logique des prédicats : homme(socrate).
- En prolog : homme(socrate)

3.3 Constitution d'un programme Prolog

Un programme prolog est constitué de clauses. Celles-ci sont de trois types : faits, règles et questions. Le programmeur doit faire la description du problème à résoudre sous formes d'objets en spécifiant les relations entre ces objets.

3.3.1 Les faits

Les faits sont des données élémentaires qu'on considère vraies (les hypothèses de travail). Ce sont des affirmations qui décrivent des relations entre les objets ou des propriétés entre les objets (voir l'exemple 3.3.1). À partir des faits, le Prolog peut rechercher des preuves pour répondre aux requêtes des utilisateurs. Généralement, on place toutes les déclarations de faits au début du programme même si ce n'est pas obligatoire.

Exemple 3.3.1. Soit l'ensemble des faits suivants :

etudiant (mohamed, 2000, informatique, 2)

masculin (mohamed)

pere (ali, mohamed) % Ali est le père de Mohamed

Ali est le père de Mohamed signifie que la relation père lie les deux objets Ali et Mohamed.
On peut généraliser la clause *père (ali, mohamed)* par *père (X, Y)* où X et Y sont des objets.

La forme générale d'un fait est la suivante :

Prédicat (arguments1, argument2, …)

3.3.2 Les règles

Un programme Prolog contient presque toujours des règles, cependant ce n'est pas une obligation. Les règles sont des relations qui permettent, à partir des hypothèses, d'établir de nouveaux faits par déduction (Si on a démontré F_1 et $F_1 \rightarrow$ F2, alors on a démontré F_2).

La forme générale d'une règle est :

$$C :- H_1, H_2, \ldots, H_n$$

Cette règle se lit : C si $H_1, H_2, \ldots, H_n$ avec C, $H_1, H_2, \ldots, H_n$ des prédicats. C (partie gauche de la règle) représente la tête de la règle ou la conclusion. Elle est obligatoirement présente. $H_1, H_2, \ldots, H_n$ (partie droite de la règle) représente le corps de la règle, c'est-à-dire les contraintes liées par l'opérateur de conjonction. Le corps de la règle comprend 0/1 ou plusieurs termes. Ce sont les prémisses de la règle, qui doivent être vérifiées pour que la tête de la règle soit vraie.

Si $[(H_1 \wedge H_2 \wedge \ldots \wedge H_n)$ *alors* $C]$ est équivalente à $[(H_1 \wedge H_2 \wedge \ldots \wedge H_n) \rightarrow C]$ ce qui est équivalent à : $\neg H_1 \vee \neg H_2 \vee \ldots \vee \neg H_n \vee C$ ce qui représente une clause d'Horn en logique. Donc, un programme prolog rassemble une suite de clause d'Horn :

$$\neg H_1 \vee \neg H_2 \vee \ldots \vee \neg H_n \vee C$$
$$\neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n \vee D$$
$$\neg T_1 \vee \neg T_2 \vee \ldots \vee \neg T_n \vee C$$

Exemple 3.3.2. La relation telle que si on est le père du père ou de la mère de quelqu'un alors on est son grand-père se traduit par :

grand_pere(X,Y):-pere(X,Z), pere(Z,Y).

grand_pere(X,Y):-pere(X,Z), mere(Z,Y).

Ou:

Grand_pere(X,Y):-pere(X,Z),(pere(Z,Y); mere(Z,Y)).

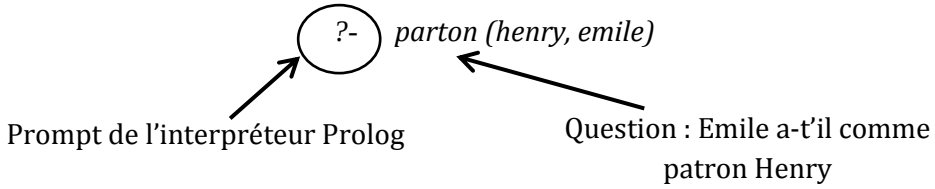
3.3.3 Les buts (Questions ou Requêtes)

A. Questions fermées :

L'intérêt de codage des informations sous forme de faits et règles dans un programme Prolog est l'interrogation de la base de connaissance, aussi bien par des questions fermées (dont la réponse est oui/non), que par des questions ouvertes à une ou plusieurs inconnues (trouver les objets pour lesquels une affirmation est vraie).

Exemple 3.3.3.1. Soit les requêtes suivantes :

1.



Ici, on interroge le système pour savoir si ce fait existe dans la base. C'est une question fermée (aucune variable), qui a comme réponse *vrai* ou *faux.*

2. *?- patron(henry, emile), patron(henry, joseph).*

Ici, on veut savoir si henry est le patron d'emile et de joseph.

B. Questions ouvertes :

- **les variables**

L'usage des variables en Prolog permet de transformer des questions fermées en questions ouvertes, c'est à dire en questions susceptibles d'avoir plusieurs réponses possibles.

Exemple 3.3.3.2.

?- *patron(X, emile).*

X=henry

?- *patron(X, Y).*

X = henry, Y = emile ;

X = henry, Y = joseph

Pour répondre à ces questions, l'interpréteur PROLOG doit remplacer la variable X par une des constantes apparaissant précédemment dans le programme, afin que le fait résultant existe dans la base de faits.

Les variables peuvent aussi être utilisées pour décrire des faits.

Par exemple : *patron (xavier, X)* est une transcription de la phrase : *« Xavier est le grand patron. »* et de la formule logique : ∀ x patron (xavier, x).

- **Les variables anonymes**

Elles se comportent comme des variables ordinaires dans la recherche des correspondances, mais elles ne prennent jamais de valeurs et n'apparaissent pas dans les réponses.

Notation : _ (*underscore)*

Exemple 3.3.3.3.

?- livre('Hugo', _, _).

Différentes occurrences du trait de soulignement _ désignent des variables anonymes différentes.

3.4 Les types de Prolog

Un terme en Prolog désigne un objet, qui peut être :

- **Une constante (appelée aussi terme atomique)** : Avec la syntaxe d'Edimbourg un atome peut s'écrire :
  - ✓ comme un **identificateur commençant par une minuscule**
    Exemple 3.2.4.1. jean, jean_paul_II
  - ✓ comme un **nombre**
    Exemple 3.2.4.2. 123, 1.23
  - ✓ comme une **chaîne de caractères entre apostrophes**
    Exemple 3.2.4.3. 'René', 'Jean-Claude'

- ✓ comme une **liste**

  Exemple 3.2.4.4. [ ], [a, b, 1, 5], ['andré',5,[paris,nancy]]

- **Une variable :** est une chaine alphanumérique commençant par une majuscule ou par _ :

  Exemple 3.2.4.5. X, Nom, _N53

- **Une fonction :** Une fonction est représenté par une chaîne alphanumérique commençant par une minuscule, suivie d'une liste de termes entre parenthèses et sépares par une virgule.

  Exemple 3.2.4.6. femme (marie)

  La fonction ≪ femme Marie ≫ est un nouvel objet construit à partir de l'objet jean. *femme/1* signifie que la fonction femme est d'arité 1, c'est-à-dire qu'elle n'accepte qu'un seul argument

3.5 Fonctionnement de Prolog

3.5.1 Séquencement

Lorsqu'on demande un but à Prolog, celui-ci se branche sur la première clause pouvant vérifier le but (basée sur le même prédicat du but). Si la première clause ne vérifie pas le but, il passe à la clause suivante et ainsi de suite. Prolog ne teste que les clauses du prédicat concerné et celles appelées par les prémisses d'une règle à exécuter.

3.5.2 Importance de l'ordre des clauses d'un prédicat

Dans la section clause, il faut toujours commencer par les faits. Ensuite les règles les plus particulières, puis les plus générales. En effet, si un fait est immédiatement concluant mais placé après une règle, il ne sera testé qu'après le déclenchement de la règle.

3.5.3 Recherche d'une solution par le moteur prolog

La recherche d'une solution par Prolog ne se termine que s'il trouve une clause vérifiant le prédicat concerné ou si toutes les clauses sont parcourues sans avoir trouvé une solution et c'est le cas d'échec. Le raisonnement Prolog est effectué sous forme arborescente suivant les appels aux différents prédicats. Et chaque prédicat essaye de se valider à travers ses différentes clauses.

3.5.4 Substitution et Unification

Répondre à une question comportant des variables consiste à déterminer les valeurs des variables telles que la question soit déductible des faits et règles du programme.
Pour démontrer une question, une des notions les plus importantes utilisées en programmation logique est la substitution.

Définition 3.5.4.1. Soient :

$Y_1, Y_2, \dots, Y_k$, k variables toutes différentes, $t_1, t_2, \dots, t_k$, k termes différentes ou non.
Une substitution σ est l'association qui fait correspondre : t_1 à Y_1, … , t_k à Y_k.
L'application d'une substitution σ (σ = { $Y_1 \leftarrow t_1$, …, $Y_k \leftarrow t_k$ }) à un terme ou une formule t consiste seulement à remplacer les occurrences de Y_1, …, Y_k dans t par les termes t_1, …, t_k (Ludovic, 2002).

Exemple 3.5.4.1.

Soit la substitution : σ = {$Y_1 \leftarrow$ Rima, $Y_2 \leftarrow$ Imene}, et soit la formule t=sœurs (X, Y).
Alors σ(t)= sœurs (Rima, Imene).
Remarque : si $Y \leftarrow t \in \sigma$, alors σ(Y) = t.

Définition 3.5.4.2.

Soient deux termes t_1 et t_2. Unifier t_1 et t_2 consiste à trouver la substitution la plus générale qui permet de faire de t_1 et t_2 un seul et même terme (Ludovic, 2002)

Remarque : En Prolog, l'unification est désigné par l'opérateur « = ».

Exemple 3.5.4.2. *Unification directe de deux termes*
$?- f(a, Y) = f(Z, T)$.
Pour unifier ces deux termes, Prolog tente d'unifier simultanément $\begin{cases} Z \leftrightarrow a \\ T \leftrightarrow Y \end{cases}$

Exemple 3.5.4.3. *Unification directe entre deux termes impossible*

$?- f(a, Y) = g(Z, T)$
No
L'unification n'est pas possible que si le nom des termes est le même pour les deux termes.
$?- f(a_1, a_2) = f(X_1, X_2, X_3)$

No

L'unification est également impossible si les deux termes ont un nombre différent d'arguments.

Exemple 3.5.4.4. *Deux unifications directes simultanées*

$?-f(a,Y)=f(Z,T),h(b,Z)=h(Y,U).$

$Y=b$

$Z=a$

$T=b$

$U=a$

yes

Pour résoudre ces deux unifications simultanées, Prolog se ramène à quatre unifications :

$a \leftrightarrow Z$

$Y \leftrightarrow T$

$b \leftrightarrow Y$

$Z \leftrightarrow U$

Et donc la solution la plus générale est : $\begin{cases} Z=U=a \\ T=Y=b \end{cases}$

Exemple 3.5.4.5. *Echec de l'unification à cause de boucle infinie*

$?-X=t(X,H).$

$X=t(t(t(t(t(\dots,\dots),H),H),H),H),H)$

Donc ici, l'unification aboutit à un échec et on doit l'arrêter pour éviter une boucle infinie.

3.6 Exécution d'un programme Prolog

- Pour répondre à une question (recherche des valeurs des variables d'un but), Prolog cherche à l'unifier, soit avec un fait, soit avec la tête d'une règle ;
- Si l'unification avec la tête de la règle réussit, Prolog procède une substitution des variables de la queue (corps de la règle) par les valeurs correspondantes dans la tête. Cette simplification des buts est appelée : la résolution **SLD** (*Sélection Linéaire Définie*, en anglais : *Selective Linear Definite*) ;
- Prolog continue à unifier jusqu'à vérification de la requête initiale ;
- Si l'unification **échoue**, Prolog répond par faux (false) ;

- Si l'unification réussit, Prolog répond soit par vrai (true), soit donne les valeurs des variables ;
- A la fin, Prolog construit une liste de couples valeur-variable.

3.6.1 Arbre SLD

Pour résoudre une question, Prolog effectue des essais consécutifs dans l'ordre de déclaration des prédicats. Ces essais peuvent être représentés un arbre de recherche ou arbre de résolution. Les nœuds de cet arbre constituent ce qu'on appelle **les points de choix.**

- La racine de l'arbre : le but à chercher
- Nœuds : points de choix ;

  Quand il y a plusieurs manières d'unifier, on a un point de choix :
  - ✓ Différentes règles définissant le même prédicat ;
  - ✓ Disjonction de prédicats dans le corps de la règle ;
  - ✓ Plusieurs possibilités d'instanciation d'une variable ;
- Passage d'un nœud vers son successeur en effectuant une unification. Les nœuds sont démontrés de gauche à droite, dans l'ordre de déclaration des règles ;
- A la fin, il y a des nœuds marqué par *succès* et des nœuds marqués par *échec.* les nœuds de succès ne contiennent plus de prémisses à vérifiées, tout a été démontré et les valeurs des variables du but sont trouvés en remontant vers la racine de l'arbre. Dans un nœud d'échec, aucune règle ne permet de vérifier la première formule du nœud.

Exemple 3.6.1. Soit le programme Prolog suivant :

$grandemere(X,Y) : -mere(X,Z), mere(Z,Y), femme(X).$
$grandemere(X,Y) : -mere(X,Z), pere(Z,Y), femme(X).$
$femme(nadia).$
$mere(nadia, rima).$
$mere(nadia, mohamed)$
$mere(rima, amine).$
$pere(mohamed, ali).$

Soit le but : ? $-grandemere(X,Y).$

L'arbre de résolution est donné par la figure suivant :

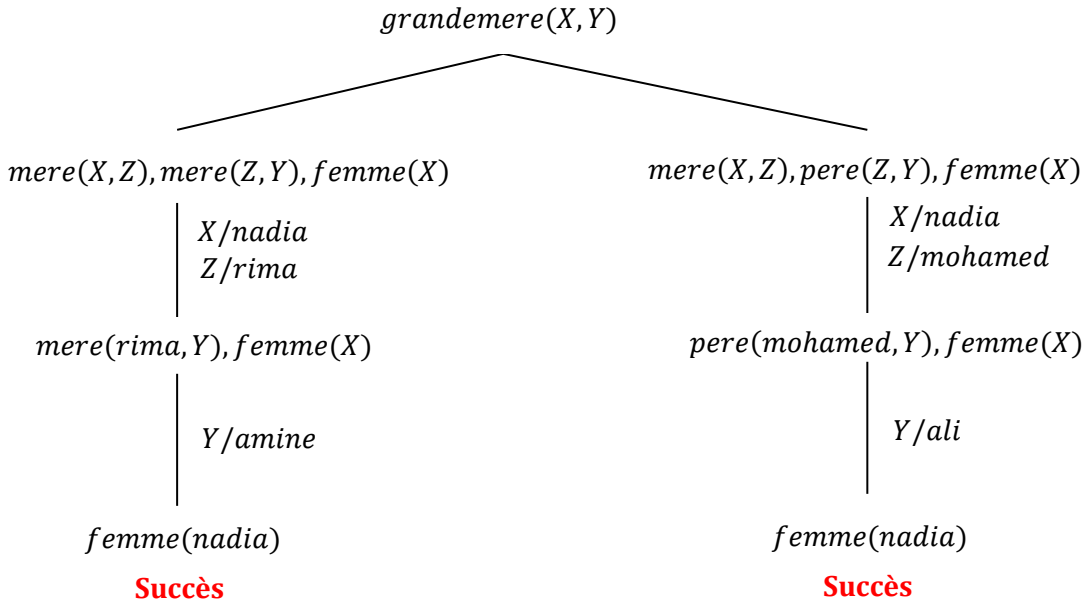


Figure 3.1 : Arbre de résolution du but grandemere(X, Y)

D'après l'arbre de résolution les solutions du but $grandemere(X, Y)$ sont :

$$\begin{cases} X = nadia, Y = amine \\ X = nadia, y = ali \end{cases}$$

Remarque : l'arbre présenté dans la figure 3.1 présente une partie de l'arbre de résolution complet. Un arbre de résolution complet sera présenté dans l'exemple suivant.

