République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

Université Abbes Laghrour -khenchela-

Faculté des sciences et de la technologie

Département de Mathématiques et Informatique

Cours

Pour les étudiants de la 1ère année Master Sécurité et Technologie Web (STW)

SYSTEME EXPERT

DR. SOUIDI MOHAMMED EL HABIB

Table des matières

Introduction

Ce polycopié est destiné aux étudiants de la première année Master spécialité Sécurité et Technologie du Web ***(STW)*** *dans le cadre de l'apprentissage de l'unité fondamentale du premier semestre. Afin de pouvoir suivre ce cours, les étudiants doivent avoir des notions fondamentales sur la logique mathématique.*

L'objectif principal de ce module est de permettre aux étudiants de s'initier à l'intelligence artificielle à travers les systèmes experts ainsi que la représentation des connaissances.

Le premier chapitre est consacré à une introduction aux principes de l'intelligence artificielle. En d'autres termes, se focalise sur la définition et le développement de L'intelligence artificielle au fil du temps. En Addition, ce chapitre met l'accent sur les différents axes de l'intelligence artificielle, ainsi que son application de nos jours.

Le deuxième chapitre de ce support se concentre sur les différents formalismes de la représentation des connaissances. En premier lieu, le formalisme logique sera introduit sous forme d'une révision aux étudiants. Ensuite, de nouveaux formalismes seront introduits tels que les réseaux sémantiques, les graphes conceptuels, et la logique de description. Ce chapitre se clôturera par des exercices corrigés concernant ces différents formalismes.

Les principes d'un système expert seront présentés dans le chapitre 3 de ce support. Plus spécifiquement, ce chapitre détaillera les différents composants d'un système expert ainsi que les types des algorithmes d'inférence tels que le chainage avant, le chainage arrière, ainsi que le chainage mixte. Ce chapitre se clôturera par des exercices corrigés concernant ces différents principes présentés.

Le chapitre 4 sera consacré à l'histoire détaillée concernant le développement des systèmes experts. Ce chapitre contiendra aussi les différentes techniques de programmation d'un système expert avec une introduction bien détaillée sur le langage de programmation PROLOG. Ce chapitre se conclura par l'application des systèmes experts dans des domaines tels que la médecine, la géologie, etc.

Enfin, le chapitre 5 sera consacré aux différentes phases construisant le cycle de vie d'un système expert. Aussi, il contiendra les différents types de systèmes experts, comme ceux basés sur les réseaux de neurone et la logique floue. Les étapes de construction d'un système expert seront aussi bien détaillées dans ce chapitre.

Enfin, j'adresse mon invitation à tout lecteur ayant des critiques ou des remarques à me contacter par email : *souidi.mohammed@univ-khenchela.com*

DR. MOHAMMED EL HABIB SOUIDI

Chapitre 1 : Introduction à l'intelligence artificielle et domaines d'application

Objectifs du chapitre

Les objectifs du chapitre 1 peuvent être résumés comme suit :

- *Assimiler quelques définitions de l'intelligence artificielle.*
- *Avoir une idée sur l'histoire de l'intelligence artificielle ainsi que sur son développement au cours du temps.*
- *Connaitre les grands domaines d'application de l'intelligence artificielle.*
- *Savoir ou en est l'intelligence artificielle aujourd'hui.*

1.1. Définition de l'intelligence artificielle

Il n'existe pas vraiment de consensus sur la définition du terme "intelligence artificielle" (il n'y en a même pas sur le terme "intelligence"). Commençons donc par citer quelques définitions (ou tentatives de définitions) que l'on peut trouver dans la littérature :

- "L'étude des facultés mentales à l'aide des modèles de type calculatoires" (Charniak et McDermott, 1985) [1].
- "Conception d'agents intelligents" (Poole et al., 1998) [2].
- "Discipline étudiant la possibilité de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme est aujourd'hui meilleur que la machine" (Rich et Knight, 1990) [3].
- "L'automatisation des activités associées au raisonnement humain, telles que la décision, la résolution de problèmes, l'apprentissage, ..." (Bellman, 1978) [4].
- "L'étude des mécanismes permettant à un agent de percevoir, raisonner, et agir" (Winston, 1992) [5].
- "L'étude des entités ayant un comportement intelligent" (Nilsson, 1998) [6].

Comme on peut remarquer, ces 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. Certaines se focalisent sur le comportement du système, tandis que d'autres considèrent que c'est le fonctionnement interne (le raisonnement) du système qui importe. Une deuxième distinction peut être faite entre celles qui définissent l'intelligence à

partir de l'être humain et celles qui ne font pas référence aux humains mais a un standard de rationalité plus général. On peut donc décliner quatre façons de voir l'intelligence artificielle :

1. Créer des systèmes qui se comportent comme les êtres humains – cette définition opérationnelle de l'IA fut promue par Alan Turing, qui introduisit son fameux "test de Turing" selon lequel 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.
2. Créer des systèmes qui pensent comme des êtres humains – si l'on adhère à cette deuxième définition, cela implique que l'IA est une science expérimentale, car il faut comprendre au préalable la façon dont pensent les humains (sinon, comment savoir si une machine pense comme un homme ?) et ensuite évaluer les systèmes par rapport à leurs similarités avec le raisonnement humain.
3. Créer des systèmes qui pensent 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. De plus, ce standard de rationalité ne peut pas être atteint en pratique car la technologie actuelle ne permet pas de réaliser des calculs aussi complexes.
4. Créer des systèmes qui possèdent des comportements rationnels – cette dernière définition de l'IA concerne le développement des agents qui agissent pour mieux satisfaire leurs objectifs. On remarque que cette définition est plus générale que la précédente car raisonner logiquement peut être une façon d'agir rationnellement mais n'est pas la seule (par exemple, le réflexe de retirer sa main d'un objet brulant est rationnel mais n'est pas le résultat d'une inférence logique).

1.2. Une courte histoire de l'IA

Gestation de l'IA (1943-1955) : Pendant cette période furent menés les premiers travaux qui peuvent être considérés comme les débuts de l'intelligence artificielle (même si le terme n'existait pas encore). On peut citer les travaux de McCulloch et Pitts [7] qui ont introduit en 1943 un modèle de neurones artificiels. Quelques années après, Hebb proposa une règle pour modifier des connections entres neurones, et Minsky et Edmonds [8] construisirent le premier réseau de neurones.

Naissance d'IA (1956) : C'est durant cette année qu'un petit groupe d'informaticiens intéressés par l'étude de l'intelligence se réunirent pour une conférence sur ce thème. Cette conférence dura deux mois, et permit de poser les fondements de l'intelligence artificielle (nom qui fut choisi à l'issue de cette conférence).

Espoirs grandissants (1952-1969): Ce fut une période très active pour le jeune domaine de l'IA. Un grand nombre de programmes furent développés pour résoudre des problèmes d'une grande diversité. . Le General Problem Solver de Newell et Simon réussissait quant a lui a résoudre des puzzles simples avec un raisonnement semblable au raisonnement humain. Samuel créa un programme jouant (à un niveau moyen) aux dames. Des étudiants de Minsky travaillèrent sur les petits problèmes ("microworlds") tels que les problèmes d'analogie (problèmes du même type que ceux des tests de QI), donnant naissance au programme ANALOGY, ou encore les manipulations de cubes (les fameux "blocks world") avec l'idée d'augmenter la complexité petit à petit pour d´développer des agents intelligents [9]. McCarthy publia un article devenu célèbre dans lequel il traite des programmes qui ont du sens commun. La recherche sur les réseaux de neurones fut également poursuivie. Ce fut aussi l'époque du Shakey, le premier robot à être capable de raisonner sur ses propres actions [10].

Premières déceptions (1966-1973) : durant ces années, Il devint de plus en plus évident que les prédictions faites par les chercheurs en IA avaient été beaucoup trop optimistes. Ce fut le cas par exemple pour la traduction automatique. Les chercheurs n'avaient compté que 5 ans pour réaliser un traducteur automatique, mais se sont vite rendu compte que leur approche purement syntaxique n'était pas suffisante (pour bien traduire un texte, il faut d'abord le comprendre). Cet échec a provoqué l'annulation en 1966 de tout le financement du gouvernement américain pour les projets de traduction automatique. De grandes déceptions se produisirent également lorsque les chercheurs en IA essayèrent d'appliquer leurs algorithmes aux problèmes de grande taille, et découvrirent alors qu'ils ne fonctionnaient pas, par manque de mémoire et de puissance de calcul. Ce fut une des critiques adressées à l'IA dans le rapport de Lighthill de 1973 [11], qui provoqua l'arrêt du financement de la quasi-totalité des projets en IA de Grande Bretagne. Et comme si cela ne suffisait pas, Minsky et Papert prouvèrent dans leur livre "Perceptrons" de 1969 [12] 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, entraînant une crise dans cette branche de l'IA.

Systèmes Experts (1969-1979) Le premier système expert, appelé DENDRAL [13], fut créé en 1969 pour la tâche spécialisée consistant à déterminer la structure moléculaire d'une molécule étant donnés sa formule et les résultats de sa spectrométrie de masse. DENDRAL, comme tous les systèmes experts, est basé sur un grand nombre de règles heuristiques élaborées par des experts humains. Après le succès du DENDRAL, d'autres systèmes d'experts furent créés, notamment le système MYCIN [14], qui réalisait un diagnostic des infections sanguines. Avec 450 règles, MYCIN réussissait à diagnostiquer à un niveau proche des experts humains et considérablement meilleur que celui les jeunes médecins.

Le retour des réseaux de neurones (1986-present) Au milieu des années 80, quatre groupes de chercheurs ont découvrirent indépendamment la règle d'apprentissage "back-propagation" [15] qui permit le développement de 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). Depuis, l'apprentissage automatique est devenu l'un des domaines les plus actifs de l'IA, et a été appliqué avec succès à de nombreux problèmes pratiques (comme par exemple la fouille de données). 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.

1.3. Les sous-domaines de l'IA

On s'en serait douté, créer des agents intelligents n'est pas si simple. Pour cette raison, l'IA s'est divisée en de nombreuses sous-disciplines qui essaient chacune de traiter une partie du problème. Voici les principales :

1.3.1. Représentation des connaissances et Raisonnement Automatique

La représentation des connaissances concerne un ensemble d'outils et de procédés qui servent d'un cote à représenter et d'un autre à organiser le savoir humain pour l'utiliser et le partager. Dans le chapitre suivant, nous allons détailler 4 types de ces outils :

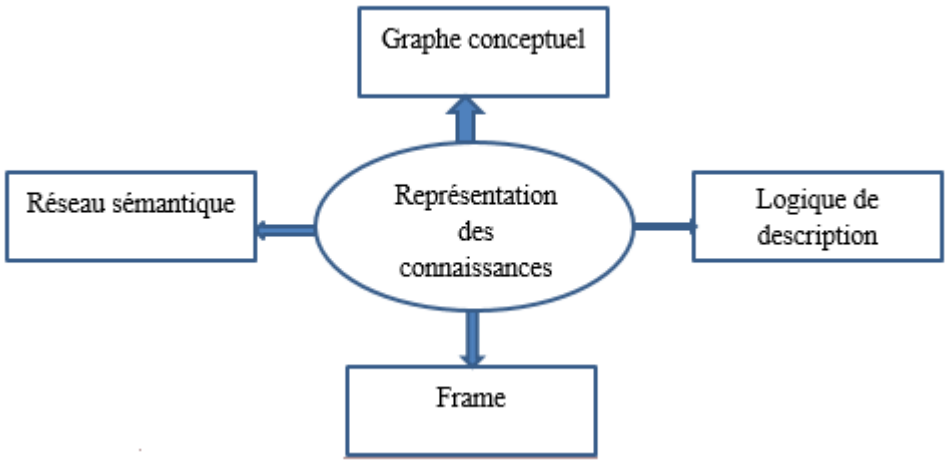
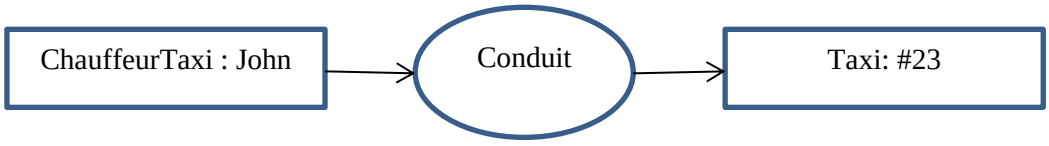


Figure 1.1 : Les types de la représentation des connaissances.

- Les réseaux sémantiques : Un réseau sémantique est un graphe orienté et étiqueté. Ce graphe est formé de nœuds - représentant les concepts – reliés par des arcs Le concept n'acquiert tout son sens que par les relations qui le lient aux autres concepts. Une liaison entre deux nœuds étiquetés par A et B ont la propriété d'être en relation par R : R(A,B).
- La logique de description : cette famille de langages de représentation de connaissance peut être utilisée pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée.
- Les graphes conceptuels sont un formalisme par lequel l'univers du discours est représenté par des concepts et des relations conceptuelles. Un concept représente un objet du discours. Les relations conceptuelles permettent d'associer les concepts. La figure 1 représente la phrase : ''John est un chauffeur de taxi qui conduit le taxi numéro 23''. La même phrase peut être représentée sous une forme linéaire ou sous une forme graphique :

[ChauffeurTaxi :John]->(conduit)->[Taxi :#23].



- Frame : Un frame est une structure de données représentant une situation prototypique. Cette méthode s'oppose aux représentations relationnelles dans lesquelles un objet est décrit à travers ses propriétés disséminées dans des règles ou formules logiques.

1.3.2. Les systèmes experts :

"Un logiciel intelligent qui utilise des connaissances et des inférences logiques pour résoudre des problèmes qui sont suffisamment difficiles pour nécessiter une expertise humaine important pour trouver une solution" (Feigenbaum 1982) [16].

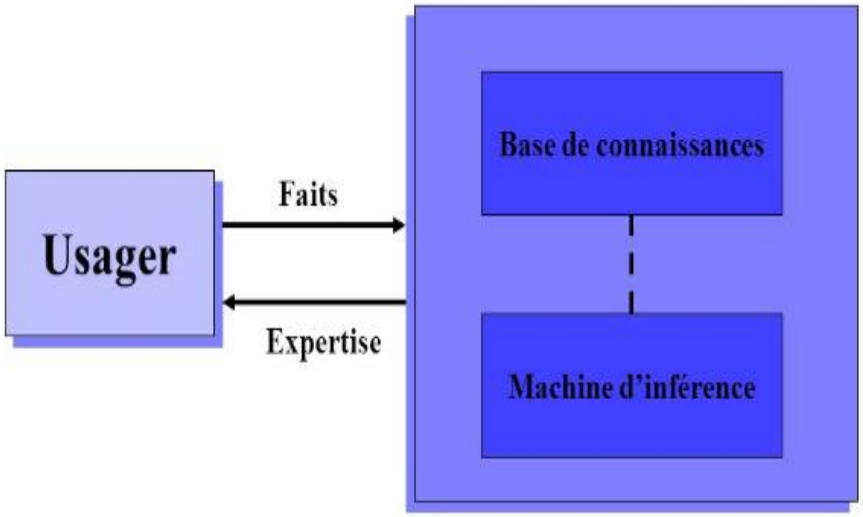


Figure 1.2 : Mécanisme d'un système expert

Un système expert est composé de :

a. La base de connaissances : cette base est composée de deux autres bases, la base de faits qui code la connaissance sur l'étude en cours Son état évolue en cours d'expertise (mémoire de travail). Deuxièmement, la base de règles qui code la connaissance sur le domaine. Fixe pour plusieurs expertises. Règle : SI condition ALORS action.
b. Le moteur d'inférences : composés des algorithmes utilisés pour la déduction : chaînage avant, chaînage arrière, et chaînage mixte.

1.3.3. Agent intelligent et système multi-agents :

Un agent peut être considérer comme une entité autonome qui interagit avec l'environnement. Cette interaction est réalisée grâce aux capteurs et aux effecteurs. Les capteurs servent à détecter (ou à lire) le contenu de l'environnement de façon partielle ou semi-partielle. Les effecteurs modifient ce contenu en concordance avec les objectifs de l'agent. L'agent peut être perçu comme la partie software d'un robot (le robot est un agent physique), un être humain ou même un logiciel.

Dans le but de faciliter le mouvement des agents, l'environnement est souvent fragmenté en unités de même taille ou tailles différentes selon le besoin (grille de cellules). Dans le cas des agents mobiles ou les agents se déplacent dans un réseau ou un système distribué, l'environnement est considéré comme graphe (un ensemble de nœuds interconnectés).

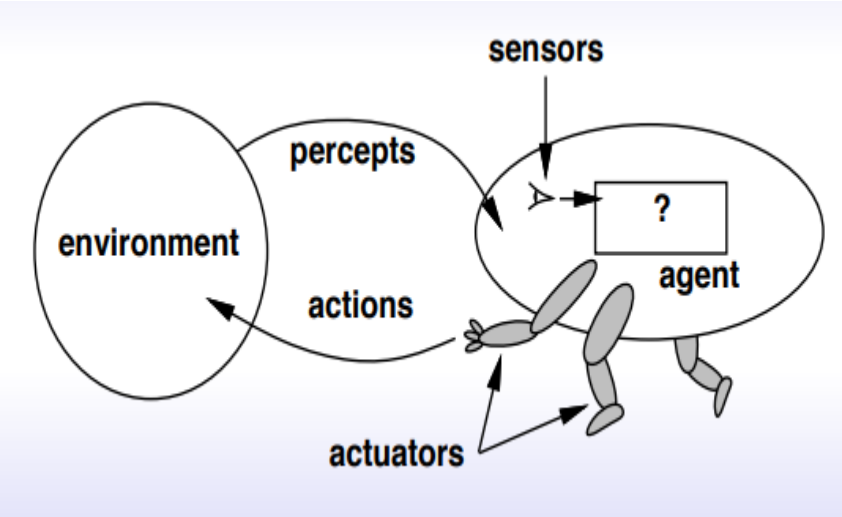


Figure 1.3 : Interaction agent-environnement.

Un système multi-agent (SMA) est un système composé d'un ensemble d'agents situés dans un certain environnement et interagissant selon certaines relations. Les SMAs sont utilisés dans les cas de problèmes complexes (irrésolvable par un seul agent) ou dans le cas d'une résolution parallèle d'un problème multitâches. Ils sont aussi très utilisés pour la simulation ou l'étude d'une communauté (interactions, partage de taches, coopérations, compétions..). Ces agents agissent en respectant un mécanisme de coordination précis.

1.3.4. L'apprentissage automatique :

L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques. Son but principal est d'extraire et exploiter automatiquement l'information présente dans un jeu de données.

Les algorithmes d'apprentissage peuvent se catégoriser selon le type d'apprentissage qu'ils emploient :

- **L'apprentissage supervisé :** L'apprentissage supervisé est une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir d'exemples annotés. On distingue les problèmes de régression des problèmes de classification. Ainsi, on considère que les problèmes de prédiction d'une variable quantitative sont des problèmes de régression tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification.

  Les exemples annotés constituent une base d'apprentissage, et la fonction de prédiction apprise peut aussi être appelée « hypothèse » ou « modèle ». On suppose cette base d'apprentissage représentative d'une population d'échantillons plus large et le but des méthodes d'apprentissage supervisé est de bien généraliser, c'est-à-dire d'apprendre une fonction qui fasse des prédictions correctes sur des données non présentes dans l'ensemble d'apprentissage.

- **L'apprentissage non-supervisé :** Il s'agit, pour un logiciel, de trouver des structures sous-jacentes à partir de données non étiquetées. Puisque les données ne sont pas étiquetées, il n'est pas possible d'affecter au résultat de l'algorithme utilisé un score d'adéquation. Cette absence d'étiquetage (ou d'annotation) est ce qui distingue les tâches d'apprentissage non-supervisé des tâches d'apprentissage supervisé.

  L'apprentissage non supervisé consiste à apprendre sans superviseur. Il s'agit d'extraire des classes ou groupes d'individus présentant des caractéristiques communes. La qualité d'une méthode de classification est mesurée par sa capacité à découvrir certains ou tous les motifs cachés.

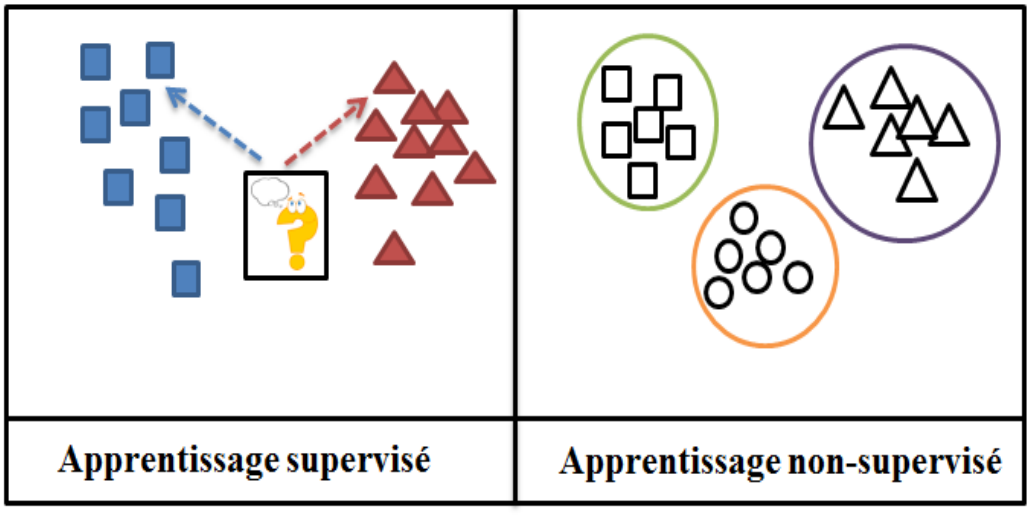


Figure 1.4 : Apprentissage supervisé VS non supervisé

On distingue l'apprentissage supervisé et non supervisé. Dans le premier apprentissage, il s'agit d'apprendre à classer un nouvel individu parmi un ensemble de classes prédéfinies: on connaît les classes à priori. Tandis que l'apprentissage non-supervisé, le nombre et la définition des classes n'étant pas données à priori.

Apprentissage supervisé	**Apprentissage non supervisé**
• On dispose d'éléments déjà classés Exemple : articles en rubrique cuisine, sport, culture... • On veut classer un nouvel élément Exemple: lui attribuer un nom parmi cuisine, sport, culture...	• On dispose d'éléments non classés Exemple : une fleur • On veut les regrouper en classes Exemple : si deux fleurs ont la même forme, elles sont en rapport avec une même plante correspondante.

- **L'apprentissage par renforcement :** l'apprentissage par renforcement consiste, pour un agent autonome à apprendre les actions à prendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps. L'agent est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, un comportement décisionnel (appelé stratégie ou politique, et qui est une fonction associant à l'état courant l'action à exécuter) optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.

Figure 1.5 : Interaction agent-environnement via AR

Les incertitudes vont être gérées en utilisant le Processus Décisionnel de Markov Parfaitement Observé (MDP) (Bellman, 1957) [17], qui peut être défini comme une formalisation mathématique de problèmes dans lesquels l'agent doit décider comment agir afin de maximiser une fonction de gain. En utilisant le MDP l'agent connaît à chaque instant sa position exacte au sein de l'environnement.

Ainsi un MDP est défini par un tuple « S,A,T,R ».

- S : ensemble fini d'états de l'environnement parfaitement identifiables ;
- A : ensemble fini d'actions ;
- T : fonction de transition entre états selon l'action effectuée. On note T(s,a,s') la probabilité de passer de l'état s à l'état s' en effectuant l'action a.
- R : fonction de gain qui permet de déterminer le(s) but(s) à atteindre et les éventuelles zones dangereuses de l'environnement. Cette fonction peut être définie de différentes manières, suivant le problème à résoudre.

1.4. L'importance de l'Intelligence Artificielle

L'Intelligence Artificielle traite de l'étude et de la construction de systèmes rationnels. L'Intelligence Artificielle présente les avantages suivants :

- Cette méthode est plus générale par rapport à la méthode des lois de la pensée.
- Cette méthode répond à la science plus qu'aux méthodes basées sur le comportement ou la pensée humaine.
- Le premier programme informatique introduit pour battre le champion d'échecs était Hitech [18]. Des compétences intellectuelles limitées sont requises pour les stratégies d'échecs.
- L'intelligence artificielle est divisée en plusieurs branches. Programmes de recherche d'intelligence artificielle pour étudier les différentes probabilités qui incluent les mouvements de jeu d'échecs. Différentes manières sont mises en place pour mener à bien plus efficacement. L'intelligence artificielle logique traite du programme logique mathématique. Un programme de reconnaissance de formes étudie et compare avec une forme particulière. Il correspond aux yeux et au nez du visage d'une carte d'identité de la police. L'épistémologie traite de l'étude des différents types de connaissances nécessaires pour résoudre des problèmes.

- Actuellement, le programme de reconnaissance vocale ne parvient pas à reconnaître chaque mot mais mène lentement à une amélioration. Cette idée guide les personnes handicapées à utiliser les ordinateurs.
- Maintenant, les ordinateurs exécutent de nombreuses tâches qui dépassaient notre imagination. En constate maintenant (année 2022) que les ordinateurs sont plus intelligents que l'humanité. Grâce à ces progrès, nous ne serons pas inférieurs aux ordinateurs car nous pouvons les éteindre lorsqu'ils ne fonctionnent pas.

1.5. L'intelligence Artificielle Aujourd'hui

1.5.1. La robotique :

Les robots sont une application pratique de l'intelligence artificielle d'une importance croissante pour les tâches industrielles, domestiques, de divertissement et militaires. Les capacités des robots d'aujourd'hui vont bien au-delà des robots d'usine du siècle dernier, dont les opérations dépendaient des conditions strictement contrôlées d'une chaîne de montage.

Les robots de terrain et de service d'aujourd'hui doivent pouvoir fonctionner dans le monde non structuré d'un atelier, d'une maison, d'une école ou d'un champ de bataille. De plus, le logiciel contrôlant de telles machines doit être beaucoup plus flexible, robuste et autonome que les langages de programmation de robots du passé. Ils doivent pouvoir être commandés pour s'acquitter de leurs tâches beaucoup plus facilement.

Afin de résoudre des problèmes aussi exigeants, il est non seulement nécessaire de construire et de tester de nouveaux types de machines, mais également de créer de nouvelles et puissantes classes d'algorithmes, dont l'applicabilité générale peut être démontrée par le fonctionnement réussi de ces machines sur des problèmes difficiles.

1.5.2. La vision par ordinateur :

La vision par ordinateur est un domaine qui comprend les méthodes d'acquisition, de traitement, d'analyse et de compréhension des images et, en général, des données de grande dimension du monde réel afin de produire des informations numériques ou symboliques, par exemple sous forme de décisions.

Les applications vont de tâches telles que les systèmes de vision industrielle industriels qui, par exemple, inspectent les bouteilles qui défilent rapidement sur une chaîne de production, à la recherche sur l'intelligence artificielle et les ordinateurs ou robots capables de comprendre le monde qui les entoure. Les domaines de la vision par ordinateur et de la vision artificielle

se chevauchent considérablement. La vision par ordinateur couvre la technologie de base de l'analyse d'images automatisée qui est utilisée dans de nombreux domaines. La vision artificielle fait généralement référence à un processus de combinaison d'analyse d'image automatisée avec d'autres méthodes et technologies pour fournir une inspection automatisée et un guidage de robot dans des applications industrielles.

En tant que discipline scientifique, la vision par ordinateur s'intéresse à la théorie des systèmes artificiels qui extraient des informations à partir d'images. Les données d'image peuvent prendre de nombreuses formes, telles que des séquences vidéo, des vues provenant de plusieurs caméras ou des données multidimensionnelles provenant d'un scanner médical.

1.5.3. Les réseaux de neurones :

Un réseau de neurones est, par essence, une tentative de simulation du cerveau. La théorie des réseaux de neurones s'articule autour de l'idée que certaines propriétés clés des neurones biologiques peuvent être extraites et appliquées à des simulations, créant ainsi un cerveau simulé (et très simplifié). La première chose importante à comprendre est alors que les composants d'un réseau de neurones artificiels sont une tentative de recréer le potentiel informatique du cerveau. La deuxième chose importante à comprendre, cependant, est que personne n'a jamais prétendu simuler quelque chose d'aussi complexe qu'un cerveau réel. Alors que l'on estime que le cerveau humain a quelque chose de l'ordre de dix à cent milliards de neurones, un réseau de neurones artificiels (ANN) typique n'est pas susceptible d'avoir plus de 1 000 neurones artificiels. Bien qu'il existe de nombreux types de réseaux de neurones artificiels, la plupart sont organisés selon la même structure de base.

Cette organisation comporte trois composants : un ensemble de nœuds d'entrée, une ou plusieurs couches de nœuds "cachés" et un ensemble de nœuds de sortie. Les nœuds d'entrée reçoivent des informations et s'apparentent à des organes sensoriels. Que l'information soit sous la forme d'une image numérisée, ou d'une série de valeurs boursières, ou à peu près n'importe quelle autre forme qui peut être exprimée numériquement, c'est là que le net obtient ses données initiales. Les informations sont fournies sous forme de valeurs d'activation, c'est-à-dire que chaque nœud reçoit un nombre, des nombres plus élevés représentant une plus grande activation. C'est exactement comme les neurones humains, sauf qu'au lieu de transmettre leur niveau d'activation en tirant plus fréquemment, comme le font les neurones biologiques, les neurones artificiels indiquent l'activation en transmettant cette valeur

d'activation aux nœuds connectés. Après avoir reçu cette activation initiale, les informations sont ensuite transmises via le réseau.

