République Algérienne Démocratique et Populaire
Ministre de L'Enseignement Supérieur et de la Recherche Scientifique

Université Aderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

Support de Cours

Ingénierie des Connaissances

Niveau : 3ème année Licence Informatique à recrutement national
Spécialité : Système d'Information

Etabli par :
Dr. Achour ACHROUFENE

2024/2025

Préface

La connaissance est un élément clé en l'intelligence artificielle au point où toute une ingénierie lui est dédiée : l'Ingénierie des Connaissances. Typiquement, l'ingénierie des connaissances modélise les connaissances en construisant des modèles formels destinés à être mis en œuvre par l'informatique. Les connaissances manipulées sont celles que détiennent les experts du domaine qu'ils mobilisent pour réaliser leurs tâches.

Sur un autre plan, l'intelligence artificielle (IA) a envahi tous les domaines en proposant des systèmes intelligents. Comme conséquence, l'enseignement de l'ingénierie des connaissances comme prélude permettra aux étudiants de Licence informatique de mieux appréhender les différentes spécialités de l'IA, plus particulièrement l'apprentissage automatique, dans la suite de leurs études en Master ou de s'en servir dans le développement de systèmes intelligents dans le parcours professionnel. En effet, ce support de cours abordera les étapes du cycle de développement d'un système à base de connaissances ou d'ontologie en focalisant sur la conceptualisation et la modélisation des connaissances.

Ce support de cours a le mérite de présenter clairement :

1. L'essence de l'ingénierie des connaissances et l'importance des types et sources de connaissances ;
2. La relation de l'ingénierie des connaissances avec d'autres disciplines, en particulier l'acquisition de connaissances et l'apprentissage artificiel ;
3. Un aperçu des modes de représentation de connaissances (graphiques, logiques, langages, etc.) utilisés pour la construction de modèles formels traduisables en programmes informatiques ;
4. Les principes de la famille des logiques de description, en particulier la logique de description minimale ;
5. Le cycle de développement d'un système intelligent dans le cadre de l'ingénierie des connaissances, soit un système à base de connaissances, soit une ontologie.

Table des matières

Introduction **1**

1 Introduction à l'Ingénierie des Connaissances **2**
1.1 Introduction . . . 2
1.2 Définitions de l'IC . . . 2
1.3 Rôle de l'ingénierie des connaissances . . . 3
1.4 Démarche générale de développement . . . 4
1.5 Information et connaissance . . . 5
1.6 Connaissance . . . 5
1.6.1 Principales sources de la connaissance . . . 5
1.6.2 Type de connaissances . . . 7
1.7 Acquisition de connaissances . . . 8
1.7.1 Définitions . . . 8
1.7.2 Acquisition des connaissances et sciences cognitives . . . 8
1.8 Apprentissage automatique . . . 9
1.8.1 Apprentissage naturel . . . 9
1.8.2 Apprentissage automatique . . . 9
1.8.3 Types d'apprentissage . . . 10
1.8.4 Acquisition des connaissances vs Apprentissage automatique . . . 11
1.9 Exercices corrigés . . . 11
1.9.1 Énoncés . . . 11
1.9.2 Solutions . . . 12
1.10 Conclusion . . . 12

2 Représentation de Connaissances **13**
2.1 Introduction . . . 13
2.2 Modes de représentation des connaissances . . . 13
2.3 Formalismes de représentation des connaissances . . . 13
2.3.1 Réseaux sémantiques . . . 14
2.3.2 Frames (schémas) . . . 19
2.3.3 Scripts . . . 20
2.3.4 Graphes conceptuels . . . 23
2.4 Exercices corrigés . . . 24
2.4.1 Énoncés . . . 24
2.4.2 Solutions . . . 25
2.5 Conclusion . . . 26

3 Les formalises logiques de représentation de connaissances 27
3.1 Introduction . . . 27
3.2 Logiques classiques . . . 27
3.2.1 Logique des propositions . . . 27
3.2.2 Logique des prédicats du premier ordre . . . 28
3.3 Logiques non classiques . . . 30
3.3.1 Logiques multivaluées . . . 30
3.3.2 Logiques modales . . . 30
3.3.3 Logiques non-monotones . . . 31
3.3.4 Raisonnement en présence de l'incertitude . . . 31
3.4 Logiques de description . . . 31
3.4.1 Principes des LD . . . 32
3.4.2 Niveau terminologique (TBox) . . . 33
3.4.3 Niveau factuel (ABox) . . . 34
3.4.4 Famille de LD . . . 34
3.4.5 Logique minimale *AL* . . . 35
3.5 Exercices corrigés . . . 37
3.5.1 Énoncés . . . 37
3.5.2 Solutions . . . 37
3.6 Conclusion . . . 38

4 Les langages de représentation de connaissances 39
4.1 Introduction . . . 39
4.2 Knowledge Interchange Format (KIF) . . . 39
4.2.1 Expressions en KIF . . . 40
4.2.2 Règles . . . 40
4.3 EXtensible Markup Language (XML)) . . . 41
4.4 RDF . . . 42
4.4.1 Entités de base de RDF . . . 43
4.4.2 Exemples RDF graphiques . . . 44
4.4.3 Syntaxe de base de RDF . . . 45
4.4.4 Types de containers RDF . . . 49
4.5 Les systèmes de représentation des connaissances . . . 50
4.5.1 KL-One . . . 50
4.5.2 CLASSIC . . . 51
4.5.3 LOOM . . . 52
4.6 Exercices corrigés . . . 52
4.6.1 Énoncés . . . 52
4.6.2 Solutions . . . 53
4.7 Conclusion . . . 54

5 Les Systèmes à Bases de Connaissances 55
5.1 Introduction . . . 55
5.2 Concepts de base . . . 55
5.2.1 Connaissances . . . 56
5.2.2 Raisonnement . . . 57

5.3 Système expert à base de règles de production . . . 58
5.3.1 Définition d'un système expert . . . 58
5.3.2 Architecture d'un SE . . . 58
5.3.3 Base de connaissances . . . 59
5.3.4 Moteur d'inférence . . . 59
5.3.5 Autres modules . . . 60
5.3.6 Domaines d'applications des systèmes experts . . . 60
5.4 Caractéristiques d'un moteur d'inférence . . . 62
5.4.1 Cycle de base . . . 62
5.4.2 Modes de raisonnement du MI . . . 63
5.4.3 Autres caractéristiques du MI . . . 65
5.5 Méthodologie de développement d'un SBC . . . 66
5.5.1 Développement d'un SBC . . . 66
5.5.2 Rôles des acteurs d'un projet système expert . . . 73
5.5.3 Générateurs de systèmes experts . . . 74
5.6 Exercices corrigés . . . 75
5.6.1 Énoncés . . . 75
5.6.2 Solutions . . . 75
5.7 Conclusion . . . 76

6 Les Ontologies 77
6.1 Introduction . . . 77
6.2 Qu'est-ce qu'une ontologie ? . . . 78
6.2.1 Définitions formelles des ontologies . . . 78
6.2.2 Composants d'une ontologie . . . 79
6.2.3 Formalisation d'ontologies . . . 81
6.3 Types d'ontologies . . . 82
6.4 Caractéristiques des ontologies . . . 83
6.5 Ingénierie ontologique . . . 84
6.5.1 Principes . . . 84
6.5.2 Cycle de vie d'une ontologie . . . 85
6.5.3 Méthodologies de l'ingénierie ontologique . . . 86
6.5.4 Langages formels pour la représentation des ontologies . . . 89
6.5.5 Environnements de développement des ontologies . . . 89
6.6 Exercices corrigés . . . 90
6.6.1 Énoncés . . . 90
6.6.2 Solutions . . . 90
6.7 Conclusion . . . 91

Conclusion 93

Références 93

Introduction

Le développement de systèmes informatiques en entreprise, de gestion tout au début, est une pratique qui date de l'avènement de l'informatique. La première et la deuxième génération de ces systèmes se contentaient de traiter les données et les informations, respectivement, ce qui a entrainé leurs limites rapidement. L'émergence des systèmes experts à la fin des années 70 a vu plutôt la manipulation des connaissances et l'introduction du concept de raisonnement.

Pour collecter les connaissances requises aux systèmes experts, nous faisons appel à la discipline de l'acquisition de connaissances, pour les conceptualiser, les modéliser et les implémenter, nous nous appuyant sur la discipline de l'ingénierie des connaissances (IC). Pour ce faire, l'IC interagit avec plusieurs branches, notamment les sciences cognitives pour conceptualiser la connaissance, l'intelligence artificielle pour la modélisation, la logique pour le raisonnement, ainsi que l'informatique pour l'opérationnalisation. Ce document s'inscrit dans ce contexte en présentant aux étudiants : les fondements de l'ingénierie des connaissances, les modes de représentation de connaissances, les systèmes à base de connaissances et les ontologies.

Ce rapport est organisé en six chapitres :

Le **premier chapitre** présente, tout d'abord, certaines définitions en relation avec l'IC, ensuite introduit la notion de connaissance, les types et les sources de connaissances. Il aborde également les disciplines de l'acquisition des connaissances de l'apprentissage artificiel.

Le **deuxième chapitre**, premièrement, introduit les modes de représentation des connaissances, ensuite se focalise sur les formalismes de représentation de connaissances non logiques en l'occurrence, les réseaux sémantiques, les graphes conceptuels, les frames et enfin les scripts.

Dans le **troisième chapitre**, nous présentons les logiques classiques, notamment la logique des propositions et la logique des prédicats, ensuite les logiques non classiques en mettant l'accent sur la famille des logiques de description.

Le **quatrième chapitre** présente les langages de représentation de connaissances permettant l'échange et l'interopérabilité entre les systèmes tels que KIF, XML et RDF ainsi que quelques systèmes de représentation de connaissances comme KL-ONE et CLASSIC.

Les systèmes à bases de connaissances (SBC) constituent le **cinquième chapitre**. Après définition des composants principaux des SBC, nous présentons en détail le cas particulier des systèmes experts à base de règles de production.

Le **sixième chapitre** a pour objet l'ingénierie ontologique. Il aborde les objectifs des ontologies, les types d'ontologies ainsi que leurs caractéristiques. De plus, il s'attarde sur le cycle, les méthodes, les langages et les outils de développement des ontologies.

A noter aussi que chaque chapitre se termine par une série d'exercices ainsi que leurs corrigés. Et cela dans l'objectif de rendre le document encore plus utile. De plus, ces exercices peuvent donner l'occasion à l'étudiant de tester les connaissances acquises et de vérifier sa compréhension.

Chapitre 1

Introduction à l'Ingénierie des Connaissances

1.1 Introduction

Ce chapitre a pour objectif l'introduction de l'ingénierie des connaissances (IC). Tout d'abord, il va donner certaines définitions en relation avec l'IC, ensuite il présente la notion de la connaissance, les types et les sources de connaissances. Il aborde également une autre discipline en relation directe avec l'IC qui est l'acquisition des connaissances. Ce chapitre se termine par un bref aperçu sur l'apprentissage naturel et l'apprentissage artificiel.

1.2 Définitions de l'IC

La première question que l'on peut se poser est "c'est quoi l'Ingénierie des connaissances ?" La réponse immédiate est : l'IC est une partie de l'intelligence artificielle (IA). Ce terme (Knowledge Engineering en anglais) est introduit par E. A. Feigenbaum en 1977 dans le contexte de l'IA [1].

Plus formellement, l'IC est composé de deux mots :

— **Ingénierie** : ensemble de techniques et de méthodes appliquées pour la résolution de problèmes complexes.

— **Connaissance** : « ensemble des notions et des principes qu'une personne acquiert par l'étude, l'observation ou l'expérience et qu'elle peut intégrer à des habiletés ».

Nous trouvons diverses définitions dans la littérature. D'une manière générale, l'IC fournit une démarche d'analyse et de modélisation d'une résolution de problèmes [2].

Une définition plus technique est celle donnée par [3] : "C'est l'étude des concepts, méthodes et techniques permettant de modéliser et/ou d'acquérir les connaissances pour des systèmes réalisant ou aidant des humains à réaliser des tâches se formalisant *a priori* peu ou pas".

En informatique, l'IC est définit par [1] "l'art d'acquérir, de modéliser et de représenter la connaissance en vue de son utilisation par des ordinateurs qui accomplissent ainsi des tâches dites intelligentes".

En résumé, l'IC correspond à l'étude des méthodes et techniques permettant de manipuler et/ou de construire des systèmes informatiques utilisant le concept de la connaissance.

1.3 Rôle de l'ingénierie des connaissances

D'après la dernière définition, le rôle ou l'objectif de l'IC est de proposer des méthodes et des outils pour le traitement de la connaissance :

— premièrement, pour la modélisation cognitive [1] et la modélisation conceptuelle [2] ;
— deuxièmement, des langages [3] de modélisation et de représentation ;
— enfin, des méthodes de travail sur corpus [4], etc.

Et cela dans des domaines connexes comme :

— l'acquisition des connaissances à partir de textes ;
— la recherche de l'information sur le Web ;
— la gestion et capitalisation de connaissances en entreprise ;
— la définition d'indicateurs de gestion et leur mise en œuvre dans des systèmes d'information.

Ainsi, nous constatons que l'IC se situe au carrefour de plusieurs réflexions, les plus notables sont :

— la linguistique qui étudie la formulation linguistique des connaissances,
— la terminologie et l'ontologie qui permet de dégager les concepts,
— la psychologie pour élaborer les méthodes d'élicitation,
— la logique pour élaborer les modèles formels,
— l'informatique afin d'opérationnaliser ces modèles,
— la sémiotique pour interpréter et s'approprier le comportement du système, etc.

En outre, ces disciplines se fécondent réciproquement et conduisent à faire évoluer le paradigme. La figure 1.1 donne un autre exemple de disciplines contribuant à évoluer l'IC [4].

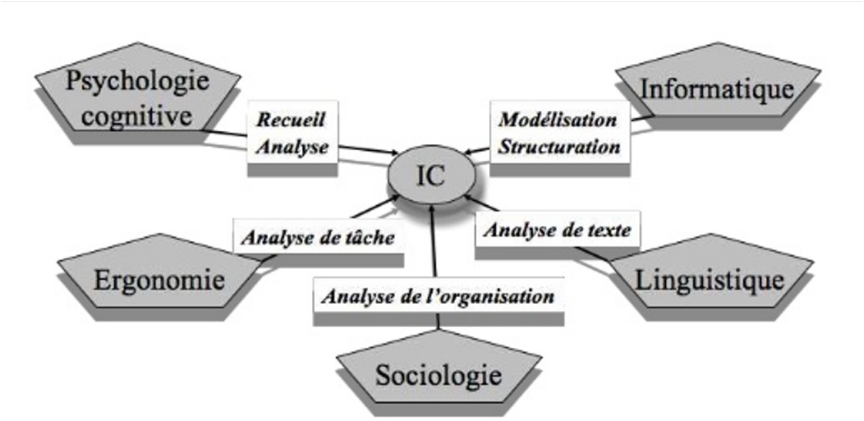


FIGURE 1.1 – L'IC, carrefour de plusieurs disciplines.

1. Modélisation cognitive : processus psychologiques.
2. Conceptuel : au niveau de la conception.
3. Langage : ensemble de symboles et de notations régit par des règles de construction.
4. Corpus : ensemble fini de textes choisi comme base d'une étude.

1.4 Démarche générale de développement

La Figure 1.2 illustre la démarche de développement d'un système d'ingénierie des connaissances. Nous pouvons distingué quatre étapes principales : le recueil, la modélisation et la représentation des connaissances, l'implémentation et la validation.

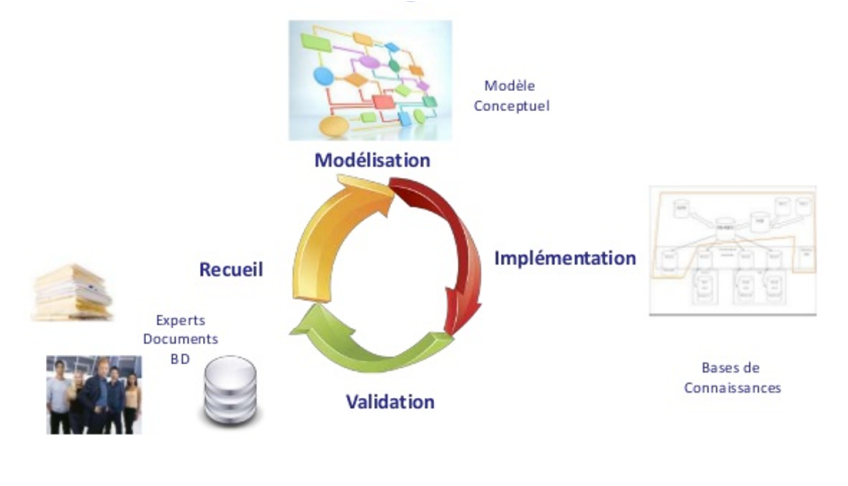


FIGURE 1.2 – Démarche de développement d'un système IC.

Recueil des connaissances

C'est d'abord identifier le domaine d'expertise et ses activités afin de collecter les connaissances. Ensuite, plusieurs techniques existent pour le recueil telles que : l'entretien libre avec l'expert, le recueil dirigé par des questionnements de l'expert, l'extraction de concepts à partir de texte « TextMining » [5] (des algorithmes sont appliqués à des corpus de texte), l'agrégation de données par d'application d'algorithmes de « DataMining » [6].

Modélisation et représentation des connaissances

Cette modélisation décrit un modèle abstrait qu'on appelle modèle conceptuel, cadre sémantique partagé entre l'expert, l'ingénieur de la connaissance, les utilisateurs destinataires de la formalisation des connaissances et dans certaines cas, l'informaticien programmeur d'un système à base de connaissances. Le langage CML « Conceptual Modelling Language » [7], par exemple, permet de représenter le modèle de domaine sous forme de concepts et le modèle de raisonnement avec des tâches et des inférences.

Implémentation et validation

Il s'agit de traduire les modèles conceptuels dans un langage de programmation de haut niveau compréhensible par la machine. Ensuite, vient l'étape de validation auprès de l'utilisateur final par un jeux de tests. La constatation d'anomalies ou d'un besoin d'amélioration conduit à ré-appliquer le processus dés le début.

1.5 Information et connaissance

En informatique, la première vague des systèmes traitaient uniquement les données numériques, alors que la deuxième vague s'intéressaient aux banques d'informations et de documents. À partir des 80, une nouvelle génération de systèmes informatiques (les systèmes à base de connaissances) est arrivée à traiter des connaissances de plus haut niveau.

— Ils peuvent représenter et traiter des principes et des règles de décision, des taxonomies, des théories, des processus et des méthodes mémorisées dans l'ordinateur.
— Ils sont capables d'aider l'usager à accomplir des tâches de façon plus intelligente.

Ces systèmes sont passées graduellement du traitement de données, au traitement des informations, ensuite des connaissances. Ces notions definies comme suit :

— **Données** : des éléments brutes qui n'ont subit aucun traitement. Sans contexte, elles n'ont pas de sens.
— **Informations** : c'est la mise en contexte d'une donnée.
— **Connaissances** : résultat de toute construction mentale effectuée par un individu à partir d'informations ou d'autres stimuli.

Ce niveau «cognitif» [5] a été développé davantage dans les systèmes d'information et dans la pratique des individus et des organisations.

1.6 Connaissance

Une autre définition répondue de la **connaissance** est «un ensemble de concepts et de liens relationnels possibles, acquis à partir des informations disponibles sur divers supports et/ou des expériences vécues.»

Ainsi, la connaissance est de plusieurs types et a plusieurs origines, elle peut être extraite :

— Des sources d'information variées ;
— De l'expertise ;
— Des explications ;
— Des stratégies ;
— De la façon de faire (savoir-faire).

Les principales sources de la conaissance sont données par la Figure 1.3.

1.6.1 Principales sources de la connaissance

La connaissance peut être donnée d'une manière :

— Explicite : sur des supports d'information de toute nature ;
— Implicite : par les spécialistes et les experts d'un domaine.

1.6.1.1 Sources implicites (Experts)

Les experts sont la source la plus importante des connaissances :

— Les experts détiennent une connaissance d'une importance stratégique.
— Cette connaissance est validée et surtout renouvelée et enrichie à chaque nouvelle expérience et/ou information.

5. Cognitif : Qui se rapporte à la faculté de connaître

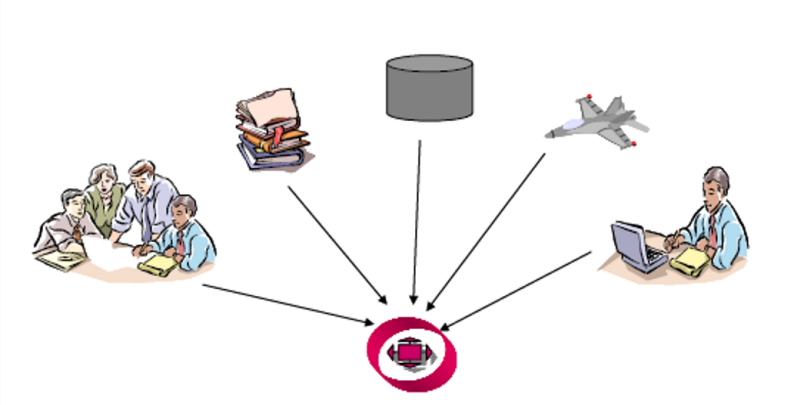


FIGURE 1.3 – Les principales sources de la connaissances.

— Elle est dynamique puisque des informations peuvent être fournies au fur et à mesure que le besoin est exprimé.
— L'extraction des connaissances se fait par les questionnaires, les interviews, ...

1.6.1.2 Sources explicites

Les sources explicites sont les documents, les bases de données, les vidéos, les enregistrements (sonores), ...

Les documents : sont les principaux moyens de sauvegarde et conservation systématique des connaissances. Ils sont susceptibles de contenir des connaissances complémentaires à celle des experts et souvent sont les seuls supports disponibles. L'extraction des connaissances à partir des données textuelles est plus complexe comparativement à partir des experts avec lequel s'offre la possibilité de dialogue ; ainsi il faut extraire cette connaissance en focalisant sur sa pertinence.

Les bases de données : constituent la mémoire des entreprises. On retrouve dans les organisations et les entreprises de très grandes masses de données, accumulées durant des années et couvrant toute leurs activités, organisées sous forme de bases de données et de fichiers. Elles représentent des sources formelles de connaissance.

Les documents vidéo : font appel au sens visuel et auditif de l'être humain. Ils sont de plus en plus utilisés dans les organisations dans divers domaines. Ils servent de support d'enseignement, pour présenter des modes opératoires, des suivis d'expériences, pour présenter d'experts en action de diagnostic, etc.

Les documents sonores : ciblent un seul sens humain qui est l'audition et donc moins difficiles à analyser comparativement aux documents vidéos. Ils sont utilisés pour mémoriser toute information sonore telle que l'échantillonnage de voix dans le cas d'expériences scientifiques, bruits caractéristiques d'outils, etc.

1.6.2 Type de connaissances

Un individu utilise différents types de connaissances dans ses activités quotidiennes : déclaratives et procédurales. La Figure 1.4 montre la hiérarchie des types de connaissances.

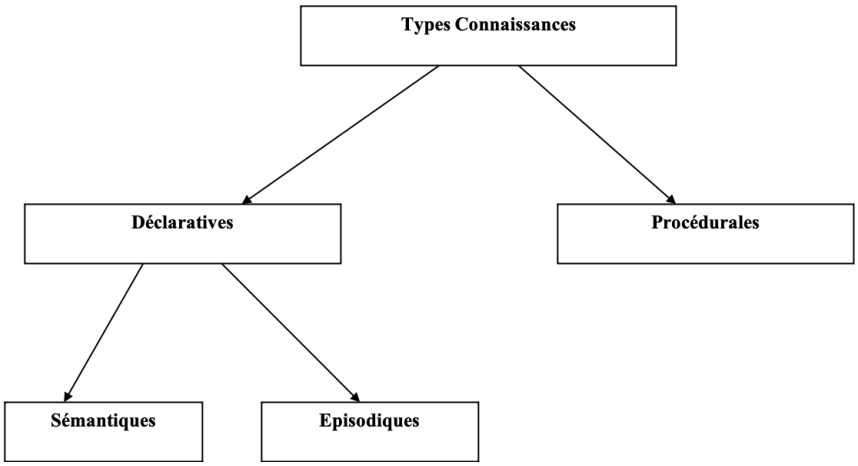


FIGURE 1.4 – Hiérarchie des connaissances

1.6.2.1 Connaissances déclaratives

Les connaissances déclaratives servent à identifier et définir les éléments de l'environnement. Elles portent sur les faits, les objets et les choses. Elles sont relativement statiques et permettent de répondre à la question «quoi ?». Exemples :

- Un être humain possède deux pieds (fait)
- Fatma a des cheveux noirs (fait)
- Les oiseaux pondent des œufs (généralisation)

Les connaissances déclaratives sont divisée en deux types secondaires : sémantique et épisodique.

La connaissance sémantique est une connaissance linguistique ou verbale. L'ensemble de ces connaissances constitue l'inventaire des mots et des significations qu'un individu utilise pour s'exprimer.

La connaissance épisodique porte sur des événements ou des expériences personnelles qui sont des éléments d'informations temporelles et contextuelles. Par exemple «le souvenir d'avoir rencontré une personne hier soir à la salle de lecture».

1.6.2.2 Connaissances procédurales

Les connaissances procédurales sont utilisées pour accomplir certaines actions. Elles portent donc sur des actions ou sur des habilités cognitives ou perceptivo-motrices orientées vers un objectif. Elles permettent de répondre à la question «comment ?». Tout simplement, c'est le «comment faire» ou le «savoir faire». Exemples :

- savoir cuisiner
- savoir nager
- savoir appliquer le théorème de Pythagore sur un triangle

Cette connaissance possède un caractère plus dynamique que la connaissance déclarative. Le résultat de l'activation de cette connaissance est la transformation d'information.

1.7 Acquisition de connaissances

1.7.1 Définitions

L'**acquisition de connaissances** peut se définir comme le transfert et la transformation d'une expertise d'une source de connaissances (humaine ou documentaire) à un programme.

Une autre définition de l'**acquisition de connaissances** répondue en IA est la définition et la mise en place de moyens méthodologiques et techniques pour fournir au système les connaissances de l'expert.

Nous notons que la 1re définition se focalise sur le processus de l'acquisition de l'expertise et la 2ème inclut de plus la conception de ce processus. En définitif, l'ultime **rôle** d'acquisition des connaissances est de fournir au système les connaissances qui lui donneront son expertise.

Souvent, l'acquisition des connaissances est confiée à un acteur aux compétences multiples appelé *cogniticien*. Lorsque l'acquisition se fait par l'intermédiaire d'*entretiens* entre l'expert et le cogniticien, on parle d'**élicitation de connaissances**.

1.7.2 Acquisition des connaissances et sciences cognitives

Dans la pratique, l'acquisition des connaissances fait face à plusieurs obstacles :
- la complexité de l'expertise,
- selon sa nature, la connaissance est difficile à expliciter sous forme verbale,
- la méconnaissance du cogniticien des processus mentaux qui composent la connaissance.
- enfin, ce n'est pas seulement un problème d'informatique mais également de psychologie.

D'autres difficultés surviennent après l'étape de l'acquisition telles que :
- la représentation des connaissances recueillies,
- et le passage de données verbales à une forme logique implémentable dans le système, etc.

Nous pouvons ainsi conclure que l'acquisition des connaissances se situe au cœur des sciences cognitives, car elle s'intéresse aux processus cognitifs humains de compréhension et de résolution de problèmes pour les reproduire. Ce problème peut être abordé :
- partiellement par l'intelligence artificielle,
- enrichi par la psychologie cognitive (modèles de raisonnement humain, modes de recueil des connaissances),
- et enrichi encore par l'ergonomie (étude de l'activité de l'expert et du futur utilisateur),
- enfin enrichi davantage par la linguistique (pour rendre plus efficace l'exploitation de documents ou guider l'interprétation de données verbales).

Réciproquement, ces disciplines se nourrissent de nouvelles problématiques associées aux questions qui se posent lors de l'acquisition de connaissances.

1.8 Apprentissage automatique

Avant d'aborder l'apprentissage automatique (artificiel) ou machine learning en anglais, nous rappelons l'apprentissage naturel.

1.8.1 Apprentissage naturel

L'apprentissage est l'acquisition de savoir-faire, c'est-à-dire le processus d'acquisition de pratiques, de connaissances, de compétences, d'attitudes ou de valeurs culturelles, par l'observation, l'imitation, l'essai, la répétition, la présentation.

D'une manière simpliste, l'**apprentissage par un individu** consiste à transformer des informations en connaissances.

En rapport avant l'environnement, l'apprentissage consiste à acquérir ou à modifier une représentation d'un environnement de façon à permettre avec celui-ci des interactions efficaces ou de plus en plus efficaces.

Quelques **méthodes d'apprentissage** connues et souvent utilisées :

— apprentissage par cœur et conditionnement ;
— apprentissage par renforcement ou punition-récompense ;
— apprentissage de concepts ;
— résolution de problèmes ;
— généralisation et spécialisation.

1.8.2 Apprentissage automatique

L'apprentissage automatique (ou apprentissage statistique) [8] est une branche de l'intelligence artificielle qui se base sur des approches mathématiques et statistiques pour la création de modèles permettent aux ordinateurs d'effectuer des tâches spécifiques sans être explicitement programmés pour cela.

Plus largement, 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 d'accomplir des tâches qu'il est difficile ou impossible d'accomplir par des moyens algorithmiques plus classiques.

Cela signifie qu'au lieu de suivre des instructions détaillées étape par étape, les ordinateurs apprennent à partir de données et d'expériences passées. Cela leur permet de faire des prédictions ou de prendre des décisions sur la base de nouvelles données.

On dit qu'un programme possède des *capacités d'apprentissage* si ses potentialités de comportement sur les données *se modifient* en fonction de ses performances au fur et à mesure qu'il traite les données. En outre, selon [9] l'apprentissage dénote des *changements* dans un système qui lui permettent de faire la même tâche *plus efficacement* la prochaine fois.

1.8.2.1 Phases d'apprentissage

L'apprentissage automatique comporte principalement deux phases :

— La première, phase « d'apprentissage » ou « d'entraînement », consiste à estimer un modèle à partir de données, appelées observations. L'estimation du modèle consiste

à résoudre une tâche pratique, telle que traduire un discours, estimer une densité de probabilité, reconnaître la présence d'un chat dans une photographie.
— La seconde phase est la **mise en production** : le modèle étant déterminé, de nouvelles données peuvent alors être soumises afin d'obtenir le résultat correspondant à la tâche souhaitée.