3.6.2 Stratégie de Prolog

Prolog est implémenté suivant une procédure de recherche en profondeur d'abord. En effet, Prolog tente d'aller le plus profond possible dans un chemin avant d'essayer un autre chemin (voir la figure 3.2).

Dans l'arbre SLD, les nœuds de succès représentent des solutions, d'autres solutions peuvent être cherchées. Si un nœud d'échec est atteint, Prolog remonte dans l'arbre de résolution jusqu'à un point de choix possédant des branches non explorées. Si un tel point de choix n'existe pas, la démonstration est terminée, et il n'y a plus d'autres solutions. Le parcours de l'arbre est fait de gauche à droite dans l'ordre des conditions. Cette stratégie de recherche de solutions est appelé le **Backtracking.**

Exemple 3.6.2.

$planete(X) : -astre(X), etoile(Y), satellite(X, Y).$

$astre(lune).$

$astre(terre)$.

$astre(soleil)$.

$astre(venus)$.

$satellite(venus, soleil)$.

$satellite(lune, terre)$.

$satellite(terre, soleil)$.

$etoile(vega)$.

$etoile(soleil)$.

Soit le but suivant : $? - planete(X)$.

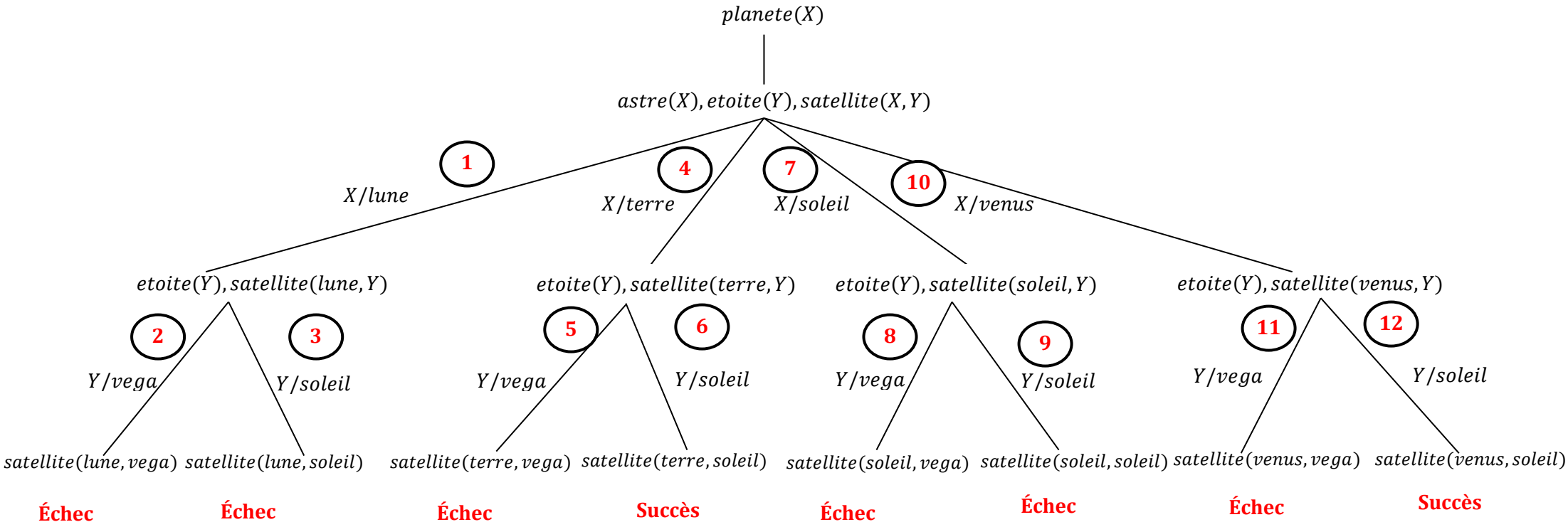


Figure 3.2 : Arbre de résolution du but planete(X).

$satellite(lune, vega)$: Échec → baccktracking (aller vers la branche 3)

$satellite(lune, soleil)$: Échec → backtracking (aller vers la branche 4)

$satellite(terre, vega)$: Échec → backtracking (aller vers la branche 6)

$satellite(terre, soleil)$: Succès → backtracking (aller vers la branche 7 pour chercher une autre solution)

3.7 Conclusion

Prolog (Programmation logique) est un langage de programmation déclaratif issu de la logique des prédicats. Il se base sur l'utilisation des faits, des règles et d'un moteur d'inférence. Prolog permet d'introduire des notions communes en IA comme les systèmes experts et le raisonnement logique.

Dns ce chapitre, nous avons présenté les notions relatives à ce langage de programmation, à travers ses constituants et son principe de fonctionnement.

Le prochain chapitre est toujours consacré aux systèmes inférentiels. Cette fois-ci nous allons présenter les systèmes experts.

3.8 Exercices

Exercice N°1

On considère les prédicats suivants :
parent(X,Y) = Y est le parent de X
femme(X) = X est une femme
homme(X) = X est un homme
Définir les prédicats familiaux suivants :
fille(X,Y) = Y est la fille X.
fils(X,Y) = Y est le fils X.
enfant(X,Y) = Y est l'enfant de X.
mere(X,Y) = Y est la mère X.
pere(X,Y) = Y est le père X.
grand_mere(X,Y) = Y est la grand-mère X.
grand_pere(X,Y) = Y est le grand-père X.
grand_parent(X,Y) = Y est le grand-parent X.
sœur(X,Y) = Y est la sœur X.
frere(X,Y) = Y est le frère de X.
oncle(X,Y) = Y est l'oncle de X.
cousine(X,Y) = Y est la cousine X.
cousin(X,Y) = Y est le cousin X.
tante(X,Y) = Y est la tante X.

ancêtre(X,Y) = Y est l'ancêtre de X.

Exercice N°2

On considère la base de données suivante, qui décrit les employés d'une entreprise (par exemple, pour la première clause, Antoine est dans le département des ventes, a une fonction de secrétaire, est dans l'entreprise depuis 6 ans, gagne 10000 € par an, et a pour chef xavier).

chef(employe(antoine,ventes,secretaire,6,10000),xavier).

chef(employe(xavier,ventes,directeur,2,15000),boss).

chef(employe(boss,direction,president,12,30000),boss).

chef(employe(lucie,achats,direction,1,14000),boss).

chef(employe(anne,achats,secretaire,11,10000),lucie).

chef(employe(jerome,achats,secretaire,11,10000),lucie).

chef(employe(etienne,achats,stagiaire,1,2000),anne).

Ecrivez des règles Prolog pour répondre aux questions suivantes :

1. departement : trouver le département dans lequel une personne travaille.
2. directeur/2 : étant donné le nom d'une personne, trouver qui est le directeur du département dans lequel elle travaille.
3. employe_valide : la structure de l'entreprise étant hiérarchique, on doit pouvoir remonter depuis n'importe quel employé vers le boss. Le prédicat employe_valide permettra de vérifier qu'un employé est bien sous les ordres du boss en remontant la chaine hiérachique..
4. salaire : donne le salaire d'un employé
5. salaire_reel : donne le salaire d'un employé en ajoutant au salaire de base un bonus, en utilisant les règles suivantes : 1/ tous les employés présents depuis 5 ans ou plus ont un bonus de 5000€. 2/ Aucune personne ne peut gagner plus que son chef (attention le cas du boss est évidemment spécial).

Exercice N°3

Soit le programme Prolog suivant :

oiseau(pigeon).

oiseau(hirondelle).

carnivore(loup).

carnivore(lion).

animal(lion).

animal(X) :-oiseau(X).

manger(X,Y) :-carnivore(X),animal(Y), X\=Y. 1.

1. Donner la solution de la requête : manger(lion,Y).

2. Donner l'arbre complet de résolution du but : manger(X,Y).

3. Déduire l'arbre de résolution de la requête : manger(lion,Y).

Exercice N°4

L'inspecteur Maigret veut connaître les suspects qu'il doit interroger pour un certain nombre de faits : il tient un individu pour suspect dès qu'il était présent dans un lieu, un jour où un vol a été commis et s'il a pu voler la victime. Un individu a pu voler, soit s'il était sans argent, soit par jalousie. On dispose de faits sur les vols, par exemple, Marie a été volée lundi à l'hippodrome, Jean, mardi au bar, Luc, jeudi au stade. Il sait que Max est sans argent et qu'Eve est très jalouse de Marie. Il est attesté par ailleurs que Max au bar mercredi, Eric au bar mardi. et qu'Eve était à l'hippodrome lundi. (On ne prend pas en compte la présence des victimes comme possibilité qu'ils aient été aussi voleurs ce jour-là). **Ecrire le programme Prolog** qui, à la question suspect(X), renverra toutes les réponses possibles et représenter l'arbre de recherche de Prolog.

Exercice N°5

Exemple des carnivores Ecrire les clauses Prolog correspondant au fait que les animaux sont herbivores ou carnivores, l'antilope est un herbivore, le lion est féroce et d'ailleurs tous les animaux féroces sont des carnivores. Les carnivores mangent de la viande et des herbivores, lesquels mangent de l'herbe. Tous boivent de l'eau. Qui consomme quoi ? Développer l'arbre de recherche.
Suivez l'ordre de l'énoncé dans la définition des clauses.
Les deux dernières clauses sont :
Consomme(X,Y) :-mange(X,Y).
Consomme(X,Y) :-boit(X,Y).

3.9 Correction

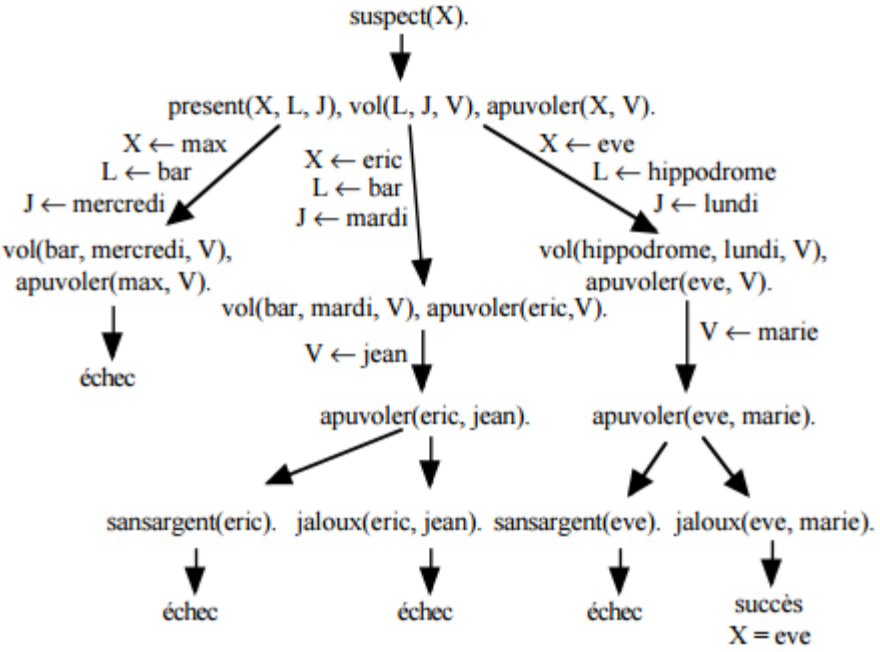
Exercice N°1

```
fille(X,Y):- parent(Y,X),femme(Y).
fils(X,Y):- parent(Y,X),homme(Y).
enfant(X,Y):- parent(Y,X).
mere(X,Y):- parent(X,Y),femme(Y).
pere(X,Y):- parent(X,Y),homme(Y).
grand_mere(X,Y):- mere(X,Z),mere(Z,Y).
grand_pere(X,Y):- pere(X,Z),pere(Z,Y).
grand_parent(X,Y):- grand_pere(X,Y);grand_mere(X,Y).
soeur(X,Y):- parent(X,Z),parent(Y,Z),femme(Y),X\=Y.
frere(X,Y):- parent(X,Z),parent(Y,Z),homme(Y),X\=Y.
cousine(X,Y):- oncle(X,Z),fille(Z,Y);tante(X,Z),fille(Z,Y).
cousin(X,Y):- oncle(X,Z),fils(Z,Y);tante(X,Z),fils(Z,Y).
tante(X,Y):- parent(X,Z),soeur(Z,Y).
oncle(X,Y):- parent(X,Z),frere(Z,Y).
ancetre(X,Y):- grand_parent(X,Z),parent(Z,Y).
```

Exercice N°4

Le programme Prolog correspondant à la requête suspect(X) :

```
suspect(X) :- present(X, L, J), vol(L, J, V), apuvoler(X, V).
 apuvoler(X, _) :- sansargent(X).
apuvoler(X, Y) :- jaloux(X, Y).
vol(hipp, lundi, marie).
vol(bar, mardi, jean).
vol(stade, jeudi, luc). sansargent(max).
jaloux(eve, marie).
present(max, bar, mercredi).
present(eric, bar, mardi).
present(eve, hipp, lundi).
```



Donc la solution est : **X=eve.**

Chapitre 4

Systèmes Experts et Applications

4.1 Introduction

La notion de systèmes experts (SE) est une notion assez ancienne qui est apparu dans les années 70 avec l'apparition du système expert célèbre MYCIN (Buchanan & Shortliffe., 1984). La version de base contenait 200 règles ensuite 300 règles concernant les méningites ont été ajoutées. Un autre système expert pionnier dans le domaine est le système DENDRAL créé par le groupe de recherche HPP (Stanford Heuristic Programming Project) (Bauchanan & Feigenbaum., 1980). Le système DENDRAL était le premier système basé sur des heuristiques pour effectuer des analyses expérimentales dans les sciences empiriques (Lindsay et *al*., 1993).

Les systèmes experts se basent sur le paradigme revendiquant la séparation entre les connaissances et le raisonnement. Aujourd'hui, les systèmes experts constituent une technologie bien définie faisant partie des *systèmes à base de connaissances*. Les systèmes experts ont comme finalité la modélisation puis la simulation, dans un logiciel, le savoir ou le savoir-faire d'un expert (ou d'un ensemble d'experts) dans un domaine donné (Shu-hsien, 2005).

4.2 Les domaines d'application des systèmes experts

Les systèmes experts ont résolu certains types de problèmes comme en médecine, en droit, en chimie, etc. Parmi les problèmes abordés par les systèmes experts, nous citons :

- Le diagnostic d'une défaillance à partir d'un ensemble d'observations ;
- La conception d'une configuration de composants à partir d'un ensemble de contraintes ;
- La planification d'une séquence d'actions pour l'accomplissement d'un ensemble de buts à partir de certaines conditions de départ et en présence de certaines contraintes ;
- La réparation d'un dysfonctionnement ;
- Le contrôle du comportement d'un environnement complexe.

4.3 Définition

Edward Feigenbaum[1] a défini les systèmes experts comme étant : « *Un programme informatique intelligent utilisant des connaissances et des procédures d'inférences pour résoudre des problèmes assez difficiles ayant besoin d'une expertise humaine importante pour leur solution* » (Giareatano & Riley., 2002).

Donc, un système expert est un outil capable de mettre en œuvre des connaissances pour imiter le comportement et/ou le raisonnement des experts humains dans des domaines d'expertise qui sont souvent basés sur un gros volume de connaissances. Plus c'est est un logiciel capable de répondre à des questions en effectuant un raisonnement à partir de faits et règles connus. Il peut servir notamment comme outil d'aide à la décision.

4.4 Composants d'un système expert

Un système expert est composé de deux parties complémentaires : la base de connaissance et le moteur d'inférence (voir la figure 4.1) (Ajith, 2005).