Chapitre 2 : Formalisme de la représentation des connaissances

Objectifs du chapitre

Les objectifs du chapitre 2 peuvent être résumés comme suit :

- *Comprendre le rôle du formalisme logique dans le domaine de la représentation des connaissances.*
- *Lever l'ambiguïté sur la représentation des connaissances à travers l'utilisation des réseaux sémantiques.*
- *Savoir comment représenter des connaissances via l'utilisation des principes des graphes conceptuels.*
- *Assimiler les principes de base des frames ainsi que leur utilité dans la représentation des connaissances.*
- *Représenter des connaissances en se basant sur l'utilisation de la logique de description.*

2.1. Formalisme logique

2.1.1. La logique de proposition

En logique mathématique [19], le calcul des propositions est considéré comme la première étape concernant la définition de la logique et du raisonnement. Ce calcul définit les règles de déduction reliant les propositions entre elles, tout en évitant d'examiner le contenu. Le calcul des propositions est ainsi le premier pas dans la construction du calcul des prédicats, qui lui par contre s'intéresse au contenu des propositions. En d'autres termes, le calcul des prédicats est une formalisation achevée du raisonnement mathématique. Le calcul des propositions est parfois connu sous le nom de logique des propositions, logique propositionnelle ou calcul des énoncés, et parfois théorie des fonctions de vérité.

2.1.2. Définition de la logique de proposition

On définit une proposition [20] comme un assemblage de mots d'une langue naturelle qui vérifie les trois propriétés suivantes :

1. Cet assemblage est reconnu syntaxiquement correct.
2. Cet assemblage est sémantiquement correct.
3. Il est possible de lui attribuer une valeur de vérité (vrai ou faux) sans aucune ambiguïté.

Exemples :

- Hassan 2 est un nombre premier : cette phrase est syntaxiquement correcte (1), mais sémantiquement incorrecte (2). Donc, cette phrase n'est pas une proposition.

-le ciel est bleu : les propriétés (1, 2 et 3) sont vérifiées : c'est une proposition qui peut être vraie ou fausse.

Le joueur est-il arrivé ? : la phrase est syntaxiquement et sémantiquement correcte, mais on ne peut pas lui attribuée une valeur de vérité. Donc, cette phrase n'est pas une proposition.

2.1.3. Syntaxe de la logique de proposition

La syntaxe d'un langage se focalise sur la définition de l'alphabet ainsi que les règles d'écriture (grammaire) des expressions du langage. D'un autre côté, elle ne se focalise pas sur leurs sens.

L'alphabet est composé des symboles formant le langage. Il contient :

– Un ensemble dénombrable de variables propositionnelles. On convient de les représenter par les lettres de l'alphabet latin (a, b, c ...). Ces lettres peuvent éventuellement être indicées.

– Des connecteurs logiques (¬, ∧, ∨,⇒,⇔).

– Des symboles auxiliaires.

Les règles d'écriture définissent la méthode avec laquelle sont assemblés les symboles de l'alphabet afin de former des expressions bien formées (ou formules) du langage propositionnel :

1. Toute variable propositionnelle est une formule

2. Si A est une formule, alors, ¬A (ou $\bar{A}$) est aussi une formule.

3. Si A et B sont des formules, (A ∧ B), (A ∨ B), (A ⇒ B) et (A ⇔ B) sont aussi des formules.

4. Une expression n'est considérée comme formule que si elle est écrite en conformité avec les règles 1,2 et 3.

Exemple :

1. p, q, r : sont des variables propositionnelles, donc des formules.

2. p ∨ q : est une formule.

3. p ⇒ (q¬r) : n'est pas une formule.

Concernant les priorités, les connecteurs sont applicable dans l'ordre (¬, ∧, ∨,⇒, ⇔), par exemple :

1.¬p ∧ q correspond à la formule (¬p) ∧ q.

2. $p \wedge q \Rightarrow r$ correspond à la formule $(p \wedge q) \Rightarrow r$.
3. $p \vee q \wedge r$ correspond à la formule $p \vee (q \wedge r)$.
4. $p \vee q \vee r$ correspond à la formule $(p \vee q) \vee r$.

2.1.4. Sémantique de la logique de proposition :

L'objectif de l'étude sémantique d'un langage pour le calcul des propositions est d'attribuer une valeur de vérité aux formules du langage. Cette étude est aussi connue sous le nom de la théorie des modèles. Une fonction de valuation est associée par la sémantique :

$$V : vp \rightarrow \{1, 0\},$$

Où vp représente l'ensemble des variables propositionnelles, 1 signifie vrai et 0 signifie faux, unique à chacun des connecteurs logiques. La description de Cette fonction est effectuée à l'aide d'un graphe appelé table de vérité (ou tableau de vérité). Une fonction de vérité unique correspond chaque formule α à n variables propositionnelles. Le graphe de cette fonction est représenté via une table de vérité à 2^n lignes reflétant la valeur de vérité de α correspondant à chaque combinaison de valeur de vérité des n variables (connue aussi comme la distribution de valeurs de vérité des variables).

Tableau 2.1. Tableau de vérité

A	B	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Comme le démontre le tableau ci-dessus :

- La conjonction de deux propositions A et B représentée symboliquement par $A \wedge B$, et qui correspond à la lecture (A et B), est vraie si et seulement si les deux propositions A et B sont vraies.
- La disjonction de deux propositions A et B représentée symboliquement par $A \vee B$ et correspond à la lecture (a ou b) est fausse si et seulement si les deux propositions a et b sont fausses.
- Le connecteur "⇒" est connu comme le connecteur d'implication, la proposition $A \Rightarrow B$ est fausse si et seulement si A est vraie et B est fausse. Soient A et B deux propositions formant la formule $A \Rightarrow B$, A est correspond á l'hypothèse (ou

antécédent), et B correspond à la thèse (ou la conséquence). (A ⇒ B) est appelée implication directe. Les implications apparentes à l'implication directe sont dénommées ainsi :

• A ⇒ B implication directe, ici A est une condition suffisante pour B (B si A).

• B ⇒ A implication réciproque (converse), dans cette formule, A est une condition nécessaire de B.

• $\bar{A} \Rightarrow \bar{B}$ est une implication contraire (inverse).

• $\bar{B} \Rightarrow \bar{A}$ est une implication contraposée.

- L'équivalence Le connecteur "⇔" est connu sous le nom de connecteur d'équivalence, la proposition a ⇔ b est vraie dans le cas où a et b ont la même valeur de vérité.

2.1.5. Limites de la logique de proposition :

Le calcul propositionnel est caractérisé par des propriétés intéressantes. Grace a ce calcul, on peut vérifier si une formule est valide ou bien satisfiable (Une formule A est dite satisfiable dans le cas ou sa table de vérité comprend au minimum une ligne où la valeur de vérité de A est vraie). D'un autre côté, la logique propositionnelle manque d'expressivité surtout dans le cas ou la modélisation des espaces infinis comme les entiers est nécessaire. On pourrait se baser une infinité de variables propositionnelles ainsi que sur une infinité d'hypothèses, mais la manipulation de ces derniers est une tâche très complexe pour des humains et encore moins pour des machines. Il peut être nécessaire de concevoir des descriptions finies qui rendent compte de cas potentiellement infinies. Cette notion est connue sous le nom de calcul des prédicats. Nous allons fournir une présentation du calcul des prédicats du premier ordre dans la section suivante de ce chapitre.

2.1.6. La logique des prédicats :

Le calcul propositionnel n'est pas illimité car il ne permet essentiellement que la représentation des opérations booléennes sur des propositions. Si on souhaite pouvoir effectuer des raisonnements sur des assertions mathématiques, il nous faudra intégrer des constructions plus riches. Par exemple, la formule ∀x ∈ N, x ≤ x2 reflète toutes les formules {0 ≤ 0, 1 ≤ 1 , 2 ≤ 4, 3 ≤ 9,. . . jusqu'à x ≤ x2}.

Un tel énoncé n'est pas supportable via l'utilisation de la logique propositionnelle sachant qu'il est basé sur des prédicats comme x ∈ N dont la valeur de vérité est dépendante d'une

variable x, ce qui est impossible en logique propositionnelle. Autrement dit, on utilise dans ce cas des quantificateurs tels que ∀, ∃ qui ne sont pas représentable en logique propositionnelle.
L'énoncé précédent est catégorisé comme un exemple de formule du calcul des prédicats du premier ordre. La terminologie premier ordre est basé sur le fait que les quantifications existentielles et universelles ne sont autorisées que sur les variables.

2.1.7. Définition de la logique des prédicats :

En logique mathématique [19], un prédicat [21] peut être défini comme un moule à propositions. En d'autres termes, c'est un énoncé syntaxiquement correct, il est sémantiquement correct, mais sa valeur de vérité est totalement dépendante de l'objet. Par exemple, le prédicat "x est pair" dépend de la valeur de l'entier x.
Un prédicat peut être à une variable P(x) (dans ce cas, on parle d'une propriété de l'objet x), un prédicat basé sur deux variables (ex : x < y, dans ce cas, le prédicat est une relation entre les deux objets x et y), ou encore un prédicat basé sur n variables (On parle ici d'un prédicat n-aire).

2.1.8. La logique des prédicats du premier ordre

L'alphabet de la logique des prédicats est formé :
– Des connecteurs logiques semblables a ceux de la logique de proposition : ¬, ∧, ∨, ⇒, ⇔.
– le quantificateur universel et existentiel : ∀, ∃.
– D'un ensemble dénombrable de variables : x, y, z, . . . ;
– D'un ensemble dénombrable de symboles de constantes qui peut être vide : a, b, e, . . .
– D'un ensemble dénombrable de symboles de fonctions qui peut aussi être vide : f0, f1, . . .
– D'un ensemble dénombrable de symboles de prédicats : P1, P2, Q, . . .
– Des symboles auxiliaires tel que : {,}.
Le Champ d'un quantificateur peut être défini comme la partie de la formule couverte par un quantificateur.

Tableau 2.2. Le champ des quantificateurs

∀x $\underline{A \wedge B}$	∀x $\underline{A} \Rightarrow B$
∃x $\underline{A \wedge B}$	∃x $\underline{A} \Rightarrow B$
∀x $\underline{A \vee B}$	∀x ($\underline{A \Rightarrow B}$)
∃x $\underline{A \vee B}$	∃x ($\underline{A \Rightarrow B}$)

– Dans une formule α, l'occurrence d'une variable x est considérée comme liée dans le cas ou elle se trouve dans le champ d'un quantificateur Qx. Dans un autre cas, elle est libre.
– Une variable x est libre dans une formule α s'il existe dans α une occurrence libre de x.
– Une variable x est liée dans une formule α s'il existe dans α une occurrence liée de x.
Exemple : $\forall yP(x, y) \wedge Q(y) \Rightarrow \exists wQ(w) \Rightarrow R(z)$
y, w : variables liées.
x, z : variables libres.

2.1.9. Le quantificateur existentiel :

L'objectif de la quantification est le pouvoir d'exprimer des propriétés sur une collection d'objets sans ressentir la nécessité de les désigner chacun par un nom [21].
La forme générale du quantificateur existentiel est comme suit : ∃ <variables><phrase>. Par exemple, pour représenter la phrase "Il y a quelqu'un de bizarre dans la maison", il faut utiliser le quantificateur comme suit :

$$\exists\ x\ \text{est-dans}(x, \text{maison}) \wedge \text{bizarre}(x)$$

Typiquement, la conjonction est le connecteur principal du quantificateur existentiel. Donc, l'utilisation de l'implication comme connecteur principal du quantificateur existentiel est une erreur fatale.
Exemple :
$\exists\ x\ \text{est-dans}(x, \text{maison}) \Rightarrow \text{bizarre}(x)$ n'est vrai que si quelqu'un n'est pas dans la maison! (Vérifiable avec la table de vérité).

2.1.10. Le quantificateur universel :

La forme générale du quantificateur universel [21] est notée: ∀ <variables><phrase>. Par exemple, la phrase : "Toute personne dans cet amphi est intelligente" doit être représentée comme suit :

$$\forall\ x\ \text{est-dans}(x, \text{amphi}) \Rightarrow \text{Intelligent}(x)$$

Typiquement l'implication est le connecteur principal du quantificateur universel. Donc, l'utilisation de la conjonction comme connecteur principal avec ce quantificateur est une erreur fatale. Par exemple :

$$\forall : \forall\ x\ \text{est-dans}(x, \text{amphi}) \wedge \text{Intelligent}(x)$$

Signifie : "tout le monde est dans cet amphi et tout le monde est intelligent".

2.1.11. Les lois de De Morgan

Les lois de De Morgan [22] peuvent être définies comme des identités entre propositions logiques. Elles ont été introduites par le mathématicien britannique Augustus De Morgan.

En logique mathématique, la négation de la disjonction de deux propositions est l'équivalence de la conjonction des négations des deux propositions, ce qui signifie que « ¬ (A ∨ B) » est identique à « (¬ A) ∧ (¬ B) ».

Toujours en logique mathématique, la négation de la conjonction de deux propositions est l'équivalence de la disjonction des négations des deux propositions, ce qui signifie que « ¬ (A ∧ B) » est identique à « (¬ A) ∨ (¬ B) ».

Les lois de De Morgan peuvent se résumer comme suit :

$$\neg\exists\ x\ \alpha \Leftrightarrow \forall\ x\ \neg\alpha$$

$$\neg\forall\ x\ \alpha \Leftrightarrow \exists\ x\ \neg\alpha$$

$$\neg\ (\alpha\vee\beta) \Leftrightarrow \neg\alpha\wedge\neg\beta$$

$$\neg\ (\alpha\wedge\beta) \Leftrightarrow \neg\alpha\vee\neg\beta$$

2.1.12. Propriétés des quantificateurs

Les propriétés des quantificateurs peuvent se résumer comme suit :

- ∀ x ∀ y retourne la même signification que ∀ y ∀ x.
- ∃ x ∃ y retourne la même signification que ∃ y ∃ x.
- ∃ x ∀ y ne retourne pas la même signification que ∀ y ∃ x. Par exemple :
    - ∃ x ∀ y Aime(x, y) signifie : "Il y a quelqu'un qui aime tout le monde".
    - ∀ y ∃ x Aime(x, y) signifie : "Chacun est aimé au moins par une personne".

La dualité des quantificateurs est le principe que les chaque quantificateur (existentiel ou universel) peut s'exprimer à l'aide de l'autre, par exemple :

- ∀ x Aime(x, Crème_vanille) ⇔ ¬ (∃ x ¬ Aime(x, Crème_vanille)).
- ∃ x Aime(x, Couscous) ⇔ ¬ (∀ x ¬ Aime(x, Couscous)).

2.2. Les réseaux sémantiques

2.2.1. Définition

Un réseau sémantique [23], ou réseau de trames, est une base de connaissances qui représente les relations sémantiques entre les concepts d'un réseau. Ceci est souvent utilisé comme une forme de représentation des connaissances.

Plus précisément, un réseau sémantique peut être défini comme un graphe composé :

- D'un ensemble de nœuds étiquetés : représentant généralement des Objets.
- D'un ensemble de liens orientés et étiquetés entre ces nœuds : représentant généralement des relations entre des Objets.

L'ensemble d'opérations d'exploitation de ce graphe constituent un mécanisme de raisonnement. Les réseaux sémantiques sont utilisés dans des applications de traitement du langage naturel telles que l'analyse sémantique et la désambiguïsation du sens des mots. D'un autre coté dans la robotique, les réseaux sémantiques sont souvent utilisés pour modéliser les états d'un système en vue d'élaborer des plans d'action.

2.2.2. Composants d'un réseau sémantique

a. Les nœuds

Les nœuds d'un réseau sémantique peuvent être représentés de deux manières [23] :

- Atomiques : entités élémentaires (valeurs, individus,...).
- Complexes : entités complexes (propositions, phrases,...).

Ils doivent être typés : concept, individu, action, ou proposition. Un nœud concept peut être défini comme une définition d'un ensemble, tant dis que le nœud individu est plus spécifique. Par exemple, dans un réseau sémantique reflétant l'expression « Titi est un Canari », Titi sera représenté par un nœud individu et Canari par un Nœud concept.

b. Les liens

- Structuraux : indépendants de la sémantique du domaine. Parmi les liens structuraux, on peut mettre l'accent sur [23]:
  - Le lien est_un : qui relie généralement un nœud concept a un nœud individu.
  - Le lien sorte_de : qui relie un nœud concept (général) a un autre nœud concept (plus général). Par exemple, dans l'expression « le Canari est une sorte d'Oiseau », le lien sorte_de est primordial pour relier ces deux concepts.

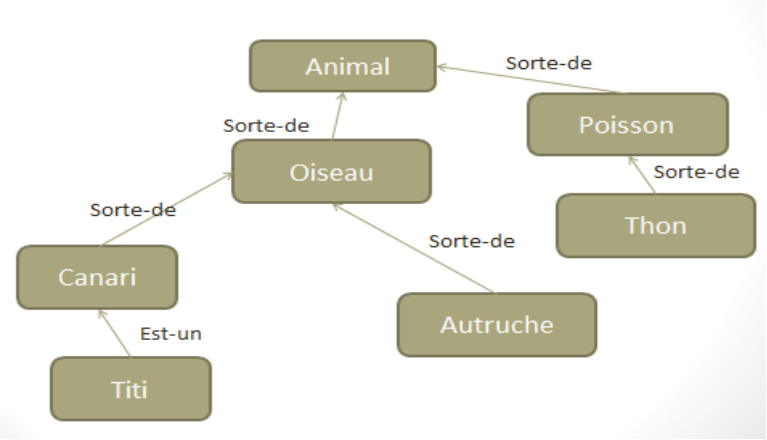


Figure 2.1. Les liens structuraux dans un réseau sémantique

- Les liens généraux tels que, Agent (d'une action), Objet (d'une action), récepteur (d'une action), et date (d'une action).

- Spécifiques : dépendants de la sémantique du domaine,

Il faut essayer d'augmenter la proportion des liens structuraux par rapport aux liens spécifiques. Par exemple, l'expression « Karim mange un pomme » peut être représentée des deux manières suivantes :

Figure2.2. Différence entre les liens spécifiques et structuraux

c. Les propriétés

Les propriétés sont des informations rattachées à chaque nœud du Réseau sémantique dans le but de renforcer le degré de connaissance reflété par le réseau. Ces propriétés sont caractérisées par leurs simplicités comme démontré dans la **figure x.** Aussi, elles permettent de répondre à des questions comme : "quel est l'âges de Titi ?" "quelle est la couleur des canaris ? "

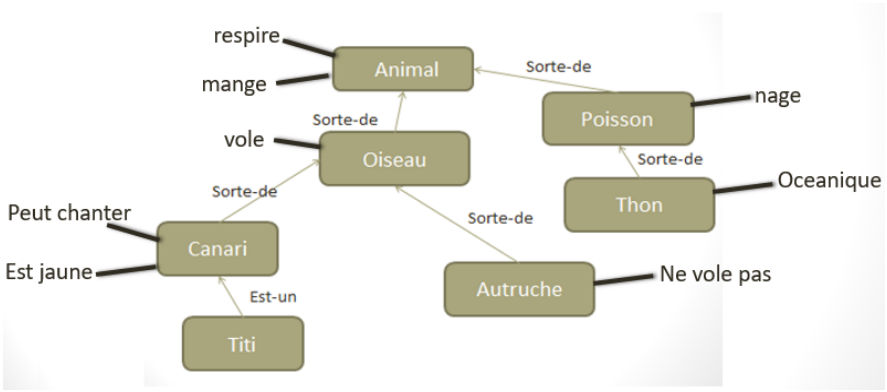
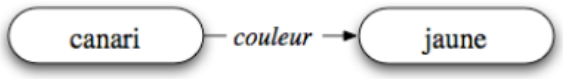


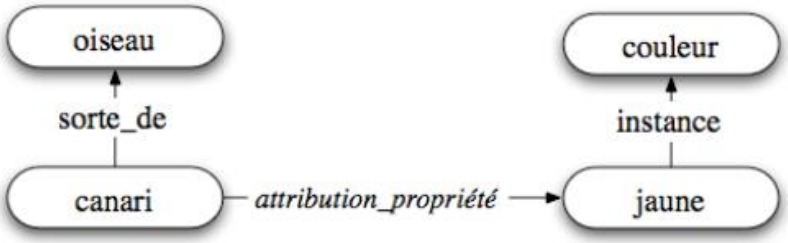
Figure 2.3. Injection des propriétés dans un réseau sémantique

d. Attributs

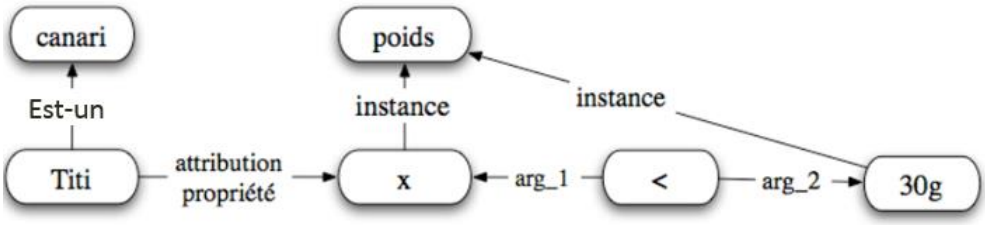
Un attribut peut être défini comme une relation qui relie un nœud concept ou un nœud individu à une valeur ou propriété. D'un autre côté, on peut le définir comme un lien spécifique dont le sens dépend du domaine d'application (interprétation ad-hoc). Par exemple, pour représenter l'expression « la couleur du canari est Jaune », il faut utiliser un lien attributs « Jaune » comme suit :



• on peut le rendre plus structurel en créant un nœud-attribut :



- Un nœud-attribut peut être relié à une ou plusieurs valeurs par l'intermédiaire d'un opérateur comme dans l'expression « Titi est canari qui pèse moins de 30g » :



Dans cet exemple, le nœud-attribut (X) qui représente le poids de Titi est interprété d'une manière ad-hoc.

2.2.3. Héritage et Partitionnement dans les réseaux sémantiques

L'héritage dans les RS [23] repose sur des liens de type « est_un » ou « sorte_de » reliant un concept à un autre concept plus élevé : exemple : canari est une *sorte d*'oiseau. Le principe d'héritage permet :

- De nombreuses déductions automatiques.
- De définir la notion de distance sémantique entre 2 concepts qui peut être défini comme le nombre de liens devant être traversés pour aller d'un concept à l'autre.

Concernant la Partition, elle peut être définie comme regroupement de nœuds et d'arcs du réseau dans des espaces spécifiant la portée de relations dans le l'intérêt de définir des contextes et de permettre la quantification.

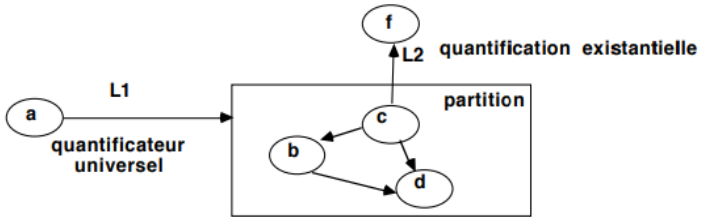


Figure 2.4. Quantification dans un réseau sémantique

Les composants de l'exemple ci-dessus peuvent être expliqués comme suit :

• cadres : définissent l'étendue des identificateurs universels

• lien L1 : quantification universelle, quelque soit (a), pointe sur un cadre représentant l'étendue de la variable quantifiée universellement.

• lien L2 : quantificateur existentiel explicite sur le nœud (f) par rapport au nœud (c).

Afin de comprendre le principe de quantification qui est traitée par la notion de partition, on prend le fait à représenter suivant : « tout chat a mordu un oiseau » basé sur l'encodage de la variable quantifiée universellement x en utilisant une partition (cadre rectangulaire) :

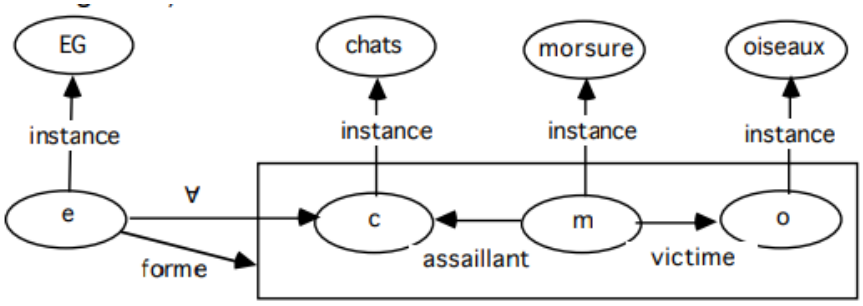


Figure 2.5. Exemple de quantification dans un réseau sémantique

- Les nœuds c, m, o sont des instances de chats, morsure, oiseaux,
- Le cadre introduit dans le réseau définit l'étendue de l'identificateur universel,

- Le nœud e représente l'assertion à représenter, instance de l'ensemble des énoncés généraux EG sur le monde,
- Chaque élément de EG possède :
  - ❖ Une connexion « forme » pointant vers le cadre de la partition et énonce l'affirmation,
  - ❖ Une ou plusieurs connexions pointant vers chaque variable quantifiée universellement, ici variable c (les variables m et o sont ici quantifiée existentiellement).

2.2.4. La logique et les réseaux sémantiques

L'approche logique de l'intelligence artificielle (IA) a comme programme de développer des méthodes permettant de représenter les connaissances des agents par des formules logiques, afin d'inférer des conclusions à partir de ces formules. Les chercheurs se sont d'abord tournés vers la logique dite classique. Le langage de cette logique met à notre disposition un jeu d'opérateurs logiques, dont les plus importants sont la négation (¬), la conjonction (/\), la disjonction {V), l'implication matérielle (⇒) et l'équivalence (⇔). Ces opérateurs ont en commun qu'ils sont tous vérifonctionnels. Par exemple la valeur de vérité de l'implication A ⇒ B est fonction de la valeur de vérité de A et de la valeur de vérité de B, puisque A ⇒ B est vrai si et seulement si A est faux ou B est vrai. Cependant, on s'est très vite rendu compte que ces opérateurs logiques vérifonctionnels ne permettent pas de parler de certains concepts qui sont importants en représentation des connaissances. Voici trois exemples de ces concepts :

- la nécessité et la possibilité.
- la connaissance et la croyance d'un agent.
- le conditionnel du type « si-alors ».

En relation aux réseaux sémantiques, seuls les opérateurs logiques de conjonction, disjonctions, et la négation peuvent être représentés :

Dans l'exemple ci-dessous, la représentation de l'expression « Gaston ira sur le lit ou sur le fauteuil » est représentée comme suit :

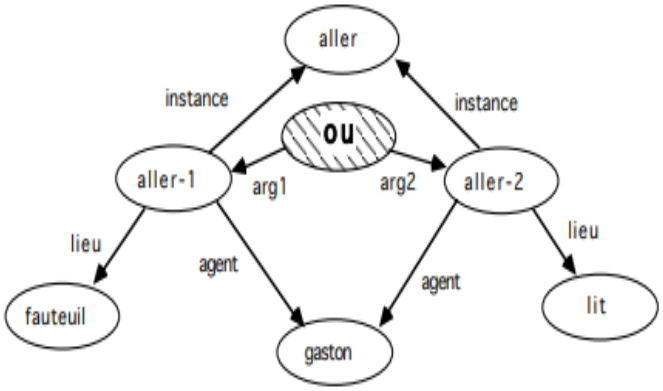


Figure 2.5. La disjonction dans un réseau sémantique

D'un autre côté, la conjonction et la négation sont reflétées comme suit dans l'expression « Marie aime Jules et n'aime pas Gaston » :

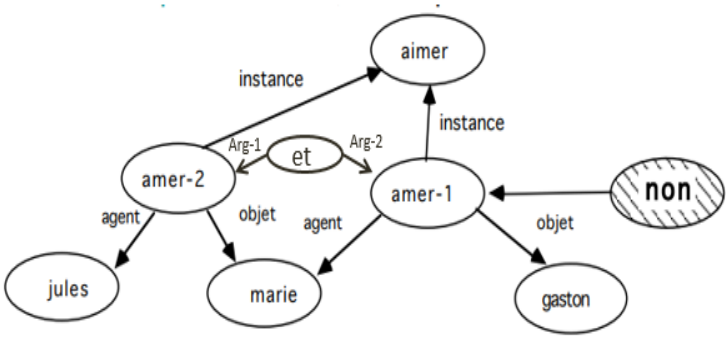


Figure 2.6. La conjonction et la négation dans un réseau sémantique

En logique mathématique, un prédicat d'un langage est une propriété des objets du domaine considéré (l'univers du discours) exprimée dans le langage en question. Afin de représenter un prédicat a travers l'utilisation des réseaux sémantiques, on se base généralement sur les liens structuraux comme le démontre l'exemple suivant :

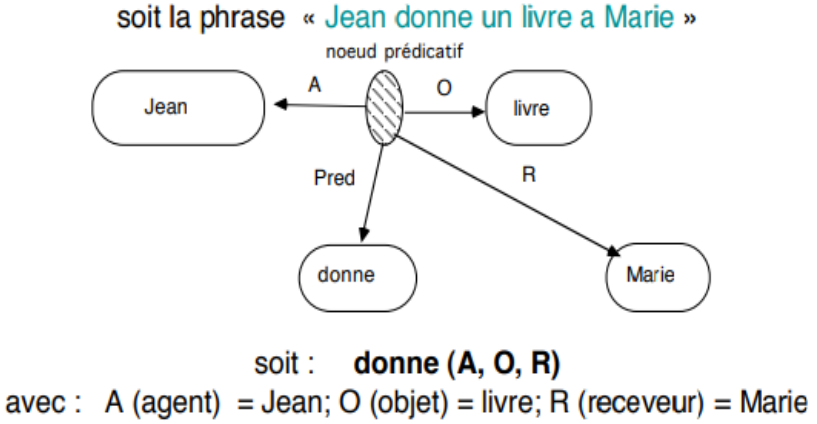


Figure 2.7. Représentation d'un prédicat dans un réseau sémantique

2.2.5. Avantages et Inconvénients des Réseaux Sémantiques

Avantages :

- Les objectifs d'extraction de connaissance dans la base de connaissances s'expriment en chemins de traversée sur la structure même de la base.
- Possèdent des principes d'organisation relativement puissants (généralisation, partition, agrégation) permettant de structurer la base de connaissances.
- Formalisme graphique : bonne compréhension, intéressant à un premier stade de formalisation de la connaissance.
- Formalisation déclarative : finesse et cohérence de représentation des concepts.

