Université Ahmed Draia Adrar

Faculté des Sciences de la Matière, Mathématiques et Informatique

Département de Mathématiques et Informatique

Modélisation à Base d'Objets et Représentation des Connaissances

POLYCOPIE DE COURS

Support de Cours destiné aux étudiants de niveau **deuxième année Master**

Spécialité : Systèmes Intelligents

Semestre : S3

Dr. KADDI Mohammed

Maître de conférences classe A

Université d'Adrar

Année 2024

Avant-Propos

Ce polycopié est rédigé à l'intention des étudiants de deuxième année Master ; Spécialité Systèmes Intelligents ; Semestre 3. Il constitue un manuel de cours de la matière " Modélisation à Base d'Objets et Représentation des Connaissances " pour présenter les notions d'algorithme et de structure de données. Dans cette matière, l'étudiant apprend les principes de modélisation à base d'objet. Les étudiants nécessitent pré requis sur les notions de modélisation et simulation.

Le contenu de la matière est :

Chapitre 1 : Introduction à la représentation des connaissances

Chapitre 2 : Langage de Modélisation par Objets Typés MOT

Chapitre 3 : Réseaux sémantiques

Chapitre 4 : Frames et classification

Chapitre 5 : Logique de description

Chapitre 6 : Graphes Conceptuels et Graphes de Sowa

Chapitre 7 : Formalismes basés sur la logique

Chapitre 8 : Web sémantique, Ontologies et langages de description (RDF)

Le crédit, le coefficient et le mode d'évaluation de cette matière sont :

Crédit : 6

Coefficients :3

Mode d'évaluation : Ecrit et Oral

Sommaire

Introduction Générale

Introduction générale :

On trouve la notion de représentation utilisée depuis l'Antiquité ; mais aujourd'hui ce concept trouve ses sources au XVII° et en particulier dans la philosophie de René Descartes (1590-1650). Cette notion a des usages multiples (anthropologues, psychologues, sociologues, en informatique, etc.) On peut s'interroger sur la nature et la fonction que l'on associe à cette notion. Cependant est-ce que ce concept renvoie à un même objet pour tous. De manière analogique, c'est associé à l'idée de croyances, de modes ou de systèmes de pensée.

Cette notion très générique qui a pour corrélat les notions de connaissances, d'idée, d'esprit, de langage, de pensée. Le terme de représenter vient du latin *representare*, signifiant rendre pensée à nouveau. Si l'on prend un dictionnaire, globalement on voit que c'est par quoi un objet est présent à l'esprit. Par exemple un usage en psychologie relèverait de la perception, une image mentale qui se rapporte à un contenu, une situation, une scène. Il s'agit de la conception ordinaire, triviale. Cela méconnait un aspect important, caractère dynamique qui renvoie aussi au processus de l'activité cognitive, du fait de se représenter quelque chose. Cela renvoie à un certain type d'opération par lequel l'esprit fait venir devant lui quelque chose qui n'est pas réellement présent mais qui va le devenir grâce à un travail cognitif, pour se donner une image, un schéma de quelque chose. Un outil relevant fondamentalement de notre parcours pouvant nous permettre ce que sous-entend l'idée de représentation serait la modélisation. Lorsque l'on modélise on simplifie, on généralise. Pour la représentation on passe par un médium, un élément qui s'interpose entre la réalité et le concept de connaître. Cela sous-entend que nous avons une connaissance indirecte du monde.

Il y a importance dans l'idée de médiation par rapport à l'être humain qui cherche à se représenter la réalité. Cela pose le problème du caractère non intuitif, non évident, non manifeste de la connaissance directe des choses. Nous n'accédons pas tel quel à la connaissance. Un modèle est un médium entre quelque chose qui cherche à connaître et la chose qu'il y a à connaître. Cela relève d'une construction cognitive qui produit une représentation qui vient se substituer à la réalité directe, d'où l'idée de modélisation. La notion de représentation peut renvoyer à une relation (x est représenté par y), c'est un intermédiaire. Cela renvoie au processus (x se substitue à y par leur relation) ; et à une entité qui peut être un contenu, un énoncé, un état de la réalité, une perception, qui représente. La notion de représentation consiste en un accès à la connaissance ; quel que soit le type de conception *représentationnaliste* de la connaissance, celle-ci consiste à analyser le rapport entre une représentation et un objet (c'est-à-dire le monde tel qu'il est indépendamment de notre capacité à le

connaître). L'idée classique de représentation suppose une connaissance de l'être sur le connaître. Cela suppose une connaissance médiée, il s'agit d'un double de la réalité. De manière commune on considère qu'il y a des choses extérieures à nous, et on doit les trouver. Il y a antécédences des choses sur notre capacité à les connaître. Le rapport de connaissance pourrait-il être inversé ? On peut dire qu'il y a une antécédence du connaître sur l'être.

La représentation des connaissances désigne un ensemble d'outils et de procédés destinés d'une part à représenter et d'autre part à organiser le savoir humain pour l'utiliser et le partager. Les connaissances n'ont jamais été, et ne sont toujours pas, systématiquement représentées par des mots et des phrases.

Les outils classiques (non électroniques) de représentation des connaissances sont les taxonomies ou classifications, qui permettent d'organiser les connaissances sur les objets du monde, et les thésaurus documentaires utilisés en indexation documentaire.

Des outils plus formels et permettant de représenter des connaissances complexes sont par exemple les graphes conceptuels ou les réseaux sémantiques.

Dans le domaine des nouvelles technologies, la représentation formelle des connaissances s'est développée dans le domaine de l'intelligence artificielle. Dans une représentation formelle, les connaissances sont représentées par des objets logiques reliés par des propriétés, axiomes et règles. Ce type de représentation est utilisé dans les systèmes experts.

Le développement du Web, et en particulier la perspective du Web sémantique a renouvelé le domaine en introduisant le terme controversé d'ontologie. Un certain nombre de langages ont été développés dans cette perspective, comme les standards RDFS, SKOS et OWL du W3C ou la norme ISO Topic Maps.

Toute connaissance se base sur un schéma cognitif propre à chaque individu. Un des enjeux de l'Intelligence Artificielle (IA) consiste alors à modéliser un domaine et à implémenter cette cognition sous une forme manipulable aussi bien par les humains que par les machines. En d'autres termes, il s'agit d'implémenter sous une forme pragmatique et opérationnelle, les structures et les mécanismes de connaissances manipulés par les individus. A cet effet, différents formalismes de représentation ont été inventés. Ces derniers varient en fonction du niveau d'expressivité, de la rigueur et de la sémantique offerts.

Dans l'histoire de ces formalismes, deux grandes périodes peuvent être distinguées. Tout d'abord avant les années quatre-vingt où l'on distinguait les formalismes basés sur la logique (descriptive, de 1er ordre, etc.) et les représentations non logiques (les réseaux sémantiques, les schémas, etc.). Puis ensuite une seconde phase, qualifiée d'hybride, car mélangeant différents formalismes de représentation comme les

schémas avec de la logique. Cela a conduit à l'apparition de nouveaux paradigmes telle que la logique de description qui est majoritairement utilisée pour la description d'ontologie.

Choisir la bonne Interprétation peut être difficile, le choix de la représentation est lui-même problématique, il est important d'avoir une représentation simple, mais cela peut interdire certains types de raisonnements, la représentation dépend beaucoup du but poursuivi même dans des situations simples, il est facile d'omettre la représentation d'informations nécessaires parce qu'elles paraissent.

Dans ce document, nous présentons les différents langages et formalismes de représentation des connaissances : le langage de modélisation par objets typés, le réseau sémantique, le frame, la logique de description, le graphe conceptuel et le graphe de Sowa, les formalismes basés sur la logique, le web sémantique, ontologies et langage de description.

Chapitre 01

Introduction à la représentation des connaissances

Introduction :

Il importe, d'entrée de jeu, **d'établir la distinction entre les concepts d'information et de connaissance**. En informatique, les premiers systèmes traitaient uniquement les données numériques. Ensuite, une seconde vague de systèmes s'est intéressée aux banques d'informations et de documents. Au cours des trois dernières décennies, une nouvelle génération de systèmes informatiques est arrivée à traiter des connaissances de plus haut niveau. Les systèmes à base de connaissances peuvent ainsi représenter et traiter des principes et des règles de décision, des taxonomies, des théories, des processus et des méthodes mémorisées dans l'ordinateur. En un mot, ils sont capables d'aider l'usager à accomplir des tâches de façon plus intelligente. Ce niveau « cognitif » est encore trop peu répandu dans les systèmes d'information et dans la pratique des individus et des organisations.

Par « **information** », nous entendons toutes les données extérieures aux personnes, communiquées oralement par d'autres ou médiatisées dans des matériels sur divers soutiens numériques, imprimés ou analogiques.

Par « **connaissance** », nous entendons le résultat de toute construction mentale effectuée par un individu à partir d'informations ou autres stimuli.

L'apprentissage par un individu consiste à transformer des informations en connaissance.

1. Concept-clé : Données-> Information -> Connaissance -> Sagesse :

La connaissance sur les services passe par les étapes classiques suivantes :

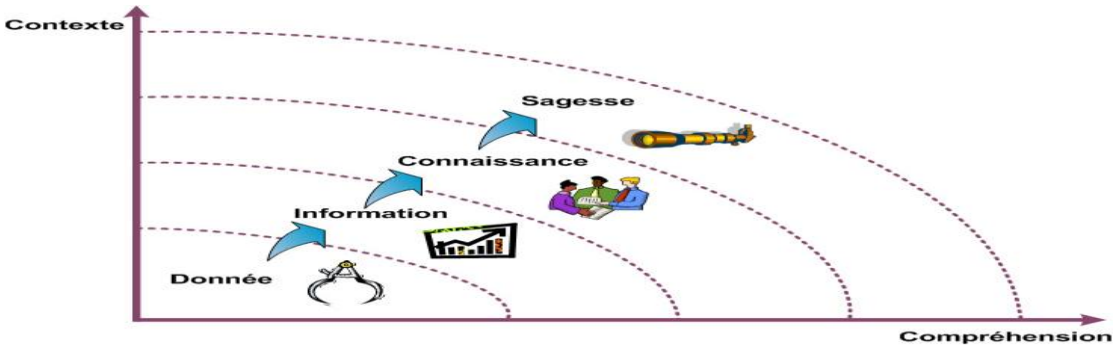


Figure 1.1 : Etapes classiques de la connaissance sur les services.

1.1 Donnée :

- Elément fondamental servant de base à un raisonnement, à une recherche.

- Une donnée est un élément brut, qui n'a pas encore été interprétée, mis en contexte.

1.2 Information :

- Elément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué.
- Renseignement, fait qui apporte des renseignements nouveaux.
- Représentation conventionnelle d'une information sous une forme convenant à son traitement par ordinateur.
- Une information permet à un responsable opérationnel de prendre une décision sur une action à mener.

→Et c'est là toute la différence entre une information et une donnée. En effet, **une information est par définition une donnée interprétée**. En d'autres termes, la mise en contexte d'une donnée crée de la valeur ajoutée pour constituer une information.

1.3 Connaissance (*Knowledge*):

- La connaissance est l'élément de base dans un Système Expert (SE), car elle permet à ce dernier de résoudre les problèmes qui lui sont posés. C'est d'ailleurs pour cela que les SE sont aussi appelés "systèmes à base de connaissances" = "knowledge based systems" (KBS).
- Ce que l'on a appris par l'étude ou la pratique. (ex : En cas d'alerte, ...).
- Les connaissance sont des règles utilisant les données pour en déduire d'autres.
- La connaissance inclut la généralisation et l'abstraction d'un grand volume de données.
- La connaissance est le résultat d'une réflexion sur les informations analysées en se basant sur :

- Ses expériences, ses idées, ses valeurs, les avis d'autres personnes consultées pour l'occasion
- Sa propre expertise et celle de ses pairs.

→On peut considérer **la connaissance comme une information comprise**, c'est-à-dire assimilée et utilisée, qui permet d'aboutir à une action. « Nonaka et Takeuchi (deux experts du knowledge management) différencient deux formes de connaissance : une connaissance tacite et une connaissance explicite.

- La connaissance tacite : c'est la connaissance que possèdent les individus. **Elle n'est pas formalisée et difficilement transmissible**. Ce sont les compétences, les expériences, l'intuition, les secrets de métiers, les tours de main qu'un individu a acquis et échangés lors d'échanges internes et externes à l'entreprise. La connaissance tacite se transmet par imitation et imprégnation. On le sait sans le savoir. On met en œuvre des pratiques sans vraiment s'en rendre compte.

- La connaissance explicite : **C'est la connaissance formalisée et transmissible sous forme de documents réutilisables.** Ce sont les informations concernant les processus, les projets, les clients, les fournisseurs, etc. La connaissance explicite se transmet par des documents formalisés et normalisés.

→L'utilisation des connaissances dans le contexte d'activité s'appelle du savoir-faire ou des compétences.

La conceptualisation de ces savoir-faire et compétences et l'expérience de ces situations de travail forment ce que l'on appelle l'expertise.

Qu'est-ce qu'un expert ? **L'expertise consiste non pas à énoncer des vérités mais à mesurer des écarts par rapport à des normes conventionnellement admises.**

1.4. Sagesse :

Elle permet de prendre des décisions à long terme et des décisions stratégiques pour l'organisation informatique.

2. Interaction entre donnée, information et connaissance :

L'étude de l'unité informationnelle nous a permis de définir les concepts de donnée, d'information et de connaissance.

La connaissance est l'élément central de l'unité informationnelle. Elle fournit les éléments nécessaires au :

- Filtre utilisé pour représenter les données à partir des stimuli,
- Modèle interprétatif utilisé pour attribuer un sens aux données,
- Schéma cognitif utilisé pour exploiter l'information.

L'utilisation d'une information fournit des résultats qui complètent la connaissance tacite de l'individu.

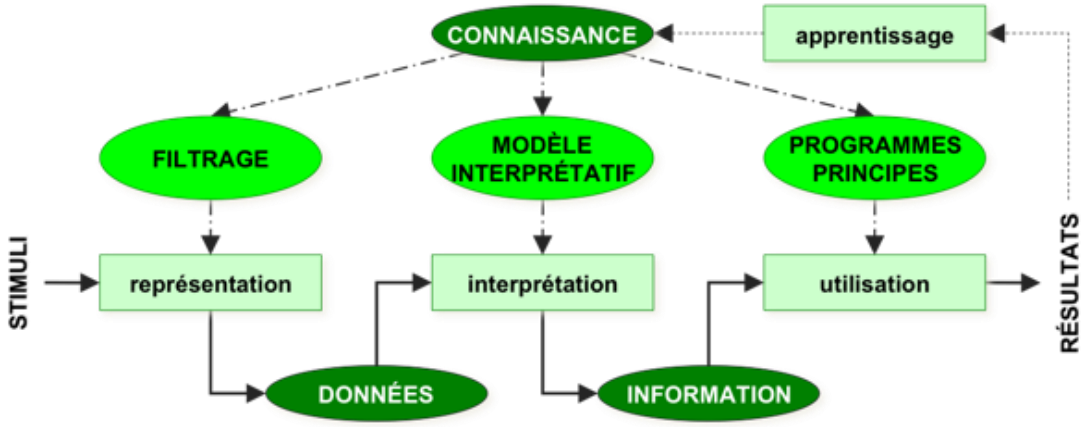


Figure 1.2 : Interaction entre donnée, information et connaissance.

Agent :

- Un agent peut-être une entité :
  - ✓ Qui va évoluer dans un environnement.
  - ✓ Percevoir son environnement (senseurs).
  - ✓ Qui va agir sur son environnement.
  - ✓ Qui va agir pour aboutir à un objectif.
- Un agent est un outil conceptuel pour analyser des systèmes : robotique, etc..

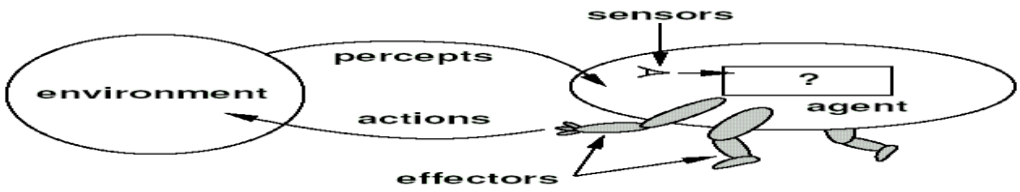


Figure 1.3 : Agent Vs. Environnement.

Connaissances et agents :

- Raisonnement intelligent n'est rien sans connaissances (expertise)
- Nécessité d'une représentation des connaissances
- Nécessité de développement des algorithmes / heuristiques pour le traitement de ces représentations.

Connaissance et raisonnement :

Raisonnement

- Penser de manière cohérente et logique.
- Inférence logique
- Le processus de création de connaissances implicites depuis des connaissances explicites.

Remarque: Nouvelles connaissances peuvent être créées depuis des connaissances à l'aide de raisonnements.

3. Qu'est-ce qu'une représentation ?

Représentation = Approximation

- Représentation : structure de symboles pour décrire un modèle du monde dans le contexte d'une tâche particulière. Si la représentation devait posséder toutes les propriétés de l'entité représentée, il s'agirait de l'entité elle-même !
- Exemple de la carte et du territoire dans la recherche d'itinéraire : les détails inutiles sont ignorés.
- Représentation : une relation entre 2 domaines :
    - ✓ Le domaine à représenter (ex: le nombre sept , le concept de femme),
    - ✓ Le domaine représente est bien souvent plus concret et accessible que le premier domaine (les symboles "7" ou "VII").

- Qu'est-ce-que la Représentation des connaissances ?

- ✓ Représentation du savoir
- ✓ Organisation du savoir
- ✓ Ensemble d'outils et de technologies destines a représenter et organiser le savoir humain pour l'utiliser et le partager (**Définition Wikipedia)**.
- ✓ **- Knowledge Representation (KR):**KR ="the field of study concerned with using formal symbols to represent a collection of propositions believed by some agent".
- ✓ ***Sciences cognitives :*** Comment les humains stockent et traitent l'information.
- ✓ ***IA :*** stocker les connaissances de sorte que des programmes peuvent la traiter.

- Pourquoi représenter la connaissance ?

- ✓ Apprentissage automatique
- ✓ Fouille de données
- ✓ Compréhension du langage
- ✓ Traduction automatique
- ✓ Raisonnement

But 1 : Partage du savoir.

But 2 : Raisonnement sur le savoir.

- ***Difficultés :***
  - ✓ Acquisition des données
  - ✓ Organisation des données
  - ✓ Pas de classement universel des différents types de connaissances
  - ✓ Ambiguïté des termes
  - ✓ Contextualisation
  - ✓ Validation des données

- *Acquisition des données :*

Comment acquérir les données ?

- ✓ Manuellement ?
- ✓ Automatiquement ?

- *Comment modéliser les connaissances ?*

- ✓ Traduction en structures manipulables par un programme

<u>- Objectifs :</u>

- Modèles formels de représentation des connaissances.
- Traiter des informations au niveau sémantique.
- Simuler le raisonnement humain par le biais de mécanismes d'inférence.

→La représentation des connaissances est au cœur de deux processus inverses et complémentaires :

- D'abord l'extraction des connaissances que possèdent certaines personnes expertes dans leur domaine, ou que d'autres personnes transmettent dans des documents, de façon à

les rendre largement disponibles (sous forme d'information) pour la formation d'autres personnes ;

- Ensuite l'acquisition de connaissances et compétences nouvelles par l'apprentissage, c'est-à-dire la transformation des informations en connaissance par des personnes, au moyen d'activités formelles ou informelles, empruntant une variété de formes et de soutiens.

4. Difficulté de concevoir la connaissance

4.1 Problématique de la connaissance

Le transfert de la connaissance du spécialiste vers le programme, pour une application particulière, est une tâche longue et difficile qui constitue un obstacle au développement des SE. Elle nécessite une grande collaboration entre deux corps de métiers parfois très éloignés : le domaine d'expertise et l'informatique. En effet, il existe des connaissances touchant aux principes de cheminement du raisonnement, aux prises de décision, aux diagnostiques de l'expert. Elles sont exprimées par des règles heuristiques, des métarègles.

On peut dire que les traits du domaine qui révèlent du "flair" de l'expert et qui constitue une part importante de son expérience, posent parfois d'énormes problèmes de formalisme.

4.2 La cognitique

La qualité de la connaissance est déterminante pour la performance d'un SE. Cette connaissance est de deux types :

- Savoir indiscuté (manuels, revues spécialisées, …).
- Connaissance heuristique (acquise par l'expert au cours des années).

La représentation de ce deuxième type de connaissance est possible grâce à des spécialistes : les ingénieurs de la connaissance ou cogniticiens.

Ces cogniticiens travaillent d'une part en relation avec l'expert où ils observent, analysent et formalisent le comportement ; et d'autre part, en relation avec l'ordinateur avec lequel ils communiquent par l'intermédiaire de langages plus ou moins évolués.

Le travail du cogniticien est en quelque sorte de "décompiler" le savoir de l'expert.

Actuellement, c'est à l'aide d'interviews et d'observations directes de l'expert en activité qu'a été la majorité des applications SE. Mais cette méthode présente un certain nombre d'inconvénients :

- L'information recueillie auprès de l'expert manque souvent d'objectivité.
- II n'y a pas de garantie de complétude de l'information.
- Les séances d'interviews sont chères et ralentissant parfois le déroulement du projet.

Les psychologues ont beaucoup aidé les ingénieurs à surmonter la difficulté de ce processus de transfert qui doit passer par la mise en place d'un document formalisant les différentes facettes de l'expertise.

5. Types de connaissance :

On peut définir le mot "connaissance" comme étant l'ensemble d'informations relatives au domaine d'expertise. On y trouve :

5.1 La connaissance factuelle

Ce sont :

- Les éléments de base, objets du monde réel liés à la perception immédiate, ils sont enregistrés tels quels.
- Les assertions et définitions : elles caractérisent d'une façon sûre les objets de base.
- Les concepts : ce sont des regroupements ou des généralisations des objets de base relatifs à une vision "individuelle" des classes.
- Les relations : elles traduisent aussi bien des propriétés élémentaires des éléments de base, que des relations de cause à effet.

5.2 La connaissance opératoire

- Les inférences : elles concernent l'ensemble des informations que nous avons sur les lois de fonctionnement de l'univers.
- Les théorèmes et règles de réécriture : ce sont des cas particuliers de règles d'inférences qui ont la caractéristique d'être sûre.
- Les stratégies et les heuristiques : ce sont des règles de comportements, innées ou

acquises, qui permettent d'inférer des actions à envisager dans une situation précise.