4.4.1 La base de connaissance

C'est la structure d'accueil de connaissance acquise dans un domaine d'application. Cette base de connaissance est constituée elle-même de deux parties :

- Une **Base de faits (l'expérience)** : contient les connaissances assertionnelles, c'est à dire les informations spécifiques et connues sur un problème (un cas) particulier traité. La valeur de vérité d'un fait est toujours vraie. La base de faits (**BF**) est la mémoire du travail d'un système expert, elle est modifiée au fur et à mesure de la progression du raisonnement.

Exemple 4.4.1. Exemple de quelques faits :

- Animal vole ;
- Poids animal est inférieur à 10 gr ;
- Animal pond des œufs.

- **Une Base de règles (le savoir-faire**) : la base de règle (**BR**) modélise la connaissance "générale" et le savoir de l'expert du domaine considéré et ce, sous

[1] Edward Feigenbaum (né en 1936) parmi les créateurs du premier système expert : DENDRAL

forme de règles codées avec le langage utilisé par le système. Les règles représentent les connaissances opératoires qui permettent la déduction pour un système expert. Ce sont des règles déductives appelées règles de production.

Une **règle** se met sous la forme suivante :

SI Condition_1 **et** Condition_2 **et** Condition_3 ... **et** Condition_n
ALORS Action_1 **et** Action_2 **et** Action_3 ... **et** Action_p.

Exemple 4.4.2. Une règle de production :

SI l'animal pond des œufs et l'animal vole **ALORS** l'animal est un oiseau

4.4.2 Le moteur d'inférence

Le système expert doit disposer, en outre, d'un système de contrôle et de raisonnement sur ces connaissances capable de répondre aux « *questions* » qui seront posées par les futurs utilisateurs du système. Ce système est connu dans le jargon des SE sous le nom de **moteur d'inférence** et il doit être assez général et indépendant du domaine concerné. Le moteur d'inférence permet d'inférer des connaissances nouvelles à partir de la base de connaissance du système.

Enfin, pour qu'un système expert soit un outil efficace, il doit être capable d'interagir avec ces utilisateurs dans le sens où il doit pouvoir (Ajith, 2005) :

- ✓ Fournir une **interface** pour acquérir ou modifier les connaissances et aussi pour recevoir les questions ;
- ✓ Expliquer son raisonnement et les réponses qu'il donne pour les questions.

Selon la représentation de connaissances, il y a deux générations principales de systèmes experts :

- Les systèmes experts de première génération : utilisent des règles **Si-Alors** pour représenter les connaissances ;
- Les systèmes experts de deuxième génération : sont plus flexibles et utilisent différentes formalismes pour représenter les connaissances.

La figure suivante, représente les composants de base d'un système expert.

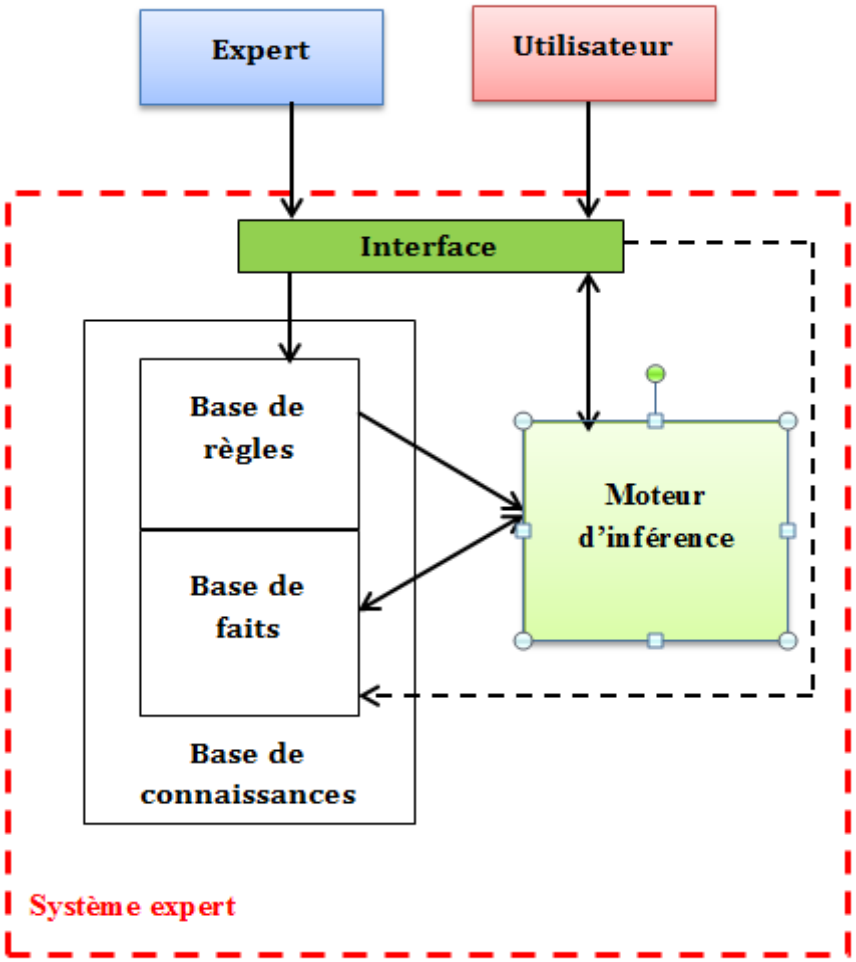


Figure 4.1 : Composants d'un système experts

4.5 Fonctionnement et caractéristiques d'un moteur d'inférence

L'indépendance entre la base de connaissances et le moteur d'inférence est une caractéristique essentielle des SE et c'est elle qui permet de faire évoluer la base de connaissances du système sans avoir à agir sur le moteur d'inférence qui peut être valide pour différents types de problèmes.

Le moteur d'inférence (appelé parfois moteur de résolution) est le cœur d'un SE. Il peut être indépendant du domaine d'application et il doit être capable d'exploiter la base des connaissances afin de résoudre les problèmes posés par les utilisateurs.

4.5.1 Cycles d'un moteur d'inférence

Un moteur d'inférence enchaîne une séquence de cycles pendant son raisonnement jusqu'à aboutit au but cherché ou jusqu'à saturation. Un cycle d'un moteur d'inférence est constitué de deux phases :

- Une phase d'évaluation ;
- Une phase d'exécution ;

Un système expert doit être capable de choisir les règles applicables vis-à-vis l'état courant de la base des faits et les faits à établir.

4.5.1.1 La phase d'évaluation

Le processus de détection des règles applicables, à la phase suivante, fait partie du cycle de travail d'un moteur d'inférence et particulièrement de sa **phase d'évaluation**. Cette phase d'évaluation s'effectue généralement en trois étapes :

- ✓ **La sélection ou restriction** : cette étape optionnelle concerne une première sélection d'un sous-ensemble de règles dans la base des règles qui –à priori- mérite d'être considéré dans l'étape suivante. Ce choix dépend de l'état courant de la base de faits.
- ✓ **Le filtrage (pattern matching)** : Parmi l'ensemble retenu à l'étape précédente, cette étape retient un sous-ensemble, dit ensemble de conflit, qui contient seulement les règles qui peuvent effectivement être applicables. Généralement, ce sous ensemble est constitué des règles dont les prémisses appartiennent à la base des faits.
- ✓ **La résolution de conflits** : Parmi les règles retenues suite au filtrage, le moteur décide de la (des) règle(s) qui sera (seront) exécutée(s).

4.5.1.2 La phase d'exécution

Comme son nom l'indique, la **phase d'exécution** concerne l'exécution de la partie 'action' des règles sélectionnées lors de la phase d'évaluation. Les nouveaux faits résultants seront ajoutés à la base des faits.

Remarque : L'arrêt de ce cycle peut survenir dans :

- ✓ la phase d'évaluation (par exemple, dans le cas d'absence de règles applicables à la situation en cours)
- ✓ la phase d'exécution (par exemple, dans le cas de l'exécution d'une règle dont la partie 'action' commande l'arrêt du cycle).

La figure suivante illustre, comment, en général, un moteur d'inférence enchaîne des cycles de travail composés de deux phases.

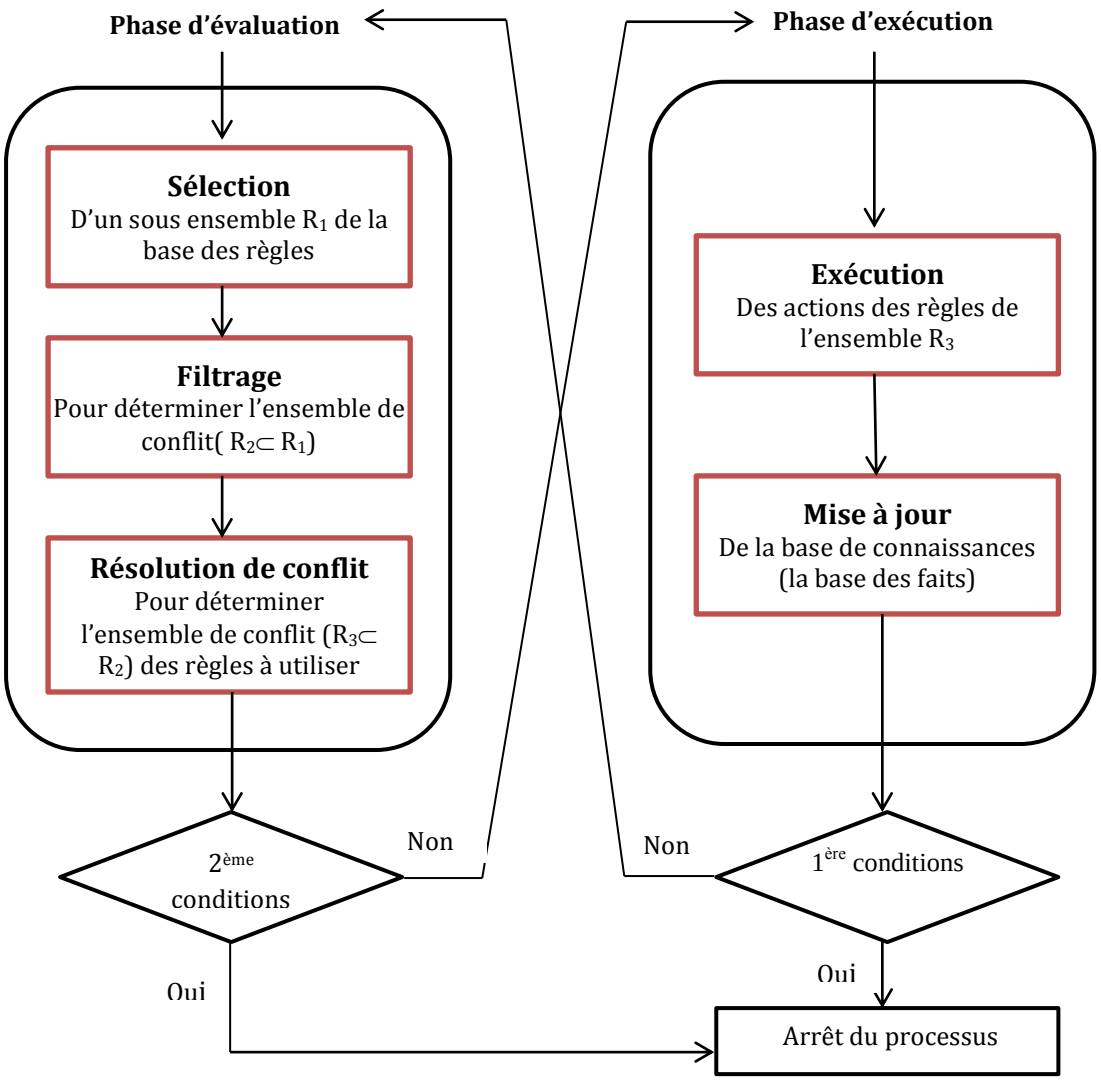


Figure 4.2 : Moteur d'inférence : les cycles de travail

4.5.2 Les caractéristiques d'un moteur d'inférence

L'objectif recherché par un moteur d'inférence lorsqu'il enchaîne des cycles de travail est de trouver une solution au problème (question) posé. En effet, la manière dont cette recherche est effectuée dépend des caractéristiques du moteur d'inférence utilisé. Les

caractéristiques de base d'un moteur de recherche sont : son ***formalisme de représentation de connaissance*** utilisé (généralement basé sur les logiques classiques ou ses extensions), son ***mode de chaînage,*** la ***stratégie de recherche ou de résolution de conflit*** qu'il utilise, son ***régime de contrôle***, et son ***type de raisonnement***. Ainsi, on choisissant à chaque fois, un formalisme de représentation de connaissance, un mode de chaînage, une stratégie de recherche, un régime de contrôle et en fixant un type de raisonnement, on obtient un moteur d'inférence complet et bien défini.

4.5.2.1 Le mode de chaînage

Les moteurs d'inférence sont caractérisés par leurs modes d'invocation des règles. Selon le mode de chaînage, En effet, on distingue trois types de moteurs d'inférence :

1) Les moteurs d'inférence à chaînage avant ;
2) Les moteurs d'inférence à chaînage arrière ;
3) Les moteurs d'inférence à chainage mixte.

Nous revenons sur ces trois modes de chaînage en paragraphe 4.6

4.5.2.2 Les stratégies de recherche

La résolution de conflit est souvent indépendante du contexte d'application et de la sémantique des règles, mais, elle est généralement guidée par des soucis d'efficacité (par exemple, choix des règles les moins complexes). Toutefois, des informations heuristiques peuvent être exploitées pour fixer un ordre de priorité entre ces règles. Les stratégies les plus fréquentes sont les suivantes :

- Stratégie en profondeur d'abord ;
- Stratégie en largeur d'abord ;
- Stratégie 'LEX' : règles triées par fraîcheur (recency) des faits de leurs conditions ;
- Stratégie basée sur la complexité : les règles dont le nombre de ces faits/conditions est le plus grand sont les plus prioritaires ;
- Stratégie basée sur la simplicité : les règles dont le nombre de leurs faits/conditions est le plus petit sont les plus prioritaires ;
- Stratégie basée sur les méta-règles : la connaissance est traduite en règles, la méta-connaissance s'exprime par des métarègles (c'est-à -dire des règles sur la manière d'utiliser les règles).

Exemple 4.5.2.2. On trouve par exemple dans MYCIN la métarègle suivante :
Si on recherche une thérapie Alors considérer dans cet ordre les règles qui permettent de :

– acquérir les informations cliniques sur le patient ;
– trouver quels organismes, s'il en existe, sont causes de l'infection ;
– identifier les organismes les plus vraisemblables ;
– trouver tous les médicaments potentiellement utiles ;
– choisir les plus adaptés en plus petit nombre.

4.5.2.3 Le régime de contrôle

Un moteur d'inférence peut être caractérisé par sa conduite dans le cas où sa phase d'exécution déboucherait sur un échec (par exemple, si R3 est vide). A ce niveau, il existe deux types de conduite :

- Les choix pris ne sont jamais remis en cause, ceci est traduit définitivement comme un échec. On dit alors que ce type de moteur suit un **régime irrévocable ;**
- Lorsqu'on peut remettre en cause l'application d'une règle si ce choix débouche sur un échec, le régime appliqué est dit par **tentatives** et il se base sur le principe de retour arrière (backtracking) pour essayer une règle écartée.

Les moteurs d'inférence qui adoptent un régime par tentatives suivent généralement des stratégies en profondeur d'abord puisque avec une stratégie en largeur le retour arrière est, en principe, inutile.

4.5.2.4 Le type de raisonnement

Un moteur d'inférence peut être basé sur une théorie logique **monotone** ou **non monotone.** Un fonctionnement monotone signifie que aucune connaissance ne peut être retirée de la base des faits ni de la base des règles**,** et que l'ajout d'un fait à la base des faits n'introduit pas de contradiction logique avec les autres faits de la base. Par contre, un moteur d'inférence non monotone, un fait ajouté à la base des faits précédemment peut être retiré dans le cas où des contradictions sont constatées ou en cas de backtracking par exemple. Les faits inférés à partir d'une règle qui n'a pas abouti, sont supprimés à cause de la remise en cause de la règle en question et le choix d'une autre règle en faisant un retour arrière.