Inconvénients :

- Manque de sémantique formelle et de terminologie standard.
- Interprétation difficile des connaissances : toujours un compromis à faire entre la complexité d'une structure de données et la complexité de l'interpréteur.
- Critique si taille du réseau est importante (nb de nœuds et liens), ce qui automatiquement induit à une explosion combinatoire.

2.3. Les Graphes Conceptuels

2.3.1. Définition :

Un graphe conceptuel [24] peut être défini un formalisme de représentation de connaissances et de raisonnements. Ce formalisme a été introduit par John F. Sowa (en) en 1984. Depuis son introduction, ce formalisme a été développé en se basant sur trois axes principaux : interface graphique de la logique du premier ordre, système diagrammatique pour la logique du premier ordre, formalisme de représentation de connaissances et de raisonnement basé sur les graphes.

D'une manière plus spécifique, Un GC est un graphe bipartite orienté [24]:

- ❖ Bipartite : 2 sortes de nœuds : concept et relation conceptuelle.
- ❖ Les nœuds sont liés par des arcs orientés.
- ❖ Un arc lie toujours un concept à une relation.
- ❖ Un nœud concept peut être isolé (non rattaché).

2.3.2. Le concept et la relation conceptuelle dans un graphe conceptuel

Un concept est représenté par un couple [type, référent] :

- ref = * : concept générique (par défaut).
- ref = #i : concept individuel.
- ref = @ : mesure.

Exemple :

- [personne : *] = concept générique (un homme).
- [personne : #123] = concept individuel (cet homme).
- [personne : Paul] = concept individuel, nom propre.
- [hauteur: @173] = concept mesure (hauteur = 1,73m).

Les relations permettent de représenter des propositions portant sur des concepts. Une relation est définie par 2 éléments (r:, nom de la relation, et a : arité ou valence qui reflète le nombre d'argument de la relation). La valence peut aussi être définie comme entier n donnant le nombre d'arc (1= monoadique, 2= dyadique, … n-adique)).

On peut définir la Signature d'une relation comme l'ensemble de n types de concepts. La valence et la signature sont fixés par le type de relation. On peut aussi distinguer quelques relations conceptuelles génériques telles que :

- **Agent :** (agnt) relie [action] à [animé] : le concept animé représente l'acteur de l'action.
- **Expérienceur :** (expr) relie [état] à [animé] : le concept animé ressent cet état (Marie a chaud).
- **Instrument :** (inst) relie [entité] à [action] : entité est impliquée de manière causale.
- **Objet :** (obj) relie [action] à [entité] sur laquelle porte l'action : le chien mange un os (os = objet).

2.3.3. Les ensembles dans un graphe conceptuel :

Dans un graphe conceptuel, un nœud concept peut représenter des concepts génériques, individuels, et même des ensembles de concepts différents [24] :

Tableau 2.3 Les ensembles dans un graphe conceptuel

Type d'ensemble	Interprétation	Forme linéaire
Concept Générique	Un homme	[homme : *]
Concept individuel	Cet homme	[homme : #1]
Nom propre	Karim	[homme : Karim]

Mesure	Hauteur :1,70m	[hauteur :@1,70]
Ensemble extensif	Karim, Amine, Hamid	[homme : Karim, Amine Hamid]
Ensemble générique	Plusieurs hommes	[homme : {*}]
Cardinal imbriqué	5 hommes	[homme : {*}@5]
Définition partielle	Karim, Hamid, et autres	[homme : {Karim, Hamid, *}]
Ensemble distributif	4 hommes ont lu ces deux livres	[homme : Dist{*}@4]
Ensemble respectif	Amine ET Karim sont les maris respectifs de Habiba et Samira	[homme : Resp {Amine, Karim} @2]

Par exemple, afin de représenter l'expression « Le petit chat mange la souris », on a besoin de 4 concepts : [manger :*], [chat: #9], [petit: *], [souris : #12], et 3 relations conceptuelles : (agent), (objet), (taille). Ce graphe conceptuel peut être représenté de deux manières (forme linéaire et forme graphique) :

[manger:*]- (objet)->[souris :#12]

(agent)->[chat : #9] - (taille)->[petit :*]

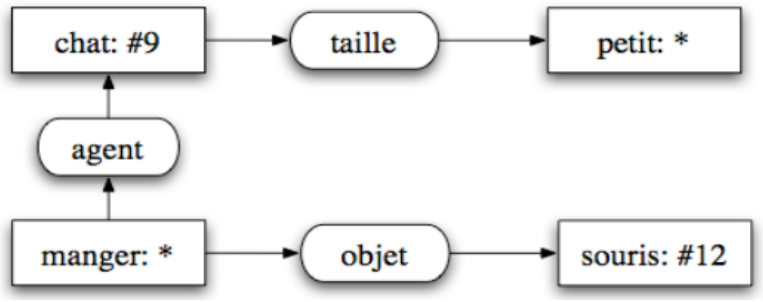


Figure 2.8. Exemple d'un graphe conceptuel

Il existe aussi un autre type de concept connu sous le nom de Contexte. Un contexte peut être défini comme un concept désignant un graphe non vide. Il peut être imbriqué comme dans la phrase « Paul croit que Marie veut se marier avec un marin » représentée comme suit :

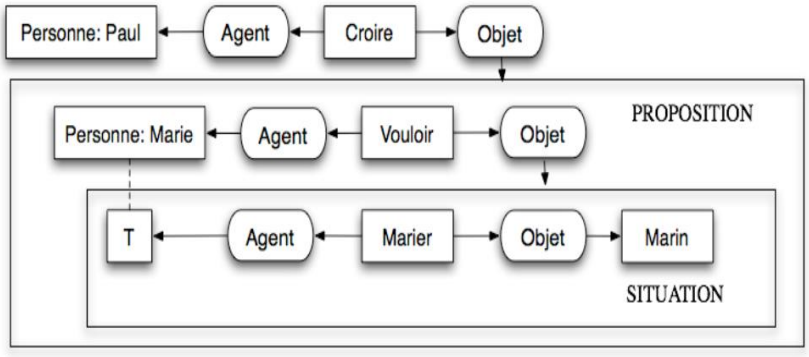


Figure 2.9. Partitionnement dans un graphe conceptuel

Forme linéaire :

[Personne: Paul]<-(Agent)<-[Croire]->(Objet)-

[Proposition: [Personne: Marie *x]<-(Agent)<-[Vouloir]->(Objet)-

[Situation: [?x]<-(Agent)<-[Marier]->(Objet)->[Marin]]].

2.3.4. Opérations dans les Graphes Conceptuels :

Généralisation et spécialisation :

Théorème : Soit u1, u2, v et w des graphes conceptuels.

- *Si u1≤v et u2 ≤ v, alors v est appelé généralisation commune de u1 et u2.*
- *Si w ≤ u1 et w ≤ u2, alors w est appelé spécialisation commune de u et u.*

Exemple :

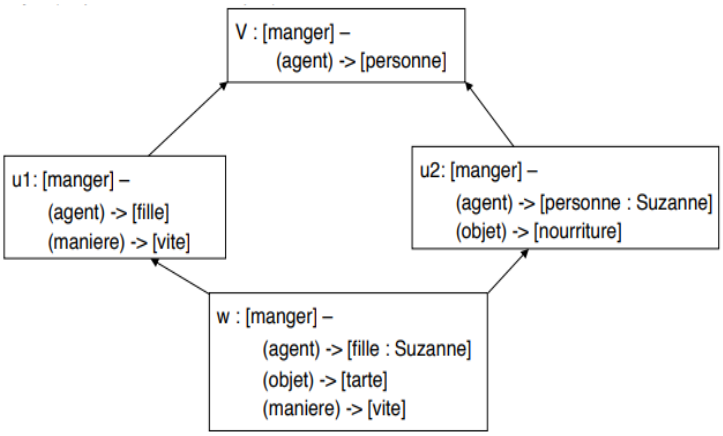


Figure 2.9. Généralisation et spécialisation dans un graphe conceptuel

Projection :

Théorème : Pour tous graphes conceptuels u et v, où u ≤ v, il existe une opération π telle que :

π : v --> u, où πv est un sous-graphe de u appelé projection de v dans u.

Exemple :

u: [aimer] –

(agent) -> [homme : Jules] <- (enfant) <- [homme : Jean]

(objet) -> [femme : Marie] <- (enfant) <- [homme : Jean]

v: [personne :*x] –> (enfant) -> [personne: *y]

projections :

πv: [homme: Jules] <- (enfant) <- [homme: Jean]

πv': [femme: Marie] <- (enfant) <- [homme: Jean]

Partie pendante :

[aimer] -> (agent)

-> (objet)

Restriction et jointure :

Théorème : La restriction d'un graphe conceptuel v se fait par rapport à un autre graphe v sachant que u ≤ v.

Exemple :

(1)[manger] –

(agent) -> [fille*]

(manière) -> [vite]

(2)[manger] –

(agent) -> [personne: Alice]

(objet) -> [tarte]

Restriction dans (2) de « personne » à « fille » :

(3)[manger] –

(agent) -> [fille: Alice]

(objet) -> [tarte]

La jointure entre les graphes (1) et (2): la jointure se fait sur les concepts identiques (après la restriction précédente) :

(4)[manger] –

(agent) -> [fille: Alice]

(objet) -> [tarte]

(manière) -> [vite]

2.3.5. Forces et Faiblesses des graphes conceptuels :

D'un côté, les graphes conceptuels sont considérés comme des puissants formalismes de représentation de la connaissance qui sont dotés d'une représentation graphique, et disposant d'une sémantique formelle (équivalence avec la logique). En rapport à la recherche scientifique, de nombreux travaux sont en cours concernant les opérateurs, et l'introduction de notions de vraisemblance, ainsi que de précision. Ils sont aussi considérés comme très efficaces en analyse du langage naturel. Un des points les plus avantageux des graphes conceptuels et la présence de nombreux outils tels que : CoGiTant, Notio (API Java), CharGer (éditeur GC). D'un autre côté, les graphes conceptuels manquent de mécanismes de

représentation par rapport aux opérations de la logique de premier ordre tels que la disjonction, problèmes de négation, Problème d'imbrication de quantificateurs.

2.4. Les frames

2.4.1. Définition

Un frame [25] est une structure de données générale permettant de décrire la connaissance que l'on a sur des objets (concrets ou abstraits). 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.
- Des informations relatives à la manipulation des objets : procédures et méthodes connues par l'objet : définition du contexte d'activation de l'objet, comment calculer une propriété, action à exécuter dans un contexte donné.

Il existe deux types de Frames :

- Les frames « prototypes » : représentent en intention une classe d'objets description du contexte attaché à cette classe
- Les frames « instances » : réalisations particulières d'une classe donnée description des individus d'une classe

Exemple:

oiseau -> classe : frame prototype

• canari -> classe : frame prototype

• titi -> individu : frame instance

2.4.2. Composants d'un frame :

- **Attribut**

Un attribut (slot) peut être défini comme une information d'un frame permettant d'introduire des propriétés (1 ou n) décrivant le frame. Dans frame prototype, un attribut permet l'introduction des valeurs permises. Tant dis que dans un frame instance, un attribut permet d'introduire des valeurs effectives. En addition, un attribut permet aussi de définir des relations (un-aire ou n-aire) entre frames.

Un attribut est structuré par des aspects. Chaque aspect introduit une valeur élément de description de l'attribut comme le montre l'exemple suivant [25]:

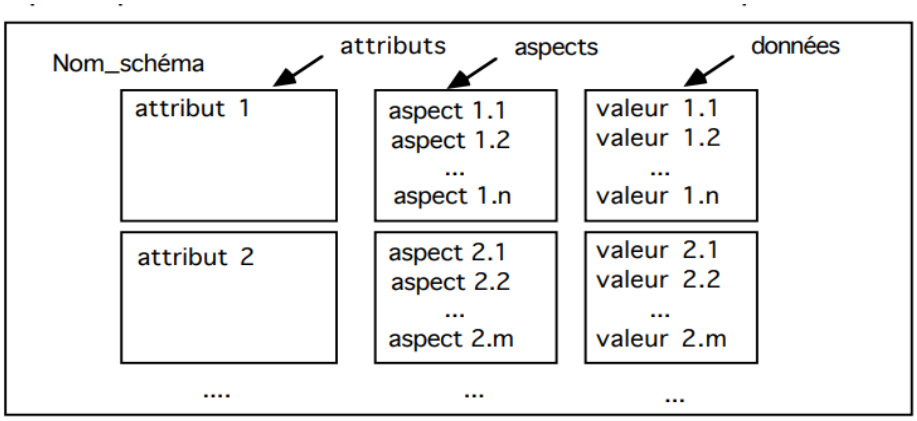


Figure 2.10. Structure d'un Frame.

Chaque attribut d'une instance doit avoir l'aspect "valeur". Le frame d'instance ne diffère du frame prototype correspondant que par la donnée supplémentaire de la valeur de l'attribut sachant qu'une instance hérite directement du frame de sa classe.

D'une manière générale, un attribut peut :

- Permettre d'introduire des valeurs effectives (frame instance).
- Être un pointeur sur un frame plus général, plus spécifique voire alternatif (vers frame prototype).
- Définir une valeur par défaut, ou définir un ensemble de valeurs permises (frame prototype).
- Constituer un démon, sorte de procédure qui peut être activée chaque fois que l'attribut doit être modifié.
- Être un pointeur sur un autre frame (récursivité de frame), ce qui permet de représenter des objets composés.

En relation a la notion d'héritage, si la valeur d'un attribut est définie au niveau d'un prototype, tous les frames (prototypes et instances) plus spécifiques en héritent. Les instances héritent des méthodes définies au niveau prototype.

❖ **Les aspects d'attributs**

Un attribut d'un frame est structuré par des aspects. Chaque aspect introduit une dimension décrivant l'attribut. L'ensemble des aspects est fixé par le langage, c'est ce qui définit la sémantique de la représentation. Ils existent 4 types d'aspects :

Aspect de typage: permet de définir le type de valeur des attributs d'un frame (valeurs permises) :

• type simple : entier, booléen, chaîne de caractères, ...

• type composé : un autre frame de telle nature.

• type intervalle : permet de définir un intervalle de valeur.

• La nono-valuation et la multi-valuation peuvent être définies respectivement par les aspects "un" et "liste_de".

Aspect d'obtention de valeur : permet de spécifier des procédures d'obtention d'une valeur :

• aspect "valeur": introduit la valeur de l'attribut.

• aspect "si_besoin": introduit une méthode de calcul de la valeur.

• aspect "défaut": introduit une valeur par défaut (traitement des exceptions).

Aspect reflexe : permet d'introduire des procédures déclenchées en cas d'ajout, de modification, de suppression d'informations, ceci notamment pour des soucis de maintien de cohérence (si_modif , si_ajout, si_supprime).

Aspect de contrôle : permet d'introduire des procédures définissant des actions à déclencher en cas d'échec ou de succès d'obtention de valeur, pouvant conduire à une propagation dans d'autres frames (si_succés , si_échec).

Dans l'exemple ci-dessous [25], on veut définir un frame prototype (Personne) contenant les attributs : nom (chaine de caractère), Age (entier), a_pour_père (pointeur vers une autre personne), grand_pere_de (pointeur vers une liste de personne), et date_de_naissance qui est aussi un attribut pointant vers un autre frame (date). Dans cet exemple, on a instancié un frame instance (Personne #1) en donnant des valeurs a tous ces attributs.

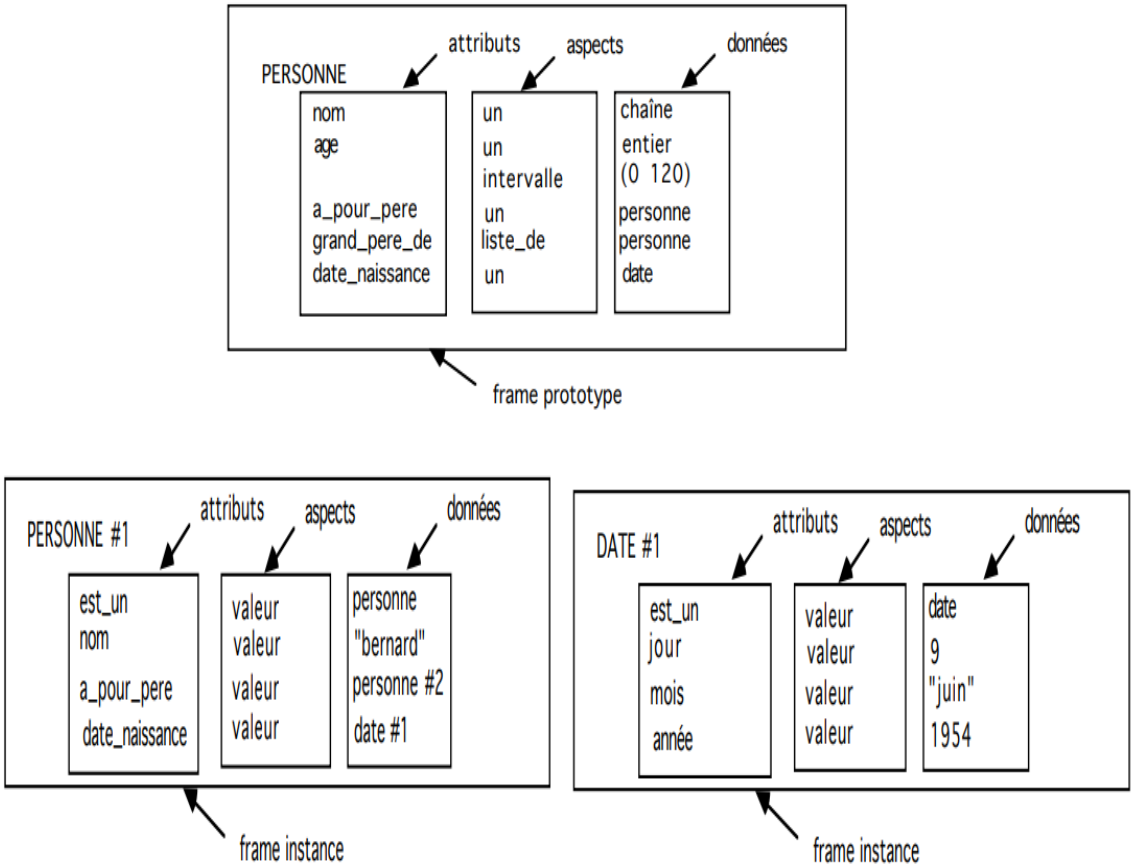


Figure 2.11. Exemple d'un Frame.

2.4.3. Héritage et inférence d'instanciation dans les frames :

Les frames s'inscrivent dans une structure de demi-treillis [25], ce qui veut dire qu'un frame peut dominer un ensemble de frames plus spécifiques et peut être dominé par un ou plusieurs frames plus généraux. D'une façon générale, si la valeur de l'attribut est définie au niveau d'un prototype, tous les frames prototypes et instances plus spécifiques en héritent. Il y a aussi l'héritage des procédures ou méthodes définies au niveau des prototypes au niveau des instances.

L'inférence d'instanciation consiste à instancier par des valeurs les attributs définis dans les frames prototypes auxquels est associé l'instance de frame considérée. Cette obtention de valeurs peut être directe (valeurs d'attributs propres à l'instance), ou par héritage. Si l'on appelle "vue" l'ensemble des attributs que possède un frame, une instance de ce frame sera l'union de toutes les vues propres et héritées.

2.4.4. Forces et faiblesses des frames :

Les avantages des frames peuvent être résumés comme suit :

- Très séduisant et pratique pour exprimer la connaissance.
- Ils sont souvent Utilisés pour le développement d'ontologies (Protégé - Frame).
- Ils sont aussi utilisés dans la réalisation de nombreux systèmes experts.

Néanmoins, ils ont des inconvenants qui peuvent être résumés comme suit :

- Pas d'équivalence avec la logique (ex : quantifieurs, disjonctions, …).
- Pas d'expression de connaissances incertaines, imprécises, hypothétiques.
- Certaines inférences sont favorisées, d'autres sont difficiles à réaliser.

2.5. La logique de description

2.5.1. Définition et objectif de la logique de description :

Les Logiques de Description (LD) [26] sont des langages de représentation des connaissances mettant l'accent sur le raisonnement. L'objectif majeur de la LD est de raisonner efficacement (temps de réponse minimal) pour la prise de décision. Elle peut aussi être définie comme une approche ontologique. En d'autres termes, pour représenter la connaissance d'un domaine, les LD demande la définition des catégories générales d'individus, ainsi que des relations logiques que les individus ou catégories peuvent entretenir entre eux.

Cette approche ontologique est naturelle pour le raisonnement. Si la majorité des interactions se déroulent au niveau des individus, la plus grande partie du raisonnement se fait au niveau des catégories. La LD a de Nombreuses correspondances avec les autres formalismes de la représentation des connaissances. Par exemple, les catégories générales d'objets et de relations fait partie de l'héritage dans les frames et réseaux sémantiques.

L'objectif majeur de la logique de description est de trouver un compromis entre la possibilité de raisonnement et la suffisance de l'expressivité :

Tableau 2.4 Raisonnement VS Expressivité

Raisonnement possible ?	Expressivité suffisante ?
Décidabilité / complexité du raisonnement.	Des concepts pertinents doivent être définissables.
Nécessite un langage de description restreint.	Des domaines d'application nécessitent des DL très expressives.

2.5.2. Les deux niveaux de description :

Les LD permettent de représenter la connaissance d'un domaine au travers des concepts (ou classes) du domaine ainsi que des relations (ou rôles) pouvant être établies entre ces classes ou entre les instances de ces classes appelées individus.

La modélisation des connaissances avec les LD se fait à 2 niveaux :

- Le niveau terminologique ou TBox, qui décrit les connaissances générales d'un domaine, et qui définit les concepts (classes) et les rôles (relations).
- Le niveau factuel ou ABox, qui décrit les individus en les nommant et en spécifiant leur classes et attributs (en termes de concepts et de rôles). La ABox sert aussi à spécifier des assertions portant sur ces individus nommés.

Dans la LD, plusieurs ABox peuvent être associées à une même TBox. Chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour les exprimer [26].

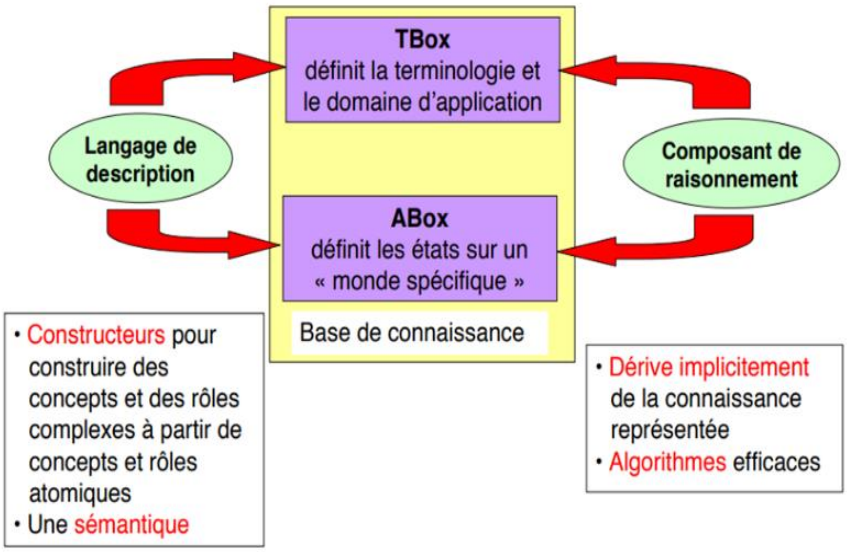


Figure 2.12. Structure de la logique de description

- **La TBox : entités atomiques et composées**

La TBox est composée des Entités Atomiques qui sont définies comme des concepts atomiques et rôles atomiques constituant les entités élémentaires de la LD. Les concepts et rôles atomiques prédéfinis minimaux sont :

- Le concept $\top$ et le rôle $\top_r$, les plus généraux de leur catégorie.
- Le concept $\bot$ et le rôle $\bot_r$, les plus spécifiques (ensemble vide).

Les concepts et rôles atomiques peuvent être combinés au moyen de constructeurs pour former des entités composées afin de former les entités composées.

Syntaxe :

- A et B dénotent des concepts atomiques.
- D dénotent des concepts composés.
- R dénote un rôle.
- Les noms de concepts commencent par une Majuscule : Ex : Homme, Femme.
- Les noms de rôles par une minuscule : Ex : relationParentEnfant.
- Entité composée : MereAvecPlusieursEnfants ≡ Mere ⊓ ≥3 aEnfant.

Les constructeurs permettent la combinaison de concepts et rôles atomiques pour former des entités composées. Par exemple, le concept composé (Mâle ⊓ Femelle) résulte de l'application du constructeur ⊓ aux concepts atomiques Mâle et Femelle, et s'interprète comme l'ensemble des individus appartenant aux concepts Mâle et Femelle.

Les LD se distinguent par les constructeurs qu'elles proposent. En d'autres termes, plus elles ont de constructeurs, plus elles sont expressives, et ont des chances d'être non décidables ou de complexité très élevée. D'autre part, les LD trop peu expressives ne permettent pas de représenter des domaines complexes.

Les axiomes terminologiques d'une des 2 formes :

- C ≡ D : énonçant des relations d'équivalence (de définition) entre concepts : C équivalent par définition à D.
- C ⊑ D : énonçant des relations d'inclusion : C est inclus dans D.

❖ **Caractéristiques de la TBox**

La Consistance : un concept (une classe) est consistant, s'il existe au moins un individu membre de cette classe. Par exemple, si on définit une classe (concept) comme étant à la fois une sous-classe des classes Homme et Femme, et que la TBox spécifie aussi que ces 2 classes sont disjointes (aucun individu ne peut à la fois être un Homme et une Femme), ce nouveau concept est alors inconsistant.

La subsomption : la subsomption consiste à déduire qu'un concept, c'est à dire une classe, est une sous-classe d'une autre classe, même si ce n'est pas déclaré explicitement dans la base de connaissances. Par exemple, si on spécifie qu'Humain est une sous-classe d'Animal, que

Mère est une sous classe d'Humain, on peut déduire qu'une Mère est une sous-classe d'Animal : Mère ⊑ Animal.

L'interprétation : tout concept est associé à un ensemble d'individus dénotés par ce concept. Une interprétation I suppose l'existence :

- D'un domaine d'interprétation Δ ou Δ^I, ensemble non vide représentant les entités du monde décrit et composé d'individus.
- D'une fonction d'interprétation I assignant à chaque concept atomique A, un ensemble A^I, tel que $A^I \subseteq \Delta^I$. Cette fonction assigne aussi à chaque rôle atomique R, une relation binaire R^I, telle que $R^I \subseteq \Delta^I \times \Delta^I$.
- L'interprétation I satisfait un axiome d'équivalence $C \equiv D$ si $C^I = D^I$(égalité des ensembles d'individus).
- L'interprétation I satisfait un axiome d'inclusion $C \sqsubseteq D$ si $C^I \subseteq D^I$ (inclusion des ensembles d'individus).
- L'interprétation I satisfait une TBox T si I satisfait tous les axiomes de la TBox T(on dit que I est un modèle de la TBox T).

❖ **La ABox : assertions d'appartenance et de rôle**

Une ABox contient 2 types d'assertions sur des individus :

- Des assertions d'appartenance : spécifiant leur classe et leurs attributs. Par exemple, Marie est une femme et elle a 2 enfants, Marie est une Mère (individu instance de la classe mère).
- Des assertions de rôle : spécifiant les relations existantes entre individus. Par exemple, une mère doit avoir au moins un enfant : la ABox devra contenir au moins un autre individu, et une relation entre celui-ci et Marie indiquant qu'il est un de ses enfants.

Soient la Tbox et la ABox suivantes [26]:

Tableau 2.5. Exemple TBox, ABox

TBox	ABox
1. Femme ≡ Personne ⊓ Femelle	1. MereSansFille(Marie)
2. Homme ≡ Personne ⊓ ¬ Femelle	2. Femme(Alice)
3. Mere ≡ Femme ⊓ ∃aEnfant.Personne	3. Pere(Pierre)
4. Pere ≡ Homme ⊓ ∃aEnfant.Personne	4. aEnfant(Marie,Pierre)
5. Parent ≡ Pere ⊔ Mere	5. aEnfant(Marie,Paul)
6. GrandMere ≡ Mere ⊓ ∃aEnfant.Parent	6. aEnfant(Pierre,Alice)

7. MereAvecPlusieursEnfants ≡ Mere ⊓ ≥3 aEnfant 8. MereSansFille ≡ Mere ⊓ ∀aEnfant.¬Femme 9. Epouse ≡ Femme ⊓ ∃aCommeMari.Homme	

Peut-on déduire GrandMere(Marie) ?

Réponse : <u>oui,</u> grâce aux règles 1 et 4 de la ABox, on peut déduire que Marie est Mère qui a un enfant (Pierre), et grâce à la règle 6, on peut déduire que pierre a une fille (Alice). Donc, selon la règle 6 de la TBox, on peut déduire que Marie est grand-mère.

2.5.3. Syntaxe de la logique de description :

Soit :

- C et D des concepts atomiques ou complexes.
- R une relation (rôle).
- ⊤ : le concept universel.
- ⊥ : le concept impossible (ou le plus spécifique).

Les constructeurs peuvent être exprimés dans le tableau ci-dessous :

Tableau 2.6. Constructeurs de la logique de description

¬ C C ⊔ D	Non C ou Complément de C L'union de concepts (C ou D)
C ⊓ D	L'intersection de concepts (C et D)
∃R.C	La quantification existentielle (existential restriction) *
∀R.C	La quantification universelle (universal restriction)

- Le constructeur ¬C : négation (complément) d'un concept désignant (pour une interprétation), l'ensemble des individus n'appartenant pas au concept atomique C. Par

exemple, soit le concept Humain représentant l'ensemble des humains, ¬Humain représente l'ensemble des individus qui ne sont pas des humains.