Globalement, l'apprentissage automatique procède comme suit :

1. observations d'un phénomène complexe (physique, biologique ou autre) ;
2. construction d'un modèle de ce phénomène ;
3. prévisions et analyse du phénomène grâce au modèle.

le tout automatiquement (sans intervention humaine).

1.8.2.2 Exemples de problèmes d'apprentissage

— identifier des caractères manuscrits à partir d'une image digitalisée,
— prévoir le prix d'un stock dans 6 mois à partir de mesures de performance de l'entreprise et de données économiques,
— prévoir un taux de pollution atmosphérique en fonction de conditions météorologiques,
— prévoir une courbe de consommation électrique pour un client SONALGAZ en fonction de variables climatiques et de caractéristiques spécifiques à ce client,
— identifier les facteurs de risque d'un certain type de cancer, en fonction de variables cliniques et démographiques, ...

1.8.3 Types d'apprentissage

Selon le type de données utilisées pour l'apprentissage, les algorithmes peuvent être classés en quatre catégories [10] :

Apprentissage supervisé : l'algorithme est entraîné sur un ensemble de données étiquetées (Chaque exemple d'entraînement est composé d'un ensemble d'entrées et de sorties correctement identifiées et classifiées). L'objectif de l'algorithme est d'apprendre à trouver la relation entre les entrées et les sorties des exemples fournis et de prédire les sorties pour de nouvelles données. Cet apprentissage peut être utilisé pour la reconnaissance et la classification d'images, la conversion de la langue parlée en texte, le diagnostic médical, ...

Apprentissage non supervisé : l'algorithme est entraîné sur des données non-étiquetées. Il cherche à découvrir des structures cachées dans les données (par exemple des densités de probabilités, des motifs permettent la classification ou le classement des données). Ces algorithmes sont appliqués pour la recommandation de produits, l'analyse de communautés dans les réseaux sociaux, la détection d'anomalies, ...

Apprentissage semi-supervisé : l'algorithme utilise une grande quantité de données non étiquetées pour améliorer la performance du modèle tout en utilisant une moindre quantité de données étiquetées pour guider son apprentissage. Cette approche est utile lorsque l'étiquetage des données est coûteux ou prend du temps. Habituellement utilisé pour le traitement du langage naturel, la reconnaissance vocale, la détection de fraudes, ...

Apprentissage par renforcement : dans cet apprentissage, au lieu d'établir les variables d'entrée avec les paramètres de sortie, l'algorithme apprend à prendre des décisions en interagissant avec un environnement dynamique. Selon les actions qu'il réalise, il reçoit des récompenses ou des pénalités, et l'objectif est de maximiser la récompense cumulée au fil du temps. Plusieurs applications sont possibles pour ce type d'apprentissage : robotique, jeux, contrôle du trafic, ...

1.8.4 Acquisition des connaissances vs Apprentissage automatique

Souvent rapprochés de l'apprentissage automatique, les travaux sur l'acquisition des connaissances ont cependant des objectifs spécifiques et un souci d'opérationnalité propre. L'objectif de l'acquisition des connaissances et la définition et la mise en place de moyens méthodologiques et techniques pour fournir au système les connaissances de l'expert. Alors que l'objectif de l'apprentissage automatique est de doter le système des capacités d'acquérir de nouvelles connaissances par lui-même. Ces deux approches peuvent être complémentaires tels que :

— les techniques d'apprentissage pouvant être un des moyens de fournir des connaissances au système, et
— les résultats de l'acquisition pouvant servir d'amorce à l'utilisation de l'apprentissage.

Néanmoins, dans certaines situations, cette distinction n'est plus si le système est doter de capacités à apprendre auprès de l'expert.

1.9 Exercices corrigés

1.9.1 Énoncés

Exercice 1.1

Donner des exemples de données, à transformer en informations, ensuite en connaissances.

Exercice 1.2

1. Quelle la différence entre le savoir et le savoir-faire ? A quel type de connaissances appartiennent-ils ?
2. Quelles sont les différentes sources de la connaissances ?

1.9.2 Solutions

Solution de l'exercice 1

Exemple	Donnée	Information	Connaissance
1	250 g	250 g le poids d'une baguette de pain	Je suis dans une boulangerie, j'ai acheté une baguette son poids est inférieur à 250 g
2	Amar	Amar porte un couteau	Amar porte un couteau, il est devant la porte de votre maison
3	Panneau de signalisation	Panneau de signalisation d'interdiction de stationnement	Vous êtes stationné à côté d'un panneau de signalisation d'interdiction de stationnement.

Solution de l'exercice 2

1. Savoir et savoir-faire
   - Le savoir constitue la connaissance théorique. Il renvoie à l'ensemble des connaissances acquises au cours des études et des formations professionnelles, etc. Le savoir est une connaissance **déclarative**.
   - Le savoir-faire constitue la connaissance pratique. Il correspond aux compétences techniques. Le savoir-faire est une connaissance **procédurale**.
2. La relation entre l'acquisition de connaissance et l'apprentissage automatique : c'est une relation de complémentarité.
   - Les modèles d'apprentissage automatiques utilisent les connaissances acquises afin de prendre des décisions et améliorer leurs apprentissage.
   - L'acquisition de connaissance peut également faire appel aux modèles de l'apprentissage pour amélioration le processus de collecter des connaissances.

1.10 Conclusion

Ce chapitre a présenté les fondements de l'ingénierie des connaissances qui est une branche de l'IA. Le développement de systèmes de l'IC passe par quatre étapes principales qui sont :

- le recueillement des besoins qui fait appel à l'acquisition des connaissances dans diverses sources, en particulier des experts ;
- la modélisation conceptuelle qui permet de modéliser et de représenter la connaissance ;
- l'implémentation pour avoir une base de connaissances utilisée lors du raisonnement.
- et l'évaluation par les utilisateurs finaux.

De plus, afin que le système soit qualifié d'intelligent, il doit intégrer l'apprentissage automatique pour assurer son évolution.

Le prochain chapitre abordera la deuxième étape du développement d'un système de l'IC en l'occurrence la représentation de connaissances.

Chapitre 2

Représentation de Connaissances

2.1 Introduction

Les problèmes liés à la connaissance sont : identifier, acquérir, **représenter**, manipuler la connaissance. Dans ce chapitre, et les deux suivants, nous nous intéressons au problème de la représentation.

Définition de la représentation des connaissances : un système définissant une série de symboles et une série d'opérations sur ces symboles en considérant une modélisation du raisonnement que cette liste de symboles supporte.

Problème de la représentation des connaissances : il revient à trouver une correspondance entre un monde extérieur et un système symbolique.

2.2 Modes de représentation des connaissances

Nous allons répartir les modes de représentation en trois catégories :

— Les formalismes de représentation des connaissances qui sont les réseaux sémantiques, les frames et les scripts, les graphes conceptuels, les logiques classiques et non classiques. La Figure 2.1 fait une distinction entre les formalismes logiques et non logiques. Ce chapitre introduit les formalismes non logique et le chapitre 3 aborde les formalismes logiques particulièrement la famille des logiques de description.

— Les langages de représentation de connaissances sont utilisés pour représenter la structure et le sens de la connaissance et permettent l'échange entre systèmes hétérogènes à l'image de KIF, XML et RDF. Le chapitre 4 abordera ces langages.

— Les systèmes de représentation des connaissances sont des cadres basés sur les formalismes et les langages précédents permettant une représentation de connaissances à l'aides de concepts prédéfinis. Les systèmes KL-One, CLASSIC et LOOM seront introduits également dans le chapitre 4.

2.3 Formalismes de représentation des connaissances

Nous allons présenter entre autres les réseaux sémantiques, les frames (schémas), les scripts et les graphes conceptuels.

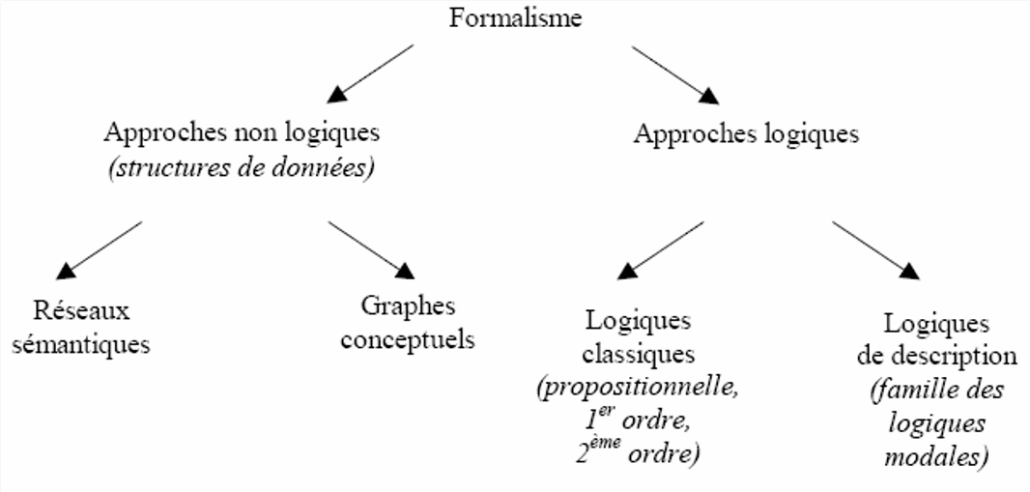


FIGURE 2.1 – Formalismes logiques et non logiques.

2.3.1 Réseaux sémantiques

Le formalisme de représentation de connaissances des réseaux sémantiques remonte aux travaux du linguiste Quillian (en 1968) sur la mémoire sémantique. Les réseaux sémantiques sont très utilisés dans les travaux sur le traitement et la compréhension des langages naturels.

Un graphe est une structure mathématique où nœuds et arcs n'ont pas de signification particulière, alors que dans un réseau sémantique, ils ont une signification spécifique d'où l'utilisation du mot sémantique.

2.3.1.1 Qu'est ce qu'un un réseau sémantique ?

- Un outil qui simule notre représentation de la mémoire.
- Un modèle qui montre comment l'information pourrait être représentée en mémoire et accédée en mémoire.
- Il désigne un ensemble de mots représentant des objets ou concepts, reliés entre eux en fonction de critères sémantiques particuliers.

Définition : Un réseau sémantique est un graphe orienté formé de nœuds -représentant les concepts– reliés par des arcs. Les nœuds comme les arcs sont étiquetés.

Une liaison entre deux nœuds étiquetés par A et B ont la propriété d'être en relation par R : R(A,B)

Il existe des relations prédéfinies telles que : est-un, sorte-de, a-un, etc.

- est-un : est un exemple de, est une instance, est un élément de,...
- sorte-de : est une sous-classe de, est un sous-ensemble de, ...
- a-un : attribut.

Nous pouvons représenter un réseau sémantique de deux manières :

Représentation textuelle : qui n'est pas très utilisée.

Exemple 2.1: «Un chat est une espèce de la classe des animaux»

On crée un réseau sémantique formé des mots *chat* et *animal* qui seront liés par une relation nommée "sorte-de". Le réseau pourrait avoir cette forme :

('chat' –sorte-de –> 'animal')

Représentation graphique : qui facilite la lecture.

Exemple 2.2: «Amar mange une orange»

La Figure 2.2 donne le réseau sémantique représentant l'énoncé. La représentation textuelle (non-graphique) peut être comme suit : (Amar, mange, orange)

FIGURE 2.2 – Réseau sémantique : "Amar mange une orange"

2.3.1.2 Éléments du réseau sémantique

Les nœuds d'un réseau sémantique peuvent être :
— Atomiques : les entités élémentaires tels que des valeurs, des individus, ...
— Complexes : les entités complexes comme des propositions, des phrases, ...
Les liens également sont de deux types :
— Structurels : lorsqu'ils sont indépendants de la sémantique du domaine représenté.
— Spécifiques : quand ils sont dépendants de la sémantique du domaine représenté.

Les relations prédéfinies sont notamment "sorte-de" et "est-un".
— sorte-de : c'est un lien entre un concept général et un concept plus général représentant l'inclusion. C'est un mécanisme de classification et de catégorisation.

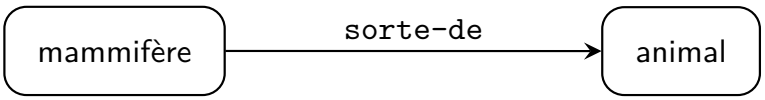


FIGURE 2.3 – Un mammifère est un animal.

— est-un (instance-de) : est un lien entre un concept individuel et un concept général représentant l'appartenance. C'est un mécanisme d'identification ou de reconnaissance.

Exemple 2.3:

La Figure 2.5 présente un exemple des liens sorte-de et instance-de qui permettent de montrer les relation entre certaines classes d'animaux et des instances de ces derniers.

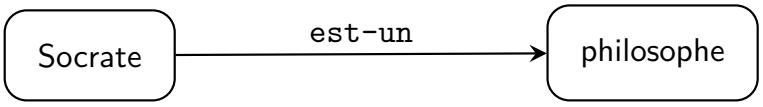


FIGURE 2.4 – Socrate est un philosophe.

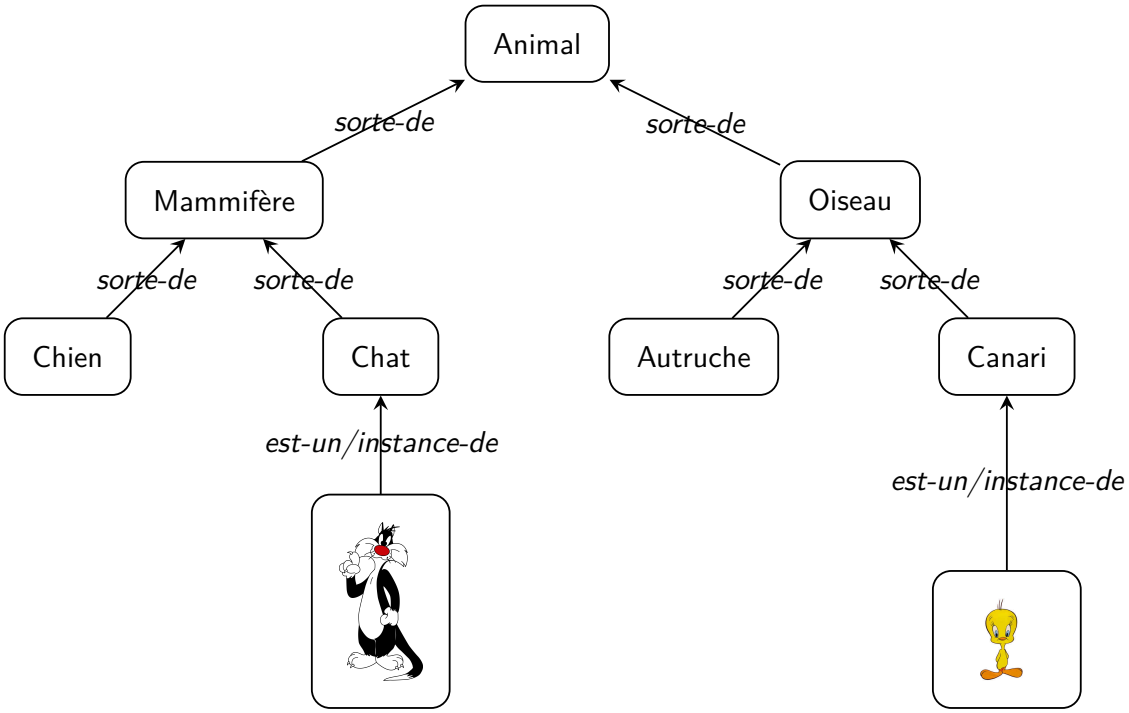


FIGURE 2.5 – Exemples de sorte-de et est-un.

Les propriétés sont des informations qui peuvent être attachées a chaque nœud du réseau sémantique.

Exemple 2.4:

La Figure montre 2.6 un exemple de représentation de propriétés dans un réseau sémantique.

La représentation d'événements ou d'actions peut se faire en utilisant les liens structurels.

Exemple 2.5: Représenter l'événement : «Gaston casse le vase».

La Figure 2.7 illustre le réseau sémantique représentant l'énoncé.

Le lien «casser» et spécifique. On peut le traduire par des liens plus structurels : agent, objet, instrument, temps, lieu, ..., comme le montre l'exemple de la Figure 2.8.

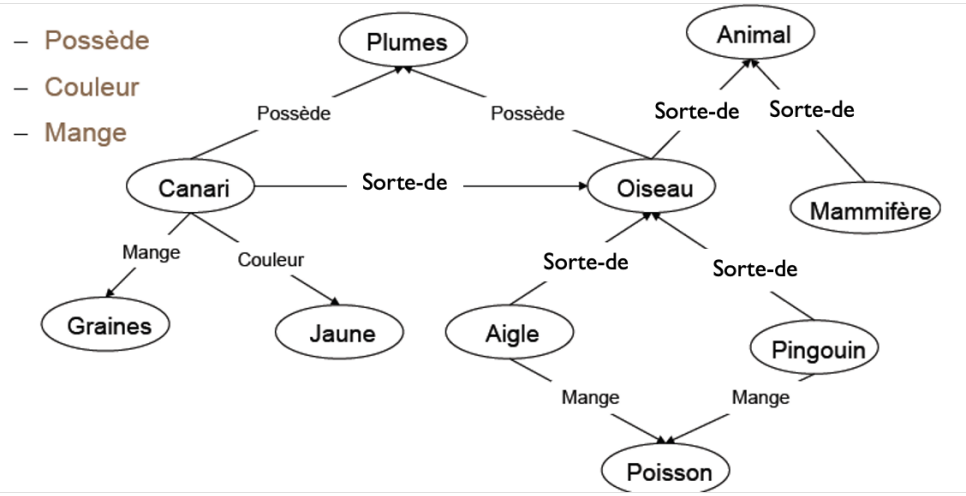


FIGURE 2.6 – Représentation des propriétés :posséder, couleur et manger.

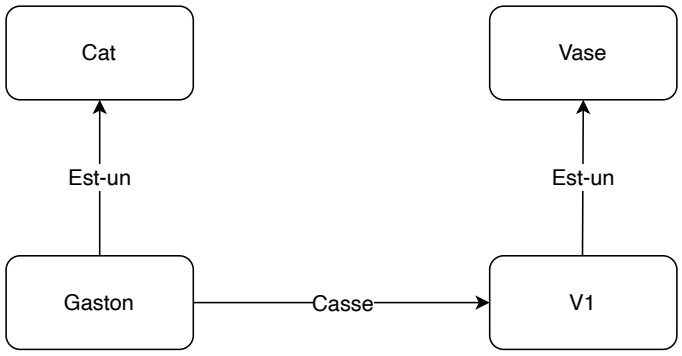


FIGURE 2.7 – «Gaston casse le vase».

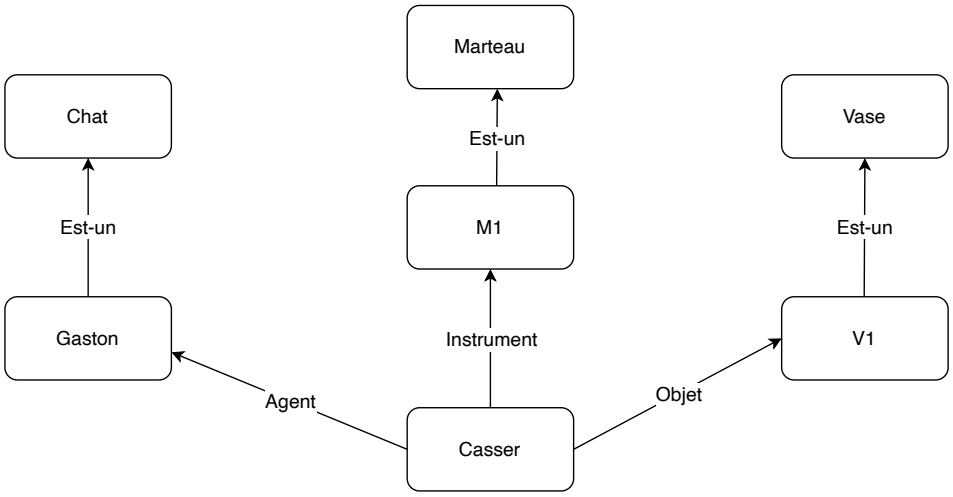


FIGURE 2.8 – Réseau sémantique avec liens structurels.

Exemple 2.6:

La Figure 2.9 montre un exemple de représentation d'un réseau sémantique en utilisant les liens structurels.

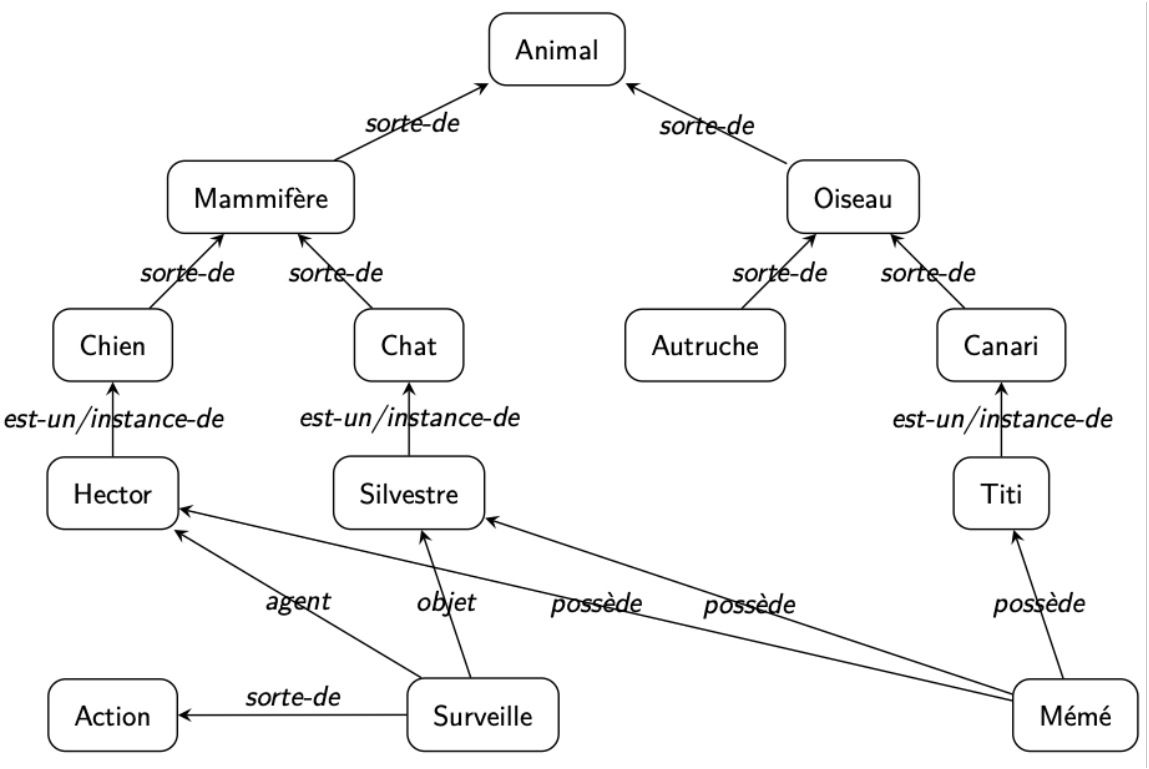


FIGURE 2.9 – Exemple de liens structurels.

2.3.1.3 Avantages et Inconvénients

Avantages : les réseaux sémantiques permettent le regroupement physique autour d'un concept de tous les éléments qui lui sont associés. Ils offrent une certaine lisibilité à l'aide de leur aspect visuel.

Inconvénients : l'inconvénient majeur des réseaux sémantiques est la lourdeur de représentation pour des grandes base de connaissances. Un autre inconvénient est l'impossibilité d'exprimer des connaissances de type procédural (c'est-à-dire programmes). Ce formalisme manque également de rigueur ce qui conduit à de nombreuses questions sur la signification qu'on peut lui attacher, l'unicité de la représentation, la représentation d'idée ou de croyances, etc.,..

Par la suite, des améliorations ont été apportées pour donner naissances aux réseaux à propagation de marqueurs, réseaux sémantiques partitionnés et graphes conceptuels.

2.3.2 Frames (schémas)

Le concept «Frame» (synomyme "schéma") a été proposé par Minsky en 1975 dans l'objectif d'offrir un support permettant de regrouper l'ensemble d'informations disponibles sur un objet (au sens large : concept, événement ou ...). En ce sens, un frame est une unité de connaissance (prototype) décrivant une situation ou un objet, i.e. une représentation des connaissances mixtes à la fois déclaratives et procédurales.

2.3.2.1 Définitions

La définition donnée par Marvin Minsky [11] est « Un frame est une structure de données représentant une situation stéréotypée [...]. De nombreuses sortes d'informations sont associées à chaque frame. Certaines d'entre elles concernent l'utilisation de ce frame. D'autres portent sur ce que l'on s'attend à ce qu'il arrive par la suite. D'autres encore portent sur ce qu'il faut faire si ces attentes ne sont pas confirmées ».

Dés lors, nous comprenons qu'un frame permet la description des objets habituellement rencontrés, des situations prototypiques suffisamment bien connus par des individus pour qu'ils aient des attentes précises, de leurs caractéristiques et le déroulement des événements de ces situations.

Exemple 2.7: Un étudiant à l'université.

Une telle réalité peut être représentée en utilisant les frames donnés par la Figure 2.10.

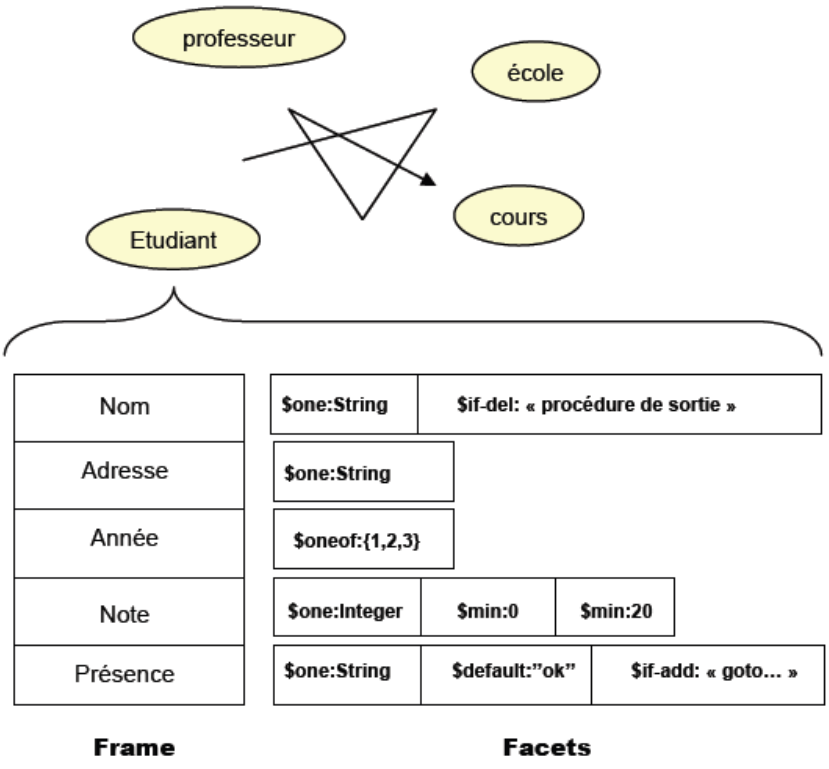


FIGURE 2.10 – Un étudiant à l'université.

2.3.2.2 Caractéristiques des Schémas

Les caractéristiques des frames sont :
— Concept : qui est une structure composée d'attributs.
— Attributs : qui contiennent des facettes (domaines) qui sont explicitement répertoriées.
— Facettes : chaque facette spécifie l'ensemble des valeurs possibles pour cet attribut (instance). Une des facettes va spécifier une valeur par défaut si cela est nécessaire.

La Figure 2.10 donne les caractéristiques de l'exemple précédent à l'aide d'un frame avec facettes.

Exemple 2.8: Exemples de frames

Frame : Chaise
— Sorte de : Meuble
— Nombre_de_pieds : doit être entier par default 4
— Style_du_dossier : doit être Droit, Rembourré
— Nombre_de_Bras : doit être 0,1 ou 2

Frame : Chaise_de_Paul
— Sorte de : Chaise
— Nombre_de_pieds : 4
— Style_du_dossier : Rembourré
— Nombre_de_Bras : 0

Remarque :

A noter que sorte-de désigne également la relation d'appartenance (est-un).

2.3.2.3 Avantages et inconvénients

Avantages : les systèmes de schémas constituent un moyen très naturel de représenter les connaissances, surtout lorsqu'elles concernent des objets aux structures complexes.
— Tout est placé dans le frame ;
— Les facettes définissent la sémantique de l'attribut ;
— Mise en correspondance : la valeur de l'attribut d'un frame peut être un frame ou un ensemble de frames.

Inconvénients : parmi les inconvénients, nous pouvons citer :
— Pas d'équivalence logique ;
— Insatisfaisant d'avoir les sémantiques des frames et de la hiérarchie des frames spécifiées opérationnellement.

2.3.3 Scripts

Le concept de Script (scénario) a été introduit par Schank et Adelson en 1977 [12] sur les modèles de frames pour décrire des scènes de la vie courante.

C'est une forme de connaissance déclarative qui peut servir à :
— Communiquer, comprendre et raisonner sur une histoire ;

— Comprendre ou raisonner sur une situation ou une séquence d'événements ;
— Raisonner sur les actions ;
— Planifier les actions.

2.3.3.1 Définitions

Un script est une représentation structurée d'une suite d'événements, dans un contexte particulier. Il est représenté par des schémas utilisés pour décrire cette suite d'événements.

Un script est composée d'une scène, des "props", des acteurs, des événements, et des contextes (situations) composé des événements. Nous allons définir ces éléments en nous appuyant sur l'exemple classique de script du restaurant, publié par Shank en 1977. Le script du restaurant, même s'il concerne l'assimilation de nourriture, est différent de celui du Fast-Food ou de celui du repas familial.