4.5.2.5 Ordre d'un système expert à règle de production

L'ordre d'u moteur d'inférence revient à la complexité des règles de production figurant dans les règles. On peut distinguer les ordres suivant dans les systèmes experts :

- Ordre 0 : c'est un système experts qui utilise que des faits booléens ;
- Ordre 0+ :c'est un système expert qui utilise des faits booléens et des relations (souvent sous la forme *relation attribut valeur) ;*
- Ordre 1 : c'est un système expert dont les règles peuvent contenir des variables (les notions de la logique du premier ordre).

4.6 Type de moteurs d'inférence

4.6.1 Moteur d'inférence à chaînage avant

Dans le mode de chainage avant (farward chaining) : (connu aussi comme le mode guidé par les données), le moteur d'inférence procède comme suit :

- ✓ Il ne sélectionne que les règles dont les conditions (prémisses) appartiennent à la base des faits pour arriver au but recherché ;
- ✓ Il applique ensuite une de ces règles afin d'ajouter d'autres faits à la base ;
- ✓ Cet enchaînement est réitéré jusqu'à ce que le but soit atteint ou plus aucun fait nouveau ne puisse être déduit (on parle alors de saturation).

Le chaînage avant procède généralement par régime irrévocable et monotone et il a l'avantage d'être simple à implémenter tout en permettant de répondre plus rapidement à tout nouveau fait ajouté. Il est, toutefois, sujet au risque de l'explosion combinatoire car il sélectionne toutes les règles même celles sans intérêt pour la résolution du but.

Exemple 4.6.1.1 Un moteur d'inférence à chainage avant en profondeur d'abord monotone avec régime irrévocable :

Soit la BF = {B, C} et soit H le fait à établir en utilisant le chainage avant.
Soit BR composée des règles suivantes :

R_1 : Si B, D, E alors F

R2 : Si G, D alors A

R3 : Si C, F alors A

R4 : Si B alors X

R5 : Si D alors E

R6 : Si X, A alors H

R7 : Si C alors D

R8 : Si X, C alors A

R9 : Si X, B alors D

Solution

1. Conflit = {R4, R7}, règle choisie R4 :, BF = {B, C, X} ; désactiver R4 ;
2. Conflit = {~~R4~~, R7, R8, R9}, règle choisie : R7, BF = {B, C, X, D} ;
3. Conflit = {~~R4~~, ~~R7~~, R5, R8, R9, }, règle choisie : R5, BF = {B, C, X, D, E} ;
4. Conflit = {~~R4~~, ~~R7~~, ~~R5~~, R1, R8, R9}, règle choisie : R1, BF = {B, C, X, E, F} ;
5. Conflit = {~~R4~~, ~~R7~~, ~~R5~~, ~~R1~~, R3, R8, R9}, règle choisie : R3, BF = {B, C, X, E, F, A} ;
6. Conflit = {~~R4~~, ~~R7~~, ~~R5~~, ~~R1~~, ~~R3~~, R6, R8, R9}, règle choisie : R6, BF = {B, C, X, E, F, A, **H**} ;

Le but **H** est établi, donc on s'arrête.

Exemple 4.6.1.2 Un moteur d'inférence à chainage avant en largeur d'abord monotone avec régime irrévocable :

On utilise la même base de connaissance de l'exemple précédant, avec le même fait H à établir.

Solution

1. Conflit = {R4, R7}, on applique R4 puis R7, BF = {B, C, X, D} ;
2. Conflit = {R5, R8, R9}, on applique R5 puis R8 puis R9, BF = {B, C, X, D, E, A} ;
3. Conflit = {R1, R6}, on applique R1 puis R6, BF = {B, C, X, D, E, A, F, H} ;

Le fait **H** est établi, donc on s'arrête.

4.6.2 Moteur d'inférence à chaînage arrière

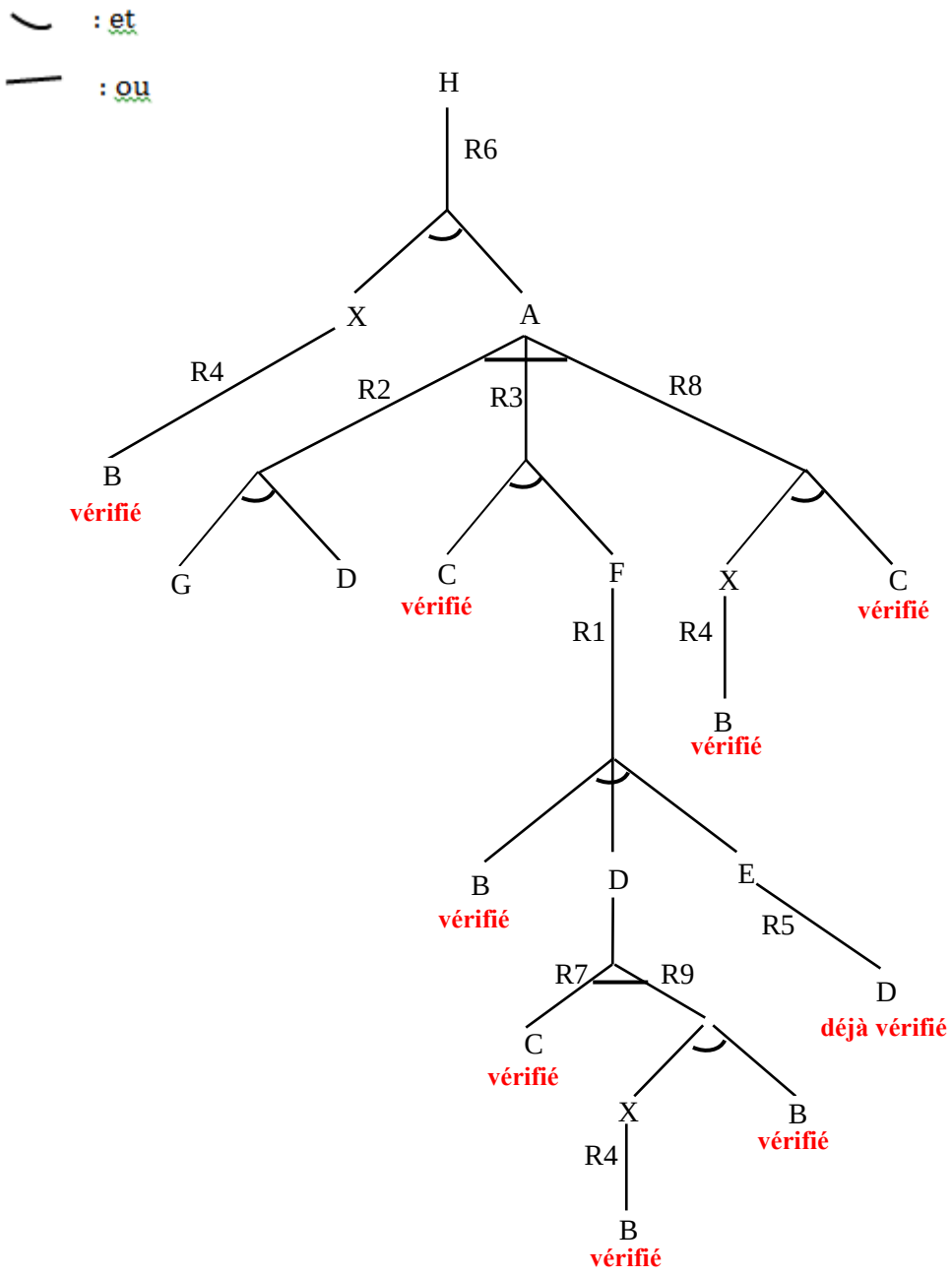
Un moteur d'inférence qui suit un chaînage arrière (backward chaining), part du but et essaie de 'remonter' aux faits pour le prouver (parcours guidé par le but).

- ✓ En effet, le moteur sélectionne les règles dont la partie 'action' (ou conclusion) correspond au but recherché. Les prémisses/conditions (partie 'situation') de ces règles deviennent elles aussi des sous buts à prouver et ainsi de suite ;

- ✓ Cet enchaînement s'arrête lorsque tous les sous buts sont prouvés- le but est alors lui aussi prouvé - ou lorsqu'il n'est plus possible de sélectionner des règles.

Le chaînage arrière nécessite un régime de contrôle par tentatives. Le moteur d'inférence opère alors un retour arrière (backtracking) pour remettre en cause l'application d'une règle qui débouche sur un échec et pour essayer une règle écartée précédemment.

Exemple 4.6.2. Nous cherchons à déduire le but H de la base de connaissance de l'exemple précédent en utilisant le chainage arrière.



4.6.3 Moteur d'inférence à chaînage mixte

Le chainage mixte est une combinaison du chaînage avant et du chaînage arrière. Un moteur qui suit ce mode opère généralement comme suit :

- ✓ Tant que des règles sont applicables (conditions satisfaites), il procède par chaînage avant jusqu'à satisfaction du but ou jusqu'à saturation de la base de faits ;
- ✓ Si le but n'est toujours pas prouvé, il choisit alors une règle dont la partie conclusion correspond au but et essaie de prouver ses prémisses inconnues par chaînage arrière en veillant à modifier la base de faits à chaque nouveau sous but prouvé;
- ✓ Si le but n'est toujours pas prouvé, il repart en chaînage avant en tenant compte des nouveaux faits prouvés et éventuellement en demandant de nouveaux faits à l'utilisateur et ainsi de suite jusqu'à ce que le but soit prouvé ou jusqu'à ce qu'aucune règle ne soit applicable.

4.7 Problèmes liés aux systèmes experts

- Les connaissances ne sont pas toujours disponibles ;
- Difficulté d'extraire l'expertise d'un être humain d'une manière certaine et précise ;
- Domaines d'utilisation restreints ;
- Difficultés d'obtenir des vérifications indépendantes aux solutions proposées ;
- Le vocabulaire et les termes techniques peuvent être difficile à transporter à un système expert ;
- Manque de confiance aux systèmes experts ;
- Parfois, les systèmes experts n'arrivent pas à des conclusions.

4.8 Conclusion

Les systèmes experts font partie des *systèmes à base de connaissance.* L'objectif de ces systèmes est de modéliser le savoir et le savoir-faire d'un expert (ou des experts) dans un domaine donné. Dans ce chapitre, nous avons présenté l'architecture interne d'un système expert, ainsi que son principe de fonctionnement et les caractéristiques du moteur d'inférence.

Dans le prochain chapitre, nous bordons une notion liée au raisonnement des systèmes experts qui est le traitement de l'incertitude ou le raisonnement incertain.

4.9 Exercices

Exercice N°1

Un polynôme de degré inférieur ou égal à 2, à coefficients entiers relatifs se décrit en mathématiques de la manière suivante :

$$ax^2 + bx + c \qquad o\grave{u}\ a, b, c \in Z$$

Mais l'écriture usuelle de ces polynômes est parfois très éloignée de cette description. Par exemple, si a = 1, b = −1 et c = 0, on écrira :

$$x^2 - x$$

Et non

$$1x^2 + -x + 0$$

Nous appliquons donc des règles identiques (ou plutôt ayant des effets identiques) de manière plus ou moins consciente. Formalisons ce problème afin qu'il puisse être traité par un générateur de système expert simple. Les données en entrée sont des variables à valeurs dans Z : a, b et c. Les sorties sont des variables :

$$sg_2, coef_2, var_2, expo_2, sg_1, coef_1, var_1, sg_0, coef_0$$

Prenant comme valeurs :

'+', '-' ou 'vide' pour sg_2, sg_1et sg_0

– 'x' ou 'vide' pour var_2et var_1

– '2' ou 'vide' pour $expo_2$

– un entier > 0 ou 'vide' pour $coef_2$, $coef_1$ et $coef_0$.

La valeur 'vide' d'une variable de sortie signifie que celle-ci ne doit pas être écrite.

Dans l'exemple précédent (a = 1, b = −1 et c = 0) :

sg_2= vide, $coef_2$= vide, var_2= x, $expo_2$= 2,

sg_1= -, $coef_1$= vide, var_1= x,

sg_0= vide et $coef_0$= vide.

Une fois obtenues les valeurs des variables de sortie, il est très facile (à la notation en exposant près) d'écrire un algorithme ou un programme qui affiche correctement le polynôme. La partie difficile, qui nécessite un expert humain, est donc celle qui consiste à passer des valeurs de a, b et c aux valeurs de sg_2, $coef_2$, etc. C'est évidemment la seule qui nous intéresse.

Question :

Ecrire ce système expert. Les règles doivent être de la forme :

Si a = 0 **Alors** (sg_2=vide et $coef_2$=vide et var_2=vide et $expo_2$=vide)

Tous les cas de figure doivent être pris en compte.

Exercice N°2

Soit la base de connaissance suivante :

Base des faits = {A, D, E, G}

Base des règles = {R1 : A, B, C→H

R2: A, U, C→F

R3: E, G, B→S

R4: D, G→C

R5: A, E → B

R6: U, S, T→ F

R7: G, H→ R

R8: D, E→T

R9: R, S, H→F

R10: A, U→B}

Question : établir le fait F en utilisant:

1. Un moteur d'inférence à chainage avant en profondeur d'abord monotone avec régime irrévocable ;
2. Un moteur d'inférence à chainage avant en largeur d'abord monotone avec régime irrévocable
3. Un moteur d'inférence à chainage arrière en profondeur d'abord monotone avec régime par tentative

Exercice N°3

Vous réaliserez un système expert avec un moteur d'inférence permettant de savoir si une ville mérite le voyage. La base des règles du système expert est la suivante :

1. Si ville historique Alors ville méritant le voyage

2. Si ville artistique Alors ville méritant le voyage

3. Si nombreuses animations Alors ville méritant le voyage

4. Si ville agréable et tradition gastronomique Alors ville méritant le voyage

5. Si belle ville et nombreux monuments Alors ville artistique

6. Si ville ancienne et nombreux monuments Alors ville historique

7. Si nombreux concerts et nombreux théâtres Alors nombreuses animations

8. Si activités sportives et traditions folkloriques Alors nombreuses animations

9. Si espaces verts et climat agréable Alors ville agréable

10. Si espaces verts et nombreux monuments Alors belle ville

11. Si nombreux restaurants et bons restaurants Alors tradition gastronomique

Question

Un système expert veut déduire « **si une ville mérite le voyage** » à travers un ensemble de questions posé à l'utilisateur. Au départ, le système ne sait pas les valeurs de vérité des faits, pour cela il pose un ensemble de questions pertinentes à l'utilisateur. Ce dernier peut répondre par : **Oui**, **Non**, **je ne sais pas**.

Supposons que l'ensemble de réponses de l'utilisateur sont les suivants :

Question 1 $\Rightarrow$ Réponse = Non

Question 2 $\Rightarrow$ Réponse = Non

Question 3 $\Rightarrow$ Réponse = je ne sais pas

Question 4 $\Rightarrow$ Réponse = oui

Question 5 $\Rightarrow$ Réponse = Non

Déduire l'ensemble de questions pertinentes posé par le système à l'utilisateur dans l'ordre induit par une exploration en profondeur d'abord.

A chaque fois que vous posez une question, donnez les faits qui ne sont plus déductibles et les questions qui ne sont plus pertinentes.

Quelle déduction le système peut faire concernant le fait « ville méritant le voyage » ?