- Le constructeur C ⊔ D : disjonction (union) de 2 concepts composés désignant (pour une interprétation), l'ensemble des individus membres soit du concept C ou soit du concept D. Par exemple, Etudiants ⊔ Enseignant représentant l'ensemble des individus qui sont étudiants OU enseignants.
- Le constructeur C ⊓ D : conjonction (intersection) de 2 concepts composés désignant (pour une interprétation) l'ensemble des individus membres à la fois du concept C et du concept D. Par exemple, Etudiants ⊓ Male représentant l'ensemble des individus qui sont étudiants ET males.
- L'ensemble ⊤ : désigne (pour une interprétation) l'ensemble de tous les objets/individus. Par exemple, ∀aEnfant.⊤ détermine les personnes qui ont des enfants.
- L'ensemble ⊥ : désigne une interprétation vide. Par exemple, ∀aEnfant. ⊥ détermine les personnes qui n'ont pas d'enfants.

2.5.4. Correspondance entre LA LD et la logique des prédicats du premier ordre

Une correspondance existe entre la LD et la logique des prédicats du premier ordre [26]:

- Un concept atomique A correspond à un prédicat unaire A(x).
- Un rôle R à un prédicat binaire R(x, y).
- Un individu correspond à une constante.
- Un concept composé à une formule avec une variable libre C(x).

2.6. Exercices

Exercice 01(logiques des prédicats)

a. Bien que personne ne fasse de bruit, Jean n'arrive pas à se concentrer.

b. Si personne ne fait de bruit, Jean répondra au moins à une question.

c. Tout le monde a menti à quelqu'un dans sa vie.

d. Tous les étudiants, sauf Jean, sont présents.

e. Aucun enfant ne fait jamais aucune bêtise.

f. Tout le monde a lu un livre de logique.

g. Certains enfants ne sont pas malades.

h. Tous les éléphants ont une trompe.

i. Tous les hommes n'aiment pas Marie.

j. Il y a une chanson qu'aucun enfant ne chante.

k. Si tous les hommes aiment Marie, alors elle est contente.

l. Tous les fermiers apprécient un ministre.

Exercice 02 (Réseaux sémantique)

Construire un réseau sémantique représentant :

- ÉVA, élève de CE1, 8 ans
- Christophe, élève de 6ème, 12 ans
- Éric, élève de CE2, 9 ans
- Anne, élève de 6ème, 13 ans
- Sophie, élève de 4ème, 13 ans

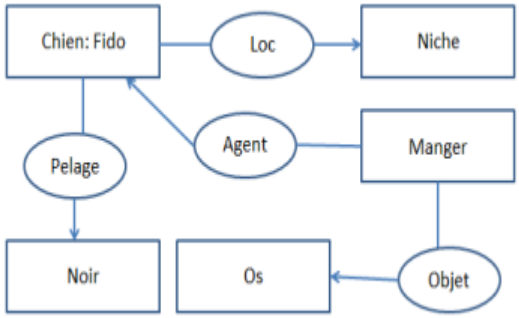
Utiliser le concept ''PERSONNE'' et éventuellement des sous-concepts.

2- Quelles propriétés des enfants obtient-on par inférence ? par héritage de propriétés(s) ?

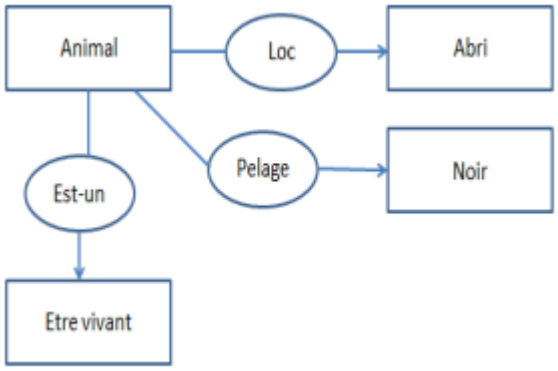
3- Construire le réseau sémantique permettant de poser la question « Quelles sont les filles qui sont en 6ème » et le réseau sémantique permettant d'y répondre pour la situation.

Exercice 03 (graphe conceptuel)

Soient les graphes 1et 2 :



Graphe 1



Graphe 2

1- Déterminez les concepts types et les concepts sous types (Restriction) entre G1 et G2.
2- Quelle est la relation entre G1 et G2 ?
3- Appliquez cette opération au graphe G2, le graphe résultant sera nommé G3.
4- Appliquez l'opération de jointure entre G1 et G3, pour obtenir G4.

Exercice 04 (Frames)

Représentez le texte suivant à l'aide des frames :

Soit le frame animal qui est défini comme un être-vivant qui a entre 0 et 4 pattes, et qui a la capacité de marcher, nager, et voler dans certains cas. Un oiseau est un animal qui a 2 pattes qui vole et marche, mais ne nage pas. Un oiseau pèse entre 20 et 30 grammes, et peut avoir plusieurs couleurs. Enfin un oiseau est caractérisé d'une race précise, et s'il ne peut pas voler il s'assoit sur une branche. Titi est canari jaune et blanc qui pèse 25 grammes.

Exercice 05 (logique de description)

Soient les Concepts suivants :

Véhicule, Bateau, Bicyclette, Voiture, Engin, Moteur, Axe, Rotation, Eau, Humain, Conducteur, Adulte, Enfant ; Roue

Soient les Rôles suivants :

a-unePartie, alimenter-Par, voyager-En, contrôler, capable-De

Représentez les expressions suivantes en logique de description :

1- Les voitures sont exactement ces véhicules qui ont des roues et sont alimentés par un moteur.
2- Les vélos sont exactement ces véhicules qui ont des roues et sont alimentés par un humain.
3- Les bateaux sont exactement ces véhicules qui voyagent sur l'eau.
4- Les bateaux n'ont pas de roues
5- Les voitures et les vélos ne voyagent pas sur l'eau.
6- Les roues sont exactement ces engins qui ont un axe et sont capables de rotation.
7- Les conducteurs sont exactement ces humains qui contrôlent un véhicule.
8- Les conducteurs sont des adultes.
9- Les humains ne sont pas des véhicules.
10- Les roues et les moteurs ne sont pas des êtres humains.
11- Les humains sont des adultes ou des enfants.
12- Les adultes ne sont pas des enfants.
13- Bob contrôle une voiture.
14- Bob est un humain.
15- Bob contrôle QE2.
16- QE2 est un véhicule qui se déplace sur l'eau.

2.7. Correction des exercices

Solution exercice 01 (logique des prédicats)

a. $(\forall x(P(x) \rightarrow \neg B(x)) \wedge \neg C(j))$

b. $(\forall x(P(x) \rightarrow \neg B(x)) \rightarrow \exists y(Q(y) \wedge R(j, y)))$

c. • a la même personne $\exists y \forall x((P(x) \wedge P(y)) \rightarrow M(x, y))$

• pour chaque personne, il y a quelqu'un a qui... $\forall x \exists y((P(x) \wedge P(y)) \rightarrow M(x, y))$

d. $\forall x((E(x) \wedge x \neq j) \rightarrow P(x))$

e. $\forall x(E(x) \rightarrow \exists y(B(y) \wedge F(x, y)))$

f. $\forall y \exists x((LdL(x) \wedge P(y)) \rightarrow L(y, x))$

g. $\exists x(E(x) \wedge \neg M(x))$

h. $\forall x(E(x) \rightarrow T(x))$

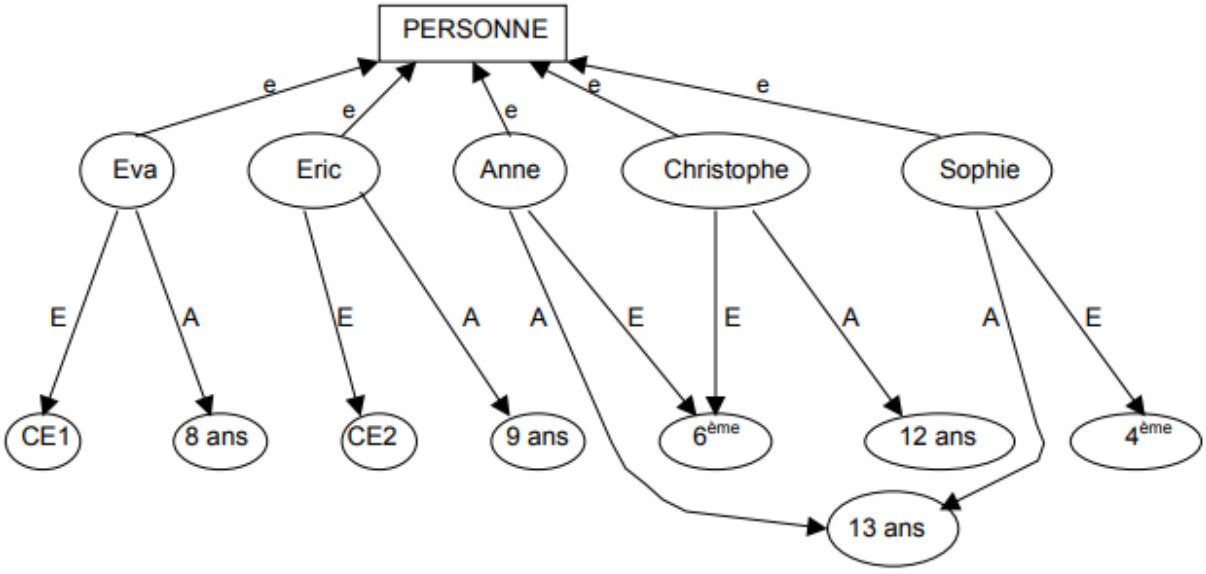
i. $\forall x(H(x) \rightarrow \neg A(x,m))$

j. $\exists x \forall y((C(x) \wedge E(y)) \rightarrow \neg C(y, x))$

k. $(\forall x(H(x) \rightarrow A(x,m)) \rightarrow C(m))$

l. $\forall x \exists y((F(x) \wedge M(y)) \rightarrow A(x,y))$

Solution exercice 02 (Réseau sémantique)



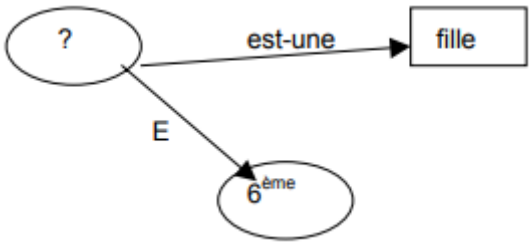
e : est-une

E : élève-en

A : âgé(e)-de

2- par inférence, on peut obtenir la classe ainsi que les âges des enfants. Par héritage, on peut déduire qu'ils sont des personnes.

3-Réseau sémantique permettant de répondre à la question « Quelles sont les filles qui sont en 6 eme ? »



Solution exercice 03 (graphe conceptuel)

1-

Concept type	**Concept sous-type**
[Animal]	[Chien : Fido]
[Abri]	[Niche]

2- G2 est une généralisation de G1, et G1 est une restriction de G2.

3-G3 :

[Chien : fido]-(loc)->[Niche]

-(est-un)->[être-vivant]

-(Pélage)->[Noir]

4-G4 :

[Manger]-(Objet)->[Os]

-(Agent)-> [chien : fido]-(loc)->[Niche]

-(est-un)->[être-vivant]

-(Pélage)->[Noir]

Solution exercice 04 (frames) :

Frame Animal :

Sorte-de	Valeur	Etre-vivant
Patte	Intervalle	[0, 4]
Marcher	Un	Booléen

Nager	Un	Booléen
Voler	Un	Booléen

Frame Oiseau :

Sorte-de	Valeur	Animal
Patte	Valeur	2
Marcher	Valeur	True
Nager	Valeur	False
Voler	Valeur	True
	Si-échec	Assoir()
Poids	Intervalle	[20g, 30g]
Couleurs	Liste-de	couleur
Race	Un	Chaine de caractère

Frame titi :

Est-un	Valeur	Oiseau
Race	Valeur	« Canari »
Poids	Valeur	25g
Couleurs	Valeur	« jaune », « blanc »

Solution exercice 05 (logique de description)

1-Voiture ≡ Véhicule ⊓ ∃a-unePartie.Roue ⊓ ∃alimenter-Par.Moteur

2- Bicyclette ≡ Véhicule ⊓ ∃a-unePartie.Roue ⊓ ∃alimenter-Par.Humain

3- Bateau ≡ Véhicule ⊓ ∃voyager-En.Eau

4- Bateau ⊑ ∀a-unePartie.¬Roue

5-Voiture ⊔ Bicyclette ⊑ ∀voyager-En.¬Eau

6-Roue ≡ Engin ⊓ ∃a-unePartie.Axe ⊓ ∃capable-De .Rotation

7- Conducteur ≡ Humain ⊓ ∃controler.Véhicule

8-Humain ⊓ ∃controler.Véhicule ⊑ Adult

9-Humain ⊑ ¬Véhicule

10-Roue ⊔ Moteur ⊑ ¬Humain

11-Humain ⊑ Adulte ⊔ Enfant

12-Adulte ⊑ ¬Enfant

12-Bob :(∃controler.Voiture)

13-Bob : Humain

14-(Bob, QE2) : controler

15-QE2 :(Véhicule ⊓ ∀voyager-En.Eau)

Chapitre 03 : Les systèmes inférentiels

Objectifs du chapitre

Les objectifs du chapitre 3 peuvent être résumés comme suit :

- *Définir un système expert d'une manière élargie.*
- *Avoir une idée précise sur les composants d'un système expert ainsi que leurs relations.*
- *Détailler la structure d'une règle d'inférence ainsi que la condition de son utilisation.*
- *Comprendre les stratégies utilisées afin de sélectionner une règle d'inférence.*
- *Assimiler le comportement des algorithmes d'inférences ainsi que les différences entre eux.*
- *Lever l'ambigüité sur les différentes phases du cycle de vie d'un moteur d'inférence.*

3.1. Définition d'un système expert

Selon Edward Feigenbaum [16], un système expert est défini comme suit : « Un programme informatique intelligent qui utilise des connaissances ainsi que des procédures d'inférences afin de résoudre des problèmes assez complexes ayant besoin d'une importante intervention d'un expert humain pour leurs trouvez des solutions ».

Autrement, un système expert peut être défini comme un outil capable de reproduire les mécanismes cognitifs d'un expert humain, dans un domaine spécifique tels que la médecine, ou le service bancaire. Les systèmes experts sont considérés comme l'une des voies tentant d'aboutir à l'intelligence artificielle.

Plus précisément, un système expert est un logiciel qui est capable de donner des réponses à des questions, en réalisant un raisonnement à partir de faits ainsi que des règles connues. Un système expert peut aussi servir notamment comme un outil d' aide à la décision. Les objectifs d'un système expert peuvent être résumés comme suit :

- La modélisation du comportement de l'être humain (expert humain).
- La résolution des problèmes complexes.
- Procurer une explication sur les raisonnements (système interprétable).

3.1.Composants d'un système expert

3.1.1. La base de connaissances :

Une base de connaissance concerne le regroupement des connaissances spéciales à un domaine spécifique donné. Ces connaissances doivent être sous une forme exploitable, et numérique. Elle peut être composée de règles, ce qui signifie qu'on parle de base de règles, des faits, ou encore d'autres représentations, ce qui veut dire l'ensemble des propriétés générales de l'existant.

La question qui se pose est comment décrire ou clowner le comportement d'un expert humain lorsqu'il est entrain de traiter problème particulier, et sa manière de le solutionner. L'objectif d'un système expert est l'obtention de l'expérience, ainsi que la connaissance pratique de l'expert, et non pas la théorie que l'on peut trouver dans la littérature ni exclusivement les règles logiques d'inférence. Si les méthodes de manipulation de faits et de règles sont nombreuses et connues, la définition de l'ensemble des faits et règles qui vont permettre de composer la base de connaissances est un problème difficile. Les méthodes d'acquisition des connaissances sont développées en équivalence avec les méthodes d'analyse de l'informatique traditionnelle.

Plus techniquement, une base de connaissances est composée de :

- ✓ **Une base de faits**

Une base de faits peut être considérée comme l'une des entrées d'un moteur d'inférence. Elle est constituée d'un ensemble de connaissances appelées "faits" qui sont considérés comme vrais. À partir de ces faits, le moteur d'inférence va pouvoir appliquer les règles issues de sa base de règles afin de déduire d'autres faits et ainsi trouver une solution a un problème de logique.

Exemple :

- Amine est le père de Karim
- Hamid est le frère de Salim
- Nabila est la mère de Sami

- ✓ **Base de règles**

La base de règles stocke la connaissance sous forme de règles : « SI ... ALORS ... » afin de maximiser l'approximation en relation à la façon naturelle de la représentation des connaissances. La partie entre le SI et le ALORS est connue sous le nom de la partie

prémisses de la règle. On peut la définir comme une conjonction d'expressions qui doivent être vérifiées pour que la règle soit déclenchable. La partie qui suit le ALORS est connue sous le nom de la partie conclusion. Elle peut être définie comme une conjonction d'actions.

Exemple :

- Si x est frère de y alors la mère de x est aussi la mère de y.
- Si x est frère de y alors le père de x est aussi le père de y.

3.1.2. Moteur d'inférence

Le moteur d'inférence [27], du verbe « inférer » qui a la même signification que « déduire », est capable de générer de nouveaux faits à partir faits et règles qui existent déjà. Ce processus est répété jusqu'à parvenir à la réponse concernant la question experte posée. Il existe de plusieurs types de moteur, pouvant faire le traitement sur différentes formes de règles logiques afin de déduire de nouveaux faits à partir de la base de connaissance. On peut distinguer trois catégories de moteurs, basées sur la manière avec laquelle les problèmes sont traités :

- Les moteurs connus comme « chaînage avant » : qui démarrent a partir des faits et règles contenus dans la base de connaissance, dans le but de tenter de s'approcher des faits recherchés par le problème (faits objectifs).
- Les moteurs connus comme « chaînage arrière » : à l'inverse du chainage avant, le chainage arrière part des faits recherchés par le problème, afin de tenter par l'intermédiaire des règles, de « remonter » à des faits qui appartiennent a la base de fait.
- Les moteurs connus comme « chaînage mixte » : ce troisième type de moteur est basé sur une combinaison des deux approches chaînage avant et chaînage arrière.

Certains moteurs d'inférence peuvent être pilotés ou contrôlés d'une manière partielle par des métarègles qui réajustent leur fonctionnement ainsi que leurs modalités de raisonnement.

Dans un système expert à base de règles, le chaînage avant est une méthode de déduction qui est basée sur l'application des règles en démarrant des symptômes afin de déduire de nouvelles conclusions. Ces faits déduis permettent d'enrichir la mémoire de travail et peuvent devenir les prémisses d'autres règles. Contrairement au chainage avant, le chainage arrière démarre des conclusions dans le but d'essayer de « remonter » aux axiomes.

La figure ci-dessous explique le principe du moteur d'inférence.

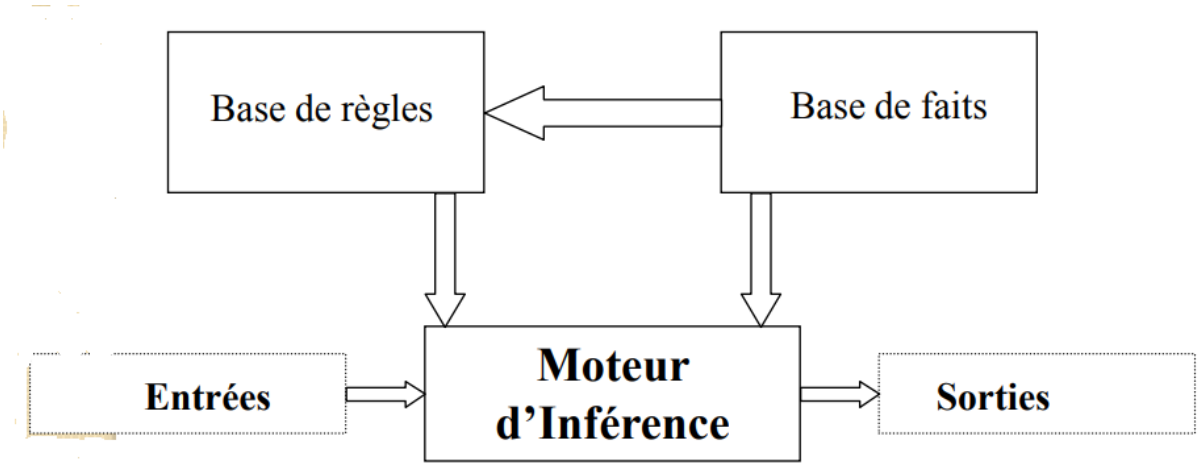


Figure 3.1. Composants d'un système expert

A partir de cette figure, on peut déduire qu'un moteur d'inférence permet aux systèmes experts de conduire des raisonnements logiques et de dériver des conclusions à partir d'une base de connaissances (base de faits et d'une base de règles).

3.2. L'interface utilisateur

Les systèmes experts communiquent avec les utilisateurs humains ainsi qu'avec d'autres systèmes logiciels et matériels. Les systèmes experts communiquent avec les utilisateurs humains via une interface utilisateur final. Le but de l'interface utilisateur final est d'obtenir des informations sur le problème de l'utilisateur final et d'afficher des solutions. Pour obtenir des informations, l'interface peut afficher des questions sur un terminal et inviter l'utilisateur final à obtenir des réponses. Les solutions peuvent consister en des déclarations textuelles. Des interfaces d'utilisateur final plus élaborées peuvent utiliser des graphiques et de l'hypertexte.

Une fonction utile d'un système expert est la capacité d'expliquer ses actions. Lors de l'utilisation d'un système expert, l'utilisateur final peut souhaiter savoir pourquoi des questions sont posées ou pourquoi certains faits ont été conclus. Lorsque la solution est présentée à l'utilisateur final, l'utilisateur peut demander une explication sur la manière dont la solution a été atteinte. L'interface utilisateur final contient des procédures qui génèrent des explications qui peuvent être présentées à l'utilisateur final.

Dans de nombreuses applications pratiques, un système expert doit s'interfacer et échanger des données avec d'autres systèmes logiciels et matériels. Le nombre de systèmes experts qui ont des utilisateurs non humains, tels que d'autres systèmes logiciels ou dispositifs de contrôle de processus électroniques, est en augmentation. Par conséquent, il est important de noter que le système expert s'interface à la fois avec l'utilisateur final et avec d'autres composants non humains.

3.3. Le raisonnement

Le raisonnement peut être défini comme un processus intelligent permettant l'obtention de nouveaux résultats ou la vérification d'un fait à travers l'appel de différentes "lois" ou expériences, indépendamment de leur domaine d'application : système qualité, système d'informations, système industrielle, système juridique, système pédagogique, ou même mathématique.

On effectue des raisonnements dans le but d'atteindre des objectifs différents, qui peuvent se combiner :

- La prise de décision.
- Le test d'une argumentation.
- Effectuer une démonstration, un théorème, ou même la confirmation d'une hypothèse.

On dit que l'individu conduit des inférences et que le mécanisme d'élaboration de ces derniers est dit raisonnement. Un raisonnement qui se base davantage sur des règles formulées en langage mathématique sera catégorisé plutôt comme raisonnement rationaliste. Ce type de raisonnement est beaucoup plus prépondérant dans les systèmes avec modèle physique. D'un autre côté, Un raisonnement qui s'appuis davantage sur des expériences vécues sera catégorisé plutôt comme raisonnement empirique. Ce type de raisonnement, qui n'exclut pas la rigueur, même dans la méthode expérimentale, est prépondérant dans les systèmes qui ne s'appuient pas sur un modèle physique.

Descartes [28] affirmait : « Il n'y a pas d'autres méthodes qui s'offrent aux hommes, pour arriver à une connaissance certaine de la vérité, que l'intuition évidente et la déduction nécessaire ». Dans cette affirmation, il admettait l'importance de l'intuition. Pourtant, tant qu'une intuition n'est pas partagée elle n'est pas forcément évidente. La logique générale se base sur la tradition des syllogismes.

Dans la logique mathématique [19], on s'accorde à considérer trois moyens de construction de raisonnements :

- La déduction : La déduction est un considérée comme type raisonnement qui a pour principe de tirer à partir d'une ou de plusieurs propositions, une autre qui en est la conséquence nécessaire. En d'autres termes, le raisonnement déductif c'est extraire du particulier à partir de l'universel.
- L'induction : L'induction est un type de raisonnement basée sur le principe de généralisation des cas particuliers. A partir d'un phénomène observé de manière récurrente, on va induire une règle ou une loi générale, sans passer par vérification de tous les exemples. En d'autres termes, L'induction consiste a extraire l'universel du particulier.
- L'abduction : le raisonnement abductif peut se voir comme un processus qui permet de donner l'explication d'un phénomène ou une observation en partance de certains faits, événement ou lois. Selon Pierce, Seul ce processus permet d'aboutir à l'universel. Le raisonnement hypothético-déductif est considéré comme une forme d'abduction. Durant la phase de recherche de solutions a un problème, les élèves sont sollicités afin de formuler des « hypothèses ». Ceci reflète le raisonnement abductif en mathématiques.

3.4. Règle d'inférence

Une règle d'inférence est constituée d'une prémisse qui implique une conclusion [29]. Les prémisses et les conclusions sont composées de faits. Sachant qu'une conclusion peut être une prémisse d'une autre règle. Dans ce cas, pour valider la conclusion, il faudra passer par la validation de sa prémisse. Autrement, un fait ne peut pas appartenir a la prémisse ainsi qu'a la conclusion de la même règle d'inférence.

Ainsi, une règle peut être vue comme une composition de faits liés avec des opérateurs logiques (des conjonctions). Les règles représentent les connaissances opératoires qui déclenchent le processus de déduction dans un système expert. Ces règles déductives sont appelées règles de production. Les règles permettent la déduction d'autres faits qui peuvent déclencher d'autres règles. Les règles sont mises en point par un cognicien en collaboration avec l'expert humain suite à son expérience dans le domaine.

Une règle est généralement composée de deux parties :

- une partie <déclencheur> de la règle, qui représente les conditions de déclenchement de la règle.
- une partie effets de la règle.

Afin de sélectionner une règle particulière, on ne fournit pas un nom propre individuel à cette dernière, mais un ensemble de faits qui peuvent être compatibles avec les conditions de déclenchement de la règle (la partie prémisse). En addition, dans une règle, on ne peut pas désigner une autre règle via un nom individuel.

Exemple de Prémisse et Conclusion :

Si <fait connu>**ALORS** <fait déduit>

a) Cas d'une Prémisse unique

SI <condition> **ALORS** <conclusion>

SI <il pleut> **ALORS** < il porte un imperméable >

Remarque : nous pouvons déduire que la règle est aussi exploitable en sens inverse :

Sachant qu'il a porté un imperméable, nous déduisons qu'il a plu.

b) Prémisses multiples (conjonction de conditions)

SI (condition1) **ET** (condition2) **ALORS** (conclusion)

SI (la pluie continue) **ET** (les égouts sont pleins) **ALORS** (la route est inondée)

Enfin, la principale description concerne l'établissement des liens entre les faits connus, et les faits déduits dans le but de proposer une règle.

Selon la description du domaine. **<**fait connu**> : <**fait déduit**>** , (il pleut : il porte un imperméable).

Exemple

SI < le malade souffre d'une fièvre > **ET** < le malade souffre d'une augmentation de la vitesse de sédimentation dans le sang > **Alors** < il est atteint d'une affection microbienne >.

De là, on va extraire les conditions de déclenchement de la règle :

< Le malade présente de la fièvre >, et < Le malade souffre d'une augmentation de la vitesse de sédimentation dans le sang >.

Alors l'effet de cette règle d'inférence est décrit par le fait : < Il est atteint d'une affection microbienne >.

Le déclanchement de cette règle se traduira alors par la déduction d'un nouveau fait qui est désormais considéré comme établi : < Le malade est atteint d'une affection microbienne >.

3.5. Cycle d'un moteur d'inférence

Pendant son raisonnement, un moteur d'inférence doit enchaîner une séquence de cycles (exécution des règles de production) jusqu'à l'aboutissement du but cherché ou jusqu'il n'y ait plus de règles à appliquer (saturation). Généralement, un cycle d'un moteur d'inférence est basé sur deux phases [29] :

- Une phase d'évaluation (détection des règles applicables).
- Une phase d'exécution (exécution des règles applicables).

Un système expert doit avoir la capacité de faire un choix sur les règles applicables en relation à l'état actuel de la base des faits et les faits à établir.

3.5.1. La phase d'évaluation :

La phase de détection des règles applicables à la phase suivante est considérée comme une partie primordiale du cycle de travail d'un moteur d'inférence, et particulièrement dans son processus d'évaluation. Ce processus d'évaluation peut s'effectuer généralement en trois étapes :

- L'étape sélection ou restriction : cette étape qui est négligeable dans certains cas se focalise sur une première sélection d'un sous-ensemble de règles de production dans la base des règles qui ont une chance d'être pris en considération dans l'étape suivante. Ce choix doit être fait en relation a l'état courant de la base de faits.
- L'étape filtrage : Parmi l'ensemble de règles de production sélectionné dans l'étape précédente, cette étape prend en considération un sous-ensemble, connu comme ensemble de conflit, qui est seulement composé des règles qui peuvent réellement être appliquées. Généralement, ce sous ensemble est composé de chaque règle dont la partie prémisse appartient à la base de faits.
- L'étape de résolution des conflits : à partir des règles sélectionnées dans l'étape de filtrage, le moteur prend la décision concernant la ou les règles qui seront exécutées.

3.5.2. La phase d'exécution

Comme son nom l'indique, l'étape d'exécution se concentre sur l'exécution de la partie action des règles de production sélectionnées lors de la dernière étape de la phase d'évaluation (étape de résolution des conflits). Les nouveaux faits résultants (la partie conclusion de chaque règle exécutée) seront ajoutés à la base des faits.

L'arrêt du cycle du moteur peut survenir dans les deux cas suivants :

- La phase d'évaluation : dans cette phase, le cycle peut s'arrêter dans le cas d'absence de règles applicables dans la situation en cours.
- La phase d'exécution : dans cette phase, le cycle peut s'arrêter dans le cas de l'exécution d'une règle dont la partie « conclusion » implique l'arrêt du cycle.