Les éléments d'un script

— Une scène (un lieu et ses composants) : l'ensemble des suites d'événements concernant les objets et les rôles. Le script est divisé en une suite ordonnée de scènes.
Exemples : l'entrée au restaurant, la commande, la consommation, la sortie du restaurant
— Des "props" (Les objets manipulés dans le script) : l'ensemble des entités pour lesquelles sont affirmés des faits.
Exemples : tables, chaises, argent
— Les acteurs (Les agents) : intervenant dans script.
Exemples garçon, client, chef, ...
— Les événements : un changement de situation. L'ensemble des actions que chaque entité doit effectuer (ou subir) dans le cadre de sa participation au script.
Exemples : le garçon sert à table et présente le menu, le client mange et paye ;
— Les Situations. Une configuration des entités et relations.

Remarque :

Tout programme, basé sur un script, devant effectuer l'interprétation d'un texte en langue naturelle devrait :
— vérifier si le texte satisfait les conditions d'entrée du script ;
— lier les objets et les personnes du texte aux variables du script ;
— utiliser les rôles des individus à l'intérieur des scènes afin de combler les informations manquantes ;
— produire une représentation structurée du texte, la plus complète possible.

Exemple 2.9: Script : Manger dans un bon restaurant

- Scène : La (les) salle(s), l'entrée, les tables (comme lieu), la cuisine
- Acteurs : Le maître d'Hôtel, Le serveur, les clients
- Props : le menu, la table (comme objet), les chaises, les couteaux, fourchettes et cuillers, la (les) verre(s)
- Situations : Entrer, S'asseoir, lire le menu, commander, manger, boire, demander l'addition, payer, partir.
- contextes ;

Exemple 2.10: Script : Achat d'une boisson au distributeur automatique

- Scène : devant la machine.
- Props : La machine, les pièces de monnaie, la boisson, le gobelet.
- Acteur : acheteur.
- Actions :
  - Sortir tes pièces de monnaie.
  - Payer.
  - Sélectionner la boisson et les options (sucre, crème, etc.)
  - Récupérer la boisson et le monnaie.

Nous pouvons avoir des scripts qui sont composés de plusieurs scripts comme le montre l'exemple suivant.

Exemple 2.11: SCRIPT : " manger-au-restaurant"

- ELEMENTS : (restaurant, agent, nourriture, menu, tables, chaises)
- ROLES : (clients, serveur, chefs)
- POINT- DE- VUE : clients
- MOMENT : (heure d'ouverture du restaurant)
- LIEU : (emplacement du restaurant)
- Scénario :
  - D'abord : script « Entrer restaurant »
  - Puis : script « Attirer -l'attention -du –client- de- restaurant »
  - Puis : script « Prendre- place- à- table »
  - Puis : script « Passer- commande »
  - Puis : script « Manger » sauf si (longue attente) alors script « Sortie-en- colère »
  - Puis : si qualité nourriture > convenable) alors script « Féliciter- le chef»
  - Puis : script « Payer- l'addition »
  - Enfin : script « Quitter-restaurant ».

2.3.3.2 Avantages et inconvénients

Les scripts sont surtout utiles pour aider à compléter l'interprétation sémantique d'une histoire. Cependant, un script peu ou trop détaillé ne favorisera pas la production d'une in-

terprétation concise et complète. Des précautions devront être prises pour identifier correctement le script du contexte courant.

2.3.4 Graphes conceptuels

Ce mode de représentation des connaissances a été défini par Sowa en 1984 [13] pour traduire la logique sous forme graphique.

2.3.4.1 Définitions

Un graphe conceptuel est un graphe biparti connexe orienté où :
— Des deux types de nœuds sont utilisés pour représenter les concepts et les relations ;
— Un arc relie toujours un concept à une relation ;
— Un concept est identifié par son type ;
— L'identificateur de ce type est inscrit dans le concept.

Remarque :

Un type peut être concret ou abstrait :
— Un concept est dit concret si on peut s'en former une image mentale (un chien, une maison, une université, etc.).
— Dans le cas contraire, un concept est dit abstrait (la beauté, la réussite, la peur, etc.).

Tous les types sont structurés par une hiérarchie.

2.3.4.2 Représentation

Ce formalisme peut être noté de trois manières différentes, sous forme graphique, sous forme linéaire et sous forme formelle pour échanges entre agents logiciels.

Forme graphique : deux types de nœuds sont utilisés :
— un nœud conceptuel représenté par un rectangle modélisant une entité ;
— un nœud relationnel représenté par une ellipse (ovale) modélisant une propriété ou une relation ;
Les flèches relient toujours deux nœuds de types différents.

Exemple 2.12: Jean lit un livre.

Dans cet exemple, on a 3 concepts : ÉTUDIANT, LIVRE et le nom « Jean » et 2 relations : LIRE et NOM. La représentation graphique est donnée par la Figure 2.11.

Forme linéaire : les concepts sont représentés entre [ ] et les relations entre ( ). La représentation linéaire de l'exemple précédent est la suivante :
[« Jean »] <- (NOM) <- [ÉTUDIANT] -> (LIRE) -> [LIVRE].

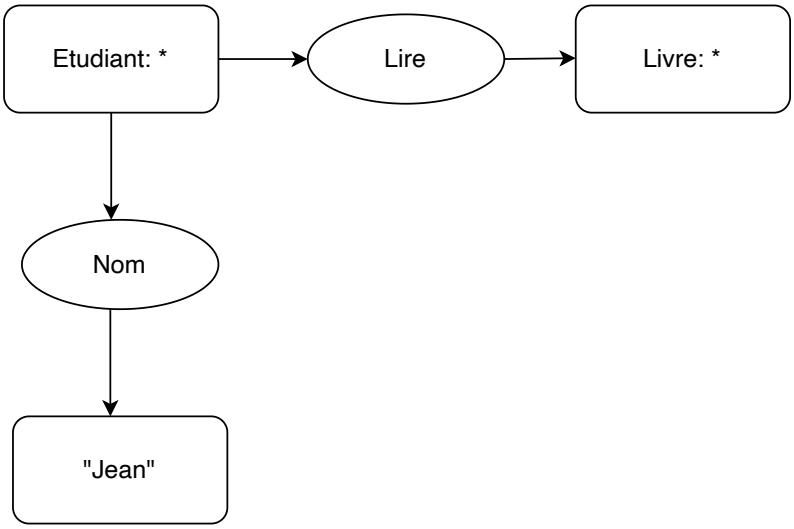


FIGURE 2.11 – Exemple de graphe conceptuel.

Conceptual graph interchange form(CGIF) : une troisième forme est proposée pour les échanges formels entre agents logiciels, c'est la conceptual graph interchange form(CGIF) détaillé dans [14].

2.4 Exercices corrigés

2.4.1 Énoncés

Exercice 2.1

Soit l'énoncé suivant : Fatima et Karim sont des personnes. Fatima possède une voiture, Karim possède aussi une voiture. Une voiture particulière est appelée Titine.

1. Représenter à l'aide d'un réseau sémantique cet énoncé.
2. Est-ce que Karim et Fatima possèdent la même voiture ?
3. A qui appartient la voiture Titine ?
4. Maintenant, nous savons que Karim est un homme et Fatima est une femme. Karima possède Titine et Fatima possède la voiture Anastasie. Compléter le premier réseau sémantique par ces nouvelles informations.

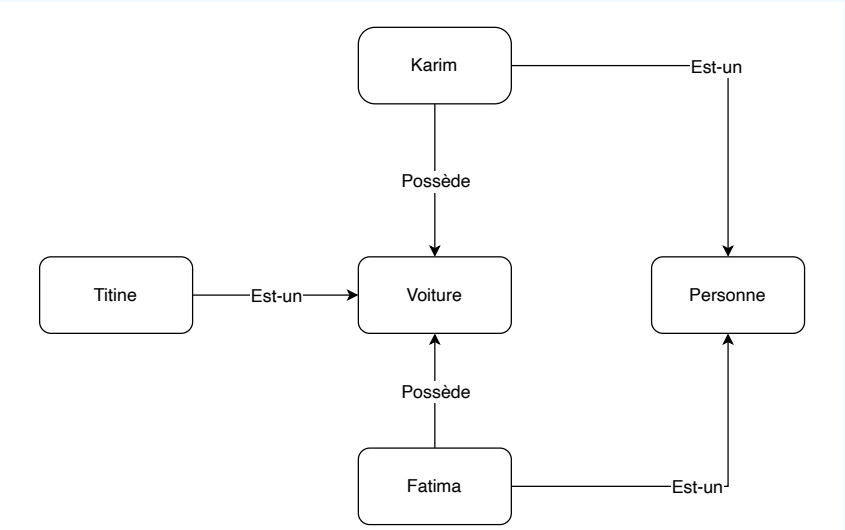
Exercice 2.2

Représenter la phrase "Nicolas décore sa maison située dans une ville entre Béziers et Montpellier" en utilisant les graphes conceptuels.

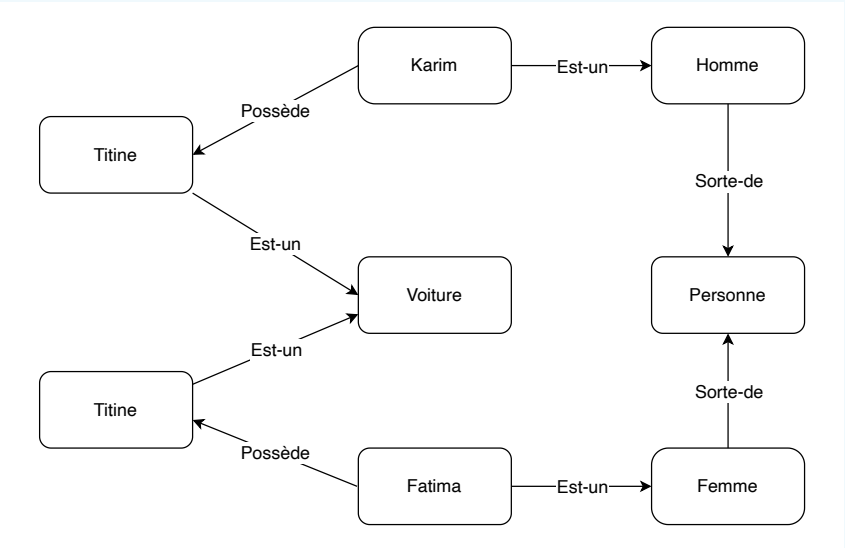
2.4.2 Solutions

Solution de l'exercice 1

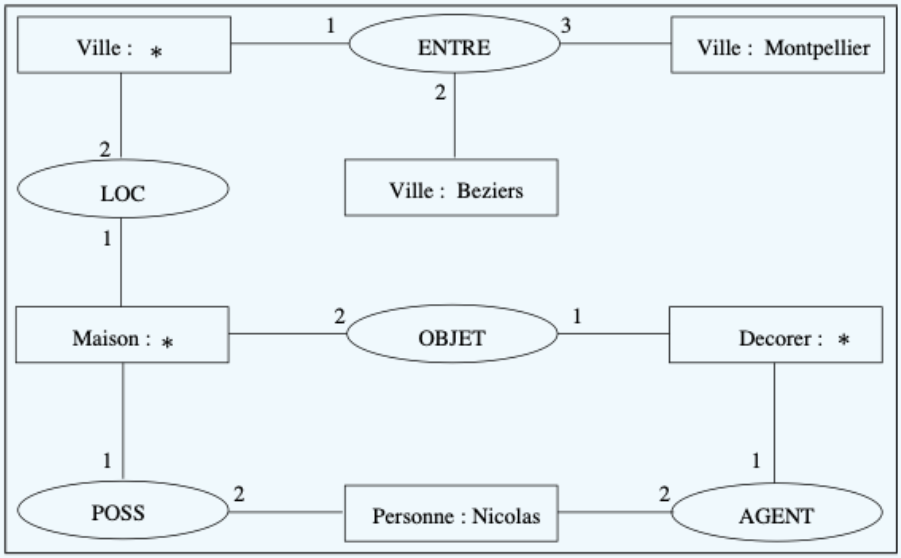
1. Réseau sémantique



2. La représentation construite ne permet pas de savoir s'ils possèdent la même voiture ou s'il s'agit de deux voitures distinctes
3. On n'a aucune information sur le rôle éventuel de Titine dans cette situation.
4. Le nouveau réseau sémantique



Solution de l'exercice 2



2.5 Conclusion

Les formalismes graphiques de représentation de connaissances sont adaptés pour représenter les concepts ainsi que les relations sous-jacentes entre eux. Ils sont adéquats et très expressifs lorsque le nombre de concepts à manipuler est relativement faible. Cependant, dans le cas de quantité de connaissances volumineuse cela devient très compliqué.

De plus, ces formalismes manquent de rigueur comparativement aux approches logiques, qui feront l'objet du chapitre suivant, même si des passages ont introduits vers ces dernières pour y remédier.

Chapitre 3

Les formalises logiques de représentation de connaissances

3.1 Introduction

La logique a été l'un des premiers formalismes proposés pour représenter de la connaissance et constitue toujours la base de nombreuses recherches en intelligence artificielle. La logique est un moyen de description des objets et de leurs propriétés et de faire des raisonnements. En général, on décrit une logique par les éléments suivants :

- un syntaxe qui répond aux questions qu'est-ce qu'une formule ? comment s'écrit-elle ?
- une sémantique qui répond à la question quel est le sens donné à chaque formule ?
- et le système de déduction qui est une méthode de preuve pour déterminer si une formule est vraie.

Dans cette chapitre, nous allons présenter les logiques classiques, en particulier la logique des propositions et la logique des prédicats, ensuite les logiques non classiques en traitant plus spécialement la famille des logiques de description.

3.2 Logiques classiques

Les logiques classiques les plus répondues sont la logique des propositions (LP) et la logique des prédicats du premier ordre.

3.2.1 Logique des propositions

Le langage logique le plus basique est la logique des propositions définie par :

Un alphabet formé de :

- Symboles de propositions : P, Q, ... (phrases)
- Phrases spéciales (valeurs) : Vrai, Faux
- Opérateurs : $\wedge$ (et), $\vee$ (ou), $\neg$ (non), $\Longrightarrow$ (implique), $\Leftrightarrow$ (équivalent)

Des règles de construction de phrases pour former des énoncés plus complexes en utilisant les connecteurs logiques.

Un calcul de valeurs de vérité de ces phrases par tables de vérités, par inférence, ...

Les phrases sont appelées des formules en logique.

Exemple 3.1: Nous voulons représenter les deux phrases suivantes en utilisant la logique des propositions :

— S'il fait beau et que l'on est pas vendredi alors je fais la cueillette des olives.
— Si je fais la cueillette des olives alors c'est l'hiver

Ce problème peut être modélisé sous la forme suivante :

— variable propositionnelle b = il fait beau
— variable propositionnelle s = on est vendredi
— variable propositionnelle f = je fais la cueillette des olives
— variable propositionnelle p = c'est l'hiver

La 1re phrase peut être donnée par : $((b \wedge (\neg s)) \implies f)$
La 2ème phrase peut être donnée par : $(f \implies p)$

L'un des inconvénients de la logique propositionnelle est elle a un pouvoir expressif limité (contrairement au langage naturel par exemple). Elle ne permet pas de représenter les énoncés tels que les suivants :

— Tous les étudiants sont intelligents.
— Tous les étudiants sont soit satisfaits ou pas par les cours de ce semestre.
— Chaque étudiant travaille avec un autre.

Donc, la plupart de nos connaissances portent sur des individus, sur des classes d'individus, sur des relations entre individus. Dans ces cas, nous pouvons utiliser la logique des prédicats du premier ordre.

3.2.2 Logique des prédicats du premier ordre

La nécessité de disposer de variables dans un langage de représentation des connaissances est liée au fait qu'une connaissance doit pouvoir s'appliquer à plusieurs objets, et parce qu'en résolution de problèmes on manipule couramment des inconnues. La logique des prédicats du premier ordre est plus riche que LP et son alphabet inclut des symboles de fonctions. Les éléments de sa syntaxe sont :

— un ensemble de variables x, y, ...,
— un ensemble de constantes a, b, , ...,
— les éléments V (vrai) et F (faux),
— un ensemble de fonctions f, g, ..., ayant chacun une arité,
— un ensemble de prédicats P, Q, ..., ayant chacun une arité,
— des connecteurs : la négation $\neg$, et des connecteurs binaires : $\wedge$, $\vee$, ...
— les quantificateurs $\forall$ et $\exists$.

Exemple 3.2: Une formule de logique du premier ordre

Considérons la formule : $\neg((\forall x)(\exists y)P(x, f(y), a))$
C'est une formule du premier ordre, où :
- $(\forall x)$: représente la quantification universelle (”pour tout x”),
- $(\exists x)$: représente la quantification existentielle (” il existe un x tel que”)
- Si P est un prédicat d'arité n, $P(X_1, X_2, ..., X_n)$ est une formule atomique.

La logique des prédicats permet de modéliser le monde en termes d'objets, de relations entre objets, de propriétés des objets et de faits ou règles concernant ces relations et propriétés. Dans les formules logiques utilisées :
- les constantes, variables et fonctions représentent des objets de la partie du monde ou du système modélisé ;
- les symboles de prédicats représentent les relations entre objets ou, pour les prédicats unaires, le fait qu'un objet possède une propriété ;
- les formules fermées représentent des faits particuliers (formules sans quantificateurs) ou et règles générales (formules quantifiées).

Exemple 3.3: Exemple de modélisation de relation entre personnes

On veut modéliser les relations entre personnes (connaissances, amitie, ...), le fait qu'une personne a envoyé un message à une autre personne, et les règles générales sur ces relations.
On utilise pour cela le vocabulaire de la logique des prédicats :
- prédicats : personne (unaire), message (ternaire), connait (binaire),
- constantes : Fatma, Omar, Ali (représentent des personnes), m1, m2, m3, ... (les messages)

Soient les faits suivants :
- Fatma est une personne.
- le message m3 a été envoyé par Fatma à Ali.
- Ali connait Houda.

Ils peuvent être représentés par :
- f1 = personne(Fatma).
- f2 = message(m3, Fatma, Ali)
- f3 = connait(Ali, Houda).

La représentation des règles générales de ce domaine passe par l'écriture de formules quantifiées. Ici l'expéditeur et le destinataire d'un message sont forcément des personnes et elles doivent se connaitre, donc :
$f4 = \forall x \forall y \forall z(message(x, y, z) \implies personne(y) \wedge personne(z) \wedge connait(y, z))$.
et pour être dans le réseau, il faut connaitre au moins une personne :
$f5 = \forall x(personne(x) \implies \exists y connait(x, y))$.

L'analogie entre les expressions du langage logique et les langages de programmation a donné naissance au langage PROLOG (PROgrammation LOGique) en 1972 [15].

L'un des inconvénients des logiques dites classiques est qu'elles ne reconnaissent comme modalités que le Vrai et le Faux.

3.3 Logiques non classiques

Les logiques classiques ne considèrent que des raisonnement rigoureux, précis et reposant sur des connaissances certaines. Dans les raisonnements humains courants ce n'est pas toujours le cas, nous pouvons avoir :

— de l'incertitude (je pense, probable, plausible, ...),
— de la graduation imprécises (rapidement, proche, mauvais, ...),
— de l'aspect temporel (prévision, ...),
— des croyances des individus ou des agents, ...

Afin de prendre en compte les différents aspects du raisonnement humain, on a vu la création de diverses logiques qualifiées de non classiques. Les logiques non classiques peuvent être regrouper en deux types :

— Des extensions de la logique classique qui conservent les caractéristiques des logiques classiques et ajoutent de nouveaux opérateurs ou objets au niveau syntaxique (logiques modales), ou de nouvelles valeurs de vérité au niveau sémantique (logiques multivaluées).
— En rupture avec les logiques classiques et ne respectent pas certaines axiomes tels que la monotonie et le principe de tiers exclu (soit une formule est vraie, soit sa négation est vraie).

3.3.1 Logiques multivaluées

Les logiques multivaluées dont les valeurs de vérité peuvent être autres que Vrai ou Faux. Nous distinguons :

— Les **logiques trivaluées** ajoutent une troisième valeur de vérité à l'ensemble Vrai, Faux dont la signification peut être suivant les logiques : « inconnu », « absurde », ...
— La **logique floue** prend en compte les énoncés imprécis ou vagues. Elle exprime l'imprécision des formules par une graduation de la vérité.

Exemple 3.4: Représentation des notions imprécises Chaud et Froid et Tempéré

Entre 0 et 10 °C est considéré Froid, entre 10 et 20 °C est considéré Tempéré, et supérieur à 15 °C est considéré Chaud. Finalement ces notions se chevauchent comme le montre la Figure 3.1.

3.3.2 Logiques modales

Les logiques modales utilisent des modalités qui peuvent être [16] :

— possibilité et nécessité (logiques aléthiques)
— connaissance et croyance (logiques épistémiques)
— obligation et autorisation (logiques déontiques)
— temporelle telles que le passé ou le futur (logiques temporelles). Par exemple : un étudiant finira par avoir un diplôme.

Une modalité est un opérateur M, tel que si F est une formule logique MF est aussi une formule. Prenons la modalité Possibilité/nécessité :

$\Box$: il est nécessaire que ... et $\Diamond$: il est possible que ...

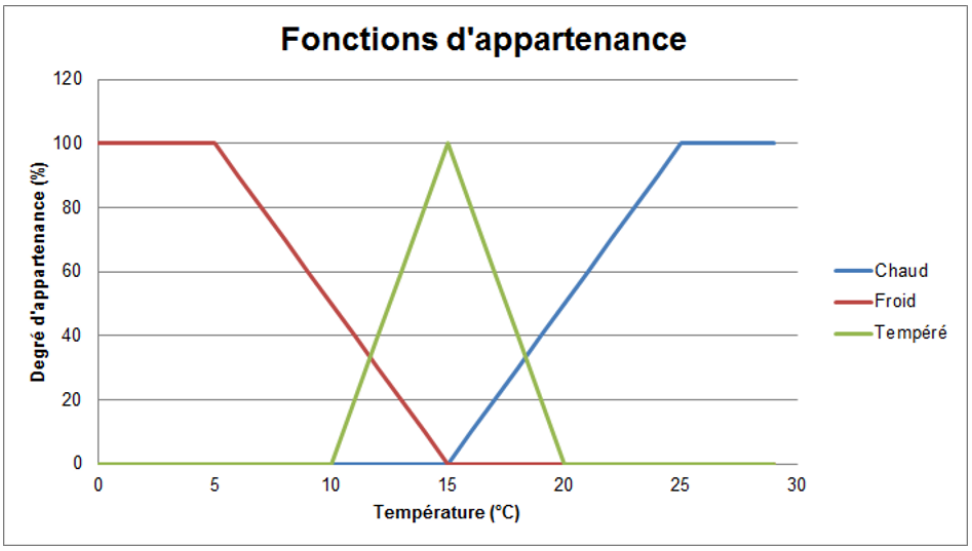


FIGURE 3.1 – Représentation de la température en logique floue.

Exemple : un étudiant peut avoir un diplôme.

3.3.3 Logiques non-monotones

Les logiques non-monotones sont nées à partir des années 1970 [17]. Elles permettent de faire des hypothèses puis, si nécessaire, de les réfuter. Par exemple les logiques des défauts (exceptions) utilisant des "règles de défaut" de la forme $\frac{\alpha\,\beta}{\gamma}$.

qui s'interprète comme suit : « Si l'on croit α et qu'il est possible de croire β, alors on (sup)pose γ »

Exemple : Tous les oiseaux volent, exception les autruches.

3.3.4 Raisonnement en présence de l'incertitude

Le raisonnement en présence de l'incertitude se fait à l'aide de trois modes de représentations :

- La logique probabiliste : associer à chaque formule une probabilité subjective en utilisant les propriétés des probabilités ;
- La logique des croyances (Dempster-Shafer) : associer à chaque sous-ensemble d'hypothèses une masse de croyance et de définir à partir de ces masses la crédibilité et la plausibilité des formules ;
- La logique possibiliste : exprime la certitude sur une formule par deux coefficients, la possibilité de cette formule et sa nécessité.

Nous présentons à présent d'une manière détaillée une logique non classique qui est la famille des logiques de description.

3.4 Logiques de description

Les logiques de description (LD) est une famille de langages de représentation de connaissances. Les LD sont influencées par la logique des prédicats, les schémas (frames), et les réseaux

sémantiques. En effet, la présence de catégories générales d'objets et de relations dans LD fait partie de l'héritage conceptuel des schémas et des réseaux sémantiques.

Les logiques de description sont caractérisées par les éléments suivants :

— **Concept** qui est une entité générique d'un domaine. Il permet de représenter un ensemble d'individus.
— **Individu** est une entité singulière, une instance d'un concept.
— **Rôle** qui représente une relation binaire entre individus.

Exemple 3.5: Cet exemple décrit les relations entre membres d'une famille :

— Concepts : Femme, Homme, Personne ...
— Rôles : enfant-de, père-de ...
— Instances : Pierre, Marie ....

En outre, le concept et le rôle possèdent une description structurée élaborée à partir d'un ensemble de *constructeurs*. Une sémantique est associée à chaque description de concept et de rôle par l'intermédiaire d'une *interprétation*.

3.4.1 Principes des LD

Les connaissances d'un domaine sont modélisées à l'aide de boites comme le montre la Figure 3.2.

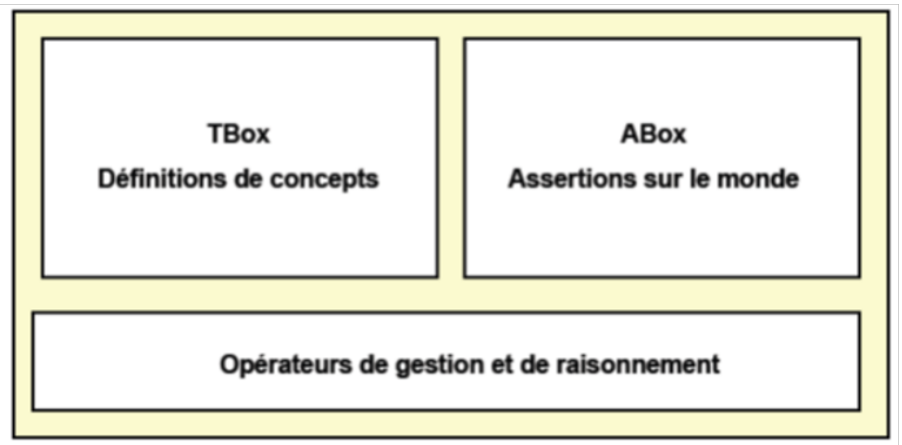


FIGURE 3.2 – Emboitement de la connaissance en logique de description.

TBox (niveau terminologique) pour la représentation des concepts et des rôles.

ABox (niveau des assertions) pour la description et la manipulation des individus qui relèvent du niveau factuel.

La Figure 3.3 montre un exemple de Tbox et Abox.

Opérateurs de gestion et de raisonnement permettant de manipuler les concept, les individus et les rôles ainsi que de raisonner sur eux.

— Organisation des concepts et rôles par la relation de *subsomption* par niveau de généralité : C subsume D si C est plus général que D au sens que l'ensemble d'individus

TBox	ABox
$Femelle \sqsubseteq \top \sqcap \neg Mâle$	$Humain(Anne)$
$Mâle \sqsubseteq \top \sqcap \neg Femelle$	$Femelle(Anne)$
$Animal \equiv Mâle \sqcup Femelle$	$Femme(Sophie)$
$Humain \sqsubseteq Animal$	$Humain(Robert)$
$Femme \equiv Humain \sqcap Femelle$	$\neg Femelle(Robert)$
$Homme \equiv Humain \sqcap \neg Femelle$	$Homme(David)$
$Mère \equiv Femme \sqcap \exists relationParentEnfant$	$relationParentEnfant(Sophie, Anne)$
$Père \equiv Homme \sqcap \exists relationParentEnfant$	$relationParentEnfant(Robert, David)$
$MèreSansFille \equiv Mère \sqcap$ $\forall relationParentEnfant.\neg Femme$	
$relationParentEnfant \sqsubseteq \top_R$	

FIGURE 3.3 – Exemple de boites TBox et ABox.

représenté par C contient l'ensemble d'individus représenté par D, ce qui représente une hiérarchie de concepts et (parfois) une hiérarchie de rôles.

— Opérations de base (classification et instanciation) :

La *classification* de concepts (ou rôles) détermine la position d'un concept (d'un rôle) dans une hiérarchie. Ainsi le processus de classification permet la construction et la maintenance de la hiérarchie.

L'*instanciation* permet de retrouver les concepts dont UN individu est susceptible d'être une instance (sens différent dans les langages à objet où l'instance ne peut être qu'un objet).

3.4.2 Niveau terminologique (TBox)

Les éléments composants la TBox sont les entités atomiques, les concepts et rôles atomiques prédéfinis, et les entités composées.

Entités atomiques : Les concepts atomiques et rôles atomiques constituent les entités élémentaires d'une TBox :

— Les noms débutant par une lettre majuscule désignent les concepts (par exemple dans le tableau 3.3 : Femelle, Mâle), (A, B des concepts atomiques).

— Les noms débutant par une lettre minuscule dénomment les rôles (par exemple dans le tableau 3.3 : relationParentEnfant), (R dénote un rôle).

Concepts et rôles atomiques prédéfinis : Les LD prédéfinissent minimalement quatre concepts atomiques :

- le concept $\top$ et le rôle $\top_R$, les plus généraux de leur catégorie respective.
- le concept $\bot$ ainsi que le rôle $\bot_R$ les plus spécifiques (l'ensemble vide).

Entités composées : Les concepts et rôles atomiques peuvent être combinés au moyen de constructeurs pour former respectivement des concepts et des rôles composés (C, D des concepts composés).

Exemple 3.6: Exemple de concept

Le concept composé $Mâle \sqcap Femelle$ résulte de l'application du constructeur $\sqcap$ aux concepts atomiques Mâle et Femelle. Il s'interprète comme l'ensemble des individus qui appartiennent aux concepts Mâle et Femelle.

Notion d'interprétation : A l'instar de la logique classique, une sémantique est associée aux descriptions des concepts et des rôles : les concepts sont interprétés comme des sous-ensembles d'un domaine d'interprétation Δ_I et les rôles comme des sous-ensembles du produit $\Delta_I \times \Delta_I$.

L'interprétation est donc utile pour expliciter formellement la sémantique d'une TBox. Elle est définit comme suit :

Une interprétation $I = (\Delta_I, .^I)$ est la donnée d'un ensemble Δ_I appelé domaine de l'interprétation et d'une fonction d'interprétation $.^I$ qui fait correspondre à un concept un sous-ensemble de Δ_I et à un rôle un sous-ensemble de $\Delta_I \times \Delta_I$.