5.3 La connaissance généralisée

- Les algorithmes de résolution : ils accomplissent certaines tâches agencées, ensuite ordonnées d'actions mémorisées en blocs. Habituellement, ces procédures sont de courtes séquences.
- Les événements : ils se composent de l'ensemble des actions que les objets accomplissent ou subissent et des états dans lesquels ils se trouvent.
- Les scénarios : ils représentent en IA des agrégations événements

5.4 La méta connaissance

Définition 1 : La connaissance étant vue comme édifiée sur des paliers successifs, la méta connaissance est introduite pour renseigner sur cette édification en forme et en contenu. Elle est souvent définie comme "connaissance sur la connaissance".

Il faut noter que cette définition est difficile à transcrire. De même, les stratégies et les heuristiques sont intégrées dans cette famille.

Définition 2 : La connaissance est un ensemble lié de données et d'inférences. Le concept de connaissance est lié à l'interprétation des données. Il exige un « interprète ». Le rôle de l'interprète est la génération de nouveaux faits à partir de données. Ce mécanisme est aussi appelé « Inférence ».

La connaissance est aussi assimilée à la compétence. C'est-à-dire qu'elle est définie par sa fonction. Elle se caractérise par ce qu'elle fait, non pas par son contenu structurel.

6. Modes (formalismes) de représentation des connaissances :

Une question fondamentale à se poser concerne le formalisme selon lequel sera exprimé l'ensemble des connaissances dont on dispose au sujet du domaine abordé. Plusieurs formalismes existent. Néanmoins, on arrive à discerner deux grandes familles:

1. Les règles de productions et le calcul des prédicats. Un formalisme basé sur des fondements mathématiques.
2. Les représentations structurées telles que : les réseaux sémantiques, les frames et les scripts.

Les techniques de représentation des connaissances sont le langage interne utilisée pour l'acquisition des connaissances,

Une autre distinction est la connaissance « superficielle » et la connaissance «profonde»

- **Connaissance superficielle :** expression symbolique des associations des faits permettant un raisonnement « abstrait »
- **Connaissance profonde :** un modèle permettant le raisonnement par simulation.

La plupart des systèmes experts : fonctionnent avec une « connaissance superficielle ».

6.1 Procédural

C'est la représentation qui mélange données factuelles et contrôle. Elle a un certain nombre d'inconvénients :

- Difficile à modifier et à étendre en fonction de l'évolution de la connaissance.
- Ne permet ni une vision globale ni partielle des connaissances utilisées.
- Exclut toute justification des solutions trouvées.

Elle englobe :

- Les automates finis.
- Les programmes.

6.2 Déclaratif

Les systèmes déclaratifs sont issus des démonstrations de théorèmes basés sur le principe de résolution. On y trouve :

- **Les règles de production**
- **La logique :** La logique des propositions, la logique des prédicats « logique du 1er ordre », logique d'ordre 0^+, logique d'ordre 2.
- **Les réseaux sémantiques**
- **Les scripts et les plans**

6.3 Les objets structurés :

L'objet est une nouvelle structure de données consistant à mêler dans une même structure des connaissances déclaratives (les données) et procédurales (les procédures ou méthodes). Il existe plusieurs approches dans ce domaine. On y trouve :

- **Les frames**
- **Les langages orientés objet**

6.4 La représentation des connaissances incertaines :

L'expert est capable de raisonner sur des connaissances incertaines, malheureusement on ne dispose que de peu d'outils pour prendre en compte cette capacité.

Un médecin par exemple n'est jamais sûr quel tel symptôme est signe de telle maladie, que tel médicament sera supporté par le malade, que le malade guérira, etc.

On peut utiliser la théorie des probabilités pour définir le degré de vraisemblance d'un fait. De nombreux générateurs de systèmes experts offrent la possibilité aux utilisateurs de nuancer leur certitude concernant un fait en leur associant un degré de vraisemblance.

Exemple :

Question : « Avez-vous mal à la tête ? ou seulement un peu mal ? »

Réponse : « Sur une échelle qui associerait 0 au fait de ne pas avoir mal à la tête et 1 au fait de ne pas pouvoir le supporter, je dirai que j'ai mal à la tête avec un coefficient 0,45 »

Cet exemple révèle une des principales difficultés de cette méthode : il n'est pas raisonnable d'attendre d'un être humain, expert ou non, qu'il puisse définir avec précision de tels degrés de vraisemblance.

6.5 Agents intelligents : Intelligence Artificielle Distribuée (IAD) :

IAD est une branche de l'IA qui s'intéresse à la modélisation de comportement intelligent par coopération entre un ensemble d'entités intelligentes appelées « agents ».

L'IAD se divise en deux branches principales :

- La résolution distribuée de problèmes (RDP) qui étudie comment distribuer des compétences au niveau de chaque partie du système, de façon à ce qu'il soit globalement plus compétent que chacune de ses parties.

- La simulation des systèmes complexes (SSC) qui concerne plus particulièrement les Système Multi-Agent (SMA). Les SMA traitent le comportement d'un ensemble d'agents autonomes qui essaient de résoudre un problème commun.

La différence notable entre la RDP et les SMA est que :

- RDP : approche descendante (top down)
- SMA : approche ascendante (Bottom up)

Conclusion :

Dans ce chapitre nous avons défini les différents concepts-clés liés à la représentation de connaissances et nous avons présenté les différents modes et formalismes de représentation des connaissances.

Chapitre 02

Langage de Modélisation par Objets Typés

MOT

Introduction :

Le langage de Modélisation par Objets Typés (MOT) conçu par Paquette (2002, 2010) est un langage de représentation graphique des connaissances.

1. Structure du langage MOT :

Comme la plupart des langages, la structure de MOT se compose d'un alphabet, d'une grammaire et d'une sémantique.

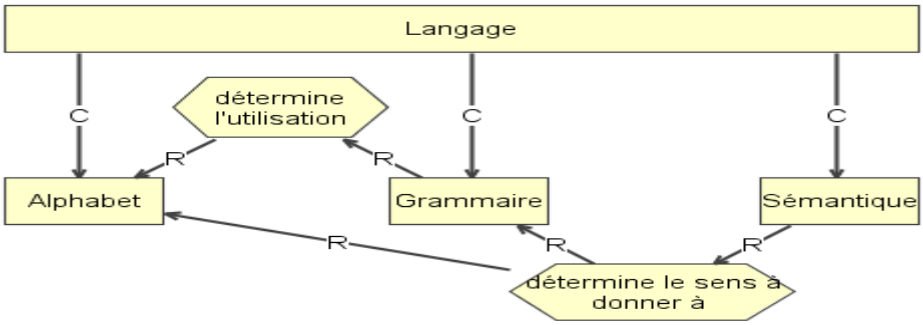


Figure 2.1 : Structure générale d'un langage.

- L'alphabet : est constitué de symboles, d'icônes ou de la représentation de base du langage (ce que l'on appelle parfois les primitives du langage).
- La grammaire : sert à définir les règles d'utilisation des symboles. L'application des règles est indépendante du sens que représentent les symboles.
- La sémantique est la définition du sens qui est donné aux symboles.

2. L'alphabet du langage MOT :

L'alphabet du langage MOT inclut deux types d'entités qui sont les connaissances et les relations. Les connaissances peuvent être « abstraites » ou « factuelles ».

Figure 2.2 : Structure de l'alphabet de MOT.

- La connaissance abstraite représente quelque chose ressemblant à une idée.
- La connaissance factuelle fait référence à une entité tangible, qu'on peut aussi nommer « objet concret ».

3. Types des connaissances en MOT :

3.1 L'alphabet de MOT associé aux types connaissances :

- Le langage semi-formel MOT différencie les types de connaissances au moyen de symboles graphiques.

- le langage MOT offre la possibilité de représenter des connaissances selon deux niveaux d'abstraction: conceptuel (Connaissance Abstraite) et factuel (Fait).

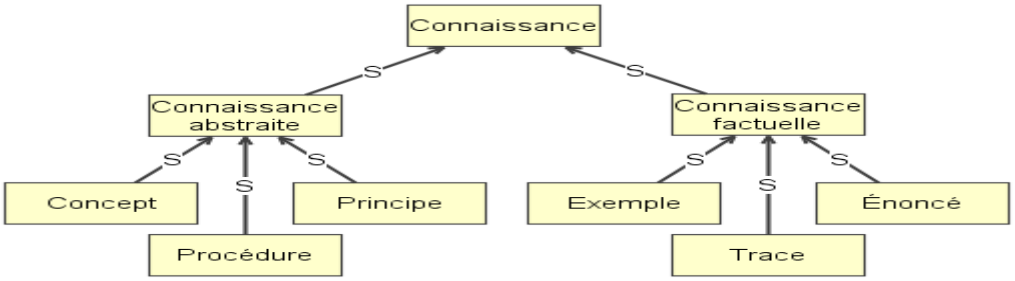


Figure 2.3: Représentation des connaissances en langage MOT.

3.2 La sémantique de MOT associé aux types de connaissances

a)- le concept représente « le quoi » des choses . Il sert à décrire l'essence d'un objet concret. Il peut être associé à l'idée de classe ou de catégorie.

- L'exemple représente l'un de ces objets en énonçant un certain nombre de faits qui le décrivent.

b)- La procédure permet de décrire « le comment » des choses. Elle désigne des opérations, des actions pouvant être accomplies.

- La trace représente l'ensemble des faits concrets obtenus lors de l'exécution d'une procédure.

c)- Le principe désigne « le pourquoi », « le quand » ou le « qui » associé à une chose. Il est une connaissance stratégique qui permet de nommer une relation qui existe entre des objets, que ce soit des concepts, des procédures ou d'autres principes. Il sert notamment à représenter une condition pouvant s'appliquer à l'exécution d'une action.

- L'énoncé représente l'instanciation d'un principe à propos d'objets concrets.

Type de connaissance	Connaissance abstraite		Connaissance factuelle	
Déclarative *Le quoi des choses*	Concept		Exemple	
Action *Le comment de choses*	Procédure		Trace	
Stratégique *Le pourquoi, le quand, le qui*	Principe		Énoncé	

Tableau 1.1 : Type de connaissances dans le langage MOT et leur symbole associé.

3.3 Le stéréotype :

Le stéréotype ne fait pas partie de la définition de base du langage MOT tel que défini par Paquette (2002). Il Largement utilisé et standardisé en modélisation UML. le stéréotype est une extension du vocabulaire qui permet à un modélisateur d'associer un élément d'un modèle à un autre domaine de connaissances. Par exemple, une connaissance procédurale P pourrait être stéréotypée par une tâche, une procédure ou encore une méthode. Le stéréotype est s'encapsule par les symboles « ».

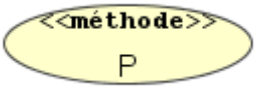


Figure 2.4: Représentation de la procédure P dont le stéréotype une méthode.

4.Type de relations dans MOT :

4.1 L'alphabet des relations :

La relation est un lien directionnel qui unie des connaissances. Le langage MOT offre un ensemble de liens qui sont typés :

- **Le lien I** représente une relation d'instanciation.
- **Le lien S** représente une relation de spécialisation.
- **Le lien R** représente une relation de régulation.
- **Le lien A** représente une association d'application.
- **Le lien P** représente la préséance.
- **Le lien IP** représente une association d'intrant/produit.
- **Le lien C/C*** représente l'association de composition et de composition multiple.
- **Le lien E** représente une relation d'englobement.

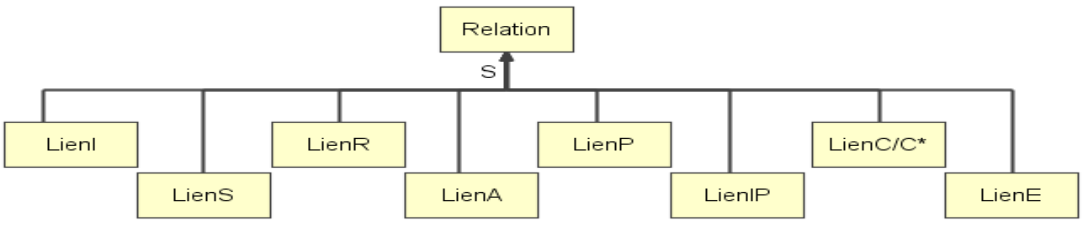


Figure 2.5: Hiérarchie des relations typées utilisées en langage MOT et leur représentation dans l'ontologie du langage semi-formel.

4.2 Sémantique des relations :

Chaque type de lien possède une sémantique propre qui respecte des règles d'intégrité.

Type de lien	**Signification**

S	Le lien de spécialisation associe deux connaissances abstraites de même type dont la première est une spécialisation de la seconde. Ce lien est notamment utile dans la description des taxonomies. Le lien de spécialisation est une relation transitive.
I	Le lien d'instanciation associe à une connaissance abstraite un fait qui caractérise une instance de cette connaissance. Le lien d'instanciation n'est pas une relation transitive.
I/P	Le lien intrant/produit sert à associer une connaissance procédurale à une connaissance conceptuelle afin de représenter l'intrant ou le produit d'une procédure. Ce lien est notamment utile dans la description des algorithmes, des processus et des méthodes. Le lien intrant/produit n'est pas une relation transitive.
P	Le lien de précédence associe une connaissance à une autre qui la suit dans une séquence temporelle de procédures ou de règle de décision (principes). Le lien de précédence est une relation transitive.
R	Le lien de régulation associe une connaissance stratégique (un Principe ou un Énoncé) à une autre connaissance afin de préciser une contrainte, une restriction ou une règle qui régit la connaissance. Le lien de régulation est une relation non-transitive.
C,C*	Les liens de composition et de composition multiple permettent de représenter l'association entre une connaissance et des connaissances qui la composent. Le lien de composition est une relation transitive.
E	Le lien englobe ne possède pas de symbolique particulière dans le langage MOT original. Il s'agit d'une relation qui unit l'élément d'un modèle aux éléments d'un sous-modèle. Le lien englobe est une relation transitive.

Tableau 2.2 : Sémantique des relations typées dans MOT (Adaptée de Paquette 2002).

Certaines règles d'association entre des connaissances sources et des connaissances destinations sont appliquées à chacun des types de relation. Ces règles définissent les relations considérées valides entre les différents types de connaissances du point de vue de la sémantique MOT.

Voici les quelques règles générales d'utilisation :

- Règle 1 : une relation ne peut pas exister seule; elle doit, à son origine et à sa destination, référer à une connaissance (factuelle et/ou abstraite selon le cas).

- Règle 2 : il est possible qu'une relation possède la même connaissance d'origine et de destination.
- Règle 3 : une connaissance peut exister seule, sans qu'elle soit l'origine ou la destination d'une relation.

Plus spécifiquement, il existe un ensemble de règles secondaires qui régissent chacune des relations en fonction de la nature des connaissances d'origine et de destination qu'elles associent.

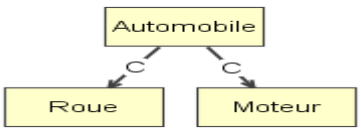
Destination	**Connaissances abstraites**			**Connaissances factuelles**		
Origine	**Concept**	**Procédure**	**Principe**	**Exemple**	**Trace**	**Énoncé**
Concept	C, S	I/P	R	I, C		
Procédure	I/P	C, S, P	C, P		I, C	
Principe	R	C, R, P	C,S,P, R			I, C
Exemple	A	A	A	A, C	A, I/P	A
Trace	A	A	A	A, I/P	A, C, P	A, C, P
Énoncé	A	A	A	A, R	A, C, R, P	A, C, R, P

Tableau 2.3 : Grammaire des relations MOT (adaptée de Paquette 2002).

5. Sémantique des éléments grammaticaux du langage MOT :

5.1 La composition :

Le lien « C », qui se lit : « lien de composition », sert à représenter les composants, les constituants d'une connaissance. Il permet d'indiquer qu'une connaissance se compose d'une ou plusieurs autres connaissances.

Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Concept	Une « Automobile » se compose de « Moteur », de « Roue » 
Procédure/ Procédure	« retirer de l'argent au guichet automatique » se compose de : » entrer la carte de débit », « entrer le NIP », « choisir un compte », « entrer le montant », « retirer la carte et l'argent ».

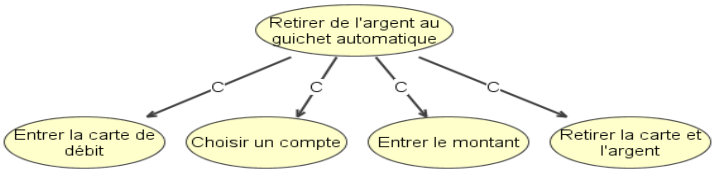
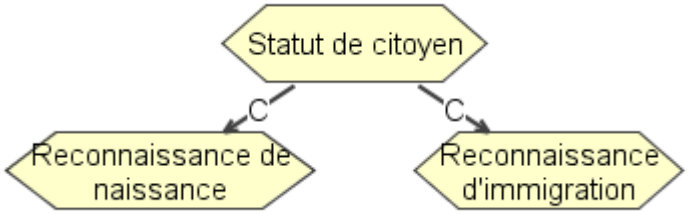
	
Principe/ Principe	Le « statut de citoyen » se compose des règles de « reconnaissance de naissance », de « reconnaissance d'immigration ». 

Tableau 4.4 : Exemple de représentation d'une composition entre connaissances.

5.2 La spécialisation :

Le lien « S », qui se lit : « lien de spécialisation », sert à représenter la spécialisation d'une connaissance par rapport à une autre. Il permet de désigner des cas particuliers de connaissances conceptuelles. Les connaissances liées par un lien de spécialisation sont des connaissances de même type.

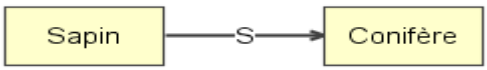
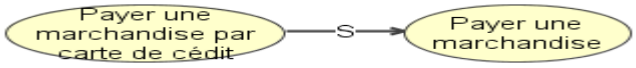
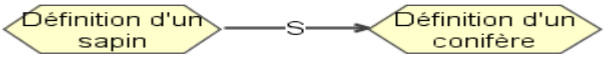
Origine/ Destination	Exemple/ Représentation en MOT
Concept / Concept	Un « Sapin » est une sorte de « Conifère » 
Procédure/ Procédure	« Payer une marchandise par carte de crédit » est une sorte de façon de « payer une marchandise » 
Principe/ Principe	La « définition d'un sapin » est une sorte de « définition de conifère ». 

Tableau 2.5: Exemple de représentation de la spécialisation de connaissances

5.3 La régulation:

Le lien « R », qui se lit : « lien de régulation », est une relation qui met en jeu un principe et l'une ou l'autre des connaissances abstraites . En tant qu'agent, norme ou contrainte, le principe est utilisé en conjonction avec un lien de régulation pour indiquer une situation de régulation d'un objet par un autre.

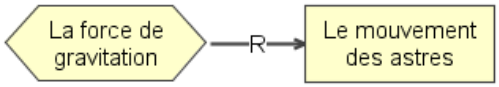
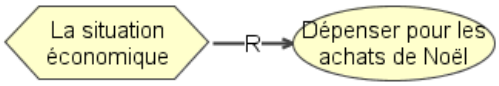
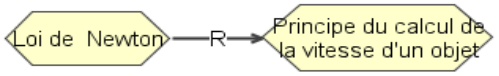
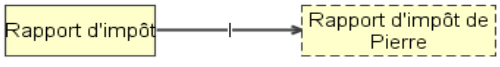
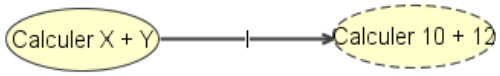
Origine/ Destination	**Exemple/ Représentation en MOT**
Principe / Concept	« La force gravitation » régit « le mouvement des astres ». 
Principe/ Procédure	«La situation économique» régit l'action « de dépenser pour les achats de Noël ». 
Principe/ Principe	La « Loi de Newton » régit le « principe du calcul de la vitesse d'un objet ». 

Tableau 2.6 : Exemple de représentation d'une régulation entre des connaissances

5.4 L'instanciation :

Le lien « I », qui se lit : « lien d'instanciation », met en relation une connaissance abstraite et la connaissance factuelle de même type. Il sert à représenter la relation entre un objet concret et l'abstraction qui lui est associée.

Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Exemple	Le « Rapport d'impôt » a pour instance « le rapport d'impôt de Pierre » 
Procédure / Trace	« Calculer 10 + 12 » est un calcul de type « calculer X + Y » 

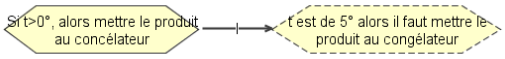
Principe / Énoncé	L'énoncé « t est de 5° alors il faut mettre le produit au congélateur » est une instanciation de la règle « Si t>0° alors mettre le produit au congélateur » 

Tableau 2.7: Exemple de représentation de l'instanciation entre une connaissance abstraite et une connaissance factuelle.

5.5 L'intrant et le produit :

Le lien « I/P », qui se lit : « lien intrant/produit », met en relation une connaissance de type procédure et de type concept. Il sert à désigner les composants nécessaires à la réalisation de la procédure ainsi que les objets produits par la procédure.

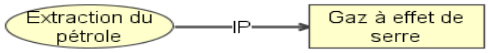
Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Procédure	L'« Essence » est l'intrant du processus de « faire rouler une automobile » 
Procédure/ Concept	Le processus d' « extraction du pétrole » produit des « Gaz à effet de serre » 

Tableau 2.8: Exemple représentation d'un intrant et d'un produit entre une connaissance procédurale et une connaissance conceptuelle.

5.6 La précédence :

Le lien « P », qui se lit « lien de précédence » met en relation des procédures ou des principes. Il sert à ordonner la séquence d'exécution des procédures ou l'ordonnancement de l'application de principes.

Origine/ Destination	**Exemple/ Représentation en MOT**
Procédure/ Procédure	le processus : « entrer la carte » précède » choisir un compte « qui précède » entrer le montant « qui précède « retirer la carte et l'argent »

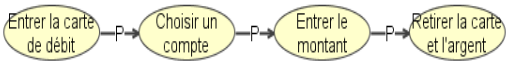
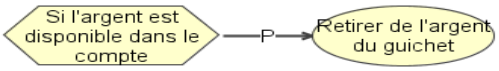
	
Principe/ Procédure	« On retire l'argent du guichet » à condition que « l'argent soit disponible dans le compte » 

Tableau 2.9 : Exemple de représentation de la préséance.

5.7 Le lien d'application :

Le lien « A », qui se lit « lien d'application », met en relation une connaissance factuelle avec une connaissance abstraite. Dans la majorité des situations, un modèle MOT sert à représenter le vocabulaire ainsi que les entités qui composent un domaine d'application spécifique. Dans certains cas, le vocabulaire d'un domaine d'application sert à la représentation d'un autre domaine. On dira alors que le vocabulaire du premier domaine d'application est la méta connaissance du vocabulaire du deuxième domaine d'application.