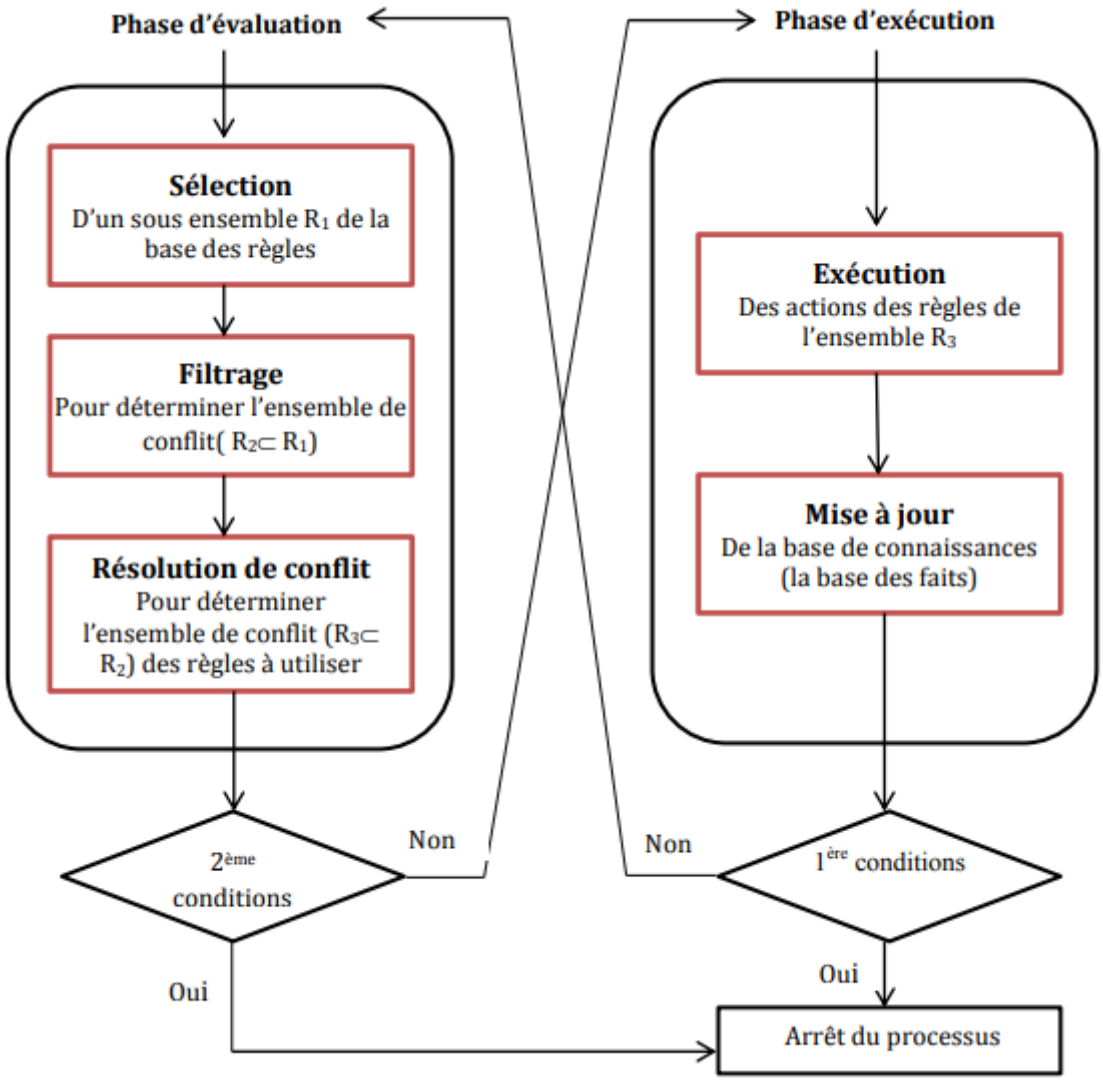


Figure 3.2. Cycle d'un moteur d'inférence [29].

3.6. Les participants au développement d'un système expert

La construction d'un système expert est connue sous le nom d'ingénierie des connaissances et ses praticiens sont appelés ingénieurs des connaissances. L'ingénieur de la connaissance doit s'assurer que l'ordinateur dispose de toutes les connaissances nécessaires pour résoudre un problème.

De plus, il est important d'identifier les différents types de personnes nécessaires pour développer et utiliser un système expert et les compétences nécessaires. En général, l'équipe

de développement de systèmes experts compte cinq membres : l'expert du domaine, l'ingénieur de connaissances, le programmeur, le chef de projet et l'utilisateur final. Le succès du système expert dépend entièrement de la façon dont les membres travaillent ensemble [30].

3.6.1. Expert du domaine

Toute personne peut être considérée comme un expert du domaine si elle possède une connaissance approfondie (des faits et des règles) et une solide expérience pratique dans un domaine particulier. En général, un expert est une personne habile qui peut faire des choses que d'autres ne peuvent pas faire. Prenons plusieurs exemples d'experts humains : un médecin, un grand maître des échecs, un assistant financier, un chef cuisinier, un ingénieur, un architecte, un pharmacien, etc.

3.6.2. Ingénieur de connaissances

C'est la personne qui encode les connaissances de l'expert sous une forme déclarative exploitable par le système expert. Ce sont aussi des personnes capables de concevoir, construire, et tester un système expert. Cette personne est chargée de sélectionner une tâche appropriée pour le système expert. Il ou elle interroge l'expert du domaine pour savoir comment un problème particulier est résolu. Grâce à l'interaction avec l'expert, l'ingénieur de la connaissance établit quelles méthodes de raisonnement l'expert utilise pour traiter les faits et les règles et décide comment les représenter dans le système expert. L'ingénieur de la connaissance choisit alors un logiciel de développement ou un shell de système expert, ou se penche sur des langages de programmation pour encoder la connaissance (et parfois l'encode lui-même). Et enfin, l'ingénieur de la connaissance est chargé de tester, de réviser et d'intégrer le système expert dans le lieu de travail. Ainsi, l'ingénieur de la connaissance est engagé dans le projet depuis la phase de conception initiale jusqu'à la livraison finale du système, et même une fois le projet terminé, il peut également être impliqué dans la maintenance du système.

3.6.3. Le programmeur

C'est la personne responsable de la programmation proprement dite, décrivant les connaissances du domaine en des termes compréhensibles par un ordinateur. Le programmeur doit avoir des compétences en programmation symbolique dans des langages d'IA tels que LISP et Prolog, ainsi qu'une certaine expérience dans l'application de différents types de shells de systèmes experts. De plus, le programmeur doit connaître les langages de programmation

conventionnels comme Python, R, Java, Julia, C, Pascal, FORTRAN e.t.c. Si un shell de système expert est utilisé, l'ingénieur de la connaissance peut facilement encoder les connaissances dans le système expert et ainsi éliminer le besoin du programmeur. Cependant, si un shell ne peut pas être utilisé, un programmeur doit développer les structures de représentation des connaissances et des données (base de connaissances et base de données), la structure de contrôle (moteur d'inférence) et la structure de dialogue (interface utilisateur). Le programmeur peut également être impliqué dans les tests du système expert.

3.6.4. Le chef de projet

C'est le chef de l'équipe de développement du système expert, chargé de maintenir le projet sur la bonne voie. Il ou elle s'assure que tous les objectifs sont atteints dans les temps, interagit avec l'expert, l'ingénieur de connaissances, le programmeur ainsi que l'utilisateur final.

3.6.5. L'utilisateur

Souvent appelé simplement l'utilisateur final, c'est une personne qui consultera le système pour obtenir des conseils qui doivent être fournis par l'expert. C'est une personne qui utilise le système expert lors de son développement. L'utilisateur peut être un chimiste analytique déterminant la structure moléculaire du sol de Mars, un jeune médecin diagnostiquant une maladie infectieuse du sang, un géologue d'exploration essayant de découvrir un nouveau gisement minéral, ou un opérateur de réseau électrique ayant besoin de conseils en cas d'urgence. Chacun de ces utilisateurs de systèmes experts a des besoins différents, auxquels le système doit répondre. L'acceptation finale du système dépendra de la satisfaction de l'utilisateur. L'utilisateur doit non seulement avoir confiance dans les performances du système expert, mais aussi se sentir à l'aise lors de l'utilisation. Par conséquent, la conception de l'interface utilisateur du système expert est également vitale pour la réussite du projet. La contribution de l'utilisateur final peut ici être cruciale.

Il est important de noter que le développement d'un système expert peut commencer lorsque les cinq participants ont rejoint l'équipe. Cependant, de nombreux systèmes experts sont maintenant développés sur des ordinateurs personnels à l'aide de shells de systèmes experts. Cela peut éliminer le besoin du programmeur et pourrait également réduire le rôle de l'ingénieur de la connaissance. Pour les petits systèmes experts, le chef de projet, l'ingénieur de connaissances, le programmeur et même l'expert peuvent être la même personne. Mais tous les ces participants sont nécessaires lorsque de grands systèmes experts sont développés.

3.7. Les algorithmes d'inférences

Les programmes informatiques conventionnels de résolution de problèmes utilisent des algorithmes bien structurés, des structures de données et des stratégies de raisonnement précises pour trouver des solutions. Pour les problèmes difficiles qui concernent les systèmes experts, il peut être plus utile d'employer des heuristiques : stratégies qui mènent souvent à la bonne solution, mais qui échouent aussi parfois. Les systèmes experts conventionnels basés sur des règles utilisent des connaissances expertes humaines pour résoudre des problèmes du monde réel qui nécessiteraient normalement une intelligence humaine. Les connaissances expertes sont souvent représentées sous forme de règles ou de données dans l'ordinateur. Selon l'exigence du problème, ces règles et données peuvent être rappelées pour résoudre des problèmes. Les systèmes experts basés sur des règles ont joué un rôle important dans les systèmes intelligents modernes et leurs applications dans la définition d'objectifs stratégiques, la planification, la conception, l'ordonnancement, la surveillance des pannes, le diagnostic, etc. Grâce aux progrès technologiques réalisés au cours de la dernière décennie, les utilisateurs d'aujourd'hui peuvent choisir parmi des dizaines de progiciels commerciaux dotés d'interfaces utilisateur graphiques conviviales.

Trois types de méthodes d'inférence sont couramment utilisés [31]. Le premier type est le chainage avant qui commence par les faits et avance jusqu'aux conclusions. Tandis que le deuxième type, le chaînage en arrière est le processus qui consiste à commencer par les conclusions et à revenir aux faits à l'appui. Le troisième type est une concaténation des deux premiers types, il est connu sous le nom de chainage mixte.

Exemple :

La différence entre le chaînage avant et le chaînage arrière peut se résumée dans l'exemple suivant :

Supposons qu'on souhaite prendre l'avion de Alger à Pékin, et qu'il n'y a pas de vol direct entre ces deux villes.

- On a la possibilité soit de commencer en cherchant tous les vols qui partent de Alger et noter leur destination. Continuer ce processus jusqu'à ce qu'on tombe sur un vol qui atterrisse à Pékin. C'est le principe du chaînage avant.
- On a une autre possibilité de commencer par Pékin et chercher tous les vols qui y arrivent. On essaye de trouver la ville de départ de tous ces vols. On continue le processus jusqu'à ce qu'on tombe sur une ville dont le point de départ est Alger. C'est le principe du chaînage arrière (on commence du but)

3.7.1. Chainage avant

Les étapes générales de l'algorithme de chainage avant sont les suivants :

Algorithme 1 : Chainage Avant

- DEBUT
  - **TANT QUE** Fait_Objectif n'appartient pas à la base de faits (BF) ET QU'il existe dans la base de règles (BR) une règle applicable **FAIRE**
    - Sélectionner une règle applicable R (étape de résolution de conflits, utilisation d'heuristiques, de métarègles) ;
    - BR = BR - R (désactivation de la règle appliquée R) ;
    - BF = BF ∪ conclusion(R) (la conclusion de R est rajoutée à la base de faits) ;
    - Mise à jour(BR) (mettre à jour la base de règles applicable suite à l'ajout de nouveaux faits) ;
  - **FIN DU TANT QUE**
  - **SI** Fait_Objectif ∈ BF **ALORS**
    - Fait_Objectif est établi;
  - **SINON**
    - Fait_Objectif n'est pas établi;
- **FIN**

Dans l'algorithme du chainage avant (farward chaining algorithm), qui est aussi connu comme le mode guidé par les données, le moteur d'inférence se comporte comme suit :

- Il ne choisit que les règles dont les conditions (prémisses) appartiennent à la base de faits afin de conclure sur le but recherché (fait objectif).
- Par la suite, il applique une de ces règles dans le but d'enrichir la base avec d'autres faits.
- Ce processus est répété jusqu'à ce que le fait objectif soit atteint (appartient à la base de fait), ou plus aucun fait nouveau ne puisse être déduit (saturation).

Le chaînage avant est généralement basé sur un régime irrévocable et monotone. Il est aussi avantagé par le fait d'être simple à implémenter tout en procurant la permission de répondre plus rapidement à tout nouveau fait ajouté. D'une autre part, il est considéré comme sujet au risque de l'explosion combinatoire sachant qu'il est aussi basé sur la sélection de toutes les

règles même celles qui n'ont rien à voir avec la résolution du but. Le chainage avant est appliqué de deux manières différentes :

a. Chainage avant en profondeur

A chaque cycle du moteur d'inférence, seule la première des règles applicable est déclenchée (Il ne s'agit que d'une sélection stratégique parmi d'autres possibles). Les étapes de l'algorithme de chaînage-avant en profondeur sont les suivants :

Algorithme 2 : Chainage Avant en Profondeur

- DEBUT
  - **TANT QUE** Fait_Objectif n'appartient pas à la base de faits (BF) ET QU'il existe dans la base de règles (BR) une règle applicable **FAIRE**
    - Sélectionner la première règle applicable R;
    - BR = BR - R (désactivation de la règle appliquée R) ;
    - BF = BF ∪ conclusion(R) (la conclusion de R est rajoutée à la base de faits) ;
    - Mise à jour(BR) (mettre à jour la base de règles applicable suite à l'ajout de nouveaux faits) ;
  - **FIN DU TANT QUE**
  - **SI** Fait_Objectif ∈ BF **ALORS**
    - Fait_Objectif est établi;
  - **SINON**
    - Fait_Objectif n'est pas établi;
- **FIN**

Exemple :

Soit la BF = {B, C} et soit le fait objectif H (le fait à déduire) en utilisant le chainage avant.

Soit BR composée des règles suivantes :

R1 : B ∧ D ∧ E ⇒ F

R2 : G ∧ D ⇒ A

R3 : C ∧ F ⇒ A

R4 : B ⇒ X

R5 : D ⇒ E

R6 : X ∧ A ⇒ H

R7 : $C \Rightarrow D$

R8 : $X \wedge C \Rightarrow A$

R9 : $X \wedge B \Rightarrow D$

Résolution via l'utilisation du chainage avant en profondeur :

- Application de R4, BF = {B, C, X} ;
- Application de R7, BF = {B, C, X, D} ;
- Application de R5, BF = {B, C, X, D, E} ;
- Application de R1, BF = {B, C, X, D, E, F} ;
- Application de R3, BF = {B, C, X, D, E, F, A} ;
- Application de R6, BF = {B, C, X, D, E, F,A, **H**} # ;

b. Chainage avant en largeur

A chaque cycle du moteur d'inférence, toutes les règles applicables, identifiées par rapport du contenu courant de la base de faits, sont déclenchées. Les étapes de l'algorithme de chaînage-avant en largeur d'abord sont les suivants :

Algorithme 3 : Chainage Avant en Largeur

- DEBUT
  - **TANT QUE** Fait_Objectif n'appartient pas à la base de faits (BF) ET QU'il existe dans la base de règles (BR) une règle applicable **FAIRE**
    - Sélectionner toutes règles applicables $\{R_1, R_2, \ldots R_n\}$;
    - BR = BR - $\{R_1, R_2, \ldots R_n\}$ (désactivation des règles appliquées R) ;
    - BF = BF ∪ conclusion($\{R_1, R_2, \ldots R_n\}$) (les conclusions de $\{R_1, R_2, \ldots R_n\}$ sont rajoutées à la base de faits) ;
    - Mise à jour(BR) (mettre à jour la base de règles applicable suite à l'ajout de nouveaux faits) ;
  - **FIN DU TANT QUE**
  - **SI** Fait_Objectif ∈ BF **ALORS**
    - Fait_Objectif est établi;
  - **SINON**
    - Fait_Objectif n'est pas établi;
- **FIN**

Exemple

Résolution de l'exemple de la section 7.1.1 en utilisant le chainage avant en largeur :

- Application des règles {R4, R7} : BF = {B, C, X, D} ;
- Application des règles {R5, R8} : BF = {B, C, X, D, E, A} ;
- Application des règles {R1, R6} : BF= {B, C, X, D, E, A, F, **H**}# ;

3.7.2. Le chainage arrière

Le principe du chainage arrière est le suivant :

- Le moteur sélectionne les règles qui concluent sur le fait objectif, et s'assurent que ces règles sont déclenchables ou récursivement déclenchables.
- La règle est déclenchable si ses prémisses sont dans la base de faits.
- La règle est récursivement déclenchable si ces prémisses sont des conclusions dans d'autres règles, et que les prémisses de ces dernières appartiennent à la base de faits.
- Si parmi les règles choisies, une règle est considérée comme déclenchable, alors le but est vérifié.
- Si ce n'est pas le cas, alors les prémisses à vérifier doivent être considérées comme les nouveaux buts, appelés sous-buts, et le processus est répété.

Les principales conditions d'arrêt de l'algorithme du chainage arrière se résument comme suit :

- L'ensemble des sous-buts est vide. Ce qui veut dire que tous les sous-buts appartiennent à la base de faits et le problème est résolu.
- Blocage ou échec : dans le cas où un sous-but n'est pas vérifiable avec la règle actuelle et il faut sélectionner une nouvelle règle pour le vérifier, et si cela impossible, alors on est dans une situation d'échec.

Un régime de contrôle par tentatives est nécessaire pour l'algorithme de chainage arrière. Dans d'autres termes, le moteur d'inférence doit faire un retour arrière (backtracking) afin de remettre en cause le choix d'application d'une règle qui débouche sur un blocage et pour essayer une règle non choisie précédemment.

Exemple

On prend l'exemple de la section 7.1.1 afin de vérifier le fait H a travers l'utilisation de l'algorithme du chainage arrière :

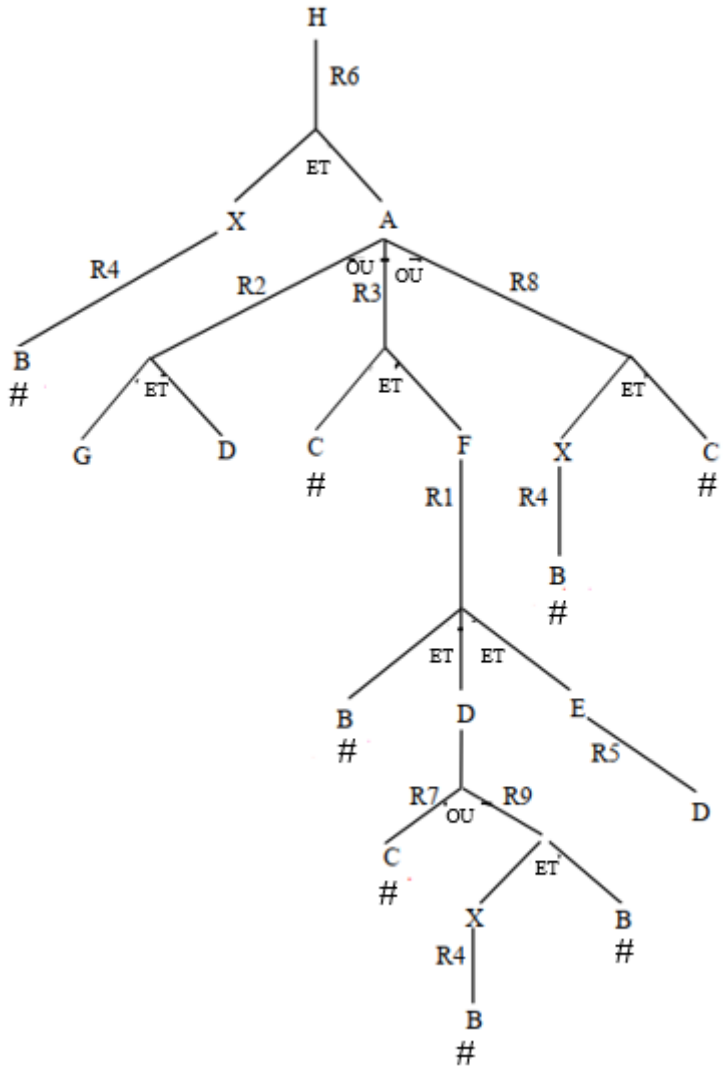


Figure 3.3. Exemple d'application du chainage arrière

3.7.3. Chainage mixte

Il existe une dernière approche de fonctionnement connue comme le chaînage mixte qui est basé sur les 2 chaînages précédents. D'une première vue, il agit comme le chaînage avant avec pour objectif de déduire un fait précis. Cette approche applique un chaînage arrière sur chaque fait déduit dans le but de déterminer les paramètres les plus probables et les plus optimisés. Ce mécanisme ouvre la porte sur de nouvelles combinaisons encore non prévues par les règles d'inférence. Il permet aussi de déterminer les facteurs discriminants lors de du processus de recherche d'une solution.

Exemple

Soient les règles suivantes applicables en chainage arrière :

R1: F ∧ H => K

R2: E ∧ A => K

R3: E ∧ B => H

Soient les règles suivantes applicables en chainage avant :

R4: A ∧ G => B

R5: B ∧ D => H

R6: G ∧ D => E

R7: A ∧ B => D

R8: A ∧ C => G

Soit la base de fait = {A, C}, et le fait objectif K.

Afin de déduire le fait K en utilisant le chainage mixte, nous allons commencer par l'utilisation du chainage avant (en profondeur) comme suit :

- Application de R8, BF = {A, C, G} ;
- Application de R4, BF = {A, C, G, B} ;
- Application de R7, BF = {A, C, G, B, D} ;
- Application de R5, BF = {A, C, G, B, D, H} ;
- Application de R6, BF = {A, C, G, B, D, H, E} ;

Le processus du chainage avant s'arrête car il n'y a plus aucun fait a pouvoir déduire (pas de fait objectif H).

En deuxième étape, nous allons essayer de déduire H en utilisant les règles applicables en chainage arrière. Sachant que la base de fait est BF = {A, C, G, B, D, H, E} :

- On peut voir que le fait K peut être déduit grâce à la règle R2. Les prémisses de R2 sont les faits E, et A, qui appartiennent bien à la base de faits. Donc, on peut dire qu'on a pu déduire le fait K grâce au chainage mixte.

3.8. Les avantages et les inconvénients des systèmes expert

a. Les avantages d'un système expert

Les avantages d'un système expert peuvent être résumés comme suit [30]:

- La permanence : Les systèmes experts n'oublient pas.
- La reproductibilité : Des copies d'un système expert peuvent être faites.
- La puissance : Pour les applications où il y a un labyrinthe de règles exposées, il peut être décrypté par le système expert.

- L'efficacité : les systèmes experts peuvent augmenter le débit et réduire les coûts de personnel.

  -Les systèmes experts sont peu coûteux à exploiter.

  -Les coûts de développement peuvent être amortis sur de nombreuses années.

  - Les systèmes experts peuvent éliminer les coûts de routine et réduire les coûts de maintenance majeurs.
- La cohérence : Avec les systèmes experts, les événements similaires sont traités de la même manière. Les systèmes experts feront des recommandations comparables pour des situations « similaires », et ne seront pas affectés par des effets récents ou primaires.
- La documentation : Les systèmes experts fournissent une documentation permanente concernant le processus de décision.
- La complétude : Un système expert peut passer en revue toutes les transactions ou possibilités.
- L'opportunité : la fraude et/ou les erreurs peuvent être évitées. Les informations sont disponibles plus tôt pour la prise de décision et l'action. Le système expert fonctionne 24 heures sur 24, toute l'année.
- La portée : Le système expert peut englober l'expertise cumulée de plusieurs experts humains.
- Le succès commercial : les propriétaires réduisent les risques inhérents à la conduite de leur entreprise grâce à :

  - Cohérence dans la prise de décision.

  - Savoir-faire acquis.
- Les effets positifs :

  - Gains de productivité et économies de coûts.

  - Nouvel outil critique pour les managers et une réponse proactive à l'attrition de l'expertise.

  - Les décisions et les solutions sont plus cohérentes et moins sujettes à des préjugés ou sensibilité à l'environnement.

  - Emploi : basculer vers un travail plus satisfaisant.

b. Les inconvénients d'un système expert

- Le bon sens : En plus de nombreuses connaissances techniques, les experts humains ont du bon sens. On ne sait pas encore comment donner du bon sens aux systèmes experts.
- La créativité : les experts humains peuvent réagir de manière créative dans des situations inhabituelles, contrairement aux systèmes experts.
- L'apprentissage : les experts humains s'adaptent automatiquement aux changements de l'environnement. Les systèmes experts doivent explicitement se mettre à jour. Le raisonnement par cas et les réseaux de neurones sont des méthodes qui peuvent intégrer l'apprentissage d'un système expert.
- L'expérience sensorielle : Les experts humains disposent d'un large éventail d'expériences sensorielles. Les systèmes experts dépendent actuellement de l'entrée symbolique.
- La dégradation : les systèmes experts ne sont pas efficaces dans la phase de reconnaissance quand aucune réponse n'existe, ou quand le problème est en dehors de leur domaine d'expertise.

3.9. Exercices

Exercice 1 :

Soient les règles d'inférences suivantes :

R1 : s'il fait beau aujourd'hui alors il fera beau demain

R2 : s'il pleut aujourd'hui alors il y a des nuages

R3 : si le vent est d'ouest alors les hirondelles volent bas

R4 : s'il y a des nuages alors le temps est à la pluie

R5 : si les araignées tissent des toiles alors les chiens pressentent la pluie

R6 : si les hirondelles volent bas alors les loups pressentent la pluie

R7 : si (les loups ou les chiens pressentent la pluie) et s'il y a des nuages alors il pleuvra demain

R8 : s'il pleut en Bretagne et le vent est d'ouest alors le temps est à la pluie

R9 : si le temps est à la pluie alors il pleut aujourd'hui.

R10: si les loups pressentent la pluie alors il fait beau aujourd'hui

1- Modélisez les règles suivantes en logique des prédicats.
2- Pouvons-nous savoir s'il pleuvra demain à partir de la BF {le vent est d'ouest, il pleut en Bretagne} en utilisant le chainage avant en profondeur ?

3- Quels sont les faits déductibles en utilisant le chainage avant en largeur à partir de la BF {Il pleut aujourd'hui, les loups pressentent la pluie, le vent est d'ouest} ?

Exercice 2:

Soient les règles d'inférences suivantes :

R1: SI Responsabilité ET Langue-facile ET Néerlandais-parlé ALORS Dynamique

R2 : SI Langue-facile ET Anglais-parlé ALORS Adaptabilité

R3: SI Slave ET Dynamique ALORS Adaptabilité

R4: SI Responsabilité ALORS Leadership

R5: SI Langue-facile ALORS Néerlandais-parlé

R6: SI Adaptabilité ET Leadership ALORS Accepté

R7: SI Slave ALORS Langue-facile

R8: SI Leadership ET Slave ALORS Adaptabilité

BF initiale : Slave, Responsabilité.

Que peut-on en déduire en utilisant le chainage avant en profondeur et en largeur ?

Exercice 03

On veut concevoir un système expert pour diagnostic médical basé sur les principes suivants : un patient est atteint de rougeole s'il est fiévreux et a des points rouges. Un patient à froid si son nez coule. Un patient qui a froid est un patient contagieux. Si le patient est fiévreux et atteint d'un torticolis alors il a une méningite. Si le patient a une rougeole ou une méningite alors il est aussi contagieux. Le patient doit être isolé s'il est contagieux et dangereux. Le patient est dangereux dans le cas où il a une méningite. En cas de froid, de rougeole, et de contagion, alors le patient est aussi dangereux.

1- Concevez une base de règles d'ordre 1 à partir de ce texte.
2- Quels sont les faits qu'on peut déduire à partir de la base de faits suivante en utilisant le chainage avant en profondeur : marie a le nez qui coule, des points rouges, et une fièvre. John a un torticolis, et une fièvre.
3- Peut-on déduire que marie doit être isolée à partir de la base de faits suivante en utilisant le chainage avant en largeur : elle a une fièvre, des points rouges, et un torticolis.
4- Peut-on déduire que John doit être isolé s'il a une fièvre en chainage arrière (construisez l'arbre avec tous les cas possibles).

3.10. Correction des exercices

Solution exercice 1

1.Modélisation avec la logique des prédicats :

R1: FBA() → FBD()

R2: PA() → N()

R3: VO() → HVB()

R4: N() → TP()

R5: ATT() → CPP()

R6: HVB() → LPP()

R7: LPP() ^ N() → PD()

R8: CPP() ^ N() → PD()

R9: PB() ^ VO() → TP()

R10: TP() → PA()

R11: LPP() → BA()

2.BF = {VO, PB} but: PD ? en utilisant le chainage avant en profondeur :

R3 : BF = {VO, PB, HVB}

R6 : BF = {VO, PB, HVB, LPP}

R9 : BF = {VO, PB, HVB, LPP, TP}

R10 : BF = {VO, PB, HVB, LPP, TP, PA}

R2 : BF = {VO, PB, HVB, LPP, TP, PA, N}

R7 : BF = {VO, PB, HVB, LPP, TP, PA, N, **PD**} #

Donc oui, on peut déduire le fait qu'il pleuvra demain à partir des faits le vent et d'ouest et il pleut en Bretagne.

3.BF = {PA, LPP, VO}, les faits déductibles en utilisant le chainage avant en largeur :

R2, R3, R11 : BF = {PA, LPP, VO, N, HVB, BA}

R4, R6, R7 : BF = {PA, LPP, VO, N, HVB, BA, TP, PD} #

L'algorithme s'arrête car il n'y a plus de règles applicables qui peuvent amener à la déduction de nouveaux faits.