Définition formelle de TBox : Une TBox contient des axiomes terminologiques de la forme $C \sqsubseteq D$ (relations d'inclusion) ou $C \equiv D$ (relations d'équivalence entre concepts).

- Une interprétation I satisfait un axiome $C \sqsubseteq D$ ssi $C^I \subseteq D^I$.
- Une interprétation I satisfait un axiome $C \equiv D$ ssi $C^I = D^I$.
- Une interprétation satisfait une TBox (est un modèle de TBox) ssi l'interprétation satisfait tous les axiomes de la TBox.

3.4.3 Niveau factuel (ABox)

Une ABox contient un ensemble d'assertions sur les individus : des assertions d'appartenance et des assertions de rôle.

Chaque ABox doit être associée à une TBox, car les assertions s'expriment en terme des concepts et des rôles de la TBox. Elle désigne les individus par des noms (individu nommé) représentés par les lettres a, b. Dans le tableau 3.3 : Anne, David, Robert et Sophie.

Une fonction d'interprétation assigne à chacun de ces noms a, un individu a^I tel que $a^I \in \Delta_I$.

3.4.4 Famille de LD

Les différentes logiques de description se distinguent par les constructeurs qu'elles proposent. Elles ont une base commune enrichie de différentes extensions :

AL **(Attributive Language) :** c'est le langage de base défini à partir des éléments syntaxiques suivants :

- A : Concept atomique
- $\top$: Le concept universel Top
- $\bot$: Le concept vide Bottom
- $\neg A$: Négation d'un concept atomique
- $C \sqcap D$: Conjonction de concepts

— $\forall r.C$: Quantificateur universel
— $\exists r$: Quantificateur existentiel non typé

Cette logique est décrite en détail dans la section suivante. D'autres langages, plus expressives, peuvent être définis en rajoutant d'autres constructeurs au langage AL, à savoir :

- $ALU = AL \cup \{C \sqcup D\}$: Disjonction de concepts ;
- $ALE = AL \cup \{\exists r.C\}$: Quantificateur existentiel typé
- $ALC = AL \cup \{\neg C\}$ (Attributive Language with Complement) : c'est la logique la plus importante, elle constitue la base de toute les logiques de description "expressifs". Ici, C est un concept primitif ou défini. On note qu'on peut coder la disjonction resp. la quantification existentielle à l'aide de la négation et la conjonction resp. la quantification universelle, pour obtenir ALU resp. ALE comme sous-logiques de ALC.
- $ALN = AL \cup \{\leq nr, \geq nr\}$: Les restrictions de nombres, désignées par la lettre N et notées par $\leq nr$ (restriction à moins de n) et $\geq nr$ (restriction à plus de n) où n représente un entier positif.

3.4.5 Logique minimale AL

On décrit une logique minimale nommée dans le sens où une logique moins expressive représente peu d'intérêt.

Constructeurs d'AL (syntaxe) : Les constructeurs offerts par AL pour l'édification de concepts composés.

$$
\begin{array}{rll}
C, D \rightarrow & A & \text{(concept atomique)} \\
& \mid \top & \text{(concept universel Top)} \\
& \mid \bot & \text{(concept vide Bottom)} \\
& \mid \neg A & \text{(négation d'un concept atomique)} \\
& \mid C \sqcap D & \text{(intersection de concepts)} \\
& \mid \forall R.C & \text{(quantification universelle complète)} \\
& \mid \exists R.\top & \text{(quantification existentielle limitée)}
\end{array}
$$

Sémantique formelle d'AL : Afin de supporter la notion de concepts composés, la fonction d'interprétation est étendue par les règles décrites ci-dessous.

$$
\begin{array}{rcl}
\top^I & = & \Delta_I \\
\bot^I & = & \emptyset \\
(C \sqcap D)^I & = & C^I \cap D^I \\
(\neg C)^I & = & \Delta_I - C^I \\
(\forall R.C)^I & = & \{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\} \\
(\exists R.\top)^I & = & \{a \in \Delta^I \mid \exists b.(a,b) \in R^I\}
\end{array}
$$

— Le constructeur $C \sqcap D$ permet de faire la conjonction de 2 concepts composés, i.e. représente l'ensemble des individus, membres des concept C et D pour une interprétation.
— Le quantificateur existentiel non typé $R.\top$ désigne l'ensemble des individus, membres du domaine d'un rôle R pour une interprétation donnée. Exemple : pour une interprétation I qui est un modèle de l'ABox et de la TBox du tableau 3.3, $\exists relationParentEnfant.\top$ équivaut l'ensemble des individus $\{Sophie^I, Robert^I\}$.

— Le quantificateur universel $\forall R.C$ évoque l'ensemble des individus du domaine d'un rôle R qui sont en relation, par le biais de R, avec un individu du concept C, pour une interprétation donnée. Exemple : pour une interprétation I qui est un modèle de l'ABox et de la TBox du tableau 3.3, $\forall relationParentEnfant.Homme$ équivaut l'ensemble des individus $\{Robert^I\}$.
— AL ne permet pas la spécification de rôles à l'aide de constructeurs (rôles composés).

3.4.5.1 Inférence

L'inférence s'effectue au niveau terminologique ou factuel.

L'inférence au niveau terminologique : Quatre principaux problèmes d'inférence se présentent au niveau terminologique : Satisfiabilité, Subsomption, Équivalence, Disjonction.

— Un concept C est satisfiable ou cohérent s'il existe une interprétation I telle que $C^I \neq \emptyset$.
— Un concept C est insatisfiable ssi pour toute interprétation I, $C^I = \emptyset$.
— Un concept D est subsumé par un concept C, noté $D \sqsubseteq C$, ssi pour toute interprétation I, $D^I \subseteq C^I$.
— Deux concepts C et D sont dits équivalents, noté $C \equiv D$, si et seulement si $C^I = D^I$ pour toute interprétation I.
— Deux concepts C et D sont incompatibles ou disjoints si et seulement si $C^I \sqcap D^I = \emptyset$ pour toute interprétation I.

Remarque :

Les quatre types de problèmes d'inférence peuvent être réduits à des problèmes de subsomption ou à des problèmes de satisfiabilité.

— Réduction des problèmes d'inférence d'une TBox à des problèmes de subsomption :
  C est insatisfiable $\iff$ C est subsumé par $\bot$
  C et D sont équivalents $\iff$ C est subsumé par D et D par C
  C et D sont disjoints $\iff$ $C \sqcap D$ est subsumé par $\bot$
— Réduction des problèmes d'inférence d'une TBox à des problèmes de satisfiabilité :
  C est subsumé D $\iff$ $C \sqcap \neg D$ est insatisfiable
  C et D sont équivalents $\iff$ $C \sqcap \neg D$ et $\neg C \sqcap D$ sont insatisfiables
  C et D sont disjoints $\iff$ $C \sqcap D$ est insatisfiable

L'inférence au niveau factuel : Le niveau factuel comprend quatre principaux problèmes d'inférence :

Cohérence : Une ABox A est cohérente par rapport à une TBox T si et seulement s'il existe un modèle I de A et T.

Vérification d'instance : Vérifier par inférence si une assertion $C(a)$ est vraie pour tout modèle I d'une ABox A et d'une TBox T.

Vérification de rôle : Vérifier par inférence si une assertion $R(a, b)$ est vraie pour tout modèle I d'une ABox A et d'une TBox T.

Problème de récupération : Pour une ABox A, un concept C d'une terminologie T, inférer les individus a $a_1^I ... a_n^I \in C^I$ pour tout modèle I de T.

3.5 Exercices corrigés

3.5.1 Énoncés

Exercice 3.1

1. Qu'apportent les logiques non classiques comparativement aux logiques classiques ?
2. Donner un exemple de logique multivaluée.
3. Donner un exemple de logique modale.

Exercice 3.2

1. Comment sont représenter les connaissance d'un domaine en logique de description ?
2. Pourquoi parle t-on de famille de logiques de description ?

Exercice 3.3

En utilisant les concepts : Homme, Femme, Humain et le rôle aEnfant, donner les définitions suivantes en logique de description :

- La classe des individus dont tous les enfants sont des femmes.
- La classe d'individus qui ont au moins un enfant.
- La classe d'individus humains qui n'ont pas d'enfants.
- La classe des individus Père.
- La classe des individus Mère.
- La classe des individus qui sont des humains ayant au moins un enfant et dont tous les enfants sont des femmes.

3.5.2 Solutions

Solution de l'exercice 1

1. Les logiques classiques ne prennent en compte que les connaissances parfaites (certaines, précises, ...) avec des valeurs de vérité Vrai ou Faux.

   Les logiques non classiques proposent de modéliser le raisonnement humain qui tolère l'imperfection tels que : l'incertitude (c'est possible), l'imprécision (très bon), l'aspect temporel (prévision), les croyances des personnes ou les leurs préférences, etc. De plus, les principes de tiers exclu et de monotonie ne sont pas nécessairement présents dans ces logiques.
2. La logique floue est une logique multivaluée. Elle accepte une infinité de valeur au lieu de 0 et 1. Nous pouvons représenter les énoncés tel que : "La course à durée long temps."
3. Une des logiques modales est la logique des possibilités qui reconnait les modalités nécessité et possibilité. Par exemple, "Si l'étudiant travaille sérieusement, il peut avoir la note complète."

Solution de l'exercice 2

1. En logique de description la connaissance est représentée sous forme de :
   — TBox pour le niveau terminologique qui représente les concepts et les rôles identifiés dans le domaine.
   — ABox pour le niveau assertionnel qui décrit la manipulation des individus.
   — Opérateurs de raisonnement permettant de manipuler les TBox et ABox et raisonner de dessus.
2. A partir d'une logique de description minimale, nous pouvons ajouter d'autres opérateurs afin de l'enrichir et d'avoir des logiques de description plus expressives mais plus complexes.

Solution de l'exercice 3

— Les individus dont tous les enfants sont des femmes :
$$ParentDeFemme \equiv \forall aEnfant.Femme$$
— Les individus qui ont au moins un enfant :
$$Parent \equiv \exists aEnfant.\top$$
— Les individus humains qui n'ont pas d'enfants :
$$PersonneSansEnfant \equiv Humain \sqcap \forall aEnfant.\bot$$
— Les individus Père :
$$Père \equiv Homme \sqcap \exists aEnfant.\top$$
— Les individus Mère :
$$Mère \equiv Femme \sqcap \exists aEnfant.\top$$
— Les individus qui sont des humains ayant au moins un enfant et dont tous les enfants sont des femmes :
$$Humain \sqcap \exists aEnfant.\top \sqcap \forall aEnfant.Femme$$
A noter que la définition suivante n'est pas juste : $Humain \sqcap \forall aEnfant.Femme$ Pour qu'un individu ne fait pas partie de cette définition il suffit de trouver au moins un de ses enfants qui n'est pas une fille. Cependant un humain qui n'a pas d'enfant fait partie de cette définition.

3.6 Conclusion

Dans ce chapitre, nous avons présenté les formalismes de représentation de connaissances basés sur la logique qui peuvent être scindés en deux grandes catégories : les logiques classique qui sont représentées par la logique propositionnelle et la logique des prédicats, et les logiques non classiques qui sont représentées par les logiques modales, les logiques multivaluées, etc. Plus particulièrement, la famille des logiques de description à introduite d'une manière plus approfondie.

Le chapitre suivant abordera quelques langages de représentation de connaissances utilisés pour échanger les données entre différents machines ou systèmes informatiques hétérogènes.

Chapitre 4

Les langages de représentation de connaissances

4.1 Introduction

Ce chapitre présente les langages de représentation de connaissances tels que KIF, XML et RDF ainsi que les systèmes de représentation de connaissances KL-ONE, CLASSIC et LOOM.

Les langages de représentation de connaissances se caractérisent par leur syntaxe, c'est-à-dire la manière dont on écrit les phrases ou les formules, et la sémantique, c'est-à-dire la manière dont sont interprétées ces formules.

4.2 Knowledge Interchange Format (KIF)

Le langage KIF a été proposé en 1992 par M. R. Genesereth et R. E. Fikes [18]. Il a été conçu pour permettre à l'origine des échanges d'information entre des systèmes disparates, plus spécifiquement des systèmes créés par différents programmeurs à différents moments dans des langages différents etc. Il n'a pas été conçu pour l'interaction avec les humains.

KIF est basé sur la Logique du Premier Ordre (LPO) et ressemble beaucoup à Prolog mais avec une syntaxe totalement parenthésée (à la Lisp). En utilisant ce langage, un agent peut exprimer :

- Des propriétés sur les choses. Exemples : Farid est jeune ; Fatah est cool ; Lucy est une chienne.
- Des relations entre les choses : Exemples : Alice et Kenza sont amies ; Lila est la fille de Kenza ;
- Des propriétés générales sur des ensembles de choses. Exemples : Chaque être humain a une mère.

Les propriétés générales de KIF ont une sémantique de type déclaratif (Il est possible de comprendre le sens des expression sans faire appel un interprète comme avec Prolog qui une sémantique opérationnelle) et expriment totalement les expressions de LPO et peuvent représenter des méta-connaissances.

4.2.1 Expressions en KIF

KIF comprend une variété d'opérateurs logiques pour faciliter le codage des informations logiques (comme les négations, les disjonctions, les règles, les formules quantifiées, etc.)

4.2.1.1 Structures de données

Nous donnons des exemples pour illustrer la structure de données utilisée par une version préfixé de KIF.

On représente l'information liée à trois salariés sous la forme d'un 3-uple de préfixe salary :

< salary Identificateur, Département, Salaire > :
(salary 015-46-8700 widgets 72000)
(salary 016-44-9832 printing 50000)
(salary 017-22-1234 sales 12398000)

On peut représenter des informations plus complexes comme la température de la variable m1 :

(= (temperature m1) (scalar 80 Celsisus))

4.2.1.2 Définition de concepts nouveaux

On peut définir des concepts en fonction de concepts précédemment définis. Par exemple, un célibataire est un homme qui n'est pas marié :

(defrelation celibataire (?x) := (and (homme ?x) (not (marie ?x))))

Où :
- ?x représente une variable
- homme et marie sont des concepts définis précédemment
- and, not comme = et := sont prédéfinis en KIF

Également, on peut définir une nouvelle relation en fonction des relations déjà existantes comme "<=" en fonctions des relations prédéfinies < et =

(defrelation <= (?x ?y) :=(or (= ?x ?y) (< ?x ?y)))

4.2.2 Règles

On peut représenter des relations entre des ensembles. Par exemple, toute chose ?x qui est une personne est automatiquement un mammifère :

(defrelation person (?x) :=> (mammal ?x))

Ou encore pour dire que x^n | n pair est un nombre positif :

(=> (and (real-number ?x) (even-number ?n)) (> expt ?x ?n) 0))

Méta connaissances : On utilise les symboles «‘» et «, » pour quoter des connaissances dans des connaissances.

Si joe veut savoir tous les items de type salary on écrira :

(interested joe ‘(salary, ?x , ?y , ?z))

Si on avait écrit (salary ?x ?y ?z) cela voulait dire que joe est intéressé par l'expression (salary ? x ? y ?z) en elle-même et non en les trois instances qu'elle représente (plus haut).

Programmes : KIF peut également être utilisé pour décrire des procédures, c'est-à-dire pour écrire des programmes ou des scripts que les agents doivent suivre. Compte tenu de la syntaxe préfixe de KIF, de tels programmes ressemblent à Lisp ou Scheme. Voici un exemple de procédure en trois étapes écrite en KIF. La première étape consiste à s'assurer qu'il y a une nouvelle ligne sur le flux de sortie standard ; la deuxième étape consiste à imprimer Hello world ! sur le flux de sortie standard ; la dernière étape consiste à ajouter un retour chariot pour obtenir une nouvelle ligne.

```
(progn (new-line t)
  (print "hello world !")
    (new-line t))
```

4.3 EXtensible Markup Language (XML))

Qu'est ce que XML ? XML est l'abréviation d'eXtensible Markup Language. C'est un langage de balisage structuré destiné pour : la description, le stockage, et le transfert de données. Il est indépendante de toute plateforme et recommandé par W3C.

Différence entre XML et HTML : XML décrit la structure de données alors que HTML permet l'affichage de données. De plus, les balises HTML sont prédéfinies (limitées) alors que les balises XML sont extensibles selon le besoin de l'utilisateur. Donc, XML ne remplace pas HTML mais ils se complètent.

Dans la plus part des application web, XML est utilisé pour le stockage et transfert de données alors que HTML est utilisé pour l'affichage.

Balises définissables : L'apport de XML par rapport à HTML c'est la possibilité d'utiliser n'importe quel symbole comme balise :

<**myTag**> ceci est une d'extension d'entité qui est un myTag </**myTag**>

Il faut s'entendre sur le sens de myTag ce qui est fait de manière définitoire par des consortium de standardisation, et l'imbrication des balises entre-elles ce qui revient à définir une grammaire.

Exemple 4.1: Code XML et HTML

La Figure 4.1 montre la différence entre le code XML et HTML.

Code XML (Atelier1_Exemple1.xml)	Code HTML (Atelier1_Exemple1.html)
<BIBLIOTHEQUE> <LIVRE> <TITRE>titre livre 1</TITRE> <AUTEUR>auteur 1</AUTEUR> <EDITEUR>editeur 1</EDITEUR> </LIVRE> <LIVRE> <TITRE>titre livre 2</TITRE> <AUTEUR>auteur 2</AUTEUR> <EDITEUR>editeur 2</EDITEUR> </LIVRE> <LIVRE> </BIBLIOTHEQUE>	 titre livre 1 auteur 1 <u>editeur 1</u> titre livre 1 auteur 1 <u>editeur 1</u>

FIGURE 4.1 – Exemples de codes XML et HTML.

Grammaire = DTD : Un document XML possède deux niveaux :

syntaxe de base pour vérifier l'imbrication correcte des balises sans tenir compte des symboles, et que deux balises ne se chevauchent pas. Par exemple, à tout <foo> correspond un </foo>.

sémantique (mais qui n'en est pas) qui tient compte des imbrications réciproques des balises. Par exemple, une balise ¡adresse¿ doit obligatoirement contenir dans cet ordre exactement une balise <rue>, une balise <numero>, une balise <code> et une balise <ville>.

<adresse>
<rue> Turing </rue>
<numéro> 2 </numéro>
<code> 91000 </code>
<ville> Orsay </ville>
</adresse>

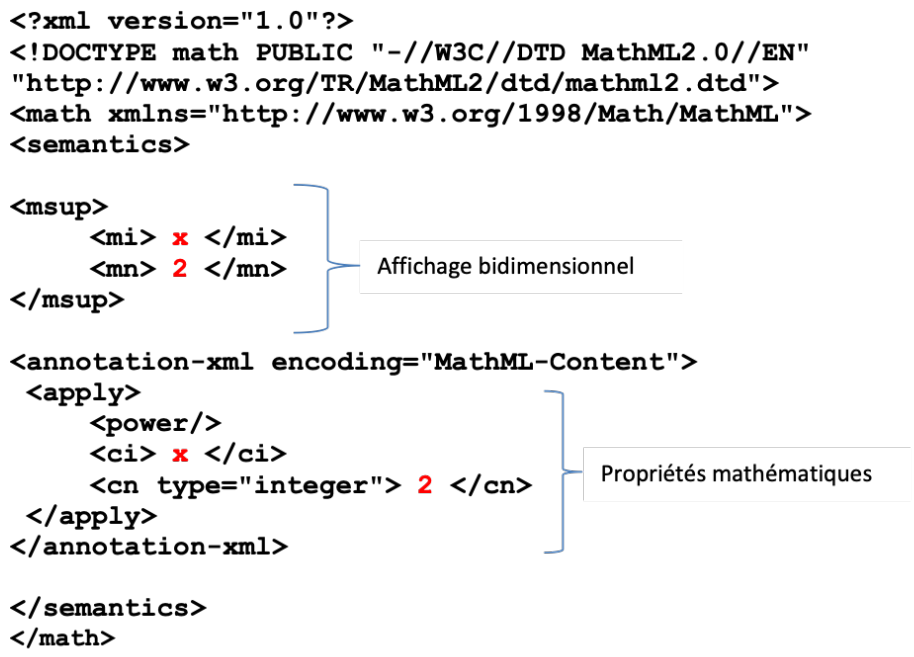
Cette grammaire est décrite dans un fichier séparé appelé DTD (Data Type Definition).

Exemple 4.2: Exemple de DTD : MathML

Voici comment Mathematica exporte la formule «x^2» au format MathML, qui est une DTD définie par le consortium W3C.
Export["test.xml", x^2, "MathML"] produit le fichier texte «test.xml» de la Figure 4.2.

4.4 RDF

L'une des critiques qui a été faite au sujet des données de l'Internet, en particulier dans le Web, est que ces informations sont «machine-readable» (compréhensible au niveau lexical–syntaxique) mais pas «machine-understandable». En effet, il faut d'autres informations (méta-informations) émanent de la part de celui qui a produit les informations pour qu'un outil logiciel indépendant puisse traiter ces informations.




```
<?xml version="1.0"?>
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML2.0//EN"
"http://www.w3.org/TR/MathML2/dtd/mathml2.dtd">
<math xmlns="http://www.w3.org/1998/Math/MathML">
<semantics>

<msup>
     <mi> x </mi>
     <mn> 2 </mn>
</msup>

<annotation-xml encoding="MathML-Content">
 <apply>
     <power/>
     <ci> x </ci>
     <cn type="integer"> 2 </cn>
 </apply>
</annotation-xml>

</semantics>
</math>
```

FIGURE 4.2 – Exemple de fichier XML.

Notion de méta données : C'est dans sens que W3C a alors élaborer une couche supplémentaire au dessus de XML appelée RDF (Resource Description Framework) afin de traiter des méta informations. RDF utilise la syntaxe XML.

4.4.1 Entités de base de RDF

En RDF, il y a trois sortes d'entités : les ressources, les propriétés, les statements (ou assertions).

Ressources : Les éléments atomiques au sujet desquelles RDF donne des descriptions (les données) sont appelées des ressources :

- Une page web «http ://www.limsi.fr/Individu/jps»
- Une partie d'une page Web : une ancre
- Un objet accessible via le web : un livre, ...

Les ressources sont toujours référées via un pointeur de type URI (Uniform Resource Identifier).

Propriétés : C'est les attributs (une caractéristique, un aspect, une propriété, ...) d'une ressource. Chaque propriété a :

- Un sens spécifique
- Un domaine de valeurs autorisé
- Le type de ressources auquel elle s'applique
- Des relations avec d'autres propriétés.

Statements (ou assertions) : Un fichier RDF est un ensemble de statements qui sont des assertions de propriétés au sujet de ressources. Un statementest est composé :

— sujet : une ressource (URI)
— prédicat : une propriété (son symbole)
— objet : la valeur de cette propriété pour la ressource

L'objet, i.e. la valeur d'une propriété, peut être, soit un pointeur vers une autre ressource (une URI), soit un littéral : valeur primitive en XML.

4.4.2 Exemples RDF graphiques

Exemple 4.3: Exemple 1

Soit la phrase descriptive suivante : «J-PSansonnet est le créateur de la page http ://www.limsi.fr/Individu/jps».
Elle peut se décomposer en trois parties :
— Sujet (ressource) : http ://www.limsi.fr/Individu/jps
— Prédicat (propriété) : Creator
— Objet (littéral) : J-PSansonnet

En RDF, on représente graphiquement un statement en utilisant les notations graphiques suivantes :
— Sujet (ressource) : un cercle
— Prédicat (propriété) : une flèche
— Objet (littéral) : un rectangle.

Cette phrase est représentée par la Figure 4.3.

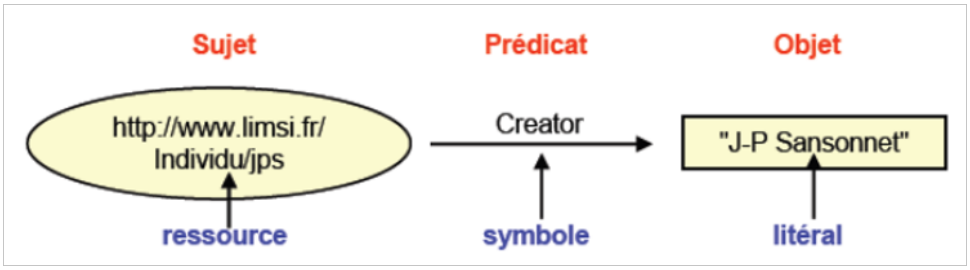


FIGURE 4.3 – Représentation de «J-PSansonnet est le créateur de la page http ://www.limsi.fr/Individu/jps»

Exemple 4.4: Exemple 2

Soit la phrase descriptive suivante : «L'individu dont le nom est J-PSansonnet et dont le mail est jps@limsi.fr est le créateur de la page http ://www.limsi.fr/Individu/jps».
L'objet n'est pas atomique mais une entité structurée. En RDF, cette entité est représentée par une ressource (un cercle) mais elle est anonyme, c'est-à-dire sans texte à l'intérieur comme le montre la Figure 4.4.

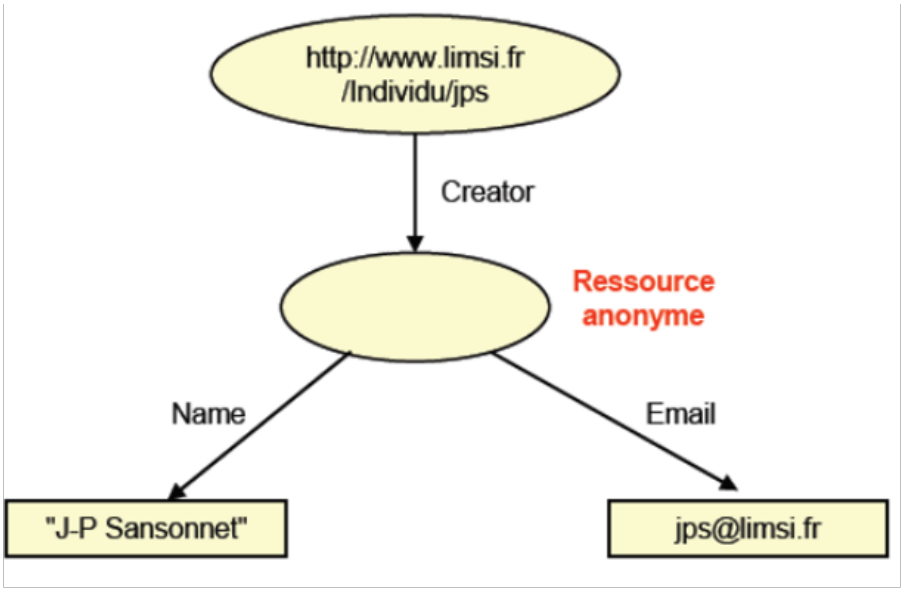


FIGURE 4.4 – Représentation de «L'individu dont le nom est J-PSansonnet et dont le mail est jps@limsi.fr est le créateur de la page http ://www.limsi.fr/Individu/jps».

Exemple 4.5: Exemple 3

Soit la phrase descriptive suivante : «L'individu qui est identifié dans notre base du personnel sous le n°123456 a pour nom J-PSansonnet et pour mail jps@limsi.fr ; il est le créateur de la page http ://www.limsi.fr/Individu/jps ».
Il est possible d'identifier la ressource anonyme de l'exemple 2 en pointant sur le fichier du personnel qui est situé à : «http ://www.limsi.fr/staff/123456 » (voir la Figure 4.5).

4.4.3 Syntaxe de base de RDF

RDF étant un modèle de données sous forme de graphe, il n'a pas à proprement parler de syntaxe. En fait, il y a plusieurs syntaxes possibles pour représenter une description RDF. La plus connue est la syntaxe RDF/XML qui, utilisant le méta-langage XML, permet de créer des descriptions facilement manipulables par la machine. Afin de pouvoir spécifier les contraintes de toute syntaxe utilisée pour représenter un graphe RDF, une syntaxe abstraite a été définie. Nous allons la décrire brièvement à travers des exemples de graphes RDF représentée dans la syntaxe RDF/XML.

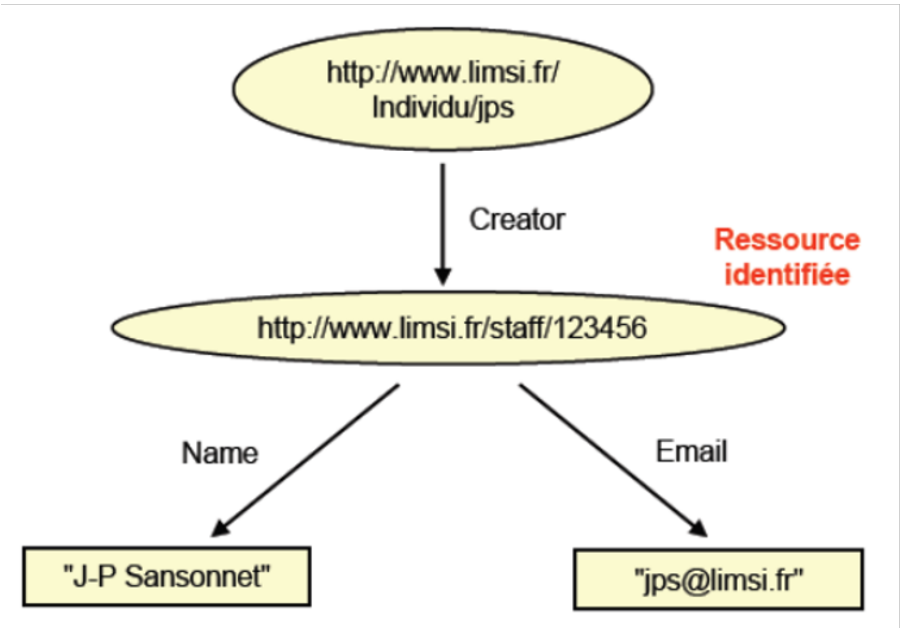


FIGURE 4.5 – Représentation de «L'individu qui est identifié dans notre base du personnel sous le n°123456 a pour nom J-PSansonnet et pour mail jps@limsi.fr ; il est le créateur de la page http ://www.limsi.fr/Individu/jps ».