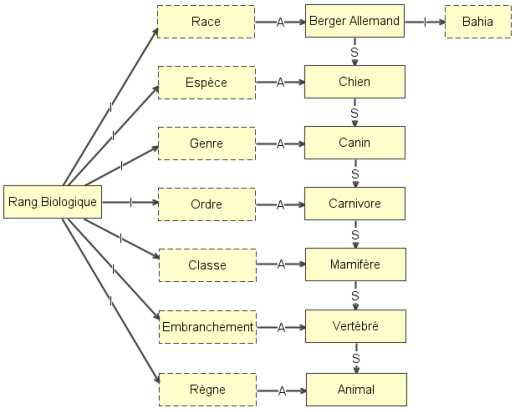
Origine/ Destination	**Exemple/ Représentation en MOT**
Connaissance factuelle/ Connaissance Abstraite	Dans cet exemple, «Bahia » qui est de la « Race » des « Berger », qui est de l' » Espèce »des« Chien »qui est du « Règne » « Animal ». La«Race », l' « Espèces», le « Genre », …, le « Règne » sont des entités du concept «Rang Biologique». 

Tableau 2.10 : Exemple de représentation d'un lien d'application.

5.8 La propriété et l'attribut :

L'attribut et la propriété sont deux associations qui mettent en relation deux connaissances abstraites. La représentation d'une relation d'attribut est exprimée par le Lien C entre deux concepts. Cette utilisation du Lien C crée une ambiguïté avec l'interprétation de composition. C'est la sémantique du domaine représentée qui permet de déterminer l'interprétation adéquate du l'utilisation du Lien C.

Par ailleurs, la représentation de la propriété nécessite une utilisation particulière du Lien R, qui ne fait pas partie de la définition initiale du langage MOT. Il s'agit d'inclure un principe entre deux concepts reliés par des Lien R. Le principe prend alors le rôle de propriété qui unit deux concepts. Cette combinaison peut aussi s'appliquer pour la définition de propriétés entre des connaissances procédurales.

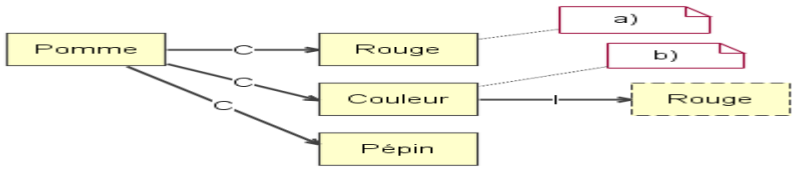
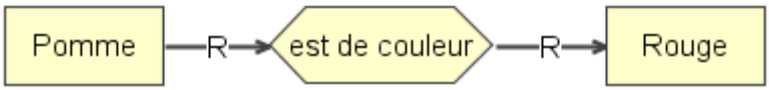
Exemple/ Représentation en MOT
« La pomme » se compose de « pépin » et a) « La pomme » a pour attribut « Rouge » ; b) de manière plus formelle: » la pomme » a pour attribut « couleur » qui a la valeur d'être « rouge » 
« la pomme » « est de couleur » « rouge ». 

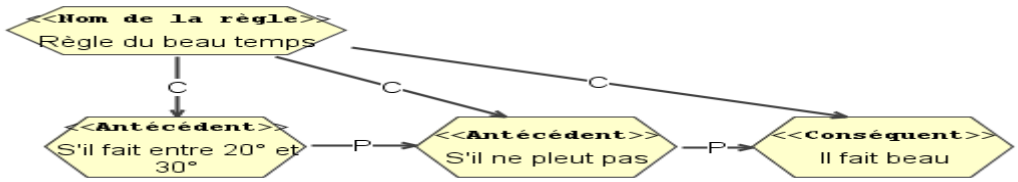
Tableau 2.11: Exemple de représentation d'un attribut et d'une propriété.

5.9 La règle :

La règle est un énoncé qui se compose du nom de la règle et d'un ensemble d'antécédents mis en conjonction pour produire une conclusion ou une opération. En MOT, il est possible de représenter une règle par une topologie particulière dans l'utilisation de principes, de Lien C, de Lien P et de procédures. La signification de chacun de ces éléments du modèle est mise en stéréotype.

« La règle du beau temps »: « S'il fait entre 20° et 30° » et « s'il ne pleut pas » alors

« il fait beau »



« La règle d'inoculation généralisée »: « S'il y a pandémie de grippe » alors « inoculer la population »

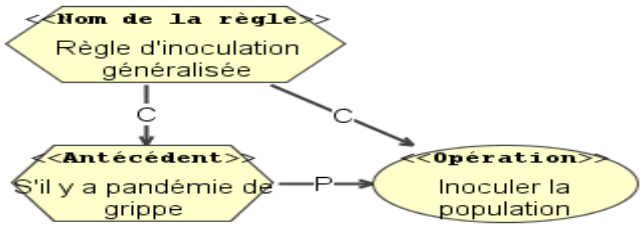


Tableau 2.12: Exemple de représentation d'une règle.

6. Exemple d'un MOT :

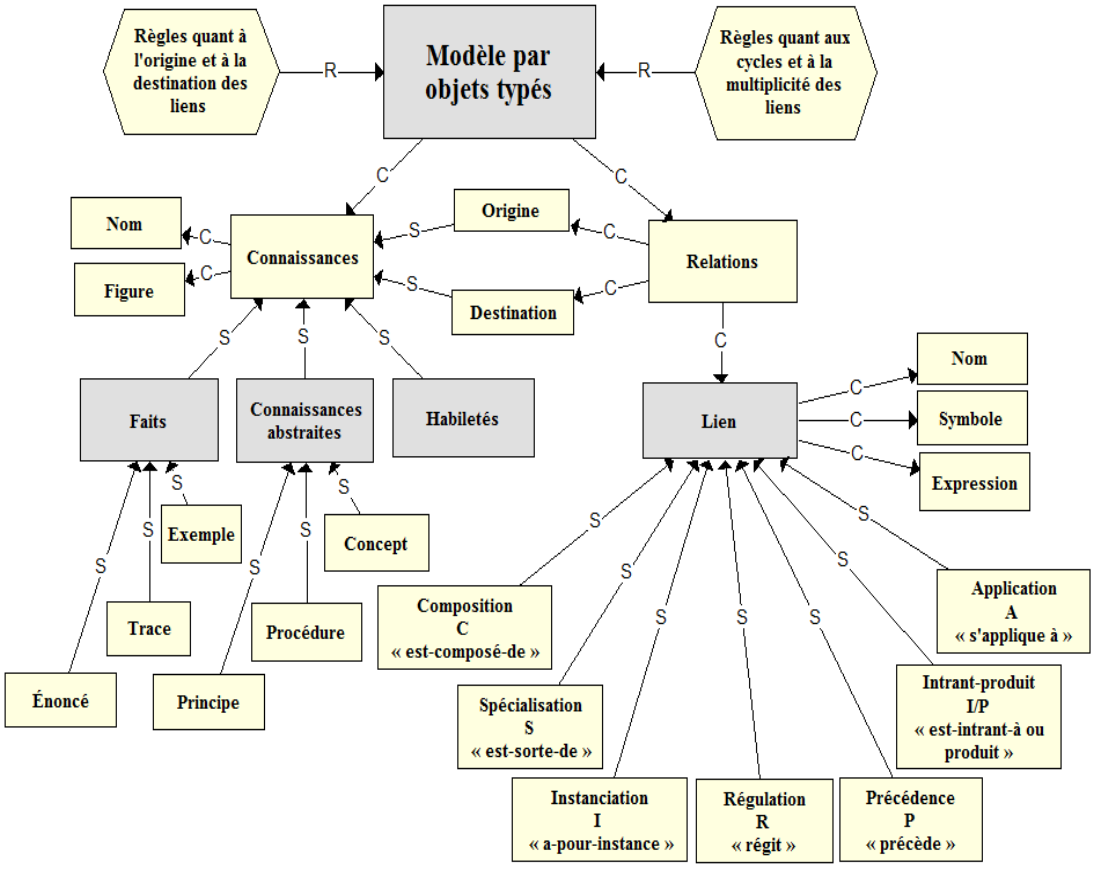


Tableau 2.13: Exemple d'un MOT.

Conclusion :

Le langage de Modélisation par Objets Typés (MOT) est un langage de représentation graphique et semi-formel de la connaissance. La richesse de l'expressivité, la souplesse et la simplicité de sa grammaire, le guidage représentationnel assuré par la typologie du vocabulaire fait de MOT un langage qui favorise l'extériorisation de la connaissance tacite en connaissance explicite. Initialement conçu afin de satisfaire les exigences de la modélisation en ingénierie pédagogique, le langage MOT peut aussi être employé dans diverses disciplines. Il est employé pour représenter graphiquement des problématiques, des théories ou encore des méthodologies. Il est aussi employé pour représenter le domaine d'un discours. En gestion des connaissances, MOT permet de représenter les acteurs, les ressources, les procédures, les forces et les contraintes ainsi que l'intrant et le produit d'un système organisationnel. En émergence de l'innovation, MOT est employé en transfert de connaissances. MOT s'adresse à des pédagogues, des coaches, des gestionnaires de projets, des experts de métier, des ingénieurs, des professionnels de la communication, des penseurs ou à toutes personnes désireuses de partager des connaissances.

Chapitre 03

Représentation graphique des connaissances : Réseaux sémantiques

Introduction :

Les réseaux sémantiques conçus à l'origine en **linguistique** pour devenir ensuite un **langage pour la représentation de concepts** très divers, une **structure informatique** utilisée en IA (QUILLIAN / COLLINS 1966)

Les réseaux sémantiques, moins formels que les représentations logiques et plus directement développés par l'intelligence artificielle, tirent leur origine de certains résultats de psychologie cognitive sur la mémoire associative et se veulent un modèle de représentation du contenu sémantique des concepts des langues qui soit psychologiquement valide.

1- Définition des Réseaux Sémantiques :

Un réseau sémantique est une structure de graphe dont la fonction est l'encodage des connaissances taxonomiques concernant des objets ainsi que leurs propriétés.

Les réseaux sémantiques sont une manière de représenter des relations entre des objets (nœuds). Les nœuds (objets) sont reliés entre eux et les liens ont une signification. Les liens sont orientés car la relation n'est pas symétrique.

un réseau sémantique est un graphe composé :

a) d'un ensemble de nœuds étiquetés : représentant généralement des objets, des concepts, des actions, des évènements.

Donc dans un réseau sémantique existent **deux types de nœuds :**

- Les nœuds étiquetés par des constantes de concepts (représentant des catégories taxonomiques).
- Les nœuds étiquetés par des constantes d'objets (représentant des instanciations des concepts ou des propriétés des concepts),

b) d'un ensemble de liens orientés et étiquetés entre ces nœuds : représentant généralement des relations entre des objets.

Dans un réseau sémantique existent **trois types d'arcs connectant les nœuds :**

- les arcs d'agrégation (spécialisation) (appelés aussi "sorte de" ou *"liens isa (IS A)" ;*
- les arcs d'instanciation (appelés aussi: " est un » ou *"liens ako (IS A KIND OF) " ;*
- les arcs de composition (appelés aussi " a "un attribut ou *"liens asa (HAS A)"*.

c) d'un ensemble d'opérations d'exploitation de ce graphe : constituants les mécanismes de raisonnement.

- représentation graphique :

Facilite la lecture, ne correspond généralement pas au formalisme d'implémentation,

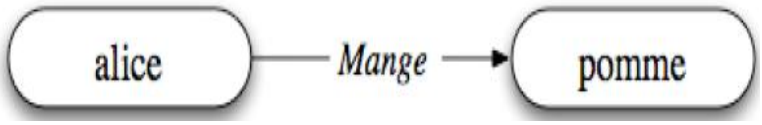


Figure 3.1 : Représentation graphique 1.

- représentation non-graphique:

(alice, manger, pomme)

2- Concepts de base des Réseaux Sémantiques (RS):

a) Les NŒUDS:

- **atomiques** : entités élémentaires (valeurs, individus,...)
- **complexes** : entités complexes (propositions, phrases,...)
- ils doivent être **typés** : concept, individu, action, proposition, etc...

b) Les LIENS:

- **structuraux** : indépendants de la sémantique du domaine,
- **spécifiques** : dépendants de la sémantique du domaine,

Remarque : il faut essayer d'augmenter la proportion des liens structuraux par rapport aux liens spécifiques.

c) Les OPERATIONS

- souvent représentées par le **programme**,
- doivent être **définies clairement**,

2.1 Nœuds concepts:

"les canaris / sont des / oiseaux"

canaris et oiseaux =

concepts (nom communs) --> classe

sont des =

relation --> inclusion de classes

lien « sorte_de »

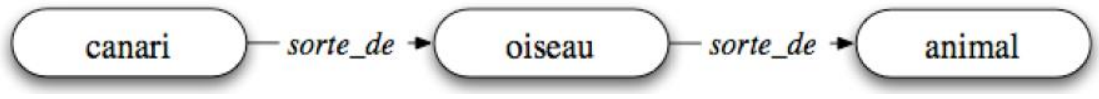


Figure 3.2 : Représentation graphique 2.

- lien **structurel** indépendant du domaine
- représente une **inclusion**
  - De **propriétés** (pt de vue intentionnel, cas général)
  - **D'individus** (pt de vue extensionnel).

2.2 Nœuds individus :

"Titi / est un / canari"

canari = **concepts**

Titi = **individu** (nom propre) --> élément d'un ensemble

est un = **relation** --> appartenance d'un élément à une classe

lien « instance »

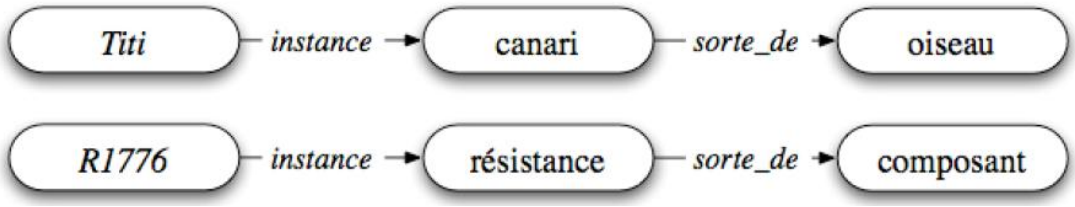


Figure 3.3 : Représentation graphique 3.

lien « instance » = lien **structurel**

2.3 Les réseaux sémantiques: taxonomies:

- liens **sorte_de** et nœuds **concepts +** liens **d'instance** et nœuds **individu**:

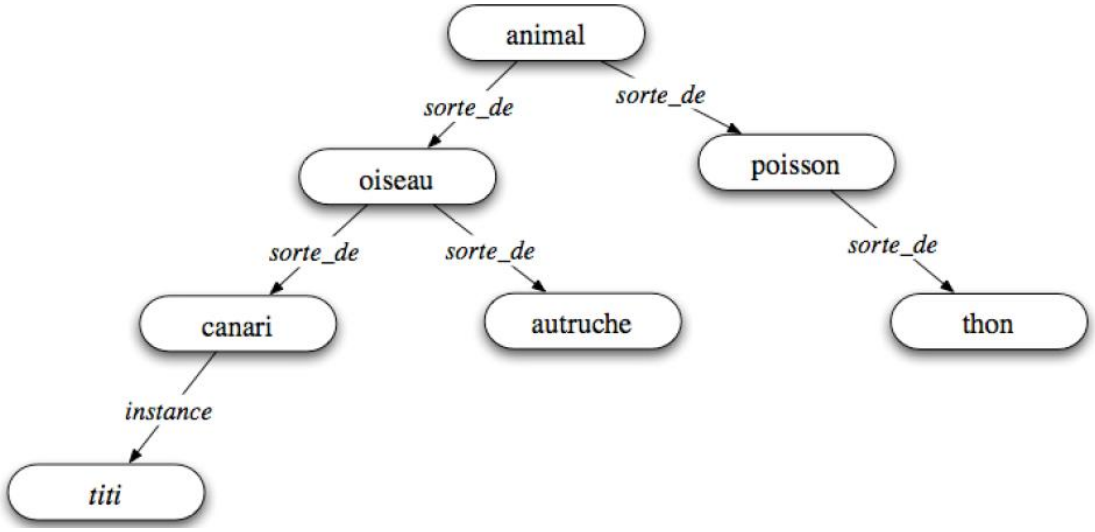


Figure 3.4 : Taxonomies dans le réseau sémantique.

2.4 Propriétés:

Les **propriétés** sont des informations rattachées à chaque nœud du RS:

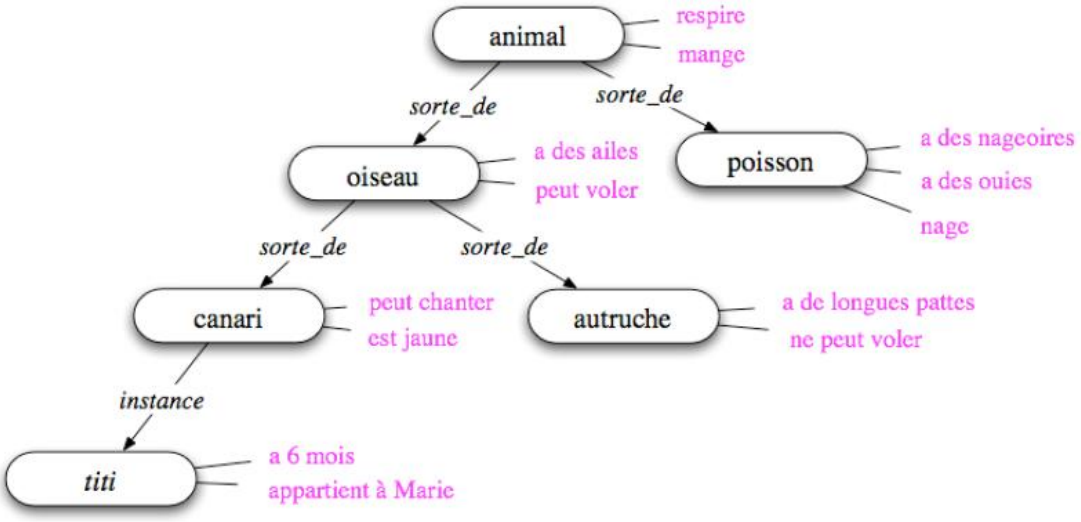


Figure 3.5 : Propriétés dans le réseau sémantique.

- simples

- elles ne permettent pas de répondre à des questions comme :

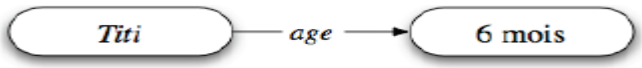
"quel est l'âges de Titi ?" "quelle est la couleur des canaris ? "

-> notion d'attribut

2.5 Attributs :

- attribut = relation qui relie un **nœud concept** ou un **nœud individu** à une **valeur** ou **propriété.**

« L'âge de Titi est de 6 mois »



« la couleur des canaris est le jaune »

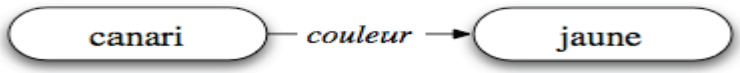


Figure 3.6 : Attributs dans le réseau sémantique.

- lien spécifique dont le sens dépend du domaine d'application -> interprétation ad-hoc,

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

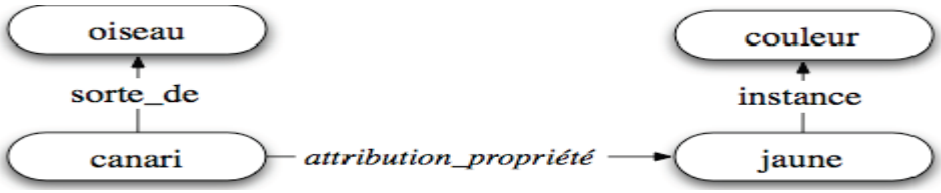


Figure 3.7 : Exemple d'un nœud attribut.

Notion **d'attribut** :

- une classe **sémantique** de nœud dont les instances sont des **propriétés.**

- un **attribut** peut lui-même être **caractérisé** :

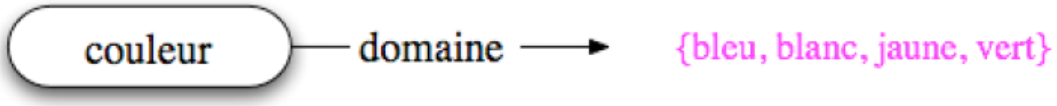
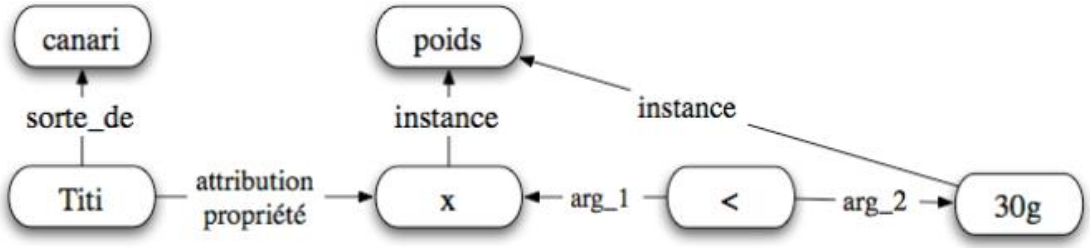


Figure 3.8 : Exemple d'un domaine d'un attribut.

- **domaine** = relation structurelle permettant de vérifier des contraintes d'intégrités.

2.6 Rapports attribut / valeur :

- un **nœud-attribut** peut être relié à une ou plusieurs valeurs par l'intermédiaire d'un opérateur:



soit en simplifiant:

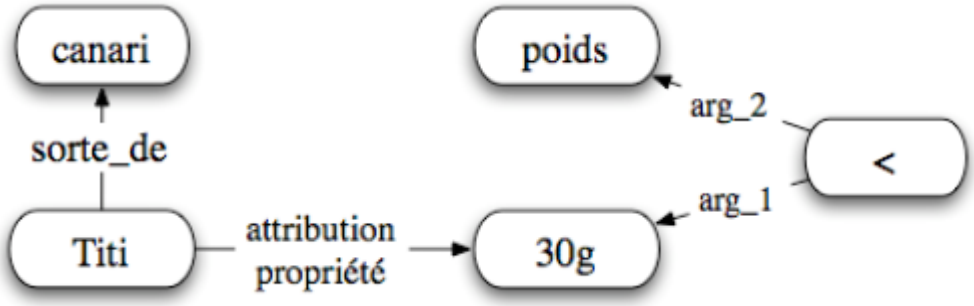


Figure 3.9 : Rapports attribut / valeur.