Solution exercice 2

1.Chainage avant en profondeur : BF : {S, R}

R4 : BF : {S, R, L}

R7 : BF : {S, R, L, LF}

R5 : BF : {S, R, L, LF, NP}

R1 : BF : {S, R, L, LF, NP, D}

R3 : BF : {S, R, L, LF, NP, D, AD}

R6 : BF : {S, R, L, LF, NP, D, AD, AC}#

Les faits qu'on peut déduire en utilisant le chainage avant en profondeur sont : Leadership, langue-Facile, Néerlandais-Parlé, Dynamique, Adaptabilité, Accepté.

2.Chainage avant en largeur : BF : {S, R}

R4, R7 :BF : {S, R, L, LF}

R5, R8 : BF : {S, R, L, LF, NP, AD}

R1, R6 : BF : {S, R, L, LF, NP, AD, D, AC}#

Les faits qu'on peut déduire en utilisant le chainage avant en largeur sont : Leadership, langue-Facile, Néerlandais-Parlé, Dynamique, Adaptabilité, Accepté.

Solution exercice 03

1.Conception de la base de règles :

R1 : $\forall x\ Fi(x) \wedge PR(x) \rightarrow R(x)$

R2 : $\forall x\ NC(x) \rightarrow Fr(x)$

R3 : $\forall x\ Fr(x) \rightarrow Co(x)$

R4 : $\forall x\ Fi(x) \wedge To(x) \rightarrow Me(x)$

R5 : $\forall x\ R(x) \rightarrow Co(x)$

R6 : $\forall x\ Me(x) \rightarrow Co(x)$

R7 : $\forall x\ Co(x) \wedge Da(x) \rightarrow Is(x)$

R8 : $\forall x\ Me(x) \rightarrow Da(x)$

R9 : $\forall x\ Fr(x) \wedge R(x) \wedge Co(x) \rightarrow Da(x)$

2.BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John)}, Chainage avant en profondeur :

R1 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie)},

R2 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie)},

R3 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie)},

R4 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie), Me(John)},

R6 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie), Me(John), Co(john)},

R8 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie), Me(John), Co(john), Da(john)},

R9 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie), Me(John), Co(john), Da(john), Da(marie)},

R7 : BF = {NC(marie), PR(marie), Fi(Marie), To(john), Fi(John), R(marie), Fr(marie), Co(marie), Me(John), Co(john), Da(john), Da(marie), Is(john), Is(marie)}, #

3.BF = {Fi(marie), PR(marie), To(marie)}, But = Is(marie), chainage avant en largeur :

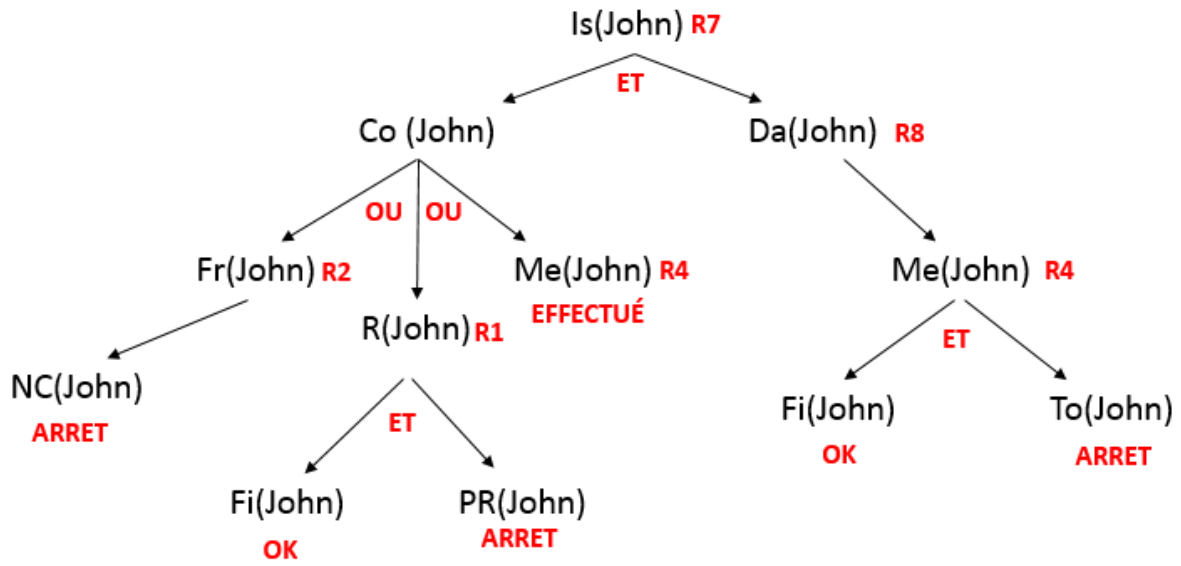
R1, R4 : BF = {Fi(marie), PR(marie), To(marie), R(marie), Me(marie)},

R5, R6, R8 : BF = {Fi(marie), PR(marie), To(marie), R(marie), Me(marie), Co(marie), Da(marie)},

R7 : BF = {Fi(marie), PR(marie), To(marie), R(marie), Me(marie), Co(marie), Da(marie), **Is(marie)**}#

Donc, oui on peut déduire que marie doit être isolée.

4.BF = {Fi(John)}, But = Is(John), Chainage arrière:



Donc, on ne peut pas déduire que John doit être isolé en sachant seulement qu'il est fiévreux en utilisant le chainage arrière.

Chapitre 4 : Systèmes experts et application

Objectifs du chapitre

Les objectifs du chapitre 4 peuvent être résumés comme suit :

- *Lever l'ambiguïté sur le développement du système expert au fil du temps.*
- *Connaitre les différentes techniques de programmation des systèmes experts.*
- *Maitriser le langage de programmation PROLOG ainsi que sa philosophie.*
- *Découvrir les différents domaines d'application des systèmes experts.*

4.1. Historique des systèmes experts

Les systèmes experts (SE) sont des systèmes qui émanent du nouveau domaine de l'informatique connu sous le nom d'intelligence artificielle (IA). L'IA est la branche de l'informatique concernée par le développement d'ordinateurs qui se comportent comme des humains. Précisément, les systèmes experts occupent une place centrale dans l'aspect des sciences cognitives de l'intelligence artificielle comme le montre la figure ci-dessous.

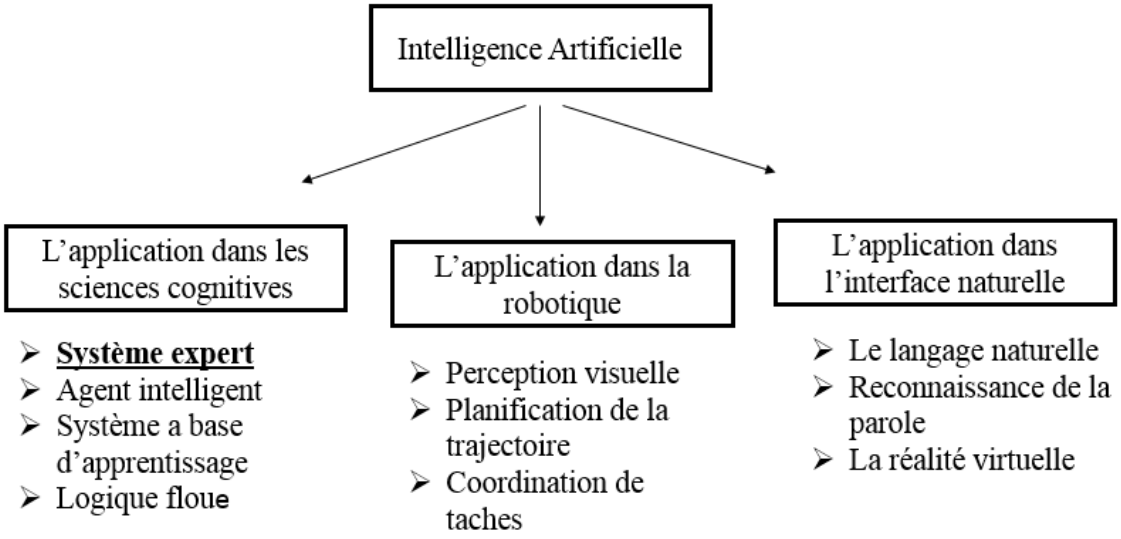


Figure 4.1. Application de l'intelligence Artificielle

Au milieu des années 1960, Edward A. Feigenbaum était l'un des chercheurs en intelligence artificielle qui a décidé qu'il était opportun de savoir ce qu'un programme informatique peut savoir et que la meilleure façon de le savoir serait d'essayer de construire un spécialiste de l'artificiel. Dans le cadre de la recherche d'un domaine d'expertise approprié, Feigenbaum a collaboré avec Joshua Lederberg, le biochimiste lauréat du prix Nobel, qui a alors suggéré que

les chimistes organiques avaient cruellement besoin d'aide pour déterminer la structure moléculaire des composés chimiques.

En 1965, Lederberg et Feigenbaum avec Bruce Buchanan, conformément aux exigences de la National Aeronautics and Space Administration, à l'Université de Stanford, ont commencé à travailler sur DENDRAL [13], le premier système expert. Ce projet a démarré parce que les systèmes informatiques conventionnels n'avaient pas réussi à fournir aux chimistes organiques un outil de prévision de la structure moléculaire. Les chimistes humains savent que la structure possible de tout composé chimique dépend d'un certain nombre de règles de base sur la façon dont différents atomes peuvent se lier les uns aux autres. Ils connaissent également beaucoup de faits sur les différents atomes des composés connus. Lorsqu'ils fabriquent ou découvrent un composé jusque-là inconnu, ils peuvent recueillir des preuves sur le composé en analysant la substance avec un spectroscope de masse, qui fournit de nombreuses données, mais aucun indice sur ce que tout cela signifie.

Le système DENDRAL peut générer automatiquement des structures moléculaires capables d'interpréter les données spectrales. Le programme est un premier programme réussi qui utilise la connaissance du problème lui-même plutôt que la technologie de recherche complexe. Le DENDRAL guide les chercheurs en IA et en systèmes experts pour réaliser que le comportement intelligent repose non seulement sur les méthodes d'interférence mais aussi sur les connaissances utilisées dans leur interférence. Les chercheurs commencent à construire le programme qui utilise les règles du code pour représenter les connaissances afin de résoudre le problème d'entrée.

L'ingénierie des connaissances peut être définie comme la science d'observer des experts humains, de construire des modèles de leur expertise et d'affiner le modèle jusqu'à ce que les experts humains conviennent que cela fonctionne. L'une des premières retombées de Dendral était Meta-Dendral, un système expert pour les personnes dont l'expertise réside dans la construction de systèmes experts. En séparant le moteur d'inférence de l'ensemble des connaissances factuelles, Buchanan a pu produire un outil pour les constructeurs de systèmes experts.

Depuis lors, le Massachusetts Institute of Technology (MIT) a également développé le système MACSYMA [32]. Le MACYSMA était un assistant de mathématicien, qui utilise l'heuristique pour transformer des expressions algébriques. Après une expansion continue, il peut résoudre plus de 600 types de problèmes mathématiques, y compris le calcul, l'opération matricielle, la résolution d'équations, etc. Le développement réussi de ces systèmes rend le système expert largement concerné par le milieu universitaire et l'ingénierie. De nombreux

chercheurs en train de développer des systèmes experts ont réalisé que la représentation des connaissances, l'utilisation des connaissances et l'acquisition des connaissances sont trois problèmes fondamentaux des systèmes d'intelligence artificielle.

Au milieu des années 1970, MYCIN [14] a été développé par Edward H. Shortliffe, médecin et informaticien à la Stanford Medical School. Les problèmes associés au diagnostic d'une certaine classe d'infections cérébrales étaient un domaine approprié pour la recherche sur les systèmes experts et un domaine de besoin humain particulièrement pressant, car les premières vingt-quatre à quarante-huit heures sont essentielles au succès du traitement de ces maladies. Avec toutes ses promesses et toutes ses implications éthiques effrayantes, la médecine apparaît comme l'un des domaines d'application les plus actifs de l'ingénierie commerciale des connaissances.

Le moteur d'inférence de MYCIN, connu sous le nom d'E-MYCIN [40], a été utilisé par des chercheurs du Stanford and Pacific Medical Center pour produire Puff, un système expert qui aide à diagnostiquer certains troubles pulmonaires. Un système encore plus récent, Caduceus, dispose désormais d'une base de connaissances plus large que celle de n'importe quel médecin. Caduceus contient des données brutes comprenant environ 80% de la littérature médicale mondiale.

Prospector [33], développé par SRI International, examine les données géologiques plutôt que les molécules ou les symptômes. Récemment, ce programme a prédit avec précision l'emplacement d'un gisement de molybdène qui pourrait valoir des dizaines de millions de dollars.

À la fin des années 1980, le système expert basé Framework a commencé à entrer dans la vision des gens. En raison de sa plus grande capacité à représenter des informations d'objet descriptives et comportementales, un système expert basé sur un Framework pourrait gérer des problèmes plus complexes que le système expert basé sur des règles. Dans le même temps, l'étude du système expert a rencontré des difficultés, exposant les défauts des systèmes d'intelligence artificielle, tels que les domaines d'application étroits, les difficultés d'acquisition de connaissances, le mécanisme de raisonnement, etc. Les chercheurs doivent se débarrasser du dilemme en explorant le point de vue de base et l'utilisation de nouvelles techniques et théories.

Les tournants qui ont joué un rôle important dans l'évolution des systèmes experts peuvent se résumés dans le tableau suivant :

Tableau 4.1. Développement des systèmes experts

Année	Evènement
1943	Introduction des règles de post-production : modèle et McCulloch et Pitts Neuron [7].
1954	Introduction de l'Algorithme de Markov pour le contrôle de l'exécution des règles [17].
1956	La Conférence de Dartmouth.
1957	Introduction Perceptron de Rosenblatt [33], ainsi que le GPS (General Problem Solver) de Newell, Shaw, et Simon [34].
1958	Introduction du langage de programmation LISP par McCarthy [35].
1962	Introduction du principe de Rosenblatt dans la neurodynamique [33].
1965	Introduction de la Logique floue par Zadeh,[36] et du premier système expert DENDRAL [13], par Feigenbaum et Buchanan.
1968	Introduction des Réseaux sémantiques et mémoire associative par Quillian [37].
1969	Introduction de MACSYMA [32] : système expert en mathématiques par Martin et Moses.
1970	Introduction de PROLOG par Colmerauer et Roussel [38].
1973	Introduction de MYCIN [14] (médecine) par Shortliffe et al, suivi de GUIDON [39] par Clancey, TEIRESIAS par Davis, EMYCIN [40] par Van Melle, Shortliffe, et Buchanan.
1975	Introduction des Frames de Minsky [41].
1978	Introduction de Meta-Dendral [42] (méta-règles et induction) par Buchanan.
1979	Introduction de l'Algorithme Rete [43] pour une correspondance efficace des modèles. Cette année a aussi vécu le début de la commercialisation de l'IA.
1983	Introduction d'un système expert conne sous le nom de KEE par Intellicorp [44].
1985	Introduction de CLIPS, un système expert par la NASA [45].

Le système expert fait référence a un système intelligent qui a la capacité d'émuler les compétences cognitives d'experts humains pour guider les utilisateurs dans des processus décisionnels complexes.

Dans les années 1970, les progrès de l'informatique ont fait repenser que pour que l'ordinateur résout un problème intellectuel il fallait en connaître la solution. En d'autres termes, il faut avoir le savoir-faire dans un domaine spécifique car un expert dans ce domaine résoudrait le problème. Les fonctionnalités de ces types de systèmes sont basées sur la connaissance de sa

tâche et des règles ou procédures logiques obtenues à partir de l'expérience d'un spécialiste dans le domaine.

Ce système utilise une base de connaissances, qui est soigneusement formulée sur la base d'un jugement d'expert, de l'intuition et de l'expérience. Environ deux douzaines de sociétés vendent actuellement des systèmes experts et des services. Teknowledge, fondée par Feigenbaum et ses associés en 1981, a été la première. IntelliGenetics est peut-être la plus exotique, spécialisée dans les systèmes experts pour l'industrie du génie génétique. Les start-ups dans ce domaine tendent vers des noms de science-fiction - Machine Intelligence Corporation, Computer Thought Corp., Symbolics, etc. Schlumberger.

4.2. Techniques de programmation des systèmes experts

La plupart des systèmes experts sont développés via des outils logiciels spécialisés appelés shells. Les shells fournissent un Framework pour produire un système expert. Ainsi, la base de connaissances et les règles sont simplement ajoutées à ce Framework. Ces shells sont équipés d'un mécanisme d'inférence (chaînage arrière, chaînage avant ou les deux) et nécessitent la saisie de connaissances selon un format spécifié. Ils sont généralement dotés d'un certain nombre d'autres fonctionnalités, telles que des outils d'écriture d'hypertexte, de construction d'interfaces utilisateur conviviales, pour manipuler des listes, des chaînes et des objets, et pour s'interfacer avec des programmes et des bases de données externes. Ces shells sont considérés comme des langages, bien qu'ils aient certainement un champ d'application plus restreint que la plupart des langages de programmation [30].

Choisir le bon outil pour construire le système expert est une étape importante. Parfois, plus d'un outil sera utilisé au cours d'un projet. Généralement, un outil est utilisé pour le prototypage, mais un outil différent est choisi pour le développement et la livraison à grande échelle.

La programmation et les outils du système expert simplifient le travail de construction d'un système expert. Ils vont du langage de programmation de très haut niveau aux installations de support de bas niveau. Ceux-ci sont divisés en quatre grandes catégories :

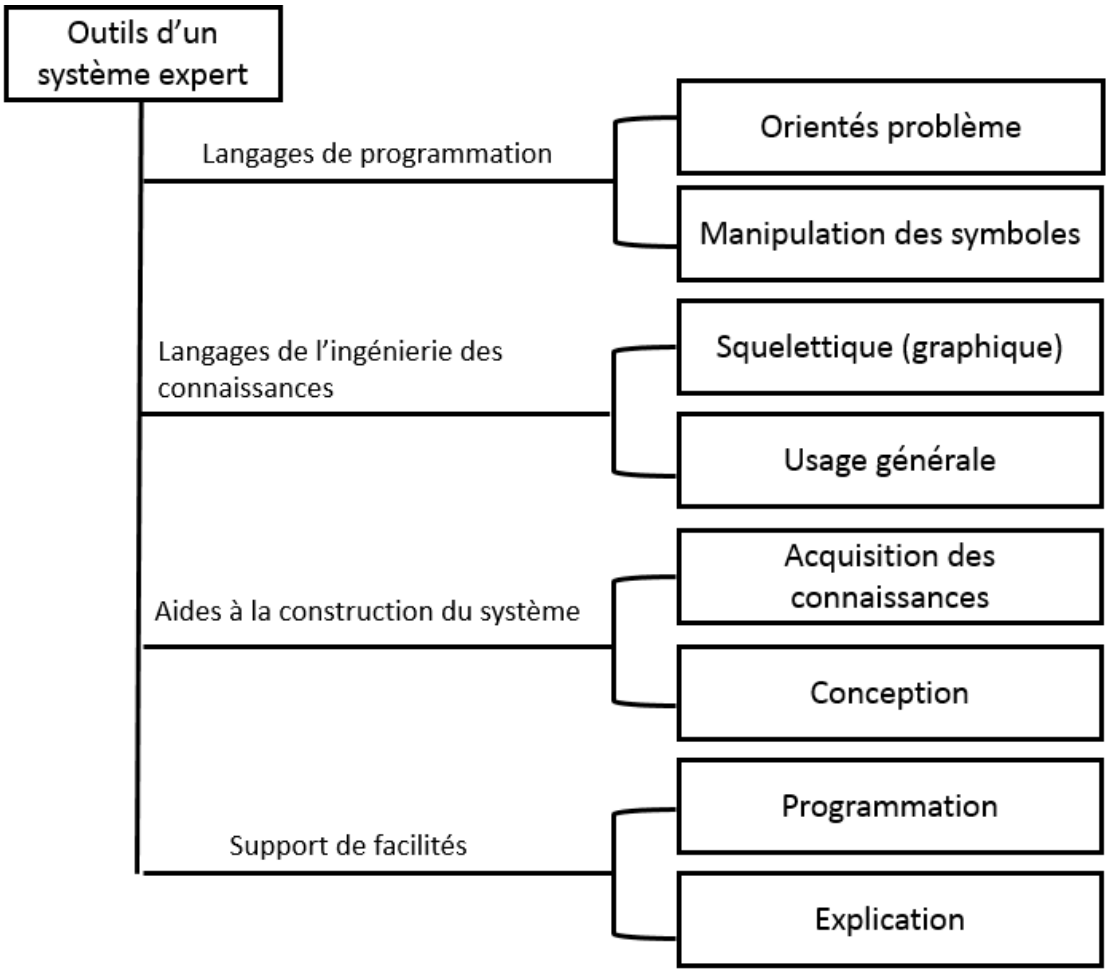


Figure 4.2. Technique de programmation d'un système expert

4.2.1. **Langages de programmation d'un système expert**

Les systèmes experts sont généralement écrits dans des langages tels que LISP [35] et PROLOG [38] ou même CLIPS [45] (système de production intégré en langage C développé au milieu des années 1980). L'utilisation de ces langages dans le développement d'un système expert simplifie le processus de codage. Les principaux avantages de ces langages par rapport aux langages de programmation conventionnels sont la simplicité d'ajout, d'élimination ou de substitution de nouvelles règles et les capacités de gestion de la mémoire. Les caractéristiques souhaitables de ces langages sont [30] :

Représentation des connaissances : CLIPS fournit un outil cohérent pour gérer une grande variété de connaissances avec la prise en charge de trois paradigmes de programmation différents.

Orienté objet et procédural : les capacités de programmation procédurale fournies par CLIPS sont similaires aux capacités trouvées dans les langages conventionnels tels que

Python, R, C, Java, Ada et LISP. Portabilité : CLIPS est écrit en C pour la portabilité et la vitesse et a été installé sur de nombreux systèmes d'exploitation différents sans changement de code. Les systèmes d'exploitation sur lesquels CLIPS a été testé incluent Windows XP, MacOS X et Unix. CLIPS peut être porté sur n'importe quel système doté d'un compilateur C ou C++ conforme à l'ANSI. CLIPS est livré avec tout le code source qui peut être modifié ou adapté pour répondre aux besoins spécifiques d'un utilisateur.

Intégration/Extensibilité : CLIPS peut être intégré dans du code procédural, appelé sous-programme, et intégré à des langages tels que C, Java, FORTRAN et ADA. CLIPS peut être facilement étendu par un utilisateur grâce à l'utilisation de plusieurs protocoles bien définis.

Développement interactif : la version standard de CLIPS fournit un environnement de développement interactif orienté texte, comprenant des aides au débogage, une aide en ligne et un éditeur intégré. Des interfaces offrant des fonctionnalités telles que des menus déroulants, des éditeurs intégrés et plusieurs fenêtres ont été développées pour les environnements MacOS, et Windows XP.

Vérification/Validation : CLIPS inclut un certain nombre de fonctionnalités pour prendre en charge la vérification et la validation des systèmes experts, y compris la prise en charge de la conception modulaire et du partitionnement d'une base de connaissances, la vérification des contraintes statiques et dynamiques des valeurs d'emplacement et des arguments de fonction, et l'analyse sémantique des modèles de règles pour déterminer si des incohérences peuvent empêcher le déclenchement d'une règle ou générer une erreur.

Entièrement documenté : CLIPS est livré avec une documentation complète comprenant un manuel de référence et un guide de l'utilisateur.

Faible coût : CLIPS est maintenu en tant que logiciel du domaine public.

4.3. PROLOG

4.3.1. Définition de Prolog

Prolog (**pro**grammation **log**ique) [38] est l'un des langages de programmation les plus utilisés dans la recherche en intelligence artificielle. Contrairement aux langages impératifs tels que C ou Java (ce dernier étant également orienté objet), il s'agit d'un langage de programmation déclaratif. Cela signifie que, lors de la mise en œuvre de la solution à un problème, au lieu de

spécifier comment atteindre un certain objectif dans une certaine situation, nous spécifions la situation (règles et faits) et l'objectif (requête) et laissons l'interpréteur Prolog dériver la solution pour nous. Prolog est très utile dans certains domaines problématiques, tels que l'intelligence artificielle, le traitement du langage naturel, les bases de données, ou les systèmes experts. D'un autre côté, il est assez inutile dans d'autres, comme les graphiques ou les algorithmes numériques.

4.3.2. Symbolisation en prolog

Prolog peut être fondamentalement considéré comme un démonstrateur de théorèmes basé sur le principe de résolution de Robinson [29]. Prolog est basé sur la logique du premier ordre (logique des prédicats). Les symboles logiques sont représentés en prolog comme suit :

Tableau 4.2. Symbolisation en Prolog

Symbole	Logique des prédicats	Prolog
Et	$\wedge$	,
Ou	$\vee$	;
Implication	$\rightarrow$	:-
Négation	$\neg$	not

Par exemple, si on veut représenter l'énoncé « Tous les animaux sont mortels » :

- En logique des prédicats : $(\forall x)$ (Animal(x) $\rightarrow$ Mortel(x)).
- En Prolog : mortel(X) :- animal(X).

En addition, pour représenter l'énoncé « Marie est une femme » :

- En logique des prédicats : femme (Marie).
- En Prolog : femme (marie).

4.3.3. Les composants d'un programme prolog :

Un programme prolog est composé de clauses. Ces dernières sont de trois types : faits, règles ainsi que questions. Le programmeur est donc dans l'obligation de faire la description du problème à résoudre à travers l'utilisation d'objets, et en spécifiant les relations entre eux.

- **Les faits :**

Les faits représentent les données élémentaires qui sont considérés comme vraies (les hypothèses de travail). On peut aussi les voir comme des affirmations qui fournissent la description des relations entre les objets ou des propriétés entre les objets. À partir des ces

faits, le Prolog pourra rechercher des preuves en guise de réponse aux requêtes des utilisateurs. De façon générale, toutes les déclarations de faits sont placées au début du programme même si ce n'est pas une obligation. Par exemple, soit l'ensemble des faits suivants :

étudiant (amine, 2000, mathématiques, 2)
masculin (amine)
mère (salima, amine) % Salima est la mère de Amine.
Salima est la mère d'Amine signifie que la relation mère lie les deux objets Salima et Amine.
On peut généraliser la clause *mère (salima, amine) par mère (X, Y) où X et Y* sont des objets.
La forme générale d'un fait est comme suit :

Prédicat (arguments1, argument2, ...)

- **Les règles :**

Un programme Prolog est composé presque toujours des règles, cependant leur présence n'est pas obligatoire. Les règles sont des relations permettant d'établir de nouveaux faits par déduction à partir des hypothèses (Si on a démontré M1 et M1→ M2, alors on a démontré M2).
La forme générale d'une règle est la suivante :

$$N :- M_1, M_2,\ldots, M_n$$

Cette règle se lit : si M_1, M_2,…, M_n alors N, sachant que M_1, M_2,…, Mn représentent des prédicats. N (partie gauche de la règle) reflète la tête de la règle ou la conclusion. Elle doit être obligatoirement présente. M_1, M_2,…, Mn (partie droite de la règle) représente le corps de la règle. En d'autres termes, les contraintes liées par l'opérateur de conjonction ou la partie prémisse. Le corps de la règle est composé de 0/1 ou plusieurs termes. Comme dans la logique traditionnelle, les prémisses de la règle doivent être vérifiées pour que la tête de la règle soit vraie.
Si $[(M_1 \wedge M_2 \wedge \ldots \wedge M_n)$ alors N] est équivalente à $[(M_1 \wedge M_2 \wedge \ldots \wedge M_n) \rightarrow N]$, ce qui est équivalent à : $(\neg M_1 \vee \neg M_2 \vee \ldots \vee \neg M_n \vee N)$, ce qui reflète une clause d'Horn en logique.
Par conclusion, un programme prolog est composé d'une suite de clause d'Horn :

$\neg M_1 \vee \neg M_2 \vee \ldots \vee \neg M_n \vee N$

$\neg L1 \vee \neg L2 \vee \ldots \vee \neg Ln \vee Q$

$\neg T1 \vee \neg T2 \vee \ldots \vee \neg Tn \vee K$

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

grand-père(X,Y):-père(X,Z), père(Z,Y).

grand-père(X,Y):-père(X,Z), mère(Z,Y).

Ou: Grand-père(X,Y):-père(X,Z),(père(Z,Y); mère(Z,Y)).

- ❖ **Les buts**
- ✓ **Requêtes fermées**

L'objectif du codage des informations à travers l'utilisation des faits et règles dans un programme Prolog est de pouvoir interroger la base de connaissance, aussi bien par des requêtes fermées (dont la réponse est oui/non), que par des requêtes ouvertes à une ou plusieurs inconnues (détecter les objets pour lesquels une affirmation est vraie). Une requête fermée peut être comme suit :

?- enseignant(mohammed, hamid), patron(mohammed, lila).

Ici, on veut savoir si mohammed est l'enseignant de hamid et de lila.

- ✓ **Requêtes ouvertes**

L'usage des variables en Prolog va nous permettre de transformer des requêtes fermées en requête ouvertes, c'est à dire en requêtes qui peuvent avoir plusieurs réponses possibles. Par exemple : ici, on veut savoir qui est l'enseignant de hamid :

?- enseignant(X, hamid).

X= mohammed

Dans la requête ci-dessous, en veut savoir qui enseigne qui :

?- enseignant(X, Y).

X = mohammed, Y = hamid ;

X = mohammed, Y = lila

Afin de répondre à ces requêtes, l'interpréteur PROLOG est dans l'obligation de remplacer la variable X par une des constantes apparaissant précédemment dans le programme sachant que le fait résultant doit exister dans la base de faits. Les variables peuvent aussi être utilisées pour décrire des faits.

4.3.4. Les types de Prolog

En prolog, Un terme fait référence à un objet, qui peut être :

- Une constante (connue aussi comme terme atomique) : Avec la syntaxe d'Edimbourg un atome peut être écrit comme suit :
  - ✓ Un identificateur qui commence par une lettre minuscule : mohammed, amine.
  - ✓ Un nombre.
  - ✓ Une chaîne de caractères entre apostrophes : 'Mohammed', 'Amine'.
  - ✓ Une liste : ['Mohammed', amine, 5].
- Une variable qui peut être définie comme une chaine alphanumérique qui débute par une majuscule ou par _ : X, Mohammed, _amine.
- Une fonction doit être représentée par une chaîne alphanumérique débutant par une minuscule, suivie d'une liste de termes entre parenthèses et sépares par une virgule : homme (amine), père (hamid, amine).