La syntaxe de base de RDF est la suivante :

1. RDF : := ['<rdf :RDF>'] description* ['</rdf :RDF>']
2. description : := '< rdf :Description' idAboutAttr ? '>' propertyElt* '</rdf :Description>'
3. idAboutAttr : := idAttr | aboutAttr
4. aboutAttr : := 'about="' URI-reference'"'
5. idAttr : := 'ID="' IDsymbol'"'
6. propertyElt : := '<' propName'>' value '</' propName'>'|'<' propNameresourceAttr'/>
7. propName : := Qname
8. value : := description | string
9. resourceAttr : := 'resource="' URI-reference'"'
10. Qname : := [NSprefix' :'] name
11. URI-reference : := string, interpretedper[URI]
12. IDsymbol : := (anylegalXML namesymbol)
13. name : := (anylegalXML namesymbol)
14. NSprefix : := (anylegalXML namespaceprefix)
15. string : := (anyXML text, with"<", ">", and"&" escaped)

Un élément RDF est un simple wrapper qui délimite dans un document XML un contenu qui est du RDF.

Exemple 4.6: Reprenons l'exemple 1

La phrase descriptive «J-PSansonnet est le créateur de la page http ://www.limsi.fr/Individu/jps» s'exprime en suivant la syntaxe textuelle de RDF par le listing 4.1.
On remarque que tout le graphe RDF est décrit à l'intérieur des balises rdf :RDF. Deux espaces de nommage y sont spécifiés **:rdf** et **:s**. La description de la ressource "J-PSansonnet" est spécifiée par la balise **rdf :Description**, avec l'URI de la ressource indiquée par l'attribut **rdf :about**.

Listing 4.1 – RDF textuel de l'exemple 1

```
<rdf:RDF>
    <rdf:Description about="http://www.limsi.fr/Individu/jps">
        <s:Creator>J-PSansonnet</s:Creator>
    </rdf:Description>
</rdf:RDF>
```

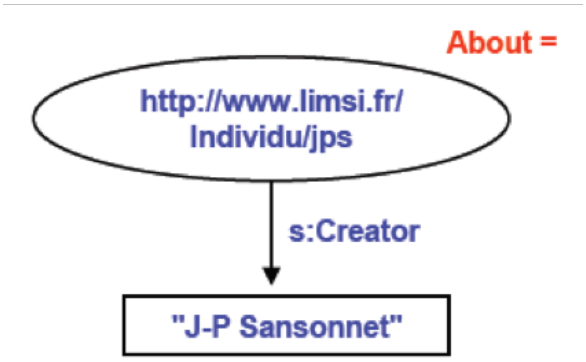


FIGURE 4.6 – Représentation graphique de l'exemple 1.

Normalement cette description est incluse dans un fichier XML qui déclare les préfixes comme **rdf :RDF** et surtout les namespaces utilisés comme **:rdf** et **:s**. Le document XML complet est donné par le listing 4.2.

Listing 4.2 – XML/RDF textuel de l'exemple 1

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://description.org/schema/">
%%Declaration des domaines tags utiliss
    <rdf:Description about="http://www.limsi.fr/Individu/jps">
        <s:Creator>J-PSansonnet</s:Creator>
    </rdf:Description>
</rdf:RDF>
```

Exemple 4.7: Reprenons l'exemple 3 précédent

La phrase descriptive «l'individu qui est identifié dans notre base du personnel sous le n°123456 a pour nom J-PSansonnet et pour mail jps@limsi.fr ; il est le créateur de la page http ://www.limsi.fr/Individu/jps » s'exprime en suivant la syntaxe textuelle de RDF du listing 4.3.

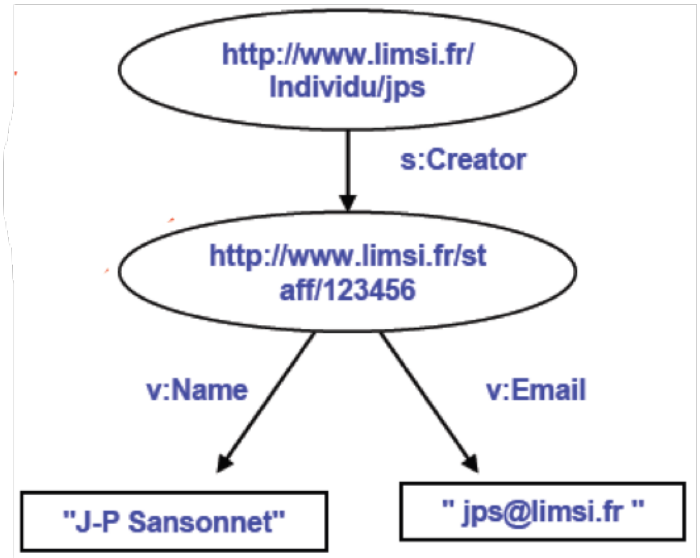


FIGURE 4.7 – Représentation graphique de l'exemple 3.

Listing 4.3 – Listing XML/RDF

```
<?xml version="1.0"?>
<rdf:RDF>
    xmlns:rdf="http://www.w3.org//1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://description.org/schema/">
    xmlns:v="http://description.org/schema/..">
    <rdf:Description about="http://www.limsi.fr/Individu/jps">
        <s:Creatorrdf:ressource="http://www.limsi.fr/Individu/jps
           /123456">
    </rdf:Description>

    <rdf:Description about="http://www.limsi.fr/Individu/jps
       /123456">
        <v:Nom> J-PSansonnet </v:Nom>
        <v:Email> jps@limsi.fr </v:Email>
    </rdf:Description>
</rdf:RDF>
```

4.4.4 Types de containers RDF

Souvent, il est nécessaire de référer non pas à un entité isolée mais à un agrégat d'entités. Alors, RDF définit trois types d'agrégats :

Bag : un ensemble non ordonné de ressources ou de littéraux. Il peut y avoir des réplications dans un bag. (RDF ne propose pas le concept d'ensemble au sens mathématique du terme où les éléments ne sont pas répliqués).

Séquence : une liste ordonnée de ressources ou de littéraux. Il peut y avoir des réplications dans une séquence.

Alternative : un ensemble non ordonné de ressources ou de littéraux qui représentent des possibilités diverses pour une même valeur de propriété. Il ne peut pas y avoir de réplication.

Ces types d'agrégats sont déclarés dans une propriété supplémentaire qui est une relation **rdf :type**

Fonction d'accès : on peut référer un élément dans un agrégat RDF en utilisant la syntaxe "_1", "_2", "_3", etc.

Exemple 4.8: Agrégat bag

La phrase «Les étudiants du cours DEA/I3/SMA sont jean, paul, pierre, louis et marie» est modélisée en RDF par un agrégat de type bag comme le montre la Figure 4.8.

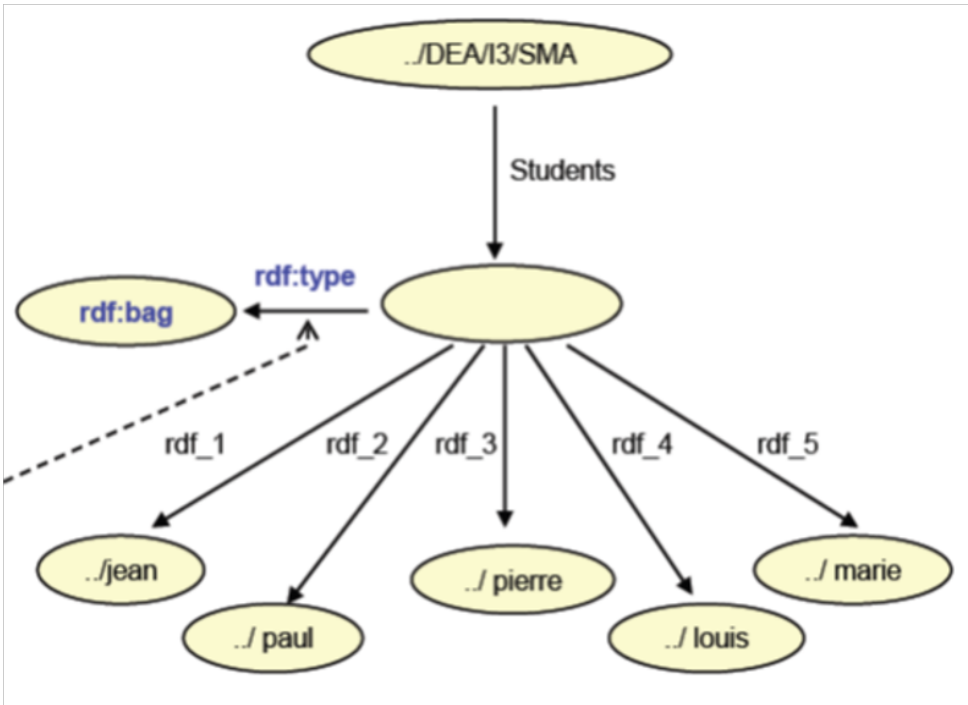


FIGURE 4.8 – Représentation graphique de l'agrégat bag.

4.5 Les systèmes de représentation des connaissances

Parmi les systèmes de représentation de connaissances les plus connus nous distinguons : KL-One, CLASSIC, LOOM

4.5.1 KL-One

KL-One est un langage de représentation des connaissances dont la conception a débuté au début des années 70. Les propriétés générales de KL-One sont comme suit :

- KL-One est spécialisé dans la représentation des concepts.
- Ce langage se situe à la frontière entre les réseaux sémantiques et les frames, donc proche de la langue naturelle avec la notion de sémantique lexicale.
- Il propose uniquement comme outils de raisonnement la classification de concepts.
- Il est muni d'une sémantique de type LPO.

Les concepts de KL-One sont munis : d'attributs, appelés rôles, qui expriment les relations entre concepts ; des contraintes sur le type, et la cardinalité de la valeur du type.

Les contraintes et la cardinalité sont associées à tout rôle, à la manière des facets d'un langage de frames. La cardinalité d'un rôle ou encore son «arité» est une couple d'entiers (min, max) spécifiant :

- min : le plus petit nombre d'arguments que prend la relation/rôle.
- max : respectivement le plus grand (peut être ∞).

Exemple 4.9: Exemple KL-One

On définit le concept **Parent** par : "$x \in$ Parent $\iff x \in$ PERSONNE $\wedge \exists y \in$ PERSONNE | enfant$(x, y, \cdot)$". Autrement dit, il y a une PERSONNE y qui apparaît comme un des arguments de la relation enfants(x, arg1, arg2, ...).
Un concept est défini par spécialisation, c'est-à-dire qu'il hérite de tous les rôles du concept dont il est tiré : tout $x \in$ *Parent* possède automatiquement un rôle date-de-naissance même si cela n'est pas explicité dans le schéma (voir la Figure 4.9).

Deux sortes de concepts sont proposés par KL-One :

Prédéfinis : donnés par le système et considérés comme génériques, i.e. admis comme tels par «tout le monde ». Ils sont notés tout en MAJUSCULES.

Définis : construits par l'utilisateur de KL-One à partir des types prédéfinis. Ils sont notés avec une Majuscule.

A noter que les rôles sont en minuscule.

Notion de spécification : deux modes de spécification sont proposés par KL-One :

- Les concepts **prédéfinis** sont *incomplètement* spécifiés. Ils expriment des conditions d'appartenance nécessaires mais pas suffisantes. Par exemple, si un individu est une PERSONNE alors il a une date de naissance mais l'inverse n'est pas vrai ; un ANIMAL possède une date de naissance mais n'est pas une PERSONNE.
- Les concepts **définis** sont considérés comme *complètement* spécifiés. Ils expriment des conditions d'appartenance nécessaires et suffisantes. Par exemple, si un individu possède et vérifie tous les rôles d'un concept défini alors il appartient au concept et vice versa.

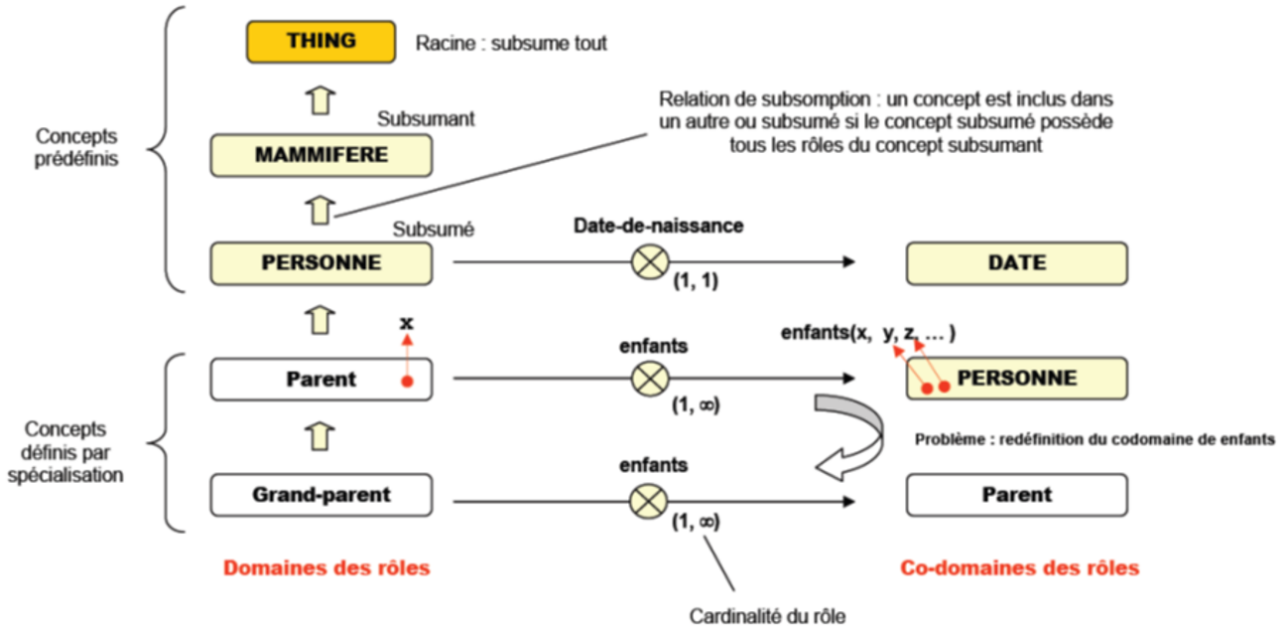


FIGURE 4.9 – Définition des concepts et rôles dans KL-One.

4.5.2 CLASSIC

Le système opérationnel CLASSIC est l'implémentation la plus connue des langages de DL. Sa syntaxe est la suivante :

Concept $\Longrightarrow$ THING | Conceptname
| And(Concept+)
| All(RoleName, Concept)
| AtLeast(Integer, RoleName)
| AtMost(Integer, RoleName)
| Fils(RoleName, Individualname+)
| SameAs(attribut, attribut)
| OneOf(Individualname*)
| Test (functionarg*)

La signification de certains opérateurs :

OneOf : permet de définir un type énuméré

SameAs : permet de dire que la valeur d'un attribut est la même qu'un autre (un enfant à le même nom que son père).

Fils : associe une valeur élémentaire à un rôle.

Test : fonction externe qui renvoie des valeurs prédicatives à trois états : T, Nil ou ?.

Règles : L'originalité du langage CLASSIC est d'associer des règles aux concepts. Si un individu est instancié (c'est-à-dire classé) comme un concept C alors la règle R(C) est appliquée :

Musicien-debutant ≡ (AND Musicien (AtMost 1 instrument)
(AtMost 1 professeur)
(ALL pratique FAIBLE)

)
(RULE instrument-musicien-debutant
Musicien-debutant
(ALL instrument instrument-etude)
)

Si un individu #joe est classé comme Musicien-debutant alors il est automatiquement subsumé par l'expression (ALL instrument instrument-etude).

Requêtes : Toute expression peut s'interpréter comme une requête. Par exemple, l'extension de l'expression conceptuelle E ≡ (AtLesat 2 enfant) fournit l'ensemble des individus qui satisfont E.

Plus généralement, une requête est représentée par un concept Q et satisfaire la requête Q consiste à classifier Q dans la hiérarchie de concepts, i.e. à trouver le plus petit concept qui subsume Q.

4.5.3 LOOM

LOOM est une plate-forme pour la représentation des connaissances. Son langage est à base de frames dans la tradition de KL-ONE avec une sémantique formelle qui mappe les déclarations dans LOOM aux déclarations dans la théorie des ensembles et Logique du premier ordre.

Les connaissances déclaratives dans LOOM sont composées de définitions, de règles, de faits, ...

Pour compiler ces connaissances, LOOM se sert d'un moteur déductif qui en réalité est un "classifieur" qui utilise l'unification sémantique ainsi que l'orienté objet.

4.6 Exercices corrigés

4.6.1 Énoncés

Exercice 4.1

Représenter en XML la bibliographie qui contient la liste de livres suivants :

- Alain Michard a écrit le livre "XML langage et applications" apparu en 2001 chez Eyrolles avec isbn 2-212-09206-7. Il est consultable sur lien http ://www.editions-eyrolles/livres/michard/.
- Le livre de Jeffrey Zeldmann intitulé "Designing with web standards édité chez New Riders en 2003 a pour isbn 0-7357-1201-8.

Remarque : l'identifiant et la langue du livre sont considérés comme des attributs.

Exercice 4.2

Soit la phrase suivante : "La personne enregistrée sous le numéro 1234 dans la base des ressources humaines a pour nom MS Kanous et mail ms.kanous@univ-bejaia.dz et est le propriétaire de la page https ://www.univ-bejaia.dz/staff/.
Représenter cet énoncé en utilisant XML/RDF.

4.6.2 Solutions

Solution de l'exercice 1

Chaque livre est caractérisé par le titre, l'auteur, l'éditeur, l'année de parution, le numéro ISBN et éventuellement une URL.

Listing 4.4 – Listing XML

```
<?xml version="1.0"?>
<bibliography>
    <book key="Michard01" lang="fr">
        <title>XML langage et applications</title>
        <author>Alain Michard</author>
        <year>2001</year>
        <publisher>Eyrolles</publisher>
        <isbn>2-212-09206-7</isbn>
        <url>http://www.editions-eyrolles/livres/michard/</
           url>
    </book>
    <book key="Zeldman03" lang="en">
        <title>Designing with web standards</title>
        <author>Jeffrey Zeldman</author>
        <year>2003</year>
        <publisher>New Riders</publisher>
        <isbn>0-7357-1201-8</isbn>
    </book>
</bibliography>
```

Solution de l'exercice 2

Listing 4.5 – Listing XML/RDF

```
<?xml version="1.0"?>
<rdf:RDF>
    xmlns:rdf="http://www.w3.org//1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://description.org/schema/">
    xmlns:v="http://description.org/schema/..">
    <rdf:Description about="https://www.univ-bejaia.dz/staff
       /">
        <s:Proprietaire:ressource=1234/>
    </rdf:Description>