- cet opérateur peut être n-aire:

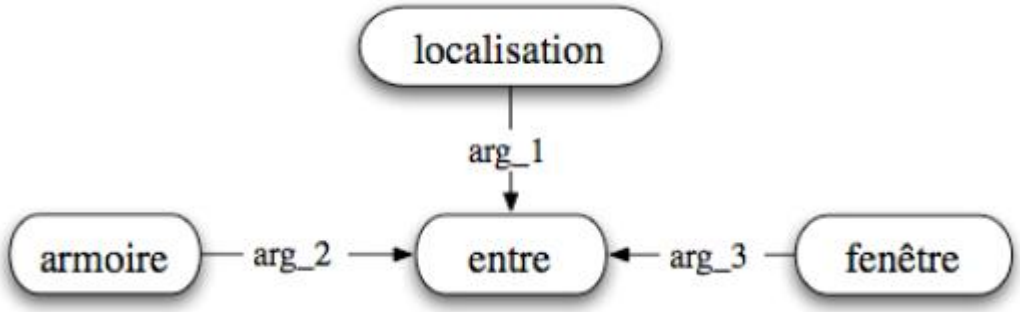


Figure 3.10 : Rapports attribut / valeur.

3- Héritage dans les RS:

L'héritage dans les RS **repose sur des liens de type « est_un »** ou **« sorte_de » reliant un concept à un autre concept plus élevé :**

Exemple : "canari" est une sorte de "oiseau"

- héritage des propriétés rattachées au concept père au concept fils :

Ainsi, on pourra dire que « le canari a des ailes et une peau » en remontant les liens « sorte_de »

- le principe d'héritage permet :

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

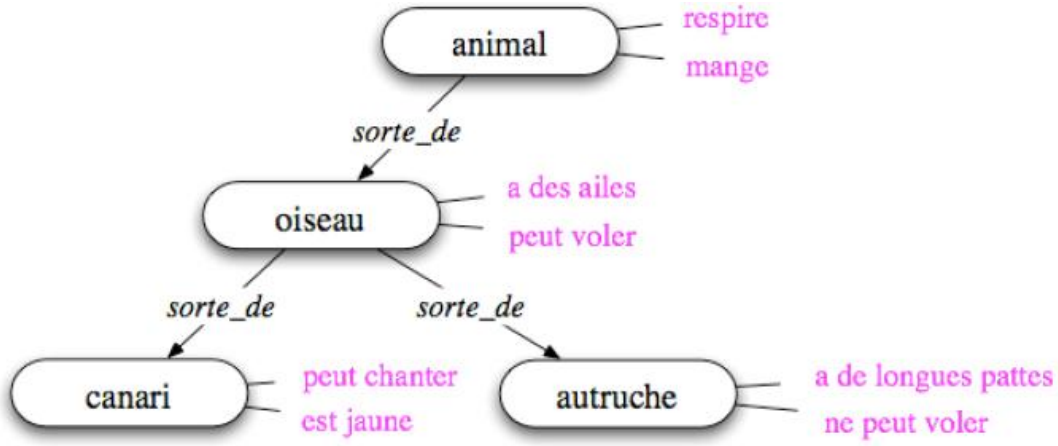


Figure 3.11 : Rapports attribut / valeur.

4- Partition dans les RS:

- **Partition**= regroupement de nœuds et d'arcs du réseau dans des espaces spécifiant la portée de relations.

- intérêts des partitions:

- définition de **contextes**
- permet la **quantification**

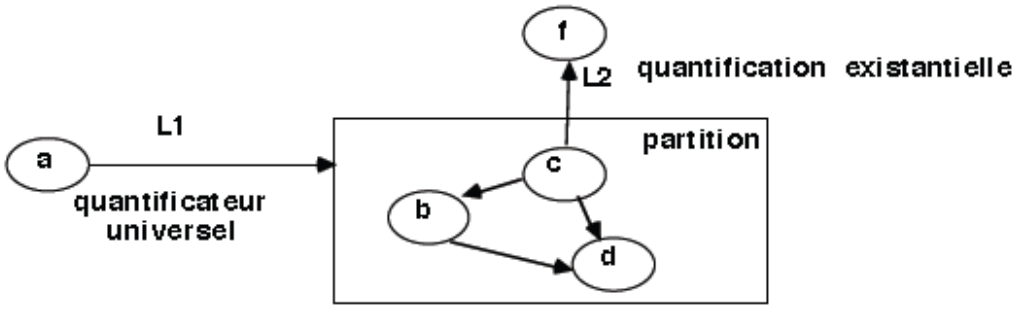


Figure 3.12 : Partition dans les RS.

- cadres : définissent l'étendue des identificateurs universels

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

- lien L2: quantificateur existentiel explicite sur le nœud f par rapport au nœud c.

5- RS et Quantification:

- Quantification traitée par la notion de **partition**

- Soit le fait à représenter suivant : « tout chat a mordu un oiseau »

-> représentation logique : ∀ x chat (x) -> (∃ y oiseau (y) ∧ mordre (x,y))

-> **encodage de la variable quantifiée universellement x** en utilisant une partition (cadre rectangulaire) :

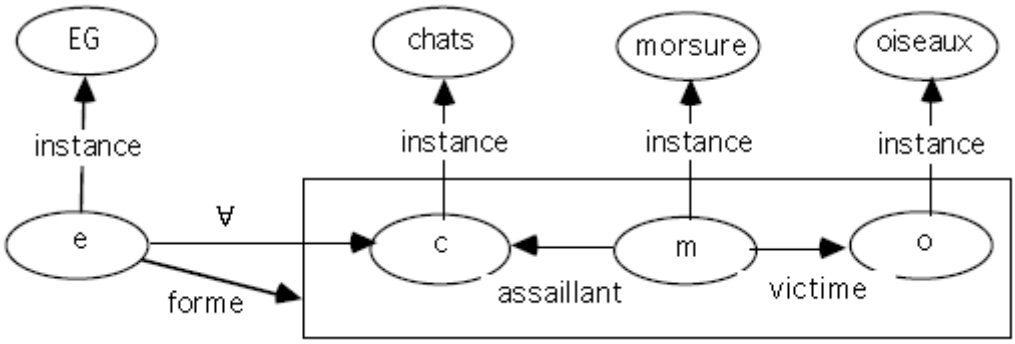


Figure 3.13 : RS et Quantification.

- les nœuds c, m, o sont des instances de chats, morsure, oiseaux,

- le cadre introduit dans le réseau définit l'étendue de l'identificateur universel,

- le nœud e représente l'assertion à représenter, instance de l'ensemble des énoncés généraux EG sur le monde,

- chaque élément de EG possède:

- une connexion « **forme** » pointant vers le cadre de la partition et énonce l'affirmation,
- une ou plusieurs connections «∀» pointant vers chaque variable quantifiée universellement, ici variable c (les variables m et o sont ici quantifiée existentiellement).

6- RS et logique :

- introduction de la **logique des prédicats du premier ordre**
- introduction d'un "**nœud prédicatif**" instancié en lui associant :
  - Un pointeur vers le prédicat
  - Un pointeur vers chaque argument du prédicat

Exemple :

Soit la phrase « Jean donne un livre a Marie »

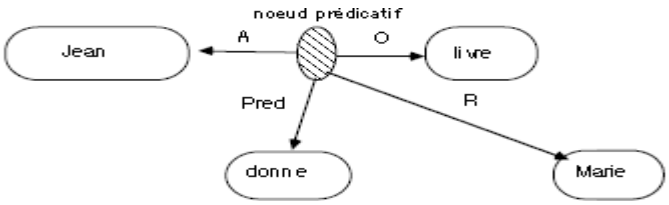


Figure 3.14 : RS et logique.

Soit : **donne (A, O, R)**

Avec : A (agent) = Jean ; O (objet) = livre ; R (receveur) = Marie

- Connecteurs logiques ET,OU:

Soit la phrase suivante: « Gaston ira sur le lit ou sur le fauteuil », une représentation:

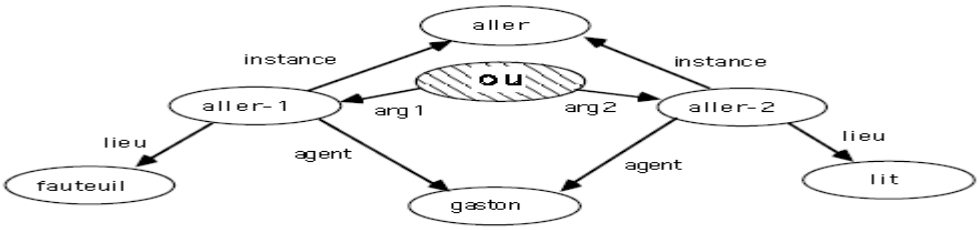


Figure 3.15 : RS et Connecteurs logiques ET,OU.

- Représentation de la négation :

Soit la phrase: « Marie n'aime pas Gaston », une représentation:

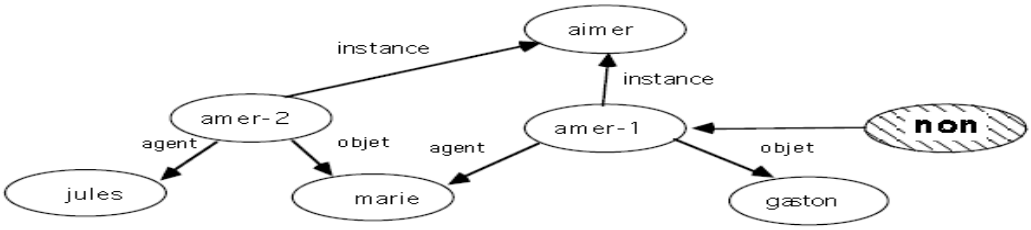


Figure 3.16 : RS et représentation de la négation.

7- Représentation d'évènements ou d'actions :

Représenter l'événement : « Gaston casse le vase »:

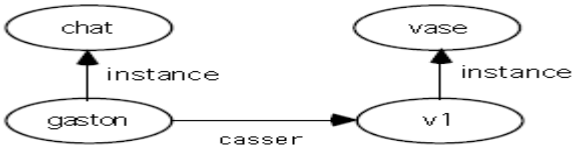


Figure 3.17: RS et représentation d'actions.

Le lien « casser » et **spécifique**. On peut s'en séparer en le traduisant par des liens plus structurels : agent, objet, instrument, temps, lieu, ... :

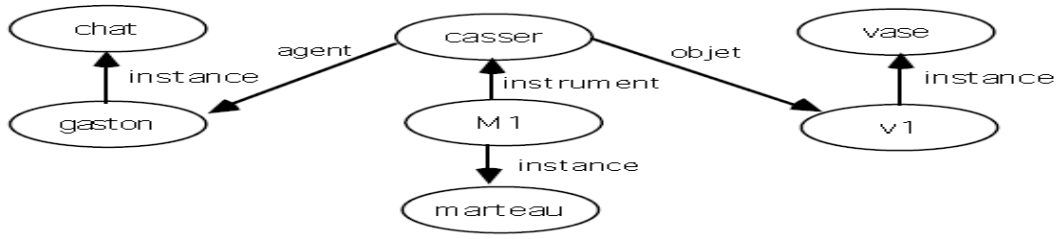


Figure 3.18 : RS et représentation d'actions avec des liens structurels.

8- Interprétation des connaissances dans les RS:

On dispose d'une base de connaissances organisée en réseau sémantique :

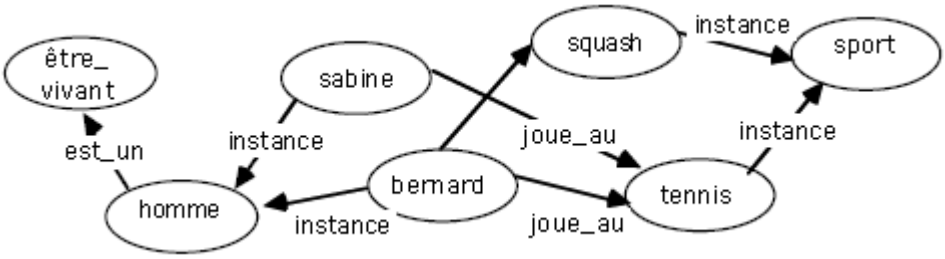


Figure 3.19 : Exemple d'un RS.

- la question « quelqu'un fait-il du sport ? » représentée par le fragment de réseau suivant :

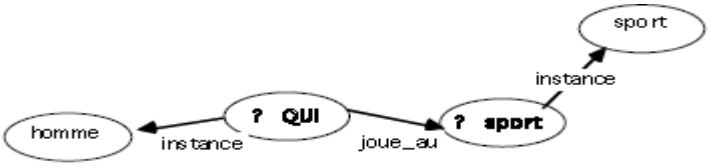


Figure 3.20 : Exemple d'un fragment dans un RS.

- les réponses seront:

sabine joue_au tennis

bernard joue_au tennis

bernard joue_au squash

si réseau important -> problèmes combinatoires ....

9- Intérêts des réseaux sémantiques:

Axes organisationnels qu'ils offrent pour **structurer** une base de connaissances:

- La classification, ensembles/sur-ensembles:

Un objet peut être associé avec son ou ses types génériques,

Ex : Titi peut être associé à oiseau, animal.

- conduit à la distinction fondamentale de type (canari) et d'occurrence (Titi).
- peut être récursive -> définir des méta-types ayant pour instance d'autres types.

- Agrégation:

- rattacher à un objet des propriétés ou d'autres objets y intervenant comme parties.
  Ex : Titi, vu comme objet physique possède des ailes, une tête et une queue, considéré dans son environnement, il possède un nid, un territoire, un chant, une nourriture.
- peut être appliqué récursivement: un composant peut être à son tour composé d'autres composants.

- La généralisation, la spécialisation:

relie un type à un autre type plus générique,

Ex : oiseau à animal

- La généralisation (lien « sorte_de ») = un ordre partiel organisant 2 types dans une généralisation ou une hiérarchie.
- économie de place en mémoire (propriétés associées à des types généraux hérités par d'autres types plus spécialisés).
- généralisation plus facile de grandes bases de connaissances (bases de données).

- La partition :

regroupe des objets et éléments de relations dans des partitions qui sont organisées de façon hiérarchiques;

Ex : si une partition P1 est au-dessous d'une autre P2, toute chose visible ou présente dans P2 l'est aussi dans P1, sans pour autant l'y avoir été spécifiée.

- Principal intérêt = **quantification**, la représentation du **temps** et de **l'hypothétique**.

10- Forces et Faiblesses des RS:

10.1 Forces des RS:

- les objectifs d'extraction de connaissance dans la base de connaissances s'expriment en **chemins de traversée sur la structure même de la base.**

- possèdent des **principes d'organisation relativement puissants** (généralisation, partition, agrégation) permettant de structurer la base de connaissances.

- **formalisme graphique :** bonne **compréhension**, intéressant à un **premier stade de formalisation de la connaissance.**

- formalisation **déclarative :** finesse et cohérence de représentation des concepts.

10.2 Faiblesses des RS :

- manque de sémantique formelle et de **terminologie standard.**

- interprétation difficile des connaissances : toujours un compromis à faire entre complexité d'une structure de données et complexité de l'interpréteur.

- critique si taille du réseau importante (nb de nœuds et liens) **-> explosion combinatoire.**

Exercice :

Représenter par un réseau sémantique les connaissances suivantes :

- Médor est un chien a 2 ans.
- Poussy est un chat.
- Le chien est un animal carnivore.
- Le chat est un animal carnivore.
- Un carnivore est un animal.
- Un carnivore a des dents pointues.
- Les chiens et les chats mangent la viande.
- Tous ces animaux boivent l'eau.
- Dupont est le propriétaire de Médor et Poussy.
- Possy n'aime pas Médor.

Conclusion :

Un réseau sémantique est un graphe marqué destiné à la représentation des connaissances, qui représente des relations sémantiques entre concepts. Le graphe est orienté ou non orienté. Ses sommets représentent les concepts, et les liens entre les sommets (nœuds) représentent les relations sémantiques, reliant les champs lexicaux. Ce chapitre est représenté les concepts de base du réseau sémantique, l'héritage dans les RS, la partition dans le RS, le RS et Quantification, la représentation d'évènements ou d'actions, l'interprétation des connaissances dans les RS. Ce chapitre se termine par une représentation des avantages et des inconvénients de RS.

Chapitre 04

Représentation Structurée de la Connaissance : Frames et classification

Introduction :

Pour comprendre une scène ou une histoire, il faut trouver une correspondance Entre les éléments perçus et les éléments qui représentent les connaissances. Les schémas fournissent une représentation déclarative pour la connaissance.

Elle permet de raisonner et comprendre les scènes, les histoires (orales ou écrites), les textes, les idées ou les plans.

L'expression de la connaissance déclarative par structure de schéma a plusieurs applications. Trois applications classiques sont les Frames, les Scripts et les Réseaux Sémantiques.

Frames : Un assemblage des schémas pour décrire une scène ou un contexte.

Script : Un assemblage des schémas pour décrire une séquence des événements.

Réseau Sémantique : Un assemblage des schémas pour décrire le sens sémantique d'un texte.

1- Notion de frame (Schéma):

- Le concept de "Frame" était proposé par Minsky (1975) pour guider l'interprétation d'une scène en vision par ordinateur.
- Un frame est une description d'un contexte composé d'une collection de rôles et relations. Ceci donne une description "explicite" du contexte qui peut servir de guider l'interprétation s'une scène, d'une situation ou d'une histoire.
- Les Frames spécifie un contexte pour guider l'interprétation dans un domaine. La structure d'un frame permet de faire les inférences par raisonnement par défaut. Le domaine est composé d'entités.
- Un frame est une structure de données représentant une situation prototypique.
- Un frame est une méthode, un modèle de représentation.
- Lorsqu'il y a beaucoup de connaissances à enregistrer sur les objets ou les catégories d'une application, il vaut mieux représenter ces connaissances dans des "Frames"
- Frame = structure de données générale permettant de **décrire la connaissance** que l'on a sur des **objets** (concrets ou abstraits).
- Un frame est défini par son nom et la liste de ses attributs (*slots*). Chaque attribut est lui-même décrit par un ensemble de facettes (aspects) et leurs valeurs.

(schéma

(attribut-1 (facette-1 valeur-1) ...

(facette-n valeur-n))

...

(attribut-k (facette-1k valeur-1k)...)

)

On peut rattacher **2 types d'informations** à un Frame :

- **des informations relatives à la description des objets : partie déclarative**

• propriétés

• relations d'états

• relations entre les objets, …

- **des informations relatives à la manipulation des objets: partie procédurale**

c.a.d. les opérations (procédures & méthodes) connues par l'objet :

• définition du contexte d'activation de l'objet

• comment calculer une propriété

• action à exécuter dans un contexte donné, …

2- types de Frames :

Un frame décrit une famille (ou classe). Un objet particulier de cette classe est décrit lui-même par un schéma, dit schéma d'instance.

2.1 Les frames « prototypes ou classe » :

- Représentent en **intention** une **classe d'objets**
- Description du **contexte** attaché à cette classe

2.2 Les frames « instances » :

- Réalisations **particulières** d'une classe donnée.
- Description des **individus** d'une classe.
- Schéma d'instance diffère du schéma de classe par la valeur donnée aux attributs.
- Une instance peut être complète ou partielle. Elle hérite du schéma de sa classe a l'aide de la liaison "est_un"

• oiseau → classe : frame **prototype**

• canari → classe : frame **prototype**

• titi → individu : frame **instance**

3. Composants d'un Frame : attributs et aspects

• **attribut** (slot) = information d'un frame permettant d'introduire :

  • des **propriétés** (1 ou n) **décrivant le frame** :

    • dans frame prototype : introduction de **valeurs permises**

    • dans frame instance : introduction de **valeurs effectives**

  • des **relations** (un-aire ou n-aire) **entre frames**

  • un attribut est **structuré** par des **aspects**

• **aspect** = chaque aspect introduit une valeur élément de description de l'attribut

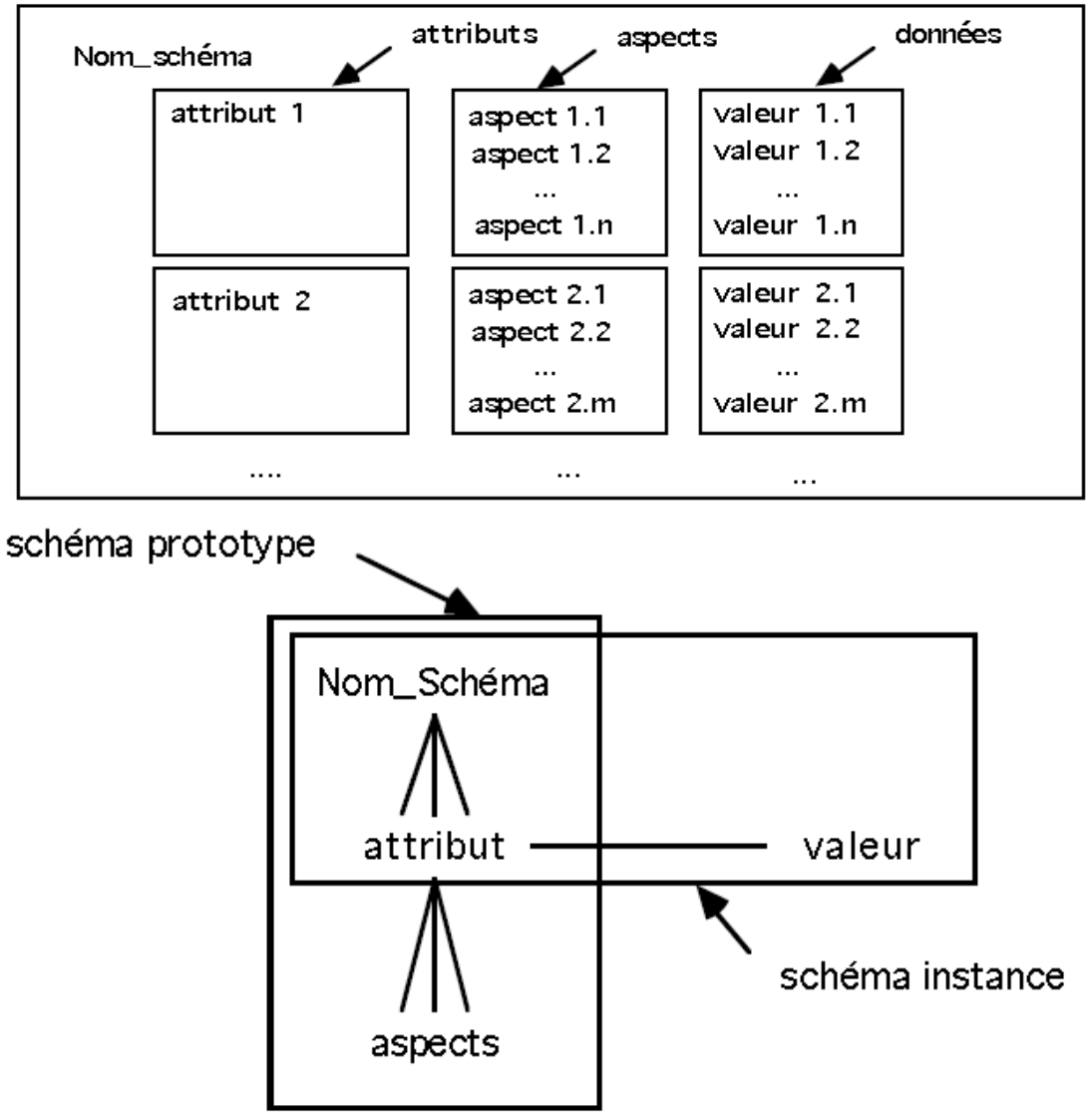


Figure 4.1 : Composants d'un Frame.