4.10 Correction

Exercice N°1

Les règles de production du système expert sont les suivantes :

1. Si a = 0 Alors (sg2 = V ide, coef2 = V ide, var2 = V ide, expo2 = V ide)
2. Si a = 1 Alors (sg2 = V ide, coef2 = V ide, var2 = x, expo2 = 2)
3. Si a > 1 Alors (sg2 = V ide, coef2 = a, var2 = x, expo2 = 2)
4. Si a = −1 Alors (sg2 = −, coef2 = V ide, var2 = x, expo2 = 2)
5. Si a < −1 Alors (sg2 = −, coef2 = |a|, var2 = x, expo2 = 2)
6. Si b = 0 Alors (sg1 = V ide, coef1 = V ide, var1 = V ide)
7. Si (b = 1 et a = 0) Alors (sg1 = V ide, coef1 = V ide, var1 = x)
8. Si (b = 1 et a ≠ 0) Alors (sg1 = +, coef1 = V ide, var1 = x)

9. Si (b > 1 et a = 0) Alors (sg1 = V ide, coef1 = b, var1 = x)
10. Si (b > 1 et a ≠ 0) Alors (sg1 = +, coef1 = b, var1 = x)
11. Si b = −1 Alors (sg1 = −, coef1 = V ide, var1 = x)
12. Si b < −1 Alors (sg1 = −, coef1 = |b|, var1 = x)

13. Si (c = 0 et |a| + |b| = 0) Alors (sg0 = V ide, coef0 = 0)

14. Si (c = 0 et |a| + |b| ≠ 0) Alors (sg0 = V ide, coef0 = V ide)
15. Si (c > 0 et |a| + |b| = 0) Alors (sg0 = V ide, coef0 = c)
16. Si (c > 0 et |a| + |b| ≠ 0) Alors (sg0 = +, coef0 = c)
17. Si c < 0 Alors (sg0 = −, coef0 = |c|)

Exercice N°2

Le problème est d'établir le fait **F**

Schéma 1 : Un moteur d'inférence à chainage avant en profondeur d'abord monotone avec régime irrévocable

1. BDF ={A, D, E, G}, On peut appliquer {R4, R5, R8}, On applique R4

⇒ BDF = {A, D, E, G, C}

2. On peut appliquer {~~R4~~, R5, R8}, R4 est désactivée car déjà appliquée. On peut considérer que conflit = {R5, R8}, on applique R5

⇒ BDF = {A, D, E, G, C, B}

3. Conflit = {R1, R3, R8}, on applique R1

⇒ BDF = {A, D, E, G, C, B, H}

4. Conflit = {R3, R7, R8}, on applique R3

⇒ BDF = {A, D, E, G, C, B, H, S}

5. Conflit = {R7, R8}, on applique R7

⇒ BDF = {A, D, E, G, C, B, H, S, R}

6. Conflit = {R8, R9}, on appliqué R8

⇒ BDF = {A, D, E, G, C, B, H, S, R, T}

7. Conflit = {R9}, on appliqué R9

⇒ BDF = {A, D, E, G, C, B, H, S, R, T, **F**}

⇒ **Succès**

2. Schéma 2 : Un moteur d'inférence à chainage avant en largeur d'abord monotone avec régime irrévocable

On utilise la même base de connaissance avec le même fait à établir

1. BDF ={A, D, E, G}, Conflit = {R4, R5, R8}, On applique R4, puis R5, ensuite R8

⇒ BDF = {A, D, E, G, C, B, T}

2. Conflit = {R1, R3}, on applique R1, puis R3

⇒ BDF = {A, D, E, G, C, B, T, H, S}

3. Conflit = {R7}, on applique R7

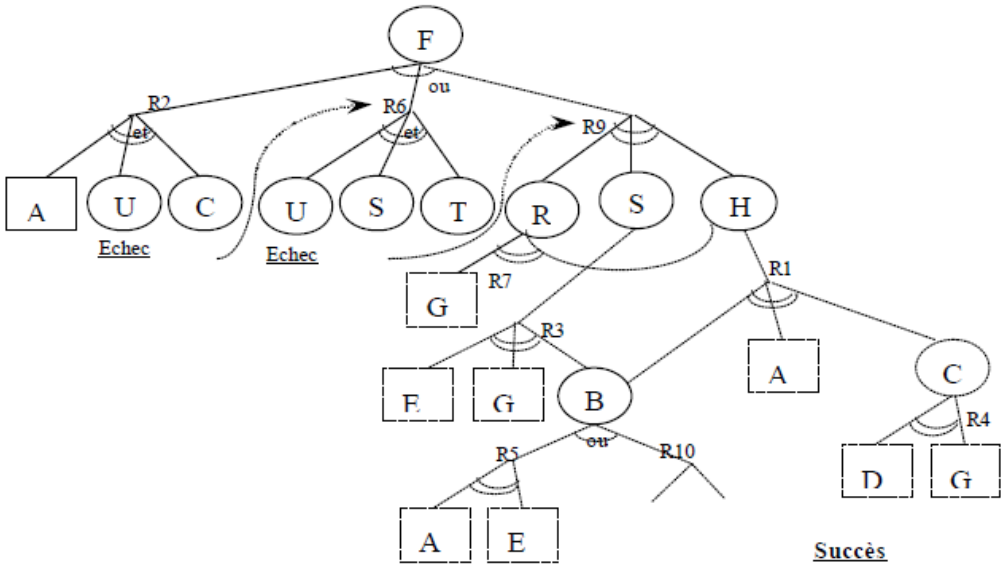
⇒ BDF = {A, D, E, G, C, B, T, H, S, R}

4. Conflit = {R9}, on applique R9

⇒ BDF = {A, D, E, G, C, B, T, H, S, R, F}

⇒**Succès**

3. Schéma 3 : Un moteur d'inférence à chainage arrière en profondeur d'abord monotone avec régime par tentative



Chapitre 5

Le Raisonnement Incertain

5.1 Introduction

La modélisation du raisonnement incertain est une nécessité souvent rencontrée dans les systèmes experts. Dans la logique classique, un fait peut être vrai ou faux, mais les connaissances humaines ne sont souvent pas aussi précises. Dans la pluparts des cas, les réponses des experts aux problèmes peuvent être incertaines et/ou inexactes. Une connaissance (faits ou règles) est incertaine si elle n'est ni vraie, ni fausse, mais seulement probable. De même, une connaissance est imprécise si elle est mal définie. Dans le domaine de la santé, par exemple, un diagnostic peut être fait sur un patient à partir de différentes observations. Plus, qu'il y a des éléments qui dirigent vers la même conclusion, plus on a confiance à ce diagnostic. Par contre, il est possible d'être dans l'erreur et/ou que d'autres observations contredisent les résultats. Donc, il peut arriver que nous soyons incertains de nos hypothèses et conclusions.

5.2 Les sources de l'incertitude

Il existe différentes sources de l'incertitude :

- La probabilité qu'un évènement se produise ou non ;
- La confiance d'un expert envers un fait et la difficulté à justifier cette confiance. De plus, la difficulté à combiner différents vues d'experts peut mener à des règles conflictuelles, d'où la nécessité de pondérer les règles de chaque expert ;
- Langage imprécis : langage naturel ambigu et présence de termes comme souvent, fréquemment, dès fois…etc ;
- Les exceptions aux règles qui peuvent être des sources d'incertitude ;
- La quantification de valeurs : par exemple, on peut quantifier la fièvre du corps humain par : « fièvre normale », « fièvre normale », « fièvre sévère » ;
- Données inconnues : cela mène à un raisonnement approximatif, tolérant la présence de valeurs inconnues ;
- Implications faibles : souvent difficile de corréler «fortement» les parties conditions et conclusions d'une règle.

5.3 L'incertitude dans les systèmes experts

Deux grandes approches de traitement de l'incertitude sont utilisées dans les systèmes experts. La première catégorie concerne les approches statistiques comme la théorie de bayes, les facteurs de certitude et la théorie de Dempster-Shafer. La deuxième comprend les approches liées aux ensembles floues.

5.3.1 Approche probabiliste et réseaux bayésiens

Cette approche est basée sur les probabilités mathématiques. Les réseaux bayésiens permettent de gérer l'incertitude dans les systèmes intelligents. Les réseaux bayésiens sont basés sur le théorème de bayes. Pour comprendre ces notions, il faut d'abord examiner de près la notion de probabilité conditionnelle.

5.3.1.1 La théorie de Bayes

La théorie probabiliste permet de déterminer la probabilité a posteriori qu'un évènement H se produise sachant que l'évènement E est réalisé, en fonction de sa probabilité a priori. La règle de Bayes permet la révision probabiliste, et est utilisée dans le cadre de probabilités conditionnelles binaires (malade / non malade sachant que je présente les symptômes). Elle nous permet de réviser nos jugements dès lors que nous obtenons une nouvelle information. Ceci équivaut à passer d'une probabilité *a priori* -précédent la prise en compte du nouvel élément- à une probabilité *a posteriori* -faisant suite à la prise de connaissance de la nouvelle information

Par exemple, si nous avons :

$P(E) =$ Probabilité à priori de E

$P(H) =$ Probabilité à priori de H

$P(H|E) =$ Probabilité à postériori de H sachant E

$P(E|H) =$ Probabilité à postériori de E sachant H

$P(E|H)P(H) = P(H \cap E) = P(H|E)P(E)$, d'où il est possible d'obtenir :

$$\frac{P(H|E)P(E)}{P(E)} = \frac{P(E|H)P(H)}{P(E)}$$

Ce qui en résulte la théorie de bayes :$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$

Exemple 5.3.1. 1. Nous empruntons à (Schalkoff, 2011) cet exemple :

Supposons que :

H : est l'évènement qu'un patient a un cancer du poumon ;

E : est l'évènement qu'un patient fume ;

Nous voulons calculer P(H|E) c'est-à-dire la probabilité qu'un patient a un cancer du poumon sachant qu'il fume, sachant que :

P(H) = 5% est la probabilité d'avoir un cancer du poumon dans la population en général ;

P(E) = 10% est la probabilité qu'une personne fume ;

Nous pouvons aussi avoir l'équation :

P(patient est un fumeur| patient a un cancer du poumon), la probabilité dans cette équation peut être obtenue à partir de l'expérience. Par exemple, nous pouvons avoir le pourcentage des personnes atteintes d'un cancer du poumon qui sont fumeur. Supposons que P(E|H)= 75%.

Donc, dans le cas où on a l'observation que le patient est un fumeur, nous pouvons calculer la probabilité d'avoir un cancer du poumon sachant qu'il est fumeur avec le théorème de Bayes comme suit :

$$P(H|E) = \frac{P(E|H)P(H)}{P(H)} = \frac{75\% * 5\%}{10\%}$$

$$P(H|E) = 37{,}5\%$$

Donc, nous pouvons conclure que :

P(H|E)=P(patient a un cancer du poumon| patient fume)= 37,5%.

5.3.1.2 Les réseaux bayésiens

Un réseau bayésien est un modèle probabiliste qui représente un ensemble de variables avec leurs dépendances conditionnelles. Il sert comme outil pour représenter les connaissances incertaines et les relations de causalités. Dans un réseau bayésien, on peut calculer la probabilité des actions non observées en se basant sur certaines

observations. Par exemple, si nous utilisons un réseau bayésien dans le diagnostic d'un patient et que nous connaissons ses symptômes, nous pouvons calculer la probabilité de chacune des maladies qui en découlent.

Définition 5.3.1.2. Un réseau bayésien $\mathcal{B}(\mathcal{G}, \mathcal{O})$ est défini par :

- $\mathcal{G}(X, E)$ graphe orienté sans cycle dont les sommets sont associés à un ensemble de variables aléatoires $X = \{X_1, X_2, \dots, X_n\}$;
- $\mathcal{O}\{P(X_i|Pa(X_i))\}$, ensemble de probabilités de chaque nœud X_i conditionnellement à l'état de ses parents $Pa(X_i)$ dans $\mathcal{G}$.

La représentation graphique du réseau bayésien indique les dépendances (ou indépendances) entre ses variables. Un exemple de cette représentation est donné sur la figure 5.1.

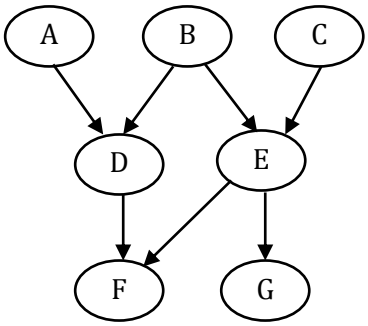


Figure 5.1 : Un graphe orienté sans cycle

D'après le graphe de la figure 5.1, les parents de F sont D et E, les ancêtres de F sont D, E, A, B, et C. les fils de B sont D et E. les descendants de A sont D et F. les non descendants de A sont B, C, E et G.

Dans un réseau bayésien, tout nœud est conditionnellement indépendants de ses non-descendants, sachant ses parents. Formellement :

$$P(X_i) = (X_i|Pa(X_i))$$

Si on tient compte de la structure du graphe, la formule précédente peut se mettre sous la forme générale équivalente :

$$P(X_i) = \prod_{i=1}^{n} P(X_i|Pa(X_i))$$

Exemple 5.3.1.2. Nous empruntons à (Dennis, 2006) l'exemple de la figure 5.2. Le réseau de cette figure traduit le comportement d'un robot dont l'univers est représenté par quatre paramètres.

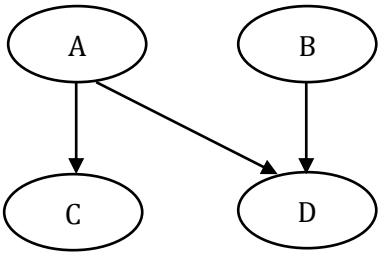


Figure 5.2 : le réseau bayésien exprimant le comportement d'un robot

A : la batterie est chargée

B : le bloc à soulever n'a pas d'autre bloc posé sur lui (il est libre)

D : le bras du robot rouge

C : la jauge de la batterie indique un chargement complet de celle-ci

Les nœuds A et B sont sans ancêtres, donc les probabilités P(A) et P(B) sont non conditionnelles. Les variables A et B sont indépendants pour la même raison. Pour le nœud C, il faut savoir P(C|A) et P(C|¬A). Pour le nœud D, il faut donner les quatre probabilités : P(D|A, B), P(D|A, ¬B), P(D|¬A, B), P(D|¬A, ¬B). Les valeurs de probabilités sont données dans le tableau suivant :

P(A)	**0.95**
P(B)	0.7
P(C|A)	0.95
P(C|¬A)	0.1
P(D|A, B)	0.9
P(D|A, ¬B)	0.05
P(D|¬A, B)	0
P(D|¬A, ¬B)	0

Ces valeurs sont données arbitrairement. Par exemple, P(C|A) =0.95 signifie qu'il y a 95% de chances que la jauge soit sur "complet" quand on sait que la batterie est chargée, et P(G | ¬B) = 0.1 signifie qu'il y a 10% de chances que la jauge soit sur "complet" quand on est sait que la batterie est déchargée.

Si nous cherchons à calculer d'autres probabilités conditionnelles, par exemple, comment notre réseau bayésien peut-il répondre aux questions suivantes :

- Quelle est la probabilité pour que le bras du robot se déplace sachant que le bloc est libre ?
- Quelle est la probabilité que le bloc ne soit pas libre sachant que la batterie est vide et que le bras ne bouge pas?

Formellement, il s'agit de calculer les probabilités conditionnelles P(D|B), P(¬B|¬A¬D).
L'inconvénient majeur de cette approche est qu'il faut savoir les probabilités de chacune des évidences (les observations) du réseau, ce qui est impossible dans plusieurs domaines d'application. Les approches probabilistes se basent sur rigoureusement sur les mathématiques. Par contre, la connaissance s'exprime sous forme de langage naturel, où il est difficile de faire une correspondance directe en probabilités.

5.3.2 Facteurs de confiance

Un des problèmes rencontrés lors de la modélisation du savoir-faire d'un expert est la prise en compte des connaissances incertaines.
De nombreux générateurs de systèmes experts offrent la possibilité de nuancer leur certitude concernant un fait en leur associant un degré d'incertitude ou de confiance.
Par exemple, pour la question : « Avez-vous très mal à la tête ? Ou pas tellement », on peut avoir la réponse : « Sur une échelle qui associerait 0 au fait de ne pas avoir mal à la tête et 1 au fait de ne pas pouvoir le supporter, je dirai que j'ai mal à la tête avec un coefficient 0,45 ».
Cet exemple relève un des problèmes de cette méthode. Un utilisateur, ou même un expert ne pourrait jamais exprimer avec précision de tels facteurs de confiance.