4.3.5. La substitution et l'unification

Le fait de répondre à une question contenant des variables a pour principe de déterminer les valeurs des variables pour que la question soit déductible des faits et règles du programme. Afin de démontrer une question, un des principes les plus importantes qu'on utilise en programmation logique est la substitution.

Précisément, soient X_1, X_2,….X_n, des variables différentes. Et soient m_1, m_2…..., m_n des qui n'est pas obligatoire qu'ils soient différents. Une substitution Λ est l'association qui fait correspondre X_1 à m_1, X_2 à m_2,……..X_n à m_n :

$$\Lambda = \{X_1 \leftarrow m_1, X_2 \leftarrow m_2, \ldots\ldots..X_n \leftarrow m_n\}$$

Unifier de termes t_1 et t_2 à pour principe de trouver la substitution la plus générale permettant de faire de t1 et t2 un seul et même terme. Pour pouvoir unifier deux termes :

- Il faut qu'ils aient le même nom.
- Il faut qu'ils aient le même nombre de paramètre.

En Prolog, l'unification est symbolisée par l'opérateur « = ».

Exemple

Soit le programme prolog suivant :

?- g (a, Y) = g (K, T), h (b, K) = h (Y, L)

Y = b

K = a

T = b

L = a

yes

Afin de résoudre ces deux unifications de manière simultanée, Prolog se ramène à quatre unifications :

a ↔ K

Y ↔ T

B ↔ Y

K ↔ L

Donc, la solution la plus générale d'après prolog est : $\begin{cases} a = K = L \\ b = Y = T \end{cases}$

4.3.6. Arbre SLD (Selective Linear Definite)

Dans le but de répondre à une question, Prolog réalise des essais répétitifs en respectant l'ordre de déclaration des prédicats. Ces essais peuvent être reflétés par un arbre de recherche qu'on appelle aussi arbre de résolution. Les nœuds de cet arbre forment ce qu'on appelle les points de choix. Ces points de choix peuvent être reflétés comme suit [29]:

- La racine de l'arbre qui représente but à chercher.
- Nœuds : points de choix. Dans le cas ou on a plusieurs manières d'effectuer des unifications, ça veut dire qu'on a un point de choix qui peut être :
  - ✓ Des règles différentes qui définissent le même prédicat.
  - ✓ Des prédicats disjoints dans le corps de la règle.
  - ✓ Une variété de possibilités d'instanciation d'une variable.
- Passage d'un nœud vers son successeur en réalisant une unification. Les nœuds sont démontrés de gauche à droite, en respectant l'ordre de déclaration des règles.
- Au final, il y a des nœuds qui retournent succès et d'autres qui retournent échec. Les nœuds de succès n'ont pas comme contenu des prémisses a vérifiées, tout a été démontré et les valeurs des variables de l'objectif sont trouvés à travers la remonté vers la racine de l'arbre. Dans l'autre cas (nœuds d'échec), aucune règle ne permet de vérifier la première formule du nœud.

Exemple : soit le programme suivant :

grand-mère(X,Y):-mère(X,Z), mère(Z,Y), femme(X)

grand-mère(X,Y):-mère(X,Z), père(Z,Y), femme(X)

femme (amina)

mère (amina, sara)

mère (amina, mohammed)

mère (sara, karim)

père (mohammed, ammar)

Soit le but : ?- grand-mère (X, Y).

L'arbre de résolution est le suivant :

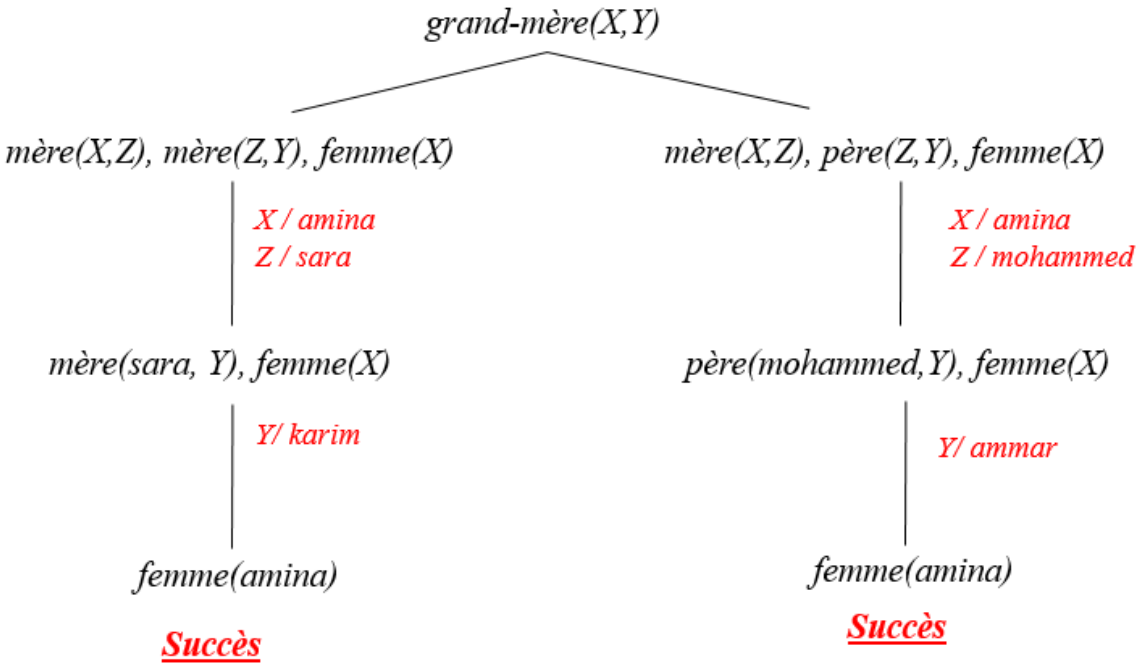


Figure 4.3. Exemple (1) d'un arbre SLD

D'après l'arbre, on peut conclure :

grand-mère (amina, karim)

grand-mère (amina, ammar)

Prolog est implémenté sur la base d'une procédure de recherche en profondeur d'abord. En effet, Prolog fait des tentatives d'aller le plus profond possible dans un chemin avant de tester un autre chemin.

Dans l'arbre SLD, les nœuds de succès reflètent des solutions, d'autres solutions peuvent aussi être cherchées. Dans le cas où un nœud d'échec est atteint, Prolog effectue une remonté dans l'arbre de résolution jusqu' à un point de choix qui possède des branches non explorées. Dans le cas où 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 connue comme le Backtracking.

Exemple : soit le programme Prolog suivant :

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).

La question est : ?-planete (X)

Solution

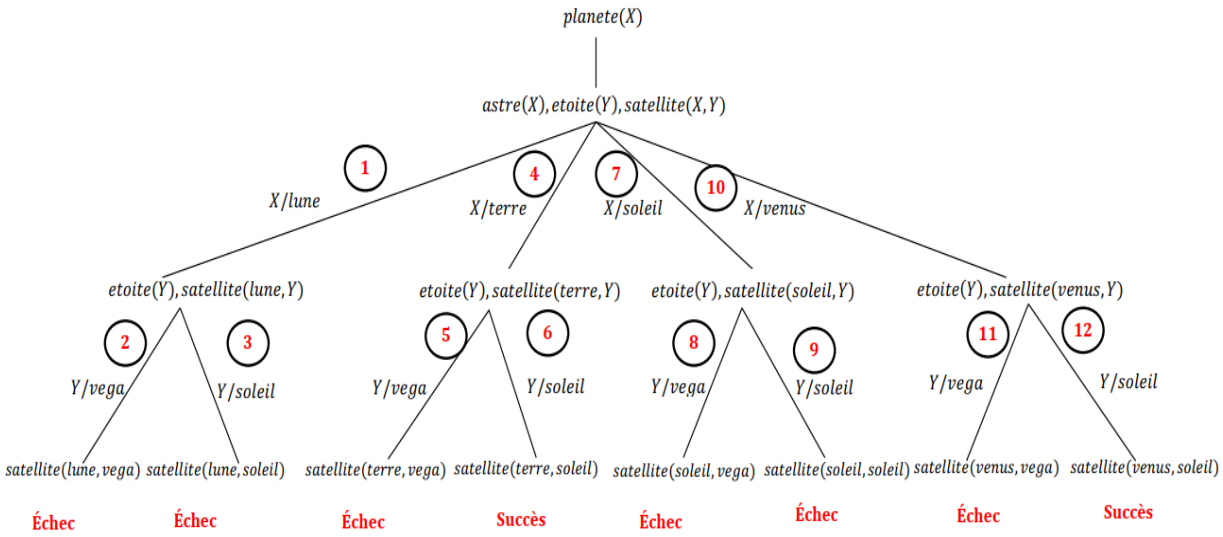


Figure 4.4. Exemple (2) d'un arbre SLD

D'après l'arbre, on peut conclure :

- planete (terre),
- planete (venus).

4.3.7. Les phases d'exécution d'un programme prolog

- Dans le but de vouloir répondre à une question (recherche à remplacer les variables d'un but par des valeurs), Prolog va essayer de l'unifier, soit avec un fait, ou avec la tête d'une règle.
- Dans le cas où l'unification avec la tête de la règle est réussie, Prolog commence le processus de substitution des variables de la queue (corps de la règle) par les valeurs correspondantes dans la tête. Cette simplification est connue comme : la résolution SLD (Sélection Linéaire Définie, en anglais : Selective Linear Definite).

- Prolog continue a exécuter le processus d'unification jusqu'à ce que la requête initiale soit vérifiée.
- Dans le cas où l'unification échoue, Prolog répond par faux (false).
- Dans le cas où l'unification réussit, Prolog répond soit par vrai (true), ou en donnant les valeurs des variables.
- Au final, Prolog construit une liste de couples valeur-variable.

4.4. Application des systèmes expert

Les systèmes experts sont maintenant utilisés à des fins commerciales et de recherche dans un certain nombre de domaines :

• KAS (Knowledge Acquisition System) et Tirésias aident les ingénieurs de la connaissance à construire des systèmes experts.

• L'ONCOCIN [31] aide les médecins à gérer des schémas thérapeutiques complexes pour le traitement des patients atteints de cancer.

• Molgen [30] aide les biologistes moléculaires à planifier des expériences sur l'ADN.

• Guidon est un expert en éducation qui enseigne aux élèves en corrigeant les réponses aux questions techniques.

• Genesis aide les scientifiques à planifier des expériences de clonage.

• Le TATR [30] est utilisé par l'Armée de l'Air pour planifier des attaques contre les bases aériennes ennemies.

Plus précisément, les systèmes experts sont souvent appliqués dans les domaines suivants :

4.4.1. Ingénierie et fabrication

L'impact de l'application des technologies informatiques de la branche de l'ingénierie telle que la branche mécanique, électrique, automobile, locomotive, chimique et autres pourrait être un sujet difficile à discuter. Les produits des industries d'ingénierie actuelles que nous pouvons trouver partout rendent notre vie beaucoup plus facile et confortable, pour l'instant un véhicule qui raccourcit notre période de déplacement, la télécommande nous permet de faire fonctionner la machine sur de longues distances, etc. Derrière ces produits d'ingénierie processus, certains processus, peut-être inefficaces pour être exécutés et manipulés de manière conventionnelle [46].

Par conséquent, un système expert impliqué dans le processus d'ingénierie et de fabrication peut fournir la meilleure assistance aux travailleurs opérationnels lors de l'exécution et de la gestion des tâches critiques et importantes. De plus, une analyse précise des résultats de

chaque pièce ou processus d'ingénierie pouvant être obtenue à partir d'un système expert qui peut éviter toute perte pour l'entreprise.

4.4.2. Le domaine médical

Les technologies dans le traitement médical sont importantes aussi bien lors du prétraitement de la consultation médicale, du diagnostic de la maladie que du traitement proprement dit par un spécialiste ou un médecin. Peu de médecins ont suffisamment d'expertise et d'expérience pour consulter les patients sur le diagnostic et le traitement de certaines maladies à haut risque, en particulier dans la plupart des pays en développement comme les pays d'Asie du Sud, d'Afrique et du tiers monde.

En outre, les patients doivent attendre plus longtemps avant que les maladies ne soient diagnostiquées par un spécialiste et lorsque le traitement est indiqué, il peut déjà tarder et le patient pourrait en souffrir toute la vie ou, dans certains cas, la mortalité des patients ne peut être évitée. Les technologies informatiques telles que le système expert peuvent résoudre les problèmes ci-dessus dans le but de diagnostiquer plus tôt les maladies pour les patients, d'identifier les symptômes de la maladie et de fournir une réponse précise immédiate pour sauver la vie du patient. De plus, les pourcentages de chance de guérir cette maladie deviennent positifs, ce qui signifie que la pratique actuelle pour le patient de consulter un spécialiste avant que le traitement ne devienne plus efficace.

4.4.3. L'agriculture

Dans le domaine de l'agriculture, des systèmes experts ont été appliqués à des problèmes tels que la gestion des cultures, la lutte contre les insectes et les considérations de productivité pour la production d'une culture donnée. Les agriculteurs et les agents du Département des services de recherche agricole doivent prendre des décisions concernant la production efficace et rentable de diverses cultures. L'expertise pour prendre ces décisions existe, mais le problème majeur est de mettre ce talent à la disposition du grand nombre d'agriculteurs.

Le système expert PLANT introduit par Boulanger en 1983 [46], prédit les dommages au maïs causés par l'invasion des vers-gris noirs. Le système obtient d'abord des informations sur la situation actuelle du champ, y compris des informations telles que la concentration de mauvaises herbes, l'état du sol et la variété de maïs cultivée. Ces informations, associées à des programmes de simulation de vers noirs, sont utilisées pour prédire le niveau attendu de dommages causés par ce ravageur.

COMAX introduit par Baker et Lemmon en 1985 [46], intègre les connaissances d'un modèle de production de coton pour fournir des conseils sur la culture et la gestion de cette culture. Le système expert utilise le modèle du coton pour prédire la croissance et le rendement des cultures en réponse aux conditions météorologiques, aux variables du sol et aux dommages causés par les ravageurs. COMAX [46] est capable de fournir des recommandations pour les décisions de gestion au quotidien afin de maximiser les rendements du coton.

CROPPRO introduit par Durkin et Godine en 1989, a été développé pour aider les agriculteurs dans quatre domaines majeurs de la production agricole tels que : les problèmes de gestion des cultures, la lutte antiparasitaire, les considérations financières et le tutorat sur divers sujets liés aux cultures. Le système est structuré pour être applicable à la plupart des cultures en s'attaquant aux problèmes communs. L'hypertexte et les graphiques interactifs sont largement utilisés, ce qui améliore l'interface et l'efficacité du système. Le champignon Shiitake a été choisi comme cas test pour ce système expert.

4.4.4. La chimie

La majorité des systèmes experts développés dans le domaine de la chimie ont été appliqués dans un environnement de laboratoire. L'avantage majeur de ces systèmes est l'assistance qu'ils apportent au technicien de laboratoire tout au long d'une expérience donnée. Ils peuvent aider à la planification et au suivi de l'expérience, ainsi qu'à l'interprétation des données d'essai.

L'un des premiers systèmes experts a été développé par l'Université de Stanford à la demande de la NASA. La NASA prévoyait d'envoyer un vaisseau spatial sans pilote sur Mars et souhaitait développer un programme informatique qui mettrait en œuvre l'analyse chimique du sol martien. La NASA fournirait des informations sur le sol sous forme de spectrogrammes de masse ; et, à partir de ces informations, le programme devrait déterminer sa structure moléculaire. Pour développer ce programme, l'équipe de Stanford a dû encoder dans le programme l'expertise d'un chimiste ayant des compétences spécialisées dans ce domaine. Le programme résultant de cet effort est devenu connu sous le nom de DENDRAL introduit par Buchanan et Feigenbaum en 1978. DENDRAL est capable de déduire la structure moléculaire d'un composé inconnu à partir des données du spectrogramme de masse.

SPEX intoduit par Iwasaki en 1982 [46], aide les scientifiques à planifier des expériences de laboratoire dans le domaine de la biologie moléculaire. Le scientifique définit et décrit les différents objets à utiliser dans une expérience, tels que l'environnement physique et la

structure de l'expérience. Le système aide à élaborer un plan pour atteindre l'objectif de l'expérience.

GA1 introduit par Stefik en 1978 [46], détermine les structures d'ADN possibles à partir des données de segmentation des enzymes de restriction. Le système utilise un modèle d'analyse de digestion enzymatique des structures d'ADN, associé à la connaissance des erreurs possibles dans les environnements de test de laboratoire, pour formuler son résultat.

4.4.5. L'informatique

Les applications de systèmes experts dans le domaine de l'informatique se sont focalisées sur la conception ou le diagnostic de divers systèmes informatiques. La configuration de systèmes informatiques qui répondent aux spécifications définies par le client, ou le diagnostic des défauts dans un système donné, peuvent être des tâches difficiles et chronophages.

XCON introduit par McDermott en 1980 [46] a été développé pour configurer les systèmes informatiques VAX pour Digital Equipment Corp.(DEC). Chacun des clients de DEC a besoin de systèmes informatiques avec certaines caractéristiques et fonctionnalités, et a certaines contraintes liées aux ressources et à l'espace disponible. Pour tenir compte de ces diverses caractéristiques et pour rendre le problème de recommandation de configuration plus efficace, DEC a tenté de développer un programme informatique conventionnel pour aider à la tâche de configuration. Incapable de trouver une solution satisfaisante, XCON (appelé à l'origine Rl) a donc été développé et permet actuellement à DEC d'économiser 20 millions de dollars par an.

DART introduit par Bennett et Hollander 1981 [46] diagnostique les défauts du matériel des systèmes informatiques. Le système intègre la connaissance de la structure et du comportement attendu d'un système afin de trouver des défauts de conception dans les nouveaux systèmes informatiques.

YES introduit par Griesmer en 1984 [46], aide les opérateurs informatiques à contrôler le système d'exploitation MVS utilisé dans les grands ordinateurs IBM centraux. Ce système expert surveille le système d'exploitation, interprète les messages MVS et fait des recommandations à l'opérateur de la console pour le contrôle du système. Les principales tâches de YES sont les suivantes : maintenir un espace de file d'attente adéquat, gérer les communications réseau, planifier les travaux par lots, répondre aux erreurs système et répondre aux erreurs matérielles du système.

4.4.6. La géologie :

L'utilisation dominante des systèmes experts dans le domaine de la géologie a été appliquée au problème de l'exploration. Les systèmes experts peuvent aider un géophysicien dans l'interprétation des données, et d'agir à leur place dans les situations où il n'y en a pas.

PROSPECTOR [33] était un système expert développé au Stanford Research Institute pour aider les géologues dans l'exploration des gisements de minerai par Duda en 1977. PROSPECTOR utilise des connaissances basées sur cinq modèles différents qui décrivent divers gisements minéraux. Le système obtient initialement des informations qui caractérisent un gisement d'intérêt particulier, y compris des informations sur l'environnement géologique, les contrôles structurels et les types de minéraux et de roches présents ou suspectés sur le site. Ces informations sont ensuite comparées aux modèles et le système note les similitudes, les différences et les informations manquantes. Ensuite, il évalue la présence potentielle d'un gisement minéral donné. En 1980, PROSPECTOR a été testé sur le terrain sur un site près de Mount Tolman, WA. Le test a abouti à la découverte d'un gisement de molybdène de 100 millions de dollars. Le succès spectaculaire de PROSPECTOR a inspiré un certain nombre d'autres systèmes experts commerciaux qui s'appuient sur certaines des fonctionnalités utilisées dans PROSPECTOR.

DIPMETER introduit par Davis en 1981 détermine la structure géologique du sous-sol d'un site donné en interprétant les diagraphies du pendomètre. Le système utilise des connaissances sur les données de pendage et la géologie de base pour découvrir des caractéristiques dans les données qui aident à l'identification des structures géologiques. Cette capacité est particulièrement importante dans l'exploration pétrolière ou minérale.

La société française d'exploration pétrolière, Elf Aquitaine, entretient un certain nombre de puits de pétrole et, comme de nombreuses compagnies pétrolières, rencontre souvent des problèmes de blocage des trépans. Il n'est pas rare que les dépenses liées au forage dépassent 100 000 $ par jour et que les arrêts causés par des problèmes de forage pendant plusieurs semaines, jusqu'à ce qu'un expert puisse être amené sur le site.

En raison de la rareté des experts dans ce domaine, Elf Aquitaine a passé un contrat avec la société californienne Teknowledge pour développer un système expert qui se substituerait à l'expert humain. Ce système s'appelait justement le DRILLING ADVISOR [46]. Le système utilise des informations sur les formations géologiques du site, les conditions du problème

actuel et des informations historiques sur d'autres problèmes rencontrés dans le passé. Le système expert effectue ensuite un diagnostic du problème, produit une recommandation pour corriger le problème et fournit en outre des conseils pour modifier les pratiques actuelles afin d'éviter le problème à l'avenir.

4.5. Exercices

Exercice 1:

Soit la base de faits suivante :

homme(albert).	femme(simone).	pere(louis,benoit).
homme(jean).	femme(marie).	mere(germaine,jean).
homme(paul).	femme(sophie).	mere(christiane,simone).
homme(bertrand).	pere(albert,jean).	mere(christiane,paul).
homme(louis).	pere(jean,paul).	mere(simone,benoit).
homme(benoit).	pere(paul,bertrand).	mere(marie,bertrand).
femme(germaine).	pere(paul,sophie).	mere(marie,sophie).
femme(christiane).	pere(jean,simone).	

Question

Traduire les questions suivantes en Prolog et vérifier les réponses :

- Est-ce que Paul est un homme ?
- Est-ce que Benoit est une femme ?
- Qui est une femme ?
- Qui est un homme ?
- Est-ce que Marie est la mère de Sophie ? de Benoit ?
- Qui est la mère de Jean ?
- Quels sont les enfants de Paul ?
- Quels sont les hommes qui sont pères ?

1.3.Définition de prédicats

Définir les prédicats suivants :

• parent(X,Y) : X est un parent de Y, père ou mère ;

• fils(X,Y) : X est le fils de Y ;

• fille(X,Y) : X est la fille de Y ;

• grand_pere(X,Y) : X est le grand-père de Y ;

• grand_mere(X,Y) : X est la grand-mère de Y ;

• frere(X,Y) : X est le frère de Y ;

• soeur(X,Y) : X est la soeur de Y.

2. Traduction d'énoncés

Traduire en Prolog l'énoncé suivant :

- Marie aime les orange.
- Pierre est un voleur.
- Pierre aime tous ceux qui aiment le vin.
- Si quelqu'un est un voleur et aime quelque chose alors il le vole.
- Qui vole quoi?

Exercice 2:

1. Ecrire un prédicat prolog qui est vrai si x est un élément de la liste L.

2. premier(E,L) est vrai si E est le premier élément de L.

3. Ecrire un prédicat Prolog der qui trouve le dernier élément d'une liste L.

4. Ecrire un prédicat Prolog avDer qui trouve le l'avant-dernier élément d'une liste L.

Définir le prédicat longueur(L,N), qui étant donnée la liste L calcule sa longueur N.

5. Définir le prédicat pair(L) qui est vrai si L a un nombre pair d'éléments.

4.6. Correction des exercices

Correction exercice 1

1.

?- homme(paul). Yes ?- femme(benoit). No ?- femme(X). X = germaine ; X = christiane ; X = simone ; X = marie ; X = sophie ; No ?- homme(X).	X = jean ; X = paul ; X = bertrand ; X = louis ; X = benoit ; No ?- mere(marie,sophie). Yes ?- mere(marie,benoit). No ?- mere(X,jean). X = germaine ;	?- pere(paul,X). X = bertrand ; X = sophie ; No ?- homme(X),pere(X,_). X = albert ; X = jean ; X = jean ; X = paul ; X = paul ; X = louis ; No

X = albert ;	No	

2.

X est un parent de Y, pere ou mere :

```
parent(X,Y) :- mere(X,Y).
parent(X,Y) :- pere(X,Y).
```

X est le fils de Y :

```
fils(X,Y) :- mere(Y,X),homme(X).
fils(X,Y) :- pere(Y,X),homme(X).
```

X est la fille de Y :

```
fille(X,Y) :- mere(Y,X),femme(X).
fille(X,Y) :- pere(Y,X),femme(X).
```

X est le grand-père de Y :

```
grand-pere(X,Y) :- parent(P,Y), pere(X,P).
```

X est la grand-mere de Y :

```
grand-mere(X,Y) :- parent(P,Y), mere(X,P).
```

X est le frère de Y :

```
frere(X,Y) :- homme(X), pere(P,X), pere(P,Y), mere(M,X), mere(M,Y), X \== Y.
```

X est la soeur de Y :

```
soeur(X,Y) :- femme(X), pere(P,X), pere(P,Y), mere(M,X), mere(M,Y), X \== Y.
```

3.

Traduction d'enonces :

```
aime(marie,orange).
aime(pierre,X) :- aime(X,orange).
vole(X,Y) :- voleur(X), aime(X,Y).
voleur(pierre).
/* vole(X,Y) donne X=pierre et Y=marie */
```

Correction exercice 2

1. element(E,L) est vrai si E appartient a L (non deterministe , si E et L instancie et si L contient plusieurs fois E) :

```
element(E,[E|_]).
element(E,[_|T]) :- element(E,T).
```

2. premier(E,L) est vrai si E est le premier element de L :

```
premier(X,[X|_]).
```

3. dernier(E,L) est vrai si E est le dernier element de L :

```
dernier(X,[X]).
dernier(X,[_|L]) :- dernier(X,L).
```

4. avdernier(X,L) :

```
avdernier(X,[X,_]).
avdernier(X,[_,Y|Ys]) :- avdernier(X,[Y|Ys]).
```

5. pair(L) est vrai si L a un nombre pairs d'elements :

```
pair([]).
pair([_,_|S]) :- pair(S).
```

Chapitre 5 : Méthode de construction d'un Système Expert

Objectifs du chapitre

Les objectifs du chapitre 5 peuvent être résumés comme suit :

- *Découvrir les différentes phases du cycle de vie d'un système expert ainsi que les différentes relations entre elles.*
- *Avoir des idées spécifiques entre les différents types des systèmes experts.*
- *Lever l'ambigüité sur le principe de l'acquisition des connaissances dans les systèmes experts.*
- *Assimiler les différentes méthodes servant à construire les systèmes experts.*

5.1. Cycle de vie d'un système expert

Le cycle de vie d'un système expert [30] peut se résumer dans les phases suivantes :

5.1.1. La phase d'identification

La première étape de la phase d'identification c'est l'identification du problème, qui est similaire à la phase de définition du problème dans le cycle de vie de développement de systèmes traditionnels. L'objectif est d'identifier, de caractériser, et de définir les problèmes que le système devra résoudre, puis de diviser le problème en sous-tâches appropriées.

Une fois le problème défini, les ressources nécessaires pour acquérir des connaissances, mettre en œuvre le système, et tester le système sont identifiées. Les ressources typiques comprennent les connaissances, le temps, les installations informatiques et l'argent. Sachant que les systèmes experts coûtent cher et que leur création prend un temps considérable, une étude de faisabilité est souvent effectuée avant que les travaux n'aillent au-delà de ce point.

En plus d'identifier les ressources, les analystes et/ou les concepteurs du système expert identifient également les buts et les objectifs du système. Il est utile d'identifier et de documenter explicitement les objectifs, car certaines approches de conception, telles que la recherche heuristique, la recherche étendue, la recherche approfondie et le raisonnement, sont axées sur les objectifs.

5.1.2. La phase de conceptualisation

La tâche centrale de la phase de conceptualisation est de schématiser les concepts et relations clés du système afin de définir une base conceptuelle pour un système prototype. Les principaux objectifs comprennent la séparation du moteur d'inférence du domaine du problème, la factorisation (analyse) du problème en méta-problèmes, l'identification des concepts clés du système et relations, et tester ces concepts et relations en les mettant au défi (avec des exemples spécifiques d'activités de résolution de problèmes) pour s'assurer qu'ils couvrent tous les cas généraux. De nombreux outils et techniques sont utilisés dans cette phase.

5.1.3. La phase de formalisation

La phase de formalisation consiste à cartographier les concepts clés, les sous-problèmes et les caractéristiques du flux d'informations isolés lors de la conceptualisation dans des représentations plus formelles basées sur divers outils d'ingénierie des connaissances et de résolution de problèmes et des cadres de représentation des connaissances. Les principaux objectifs sont d'identifier l'espace des solutions (un domaine avec une collection de toutes les solutions possibles), l'espace des hypothèses (l'espace des solutions hypothétiques), le modèle sous-jacent et les caractéristiques des données. Pour définir la structure de l'espace des hypothèses, les analystes ou concepteurs de systèmes doivent formaliser les concepts (connaissances sous une forme abstraite pouvant servir à guider un processus de recherche ou de raisonnement) et déterminer comment ils s'articulent pour former une hypothèse.

Les concepts fournissent des indices sur la nature de l'espace, par exemple s'il est fini, si une hiérarchie doit être considérée, si certains niveaux d'abstraction peuvent être appliqués et si une classe spécifique du concept doit être générée. Des techniques de recherche telles que la recherche aveugle, la recherche heuristique et l'abstraction de l'espace des solutions sont souvent utilisées. Les techniques de raisonnement telles que la construction d'hypothèses, la construction de justifications et la technique des contraintes et des objectifs aident à identifier le modèle sous-jacent du processus utilisé pour générer des solutions dans le domaine.

5.1.4. La phase de conception du système

Au cours de la phase de conception du système (parfois appelée phase de conception logique), l'analyste et/ou le concepteur précisent comment le système répondra aux exigences identifiées au cours des trois phases précédentes. Généralement, les rapports et autres sorties que les systèmes doivent produire sont définis en premier. Cette phase est similaire à l'étape de conception dans le cycle de vie de développement de systèmes traditionnels. Notant

cependant que les schémas de représentation utilisés pour décrire les connaissances diffèrent des méthodologies traditionnelles.