• chaque attribut d'une instance doit avoir l'**aspect "valeur"**,

• le frame d'instance ne diffère du frame prototype correspondant que par la donnée supplémentaire de la valeur de l'attribut,

• une instance **hérite** directement du frame de sa classe

• une instance peut être **partielle** ou **complète**,

• une instance de frame est liée à son frame prototype par l'attribut "**est_un**"

• 2 frames prototypes sont liés par l'attribut "**sorte_de**"

Exemple:

[TITI Est-un CANARI with

couleur JAUNE

sexe MALE

age 2

sante EXCELLENT

propriétaire ANNE-SOPHIE]

3.1 Attribut d'un Frame:

un attribut peut :

• permettre **d'introduire des valeurs effectives** (frame instance),

• être un **pointeur sur un frame** plus **général**, plus **spécifique** voire alternatif (vers frame prototype),

• **définir une valeur par défaut**, ou définir un ensemble de valeurs permises (frame prototype),

• **constituer un démon**, sorte de procédure qui peut être activée chaque fois que l'attribut doit être modifié,

• être un **pointeur sur un autre frame** (récursivité de frame), ce qui permet de représenter des objets composés.

3.1.1 Attributs et héritage :

• si la valeur d'un attribut est défini au niveau d'un prototype, tous les frames (prototypes et instances) plus spécifiques en héritent

• les instances héritent des méthodes définies au niveau prototype

Exemple de frame:

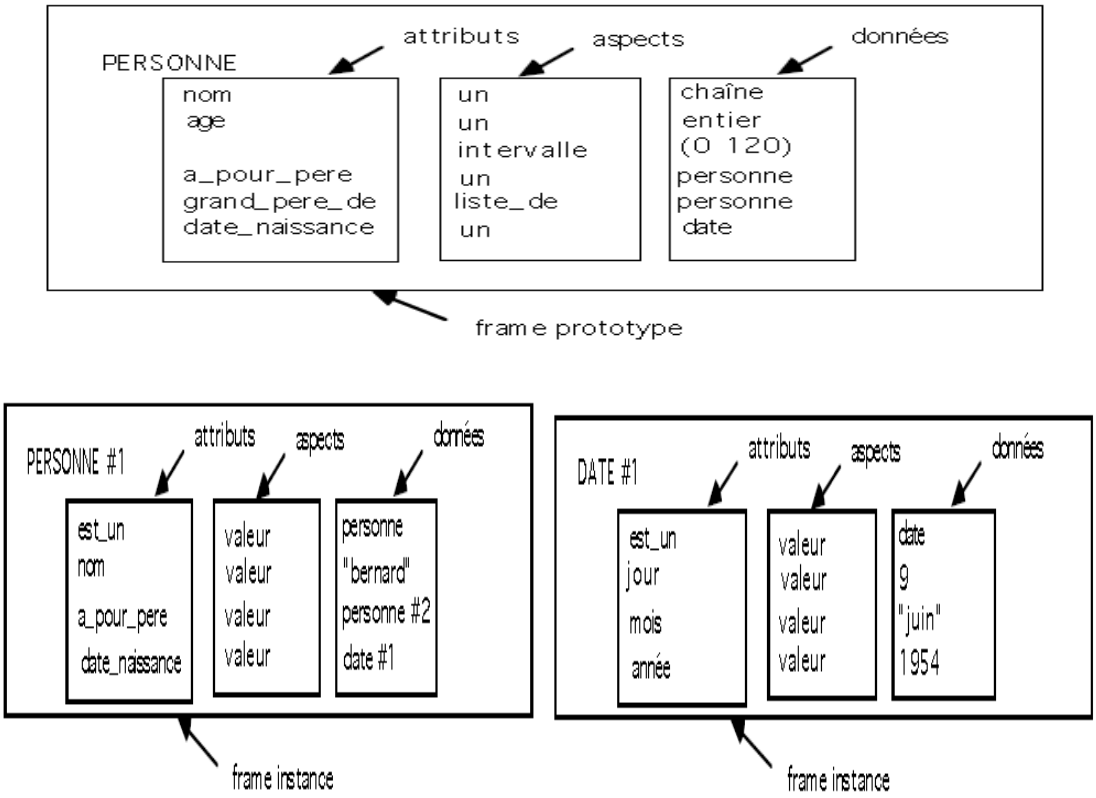


Figure 4.2 : Attributs et héritage dans le frame.

3.2 Les aspects (facette) d'attributs de Frame:

• un attribut est structurés par des aspects (facettes)

• l'ensemble des facettes d'un attribut décrit les diverses connaissances ou points de vues sur cet attribut (sa nature, sa valeur, sa valeur par défaut, des moyens d'obtenir sa valeur). Les valeurs sont elles-mêmes des frames ou des références à des frames.

• chaque aspect introduit **une dimension décrivant l'attribut**

• l'ensemble des aspects est fixé par le langage, c'est ce qui **définit la sémantique de la représentation**

Exemple:

(personne#1

(est-un (valeur Personne))

(nom (valeur "Arthur"))

(a-pour-père (valeur personne#10))

(date-naissance (valeur date#2)))

(date#2

(est-un (valeur Date))

(jour (valeur 14))

(mois (valeur "juillet"))

(année (valeur 1950)))

(Personne

(nom (doit-être chaîne))

(âge (doit-être entier) (intervalle 1 120))

(a-pour-père (doit-être Personne))

(grand-père-de (liste-de personne))

(date-naissance (doit-être Date)))

3.2.1 Les types d'aspects (Les types de facettes):

• plusieurs types d'aspects peuvent être rencontrés :

- **Aspect de typage**
- **Aspect d'obtention de valeur :**
- **Aspect réflexe**
- **Aspect de contrôle**
- **Aspect de communication homme-machine**

-

a) Aspect de typage : permet de définir le type de valeur des attributs d'un frame (valeurs permises). Les facettes "liste-de" et "doit-être" permettent de définir le type des valeurs des attributs.

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

• type composé (Type défini par un frame): référence a une instance de ce frame ou d'un frame plus spécifique. un autre frame de telle nature, frame "personne" dans l'exemple précédent.

peut aussi donner une indication de **restriction de typage**:

• sur un intervalle de valeur; aspect "**intervalle**",

• une mono-valuation ou multi-valuation; aspect "**un**" et "**liste_de**",

• l'aspect "**à_vérifier**" introduisant une procédure à satisfaire pour toute valeur

Exemple:

(Jour

(nom (possible (dimanche lundi mardi mercredi jeudi vendredi samedi)))

(numéro (restriction (plus-grand-que 0)

(plus-petit-que 31))))

b) Aspect d'obtention de valeur: permet de spécifier des procédures d'obtention d'une valeur. Les facettes qui décrivent un moyen d'obtenir la valeur d'un attribut sont:

• aspect "**valeur**": précise la valeur de l'attribut

• aspect "**si_besoin**": permet d'associer des méthodes de calcul des valeurs (attachement procédural).

• aspect "**défaut**": permet d'associer une valeur par défaut qui sera retenue en cas d'absence d'autres informations sur la valeur de l'attribut.

Exemple:

```
(Personne
    (âge   (doit-être entier)
        (si-besoin (compter-années date-jour date-naissance)))
(Planète (atmosphère (doit-être booléen)
            (défaut faux)) )
```

c) Aspect reflexe: permet d'introduire des procédures déclenchées en cas d'ajout, de modification, de suppression d'informations, Elles permettent de maintenir la cohérence de la base d'instances en propageant les modifications:

aspect "si_modif", "**si_ajout**", "**si_supprime**",

Les facettes "si-ajout", "si-modif" et "sisupprime" sont suivies de la description d'une procédure, appelée respectivement en cas d'ajout, de modification et de suppression d'une valeur de l'attribut.

Exemple:

```
(Carré
    (largeur (doit-être réel)
       (si-modif (calcule-surface largeur)) )
    (surface (doit-être réel)) )
```

d) Aspect de contrôle:

permet d'introduire des procédures définissant des actions à déclencher en cas d'échec ou de succès d'obtention de valeur, pouvant conduire à une propagation dans d'autres frames. Par exemple:

aspect "**si_succés**" , "**si_échec**", ...

e) Aspect de communication Homme-machine: permet d'introduire des procédures permettant de passer d'une vision interne à une vision externe des valeurs ou réciproquement:

aspect "**lire**", "**écrire**", ...

4. Héritage dans les frames :

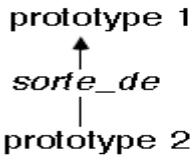
4.1 Héritage simple :

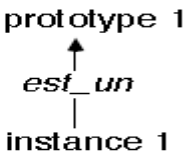
Les frames s'inscrivent dans une structure de demi-treillis:

• un frame peut **dominer** un ensemble de frames plus spécifiques et

• un frame peut **être dominé** par un ou plusieurs frames plus généraux

Deux situations d'héritage simple:





• d'un frame instance de son frame prototype (relié par un attribut "est_un")

• d'un frame prototype d'un autre frame prototype (relié par un attribut "sorte_de"

D'une façon générale :

• si la valeur de l'attribut est défini au niveau d'un **prototype**, **tous les frames prototypes et instances plus spécifiques en héritent**

• il y a aussi héritage des **procédures** ou **méthodes** définies au niveau des prototypes au niveau des instances.

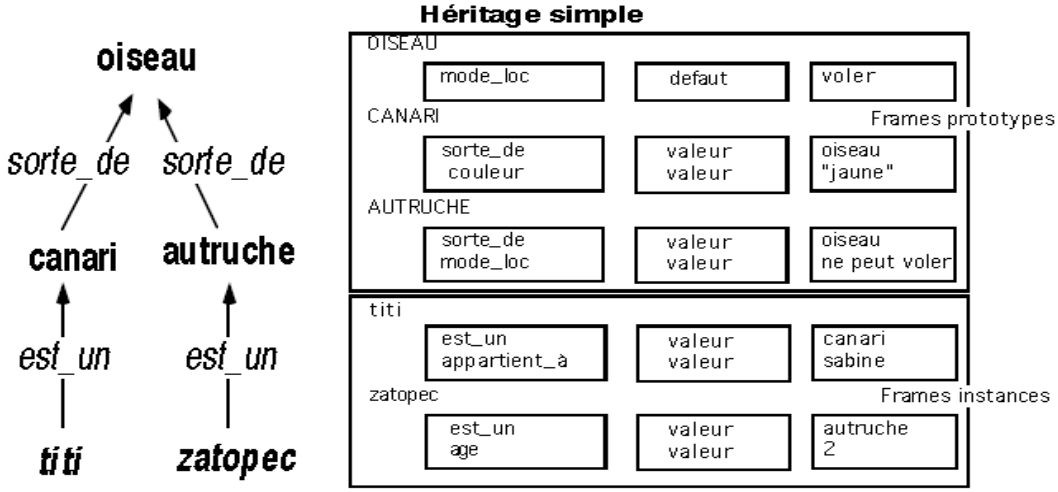


Figure 4.3 : Héritage simple Vs. frame.

4.2 Héritage multiples :

• **lorsqu'un schéma est dominé par plusieurs autres frames**

• **définir l'ordre d'application des héritages**,par exemple par une liste de précédence ou une énumération des frames à héritage:

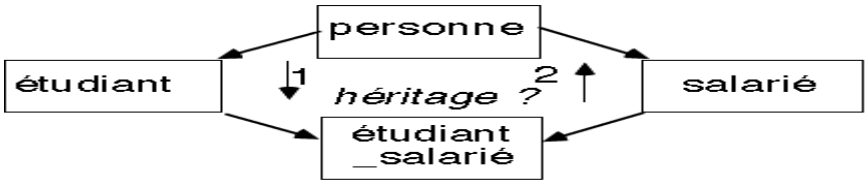


Figure 4.4: Héritage multiples Vs. Frame.

• sens de parcours possible pour l'héritage :

1) étudiant_salarié

étudiant

personne

salairé

2) étudiant_salarié

salairé

personne

5. Inférence dans les frames:

5.1 Inférence d' instanciation dans les frames:

• **l'instanciation de frames** de type instance constitue **l'inférence fondamentale** dans les systèmes de frames.

• consiste à "**construire**" ou "**compléter**" une instance de frame = **instancier par des valeurs les attributs définis dans les frames prototypes** auxquels est associé l'instance de frame considérée.

• cette obtention de valeurs peut être **directe**; valeurs d'attributs propres à l'instance,ou par héritage.

• Si l'on appelle "**vue**" l'ensemble des attributs que possède un frame,une **instance de ce frame sera l'union de toutes les vues propres et héritées**:

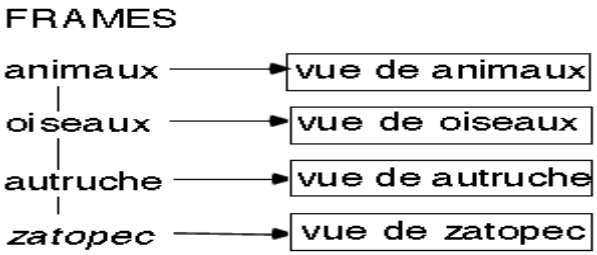


Figure 4.5: Héritage multiples Vs. Frame.

5.2 Inférence par filtrage dans les frames:

• intervient quand **l'aspect valeur d'un frame prototype renvoie sur un autre frame**

• ce dernier frame intervient alors comme un **filtre**:

- **décrivant les instances possibles** : valeurs que cet attribut pourra prendre
- **contenant généralement des conditions supplémentaires à vérifier** sur les valeurs de ce frame restreignant encore les instanciations possibles.

Exemple:

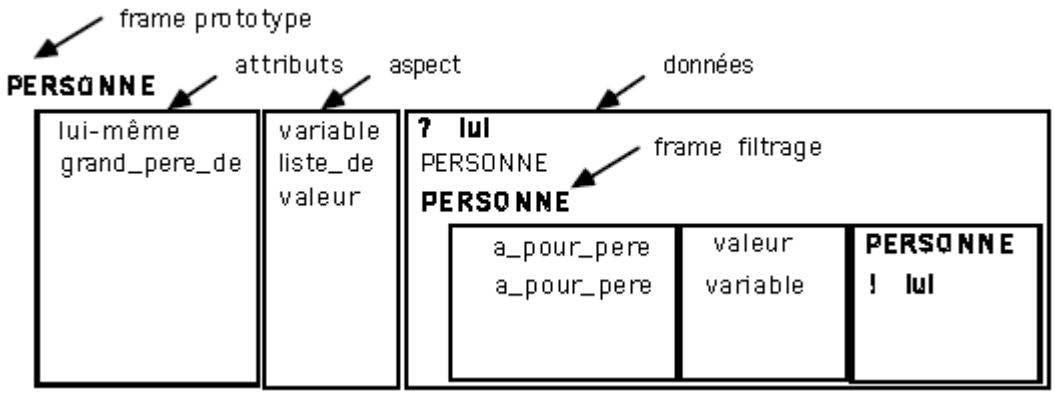


Figure 4.6: Inférence par filtrage dans les frames.

dans "**! X**", préfixe "! ..." valeur de la variable X doit être instanciée à l'appel du filtre (**variable d'entrée**)

• dans "**? X**", préfixe "? ..." valeur de la variable X peut être inconnu à l'appel du filtre (**variable de sortie**)

Dans cet exemple:

- l'attribut "**lui_même**" du frame prototype principal représente le frame,

- l'aspect "variable" de cet attribut nomme une variable "**?lui**" qui représente l'attribut en question,

- à l'instanciation, dès que l'attribut reçoit une valeur, cette valeur est affecté à la variable,

- le déclenchement du filtre peut conduire à un retour arrière.

5.3 Inférence procédurale dans les frames :

• **obtenue par activation des procédures**, **méthodes** associées aux attributs afin d'en obtenir des valeurs d'instanciation.

• ces procédures sont elles-mêmes décrites par des frames prototypes, dont les attributs sont leurs valeurs d'entrée et de sortie.

• c'est l'aspect "**si-besoin**" qui distingue un frame de filtrage d'une telle procédure:

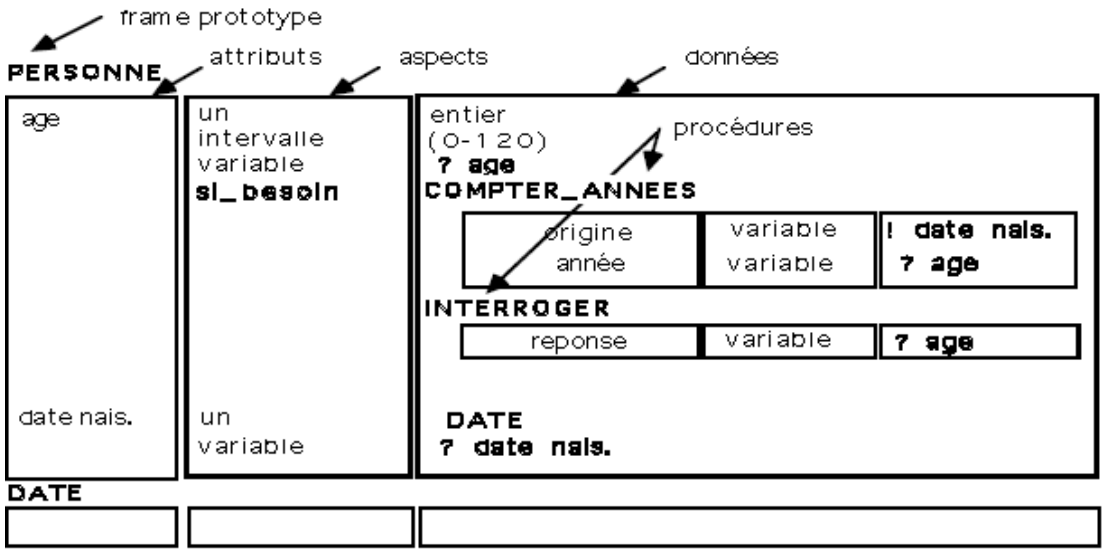


Figure 4.7: Inférence procédurale dans les frames.

Exemple :

• l'âge est calculé à partir de la date de naissance "date nais." par la procédure COMPTER_ANNEES.

• Si cette date de naissance n'est pas connue, on interroge l'utilisateur pour obtenir l'âge avec la procédure INTERROGER.

6. Un modèle de langage de frames

6.1. Anatomie d'un frame

Un frame est une structure à trois niveaux, ***frame-attribut-facette***, utilisée pour représenter un concept et ses différentes propriétés. "Un attribut, appelé aussi propriété, est décrit par des facettes qui spécifient la nature de l'attribut et le comportement qui lui est associé**. Les facettes déclaratives** décrivent la valeur de l'attribut tandis que **les facettes procédurales** introduisent des réflexes qui se déclenchent lors des accès à l'attribut. Les réflexes se divisent en réflexes en lecture et en écriture ; les seconds se subdivisent à. leur tour en réflexes a priori ou a posteriori, selon qu'ils sont activés avant ou après la modification de la valeur de l'attribut auquel ils sont associés. Toute action sur la valeur d'un attribut résulte d'un accès à cet attribut. Ce style de programmation est appelé ***programmation dirigée par les accès.***

6.2 Les facettes de typage et de restriction de type

Les facettes déclaratives se divisent en facettes de typage, de restriction de type, de cardinalité et de valeur. Les facettes de typage ***$un*** et ***$liste-de*** définissent le type d'un attribut, respectivement monovalué et multivalué.

Une restriction de type est indiquée par la facette ***$intervalle*** pour un type numérique et ***$domaine*** pour un type donné en extension. Les langages de frames sont généralement à typage dynamique : même si le type d'un attribut est déclaré statiquement, la validité de la valeur d'un attribut n'est vérifiée qu'à l'exécution. Les facettes de cardinalité ***$min*** et ***$max*** précisent respectivement le nombre minimal et maximal de valeurs que peut prendre un attribut.

La figure 4.8 montre la représentation des frames ***Point*** et ***Fenetre***, définis grâce à la primitive ***defmodele*** comme des spécialisations du frame le plus général ***Objet- ideal***. Un point est composé d'une abscisse et d'une ordonnée, l'abscisse pouvant prendre des valeurs entières entre a et 1000, l'ordonnée des valeurs entières entre a et 800. Une fenêtre située sur un écran est définie par une origine, qui est le point situé en haut et à gauche, et par un coin, qui est le point situé en bas et à droite. La donnée de l'origine est obligatoire, comme l'indiquent les facettes de cardinalité alors que celle du coin ne l'est pas.

```
(defmodele Point
  (sorte-de
    ($valeur Objet-ideal))
  (abscisse
    ($un entier)
    ($intervalle [0 1000]))
  (ordonnee
    ($un entier)
    ($intervalle [0 800]))
  (region
    ($liste-de Fenetre)
    ($lien-inverse origine)))

(defmodele Fenetre
  (sorte-de
    ($valeur Objet-ideal))
  (origine
    ($un Point)
    ($min 1)
    ($max 1)
    ($lien-inverse region)
    ($si-ajout dessiner-cadre)
    ($si-enleve effacer-cadre))
  (coin
    ($un Point)
    ($defaut Bas-droit)
    ($si-possible inferieur-a-origine)
    ($si-enleve effacer-cadre))
  (hauteur
    ($un entier)
    ($si-possible nil)
    ($si-besoin +hauteur))
  (largeur
    ($un entier)
    ($si-possible nil)
    ($si-besoin +largeur))
  (etiquette
    ($un chaine)
    ($si-besoin demander)
    ($si-ajout afficher-etiquette)))
```

Figure 4.8: Les frames représentant les modèles **Point** et **Fenetre**.

6.3 Les facettes de valeurs

Les facettes ***$valeur*** et ***$defaut*** donnent respectivement la valeur courante, appelée ***valeur fixe***, et la valeur par défaut de l'attribut. Par opposition à une valeur fixe, une valeur par défaut est "faiblement" couplée avec un attribut : elle est héritée par les spécialisations du frame ct n'est prise en compte qu'en l'absence de toute autre information. Ainsi, pour créer une fenêtre particulière, seule son origine doit être connue et donnée explicitement par l'utilisateur. Le coin d'une fenêtre est par défaut le point ***Bas-droit***, qui est supposé avoir pour coordonnées (1000 800), mais l'utilisateur a la possibilité de choisir n'importe quel autre point qui ne soit pas inférieur, pour la relation d'ordre définie sur les points, à l'origine de la fenêtre (l'origine d'une fenêtre est supposée être le point supérieur gauche, le coin le point inférieur droit).