    <rdf:Description about="1234">
        <v:Nom> MS Kanous </v:Nom>
        <v:Email> ms.kanous@univ-bejaia.dz </v:Email>
    </rdf:Description>
</rdf:RDF>
```

4.7 Conclusion

Cette conclusion fait l'objet des chapitres 2, 3 et 4 qui ont présentés divers modes de représentation de connaissances. En effet, nous avons vu des formalismes qui offrent une certaine facilité et lisibilité mais manquent de rigueur tels que les réseaux sémantiques, les graphes conceptuels, les frames et les scripts. Des formalismes logiques : les logiques classiques (des propositions, des prédicats) qui ne permettent de représenter que les énoncés vrais ou faux et les logiques non classiques (floue, modales, etc.) pour la représentation des informations imparfaites.

A travers les logiques de description nous avons conclu que la richesse en expressivité conduit à plus de complexité (lecture toujours difficile) et les mécanismes de « calcul » (projection, subsomption, etc.) ne permettent pas des expressions de requêtes toujours « simples ».

Nous avons également brièvement revu plusieurs langages de représentation de connaissances tels que KIF, XML, RDF qui permettent d'inter-changer des données entre des systèmes hétérogènes. Sans oublier les systèmes de représentation de connaissances à l'image de KL-ONE, CLASSIC et LOOM.

Chapitre 5

Les Systèmes à Bases de Connaissances

5.1 Introduction

Les systèmes à bases de connaissances (SBC) constituent un domaine majeur en intelligence artificielle sur lequel ils ont eu un impact certain. Apparus vers 1975, sous le nom de système expert qui disparaît au profit du concept plus général de SBC. Ils sont conçus pour approcher les performances d'experts humains en permettent l'exploitation des grandes quantités de connaissances acquises auprès de ceux-ci dans des domaines spécifiques. Sous un autre angle, ils sont vus comme des auxiliaires d'aide à la décision.

5.2 Concepts de base

Le principe de base des SBCs est la séparation entre les connaissances nécessaires pour résoudre un problème et les mécanismes de raisonnement exploitant ces connaissances comme le montre la Figure 5.1.

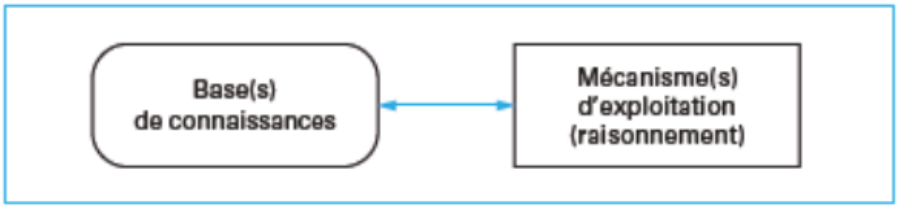


FIGURE 5.1 – Schéma simplifié d'un SBC.

Le **mécanisme de raisonnement** est appelé également interpréteur, moteur d'inférence, raisonneur ...

Les **caractéristiques importantes** de cette séparation sont :

— la maintenance facile des SBCs,
— la modification de connaissances,
— et l'ajout de nouvelles connaissances à la base existante.

5.2.1 Connaissances

Les connaissances exploitées dans les SBC peuvent être de différentes formes telles que :
— des objets du monde ;
— des faits concernant ces objets, par exemple « La Terre est ronde, légèrement aplatie aux pôles. » ;
— des classifications, par exemple, les taxinomies en zoologie ;
— des évènements comme « La température du four a commencé à fluctuer. » ;
— enfin, des règles heuristiques de savoir-faire comme « Si l'étudiant reste bouche bée, souvent il n'a rien compris. », etc.

Nous trouvons également les **méta-connaissances** (ou connaissance sur la connaissance) qui joue un rôle de guidage du raisonnement. Elle représente souvent une part notable de l'expertise humaine.

5.2.1.1 Types de connaissances

Les connaissances exploitées dans les SBCs peuvent être relatives au temps, à savoir :
— permanentes, par exemple « L'être humain a deux pieds. » ;
— temporaires, comme la description d'une maladie fournie à un système de diagnostic, par exemple « Le patient a de la fièvre et de la toux. » ;
— des hypothèses émises au cours d'un raisonnement telle que «le patient a peut-être attrapé à un coup de froid».

D'autres attributs peuvent qualifier les connaissances, nous pouvons citer la précision ou la certitude avec laquelle elles sont données et l'évolution au cours du temps, etc.

5.2.1.2 Modes de représentation des connaissances

Le fait de parler de raisonnement sur des connaissances dans un système d'IA implique les modes de représentation de connaissances, les plus courants sont :
— les représentations logiques (vues dans le chapitre 3) ;
— les règles de production (présentées dans la 5.3.3) ;
— les réseaux sémantiques et les objets structurés (vus dans le chapitre 2) ;
— les logiques de description (vues dans le chapitre 3) ;
— les ontologies (présentées dans le chapitre 6).

Remarque : une base de connaissances peut être codée avec un ou plusieurs modes de représentation.

5.2.1.3 Base de connaissances

Dans de nombreux systèmes, une distinction est faite entre la base de connaissances et la mémoire de travail, parfois appelée base de faits.
— la base de connaissances contient la connaissance opératoire.
— la mémoire de travail contient les données initiales et les hypothèses successives émises.

Ces deux bases codent l'ensemble des connaissances nécessaires à la résolution du problème.

5.2.2 Raisonnement

5.2.2.1 Modes de raisonnement et inférence

Les modes d'inférence utilisés dans les SBC sont très variés. Nous allons décrire brièvement les plus utilisés.

Déduction : fournit des conséquences à partir de faits.

Si la base de faits contient A et on a la proposition A → B
alors on peut déduire B

Contraposition : permet de raisonner sur les non-observations.

Si l'on a A → B et non B
alors on peut inférer non A

Abduction : cherche à remonter aux causes expliquant les observations.

De A → B et de l'observation de B,
on infère que A est une cause possible de B

Induction : permet d'inférer des règles à partir d'observations régulières ou habituelles.

Si l'on a B à chaque fois que l'on a A,
on peut inférer A → B

Projection : fournit des conséquences à partir d'actions.

Si la base de faits contient la proposition A → B et que l'on a le fait A,
on s'attend à ce que B soit réalisé

Planification : établit les actions à exécuter pour attendre des buts.

Si l'on veut avoir le fait B et que la base contient A → B,
alors on peut inférer l'action A.

5.2.2.2 Raisonnement monotone et non monotone

Selon le type de la logiques utilisé pour la représentation de la connaissance, nous distinguons deux catégories de raisonnement :

Le raisonnement monotone : dans ce type de raisonnement, plus d'informations conduit naturellement à plus de conclusions. Si à partir d'une base de connaissances KB, on arrive à déduire le fait A, on déduira également A de KB ∪ B. C'est le cas des logiques propositionnelle et du premier ordre.

Le raisonnement non monotone : dans ce cas, une nouvelle information peut invalider des conclusions précédentes :

— Afin de raisonner en présence de connaissances imparfaites, on est amener à émettre des hypothèses.
— Cependant, peuvent elles être remises en cause si de nouvelles informations sont disponibles.
— Les hypothèses à la source de la non-monotonie sont l'utilisation de propriétés typiques ; les exceptions possibles ; l'hypothèse de monde fermé, etc.
— Donc, la non-monotonie repose sur les notions de préférence, révision de croyances, ...

5.3 Système expert à base de règles de production

5.3.1 Définition d'un système expert

Un système expert (SE) est un programme conçu pour simuler le comportement d'un humain qui est un spécialiste ou un expert dans un domaine.

— Il incorpore à la fois des connaissances formelles et des jugements personnels de l'expert ;
— Il explique à la fois le raisonnement conduisant aux réponses aux questions et le savoir qu'il contient ;
— Il est capable d'incorporer du nouveau savoir-faire dans le savoir existant de manière incrémentale.

5.3.2 Architecture d'un SE

La Figure 5.2 illustre l'architecture d'un système expert. Ses différents modules sont présentés dans les sous-sections suivantes.

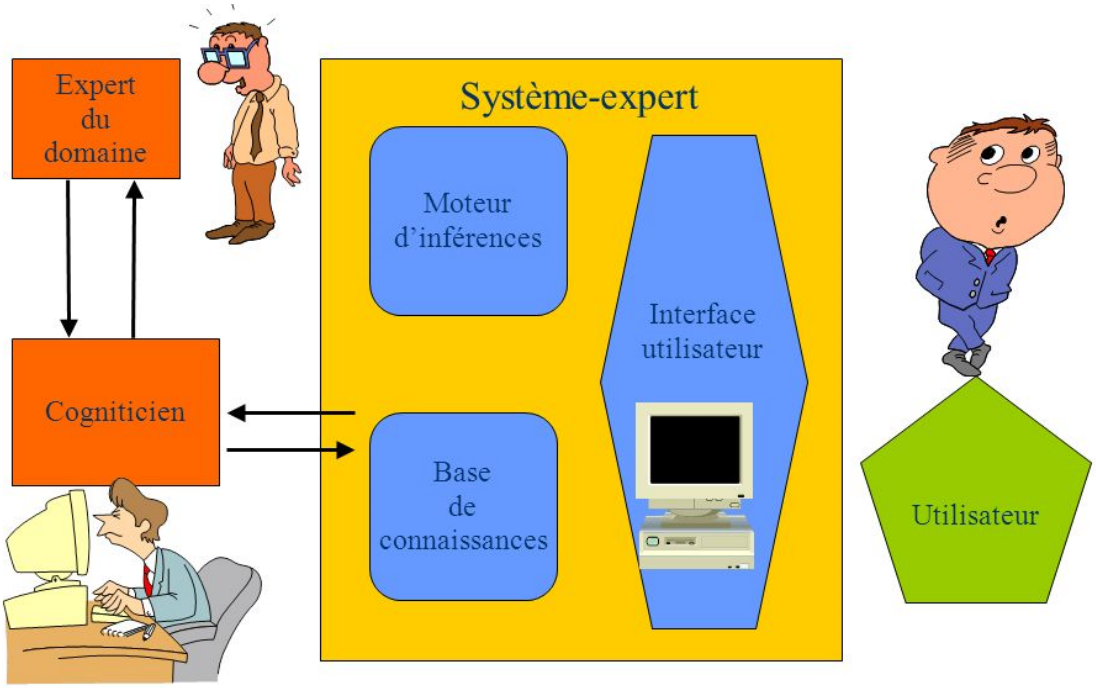


FIGURE 5.2 – Structure générale d'un système expert

5.3.3 Base de connaissances

La base de connaissance (BC) représente la connaissance spécifique d'un expert. Elle est répartie en une base de faits et une base de règles (une répartition purement logique).

Base de faits (BF) : contient les connaissances représentant des états considérés comme prouvés. Elle constitue la mémoire de travail du SE, variable au cours de l'exécution et vidée lorsque l'exécution se termine. Les faits peuvent prendre des formes plus ou moins complexes. Un exemple de fait est « La température est de 39.5°C. »

Base de règles (BR) : contient les connaissances déductives ou règles, de type opératoire. Elle n'évolue pas au cours d'une session de travail. Un exemple de règle est «Si le malade a une température élevée, s'il a mal à la tête et s'il a des vomissements, alors penser à la possibilité d'une méningite.»

Le SE peut disposer aussi d'une méta-connaissance exprimée par des **méta-règles** (des règles sur la manière d'utiliser les règles).

Règle de production : dans les SEs à base de règles de production, une règle est de la forme :

Si conditions Alors conclusion (α)

- conditions (prémisse, antécédent) : une formule logique qui doit être vérifiée pour que la règle s'applique.
- conclusion (conséquent) : correspond à l'ajout ou au retrait d'un fait ou d'une hypothèse, au déclenchement d'une action...
- coefficient numérique α : veut dire que la connaissance exprimée par la règle est entachée d'imperfection. Le coefficient peut être un degré de vraisemblance, un facteur de certitude, de plausibilité, de degré de vérité, etc. Ces coefficients sont pris en compte par les mécanismes de raisonnement dits approximatifs.

A noter que les règles de production manipulées dans ce cours sont nettes, c'est-à-dire sans le coefficient α.

5.3.4 Moteur d'inférence

Un moteur d'inférence (MI) : est un programme qui exploite la connaissance et scrute à travers la BC et détermine comment les faits et les règles doivent être gérés fonction de la situation courante fournie par la BF.

Un MI peut fonctionner selon trois mécanismes :

- en chaînage avant qui est guidé par les données, au cours duquel les règles sont utilisées dans le sens « conditions vers conclusion » ;
- en chaînage arrière qui est guidé par un but ou un modèle, revenant à utiliser une règle dans le sens « conclusion vers conditions » ;
- et en chainage mixte où le MI fonctionne dans les deux sens.

5.3.5 Autres modules

Les autres modules d'un SE sont les suivants :

Le module d'acquisition de connaissances : C'est un outil précieux qui aide le concepteur lors de l'élaboration de la BC : constitution, mise au point, consultation, mise à jour et vérification de la cohérence.

Le module interface : Il est destiné à faciliter le dialogue, si possible en langue naturelle ou à l'aide de graphe entre l'utilisateur et le SE.

Le module d'explication d'explication : Il est sollicité lors d'une consultation du système pour justifier la conclusion à laquelle il a abouti en spécifiant, par exemple, les raisons pour lesquelles il a activé certaines règles.

5.3.6 Domaines d'applications des systèmes experts

Les SE sont tous liés à des domaines bien définis car il est très difficile de regrouper des connaissances hétérogènes dans un même système. Néanmoins, il existe certains systèmes qu'on désigne par multi-experts et qui sont conçus pour manipuler ces connaissances hétérogènes.

Diverses applications de SE existent dans différents domaines, nous allons citer quelques unes :

Médecine

- MYCIN (Université, Ecole Polytechnique de Lausanne, 1984)
  - Il dialogue en langage naturel avec le médecin qui l'assiste pour diagnostiquer et même traiter les maladies bactériennes du sang.
  - Plusieurs variantes de ce système ont été développées pour entraîner des étudiants en médecine.
  - Diagnostic médical tentant de cerner les germes responsables de maladies infectieuses au niveau du sang et du liquide céphalo-rachidien (méningite).
- TOUBIB (IBM - France, Paris, 1982)
  - Un système d'aide au diagnostic en médecine d'urgence.

Géologie

- PROSPECTOR (S.R. International, Stanford USA) : 1600 règles
  - Il aide le géologue à évaluer l'intérêt d'un site en vue de prospection minière.
- LITHO (Schlumberger-France, Clamart 1981) : 500 règles
  - Il se préoccupe de la détermination des lithofaciès les plus plausibles en fonction des paramètres recueillis au cours de la descente d'outils de forage.

Biologie

- MOLGEN (Université de Stanford, U.S.A., 1977)
  - Il engendre un plan de manipulations génétiques en vue de construire une entité biologique donnée.

Administration - Gestion

— AGE (Bell Laboratoires, USA 1983) : 100 règles.
- Il est destiné à établir des rapports synthétiques sur l'état de câbles téléphoniques, à partir des enregistrements d'incidents

— TROPICAID (C.H.U la pitié Selpetrière, 1984).
- Système d'aide à la décision médicale pour les infirmiers des pays en voie de développement.
- Il comporte une base de données pharmacologiques concernant 200 médicaments,
- Il comporte aussi une BD thérapeutique concernant 500 maladies et une BD diagnostiques se référant à 2000 signes cliniques ou para cliniques.
- Le système est développé sur un Micro-ordinateur portatif.

Production Manufacturière

— GUNNEX (1984) : 300 règles
- Il engendre des gammes de fabrication de produits en caoutchouc.
- Il est écrit en Prolog.

Chimie

— Heuristic DENDRAL (Université de Stanford, U.S.A., 1969)
- Il recherche la formule développée d'un corps organique à partir de la formule brute et du spectrogramme.

— CRYSTALIS (Université de Stanford, U.S.A., 1979)
- Il recherche la structure de protéines à partir de résultats d'analyse cristallographique.

Mathématiques

— MUSCADE (G.R. 22, Université Paris VI, 1984)
- Il est capable d'établir des démonstrations de théorèmes complexes en manipulant le concept de méta-connaissances.

— MACSYMA
- Il fait la résolution de problèmes mathématiques (intégrales, différentielles, systèmes d'équations...).

Électricité

— SOPHIE (Université de Stanford, U.S.A., 1975)
- Il est utilisé pour le diagnostic des pannes des circuits électriques.

— PEACE1 et PEACE2 (O.N.E.R.A. - C.E.R.T, Toulouse, 1981)
- Il est utlisé pour l'analyse et la conception des circuits électroniques passifs.

5.4 Caractéristiques d'un moteur d'inférence

Le moteur d'inférence ou de résolution peut être indépendant du domaine d'application. Il est capable d'exploiter la BC afin de résoudre les problèmes posés par les utilisateurs. Il est caractérisé par les éléments suivants :

— Un cycle de base ;
— Les méthodes de chaînage ;
— Les stratégies de recherche que nous n'allons pas traiter puisqu'elles sortent du cadre de ce cours.

5.4.1 Cycle de base

Le cycle de base est donné par la Figure 5.3. Il se fait en deux grandes phases : évaluation et exécution.

5.4.1.1 Phase d'évaluation

Sélection

C'est la sélection d'un sous-ensemble de règles dans la base des règles. Ce choix dépend de l'état courant de la base de faits (en utilisant les méta-règles par exemple).

Filtrage

C'est le choix des règles applicables (un sous-ensemble dit de conflit). En effet, le MI compare une règle avec les faits de la BF pour déterminer si celle-ci est applicable.

Résolution de conflits

Le moteur décide de la (des) règle(s) qui sera (seront) exécutée(s). Ce choix est souvent guidé :

— par des soucis d'efficacité (par exemple, la première règle de la liste, la moins complexe, la moins utilisée...).
— par des informations heuristiques ou des méta-règles pour fixer un ordre de priorité entre ces règles (la plus prometteuse, la plus fiable,...).

5.4.1.2 Phase d'exécution

Exécution

Cela revient à appliquer (ou activer) la règle sélectionnée et généralement ajouter un ou plusieurs faits à BF :

— Modification de l'ensemble de faits (générer des états intermédiaires) ;
— Questionner l'usager afin d'avoir plus d'informations ;
— Exécuter les actions externes ;

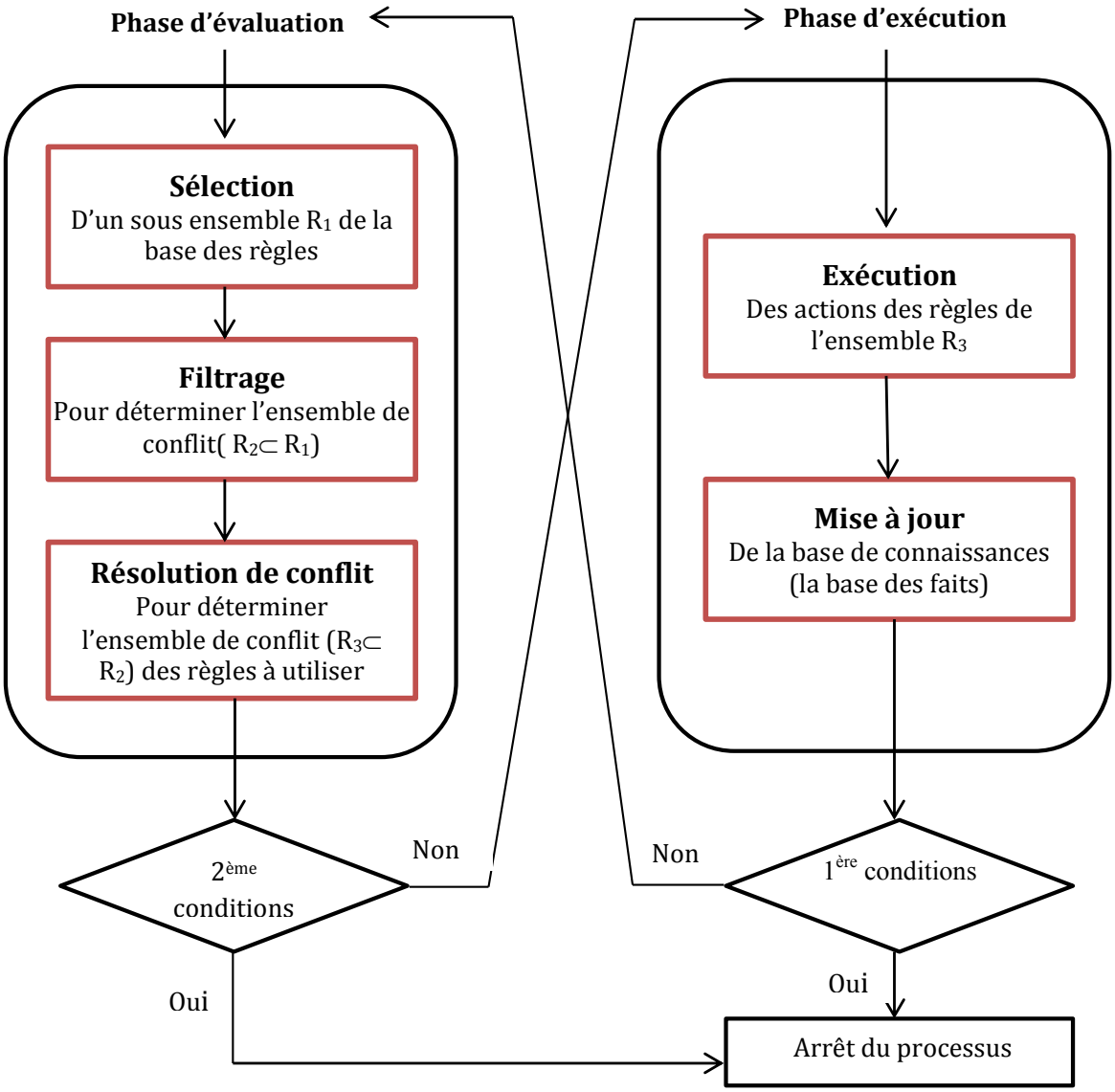


FIGURE 5.3 – Cycle de base d'un SE.

Arrêt de ce cycle

Le cycle peut s'arrêter dans la phase :

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

5.4.2 Modes de raisonnement du MI

L'invocation des règles peut se faire en chainage avant, en chainage arrière ou d'une manière mixte.

5.4.2.1 Chaînage avant

Principe

Le chainage avant est guidé par les données, i.e. partir des faits pour arriver au but. Il ne sélectionne que les règles dont la partie prémisse est vérifiée par les faits présents dans BF. Il ajoute les nouveaux faits à l'ensemble des faits. Ce processus est réitéré et s'arrête :

- avec succès dès que le but est atteint ;
- avec échec quand il n'y a plus de règles applicables.

Nous allons appliquer le chainage avant sur l'exemple de BR et BF suivantes :

Exemple 5.1: Exemple de base de connaissances (BR+BF)

Base de règles
R1 : A → E
R2 : B → D
R3 : H → A
R4 : E et G → C
R5 : E et K → B
R6 : D, E et K → C
R7 : G, K et F → C

Base de Faits initiale (H,K)

Déroulement

On veut savoir si on peut déduire le fait C de BC .
Les règles sont déclenchées dans l'ordre d'écriture de BR et les faits sont insérés immédiatement dans BF. Après application d'une règle, on reprend dés le début de BR.

- Seule R3 est applicable. Déclenchement de R3 et BF devient BF = (H, K, A) ;
- Seule R1 es applicable. Déclenchement de R1, BF = (H, K, A, E)
- Seule R5 est applicable. Déclenchement de R5, BF = (H, K, A, E, B)
- Seule R2 est applicable. Déclenchement de R2, BF = (H, K, A, E, B, D)
- Seule R6 est applicable. Déclenchement de R6 et obtention du fait C.

Le moteur s'arrête car il a prouvé le fait C.

5.4.2.2 Chaînage arrière

Principe

Ce mode est dirigé par les buts, i.e. partir du but comme hypothèse et tenter de remonter aux faits pour le démontrer. Il ne sélectionne que les règles dont la conclusion contient le but. Les conditions inconnues de la prémisse de ces règles deviennent des sous-buts à démontrer. Il s'arrête :

- avec succès lorsque tous les sous-buts sont vérifiés ;
- avec échec lorsqu'il ne peut plus sélectionner de règles ou poser de questions (nombre insuffisant de règles, incohérence entre elles, ...)

Nous allons appliquer le chainage arrière sur l'exemple 5.1 précédent.

Déroulement

On veut savoir si on peut déduire le fait C de la base de BC.
Les règles sont déclenchées dans l'ordre d'écriture de la base BR. BF ne change que si l'on arrive à prouver un fait.
Pour prouver C, les règles déclenchables sont R4, R6, R7.

— La règle R4 est choisie, et le moteur veut prouver E et G ;
  - Prouver E : La règle R1 est choisie, et le moteur veut prouver A ;
    - Prouver A : La règle R3 est choisie, et le moteur veut prouver H ;
      H est dans BF, donc A et E sont prouvés et sont ajoutés à BF ; BF = (H, K, A, E) ;
  - Prouver G : Aucune règle ne contient G dans sa partie conclusion, donc aucune règle n'est déclenchable.

Donc, la règle R4 est un **échec**.

5.4.2.3 Chaînage mixte

Le MI fonctionne en chaînage mixte ou bidirectionnel et BF comprend des faits considérés comme établis et des faits à établir.

— Le chaînage **mixte** utilise les mêmes règles (dîtes alors mixtes), soit en avant, soit en arrière, suivant les cas. Les conditions de déclenchement des règles peuvent porter simultanément sur des faits établis ou à établir.
— Le chaînage **bidirectionnel** enchaîne des cycles en chaînage avant (avec des règles en avant) et des cycles en chaînage arrière (avec des règles en arrière).

5.4.3 Autres caractéristiques du MI

5.4.3.1 Régimes de contrôle

Dans le cas où la phase d'exécution déboucherait sur un échec (par exemple, si l'ensemble R3 est vide dans le cycle de base), un MI adopte deux types de conduites :

Régime irrévocable

Si au cours d'une résolution, lors de l'étape de déduction, le moteur ne peut déclencher aucune règle ; il s'arrête s'il est en mode de fonctionnement irrévocable et ceci même si le but n'est pas atteint.

Par tentatives

On peut remettre en cause l'application d'une règle si ce choix débouche sur un échec. On fait retour arrière, i.e. exploration d'un seul chemin avec retour au dernier choix effectué.

En général, les MI en chaînage arrière utilisent une stratégie par tentatives alors que les MI en chaînage avant procèdent par stratégies irrévocables.
Nous allons revenir sur l'exemple de chaînage arrière (appliqué sur l'Exemple 5.1).

Déroulement

On veut savoir si on peut déduire le fait C de la base de BC.
Les règles sont déclenchées dans l'ordre d'écriture de la base BR. BF ne change que si l'on arrive à prouver un fait.
Pour prouver C, les règles déclenchables sont R6 et bR7 avec R4 qui était un échec. On reprend BF avec les ajoutés BF = (H, K, A, E) ;

— La règle R6 est choisie, et le moteur veut prouver D, E et K ;
  - Prouver D : La règle R2 est choisie, et le moteur veut prouver B ;
    - Prouver B : La règle R5 est choisie, et le moteur veut prouver E et K ;
      E et K sont dans la base de faits ; donc R5 est un succès, B est prouvé et BF = (H, K, A, E, B) ;

    Puisque B est prouvé alors R2 est un succès et D est prouvé ; BF = (H, K, A, E, B, D) ;
  - Prouver E et K : E et K sont dans la base de faits ; BF = (H, K, A, E, B, D) ;

  La règle R6 est un **succès** et C est prouvé.

Le moteur s'arrête car il a **prouvé** le fait C.

5.4.3.2 Types de MI

Le mode d'inférence d'un système expert est lié aux types d'informations traitées (et bien sûr est issue de la logique utilisée : propositionnelle, des prédicats ou des logiques non classiques). Il existe plusieurs types de moteurs :

— Moteurs d'ordre 0 capables d'exploiter des règles ne comportant pas de variable quantifiée ;
— Moteurs d'ordre 1 permettant un usage illimité de variables dans les règles ;
— Moteurs d'ordre 0+ autorisant un usage limité des variables (dans la mise en correspondance des prémisses avec les faits de la base ou des conclusions avec les buts courants, il n'est possible que de substituer une constante à une variable). Il s'ensuit une meilleure maîtrise de l'explosion combinatoire pouvant se produire lors de l'utilisation d'un moteur d'ordre 1.

5.5 Méthodologie de développement d'un SBC

La Figure 5.4 est un schéma illustratif des éléments qui rentrent dans le développement d'un projet SBC. Nous distinguons trois niveaux, 1) l'acquisition de connaissances, 2) l'application en question (le système expert) et enfin 3) les utilisateurs de l'application. Comme toute application informatique, le développement d'un projet SBC suit un cycle de vie.

5.5.1 Développement d'un SBC

La Figure 5.5 présente un cycle de vie d'un projet SBC.

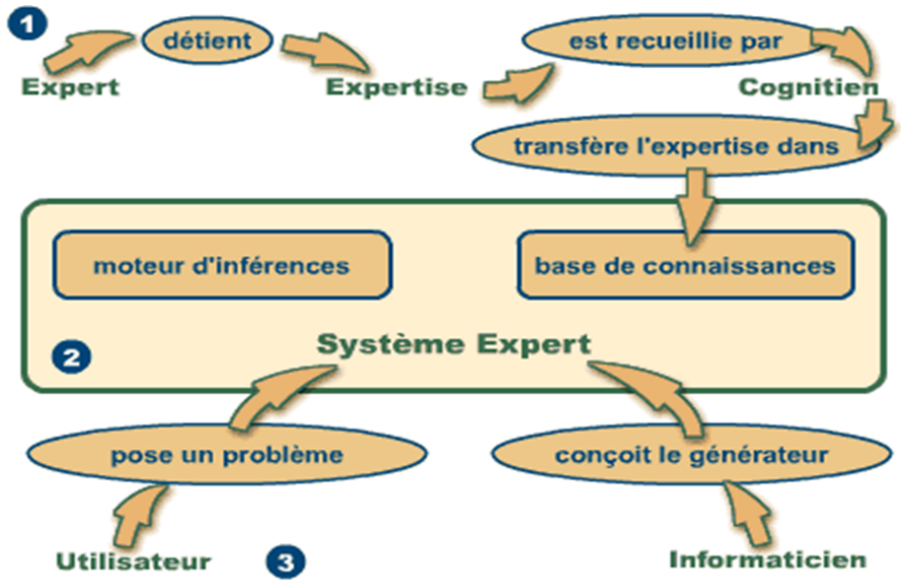


FIGURE 5.4 – Développement d'un SBC

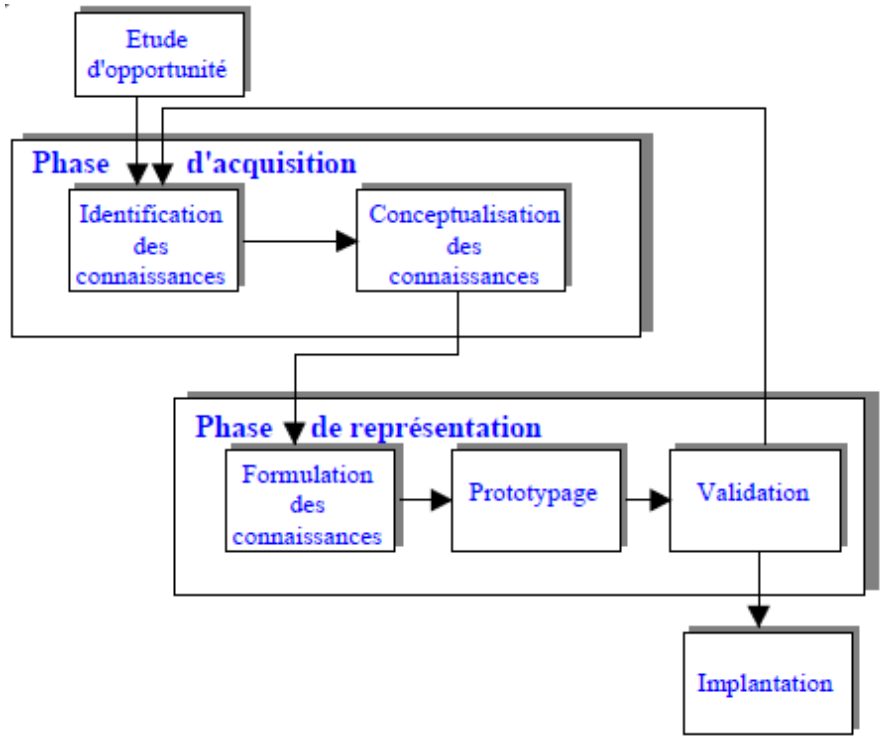


FIGURE 5.5 – Cycle vie d'un projet SBC

5.5.1.1 Phase Etude d'opportunité

Finalité

Définir les objectifs globaux du projet et en déterminer l'opportunité de réalisation (faisabilité, rentabilité, délais, charges de réalisation, conséquences pour le personnel)

Que doit-on faire ?

- Spécifier les objectifs et l'intérêt du système expert : tâches à accomplir et problème à résoudre ;
- Etudier la faisabilité technique du projet ;
- Identifier les différentes solutions possibles au problème ;
- Localiser l'expertise ;
- Identifier les futurs utilisateurs du système ;
- Evaluer les ressources humaines, matérielles et logicielles à allouer au projet ;
- Déterminer l'opportunité de réalisation du projet (Coûts/Intérêt).

5.5.1.2 Phase d'acquisition des connaissance

Finalités

Cette phase sert à repérer les éléments de base constituant l'expertise tels que :