Cette notion est apparue la première fois avec MYCIN. Ce dernier utilisait un facteur de confiance allant de -1 à 1. Le facteur -1 indique un manque de confiance total, alors que le 1 indique un facteur de confiance absolue en l'affirmation ou le fait qu'elle soit vraie. De façon générale, les facteurs de confiance permettent d'associer à un fait un degré de confiance. Plus le facteur est élevé, plus nous croyons que l'affirmation est vraie et inversement.

En plus d'associer des mesures de confiance aux faits, la combinaison de ces degrés est un problème à résoudre. Si les faits A et B sont connus respectivement dans la base par les facteurs de confiance p et q, et si la règle :

$$A\ et\ B \rightarrow C$$

figure dans la base des règles, que peut-on déduire du facteur de certitude de C ?

Exemple 5.3.2. Supposons que nous avons la règle R1 qui possède le facteur de confiance CF_{R1} (CF_{R1}= 0.9) :

$$R1: A\ et\ B \rightarrow C \quad (CF_{R1}= 0.9)$$

Supposons que A et B sont deux faits avec les facteurs de confaince CF_A (CF_A =0.8) et CF_B (CF_B= 0.6) et que lorsqu'une règle de certitude CF est déclenchée par des prémisses de certitudes($CF_{1..,}CF_n$), sa conclusion prend la certitude :

$$CF * Mini_{i=1}^{i=n} CF_i:$$

En appliquant cette règle de propagation de certitude, le degré de confiance associé à la conclusion C de la règle R1 est :

$$CF_C = 0.9 * Min(0.8, 0.6) = 0.54$$

Donc la conclusion C de la règle R1 a un degré de confiance CF_C=0.54.

5.3.3 La logique multivalente

La logique classique est une logique à deux valeurs, les énoncés sont vrais ou faux. Par contre, dans la logique polyvalente (ou multivalente ou multivaluée), les énoncés peuvent prendre une autre valeur que le vari et le faux. Dans ce type de logique, on peut avoir un degré d'appartenance dans l'ensemble vari ou l'ensemble faux. Par exemple, on peut trouver la logique à 3 valeurs comme les logiques de kleene, de Priest, et de Lukasiewicz (Nault, 2012). Ces logique, consiste à ajouter une troisième valeur possible. C'est-à-dire, qu'un énoncé peut être vrai ou faux ou indéterminé. Nous retrouvons la logique à valeurs finies (à plus de trois valeurs), comme la logique quaternaire (4 valeurs). Finalement, il y a les logiques à valeurs infinies comme la logique floue. Dans ce dernier type de logique, la valeur de vérité d'une information est un nombre réel entre 0 et 1 (voir chapitre 2).

5.3.4 La théorie de Dempster-Shafer

La théorie de Dempster, complétée par les propositions de Shafer, a donné naissance à un modèle mathématique connue sous le nom de la théorie de l'évidence. Cette théorie est fondée sur la modélisation de l'incertitude par un intervalle de probabilité plutôt qu'une simple probabilité numérique. Elle utilise deux notions : la notion de *l'incertitude*

et la notion de *l'ignorance.* La théorie de Dempster-Shafer associe à une proposition donnée l'intervalle : [Confiance, Plausibilité]. Cet intervalle est compris entre 0 et 1. La plausibilité est définie par : $Pl(S) = 1 - conf(\bar{S})$.

Cet intervalle permet de mesurer le niveau de confiance de certaines propositions, mais également la quantité d'information disponible. Supposons que nous ayons trois hypothèses A, B et C . Si aucune information n'est disponible on associe à chacune l'intervalle [0,1]. Noter bien que cela est différente de l'approche bayésienne qui aurait affecté à chaque hypothèse la valeur 0.33.

5.4 Conclusion

Le raisonnement incertain est l'un des caractéristiques du raisonnement humain. De ce fait, le raisonnement incertain est l'un des problèmes les plus rencontrés dans les systèmes experts. Dans ce chapitre, nous avons introduit la notion de l'incertitude dans les systèmes experts. Deux approches qui traitent cette notion ont été présentées, à savoir, les approches statistiques et les approches liées aux ensembles floues.

Dans le dernier chapitre, nous bordons le processus de développement des systèmes experts.

5.5 Exercices

Exercice N°1

Une banque utilise un système expert pour accorder un prêt. Les variables suivantes sont employées pour décrire les propositions associées :

– OK : le prêt est accordé
– CO : le conjoint se porte garant
– PA : le candidat au prêt peut payer ses traites
– RE : le dossier du candidat est bon
– AP : les revenus du conjoint sont élevés
– RA : le taux d'intérêt est faible
– IN : les revenus du candidat sont supérieurs à ses dépenses
– BA : le candidat n'a jamais de découvert sur son compte courant
– MB : le conjoint doit hériter

Les règles sont les suivantes (la colonne 3 sera utile plus tard) :

1	Si MB alors CO	0,8
2	Si AP alors CO	0,7
3	Si RA alors RE	0,8
4	Si IN alors PA	0,8
5	Si BA, RE alors OK	0,9
6	Si CO, PA, RE alors OK	1

Question1 : Soit un moteur d'inférence fonctionnant en chaînage arrière et profondeur d'abord. Donner son graphe ET/OU complet pour la base de faits initiale {BA, RA, MB, AP, IN} avec OK comme but à établir.

Question 2 : On associe à chaque règle le "coefficient de certitude" CA de la colonne 3. Il est compris entre 0 et 1. Par exemple, la règle 5 s'interprète maintenant comme suit :
"Si BA et RE sont certains, Alors OK a une certitude de 0.9".

Chaque fait initial possède lui-même un coefficient de certitude CF entre 0 et 1 : un client qui fait la demande d'un prêt est donc décrit par un vecteur de 5 valeurs entre 0 et 1.

La propagation de la certitude dans le moteur d'inférence se fait ainsi :

– Lorsqu'une règle de certitude CA est déclenchée par des prémisses de certitudes($CF_{1..,}CF_n$), sa conclusion prend la certitude $CA * Mini_{i=1}^{i=n} CF_i$

– Si un fait a déjà une certitude CF1 et qu'il est inféré à nouveau avec la certitude CF_2 on lui attribue la certitude $CF_1 + CF_2 - CF_1 * CF_2$

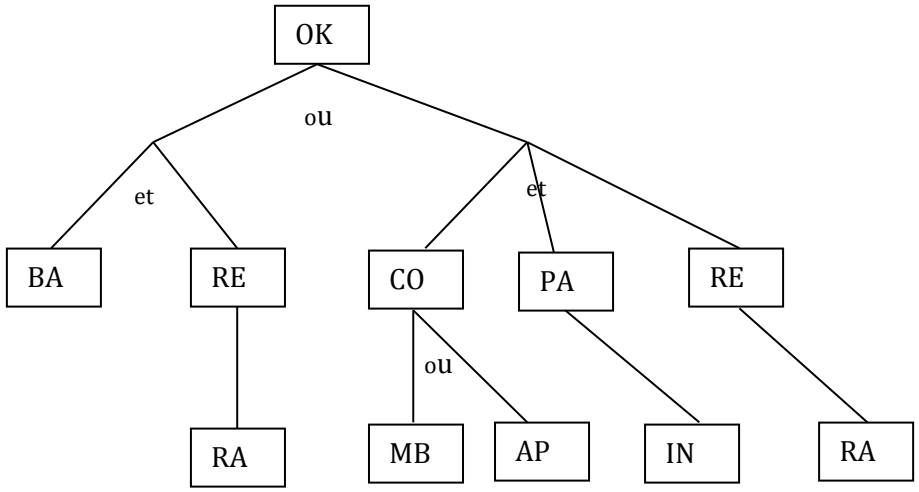
Quel est la certitude d'attribuer un prêt au client décrit par la base de faits initiale suivante?

BA	RA	MB	AP	IN
0.9	0.4	0.9	0.2	0.8

5.6 Correction

Exercice N°1

1. Le graphe ET/OU complet :



2. Calcul de certitude :

La certitude du nœud RE est **0.32** (0.8*0.4= 0.32)

La certitude du nœud PA est **0.64** (0.8*0.8= 0.64)

La certitude du nœud CO est **0.76** (il a deux fils OU, donc ((0.8*0.9)+(0.7*0.2))-((0.8*0.9)*(0.7*0.2))= 0.572)

La certitude qui vient du fils gauche de OK est: 0.9*min(0.32, 0.9)= **0.288**

La certitude qui vient du fils droit de OK est : 1*min(0.76, 0.64, 0.32)= **0.32**

Donc :

La certitude du nœud OK est 0.516 (0.288 + 0.32 – 0.288*0.32= 0.516)

Chapitre 6

Méthodologie de Construction des Systèmes Experts

6.1 Introduction

Pour résoudre un problème, l'être humain fait appel à ses connaissances pour faire des raisonnements. L'expert humains acquiert et développe sa connaissance au cours de son travail en percevant de l'information. Au fur et à mesure, ces informations se structurent et s'organisent en connaissances sans effort de la part de l'expert.

Faire reproduire le comportement de raisonnement humain par un programme est une tâche lente et délicate qui nécessite de passer par des phases d'extraction, de modélisation et de formalisation de la connaissance. Chacune de ses phases devra faire appel à des méthodes et outils adéquats. L'acquisition de la connaissance est certainement la tâche la plus complexe et la moins formelle.

6.2 Les problèmes adaptés aux systèmes experts

Les chercheurs ont défini un ensemble informel de critères pour déterminer si un problème est adapté ou non à un système expert :

- Le champ d'application des systèmes experts est vaste : ils permettent par exemple d'aborder des problèmes pour lesquels les données sont imprécises, voire incomplètes ;
- Le problème est de nature symbolique, faisant intervenir des heuristiques et ne nécessitant pas des méthodes numériques classiques ;
- Les connaissances mises en œuvre sont de nature qualitative que quantitative ;
- Une solution algorithmique connue possible n'existe pas ;
- L'utilisation du système expert doit être justifiée :
  - L'évolution rapide des connaissances qui nécessite des mises à jour fréquentes ;
  - La mémorisation du savoir-faire accumulé par des personnes travaillant sur un problème donné ;
  - La grande taille et la complexité de l'information à manipulé.

6.3 Historique

Dans cette section nous présentons une liste non exhaustive de quelques systèmes experts bien connus dans différents domaines.

- **DENDRAL (1965, chimie) :** est le premier système expert permettant d'avoir un raisonnement dans le domaine de la chimie.
- **MYCIN (1972-1974, Medecine)** : est dédié au diagnostic des maladies infectieuses du sang (et des méningites dans une version ultérieure) et à la prescription médicale. C'est le premier à bien séparer le moteur d'inférence de la base de connaissances et à pouvoir expliquer ses raisonnements.
- **PROSPECTOR (1976, Géologie)** : utilise la logique floue ou probabilité avec un raisonnement bayésien pour l'évaluation de la richesse en minerais d'un site géologique.
- **XCON (1978, Informatique)** : est développé par Digital Equipment Corporation (DEC) pour la configuration d'ordinateurs VAX-11 à partir de spécifications client.
- **TAXMAN (1977, industrie)** : développé par l'équipe de MacCarthy. TAXMAN est un système expert capable de conseiller une entreprise sur la meilleure façon de payer le moins de taxes possible. Depuis, le système a été repris et étendu par diverses entreprises.
- **LPS (1979, Mathématique)** : système expert de résolution des problèmes de géométrie.
- **DELTA (1980, industrie):** développé pour le diagnostic des pannes de locomotives.
- **DART (1981, Informatique) :** développé pour le diagnostic des pannes d'ordinateurs et de systèmes

6.4 Le développement d'un système expert

6.4.1 Etapes de développement d'un système expert

Le développement des systèmes informatiques suit une démarche planifiée et fait appel à des méthodes et des outils de travail appropriés. De même, le développement des systèmes experts fait recours à des méthodes déjà éprouvées dans d'autres domaines d'informatique. Toutefois, la particularité inhérente aux systèmes experts et aux

systèmes à base de connaissances d'une manière générale nous obligent à employer des méthodes spécifiques de l'ingénierie de connaissance[2].

La figure 6.1 montre les principales tâches prises en charge par l'ingénierie des connaissances à celles d'un système conventionnel. Cette démarche fait place à l'analyse, à la conception, à la programmation, aux tests et à l'implantation. Ces tâches. Ce sont des tâches couramment exécutées lors du développement de n'importe quel système informatique. La principale différence dans le processus de développement des systèmes à base de connaissance réside dans l'importance accordée aux connaissances. Elles font l'objet de trois étapes : l'identification, l'acquisition et la représentation.

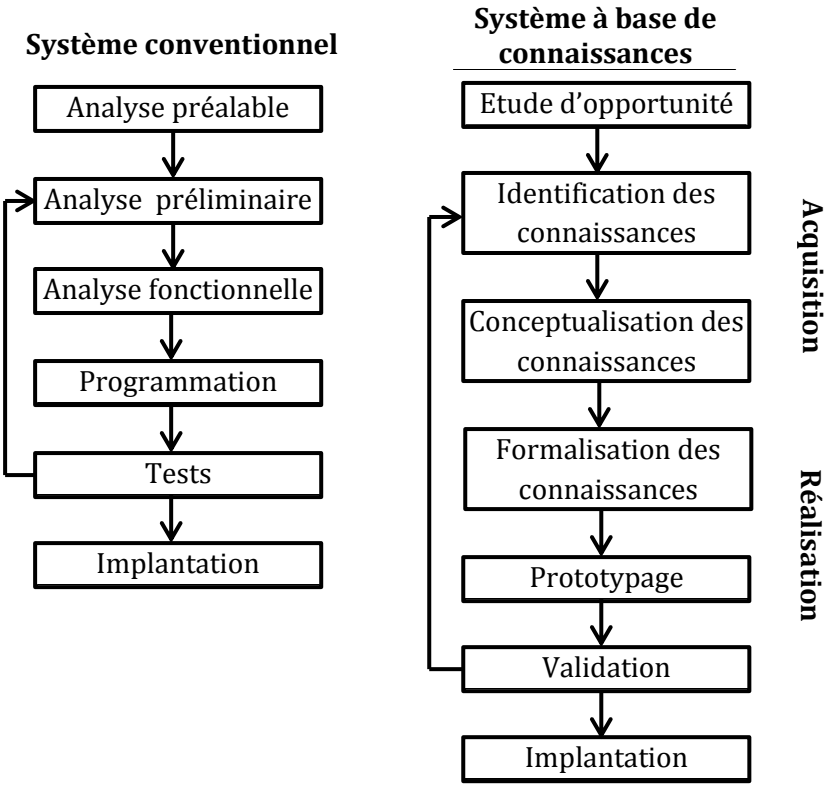


Figure 6.1 : Etapes de développement des systèmes informatiques (Paquette & Roy, 1990).

- **Etude de l'opportunité** : étudie le domaine de connaissance à modéliser, les tâches effectuées par les experts pour évaluer la pertinence et le rapport coûts-bénéfices d'envisager une solution informatique ;

[2] L'ingénierie de connaissances étudie le processus de développement des systèmes à base de connaissances. Elle évoque des techniques pour manipuler des connaissances sur ordinateur.

- **Identification des connaissances** : définit le type de données et solutions produites par l'expert. Ensuite, identifié les tâches que devrait effectuer le système expert pour imiter le comportement d'un expert ;
- **Acquisition des connaissances** : recueille les connaissances nécessaires à la réalisation du système expert auprès des experts ;
- **Représentation des connaissances** : c'est l'étape de la formalisation des connaissances pour les traduire en règles de production ou autre formalisme choisie par le système expert ;
- **Prototypage** : c'est l'implémentation progressive du système expert et la conception de l'interface utilisateur ;
- **Validation** : vérifie les résultats fournis par le système auprès des experts, des utilisateurs et du responsable du projet ;
- **Implantation :** c'est la mise en marche du système dans son contexte et l'initiation des utilisateurs à son exploitation.