6.4 Les relations

Une *relation* permet de créer une dépendance entre deux frames. La relation et sa relation inverse sont définies grâce à deux liens orientés, réciproques l'un de l'autre, chaque lien étant représenté par un attribut muni d'une facette de typage et d'une facette ***$lien-inverse***. Le frame "de départ", auquel appartient l'attribut définissant la relation, s'appelle le ***domaine*** de la relation, tandis que le frame "d'arrivée", dans lequel la relation prend ses valeurs, s'appelle ***co-domaine*** de la relation. Deux attributs définissant une relation sont symétriques : chaque fois que la valeur d'un des attributs est modifiée, celle de l'attribut avec lequel il est en relation l'est automatiquement. Par exemple, lorsqu'une fenêtre ***F1*** est créée, la donnée de l'***origine*** de ***F1***, notée ***O-F1***, provoque la mise en place de la valeur ***F1*** dans l'attribut ***region*** associé au point **O-F1**. Réciproquement, donner la valeur ***F1*** à l'attribut ***region*** du point noté ***P1*** provoque la mise en place de la valeur ***P1*** dans l'attribut ***origine*** associé à l'objet décrivant ***F1***, et par conséquent la création de la fenêtre.

Dans certains cas, il peut être utile de déclarer que deux attributs ont la même valeur, pour exprimer par exemple qu'un frère et une sœur ont les mêmes parents, qu'une épouse a les mêmes enfants que son mari. Ce type de connaissance est représenté comme un cas particulier de relation, en remplaçant la facette ***$lien-inverse*** par la facette ***$meme-que***. Cette dernière facette est suivie d'une valeur ou bien d'une liste de valeurs et joue un rôle similaire à la facette ***$lien-inverse***, en contrôlant que la valeur des deux attributs en relation évolue de la même façon. La valeur d'un attribut

pointée par une facette ***$meme-que*** peut elle-même pointer vers un autre attribut et ainsi de suite. Cette façon de faire permet donc de définir un frame relativement à d'autres frames, augmentant ainsi les possibilités de partage d'informations et facilitant les mises à jour.

6.5 Un réflexe en lecture

Le réflexe si-besoin est le réflexe en lecture standard. Il est introduit par la facette ***$si -besoin*** et sert à calculer la valeur de l'attribut auquel il est associé, lorsqu'il existe une méthode de calcul pour le faire. Il est déclenché à la lecture de la valeur de l'attribut, lorsqu'aucune valeur n'est présente. Par exemple, les valeurs des attributs ***hauteur*** et ***largeur*** sont calculées grâce aux réflexes supposés définis par ailleurs ***+hauteur*** et ***+largeur***, uniquement par nécessité.

6.6 Les réflexes a priori et a posteriori en écriture

Le réflexe ***a priori*** standard est introduit par la facette ***$si-possible***. Il sert à vérifier la validité d'une valeur à mettre en place. Il est généralement à valeur booléenne et interdit toute écriture s'il retourne faux. Un tel réflexe est surtout utilisé pour implanter des contraintes qui ne peuvent pas être exprimées simplement à l'aide d'une facette de typage car elles nécessitent un calcul ou bien mettent en jeu les valeurs de plusieurs attributs. Le réflexe ***si-possible*** associé à l'attribut ***coin*** vérifie que le point donné en valeur a une abscisse et une ordonnée qui sont supérieures, pour la relation d'ordre définie sur les points, à celles du point donné comme origine.

Les facettes ***$si-ajout*** et ***$si-enleve*** introduisent les deux réflexes ***a posteriori*** standard. Un réflexe si-ajout est activé après une première affectation ou après une modification de valeur, un réflexe si-enlève l'est en cas de suppression de valeur. Par exemple, une fois que l'origine d'une fenêtre particulière est connue, le réflexe ***dessiner-cadre*** est activé pour dessiner le cadre de la fenêtre. Si cette origine est supprimée, le réflexe ***effacer-cadre*** est activé pour effacer cette fenêtre. De tels réflexes sont généralement utilisés pour propager des valeurs en cas de modification de l'environnement. Ils peuvent également servir à tracer l'historique de la valeur d'un attribut en mémorisant dans une facette ou une variable spéciale toute modification de valeur.

6.7 Spécialisation et partage de propriétés

L'ensemble des frames est organisé en une ***hiérarchie* ou *graphe d'héritage***, dans lequel un frame est une spécialisation d'un ou de plusieurs superframes, dont il hérite les couples attribut-facette. L'héritage est dit multiple si un frame peut avoir plusieurs superframes, et c'est généralement le cas, sinon il est dit simple. Un frame hérite les caractéristiques de ses ancêtres selon un ordre donné par la *liste de priorité* du frame. En héritage simple, la liste de priorité n'est autre que la fermeture transitive de la relation ***sorte-de***. En héritage multiple, cette fermeture transitive n'est plus un ordre total et il faut avoir recours à des stratégies de linéarisation qui s'appuient sur des critères comme l'ordre et la multiplicité de la relation d'héritage, ainsi que sur la modularité du graphe d'héritage.

L'héritage des couples attribut-facette est dynamique : les couples hérités ne sont pas recopiés dans le frame mais sont consultés dans les superframes, lorsque c'est nécessaire. L'héritage dynamique favorise la représentation de connaissances incomplètes et celle d'objets dont le comportement et la structure évoluent au cours du temps. Toute modification de couple attribut-facette dans l'un des superframes se répercute automatiquement aux spécialisations qui héritent le couple. Lorsque le partage de propriétés est statique, comme dans certains langages de classes, il est nécessaire de prévoir la mise à jour des descendants des objets modifiés.

6.7.1 Enrichissement, masquage et sémantique des facettes déclaratives

Un frame est spécialisé par ***enrichissement*** lorsqu'il est doté de nouveaux couples *attribut-facette*. Si l'attribut d'un nouveau couple est déjà défini dans un des superframes, la nouvelle facette complète la liste des facettes existantes. Par exemple, le point ***Origine-horloge*** de coordonnées (900 0) est la valeur par défaut donnée à l'attribut ***origine*** du frame ***Horloge***, sous-frame de ***Fenetre*** (figure 4.9). De même, la valeur de l'attribut ***etiquette*** est fixée dans le frame ***Horloge*** alors qu'elle ne l'est pas dans le frame ***Fenetre***.

Un frame est spécialisé par ***substitution*** lorsque certaines facettes des couples ***attribut-facette*** hérités sont ***masquées***, la valeur de la facette du couple héritée est redéfinie. Ainsi, le coin par défaut ***Bas-droit*** de coordonnées (1000 800), déclaré par la facette ***$defaut*** associée à. l'attribut ***coin*** de ***Fenetre*** est masquée dans ***Horloge*** par le point ***Coin-horloge***, de coordonnées (1000 100).

N.B : Le masquage ne porte que sur les facettes déclaratives et sur la facette ***$si-besoin.***

```
(defmodele Horloge
  (sorte-de
    ($valeur Fenetre))
  (origine
    ($defaut Origine-horloge)
    ($si-ajout afficher-heure)
    ($si-enleve effacer-heure))
  (coin
    ($defaut Coin-horloge)
    ($si-enleve effacer-heure))
  (etiquette
    ($valeur "*Horloge*")))
```

Figure 4.9: Le frame Horloge est un sous-frame de **Fenêtre**. Par défaut, une horloge s'affiche dans le coin supérieur droit de l'écran.

7. Forces et Faiblesses :

7.1 Forces :

• Très **séduisant** et **pratique** pour **exprimer la connaissance**

• Structure **plus souple** et **plus efficace** que les représentations réseaux sémantiques, règles de production, prédicats, procédures.

• Utilisé pour le **développement d'ontologies** (Protégé - Frame)

• Dans de nombreux **systèmes experts**

• ! Extension du concept de frame (R.C.Schank et R.Abelson), **scripts** = séquence

• d'évènements dans un contexte particulier, associés au concept de sénario

• …

7.2 Faiblesses :

• **Pas d'équivalence avec la logique** (ex : quantifieurs, disjonctions, …)

• **Pas d'expression de connaissances incertaines**, **imprécises**,**hypothétiques**

• Certaines **inférences** sont favorisés, d'autres sont **difficiles à réaliser**

• …

8. Langages de frames (Quelques implémentations de fames):

De nombreuses implémentations de frames ont été réalisées en langage LISP ou Prolog

• **FRL:** "Frame Représentation Language" [ROBERTS & GOLDSTEIN 1977] (développé au MIT : un précurseur, assez rudimentaire)

- les frames entre eux sont organisés hiérarchiquement
- une base de connaissances en FRL = {frames} dont les attributs peuvent avoir les aspects suivants: valeurs limites, valeurs par défaut, contraintes et procédures (développées en LISP) pouvant être déclenchées quand une valeur sera dépassée ou non dépassée, voire utilisée par un autre attribut.

• **KRL:** (BOBROW & WINOGRAD - Xerox PARC)

- s'inspire directement des idées de M.MINSKY, est plus ambitieux que le précédent.
- il dispose d'opération d'unification de frames qui peuvent être contrôlées par le concepteur de la base de connaissances
- il possède aussi la notion de "contexte de croyance" pouvant servir à définir un "mécanisme d'attention" (attention focusing mechanism)
- il inclut de la "self Knowledge" dans le base de connaissance en fournissant des descriptions de descriptions.

• les "unit" de KRL:

- toute entité conceptuelle à représenter est décrite par un frame appelé "**unit**" appartenant à une "catégories" (7 catégories dans KRL) définissant son niveau d'abstraction: objet générique, spécialisé ou élémentaire
- une unit est définie par un ensemble d'attributs, décrits par de descripteurs.
- les concepts généraux sont décrits par des units appartenant aux catégories "**basic**" et "**abstract**" :
  - les catégories "**basic**" définissent des particules de connaissances en arbre d'héritage simple dont elles sont les racines, ces particules peuvent

être instanciées

EX : définition de l'unit basic "CHAT":

Chat UNIT Basic

<SELF>

<nom (a string)>

<race (a Race) Persan; DEFAULT) >

<age (an integer)>

<propriétaire (a string)>]

• les catégories "**abstract**" définissent des particules de connaissances plus générales qui ne peuvent être spécialisées.

• **KL-ONE** (Brachman & Schmolze, 1985) ;

• **Graphes conceptuels** (Sowa, 1984) ;

• **CYCL** (Lenat & Guha, 1990).

Conclusion :

Ce chapitre a examiné les principales caractéristiques des langages de frames. Cette dernière permet de construire et de gérer des représentations à objets où les connaissances relatives à un domaine sont décrites par une hiérarchie de frames. Le vocabulaire utilisé est principalement emprunté au langage de frames formel. Ce chapitre a présenté d'abord le noyau minimal d'un langage de frames, puis les extensions qui peuvent y être apportées.

Chapitre 05

Logique de description

Introduction :

Les logiques de description aussi appelées logiques descriptives (LD) sont une famille de langages de représentation de connaissance qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée. Le nom de logique de description se rapporte, d'une part à la description de concepts utilisée pour décrire un domaine et d'autre part à la sémantique basée sur la logique qui peut être donnée par une transcription en logique des prédicats du premier ordre. La logique de description a été développée comme une extension des langages à cadres, une famille de langage de programmation pour l'intelligence artificielle, et des réseaux sémantiques, qui ne possédaient pas de sémantique formelle basée sur la logique. Les logiques de description ont été conçues à partir des réseaux sémantiques de Quillian qui sont des graphes orientés étiquetés dans lesquels on associe des concepts aux nœuds et des relations aux arcs, et de la sémantique des cadres de Minsky où l'on a des concepts représentés par des cadres qui sont caractérisés par un certain nombre d'attributs (appelés aussi slots) qui contiennent de l'information sur leur contenu.

Les logiques de description forment une famille de langages de représentation de connaissances qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application d'une façon structurée et formelle. Le nom « logique de description » peut être interprété de deux manières. D'une part, ces langages ont été élaborés pour écrire la « description » des concepts pertinents d'un domaine d'application. D'autre part, une caractéristique cruciale de ces langages est qu'ils ont une sémantique formelle définie en logique du premier ordre (à la différence des propositions précédentes comme les cadres de Minsky). Dans ce sens, nous pouvons dire que les LDs ont une sémantique « descriptive » formelle.

1- Définition de Logique de description :

- est en fait une famille de formalismes pour représenter une base de connaissances d'un domaine d'application (une famille de langages de représentation de connaissances "RC")
- définit les concepts (aussi appelés *classes*) pertinents d'un domaine (terminologie)
- utilise les concepts pour spécifier les propriétés des objets, des individus dans le domaine et les relations (aussi appelées *rôles*) qui peuvent être établies entre les instances de ces classes
- a une sémantique logique formelle pour déduire des connaissances implicites via la classification de concepts et d'individus.

2- Les éléments de base de LD (Vocabulaire de LD):

- Concepts: pour déclarer des entités, classes.**(Constantes** : $\top, \bot$ ou **Concepts:** *A, B, C, D*; Etudiant ..)

- Objets: instances des classes (Ex : pierre).

- Rôles: pour déclarer des propriétés, des relations, *R :*relations binaires (Ex : possedeAmi)

- Constructeurs: pour définir les expressions des concepts: $\neg$, $\cap$,…

- Quantificateurs: $\forall$, $\exists$

3- syntaxe d'une LD

$\top$: concept universel

$\bot$: concept impossible

A : concept atomique

$\neg A$: négation d'un concept atomique

$C \cap D$: intersection de concepts quelconques

$\forall R.C$: restriction de valeurs pour des concepts quelconques

$\exists R.\top$: quantification existentielle limitée

4- Structure générale de système LD:

Dans une base de connaissances en logique descriptive, on distingue deux composantes : la *TBox* et la *ABox*.

TBox (**le niveau terminologique)*:*** est un ensemble d'axiomes "terminologiques" ,elle contient tous les axiomes définissant les concepts du domaine

ABox(**le niveau factuel ou ABox** ***) :*** est un ensemble d'axiomes "assertionnels", elle contient des assertions sur des individus, en spécifiant leur classe et leurs attributs

Donc Base de Connaissances (*BC)* = *TBox* $\cup$ *ABox*

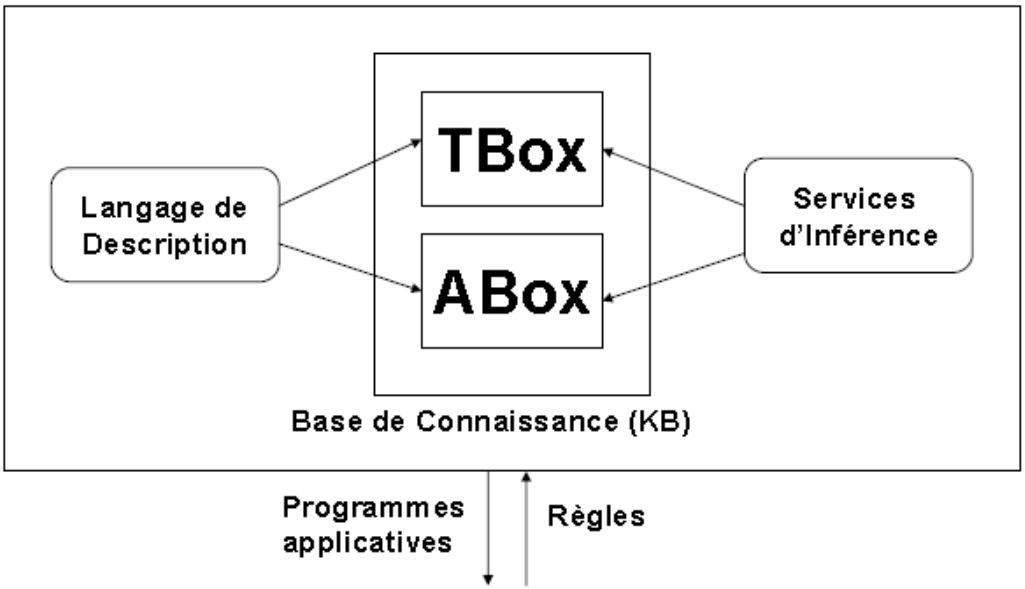


Figure 5.1: Structure générale de système LD.

4 .1 La TBox :

a) La TBox contient :

- **Les Entités Atomiques :** concepts atomiques et rôles atomiques constituant les entités élémentaires de la LD

- **4 Concepts et Rôles atomiques prédéfinis minimaux:**

  - le **concept** $\top$ **et le rôle** $\top_R$**,** les plus généraux de leur catégorie
  - le **concept** $\bot$ **et le rôle** $\bot_R$**,** les plus spécifiques (ensemble vide)

- **Les Entités Composées :**

  - **concepts** et **rôles atomiques** peuvent être **combinés** au moyen de **constructeurs** pour former des entités **composées**

Conventions :

- ***A*** *et* ***B*** *dénotent des* ***concepts atomiques***
- ***C*** *et* ***D*** *dénotent des* ***concepts composés***
- ***R*** *dénote un* ***rôle***
- *les nom de* ***concepts*** *commencent par une* ***Majuscule*** *: Ex : Homme, Femme, ..*
- *les noms de* ***rôles*** *par une* ***minuscule*** *: Ex : relationParentEnfant, …*

AL(Attributive Language): logique de description de base : Axiomes

b) Les axiomes terminologiques d'une des 2 formes :

- **C** $\equiv$ **D** : énonçant des **relations d'équivalence** (de définition) entre concepts : *C équivalent par définition à D*

- **C ⊑ D** : énonçant des **relations d'inclusion :** *C est inclus dans D*

c) Consistance de concepts :

- un **concept** (une classe) est **consistant**, s'il existe **au moins un individu membre de cette classe :**

Remarque: si on définit une classe (concept) comme étant à la fois une sous-classe des classes Homme et Femme, et que la TBox spécifie aussi que ces 2 classes sont disjointes (aucun individu ne peutà la fois être un Homme et une Femme) : ce nouveau concept est alors inconsistant.

d) Subsomption de concepts:

la **subsomption** consiste à **déduire qu'un concept, c-a-d une classe, est une sous-classe d'une autre classe**, même si ce n'est pas déclaré explicitement dans la base de connaissances:

Ex: $\begin{matrix} \text{Woman} \sqsubset \text{Person} \\ \text{Person} \sqsubset \text{Animal} \end{matrix} \Rightarrow \text{Woman} \sqsubset \text{Animal}$

4.2 *La ABox:*

la ABox utilise les axiomes de la TBox ,une ABox contient 2 types d'assertions sur des individus :

- **des assertions d'appartenance** : spécifiant leur classe et leurs attributs :

Ex : Marie est une femme et qu'elle a 2 enfants ; Marie est une Mère (individu instance de la classe mère)

C(*a*) avec *C* : concept et *a* : individu

- des assertions de rôle : spécifiant les relations existantes entre individus:

Ex : une mère doit avoir au moins un enfant : la ABox devra contenir au moins un autre individu, et une relation entre celui-ci et Marie indiquant qu'il est un de ses enfants.

R(*a*, *b*) avec *R* : rôle, *a*, *b* : individus

Convention :

les individus nommés sont représentés par des lettres a, b

Exemple01 :

- Soit les définitions suivantes

TBox
- Woman ⊏ Person
- Mother ≡ Woman ⊓ ∃hasChild.Person

ABox
- hasChild(MARY,PETER)
- Woman(MARY)
- Person(PETER)

- Déduire de nouveaux faits
  - Person(MARY)
  - Mother(MARY)

Exemples02 :

concepts atomiques : *Personne*, *Homme*

rôles atomique : *aEnfant*

individus : anne, paul

axiomes:

Personne ⊑⊤ *Homme* ⊑*Personne*

Femme ≡ *Personne* ∩ ¬ *Homme*

assertions :

Femme(*anne*)

aEnfant(*anne, paul*)

Exemple03:

concepts atomiques : *Personne*, *Homme*

rôles atomique : *aEnfant*

femme : *Personne* ∩ ¬ *Homme*

personnes qui ont au moins un enfant :

Personne ∩ ∃*aEnfant*. ⊤

personnes dont tous les enfants sont des hommes :

Personne ∩ ∀*aEnfant.Homme (erreur)*

Le problème, c'est qu'un personne qui n'a pas d'enfants serait aussi une instance de ce concept.

Réponse correcte: Personne ∩ ∃*aEnfant*. ⊤ ∩ ∀*aEnfant.Homme*

personne qui n'a pas d'enfant : *Personne*∩ ∀*aEnfant*.⊥

Exemple04:

définition de concepts

concepts atomiques : Homme

rôles atomique : aEnfant, marieAvec

Parent ≡ ∃ aEnfant.⊤

ParentDeFemme ≡ ∃ aEnfant.⊤ ⊓ ∀ aEnfant.¬Homme

Célibataire ≡ ∀ mariéAvec.⊥

HommeMarié ≡ Homme ⊓ ∃ mariéAvec.⊤

5- Sémantique du langage AL

Les LD sont une famille de langage basée sur le langage *AL (Attributive Language)* introduit par Schmidt-Schauβ en 1991. La sémantique du langage *AL* fait appel à la théorie des ensembles. Essentiellement, à chaque concept est associé un ensemble d'individus dénotés par ce concept. Une interprétation suppose donc l'existence d'un ensemble non vide Δ qui représente des entités du monde décrit. Nous avons une fonction d'interprétation I qui associe à chaque description un sous-ensemble de Δ. Supposons que pour chaque concept atomique A, la fonction $I(A)$ associe un sous-ensemble $I(A) \subseteq \Delta$,et pour chaque rôle atomique R,une relation binaire $I(R) \subseteq \Delta \times \Delta$.

Interprétation :

I interprétation : $(\Delta^I, I(f_A), I(f_R))$

Δ^I : domaine

Fonction $f_A{}^I: A \rightarrow I(A) \subseteq \Delta$

fonction $f_R{}^I: R \rightarrow I(R) \subseteq \Delta^I \times \Delta^I$

Voici comment est définie cette fonction d'interprétation pour les autres descriptions possibles :

$I(\top) = \Delta$

$I(\bot) = \varnothing$

$I(\neg A) = \Delta \backslash A^I$

$I(C \sqcap D) = I(C) \cap I(D)$

$I(\forall R.C) = \{a \in \Delta \mid \forall b, (a, b) \in I(R) \rightarrow b \in I(C)\}$

$I(\exists R.\top) = \{a \in \Delta \mid \exists b, (a, b) \in I(R)\}$

Exemple 1:

$\Delta = \{a, b, c, d, e, f, g\}$

$I(\text{Homme}) = \{a, b, c, g\}$

$I(\text{aEnfant}) = \{(a, c), (b, d), (b, e), (c, g)\}$

I(mariéAvec) = {(b, f), (f, b)}

Avec cette interprétation, on obtient les ensembles suivants pour chacune des classes définies:

Parent : {a, b, c}

ParentDeFemme : {b}

Célibataire : {a, c, d, e, g}

HommeMarié : {b}

Exemple 2:

Δ = {a}

I(Homme) = {}

I(aEnfant) = {}

I(mariéAvec) = {}

Il s'agit d'une interprétation qui correspond à un monde qui ne comprend qu'un seul individu qui est une femme célibataire n'ayant pas d'enfants.