- Les objets permettant de décrire un état du problème ;
- Les relations qui décrivent les liens entre les objets ;
- Les opérateurs qui indiquent comment modifier un état du problème ;
- Les procédures qui décrivent un ordre d 'application des opérateurs ;
- Les règles qui décrivent les inférences logiques
- Les plans qui organisent l'application des règles ;
- Les stratégies qui organisent l'enchaînement des plans.

Elle sert également à définir le modèle conceptuel des connaissances.

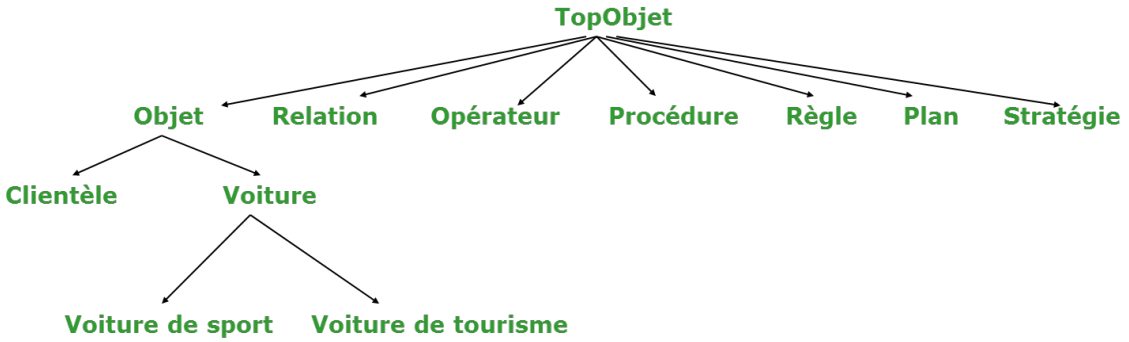
Que doit-on faire ?

- Étudier la documentation existante ;
- Planifier des entrevues avec l'expert ;
- Appliquer des techniques et méthodes d'acquisition des connaissances.

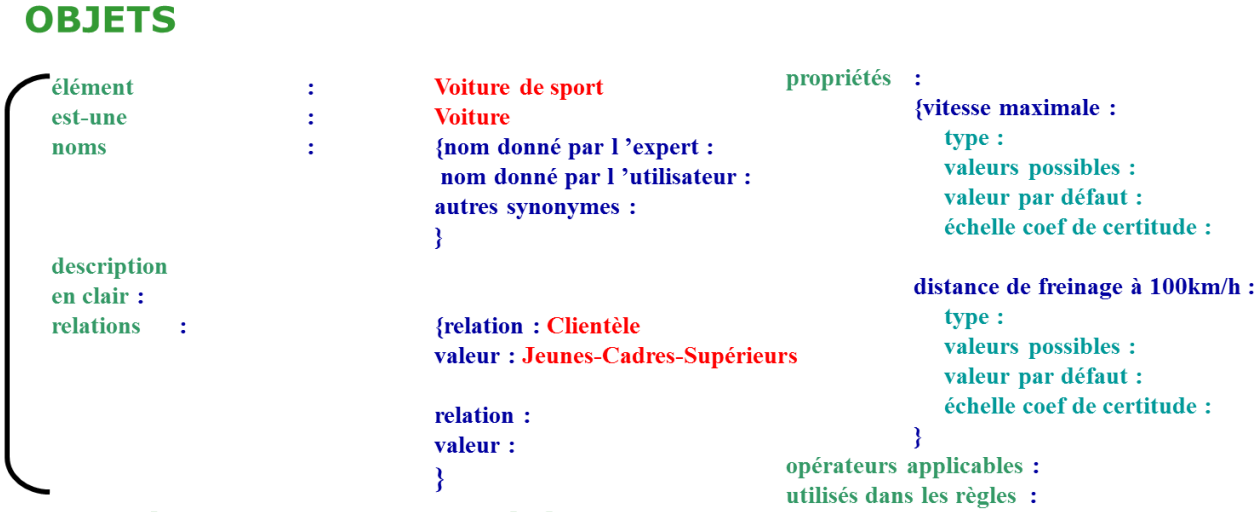
Identification des connaissances C'est l'identification des éléments clefs de la connaissance. Par exemple, les connaissances d 'un commercial spécialisé dans la vente des voitures données par les Figures 5.6 (a), (b) et (c).

Conceptualisation des connaissances Les questions à se poser avant les entrevues :

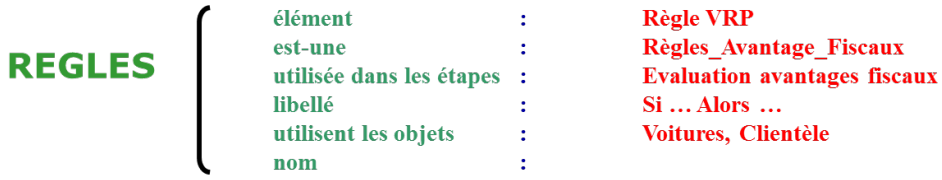
- Quelles sont les données à l 'entrée (les problèmes) et en sortie (les solutions)
- Quelles sont les relations qui existent entre les données ?
- Peut-on fragmenter les problèmes en unités plus petites ?



(a) Structure hiérarchiques des éléments de la connaissance.



(b) Modèle descriptif des éléments de la connaissance.



(c) Modèle descriptif des éléments de la connaissance.

FIGURE 5.6 – Des éléments de la connaissance.

— Dans quel ordre et sous quelle forme les données à l'entrée sont-elles acquises ?
— Dans quel ordre et sous quelle forme les données en sortie sont-elles produites ?
— Quelle est l'importance et quelle est la précision des diverses données ?

— Quelles sont les données susceptibles de manquer ?
— Quels sont les types d'entrées qui causent des difficultés à l 'expert ?
— Quels sont les postulats de l 'expert ?
— Quelles sortes d'inférences fait-il ?
— Comment en arrive-t-il à des hypothèses ?
— Quelles sont les relations entre ces hypothèses ?
— Comment l'expert passe-t-il d'un stade d'opinion à un autre ?

Les techniques utilisées pour la conceptualisation des connaissances sont :

Les entretiens sont de plusieurs types : discussion ouverte, discussion guidée et inversion des rôles.

Les questionnaires comme les exemples de la Figure 5.7.

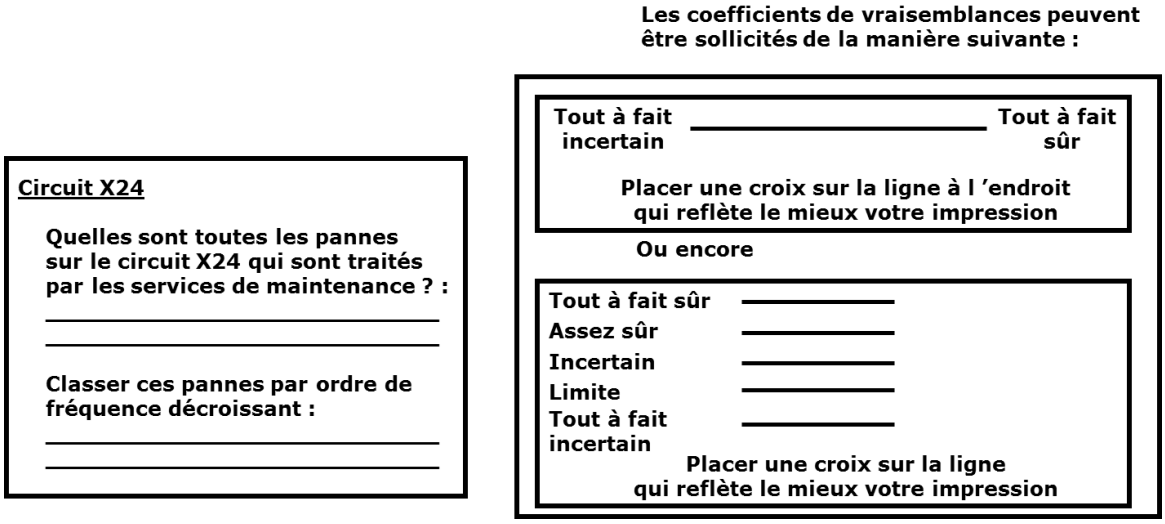


FIGURE 5.7 – Exemples de questionnaires.

Les observations sur le site comme les exemples de la Figure 5.8.

FIGURE 5.8 – Exemples d'observations sur le site.

L'induction par la machine a pour principe de générer automatiquement et par induction de nouvelles connaissances à partir d'exemples choisis par l 'expert du domaine. Le processus du Data mining est donné à titre d'exemple par la Figure 5.9.

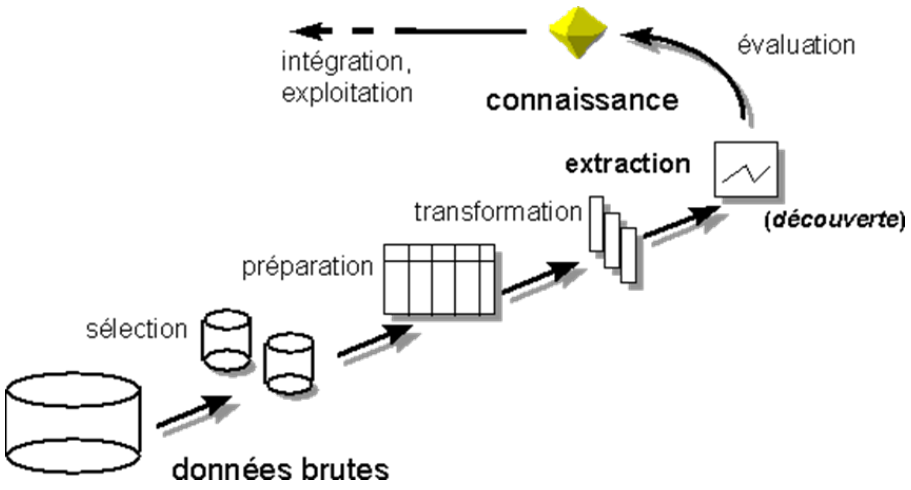


FIGURE 5.9 – Processus de data mining.

5.5.1.3 Phase de représentation des connaissances

Cette phase est composée de la formalisation de connaissances, le prototypage et la validation.

Formalisation des connaissances :

Finalité

Transposer dans une représentation appropriée (un système à base de règles par exemple) le modèle des connaissances que le cogniticien a conçu en phases d 'acquisition.

Cela se fait en trois étapes qui sont :
— Le choix du formalisme : Frames, Réseaux sémantiques, Règles de production, ...
— Le choix des méthodes d'inférences ;
— La conception de la base de connaissances.

Prototypage :

Finalité

Coder le formalisme choisi lors de la phase précédente pour obtenir un prototype du système expert.

Cela se fait en plusieurs étapes :
— Le choix d'un outil de développement ;
— Le codage du formalisme ;

— La mise au point de l'interface utilisateur ;
— La préparation d'un protocole de validation.

Validation :

Finalité

— Définir un protocole de validation ;
— Évaluer qualitativement et/ou quantitativement le comportement du prototype ;
— Vérifier la conformité du prototype aux spécifications du cahier des charges.

Un protocole de validation doit être défini sur :

Les éléments à valider : la base de connaissances, l'ergonomie, la maintenance, la documentation, la formation des utilisateurs, et l'outil de développement.

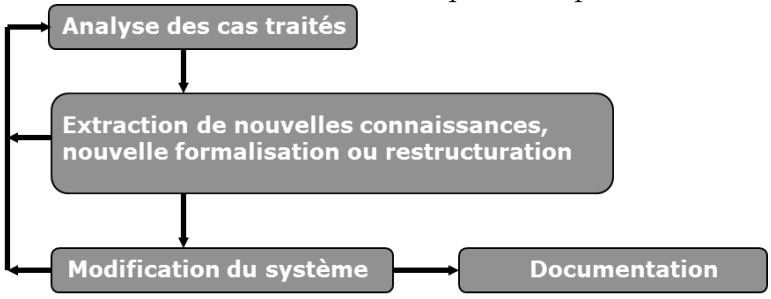
Les critères de validation :

— Performance : Justesse du cheminement, exactitude des conclusions, précision des conclusions, temps de réponse, ressources informatiques utilisées.
— Sensibilité et robustesse.
— Facilité d'apprentissage par les utilisateurs et acceptabilité.
— L'approche : Comment valider ? Quantitativement (statistiques) ou qualitativement (consensus).
— Le contenu : Que faut-il valider ? Quelle version de prototype il faut valider ?
— Le lieu : Où valider ? Sur des sites pilotes (cas réels) ou sur des sites « de validation » (cas simulés) ?
— Les acteurs : Avec qui faut-il valider ? Les experts, les utilisateurs, le cogniticien, le chef de projet ?

Les tests de validation :

— Représentatifs ;
— Nombreux : Prototype : 50-100 ; Système final : 100-300 ;
— Impliquent : Temps des valideurs, formation des valideurs, ressources matérielles, tenue des dossiers.

Le scénario de validation : comme par exemple le scénario suivant :



5.5.1.4 Phase Implantation

Finalité

Mettre le système final à la disposition des utilisateurs.

Les étapes pour cela sont la préparation, l'installation et la mise au point, la mise en exploitation, la définition d'un plan de formation et la mise en place des procédures de maintenance.

5.5.2 Rôles des acteurs d'un projet système expert

Les acteurs d'un projet système expert sont le Cogniticien (1), l'Expert (2), le Chef de projet (3) et les Utilisateurs (4). La Figure 5.10 montre les phases dans lesquelles chaque acteur intervient.

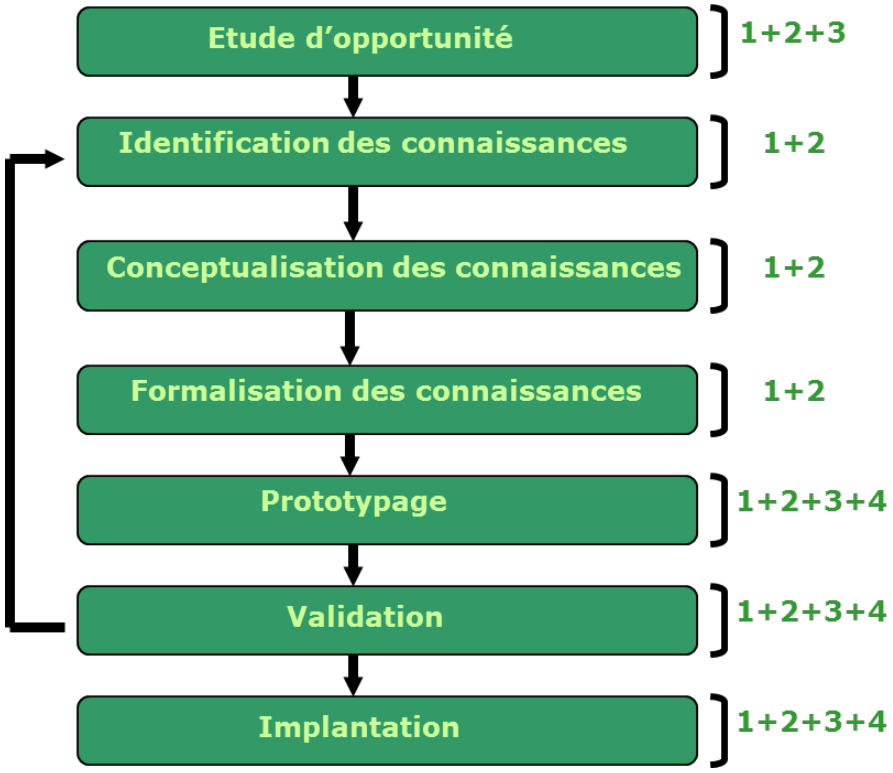


FIGURE 5.10 – Intervention des acteurs dans les phases d'un projet SBC.

Le cogniticien

Assure deux types d'interfaces
- Interface Expert-Machine : Il acquiert le savoir de l'expert, le modélise et l'implante dans la machine ;
- Interface Utilisateurs-Machine : Il étudie les besoins de l'utilisateur et les intègre dans le système final.

L'expert

Il communiquer son savoir au cogniticien.

Le chef de projet

— Il est le responsable de la conduite du projet. Il fait le choix des solutions techniques, de la planification, de la répartition des tâches et de la gestion du budget.
— Il est le garant du respect du cahier des charges : l'assurance qualité, le respect des délais et la conformité spécifications/réalisations.

5.5.3 Générateurs de systèmes experts

Définition

C'est un système permettant à un expert humain (appuyé par un cogniticien) d'écrire un système expert dans son domaine d'expertise. Les synonymes **noyau**, **coque** ou **shell** sont également utilisés pour le désigner.

Les composantes d'un générateur de systèmes experts

— Une interface homme-machine permettant de mettre des règles écrites en langage pseudo-naturel sous forme de clauses ;
— Un moteur d'inférence ;
— Un ensemble d'outils logiciels pour contrôler les différentes étapes de la réalisation du système expert.
  — Editeurs spécialisés pour la gestion des connaissances ;
  — Dictionnaires des connaissances ;
  — Traceurs pour l'explication du raisonnement ;
  — Utilitaires de maintien de la cohérence et de compilation de la base de connaissances ;
  — Modules d'apprentissage automatique.

Les outils de la gestion des connaissances permettent, par exemple, le maintien de la cohérence d'une base de connaissances lors des tests de validation :

Redondance

• Si A Alors C, D (1)
• Si A Alors C, D, E (2)
La règle (1) est redondante

Contradiction

• Si A Alors C (1)
• Si A Alors ¬C, D (2)
Les règles (1) et (2) sont contradictoires

Bouclage

• Si A Alors E (1)
• Si E Alors B, F (2)
• Si B, F Alors A (3)
Les règles (1), (2), (3) sont en boucles

5.6 Exercices corrigés

5.6.1 Énoncés

Exercice 5.1

Soit la base de connaissances composée de la base de règles suivante :

R1 : Si S et T alors P
R2 : Si U alors S
R3 : Si V alors U
R4 : Si P et Q alors R

et de la base de faits (P, Q, V)

1. Quel est le chaînage le mieux adapté pour déduire R à partir de la base de connaissances ?
2. Justifiez votre réponse.

Exercice 5.2

Soit la base de connaissances suivante :

Base de règles	Base de faits
R1 : Si B et D alors A	E
R2 : Si E et B alors C	F
R3 : Si J et H alors B	
R4 : Si D et E alors B	
R5 : Si B et D alors F	
R6 : Si E et F alors D	

1. On veut prouver le fait C par chaînage arrière : quelle est la suite de règles essayées ? On indiquera si chaque règle essayée a été un succès ou un échec.
2. On veut prouver le fait C en chaînage avant : quelle est la suite des faits prouvés en admettant que l'on parcourt la base de règles dans l'ordre dans laquelle elle est écrite ?

5.6.2 Solutions

Solution de l'exercice 1

1. Le chainage le mieux adapté pour prouver R :

   Chainage avant : nous devons appliquer la suite de règles R3, R2 et R4 pour avoir en conclusion le fait R.

   Chainage arrière : l'application de la règle R4 (seule déclenchable) donne immédiatement le fait R.

2. Donc, le chainage le mieux adapté est le chainage arrière puisque le nombre de règles appliqués est 1 alors que ce nombre est 3 dans le cas de chainage avant.

Solution de l'exercice 2

1. Chainage arrière : le fait à prouver C.
   - On prend la règle R2, E ∈BF, à prouver B. Pour prouver B, nous avons deux règles R3 et R4.
   - On prend R3 : J et H. Aucune règle en contient le fait J dans sa partie conclusion, donc l'application de la règle R3 est un échec.
   - On prend alors la règle R4 : D et E ∈BF. A prouver D.
   - On prend R6 : E ∈BF et F ∈BF. Donc, l'application de R6 est un succès et D est ajouté à BF, R4 est un succès et B est ajouté à BF.

     Enfin l'application de R2 est un succès et le fait C est prouvé.
2. Chainage avant : le fait à prouver est C.

N°	Règles applicables	Règle appliquée	BF
1	R6	R6	E, F, D
2	R4	R4	E, F, D, B
3	R1, R2, R5	R1	E, F, D, B, A
4	R2, R5	R2	C prouvé

5.7 Conclusion

Méthodologie de développement

Quelle que soit la nature de l'application informatique, l'utilisation d'une méthodologie de développement et d'outils appropriés est importante. Dans le cas des systèmes experts à base de règles de production et plus généralement des SBCs, l'accent est mis principalement sur les connaissances à identifier, à acquérir et à représenter. La méthodologie recommandée repose sur une séquence d'étapes à effectuer, de l'étude d'opportunité à l'implantation d'un SBC. Un processus par raffinement graduel (itératif et incrémental) est souvent adopté.

Acquisition et Ingénierie des connaissances

L'acquisition des connaissances est certainement l'aspect le plus difficile dans la construction d'un SBC pour diverses raisons :

- la méconnaissance de la nature exacte de l'expertise à modéliser ;
- les modes de représentation de connaissances répondus ne conviennent pas toujours pour représenter certains aspects de l'expertise ;
- difficultés d'accès aux experts et sources d'expertise ; des problèmes de disponibilités, de confidentialité, ...
- la communication avec les experts est un processus long et difficile même lorsque l'expert est disponible.

L'identification, la représentation et la formalisation des connaissances constituent les tâches essentielles d'une nouvelle discipline qui est l'ingénierie des connaissances pratiquée par les cogniticiens.

Chapitre 6

Les Ontologies

6.1 Introduction

Dans le chapitre précédent nous avons vu que durant les années 80 de nombreux systèmes experts ont été développés dans des domaines variés. L'expérience de leur développement a cependant montré que la construction d'une base de connaissances était un processus qui est loin d'être simple et demande un temps considérable. D'où la question de pouvoir réutiliser et partager des bases de connaissances ou au moins des parties de ces bases.

Plusieurs pistes ont été proposées [19] :

— Réutiliser une base de connaissances pour faire une extension de la classe de problèmes pouvant être résolus par le système expert.

— Réutiliser une base de connaissances pour la conception d'un nouveau système expert réalisant une tâche différente dans le même domaine.

— En cours de raisonnement, faire appel à des compétences d'autres systèmes experts pour accéder à une expertise pointue (par exemple pour obtenir un conseil sur une pathologie rare) ou au contraire pour résoudre un problème reposant sur des connaissances de sens commun (par exemple le raisonnement temporel).

Néanmoins, ces pistes se confrontent à plusieurs obstacles [19] :

— Les systèmes experts utilisent différents langages pour représenter les connaissances tels les règles de production et les réseaux sémantiques, et différents mécanismes d'inférences comme le chaînage des règles et l'héritage de propriétés.

— En outre, indépendamment du langage de représentation utilisé, les modèles de connaissances font usage souvent des termes différents, même dans le cas de systèmes experts réalisant des tâches similaires.

— Dans certaines situation, même si les mêmes termes sont employés, il n'est pas certain que ce soit dans le même sens. Encore, ce sens n'est habituellement pas représenté explicitement dans les systèmes experts.

Dans l'objectif de réduire le travail fastidieux de l'acquisition des connaissances et permettre le partage et la réutilisation des bases de connaissances, au début des années 90, les chercheurs [1] en intelligence artificielle ont pensé à développer les ontologies pour la représentation explicite du sens.

1. Des chercheurs se sont réunis au sein du projet américain « Knowledge Sharing Effort » soutenu notamment par la DARPA (Defense Advanced Research Projects Agency).

Ce chapitre a pour objet d'introduire les fondements des ontologies. Après définition et présentation des composants d'une ontologie, il aborde la formalisation et la capacité d'expression des ontologies, les types d'ontologies ainsi que le processus de développement d'ontologie.

6.2 Qu'est-ce qu'une ontologie ?

Le terme ontologie remonte loin dans l'histoire. Son origine philosophique nous renvoie à la théorie de l'Existence : *onto* pour être et *logie* pour étude, ce qui donne "étude de l'être".

Couramment, l'ontologie c'est se mettre d'accord sur le sens des termes employés dans une organisation, une communauté, un métier, etc.

Dans ce sens, une ontologie permet d'enlever par exemple des ambigüités et de s'entendre sur la signification de termes synonymes.

Exemple 6.1: Ambigüité et synonymie

'chambre' : ambigüité	chat : synonymie
Chambre d'hôtel ?	Cat
Chambre d'écho ?	Chat
Chambre des députés ?	Greffier
Chambre d'enregistrement ?	Matou
Chambre noire ?	Minou
Chambre funéraire ?	...

Différentes acceptions du mot ontologie :
- Vocabulaire technique,
- Référentiel métier,
- Terminologie/thesaurus,
- Système de classes d'une représentation par objet : UML ?
- Base de connaissances terminologique.

6.2.1 Définitions formelles des ontologies

Plusieurs définitions formelles sont données dans la littérature. Nous allons en présenter trois connues dans en IA.

Définition 1

Pour Tom Gruber « An ontology is an explicit specification of a conceptualization. » [20]
Traduite en français par "une ontologie est une spécification explicite d'une conceptualisation."

Le terme « conceptualisation », dans la définition, fait référence à un système de concepts, autrement dit à un ensemble structuré de concepts. L'expression « spécification explicite » signifie, pour sa part, que la conceptualisation est représentée dans un langage. Ce langage peut être une langue naturelle (ex : français, anglais) ou un langage formel (ex : logique du 1er ordre, réseau sémantique).

Définition 2

La définition donnée par Riichiro Mizoguchi est en relation avec les bases de connaissances : « L'ontologie est la connaissance de base de n'importe quelle base de connaissances. » [21]

Définition 3

En ingénierie des connaissances, Jean Charlet a formulé la définition suivante : «Une ontologie est une spécification normalisée représentant les classes des objets reconnus comme existant dans un domaine. Construire une ontologie, c'est aussi décider d'une manière d'être et d'exister des objets de ce domaine.» [22]

En résumé, c'est des modèles des connaissances d'un domaine qui sont pertinentes pour une application, une tâche donnée. Conceptualisation en classes génériques, relations et règles.

Thésaurus vs. Ontologie : une comparaison est souvent faite entre les thésaurus et les ontologies.

Thésaurus :

- Le contenu d'un thésaurus est un ensemble de descripteurs et de mots-clés qui sont liés par les relations de «is-a», « synonyme » (terme préférentiel), «voir-aussi»,...
- généralement, ils sont utilisés par un agent humain (documentaliste, spécialiste) pour indexer des documents.

Ontologie :

- Le contenu d'une ontologie est une taxinomie des concepts, une taxinomie de relation, et des « rôles ».
- Elle est décrite dans un langage de représentation des connaissances et exploitée par un système informatique. Elle est dotée de possibilité de comparer et de classer des concepts, de capacité générative et d'inférences.

Utilité des ontologies : nous pouvons citer

- La communication entre les spécialistes d'un domaine ;
- Les personnes et les logiciels se comprennent ;
- L'échange entre les systèmes d'information représentés différemment ;
- L'acquisition des connaissances ;
- La réutilisation et le partage des bases de connaissances ;
- L'indexation des informations ;
- Utile pour des applications distribuées telle que le Web.

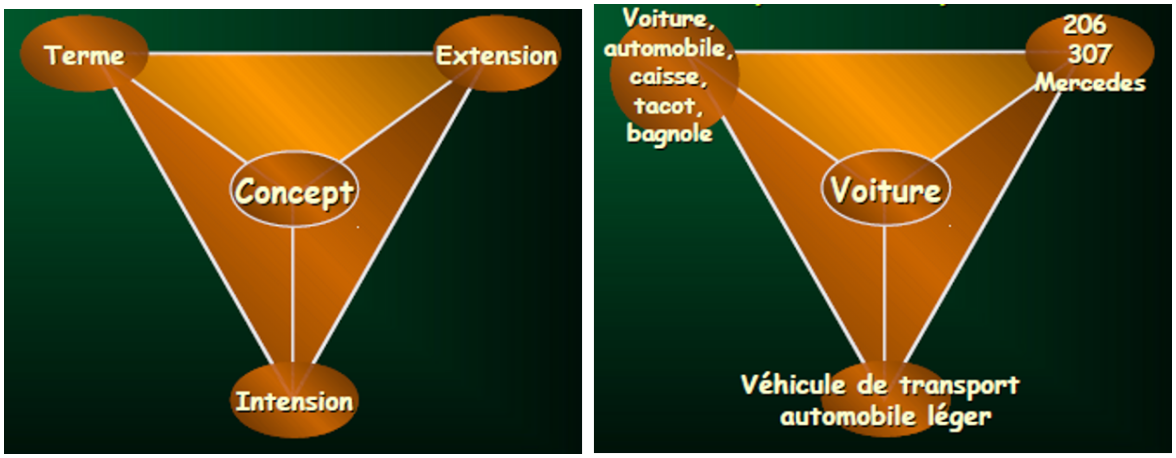
6.2.2 Composants d'une ontologie

Les composants d'une ontologie sont : les Concepts, les Relations, les Fonctions, les Axiomes et les Instances.

Les concepts : sont un ensemble de mots ou termes pertinent pour un domaine. Un concept peut se définir comme une entité composée de trois éléments :

— Le(s) terme(s) exprimant le concept en langue.
— La signification du concept, appelée également « notion » ou « intension » du concept.
— Le(s) objet(s) dénoté(s) par le concept, appelé(s) également « réalisation » ou « extension » du concept.

La Figure 6.1 montre l'utilisation du triangle sémantique pour représenter un concept. Ici, nous considérons le concept "Voiture".



(a) (b)

FIGURE 6.1 – (a) Triangle sémantique, (b) Le concept : Voiture

Les relations : traduisent les interactions existantes entre les concepts tels que composé de, fabriqué par, appartient à, ...

La subsomption « is a » est la relation centrale dans les ontologies. La Figure 6.2 montre un exemple de relations entre un ensemble de concepts.

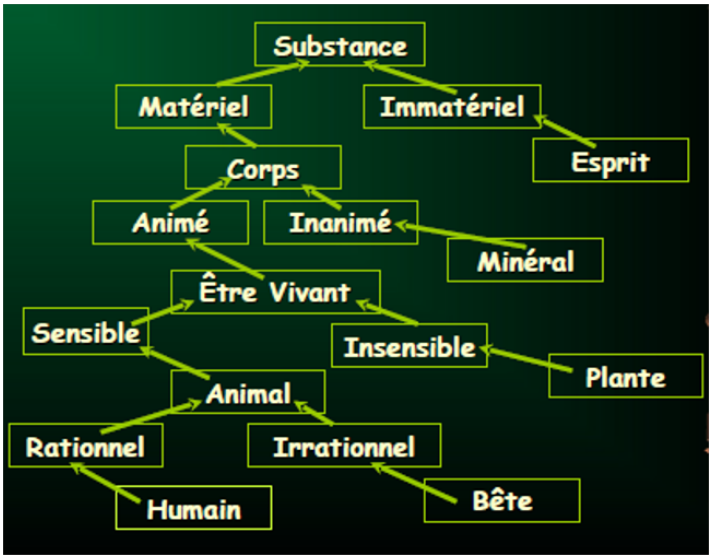


FIGURE 6.2 – Hiérarchie de concepts

Les Fonctions : sont des cas particuliers de relations dont lesquelles un élément de la relation est défini de manière unique à partir du reste des éléments. Par exemple, le prix d'une voiture d'occasion (pv) est déterminé par le modèle de la voiture (m), la date de fabrication (d) et le nombre de kilomètres parcourus (k). Il peut s'écrire mathématiquement par :

$$pv = f(m, d, k)$$

Les Axiomes : permettent de combiner des concepts, des relations et des fonctions pour définir des règles d'inférences qui peuvent intervenir, par exemple, dans la déduction, la définition des concepts et des relations, ou alors pour restreindre les valeurs des propriétés ou les arguments d'une relation.

Les Instances : ou individus pour représenter les extensions des concepts.

Un exemple d'ontologie dans le domaine médical est donné par la Figure 6.3.

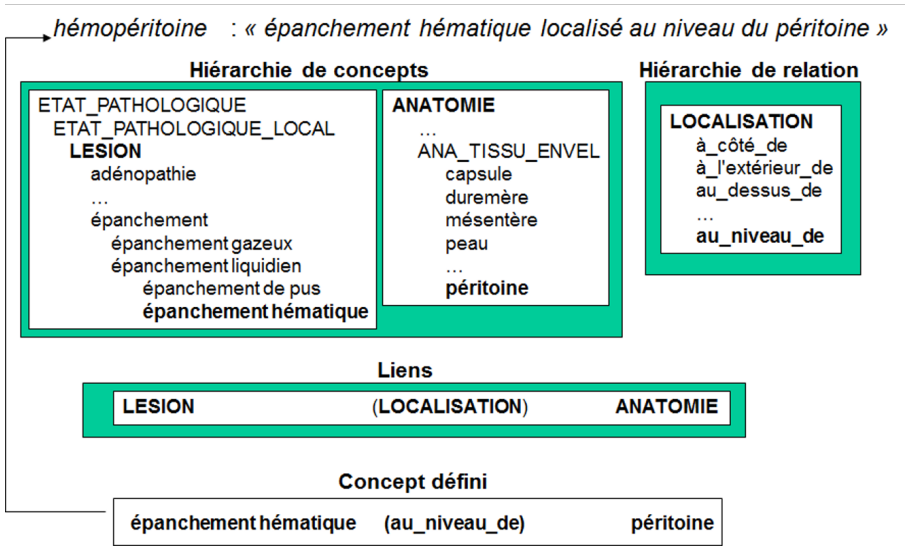


FIGURE 6.3 – Exemple d'ontologie

6.2.3 Formalisation d'ontologies

Pour la formalisation des ontologies, généralement, les formalismes logiques sont utilisés, en particulier les logiques de description.

Partie terminologique :

Les **concepts** sont organisés en hiérarchie, définis par leurs relations. Par exemple :

(DEF-CONCEPT chercheur
  (and personnel-recherche
    (ATLEAST 1 Grade)(ATMOST 1 Grade) (ALL Grade Grade)
    (ALL encadre Thésard)))

Les **contraintes** telles que

- Les relations d'exclusion entre concepts de base : EquipementCulturel ∩ equipement-Sportif ⊆ ⊥
- Le typage des rôles comme (ALL encadre Thésard) dans la définition Chercheur.

Partie déductive :

- Les relations autres que unaires et binaires :
  R1 : VolAR(villeDépart, dateDépart1, villeArrivée, dateDépart2) <= Vol(v1), lieu-Depart(v1, villeDépart), lieuArrivée (v1, villeArrivée), Vol(v2), lieuDepart(v2, villeAr-rivée), lieuArrivée (v2, villeDépart), dateDépart(v1,dateDépart1), dateDépart(v2,dateDépart2), antérieure (dateDépart, dateDépart2)
- Relations disjonctives : autant de règles que d'alternatives
  ProduitJeune(x) <= produit(x), (ATMOST 1 produitServiceAssocié)
  ProduitJeune(x) <= produit(x), produitServiceAssocié(x,y), bonMarché(y)
- Relations inverses,
- Raccourci d'enchaînement de rôles.

Expression de requêtes :

SéjourAuSoleil(s,p) <= CombinéSéjour(s), LogementAssocié(s,l), lieuDeRésidence(r), SituéDans(r,p), LieuAuSoleil(p)

Calcul de plans de requête : cela revient à vérifier la satisfiabilité, a substituer chaque terme de la requête par sa définition logique, i.e. par toutes ses spécialisations possibles $\Longrightarrow$ plusieurs requêtes disjonctives.

Ensuite, la réécriture de chaque requête conjonctive (même principe) à partir de la réécriture de chacun des atomes $\Longrightarrow$ identification des faits correspondants.

Affinement de requêtes : c'est le repérage de conflits, l'exploitation de la hiérarchie pour généraliser et le calcul de requête satisfiable à partir de requêtes insatisfiables par généralisation de concepts.

6.3 Types d'ontologies

Les différents types d'ontologies sont donnés par la Figure 6.4.

Méta-ontologies ou ontologies de représentation : servent à la formalisation des connaissances dans un système de représentation de connaissances.

Exemple 6.2: Ontologie de représentation orientée objet

Définit les termes/concepts pour la capture/spécification des conventions utilisées dans les systèmes orientés objet de représentation de connaissances : concept de classe, d'association, de fonction, de relation unaire, de relation binaire, ...

Ontologies générales/génériques (ontologies de niveau supérieur) : Elles synthétisent des notions très générales, le sens commun de la connaissance, indépendantes de tout domaine ou problème particulier. Elles classifient également les différentes catégories d'entités existant dans le monde. Ces ontologies incluent le vocabulaire lié aux : choses, évènements, espace, etc.

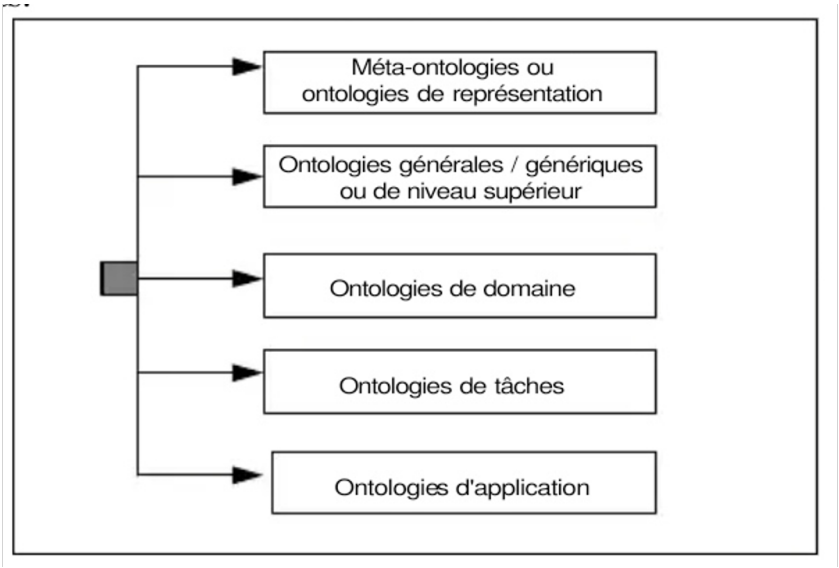


FIGURE 6.4 – Types d'ontologies

Ontologies de domaine : synthétisent des connaissances spécifiques à un domaine particulier où un processus. Ils sont considérés les concepts, les relations entre les concepts, et les règles qui les régissent. Les connaissances suivent une organisation hiérarchique (peut être sous forme de thésaurus ou taxonomies de termes, ...)

Ontologies de tâches : Elles fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles sont utilisées pour gérer des tâches spécifiques liées à la résolution de problèmes telles que les tâches de diagnostic, de planification, de configuration, etc.