Grâce aux connaissances acquises et à l'outil sélectionné, on peut maintenant commencer la conception du système expert. Tout d'abord, on doit créer un plan, un organigramme hiérarchique, une matrice, un tableau de décision ou un autre format qui nous aidera à organiser et à comprendre les connaissances. À l'aide de ces points, on converti les connaissances en règles IF-THEN. Il est préférable de suivre les procédures spécifiques recommandées par l'outil logiciel que nous utilisons. Une fois la conception de base terminée, on peut commencer à utiliser l'outil pour créer un prototype d'un segment du système. Traduire une partie des connaissances en règles et tester le segment nouvellement créé. Tester le concept avant d'aller de l'avant avec l'ensemble du programme.

5.1.5. La phase de développement du système

Un prototype de système expert est créé au cours de la phase de développement du système (ou de conception physique). Cette étape est similaire à l'étape de développement dans le cycle de vie de développement de système traditionnel. Une fois que nous nous sommes assuré que le système fonctionnera de manière satisfaisante, on peut maintenant commencer à étendre le prototype au système final. La meilleure façon d'y parvenir est d'étendre le prototype un segment à la fois.

5.1.6. La phase de test et d'évaluation

Au cours de cette phase, le système prototype est évalué. Cette phase est parallèle à l'étape de test dans le cycle de vie de développement de système traditionnel. Cependant, en plus des outils et des techniques de test, les systèmes experts utilisent une technique de test dynamique pour vérifier le processus de raisonnement et/ou d'inférence.

Une fois le système expert développé, on doit passer du temps à le tester et à le déboguer. Aucun système expert ne sera parfait du premier coup, et une quantité considérable de travail sera nécessaire pour le valider. Les commentaires des utilisateurs vous indiqueront où apporter les dernières modifications, corrections et ajouts pour obtenir les performances souhaitées.

5.1.7. La phase de révision du prototype

Un système expert évolue au fil du temps, nécessitant une révision presque constante, un trait que les systèmes experts partagent avec la plupart des prototypes. Sur la base des résultats de la phase de test/évaluation, les concepts et les relations sont affinés, l'espace de solution, le modèle, les caractéristiques des données sont reformalisés et le système est reconçu.

5.1.8. La phase de maintenance du système

Une partie importante du développement d'un système expert est la maintenance continue, la mise à jour du système avec de nouvelles connaissances, la suppression des connaissances qui ne sont plus applicables et le réglage fin du système pour le maintenir entièrement à jour et applicable au problème.

Les relations ainsi que les principes des différentes phases sont résumées dans le diagramme suivant :

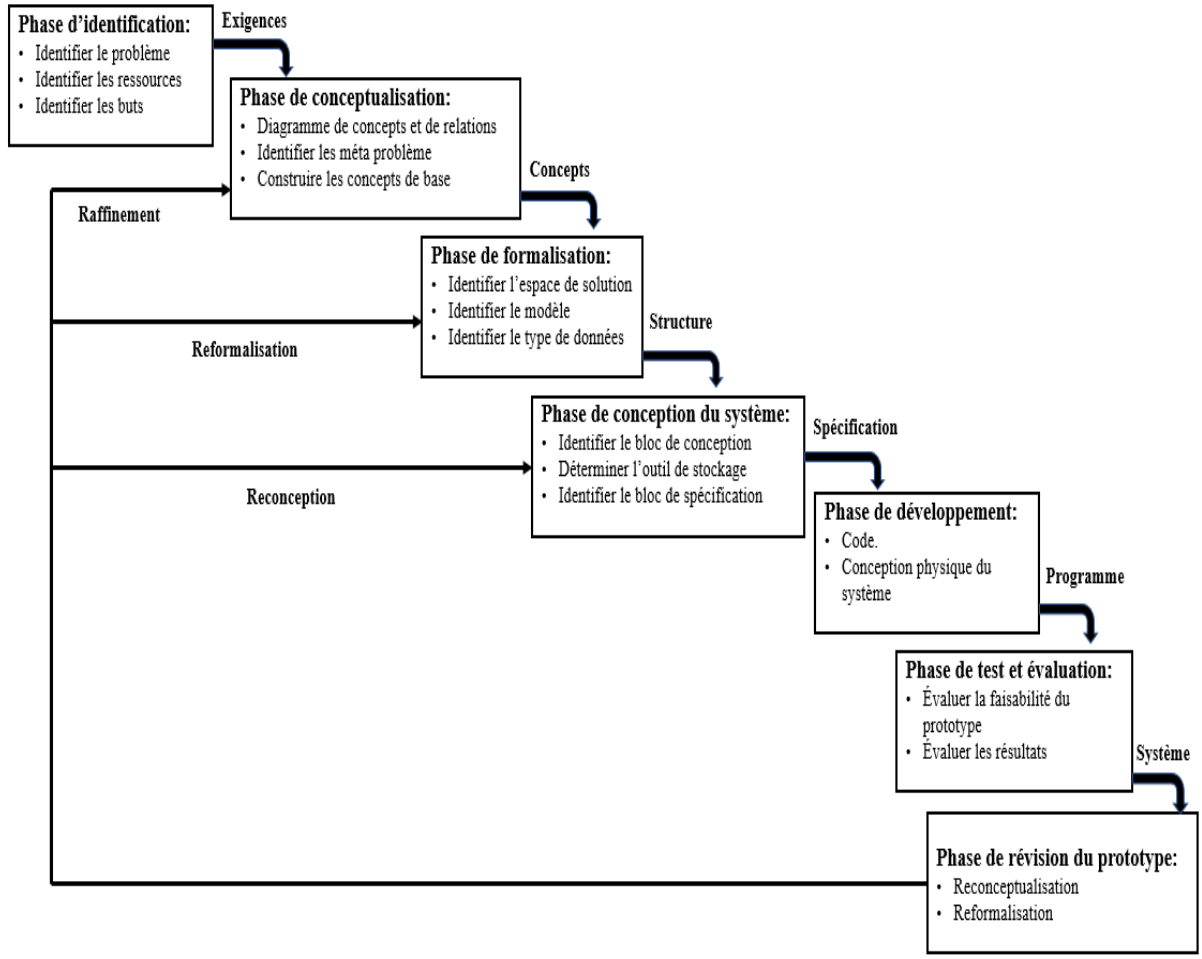


Figure 5.1. Cycle de vie d'un système expert

5.2. Les différents types des systèmes experts

Après avoir expliqué les systèmes experts basés sur des règles dans le chapitre 3 de ce support, nous allons aller plus loin pour expliquer les systèmes experts flous et basés sur des neurones. Aussi, dans cette section, nous allons introduire les systèmes experts basés sur les frames.

5.2.1. Les systèmes experts basés sur la logique floue

En 1965, le mathématicien américain Zadeh [36] a proposé pour la première fois le concept d'ensemble flou, marquant la naissance des mathématiques floues. Le flou fait référence à la nature indiscriminée des choses ou attributs objectifs, et il existe une série d'états de transition entre eux, sans ligne de démarcation évidente. La théorie floue permet aux gens d'utiliser des outils mathématiques pour traiter le phénomène non précis dans le monde réel.

Dans le système expert basé sur la logique floue, la logique floue est la base du raisonnement du système expert. Ce type de méthode de raisonnement utilise la règle floue comme condition préalable et utilise la règle du langage flou pour déduire une conclusion de jugement floue approximative. Les règles de langage flou incluent le Modus Ponens Généralisé (GMP) et le Modus Tollens Généralisé (GMT).

La règle GMP peut s'exprimer ainsi :

Prérequis 1 : x est A'.

Précondition 2 : Si x est A, alors y est B.

Conclusion : y est B'.

La règle GMT peut être exprimée comme suit :

Prémisse 1 : y est B,

Prémisse 2 : si x est A, alors y est B,

Conclusion : x est A'.

A, A', B et B' dans la formule ci-dessus sont des ensembles flous, et x et y sont des variables linguistiques.

La fonction floue A → B est considérée comme une instruction "si - alors", formant une règle de raisonnement flou.

Le résultat du raisonnement flou est un ensemble flou, mais en pratique, une certaine valeur de sortie est requise. La décision floue est le processus d'obtention d'une valeur unique représentant un ensemble flou. La méthode la plus simple est la méthode d'appartenance

maximale, qui prend la valeur maximale d'appartenance dans tous les ensembles flous en sortie.

L'avantage du système expert basé sur la logique floue est qu'il peut exprimer la haute compétence des compétences expertes et a suffisamment de robustesse, et peut également effectuer un raisonnement heuristique et provisoire. Cependant, ce type de système expert a du mal à obtenir des connaissances et son raisonnement dépend d'une base de connaissances floue, la capacité d'apprentissage n'est pas forte, sujette à l'erreur.

5.2.2. Les systèmes experts basés sur les réseaux de neurones

Le réseau de neurones ou les réseaux de neurones artificiels (ANN) sont un ensemble de modèles d'inspiration biologique qui facilitent l'apprentissage des ordinateurs à partir des données observées. Les neurones biologiques sont la motivation de base pour le développement de réseaux de neurones artificiels. Le cerveau compte environ 100 milliards de neurones, chaque neurone a en moyenne 10 000 connexions qui le relient directement aux autres neurones. Ainsi, il existe environ un million de milliards de ces connexions, ce qui rend le cerveau La structure la plus complexe, naturelle ou artificielle, sur terre.

Les impulsions électriques transmises sur ces connexions créent les interconnexions dans le cerveau ; ce réseau d'interconnexions signifie que l'activation d'un neurone peut influencer une moyenne de 10 000 neurones, créant un nombre immense de modèles « marche/arrêt ».

Un réseau de neurones artificiels (ANN) est un système de traitement de l'information qui adopte la structure neuronale du cerveau humain pour analyser les données, trouver des modèles, la classification et la prédiction grâce à un processus d'apprentissage utilisant une série d'équations mathématiques. Ils sont capables de prendre des décisions en utilisant ce qu'ils apprennent tout en rencontrant des problèmes. Muller & Guido (2017) ont fait référence aux réseaux de neurones comme une famille d'algorithmes qui a récemment connu un renouveau sous le nom de "deep learning". Le concept d'apprentissage en profondeur est très prometteur dans de nombreuses applications d'apprentissage automatique. Le modèle de réseau neuronal est fondamentalement différent du système logique décrit ci-dessus. Dans le réseau de neurones, les connaissances ne sont plus transformées en règles explicites par un traitement manuel, mais sont automatiquement acquises par l'algorithme d'apprentissage et produisent ses propres règles implicites.

Comparé au système expert traditionnel, le réseau neuronal a une fonction plus puissante : il est plus efficace que le fonctionnement en série traditionnel ; il a un certain degré de tolérance aux pannes ; le poids de la connexion au réseau de neurones peut être modifié, etc.

Le réseau de neurones acquiert automatiquement des connaissances grâce à des instances d'apprentissage. Les experts fournissent des exemples et des attentes de la solution, l'algorithme d'apprentissage du réseau neuronal modifie constamment la répartition du poids du réseau, obtenant une sortie stable après la formation. Étant donné que l'entrée et la sortie du réseau de neurones sont numériques, il est nécessaire d'encoder l'instance lors de l'utilisation du réseau de neurones pour acquérir les connaissances.

Le système expert basé sur un réseau de neurones présente également des faiblesses inhérentes : les performances du système sont limitées par l'ensemble d'échantillons d'apprentissage. Dans le cas d'une mauvaise sélection de l'ensemble d'échantillons ou d'un trop petit échantillon, la capacité de raisonnement par induction du réseau de neurones est très faible. De plus, le réseau neuronal est incapable d'expliquer son propre processus de raisonnement et l'importance du stockage des connaissances, car son modèle est basé sur l'activité neuronale superficielle humaine. Différents modèles peuvent être appliqués aux exigences spécifiques du système expert.

5.2.3. Les systèmes experts basés sur les frames

Le système expert basé sur les frames est associé aux objets d'intérêt et le raisonnement consiste à confirmer les attentes pour les valeurs des créneaux. De tels systèmes incluent souvent aussi des règles. Dans un système basé sur de frames, le moteur d'inférence recherche également l'objectif, ou en d'autres termes l'attribut spécifié, jusqu'à ce que sa valeur soit obtenue. Dans un système expert à base de règles, l'objectif est défini pour la base de règles. Dans un système basé sur des frames, les règles jouent un rôle auxiliaire. Les frames représentent ici une source majeure de connaissances, et des méthodes et des démons sont utilisés pour ajouter des actions aux frames. Ainsi, nous pourrions nous attendre à ce que le but d'un système basé sur des frames puisse être établi soit dans une méthode, soit dans un démon [29].

- Un démon a une structure SI-ALORS. Il est exécuté chaque fois qu'un attribut de l'instruction IF du démon change de valeur. Dans ce sens, les démons et les méthodes sont très similaires, et les deux termes sont souvent utilisés comme synonymes.
- Cependant, les méthodes sont plus appropriées si nous devons écrire des procédures complexes. Les démons, en revanche, sont généralement limités aux instructions SI-ALORS.

Le système expert basé sur le Framework se structure comme "frame". Le frame contient le nom du concept, l'emplacement d'étiquette de l'attribut/de la caractéristique principale et les

valeurs possibles de chaque attribut, ou le processus de capture des informations procédurales sur le concept. Lorsqu'une instance particulière du concept est rencontrée, la valeur propre pertinente de l'instance est entrée dans le frame, qui est appelé l'instance. Le cadre peut être utilisé pour le raisonnement. Le frame contient des informations multi-aspects d'un concept qui peuvent être utilisées même si les informations elles-mêmes ne sont pas observées. Par exemple, si nous pouvons ou non voir les racines, parce que "l'arbre" contient la "racine" de l'étiquette, nous pouvons donc penser que l'arbre a une racine.

Dans les systèmes experts basés sur des frames, les frames fournissent un moyen naturel pour la représentation structurée et concise des connaissances. Un frame fournit un moyen d'organiser les connaissances en créneaux pour décrire divers attributs et caractéristiques de l'objet. Les frames sont une application de la programmation orientée objet pour les systèmes experts. Le concept de frames est défini par un ensemble de slots. Chaque slot décrit un attribut particulier ou une opération de la trame. Les slots sont utilisés pour stocker des valeurs. Un créneau peut contenir une valeur par défaut ou un pointeur vers une autre frame, un ensemble de règles ou une procédure par laquelle la valeur du créneau est obtenue. Les frames incluent des informations sur la façon de les utiliser, des informations sur les attentes (qui peuvent s'avérer erronées) et des informations sur ce qu'il faut faire si les attentes ne sont pas confirmées. Des ensembles de tels frames doivent être organisés en systèmes de frames dans lesquels ils sont interconnectés.

5.3. Système expert et acquisition des connaissances

La connaissance est considérée comme l'élément clé de la performance d'un système expert selon Feigenbaum. En addition, l'une des étapes les plus importantes dans le développement des systèmes experts est l'acquisition de connaissance. Cette étape consiste à aider le cogniticien à mettre en point sa base de connaissance.

Pour la construction des systèmes experts de première génération, l'acquisition de connaissances est considérée essentiellement comme une activité de transfert, consistant à traduire l'énoncé d'un expert en une représentation choisie.

Les méthodes de transfert de connaissances dans la première génération des systèmes experts étaient entièrement empiriques. Ensuite elles sont spécifiées, structurées (méthodes analytiques), certaines sont automatisées (méthodes automatiques). Ci-dessous, nous introduisons les deux premières approches [29].

5.3.1. L'approche empirique :

Il s'agit d'une approche axée sur l'implémentation dans laquelle les cogniciens collaborent avec des experts afin de développer des modèles de systèmes réels. Plus précisément, les cogniciens procèdent de manière itérative à un prototypage rapide, en entrant dans un cycle « d'extraction des connaissances, de modélisation des connaissances extraites, ainsi que de validation de la base de connaissances » jusqu'à ce que les connaissances soient correctes.

5.3.2. L'approche structurée (ou analytique) :

Il s'agit d'une approche pilotée par les modèles, plus structurée que les méthodes empiriques. Le problème est défini puis l'expertise rassemblée est modélisée avant de concevoir le système expert. Les approches analytiques de l'acquisition des connaissances nécessitent la présence d'experts cognitifs. Plusieurs approches ont été utilisées pour extraire des connaissances. La plupart de ces méthodes sont issues de recherches menées en ingénierie des connaissances. Parmi ces méthodes, KADS et KOD sont suffisamment générales pour intervenir dans tout type de domaine, considérer le processus d'acquisition des connaissances et proposer respectivement des modèles de connaissances expertes.

5.3.2.1.KADS (Knowledge Acquisition and Design Structuring):

L'analyse complète des données précède la conception et la mise en œuvre du système expert, contrairement au prototypage rapide. Dans l'approche KADS, le processus de création d'un système expert est considéré comme une activité, en d'autres termes, une série de transformations de modèles. Un atelier logiciel guidant la démarche cognitive experte accompagne cette démarche à travers :

- Un module permettant de donner des conseils concernant l'utilisation de la méthodologie.
- Des modules permettant la structuration ainsi que l'analyse des interviews.
- La conceptualisation des données en formes d'entités et relations.
- La proposition de plusieurs modèles concernant l'interprétation.

5.3.2.2.KOD (Knowledge Oriented Design) :

La méthode KOD repose sur des techniques issues de la linguistique. Elle est basée 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

accompagnée d'un logiciel qui fournit des outils informatiques afin de garantir la mise en œuvre de ses différentes étapes, indépendamment de l'implémentation.

5.4. Méthode de construction d'un système expert

Le processus de construction d'un système expert peut être grossièrement divisé en trois étapes : l'acquisition des connaissances, l'ingénierie des connaissances et les tests. Au cours de l'acquisition des connaissances, le ou les ingénieurs de connaissances et les experts identifient et formulent ensemble le contenu requis de la base de connaissances. L'acquisition de connaissances implique de nombreuses discussions entre l'expert et l'ingénieur du savoir. C'est une étape très difficile dans la construction de systèmes experts. Parce que l'interaction personnelle entre l'ingénieur de la connaissance et l'expert est cruciale, chaque processus d'acquisition de connaissances à ses propres particularités. Par conséquent, un processus d'acquisition de connaissances généralement applicable et structuré n'a pas encore été dérivé.

Une fois que les connaissances ont été extraites de l'expert, elles doivent être structurées de manière à pouvoir être mises en œuvre dans l'outil de construction de système expert. L'ensemble des étapes de structuration et de mise en œuvre s'appelle l'ingénierie des connaissances. L'ingénierie des connaissances comprend la sélection d'un schéma de représentation des connaissances et d'un moteur d'inférence appropriés. L'ingénieur de la connaissance fait la sélection la plus appropriée dans les discussions avec l'expert. Si ces discussions ne conduisent pas à un choix satisfaisant de représentation des connaissances, il peut être avantageux de construire un certain nombre de (petits) systèmes prototypes pour pouvoir comparer les différences de caractéristiques de chaque représentation.

Lorsqu'une première version d'un système a été construite, le processus de test peut commencer. En général, les tests impliquent la validation de la base de connaissances par l'expert et l'évaluation du système par un ou plusieurs utilisateurs. Pendant la phase de validation, la première version sera testée par l'expert pour voir si des résultats inattendus se produisent. Si ce n'est pas (ou plus) le cas, le système peut être évalué par d'autres experts quant à la validité des raisonnements employés et par de futurs utilisateurs quant à son utilisation en pratique.

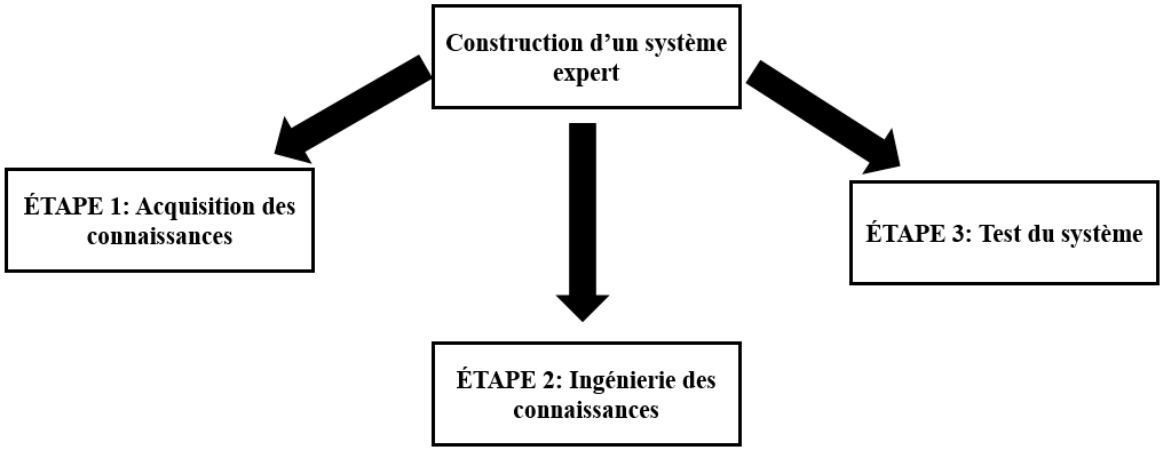


Figure 5.2. Étapes de construction d'un système expert

Prototype d'examen final

Université Abbès Laghrour – Khenchela- **Le 27/02/2019**
1ère année Master STW **Durée : 01h30**

Examen Final : Système Expert

Question de cours :3p

1- Citez 3 types avec exemples concernant les ensembles dans les graphes conceptuels.
2- Quelle est la différence entre un aspect reflexe et un aspect de contrôle dans un frame ? (Réponse avec exemples).

Exercice 1 : 6p

R1 : s'il fait beau aujourd'hui alors il fera beau demain
R2 : s'il pleut aujourd'hui alors il y a des nuages
R3 : si le vent est d'ouest alors les hirondelles volent bas
R4 : s'il y a des nuages alors le temps est à la pluie
R5 : si les araignées tissent des toiles alors les chiens pressentent la pluie
R6 : si les hirondelles volent bas alors les loups pressentent la pluie
R7 : si (les loups ou les chiens pressentent la pluie) et s'il y a des nuages alors il pleuvra demain
R8 : s'il pleut en Bretagne et le vent est d'ouest alors le temps est à la pluie
R9 : si le temps est à la pluie alors il pleut aujourd'hui.
R10: si les loups pressentent la pluie alors il fait beau aujourd'hui

4- Modélisez les règles suivantes en logique des prédicats.
5- Pouvons-nous savoir s'il pleuvra demain à partir de la BF {le vent est d'ouest, il pleut en Bretagne} en utilisant le chainage avant en profondeur ?
6- Quels sont les faits déductibles en utilisant le chainage avant en largeur à partir de la BF {Il pleut aujourd'hui, les loups pressentent la pluie, le vent est d'ouest} ?

Exercice 2 : 6p

Soit les Concepts suivants : Voiture, Panne, Roue, Vélo.

Soit les Rôles suivants: a-unePartie, contrôler.

Représentez les expressions suivantes en logique de description :

1- Une voiture qui a une partie en panne est en panne.
2- Les voitures ont entre 3 et 4 roues.
3- Les vélos ont exactement deux roues.

Exercice 3: 5p

''Le real Madrid, Juventus, et Chelsea sont des clubs de Football. Le real Madrid propose deux moyens de payement à ces joueurs : Paie mensuelle et annuelle. Ronaldo, joueur du real Madrid pense que la Juventus propose un moyen de payement plus satisfaisant que ceux proposés par le real Madrid.''
Représentez ces informations en un seul réseau sémantique.

Bon courage,

Remarque
Concernant Exercice 1, vous pouvez réécrire les faits de la façon suivante pour ne pas perdre de temps :
Il fait beau aujourd'hui = FBA
Il fera beau demain = FBD
Il pleut aujourd'hui alors = PA
Il y a des nuages = N
Le vent est d'ouest = VO
Les hirondelles volent bas =HVB
Le temps est à la pluie = TP
Les araignées tissent des toiles = ATT
Les chiens pressentent la pluie =CPP
Les loups pressentent la pluie=LPP
Il pleuvra demain=PD
Il pleut en Bretagne=PB

Références

[1] CHARNIAK, Eugene et MCDERMOTT, Drew. Introduction to Artificial. *Intelligence (Addison Wesley, Reading, MA, 1984)*, 1985.

[2] POOLE, David I., GOEBEL, Randy G., et MACKWORTH, Alan K. *Computational intelligence*. New York : Oxford University Press, 1998.

[3] RICH, Elaine, KNIGHT, Kevin, WINSTON, P. H., *et al.* Artificial Intelligence TATA McGRAW-HILL. 1991.

[4] BELLMAN, Richard. *An introduction to artificial intelligence: can computers think?*. Thomson Course Technology, 1978.

[5] WINSTON, Patrick Henry. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[6] NILSSON, Nils J. et NILSSON, Nils Johan. *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.

[7] MCCULLOCH, Warren S. et PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 1943, vol. 5, no 4, p. 115-133.

[8] MINSKY, Marvin. *Heuristic aspects of the artificial intelligence problem*. Ed. Services Technical Information agency, 1956.

[9] EVANS, Thomas G. *A program for the solution of a class of geometric-analogy intelligence-test questions*. Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, 1964.

[10] RAPHAEL, Bertram. *Robot Research at Stanford Research Institute*. STANFORD RESEARCH INST CA, 1972.

[11] LIGHTHILL, James, *et al.* Artificial intelligence: a paper symposium. *Science Research Council, London*, 1973.

[12] MARVIN, Minsky et SEYMOUR, A. Papert. Perceptrons. *Cambridge, MA: MIT Press*, 1969, vol. 6, p. 318-362.

[13] BUCHANAN, Bruce, SUTHERLAND, Georgia, et FEIGENBAUM, Edward A. Heuristic DENDRAL: A program for generating explanatory hypotheses. *Organic Chemistry*, 1969.

[14] SHORTLIFFE, Edward H. Mycin: A knowledge-based computer program applied to infectious diseases. In : *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association, 1977. p. 66.

[15] PINEDA, Fernando. Generalization of back propagation to recurrent and higher order neural networks. In : *Neural information processing systems*. 1987.

[16] AVRON, Barr et FEIGENBAUM, Edward A. The handbook of artificial intelligence. 1982.

[17] BELLMAN, Richard. A Markovian decision process. *Journal of mathematics and mechanics*, 1957, p. 679-684.

[18] BERLINER, H. J. et EBELING, Carl. Hitech. In : *Computers, Chess, and Cognition*. Springer, New York, NY, 1990. p. 79-109.

[19] SHOENFIELD, Joseph R. *Mathematical logic*. AK Peters/CRC Press, 2018.

[20] POSPESEL, Howard. Introduction to logic: Propositional logic. 1974.

[21] KOWALSKI, Robert. Predicate logic as programming language. In : *IFIP congress*. 1974. p. 569-544.

[22] HOBART, Michael E. et RICHARDS, Joan L. De Morgan's logic. *British Logic in the Nineteenth Century*, 2008, vol. 4, p. 2004-14.

[23] Bernard ESPINASSE. Représentation des Connaissances : Introduction aux Réseaux Sémantiques, Université d'Aix-Marseille, France 2008.

[24] Bernard ESPINASSE. Représentation des connaissances : Introduction aux Graphes Conceptuels, Université d'Aix-Marseille, France 2008.

[25] Bernard ESPINASSE. Représentation des connaissances : Introduction aux Frames, Université d'Aix-Marseille, France 2008.

[26] Bernard ESPINASSE. Introduction aux Logiques de Description (LD), Université d'Aix-Marseille, France 2016.

[27] FORSYTH, Richard (ed.). *Expert systems: principles and case studies*. Chapman & Hall, Ltd., 1984.

[28] CLARKE, Desmond M. Descartes' philosophy of science. Manchester University Press, 1982.

[29] Mehenaoui, Zohra. Systemes Experts. Université 08 Mai 1945, Guelma, Algeria, 2021.

[30] Olayanju Taiwo Abolaji. Introduction to Expert Systems. NATIONAL OPEN UNIVERSITY OF NIGERIA. Course Code: CIT 474.

[31] JACKSON, Peter. Introduction to expert systems. 1986.

[32] MARTIN, William A. et FATEMAN, Richard J. The MACSYMA system. In : *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*. 1971. p. 59-75.

[32] GASCHNIG, J. Prospector: an expert system for mineral exploration. *D. Michie, ed*, 1982, vol. 1982, p. 47-64.

[33] ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958, vol. 65, no 6, p. 386.

[34] NEWELL, Allen, SHAW, John C., et SIMON, Herbert A. Report on a general problem solving program. In : *IFIP congress*. 1959. p. 64.

[35] MCCARTHY, John. History of LISP. In : *History of programming languages*. 1978. p. 173-185.

[36] ZADEH, Lotfi A. Fuzzy logic. Computer, 1988, vol. 21, no 4, p. 83-93.

[37] QUILLIAN, M. Ross. Semantic networks. 1968.

[38] COLMERAUER, Alain et ROUSSEL, Philippe. The birth of Prolog. In : *History of programming languages---II*. 1996. p. 331-367.

[39] CLANCEY, William J. *Knowledge-based tutoring: The GUIDON program*. MIT press, 1987.

[40] ULUG, Fikret. *Emycin-Prolog expert system shell*. NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 1986.

[41] MINSKY, Marvin. A framework for representing knowledge. 1974.

[42] BUCHANAN, Bruce G. et FEIGENBAUM, Edward A. DENDRAL and Meta-DENDRAL: Their applications dimension. *Artificial intelligence*, 1978, vol. 11, no 1-2, p. 5-24.

[43] BERSTEL, Bruno. Extending the RETE algorithm for event management. In : *Proceedings Ninth International Symposium on Temporal Representation and Reasoning*. IEEE, 2002. p. 49-51.

[44] TÜSHAUS, U. Decision Support for Qualitative Data Analysis—KEE Shell Linked to FORTRAN Programs—. In : Data, Expert Knowledge and Decisions. Springer, Berlin, Heidelberg, 1988. p. 204-212.

[45] WYGANT, Robert M. CLIPS—a powerful development and delivery expert system tool. *Computers & industrial engineering*, 1989, vol. 17, no 1-4, p. 546-549.

[46] DURKIN, John. Research review: Application of expert systems in the sciences. 1990.