6- Les constructeurs de la famille AL :

6.1 Constructeur d'union U:

Syntaxe : $C \sqcup D$

Sémantique : $I(C \sqcup D) = I(C) \cup I(D)$

Exemples:

$Personne \equiv Homme \sqcup Femme$

Représentation d'un magasin qui ne vend que des chats et des chiens:

$Magasin \sqcap \forall vend.(Chat \sqcup Chien)$

6.2 quantification existentielle complète ∃:

Dans le langage AL, on peut utiliser le quantificateur existentiel pour spécifier qu'une entité doit avoir au moins une relation avec un autre objet, mais on ne peut pas spécifier la classe de cet autre objet.

Syntaxe: $\exists R.C$

Sémantique: $I(\exists R.C) = \{a \in \Delta \mid \exists b, (a, b) \in I(R) \wedge b \in I(C)\}$

Exemple: la classe des gens qui possèdent au moins un chien: $\exists$ possède. Chien

6.3 Constructeur de fonctions F:

Un autre ajout utile (dénommé F) est la possibilité de spécifier qu'un rôle est en fait une fonction, c'est-à-dire une relation telle que aucune entité ne peut être reliée à plus d'une autre entité par cette relation.

Sémantique: si $(x, y) \in R^I$ et $(x, z) \in R^I$ alors $y = z$

Exemple: la relation mariéAvec, qui nous permettrait de définir le concept HommeMarié :

HommeMarié $\equiv$ Homme $\sqcap \exists$ mariéAvec. $\top$

Si on ne spécifie rien de plus, rien n'empêche une instance de cette classe d'être mariée avec plus d'une personne. Pour empêcher cela, il nous faut une manière de spécifier que la relation mariéAvec en fait une fonction. On écrira donc un axiome de la forme Fun(R) pour dire qu'un rôle R est une fonction.

la relation mariéAvec soit une fonction :

HommeMarié $\equiv$ Homme $\sqcap \exists$ mariéAvec. $\top \sqcap \leq 1$ mariéAvec

Cet axiome spécifie qu'un homme marié doit être marié avec exactement une personne. Mais il s'agit d'une restriction qui ne s'applique qu'à cette description.

On pourrait, par exemple, définir le concept de femme mariée de manière à permettre qu'une femme ait plusieurs maris, et ce en utilisant la même relation :

FemmeMariée $\equiv$ Femme $\sqcap \exists$ mariéAvec.Homme

Propriétés:

$\geq 1\, R \equiv \exists R. \top$

$\geq 0\, R \equiv \top$

$\leq 0\, R \equiv \forall R. \bot$

6.4 Constructeur d'inversion I :

Syntaxe : R^-

Sémantique : $I(R^-) = \{(y, x) \mid (x, y) \in I(R)\}$

Exemple: si Maria regarde Paulo, on sait aussi que Paulo est regardé par Maria. Si on utilise deux relations regarde et estRegardéPar, on aimerait que tout fait de la forme regarde(x, y) implique nécessairement le fait estRegardéPar(y, x). Pour ce faire, on utilisera le constructeur d'inversion (dénommée par $^-$) :

$estRegardéPar \equiv regarde^-$

6.5 Constructeur de restriction de cardinalité N:

$I(\geq n\ R) = \{a \in \Delta \mid \{b \mid (a, b) \in I(R)\}\mid \geq n\}$

$I(\leq n\ R) = \{a \in \Delta \mid \{b \mid (a, b) \in I(R)\}\mid \leq n\}$

Exemples :

$Homme \sqcap \geq 2 aEnfant$

$Homme \sqcap \leq 2 aEnfant$

6.6 Restriction de cardinalité qualifiée Q:

$I(\geq n\ R.C) = \{a \in \Delta \mid \{b \mid (a, b) \in I(R) \wedge b \in I(C)\}\mid \geq n\}$

$I(\leq n\ R.C) = \{a \in \Delta \mid \{b \mid (a, b) \in I(R) \wedge b \in I(C)\}\mid \leq n\}$

Avec ces constructeurs à notre disposition, nous sommes maintenant en mesure de décrire la classe des gens qui possèdent deux chiens ou plus :

≥ 2 possède.Chien

6.7 Constructeur de négation sans restriction *C* :

Syntaxe : $\neg C$

Sémantique: $I(\neg C) = (\Delta) \setminus I(C)$

propriétés de Constructeurs U, ¬,E,nR

$\bot \equiv C \sqcap \neg C$

$C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$

$\neg(C \sqcup D) \equiv (\neg C \sqcap \neg D)$

$\neg(C \sqcap D) \equiv (\neg C \sqcup \neg D)$

$\neg \neg C \equiv C$

$\exists R \equiv \exists R.\top$

$\exists R.C \equiv \neg(\forall R.\neg C)$

$\neg \exists R.C \equiv \forall R.\neg C$

$\neg \forall R.C \equiv \exists R.\neg C$

$\neg(\geq n\ R) \equiv \leq (n - 1)\ R$

$\neg(\leq n\ R) \equiv \geq (n + 1)\ R$

Exemple:

décrire la classe des gens qui n'ont pas d'enfants de sexe masculin :

$\neg \exists aEnfant.Homme \equiv \forall aEnfant.\neg Homme$

$\exists aEnfant.Femme \equiv \neg(\forall aEnfant.\neg Femme)$

Exercice:

Définir une terminologie de famille contenant les éléments suivants

concepts: femme , homme, père, mère, parents, Grand-mère, Grand-père, grand parent, père avec seulement fils, Mère sans filles, Mère avec au moins trois enfants, le Père avec exactement deux fils, Épouse, Célibataire

● à partir de

- Concepts atomiques: Personne, Feminin
- Rôles atomique: aEnfant

Solution:

Femme $\equiv$ Personne $\sqcap$ Feminin

Homme $\equiv$ Personne $\sqcap \neg$ Femme

Mère $\equiv$ Femme $\sqcap \exists$ aEnfant. Personne

Père $\equiv$ Homme $\sqcap \exists$ aEnfant. Personne

Parent $\equiv$ Père $\sqcup$ Mère

Grandmère $\equiv$ Mère $\sqcap \exists$ aEnfant. Parent

Grandpère $\equiv$ père $\sqcap \exists$ aEnfant. Parent

Pèreavec seulement fils $\equiv$ père $\sqcap \forall$ aEnfant.Homme

MèreSansFille $\equiv$ Mère $\sqcap \forall$ aEnfant. $\neg$ Femme

MèreDeFamilleNombreuse ≡ Mère ⊓ ≥ 3 aEnfant

Père avec exactement deux fils ≡ père ⊓ =2 aEnfant ⊓ ∀ aEnfant.Homme

Épouse ≡ Femme ⊓ ∃ mariéAvec. Homme

Célibataire ≡ Personne ⊓ ∀ mariéAvec. ⊥

EX02

On pourrait aussi représenter la classe des gens qui n'ont aucun fils qui est marié :

∀ aEnfant. ¬ (Homme ⊓ ≥ 1 mariéAvec) ou

∀ aEnfant. (¬ Homme ⊔ ∀ mariéAvec. ⊥)

$$Man \equiv Person \sqcap Male$$
$$Woman \equiv Person \sqcap \neg Man$$
$$Mother \equiv Woman \sqcap \exists hasChild.Person$$
$$Father \equiv Man \sqcap \exists hasChild.Person$$
$$Parent \equiv Father \sqcup Mother$$
$$GrandMother \equiv Mother \sqcap \exists hasChild.Parent$$
$$GrandFather \equiv Father \sqcap \exists hasChild.Parent$$
$$GrandParent \equiv GrandMother \sqcup GrandMother$$
$$FatherWithOnlySons \equiv Father \sqcap \forall hasChild.Man$$
$$MotherWithAtLeast3Children \equiv Mother \sqcap \geq 3hasChild$$
$$MotherWithoutDaughter \equiv Mother \sqcap \forall hasChild.Man$$
$$FatherWith2Sons \equiv Father \sqcap = 2hasChild \sqcap \forall hasChild.Man$$

Définir la terminologie suivant les caractéristiques suivantes :

- A man is a human
- A woman is a human
- No man is a Woman, and vice-versa.
- A team is defined as the set with at least 2 members which are all humans.
- A small team is defined as a team with at most 5 members.
- A modern team is defined as a team with at most 4 members and with at least 1 leader, which is a member, and all leaders are women.

$$Human \sqsubseteq Top$$
$$Man \equiv Human$$
$$Woman \equiv Human \sqcap \neg Man$$
$$Team \equiv \forall member.Human \sqcap \leq 2member$$
$$SmallTeam \equiv Team \sqcap \geq 5member$$
$$leader \sqsubseteq member$$
$$ModernTeam \equiv Team \sqcap \leq 1leader \sqcap \forall leader.Woman$$

Exemple : Base de connaissances

- Un Homme est une Personne.
- Une Femme est une Personne.
- Aucune Femme n'est un Homme et vice-versa.
- Une Equipe est (définie comme) un Ensemble ayant au moins 2 membres qui sont tous des Personnes.
- Une Petite-équipe est (définie comme) une Equipe ayant au plus 5 membres.
- Une Equipe-moderne est (définie comme) une Equipeayant au moins 4 membres, ayant au moins 1 chef, et don't tous les chefs sont des Femmes.

Solution:

La base de connaissances

Concepts primitifs : Personne, Ensemble

Rôles primitifs : membre

Femme $\sqsubseteq$ Personne

Homme $\sqsubseteq$ Personne $\sqcap \neg$Femme

chef $\sqsubseteq$ membre

Equipe $\sqsubseteq$ Ensemble $\sqcap$ ($\forall$membre Personne) $\sqcap$ ($\geq$ 2membre)

Petite-équipe = Equipe $\sqcap$ ($\geq$5 membre)

Equipe-moderne = Equipe $\sqcap$($\geq$4 membre)$\sqcap$ $\exists$chef $\sqcap$($\forall$chef.Femme)

Conclusion :

Ce chapitre a présenté les concepts base des logiques de description. Les logiques de description sont utilisées pour de nombreuses applications et Workshop on Applications of Description Logics). Sans être exhaustifs, nous pouvons dire que ces applications font partie des différents domaines tels que le web sémantique, la médecine/bio-informatique, le traitement automatique des langues et représentation de la connaissance, l'ingénierie de processus, l'ingénierie de la connaissance et l'ingénierie logicielle.

Chapitre 06

Graphes Conceptuels et Graphes de Sowa

Introduction :

Un graphe conceptuel est un formalisme de représentation de connaissances et de raisonnements. Ce formalisme a été introduit par John F. Sowa (en) en 1984. Depuis cette date, ce formalisme a été développé suivant trois directions principales : interface graphique de la logique du premier ordre, système diagrammatique pour la logique du premier ordre, formalisme de représentation de connaissances et de raisonnement basé sur les graphes.

1- Graphes Conceptuels (GC):

Les graphes conceptuels (GC) :

- ont été proposés par Sowa (1984), complété en 1988, puis 1992
- ils constituent une notation graphique pour la logique
- ils sont plus formalisés que les réseaux sémantiques (dispose d'une sémantique formelle)
- ils sont beaucoup utilisés dans le traitement des langues naturelles

Différents niveaux dans le formalisme:

- Graphes simples
- Graphes imbriqués

Représentation des Graphes Conceptuels:

- Affichage graphique - DF (Display Form)
- Lecture linéaire – LF (Linear Form)
- CGIF (Conceptual Graph Interchange Format)

1.1 Définition d'un Graphe Conceptuel (GC):

Un GC est un graphe bipartite orienté

- Bipartite : 2 sortes de nœuds : concept et relation (les concepts représentés avec des **rectangles** et les relations, représentées à l'aide d'**ovales"** ou **cercles**").
- Les nœuds sont liés par des arcs orientés.
- Un arc lie toujours un concept à une relation.
- Toute relation est permise (partie de, couleur, etc…).
- La « sémantique » (au sens de la logique) est apportée par les relations.
- L'orientation des flèches doit être guidée par l'association du sous-graphe [CONCEPT1] -> (REL) -> [CONCEPT2] à la phrase « La REL de CONCEPT1 est CONCEPT2 ».
- Un graphe conceptuel possède les propriétés suivantes :
    - il a un nombre fini de nœuds.
    - il est connexe, car tous ses nœuds sont reliés. Si deux parties ne sont pas connectées, ces deux parties seront considérées comme deux graphes conceptuels.

- il est bipartite, dans le sens où il n'est composé que de deux types de nœuds.

- Un concept peut former à lui seul un graphe conceptuel, ce qui n'est pas le cas d'une relation conceptuelle.

Figure 6.1 : Relation conceptuelle.

Exemples :

« Le chat est sur le tapis »

Forme graphique (Display Form – DF) :

Figure 6.2 : Exemple de forme graphique d'un graphe Conceptuel.

2 concepts : chat et tapis et 1 relation : sur

Forme linéaire (Linear Form – LF) :

[chat]->(sur)->[tapis]

1.2 Définition des concepts dans les GC:

Les concepts sont des « objets mentaux » qui ont une existence reconnue par le cerveau humain et peuvent être définis. Ils ont éventuellement un numéro, une marque qui les différencient les uns des autres. Nous pourrions utiliser comme concept n'importe quel objet qui nous viendrait à l'esprit, car un concept est théoriquement capable de représenter n'importe quel objet dicible.

a) Type de concept

Le **type** indique la classe à laquelle appartient le concept.

Soit la fonction *type* de l'ensemble des concepts vers T, l'ensemble des noms de concepts. Les concepts c1 et c2 sont dits de même type si *type*(c1) = *type*(c2).

Dans les graphes conceptuels, deux concepts ayant le même nom de type (le même nom entre crochets) sont deux concepts de même type.

La **dénotation** du nom de type t, notée δ**t**, est l'ensemble de toutes les entités qui sont des instances de tout concept de type t.

Sur les types de concepts, une relation d'ordre partiel "≤" définie permet de construire une hiérarchie de type.

Soient s,t et u des noms de type.

- Si s ≤ t, alors s est dit **sous-type** de t et t **surtype** de s (t ≥ s).

- Si s ≤ t et s ≤ u, alors s est appelé **sous-type commun** de t et u.

- Si s ≥ t et s ≥ u, alors s est appelé **surtype commun** de t et u.

Cette hiérarchie de type forme ce qui est appelé **un treillis de type** contenant

- un type universel noté T,
- un type absurde ⊥

tels que pour tout nom de type t , ⊥ ≤ t ≤ T

Dans ce treillis, toute paire (s,t) de noms de type possède un **surtype commun**

Minimal (qui est la borne inférieure des sur-types communs), noté **s** ∪ **t**. . Pour tout surtype u de s et t, u est aussi surtype de **s** ∪ **t**...

Toute paire (s,t) possède un **sous-type commun maximal** (la borne supérieure des sous-types communs), noté **s** ∩ **t**, et tout sous-type u de s et t est aussi sous-type de s∩ t.

Dans le treillis ci-dessous,

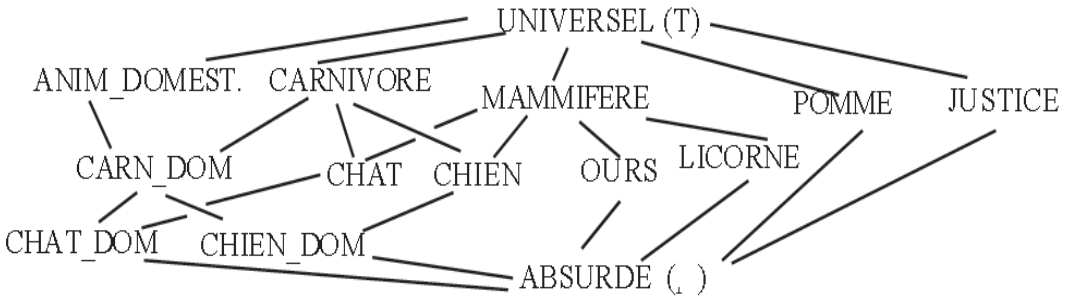


Figure 6.3 : Type de concept dans le graphe Conceptuel.

CHAT, CHIEN, OURS, LICORNE, POMME, JUSTICE n'ont pas de sous-type commun.

CHAT_DOM est le sous-type commun maximal de CHAT et ANIMAL_DOMESTIQUE.

CARNIVORE_DOMESTIQUE est le surtype commun minimal de CHAT_DOM et CHIEN_DOM.

MAMMIFERE est le surtype commun de LICORNE et CHAT.

b) Le référent

Des **marqueurs individuels** permettent de préciser l'individu du monde réel concerné.

Soit l'ensemble des marqueurs individuels I = {#1, #2, #3, ...}.

Sowa définit de la manière suivante la fonction *referent* qui s'applique à un concept c:

- référent(c) ∈ I, le concept c est un **concept individuel**

- référent(c) = *, le concept c est un **concept générique**.

Le concept [CHAT:*] ou [CHAT] renvoie à un chat non spécifié.

Le concept [HOMME:*x] renvoie aussi à un homme non spécifié.

Le concept [CHAT:#9456] renvoie à un chat particulier auquel le système aurait affecté le numéro 9456.

[Chat: ∀] ou [Chat: @every] = Tous les chats

[chien:?] = Quel chien

[féminin:#] = Elle

[personne : *] = concept générique (un homme)

[personne : #123] = concept individuel (cet homme)

[personne : Paul] = concept individuel, nom propre

[hauteur: @1,73] = concept mesure (hauteur = 1,73m)

C) Les ensembles dans les GC

Type d'ensemble	**Interprétation**	**Forme linaire**
Concept générique	un homme	[homme : *]
Concept individuel	cet homme	[homme : #123]
Nom propre	Paul	[homme : Paul]
Mesure	Hauteur = 1,73 m	[hauteur : @1,73]
Ensemble extensif	Paul, Alain, Max	[homme : Paul, Alain, Max]
Ensemble **générique**	Plusieurs hommes	[homme : {*}]
Cardinal imbriqué	4 hommes	[homme : {*}@4]
Définition partielle	Paul, Alain, d'autres	[homme : {Paul, Alain, *}]
Ensemble **distributif** (Dist)	4 hommes(ont lu ces 2 livres)	[homme : Dist{*}@4]
Ensemble **respectifs** (Resp)	Paul, Alain (sont les maris respectifs de Alice, Marie)	[homme : Resp {Paul, Alain}@2]
le **cardinal** de l'ensemble des entités auxquels se réfère le concept ensembliste.	3 personnes dont Jean et Nicolas.	[PERSONNE:{Jean,Nicolas} @3]

Tableau 6.1 : Les ensembles dans les GC.

1.3 Relations conceptuelles :

Une relation conceptuelle relie des concepts d'un graphe et permet de spécifier le rôle joué par chaque concept au niveau du graphe (ou sous-graphe).

Les **relations** permettent de représenter des propositions portant sur des concepts, les relations conceptuelles permettent d'assembler les concepts pour **construire une phrase, une idée, une proposition.**

Une relation est définie par 2 éléments (r, a) :

r: nom de la relation

a: **arité** ou **valence,** elle indique le nombre de concepts que la relation lie. Elle est supérieure ou égale à 1. La relation est **unaire (**monoadique) lorsque son arité est égale à 1. La relation est dite **binaire (**dyadique) lorsque son arité est égale à 2. La relation est dite n-aire (n-adique) lorsque son arité est égale à n.

On peut définir :

- Signature d'une relation: ensemble de n types de concepts.

- Valence et signature sont fixés par le type de relation.

Les flèches sont orientées dans le sens concept-relation pour les arcs numérotés de 1 à n-1. Le nième arc est orienté dans le sens relation-concept.

Remarque: Pour limiter le nombre de relation, Doit-il y avoir autant de relations que de verbes.

2. Graphes de Sowa:

Graphes de Sowa = Graphe conceptuels + Primitives sémantiques

Idée:

– transformer les verbes en concepts

– Introduction de « primitives sémantiques »

Les primitives sémantiques les plus utilisées sont:

AGENT: l'instigateur de l'action (le sujet).

OBJET: ce sur quoi porte l'action (COD).

INSTRUMENT: par quel intermédiaire l'action est accomplie.

CONTRE-AGENT: la force contre laquelle l'action s'oppose.

RESULTAT: ce qui est créé par l'action.

SOURCE: lieu de départ.

BUT: lieu d'arrivé.

PATIENT: l'entité qui reçoit ou subit les effets de l'action.

……

Exemple01 : le verbe **Casser** commande trois cas: OBJET, INSTRUMENT, AGENT

La phrase « Un homme casse une branche avec son pied » peut être représentée par le

graphe conceptuel ci-dessous:

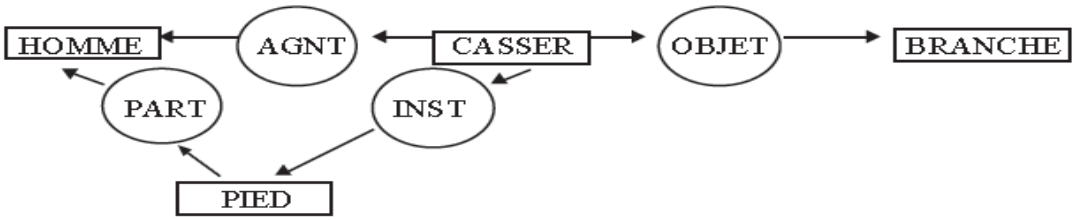


Figure 6.4 : Exemple 1 d'un graphe de Sowa.

Forme linéaire:

[CASSER] -

(AGNT) -> [HOMME:*x],

(OBJ) -> [BRANCHE],

(INST) -> [PIED] ---> (PART) --> [HOMME:*x].

Les crochets sont utilisés pour marquer les concepts, tandis que les parenthèses délimitent les relations. Un concept tête est tout d'abord choisi. Il est suivi d'un tiret, pour indiquer que les relations conceptuelles qui lui sont reliées suivent de manière séquentielle. La virgule est utilisée pour terminer un niveau du graphe et le point termine le graphe. Le choix du concept tête, ici CASSER, dépend du point de vue où l'on se place.

Une autre forme linéaire peut être obtenue en changeant de concept tête. Les différentes formes linéaires obtenues sont équivalentes.

Exemple02:

Langage naturelle : « le petit chat mange la souris »

DF (display form) :

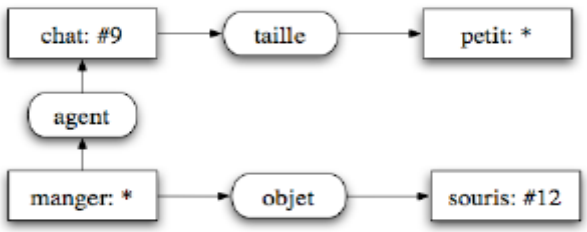


Figure 6.5 : Exemple 2 d'un graphe de Sowa.

Forme linéaire:

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

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

(agent)->[chat : #9].