Ontologies d'application : ce sont les ontologies les plus spécifiques. Elles permettent de décrire les connaissances liées à la fois à une tâche et à un domaine particulier. Elles mettent en correspondance les concepts du domaine considéré avec les concepts faisant partie de la description d'une méthode de résolution de problèmes considérée.

6.4 Caractéristiques des ontologies

La Figure 6.5 présente les caractéristiques des ontologies.

L'objet de conceptualisation : quels sont les concepts représentés. Cela renvoie aux types d'ontologies.

Niveau de détail : qui signifie que les ontologies peuvent décrire les objets de la réalité à des degrés de détails différents. On distingue :

— Granularité fine correspondant à des ontologies très détaillées, possédant ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche.

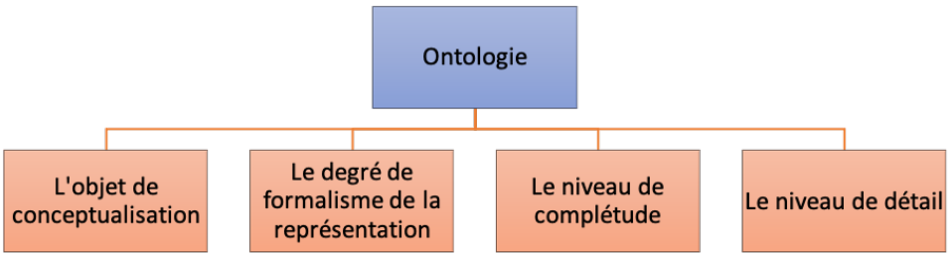


FIGURE 6.5 – Caractéristiques des ontologies

— Granularité large correspondant à un vocabulaire moins détaillé comme dans les scénarios d'utilisation spécifiques où les utilisateurs sont déjà préalablement d'accord à propos d'une conceptualisation sous-jacente.

Niveau de complétude : est ce que ces niveaux sont couverts dans l'ontologie :
— Niveau sémantique : assure que chaque concept aura un sens univoque et non contextuel associé ;
— Niveau référentiel : spécifie les objets du domaine qui peuvent être associés au concept par extension.
— Niveau opérationnel : les concepts du niveau opérationnel sont caractérisés par les opérations qu'il est possible de leur appliquer pour générer des inférences.

Degré de formalisme de représentation : les ontologies peuvent être modélisées avec des langages de différents niveaux de formalité.
— Informelles : ontologies opérationnelles dans un langage naturel (sémantique ouverte) ;
— Semi-informelles : utilisation d'un langage naturel structuré et limité ;
— Semi-formelles : langage artificiel défini formellement ;
— Formelles : utilisation d'un langage artificiel contenant une sémantique formelle, ainsi que des théorèmes et des preuves de propriétés telles la robustesse et l'exhaustivité.

6.5 Ingénierie ontologique

Dans l'ingénierie ontologique, c'est plutôt la construction de l'ontologie qui est visée. Selon Tort et al. [23], l'**ingénierie** est « Un ensemble de concepts, d'outils et de méthodes généralisables, reproductibles et évaluables ». Et l'ultime objectif de l'ingénierie ontologique est : « To provide a basis of building models of all things in which computer science is interested » [24]. Enfin, pour Guarino et al. « L'ingénierie ontologique est la branche de l'ingénierie des connaissances qui exploite les principes de l'ontologie (formelle) pour construire des ontologies » [25].

Le processus de construction d'ontologies peut être décrit selon les principes qui le gouvernent, et les méthodologies et les outils qui le soutiennent.

6.5.1 Principes

Les principes de l'ingénierie ontologique sont [26, 27, 28] :

Clarté et objectivité : la signification des termes avec des définitions objectives avec la documentation en langage naturel.

Complétude : une définition avec des conditions nécessaires et suffisantes et non partielle.

Cohérence : toutes les inférences soient conforment aux définitions.

Extensibilité monotone maximale : ajouter de nouveaux termes génériques ou spécifiques sans entrainer la révision des définitions existantes.

Engagments ontologiques minimaux : fixer un minimum d'objectifs -spécifier le moins possible la signification de ses termes- concernant le monde à modéliser pour pouvoir instancier l'ontologie en fonction des besoins.

Principe de dsitinction ontologique : les classes devraient être disjointes. A chaque fois que l'on peut identifier et isoler un noyau de propriétés considérées comme invariables pour une classe (critère d'identité), il faut créer une nouvelle classe de concepts.

Diversification des hiérarchies : Il faut diversifier les taxinomies pour augmenter la puissance fournie par les mécanismes d'héritage. Ceci signifie qu'il faut subdiviser les taxonomies dès que cela est possible. Ce principe est adopté pour augmenter la puissance fournie par les inférences.

Modularité : minimiser les couplages entre les modules.

Distance sémantique minimale : Il faut minimiser la distance sémantique entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous-classes d'une classe, et devraient être définis en utilisant les mêmes primitives, étant donné que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie.

Nomaliser les termes : Il faut normaliser les termes chaque fois que c'est possible. Ceci permet d'assurer une uniformité dans la façon de libeller les termes.

Une multitude de méthodologies ont été recensées [29, 26], néanmoins il n'existe pas encore de consensus en matière de normes de construction d'ontologies. Ceci relève beaucoup plus du savoir-faire que de l'ingénierie.

6.5.2 Cycle de vie d'une ontologie

Le cycle de vie proposé par [30] donné par la Figure 6.6 comprend sept étapes. La première étape est la **détection et la spécification** des besoins qui permet de circonscrire le domaine de connaissances. La deuxième étape est la conception qui se subdivise en trois phases :

Normalisation sémantique qui a pour objectif de rendre explicite le sens des expressions linguistiques. Il s'agit d'en faire des primitives du domaine, i.e. d'identifier les notions élémentaires à partir desquelles l'ensemble des connaissances du domaine sont construites.

Formalisation cette seconde phase permet de formaliser les connaissances du domaine à représenter. Il s'agit de définir des concepts, et non plus des notions, qui pourront servir comme primitives dans un langage formel de représentation des connaissances. Cette phase permet également de formaliser les relations qui existent entre les concepts en définissant leur arités et les ensembles d'extensions de concepts qu'elles relient.

Opérationnalisation cette dernière phase s'attache à informatiser l'ontologie référentielle dans un langage opérationnel de représentation des connaissances (comme les graphes conceptuels ou les logiques de description). Un système informatique peut manipuler les concepts en suivant les règles et les opérations qu'il peut leur associer.

Suit ensuite, une étape de **déploiement et diffusion**, une étape d'**utilisation**, une étape d'**évaluation**, et une sixième étape consacrée à l'**évolution et maintenance** du modèle. Après chaque utilisation significative, l'ontologie et les besoins doivent être réévalués et l'ontologie peut être étendue et, si besoin, en partie reconstruite. Enfin, la **validation** du modèle de connaissances est au centre du processus et se fait de manière itérative. Elle permet de vérifier la validité de tout changement fait dans la structure de l'ontologie afin d'assurer une cohérence d'ensemble.

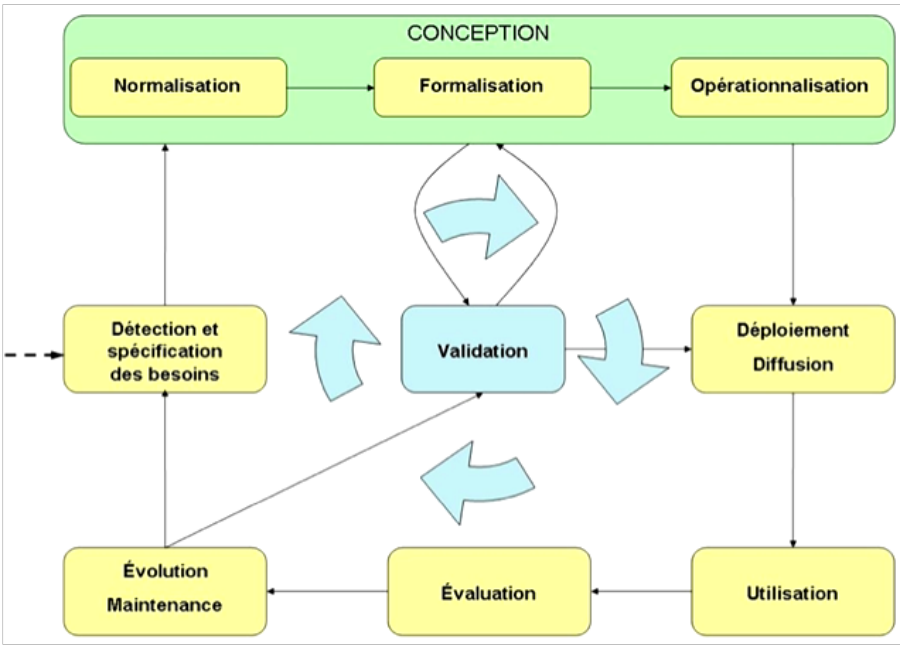


FIGURE 6.6 – Cycle de vie d'un ontologie.

6.5.3 Méthodologies de l'ingénierie ontologique

Les méthodologies recensées permettent la construction d'ontologies 1) à partir de zéro, 2) par intégration ou fusion avec d'autres ontologies, 3) par re-ingénierie, 4) par construction collaborative, ainsi que 5) par l'évaluation des ontologies construites. La Figure 6.7 [31] montre comment ces différentes méthodologies s'intègrent et parfois se complètent à l'intérieur d'un cycle de vie de construction, d'évaluation et de maintenance des ontologies.

6.5.3.1 Méthodologies de construction à partir de zéro

Différentes méthodologies de développement à partir de zéro ont été proposées. Nous pouvons citer les travaux de Uschold et Gruninger (1996), Methontology (1997), les travaux de Guarino et Welty (2000), ARCHONTE (2000).

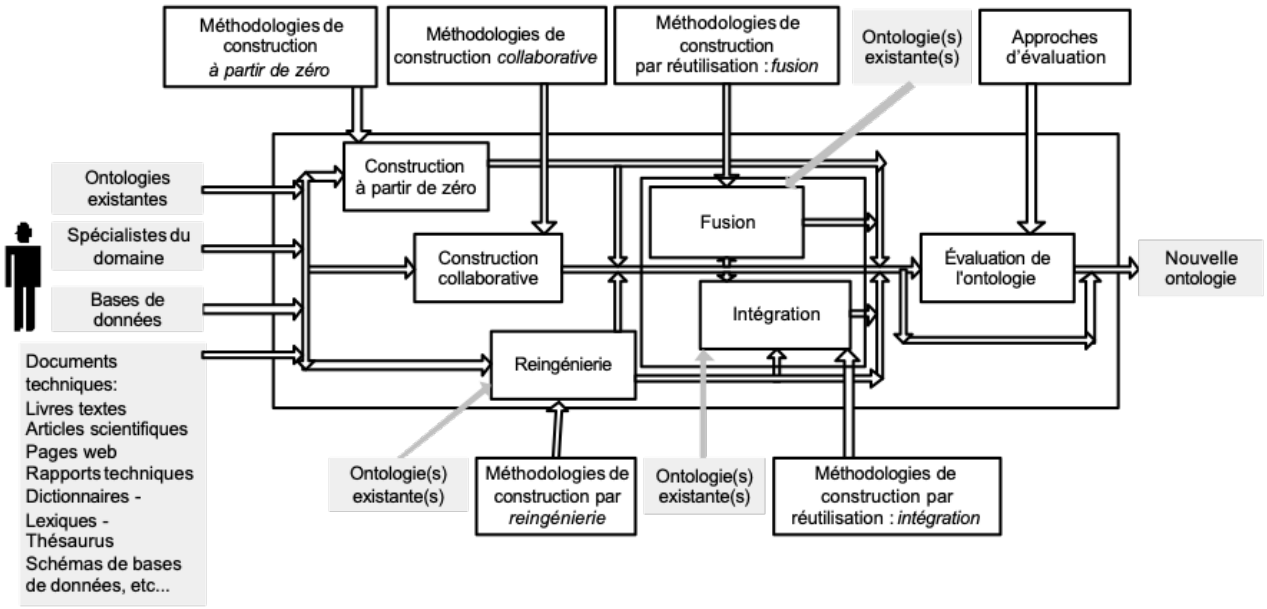


FIGURE 6.7 – Analyse comparative des méthodologies

Exemple 6.3: Methontology [32]

Elle est développée par l'équipe du LAI de l'université de Madrid. Elle prend en charge le cycle de vie de l'ontologie selon les étapes suivantes :

1. Glossaire des termes
2. Taxinomie de concepts
3. Diagramme des relations binaires
4. Définir les attributs des concepts
5. Décrire les relations
6. Décrire les attributs d'instances
7. Décrire les attributs de classes
8. Décrire les constantes
9. Décrire les axiomes formels
10. Décrire les contraintes sur les valeurs des attributs

Vue l'absence d'assise théorique pour les méthodologies à partir de zéro, la construction d'ontologies est plus un « art » qu'un processus d'ingénierie classique.

6.5.3.2 Méthodologies de construction par ré-ingénierie

Methontology permet également la ré-ingénierie ontologique comme l'illustre la Figure 6.8.

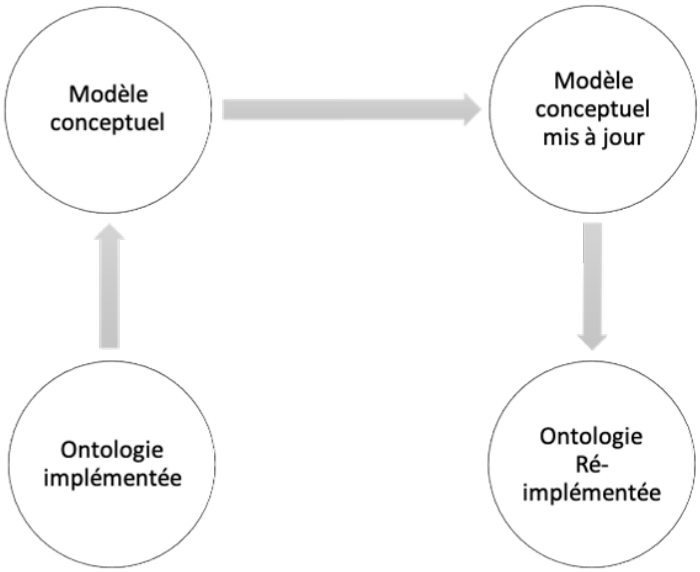
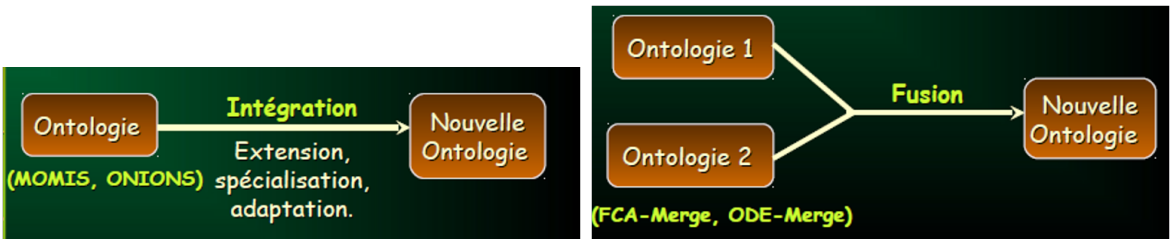


FIGURE 6.8 – Ré-ingénierie ontologique

6.5.3.3 Méthodologies de construction par réutilisation d'ontologies existantes

La construction par réutilisation des ontologies existantes peut se faire par intégration (extension, spécialisation ou adaptation par exemple) ou par fusion à l'aide d'opérateurs tel que le montre les Figure 6.9 (a) et (b), respectivement.



(a) (b)

FIGURE 6.9 – (a) construction par intégration, (b) construction par fusion

Ces méthodologies ont des limites, l'intervention de l'ontologiste à chaque étape est inévitable, les décisions importantes lui reviennent, surtout dans les cas d'ontologies lourdes.

6.5.3.4 Méthodologies pour l'évaluation des ontologies

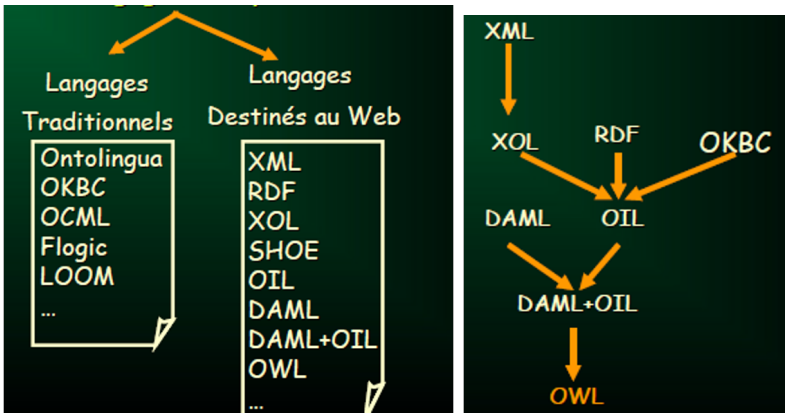
A travers l'évaluation des ontologies, nous pouvons également obtenir de nouvelles ontologies et cela à l'aide des outils (comme OntoAnalyser, ontoGenerator ou encore OntoClean) qui détectent les dysfonctionnements, les incohérences ou les erreurs de langages. Cependant, il y a besoin de définir des critères standards d'évaluation des ontologies.

6.5.4 Langages formels pour la représentation des ontologies

Plusieurs langages d'implémentation d'ontologies ont été identifiés dans la littérature. Ces langages peuvent être classifiés en deux catégories (Figure 6.10(a)) :

— Les langages d'ontologie basés sur l'intelligence artificielle. Ceux bâtis sur la logique du premier ordre comme KIF, ceux qui combinent les frames avec la logique du premier ordre tels que Ontolingua, FLogic et OCML, ceux basés sur la logique de description à l'image de LOOM.

— Les langages d'ontologie basés sur le Web afin d'exploiter les caractéristiques du Web. Leur syntaxe est basée sur des langages de balisage tels que HTML et XML. Les plus importants sont : SHOE (Simple HTML Ontology Extension), XOL (Ontology Exchange Langage), RDF, RDF Schema, OIL (Ontology Inference Layer), DAML+OIL (DARPA Agent Markup Langage + Ontology Inference Layer) et OWL (Web Ontology Langage). La Figure 6.10(b) illustre les relations entre ces langages.



(a) (b)

FIGURE 6.10 – (a) Les langages d'ontologie, (b) Les relations entre les langages de balisage.

6.5.5 Environnements de développement des ontologies

Puisque la construction des ontologies n'est pas une tâche simple à réaliser, divers environnements ou outils ont été développés pour donner un soutien technologique dans la plupart des activités du cycle de vie de l'ontologie tels que Protégé [33], Hozo [34], NeOn Toolkit [35], ...

Exemple 6.4: Protégé

Une plateforme open source développée par l'université de Stanford vers 1995. Elle est dotée d'une bibliothèque de modules (plugins) qui peuvent ajouter plus de fonctionnalités à la plateforme. Protégé permet de réaliser plusieurs tâches telles que la création, le chargement, la visualisation, et la sauvegarde d'ontologies en plusieurs formats (par exemple RDF et OWL2). Cette plateforme est toujours soutenue par l'université de Stanford, la version 5.6.5 a été publiée en 2024.

Cependant, ils souffrent de certains manques tels que l'absence de prise en charge de toutes les étapes du développement des ontologies et le manque d'interopérabilité entre les plateformes.

6.6 Exercices corrigés

6.6.1 Énoncés

Exercice 6.1

Considérant les concepts suivants : document, support de cours, document pdf, document pédagogique, page personnelle, document électronique, page Web.

— En utilisant la notion courante des ces concepts, construire une taxinomie reposant sur des liens de subsomption.

Exercice 6.2

1. Parmi les objectifs visés par la construction d'ontologies est l'interopérabilité entre applications pour le partage de connaissances :
   — au niveau sémantique
   — au niveau syntaxique
   — autre
2. Une ontologie peut être vue, ontologiquement parlant, comme :
   — un objet conceptuel
   — un objet syntaxique
   — autre
3. Deux concepts ayant des notions différentes et étant exprimés par des termes différents ont pour réalisation :
   — un même ensemble d'objets
   — des ensembles d'objets différents
   — autre

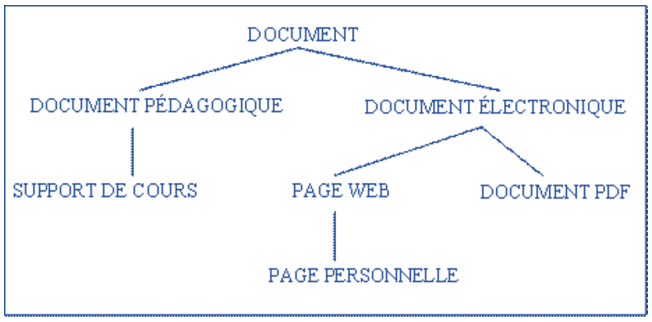
Exercice 6.3

Quels sont les types d'ontologies ? Lesquels sont réutilisables ?

6.6.2 Solutions

Solution de l'exercice 1

La taxinomie des concepts en utilisant les liens de subsomption est donnée par la figure. **Remarque** : le fait que deux concepts spécialisent un même concept ne signifie pas que l'intersection de leur extension est vide. Par exemple, il existe des documents pédagogiques qui sont également des documents électroniques.



Solution de l'exercice 2

1. L'interopérabilité entre applications vise le partage de connaissances **au niveau sémantique**. Il s'agit de partager le contenu de bases de connaissances indépendamment des langages de représentation utilisés par les applications.
2. Une ontologie peut être vue, ontologiquement parlant, comme **un objet syntaxique** sur un support matériel (papier ou électronique) ce qui permet de les échanger.
3. Deux concepts ayant des notions différentes et étant exprimés par des termes différents ont pour réalisation **autre**. En général, de tels concepts ont pour extension des ensembles d'objets différents, mais il existe des exceptions. Par exemple, les concepts génériques POLYGONE À TROIS CÔTÉS et POLYGONE À TROIS ANGLES ont pour réalisation commune, l'ensemble des triangles. De tels concepts sont dits « co-référentiels ».

Solution de l'exercice 3

Les types d'ontologies :
- de représentation (méta-ontologies) : réutilisables
- génériques : réutilisables
- de domaine : réutilisables
- d'application : non réutilisables
- de tâches : non réutilisables

6.7 Conclusion

Nous pouvons conclure par dire que les ontologies ont beaucoup apporté pour le traitement des connaissances comme les fondements ontologiques, la formalisation, la capacité de raisonnement et les standards pour la représentation des connaissances.

Cependant, l'évolution de l'ingénierie ontologique est freinée par les aspects suivants : l'hétérogénéité des méthodes, la diversité des outils et langages de construction des ontologies (absence de normes) et l'absence d'assise théorique des méthodologies de construction des ontologies. Ajouter à cela, l'absence de normes en matière d'évaluation des ontologies.

Conclusion

Ce document a traité une des branches de l'IA qui est l'ingénierie des connaissances qui consiste en l'identification, la représentation et la formalisation des connaissances. Dans un premier temps, il a défini la notion de la connaissance et les fondements de l'IC. Dans un deuxième temps, il a présenté les divers modes de représentation de connaissances. Ensuite, il a abordé les systèmes à base de connaissance en mettant l'accent sur les systèmes experts à base de règles de production. Enfin, il a introduit l'ingénierie ontologique.

Nous pouvons retenir que le développement de systèmes de l'IC passe par quatre étapes principales qui sont : le recueillement des besoins qui fait appel à l'acquisition des connaissances dans diverses sources, en particulier des experts ; la modélisation conceptuelle qui permet de modéliser et de représenter la connaissance ; l'implémentation pour avoir une base de connaissances a utiliser lors du raisonnement et enfin la validation par l'utilisateur final. De plus, afin que le système soit qualifié d'intelligent, il doit intégrer l'apprentissage automatique pour assurer son évolution.

Nous avons vu, en outre, que les formalismes graphiques de représentation connaissances offrent une certaine facilité et lisibilité mais manquent de rigueur comparativement aux formalismes logiques. Ces derniers grâce à l'introduction de nouveaux opérateurs et la graduation des valeurs de vérités permettent aussi la représentation des informations imparfaites dans les logiques non classiques. A noter qu'à travers les logiques de description, nous avons conclu que la richesse en expressivité conduit à plus de complexité et les mécanismes de « calcul » ne permettent pas des expressions de requêtes toujours « simples ». Sans oublier de souligner que des langages de représentation de connaissances ont été crées pour permettre d'inter-changer des données entre des systèmes informatiques hétérogènes.

Enfin, ce document peut être utilisé par les étudiants, les enseignants ou tout professionnel cherchant à manipuler la connaissance ou à concevoir des systèmes à base de connaissances ou des ontologies. De plus, la lecture de ce document leurs permettra de distinguer les types d'ontologies et le choix de celui à développer afin de répondre aux besoins de l'application visée, notamment que les ontologies de haut niveau d'abstraction peuvent être instanciées en systèmes à base de connaissances.

Bibliographie

[1] E. A. Feigenbaum et al., "The art of artificial intelligence : Themes and case studies of knowledge engineering," 1977.

[2] J. Charlet, "L'ingénierie des connaissances : développements, résultats et perspectives pour la gestion des connaissances médicales," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2002.

[3] J. Charlet, M. Zacklad, G. Kassel, and D. Bourigault, "Ingénierie des connaissances : recherches et perspectives," Ingénierie des connaissances, évolutions récentes et nouveaux défis. Eyrolles, 2000.

[4] (2017) Projet bourbakem, elément n°12, l'ingénierie des connaissances. [En ligne]. Disponible : https://www.agecso.com/wp/bourbakem/bourbakem12/ (Consulté le 13-apr-2025).

[5] R. Feldman and J. Sanger, The text mining handbook : advanced approaches in analyzing unstructured data. Cambridge university press, 2007.

[6] K. J. Cios and L. A. Kurgan, "Trends in data mining and knowledge discovery," in Advanced techniques in knowledge discovery and data mining. Springer, 2005, pp. 1–26.

[7] G. Schreiber, B. Wielinga, H. Akkermans, W. Van de Velde, and A. Anjewierden, "Cml : The commonkads conceptual modelling language," in A Future for Knowledge Acquisition : 8th European Knowledge Acquisition Workshop, EKAW'94 Hoegaarden, Belgium, September 26–29, 1994 Proceedings 8. Springer, 1994, pp. 1–25.

[8] A. Cornuéjols and L. Miclet, Apprentissage artificiel : concepts et algorithmes. Editions Eyrolles, 2011.

[9] H. A. Simon, "Why should machines learn ?" in Machine learning. Elsevier, 1983, pp. 25–37.

[10] M. E. Webb, A. Fluck, J. Magenheim, J. Malyn-Smith, J. Waters, M. Deschênes, and J. Zagami, "Machine learning for human learners : opportunities, issues, tensions and threats," Educational Technology Research and Development, vol. 69, no. 4, pp. 2109–2130, 2021.

[11] M. MINSKY, "A framework for representing knowledge," The psychology of computer vision, pp. 211–277, 1975.

[12] I. Stoica, "Script theory," Research Starters : Education (Online Edition), 2015.

[13] J. F. Sowa, "Conceptual graphs for representing conceptual structures," Conceptual Structures in Practice, pp. 119–154, 2016.

[14] J. F. Sowa, "Conceptual graphs," Foundations of artificial intelligence, vol. 3, pp. 213–237, 2008.

[15] L. Cervoni and J. Brasseur, "Prolog, 50 ans d'histoire de l'intelligence artificielle."

[16] P. Marquis, O. Papini, and H. Prade, Représentation des connaissances et formalisation des raisonnements. Cepadues Editions, 2014.

[17] R. Caferra, Logique pour l'informatique et pour l'intelligence artificielle. Hermés science, 2010.

[18] M. R. Genesereth, R. E. Fikes et al., "Knowledge interchange format-version 3.0 : reference manual," 1992.

[19] Notion d'ontologie. [En ligne]. Disponible : https://perso.liris.cnrs.fr/alain.mille/enseignements/DEA-ECD/ontologies/notion_ontologie.htm (Consulté le 09-apr-2025).

[20] T. R. Gruber, "A translation approach to portable ontology specifications," Knowledge acquisition, vol. 5, no. 2, pp. 199–220, 1993.

[21] R. Mizoguchi, "Part 1 : Introduction to ontological engineering," New generation computing, vol. 21, no. 4, pp. 365–384, 2003.

[22] N. Aussenac-Gilles, M. BAZIZ, and N. HERNANDEZ, "Ontologies pour la recherche d'information : importance de la dimension terminologique," Terminologie et accès à l'information spécialisée, éd. par Widad Mustapha El Hadi, Techniques et traités des sciences et techniques de l'information, Hermès, pp. 211–234, 2006.

[23] C. Reynaud and F. Tort, "Diriger la réutilisation de composants à l'aide d'ontologies," Ingénierie des connaissances : évolutions récentes et nouveaux défis, pp. 59–74, 2000.

[24] A. Banu and A. Ameen, "Transforming medical data into ontologies for improving semantic interoperability," in Emerging IT/ICT and AI Technologies Affecting Society. Springer, 2022, pp. 71–85.

[25] N. Guarino and P. Giaretta, "Ontologies and knowledge bases," Towards very large knowledge bases, pp. 1–2, 1995.

[26] W. Bannour, "Une approche d'aide à la décision pour la réponse à une crise," Ph.D. dissertation, Ecole Nationale des Sciences de l'Informatique Tunis (Tunisie), 2022.

[27] L. MADI, "D'une modélisation dans la conception d'un système d'information vers une architecture orientée services (soa)," Ph.D. dissertation, 2012.

[28] V. Psyché, "Rôle des ontologies en ingénierie des eiah : cas d'un système d'assistance au design pédagogique," Ph.D. dissertation, Université du Québec à Montréal, 2007.

[29] H. Taktak, "Approche orientée services sémantiques pour l'intégration de données massives et multi-sources pour la détection des catastrophes naturelles," Ph.D. dissertation, Université Jean Moulin lyon 3 ; Faculté des sciences Economiques et de ..., 2024.

[30] A. Baneyx, "Construire une ontologie de la pneumologie aspects théoriques, modèles et expérimentations," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2007.

[31] V. Psyché, O. Mendes, and J. Bourdeau, "Apport de l'ingénierie ontologique aux environnements de formation à distance," STICEF (Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation), vol. 10, pp. 89–126, 2003.

[32] O. Corcho, M. Fernández-López, A. Gómez-Pérez, and A. López-Cima, "Building legal ontologies with methontology and webode," Law and the semantic web : legal ontologies, methodologies, legal information retrieval, and applications, pp. 142–157, 2005.

[33] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of protégé : an environment for knowledge-based systems development," International Journal of Human-computer studies, vol. 58, no. 1, pp. 89–123, 2003.

[34] R. Mizoguchi, E. Sunagawa, K. Kozaki, and Y. Kitamura, "The model of roles within an ontology development tool : Hozo," Applied Ontology, vol. 2, no. 2, pp. 159–179, 2007.

[35] P. Haase, H. Lewen, and R. Studer, "The neon ontology engineering toolkit," 2008.