6.4.2 Les intervenants et leurs rôles

Intervenants est un terme générique désigne les personnes impliquées dans le développement d'un système informatique. La réalisation d'un système expert repose essentiellement sur trois acteurs : l'expert, le cogniticien et l'informaticien.

- **L'expert du domaine** : c'est la personne qui a la plus grande expérience ou connaissance dans un domaine particulier et qui est capable et habile dans la résolution des problèmes relatifs à son domaine d'expertise, ainsi l'expérience et les connaissances qu'il possède constitueront la base sur laquelle le système sera développé ;
- **Le cogniticien** : ou l'ingénieur de la connaissance. Le cogniticien recueille et analyse la connaissance fournie par l'expert ;
- **L'informaticien** : ou le programmeur choisit l'outil de développement et structure les données pour les entrer en machine.

En plus de ces trois acteurs, on trouve le responsable de projet et les utilisateurs. La figure 6.2 illustre les échanges entre les intervenants lors du développement d'un système expert :

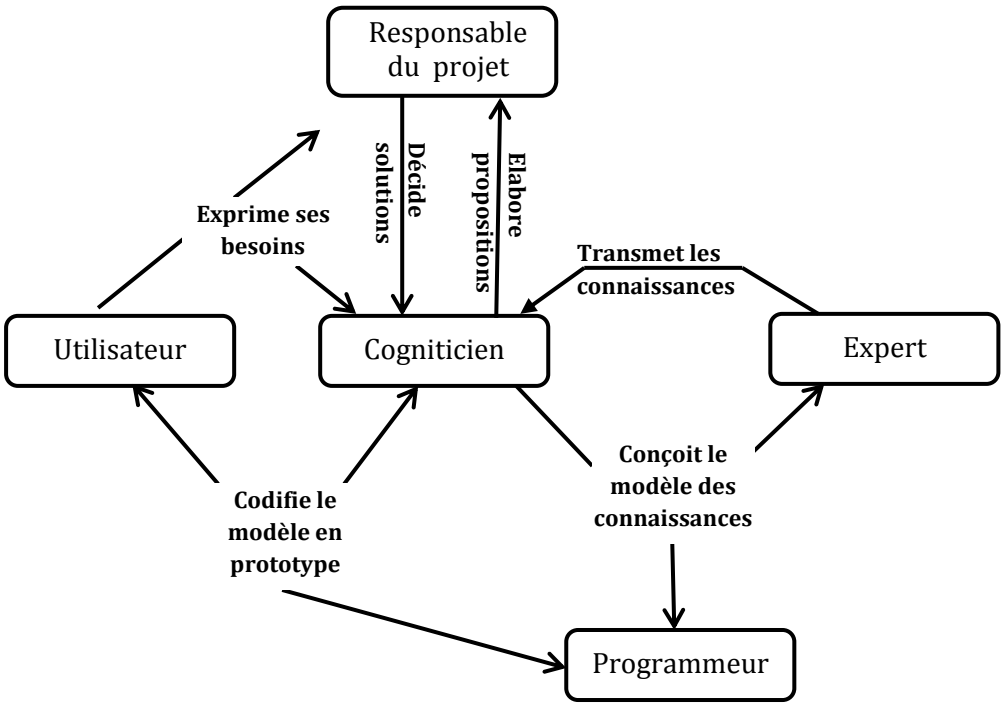


Figure 6.2 : Les échanges entre les intervenants dans le développement du système expert (Paquette & Roy, 1990).

6.4.3 Acquisition de connaissance

Selon Feigenbaum, la connaissance est le facteur clé de la performance d'un système expert. L'acquisition de connaissance est l'une des étapes importantes pour le développement des systèmes experts. C'est l'étape qui aide le cogniticien à créer sa base de connaissance.

L'Acquisition des connaissances, pour la construction des systèmes experts de première génération, était essentiellement considérée comme une activité de transfert, consistant à traduire le discours de l'expert dans le formalisme de représentation choisi.

Les méthodes de transfert de connaissances dans les premiers systèmes experts étaient totalement empiriques. Puis elles se sont précisées, structurées (méthodes analytiques), certaines s'automatisent (méthodes automatique). Dans ce qui suit, nous présentons les deux premières méthodes.

- **La méthode empirique** : c'est une méthode dirigée par l'implémentation, en collaboration avec l'expert le cogniticien développe une maquette du système réel. Les cogniticiens procédaient, de façon itérative, par prototypage rapide, en entrant dans le cycle « extraction de la connaissance, modélisation de la connaissance

extraite, validation de la base de connaissance », jusqu'à ce que la connaissance soit correcte.

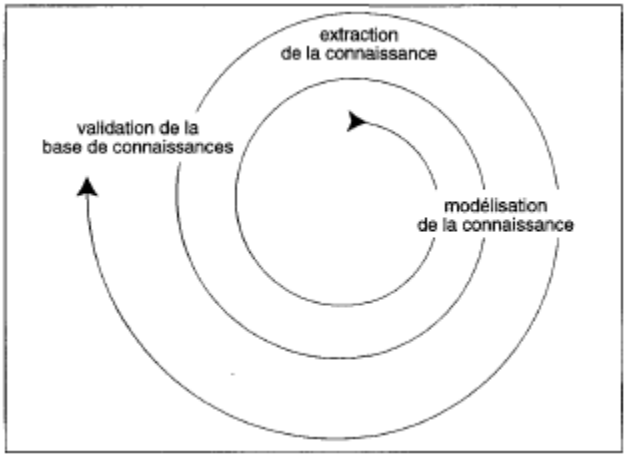


Figure 6.3 : La méthode empirique d'acquisition des connaissances (Dubus, 1995)

- **Les méthodes structurées (ou analytiques)** : c'est une approche dirigée par les modèles, plus structurés que l'approche empirique. Le problème est défini, puis l'expertise recueillie est modélisée avant de concevoir le système expert. Les méthodes analytiques d'acquisition de connaissances nécessitent la présence d'un cogniticien.

Plusieurs méthodologies ont été mises en place pour l'extraction de connaissances. La plupart de ces méthodologies proviennent des recherches effectuées en ingénierie de connaissance. Parmi ces méthodes, KADS et KOD sont suffisamment générales pour pouvoir intervenir dans n'importe quel type de domaine, de prendre en compte le processus d'acquisition de la connaissance, et de proposer chacune un modèle de la connaissance experte.

- **KADS (Knowledge Acquisition and Design Structuring)**: Une analyse complète des données précède la conception et l'implantation du système expert, ce qui diffère du prototypage rapide. Dans la méthodologie KADS, le processus de réalisation d'un système expert est vu comme une activité, c'est-à-dire une succession de transformation de modèles (figure 5.4). Un atelier logiciel guidant la démarche du cogniticien accompagne cette méthode, à travers :
  - ✓ un module de conseil pour l'utilisation de la méthodologie ;
  - ✓ des modules pour structurer et analyser les interviews ;
  - ✓ la conceptualisation des données en entités et relations ;

- ✓ La proposition de plusieurs modèles d'interprétation.

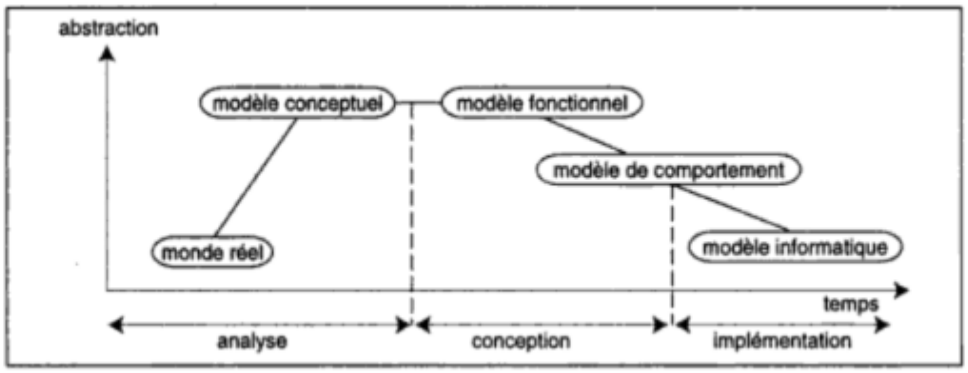


Figure 6.4 : KADS : le cycle de vie d'un système expert (Dubus, 1995)

- **KOD (Knowledge Oriented Design)** : la méthodologie KOD utilise des techniques issues de la linguistique. Elle repose sur trois modèles de connaissances (le modèle pratique, le modèle cognitif, et le modèle informatique), et sur trois paradigmes (représentation, action, interprétation). Cette méthode est assortie d'un logiciel fournissant des outils informatiques garantissant la mise en œuvre de ses différentes étapes, indépendamment de l'implémentation.

6.4.4 Mode de représentation de connaissance

La qualité de représentation de connaissance est déterminante pour la performance d'un système expert. Pour assurer une bonne représentation de la connaissance dans un système expert, il faut prendre en considération les points suivant :

- **Extensibilité** : les modèles envisagés doivent être souples pour permettre une modification aisée du système ;
- **Simplicité** : les concepts retenus ne doivent pas être trop complexes, de manière à ce que la transmission du savoir au système soit facile ;
- **Caractère explicité** : la connaissance doit être explicite, c'est le facteur important dans la recherche des erreurs et la justification du système.

6.4.5 Les langages spécialisés

Un système expert peut être réalisé en se basant sur un langage tel que Pascal ou C, mais on choisira plus facilement Lisp, Prolog, ou un lange orienté objet (C++, Smaltalk…etc). En effet, ces langages sont mieux adaptés au raisonnement des systèmes experts que les

langages classiques qui sont plus appropriés à une programmation procédurale et séquentielle.

6.4.6 Systèmes experts généraux

Un système expert général est obtenu par l'union d'un moteur d'inférence et d'une interface. On l'appelle aussi coquille, shell ou encore un système expert vide. C'est un système expert dont la base de connaissance est à remplir. Ces systèmes sont appelés aussi : les générateurs de systèmes experts. Ce sont des environnements complets et généraux qui intègrent un moteur d'inférence (indépendant de la base de connaissance) ainsi que divers modules d'aide au développement tels qu'un éditeur de connaissances pour le remplissage de la base, un tableur, un gestionnaire de base de données…etc.
Pour construire un nouveau système expert, il faut se baser sur la construction de la base de connaissance en respectant le formalisme et la syntaxe de la représentation de connaissances allant avec ce système. Le rajout d'une base de connaissance au système shell est appelé : instanciation su système expert.
L'un des outils les plus utilisés est CLIPS (C Language Integrated Production System). (CLIPS'94). Il a été développé en 1985 et fournit un environnement de développement de système expert complet. Il permet de représenter les connaissances sous forme de règles ou objets ou encore une représentation procédurale. Un autre outil de même genre que CLIPS est ART*Entreprise*. Ce qui le distingue de CLIPS est qu'il permet d'utiliser le raisonnement à base de cas. De même, Jess est un environnement de développement des systèmes experts adapté de CLIPS pour l'environnement Java. Drools est un générateur de systèmes experts écrit en java et qui a vu le jour en 2001. eGanges est un outil très léger (moins de 0.25 Mo) qui fournit une interface graphique présentant les différentes règles, procédures et stratégies sous forme de carte interactive. Aussi parmi les générateurs de systèmes experts les plus connu, le NEXPERT. C'est un logiciel crée par Neuron Data. Les connaissances sont représentées sous forme de règles de production et son moteur d'inférence fonctionne en chainage mixte.
D'autres générateurs de systèmes experts existent comme : DIAGNEX, GURU, N-, PC+, SNARC, …etc.

6.5 Evolution des systèmes experts

La convivialité et l'efficacité d'un système expert dépend fortement des modules, des fonctions, interfaces, logiciels qui tournent autour du système. La principale difficulté

rencontrée lors du développement d'un système expert se trouve au niveau de l'acquisition des connaissances qui est toujours une tâche longue et fastidieuse. Cette tâche met en collaboration deux personnes l'expert et le cogniticien.

Au début, les systèmes experts n'avaient pour objet que la résolution automatique des problèmes. Les systèmes à base de connaissances qui leur ont succédé avaient comme objectif de créer une coopération entre le système et l'utilisateur à travers le dialogue, au lieu de restituer une solution au problème par manipulation aveugle da la connaissance. Dans ce cas, le système doit avoir accès non seulement aux termes utilisés par l'être humain, mais également à la sémantique que ce dernier associe aux différents termes pour une meilleure communication (Frust, 2002). Pour modéliser la richesse sémantique des connaissances, l'emploie des règles de production par les systèmes experts, ne suffit plus. De nouveaux formalismes ont été introduits. Ces formalismes sont basés sur l'utilisation des ontologies. Les langages à base de frame, les logiques de descriptions, les graphes conceptuels sont des exemples de tels formalismes.

6.6 Conclusion

Les systèmes experts sont moins à la mode en 2019 qu'en 1974. L'idée de ces systèmes est apparue au début des années 60 avec les premiers pas de l'intelligence artificielle et le premier système experts (DENDRAL). Depuis les années 1990, nous parlons beaucoup plus de « Systèmes Interactifs d'Aide à la Décision », une façon de réintégrer l'être humain dans le processus de prise de décision de l'ordinateur. Enfin, nous pouvons dire que les systèmes experts peuvent apporter un plus dans tous les domaines où la prise de décision nécessite des connaissances volumineuses.

Conclusion générale

Les systèmes experts constituent l'un des domaines de l'intelligence artificielle qui ont trouvé des applications dans le monde industriel. La notion des systèmes experts est assez ancienne qui est apparue dans les années 70 avec l'apparition du système experts célèbre MYCIN. Les systèmes experts ont comme objectif la modélisation puis la simulation du savoir et du savoir-faire d'un expert (ou des experts) dans un domine donné. Un système expert est composé de deux parties complémentaires : une base de connaissance et un moteur d'inférence, et il se caractérise par la séparation entre les connaissances et le raisonnement.

Au début, les systèmes experts avaient comme objectif « le remplacement de l'expert », mais à la fin des années 80, ils avaient du mal à tenir leur promesse (remplacer les experts). La principale difficulté rencontrée lors du développement d'un système expert se trouve au niveau de l'acquisition de connaissances qui est toujours une phase longue et fastidieuse et qui met en collaboration deux personnes l'expert et le cogniticien. De plus, contrairement aux technologies récentes de L'IA, les systèmes experts ne permettent pas l'apprentissage ce qui limite leurs portées

Depuis les années 90, nous ne parlons pas de systèmes experts qui peuvent remplacer les experts humains, mais nous assistons à un nouvel essor de l'intelligence artificielle et une redéfinition du rôle du système expert comme étant un système d'assistance et d'aide à la décision.

Prototype d'un examen final

Exercice N°1

Construire un réseau sémantique représentant :

Eva élève de CE1, 8 ans

Christophe, élève de 6ème, 12 ans

Eric, élève de CE2, 9 ans

Anne, élève de 6ème, 13 ans

Sophie, élève de 4ème, 13 ans

1. Utiliser le concept personne et éventuellement des sous-concepts
2. Construire le réseau sémantique permettant de répondre à la question « quelles sont les filles qui sont en 6ème »

Exercice N°2

Soit la base de règles suivantes :

R 01 : Si le candidat a actuellement un poste à responsabilité (**A**)
Et Le candidat a des facilités pour apprendre les langues (**B**)
Et Le candidat parle Français (**C**)
Alors le candidat est dynamique (**D**)

R 02 : Si le candidat a des facilités pour apprendre les langues (**B**)
Et Le candidat parle anglais (**E**)
Alors le candidat a une bonne adaptabilité (**F**)

R 03 : Si le candidat est slave (**G**)
Et Le candidat est dynamique (**D**)
Alors le candidat a une bonne adaptabilité (**F**)

R 04 : Si le candidat a des capacités de communication (**H**)
Et le candidat a reçu une formation convenable (**I**)
Alors le candidat a une capacité de leadership (**J**)

R 05 : Si le candidat a des facilités pour apprendre les langues (**B**)
Alors le candidat parle néerlandais (**K**)

R 06 : Si le candidat a une bonne adaptabilité (**F**)
Et Le candidat a une capacité de leadership (**J**)
Alors le candidat est accepté (**L**)

R 07 : Si le candidat est slave (**G**)
Alors le candidat a des facilités pour apprendre les langues (**B**)

R 08 : Si le candidat a une capacité de leadership (**J**)
Et Le candidat est slave (**G**)
Alors le candidat a une bonne adaptabilité (**F**)

Un système expert veut déduire « **si le candidat est accepté** » à travers un ensemble de questions posé à l'utilisateur. Au départ, le système ne sait pas les valeurs de vérité des faits, pour cela il pose un ensemble de questions pertinentes à l'utilisateur. Ce dernier peut répondre par : Oui, Non, je ne sais pas.

Supposons que l'ensemble de réponses de l'utilisateur sont les suivants :

Question 1 ⇒ Réponse = Oui
Question 2 ⇒ Réponse = Non
Question 3 ⇒ Réponse = Oui
Question 4 ⇒ Réponse = Oui
Question 5 ⇒ Réponse = Je ne sais pas

10. Déduire l'ensemble de questions pertinentes posé par le système à l'utilisateur dans l'ordre induit par une exploration en profondeur d'abord (arbre ET-OU).
11. Est-ce que le candidat est accepté ?

Exercice N°3

Soit l'énoncé suivant :

1. Les animaux sont toujours mortellement offensés si je ne fais pas attention à eux
2. Les seuls animaux qui m'appartiennent se trouvent dans ce pré
3. Aucun animal ne peut résoudre une devinette s'il n'a reçu une formation convenable dans une école
4. Aucun des animaux qui se trouvent dans ce pré n'est un raton-laveur
5. Quand un animal est mortellement offensé, il se met toujours à courir en tous sens et à hurler
6. Je ne fais jamais attention à un animal qui ne m'appartient pas
7. Aucun animal qui a reçu dans une école une formation convenable ne se met à courir en tous sens et à hurler

On peut coder les propositions précédentes par des règles ne faisant intervenir que des faits booléens. On définit les abréviations :

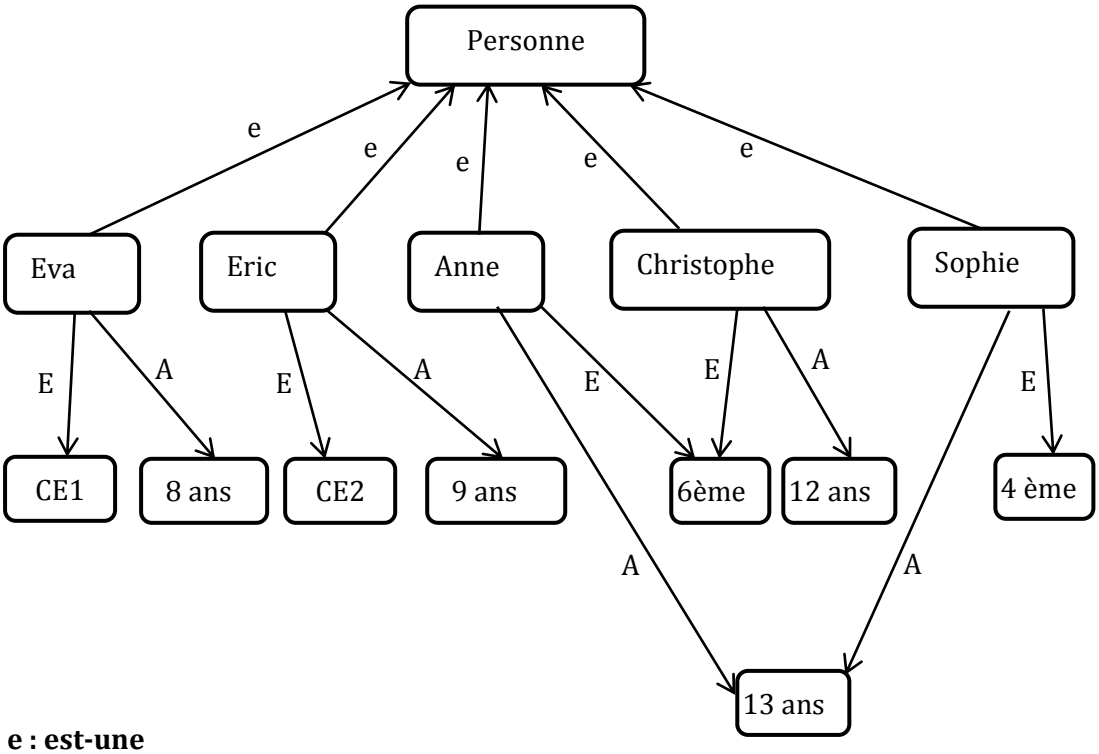
o : il est mortellement offensé
a : je fais attention à lui
ap : il m'appartient
p : il est dans ce pré
d : il peut résoudre une devinette
f : il a reçu une formation convenable dans une école
r : c'est un raton-laveur
c : il se met à courir en tous sens et à hurler

1- Considérant le texte et la codification précédente écrivez la base de règles ?
2- Un utilisateur demande ce que l'on peut déduire des propositions précédemment proposé si l'on énonce « cet animal est un raton-laveur » (c a d BF={r})

Pour cela : Utilisez le chaînage avant pour répondre à la question utilisateur

Corrigé type

Exercice N°1

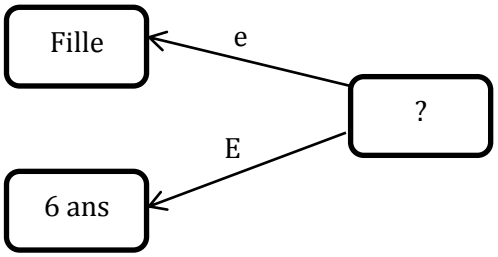


e : est-une

E : élève-en

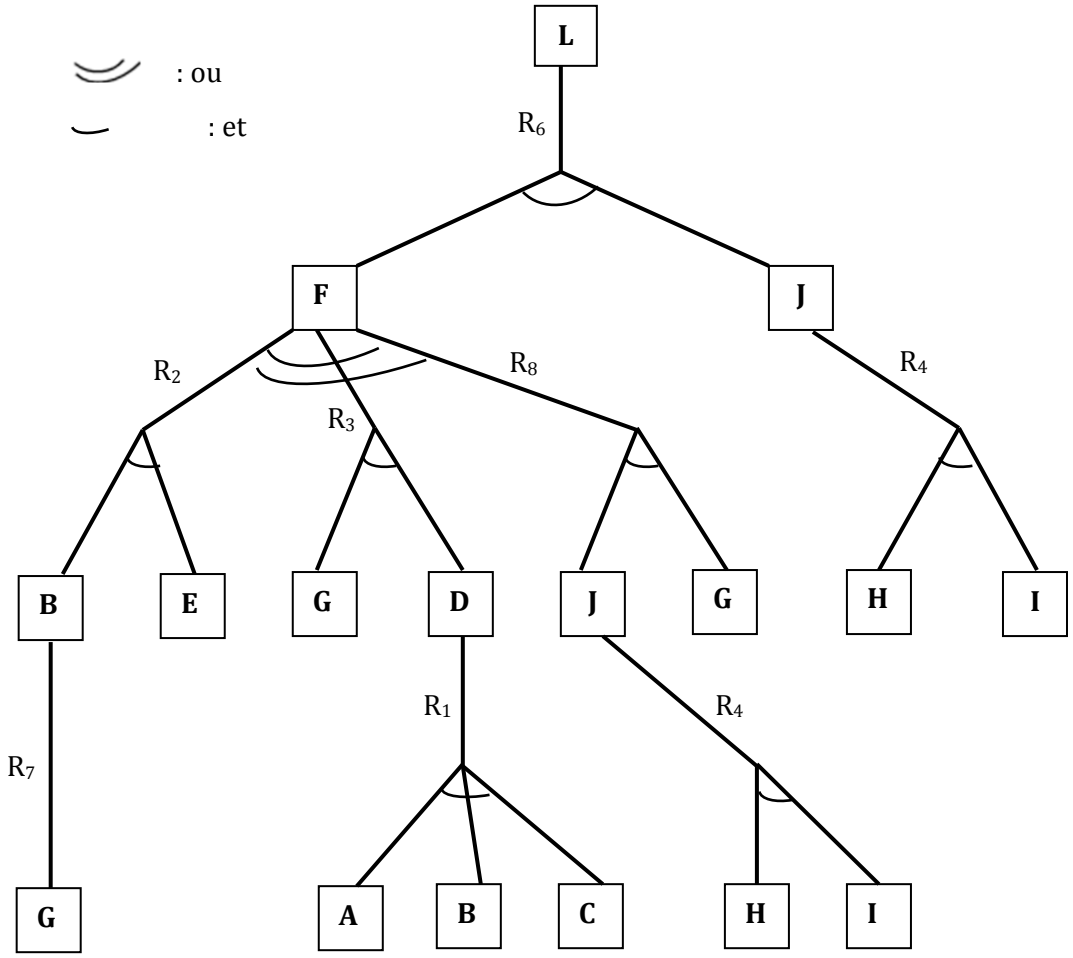
A : âgé(e)-de

Réseau sémantique permettant de répondre à la question « Quelles sont les filles qui ont 6 ans ? »



Exercice N°2

D'abord on doit construire l'arbre ET-OU (on utilise la codification des règles donné dans l'exercice).



On explorant l'arbre précédant en profondeur, les questions pertinentes posées par le système sont les suivantes :

Q1 : Est-ce que le candidat est salve (**G**) ?

Q2 : est-ce que le candidat parle anglais (**E**) ?

Q3 : est-ce que le candidat a un poste de responsabilité (**A**) ?

Q4 : Est-ce que le candidat parle français (**C**)?

Q5 : est-ce que le candidat a des capacités de communication (**H**) ?

2. Selon les réponses de l'utilisateur, le candidat **n'est pas accepté.**

Exercice N°3

1) La base de règles est constituée de :

1. si ¬a alors o
2. si ¬p alors ¬ap
3. si ¬f alors ¬d
4. si r alors ¬p
5. si o alors c
6. si a alors ap
7. si c alors ¬f

2) Par chaînage avant, de r on peut déduire ¬p, de ¬p on peut déduire ¬ap... et c'est terminé ! Il est pourtant possible de déduire logiquement plus de choses que celà. Pour le voir, il suffit de remarquer que (règle 6) : (a ⇒ ap) est logiquement équivalent à sa contraposée (règle 6bis) (¬ap ⇒ ¬a).

On peut donc logiquement conclure (règle 6bis) que ¬a est vrai, puis que o est vrai (règle 1), que c est vrai (règle 5), que ¬f est vrai (règle 7) et finalement que ¬d est vrai. Donc, cet animal est donc incapable de résoudre une devinette.

Bibliographie

Ajith , A., (2005), "Rule-based Expert Systems HEURISTICS," Ed: Wiley Online Library, pp. 909-919.

Bauchanan, B. G., Shortliffe, E. H., (1984), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project:* Addison- Wesley.

Bauchanan, B. G., Feigenbaum, E. A., (1980), The Stanford Heuristic Programming Project: Goals and Activities, *AI Magazine,* vol. 1, pp. 25-30.

Bellman, R., (1978), *An introduction to artificial intelligence: Can computers think?,* Boyd & Fraser Publishing Company.

Boisard, O., (2014), Cours D'intelligence artificielle, http://www.planete-a-roulettes.net/DOC/IA_BOISARD.pdf, (dernier accès Novembre 2019).Grzymala, J.W., (1991), *Managing uncertainty in expert systems.* Norwell, Kluwer academic Publishers, Boston, London.

Caplat, G., (2002), Modélisation cognitive et résolution de problèmes, Presses Polytechniques et Universitaires Romandes (PPUR).

Charbonier, M., (2008), Les systèmes experts : Etat de l'art et application possible aux SIG, Projet bibliographique dans le cadre du Master ASIG.

Chaudet, H., Liliane Pellegrin, L., (1998), *Intelligence artificielle et psychologie cognitive*, Dunod.

Chenfour, N., Systèmes Experts et Moteurs d'inférence, Chapitre 5, Intelligence Artificielle et Systèmes Experts.

Chikvina, O., (2011), *JBoss Developer Studio 4.0 Drools Tools Reference Guide.*

CLIPS'94, (1994), "Third Conference on CLIPS Proceedings," Lyndon B. Johnson Space Center.

CLIPS, (2012), *CLIPS: A Tool for Building Expert Systems*. *http://www.clipsrules.net/AboutCLIPS.html* (Dernier accès Novembre 2019).

Desharnais, J., (2006), Logique et Techniques de preuve et Mathématiques pour Informaticiens, Notes de cours, Université Laval, Québec, http://www2.ift.ulaval.ca/~dadub100/cours/H09/22257/ntsLogique.pdf (dernier accès Septembre 2019).

Dubus, N., (1995), Le transfert de la connaissance pour la réalisation d'un système expert, Deuxième rencontres de Théo Quant, Besançon 4-5 octobre.

Ganascia, J.G., (1990), *L'âme-machine : les enjeux de l'intelligence artificielle*, Éditions du Seuil, Paris.

Frappier,M., (2019), Logique et mathématiques discrètes, Notes de cours (version 2019), Université de Sherbrooke,

http://www.dmi.usherb.ca/~frappier/IFT734/ref/logique/logique.pdf (Dernier accès Octobre 2019).

FÜRST, F., (2002), « L'ingénierie ontologique », Rapport de recherche, Institut de Recherche en Informatique de Nantes France.

Giareatano, J.C., Riley, G., (2002), *Expert_Systems_Principles_and_Programming, Ed.* Thomson Course Technology.

Kayser, D., (1997), *La représentation des connaissances*, Hermes.

Laalam, L.F., (2007), Modélisation et gestion de la maintenance dans les systèmes de production, thèse de doctorat, Université de Annaba.

Lesage, M., (2004), Notes de cours d'intelligence artificielle, Département de mathématique, d'informatique et de génie, Université de Québec.

Ludovic, M., (2002), Programmation fonctionnelle et logique, Dossier sur Prolog, https://fr.scribd.com/document/58615697/Cours-Prolog (Dernier accès Septembre 2019).

Lindsay, R. K., Buchanan, B. G, Feigenbaum, E.A., Lederberg.J., (1993), DENDRAL: a case study of the first expert system for scientific hypothesis formation," *Artificial Intelligence,* vol. 61(2), pp.209-261.

McCarthy, J., Minsky, M., Rochester, N., Shannon, C., (1995), Proposal for the Darmouth summer research project on Artificial Intelligence, *Technical report,* Darmouth College.

Negrello, L, Système experts et Intelligence Artificielle, CT 157 édition Novembre 1991.

Nexpert, Documentation technique, Societe Neuron Data.

Nault, J.G., (2012), Système intelligent d'aide au contrôle de procédé dans un four à PARC, Mémoire de maitrise en Informatique, Université de Quebec.

Dennis, F., (2006), Intelligence artificielle : les systèmes experts, Notes de cours, Laboratoire d'Informatique Fondamentale, Marseille.

Paquette, G., Roy, L., (1990), Système à base de connaissances, Sainte-Foy, Télé-université et Montréal : Beauchemin.

Poole, D., Mackworth, A.K., Goebel.R., (1998), Computational intelligence: A logical approach. Oxford University Press, UK.

Russell, S., Norvig, P., (1994), *Artificial Intelligence: A Modern Approach*, Prentice Hall.

Schalkoff, R, J., (2011), *Intelligent systems: principles*, *paradigms and pragmatics:* Jones and Bartlett Publishers.

Shu-hsien, L., (2005), Expert system methodologies and applications - a decade review from 1995 to 2004," *Expert Systems with Applications,* vol. 28, pp. 93-103,

Thayse, A., (1990), *Approche logique de l'intelligence artificielle*, Dunod informatique.

Turing, A., (1950), Computer machinery and intelligence, *Mind*, vol. 59, pp.433-460.