Exemple03:

Langage naturelle : « Paul va à Marseille en bus »

DF (display form) :

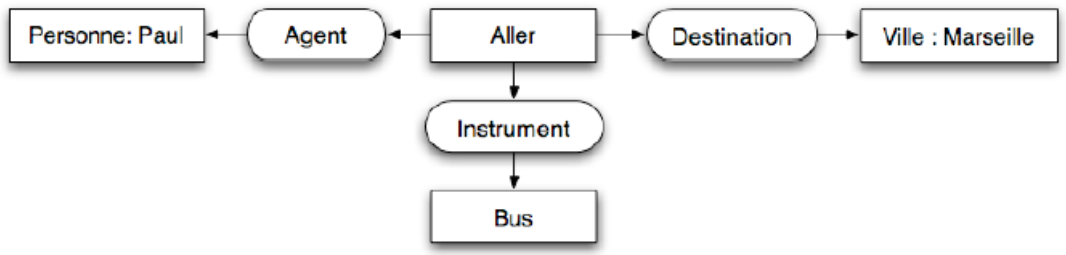


Figure 6.6 : Exemple 3 d'un graphe de Sowa.

Forme linéaire:

[Aller]-

(Agent)->[Personne: Paul],

(Destination)->[Ville: Marseille],

(Instrument)->[Bus].

Exemple 04 :

Langage naturelle : « Une personne est entre un rocher et un mur blanc »

DF (display form) :

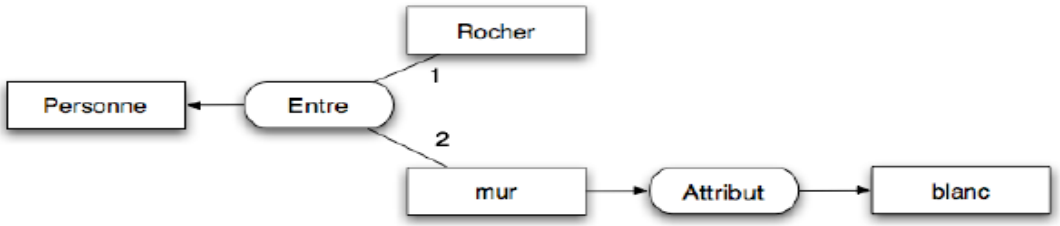


Figure 6.7 : Exemple 4 d'un graphe de Sowa.

• **valence de la relation Entre** = 3

• **signature de la relation Entre** = (Entité, Entité, Entité)

Forme linéaire:

[Personne]<-(Entre)-

<-1-[Rocher]

<-2-[Mur]->(Attribut)->[Blanc].

3- Hiérarchie des types (Types de concepts et de relations):

L'ensemble de tous les types (concepts et relation) représentés est partiellement ordonné .La relation de subsomption, par ses propriétés précitées, permet de classer les types dans une taxonomie. Nous pouvons la caractériser comme une hiérarchie, car l'ordre partiel créé interdit la formation de cycles. Il et donc possible de comparer tous les types entre eux et d'obtenir une réponse en un temps fini.

Par ailleurs, cette hiérarchie se rapproche beaucoup du réseau sémantique dans sa structure. Elle ne contient que les labels de types (qui contiennent la sémantique du système) et se présente sous la forme d'un graphe, par exemple :

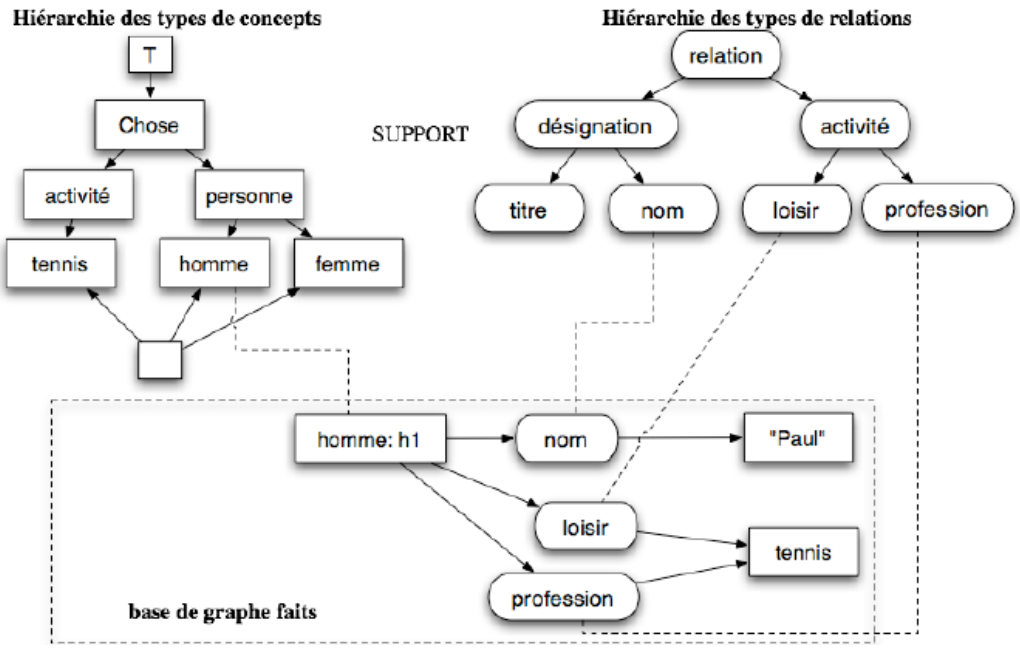


Figure 6.8 : Hiérarchie des types.

Signature de relation profession est : profession(homme, tennis)

Signature de relation loisir est : loisir(homme, tennis)

Les marqueurs individuels et leur type privilégié:

I = {Paul,….}

I(Paul)=homme

4- Contexte dans les GC:

Le contexte = concept désignant un graphe non vide. Les contextes peuvent être imbriqués

Exemple :

Langage naturelle: la phrase : « Paul croit que Marie veut se marier avec un marin »

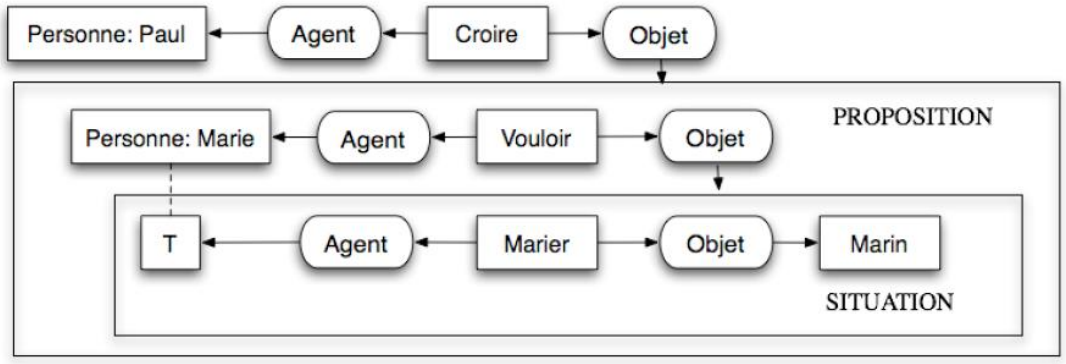


Figure 6.9 : Contexte dans les GC.

5- Forme normale d'un ensemble des graphes:

La forme normale est obtenue en fusionnant tous les sommets concept individuel ayant le même marqueur individuel.

Conclusion :

Les graphiques conceptuels et les graphiques de Sowa, ainsi que toutes leurs idées fondamentales, ont été introduits dans ce chapitre. Sowa a inventé les graphes conceptuels, une notation graphique pour la logique et le langage naturel, basée sur les structures des réseaux sémantiques et sur les graphes existentiels de Charles S. Peirce. Les graphes conceptuels offrent une notation de la logique plus proche des propositions en langage naturel que la logique des prédicats du premier ordre.

Chapitre 07

Formalismes basés sur la logique

Introduction :

Ce chapitre traite des différents formalismes de représentation des connaissances basés sur la logique.

Dans les formalismes de représentation de connaissances basés sur la logique on trouve :

- Logique des propositions (pas de quantificateurs et pas de variables)
- Logique des prédicats de premier ordre (introduction de variables etde quantificateurs)
- Règles de production

1. Logique propositionnelle

- La logique est un langage formel pour représenter l'information et permettre d'en tirer des conclusions.
- La logique des propositions (propositionnelle) une suite de symboles séparés par des conjonctions (et), des disjonctions (ou) ou des négations (non)
- La logique des propositions permet d'exprimer :
- Des faits sur le monde réel: "***Ali habite à M'sila***"
- Des négations: "***Mohamed n'habite pas à Msila***"
- Des conjonctions et des disjonctions « ***Mohamed est un étudiant, et ilhabite à M'sila*** »
- Des phrases avec "conséquence" logique : "***Si le fils ne rencontre passa mère, la mère ne le rencontre pas non plus***."
- Une proposition est une expression (phrase) à propos du monde quiest soit (Vraie) soit (Fausse).

Exemple: Représenter les expressions suivantes en logique de proposition

1. Socrate est un homme →HommeSocrate/ Homme(Socrate)
2. Platon et Socrate sont des hommes →

HommePlaton∧HommeSocrate

Homme(Platon) ∧Homme(Socrate)

3. Zola a écrit Germinal →***(AUTEUR, ZOLA)*** => ***(LIVRE,*** **GERMINAL**)

Auteur(Zola) => Livre(Germinal)

1.1 Éléments de base de la logique propositionnelle :

- Symboles de propositions : P, Q, … (Phrases)
- Phrases spéciales (valeurs) : Vrai, Faux
- Opérateurs : ⋀ (et), ∨ (ou), ¬ (non), ⇒ (implique), ⇔ (équivalent)

1.2 Formation de phrases (propositions) composées :

1. Négation : ¬ P
2. Conjonction : P∧ Q
3. Disjonction : P ∨ Q
4. Implication : P ⇒ Q (si P alors Q)
5. Equivalence : P ⇔ Q (P et seulement si Q)

Exemples :

1. Soient les propositions composées suivantes :

$$\neg (P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))\ P \wedge Q \wedge (Q \Rightarrow R)$$

$$\neg (P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))((P \wedge Q) \Rightarrow R)$$

$$(A \Rightarrow B) \vee (\neg C)$$

$$\neg (P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))$$

OBS : Ordre de précédence : ¬, ∧, ∨, ⇒

Exemples :

¬ A ∨ B ⇒ C est équivalent à((¬A) ∨B) ⇒CA ⇒ B ⇒ C est incorrect

2. Supposons : P = "le fils rencontre sa mère" Q = "La mère rencontreson fils"

Comment représenter la phrase ?

"Si le fils ne rencontre pas sa mère, alors elle ne le rencontre pas nonplus" ¬ P ⇒¬ Q

Table de vérité

P	¬ P	Q	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
V	F	V	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	V	F	F	F	V	V

Tableau 7.1: Table de vérité

OBS : La table de vérité aide à calculer la valeur logique de n'importe quelle phrase (proposition composée).

1.3 Logique propositionnelle et inférence

1.3.1 Définition : l'inférence est un mécanisme qui établit la sémantique d'une phrase (la valeur de vérité) en lui faisant subir une suite des transformations syntaxiques (sans utiliser la TV) en réduisant le problème original en un problème ne dépendant que de faits connus.

Exemple :

Prémisses:

1. "Si le fils ne rencontre pas sa mère, elle ne le rencontre pas non plus"

2. "La mère rencontre son fils"

Conclusion :

Alors "Le fils rencontre sa mère" est vrai

1.3.2 Correspondance en logique propositionnelle

Soit : P ≡ "Le fils rencontre sa mère" Q ≡ "La mère rencontre son fils"

1.3.3 Inférence

Prémisses en logique propositionnelle

1. ¬ P ⇒ ¬ Q

2. Q

Conclusion

Alors P

a) Composantes de base de l'inférence :

- ***Axiomes :*** faits connus du monde
- ***Règles :*** transformations syntaxiques qui propagent la valeursémantique

b) Pourquoi l'inférence est-elle intéressante ?

- Elle fournit un formalisme pour modéliser le raisonnement
- Elle réduit l'analyse sémantique à des manipulations syntaxiques
- Permet l'automatisation de ces manipulations syntaxiques.

Exemple d'une base de connaissance formulée en logiquepropositionnelle

1. Batterie-OK ∧Ampoules-OK ⇒Phares-OK (R1)
2. Batterie-OK ∧Starter-OK ∧¬Réservoir-Vide ⇒Moteur-Démarre (R2)
3. Moteur-Démarre ∧¬Pneu-Plat ⇒Voiture-OK (R3)
4. Phares-OK (f1)
5. Batterie-OK (f2)
6. Starter-OK (f3)
7. ¬Réservoir-Vide (f4)
8. ¬Voiture-OK
9. Batterie-OK ∧Starter-OK <= (5+6) (f5)
10. Batterie-OK ∧Starter-OK ∧¬Réservoir-Vide <= (9+7) (f6)
11. Moteur-Démarre <= (2+10) (f7)
12. ¬Moteur-Démarre ∨Pneu-Plat <= (8+3) ≡Moteur-Démarre ⇒Pneu-Plat f8)13. Pneu-Plat <= (11+12) (f9)

c) Systèmes de preuve

- Un système de preuve est une collection d'axiomes et de règlesd'inférence,
- Il existe de nombreux systèmes de preuve pour la logiquepropositionnelle.
- Les logiciens préfèrent les "petits" système de preuve (une seule règleet peu d'axiomes),
- Les spécialistes en IA préfèrent les systèmes avec des règles fortes (etnombreuses) et peu d'axiomes.

d) Validité & adéquation des phrases

1. ***Validité :*** certaines phrases sont toujours vraies (valides) etpeuvent donc être toujours supposées comme telle.

2. ***Adéquation :*** une règle d'inférence est "adéquate" si elleproduit à partir de prémisses une conclusion toujours vraie.

e) Système de preuve incomplet Vs système de preuve complet

1. ***Système incomplet :*** si on supprime n'importe quel axiome ou règle d'inférence, on ne peut plus rien prouver

⇒On obtient alors un système de preuve incomplet

2. ***Système complet :*** un système possédant suffisammentd'axiomes et de règle pour prouver n'importe quelle phrase(même si on supprime des axiomes et des règles).

Base de connaissances

Base de connaissance = ensemble de phrases exprimées dans unlangage formel (logique par exemple).

2. Logique des prédicats de premier ordreMotivation

- La logique propositionnelle a un pouvoir expressif limité(contrairement au langage naturel par exemple)

Exemple :

Comment exprimer les phrases suivantes en logique des propositions ?

- Tous les étudiants du master sont des algériens
- Tous les étudiants étaient soit satisfaits par le cours de leurenseignant soit ils n'étaient pas.
- Chaque étudiant a une relation d'amitié avec un autre.

2.1 Définition de la logique des prédicats

- Une suite de symboles, de variables et de relations avec desquantificateurs universels et existentiels.

- Au lieu des symboles propositionnels P, Q, …, utiliser des ***prédicats***

et des ***termes:***

Exemples

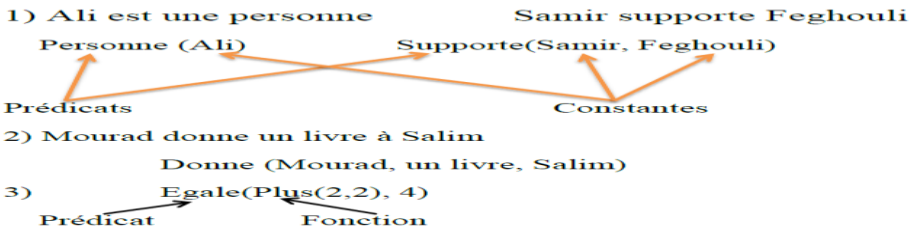


Figure 7.1 : Prédicats et termes.

2.2 Eléments de base de la logique des prédicats

- Constantes : Ali, 2, Unix, ...
- Prédicats :Père, Frère, >, Egale, supporte, ...
- Fonctions : Sqrt(.), JambeGaucheDe(.), Plus, ...
- Variables : x, y, a, b, ...
- Connecteurs : ⋀, ∨, ¬, ⇒, ⇔
- Égalité : =
- Quantificateurs :∀,∃

2.3 Forme d'une expression en logique des prédicats

Phrases atomiques = Prédicat(terme$_1$, … , terme$_n$)

OU(***terme$_1$ = terme$_2$ =...***)

Avec : ***Terme = Fonction(terme$_1$, … , terme$_n$)***

OU ***Constante*** ou ***VariableExemples***

1) Frère(Mourad, Nassim)

2) > (Longueur(JambeGaucheDe(Omar)),1.2)

Phrases complexes = phrases atomiques réunies par desconnecteurs

Tells ques: ¬ S, S1⋀ S2 , S1∨S2 , S1 ⇒ S2 , S1 ⇔ S2

Exemples

1) Frère(Mourad, Nassim) ⇒ Frère(Nassim, Mourad)

2) 2) > (2, 1) ∨ ≤ (1, 2)

3) > (2, 1) ⋀ ¬ > (1, 2)

2.4 Quantification

La quantification permet d'exprimer des propriétés sur une collectiond'objets sans avoir à les

désigner chacun par un nom,

Deux types de quantificateurs:

- Quantificateur existentiel: ∃
- Quantificateur universel: ∀

2.4.1 Quantification existentielle

Forme générale:∃ <***variables***><***phrase***>

Exemple:

1. Il y a quelqu'un d'intelligent à Unix

→∃ ***x A(x, Unix) ∧ Intelligent(x)***

∃ x P: est équivalent à la disjonction d'instanciations de P

A (Ali, Unix) ∧ Intelligent(Ali)

∨ A(Omar, Unix) ∧ Intelligent(Omar)

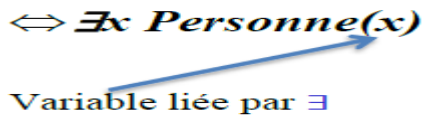
∨ A(Said, Unix) ∧ Intelligent(Said)

∨ ...

- Typiquement A est le connecteur principal avec ∃
- Erreur fréquente: utiliser ⇒ comme connecteur principal avec ∃:

Ex : ∃ x A(x, Unix) ⇒ Intelligent(x)

2. il y a une personne”



- Les variables dans une expression désignent des emplacementspossibles pour des constantes:

– x est une variable libre dans ***Personne(x)***

– x est une variable liée dans ***∃ x Personne(x)***

Exemples :

1. ∃ y (Mange(Walid, y) ∧ Biscuit(y))

⇔ “Walid mange un biscuit”

- Sans les parenthèses extérieures il y a une ambiguïté:

2. ∃ y Mange(Walid, y) ∧ Biscuit(y)

Pourrait être interprété comme :

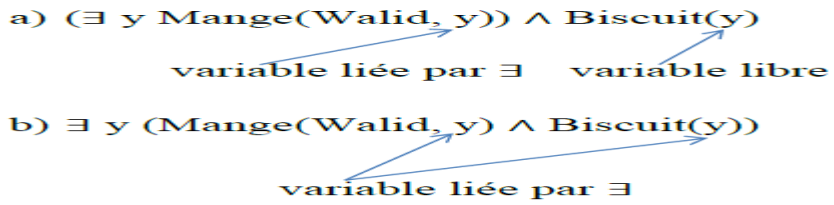


Figure 7.2 : Prédicats et termes.

2.4.2 Quantification universelle

Forme générale: ∀<*variables*><*phrase*>

Exemple:

1. Toute personne à Unix est intelligente

⇔∀ *x A(x, Unix) ⇒ Intelligent(x)*

∀ x P : est équivalent à la conjonction d'instanciations de P

A(Ali, Unix) ⇒ Intelligent(Ali)

∧ *A(Omar, Unix) ⇒ Intelligent(Omar)*

∧ *A(Said, Unix) ⇒ Intelligent(Said)*

A ...

- Typiquement ⇒ est le connecteur principal avec ∀
- Erreur fréquente: utiliser A comme connecteur principal avec ∀ :

Ex : ∀ *x A(x, Unix)* ∧ *Intelligent(x)*

⇨ Signifie: "***Tout le monde est à Unix et tout le monde est intelligent***"

Lois de Morgan

- ¬∃ *x φ* est équivalent à ∀ *x* ¬*φ*
- ¬∀ *x φ* est équivalent à ∃ *x* ¬*φ*
- ¬ *(φ* ∨ *ψ)* est équivalent à ¬*φ* A ¬*ψ*
- ¬ *(φ* ∧ *ψ)* est équivalent à ¬*φ* ∨ ¬*ψ*

2.4.3 Propriétés des quantificateurs

- ∀*x*∀ *y* est équivalent à ∀*y* ∀*x (permutation)*
- ∃*x* ∃*y* est équivalent à ∃*y* ∃*x (permutation)*
- ∃*x* ∀*y* n'est pas identique à ∀*y* ∃*x*

Ex: ∃*x* ∀*y Aime(x, y)* "Il y a quelqu'un qui aime tout le monde"

∀*y* ∃*x Aime(x, y)* "Chacun est aimé au moins par une personne"

2.4.4 Dualité des quantificateurs

Chaque quantificateur peut être exprimé à l'aide de l'autre :

1. $\forall x\ Aime(x, CrêmeGlacée)$ $\neg \exists x \neg$ Aime(x, CrêmeGlacée)
2. $\exists x\ Aime(x, Kiwi)$ $\neg \forall x \neg$ Aime(x, Kiwi)

Puissance de la logique des prédicats du 1ère ordre

- Presque toutes les phrases du langage naturel peuvent être représentéesen logique des prédicats du 1er ordre.
- Il n'y a pas de correspondance unique entre une phrase en langagenaturel et une expression logique,

Exemples:

1. La mère de Ali est mariée au père de Ali

$\Leftrightarrow Marié(Père(Ali), Mère(Ali))$

2. Ali vit dans une maison jaune

$\Leftrightarrow a)\ Vit(Ali, Maison) \wedge Couleur(Maison, Jaune)$

$b)\ \exists x\ Maison(x) \wedge Couleur(x, Jaune) \wedge Vit(Ali, x)$

Si la voiture appartient à Ali, alors elle est verte

$\Leftrightarrow a)\ Possède(Ali, Voiture) \Rightarrow Couleur(Voiture, Vert)$

$b)\ \exists x\ Voiture(x) \wedge Possède(Ali, x) \Rightarrow Couleur(x, Vert)$

3. Certaines personnes aiment les serpents

$\Leftrightarrow$ **a)** $\exists x\ (Personne(x) \wedge Aime(x, Serpent))$

$b)\ \exists x\ \forall y\ (Personne(x) \wedge Serpent(y)) \Rightarrow Aime(x,y)$

4. Tous les étudiants passent des examens

$\Leftrightarrow \forall x\ Étudiant(x) \Rightarrow Passe\text{-}examen(x)$

$b)\ \forall x\ (Étudiant(x) \Rightarrow \exists y\ Examen(y) \wedge Passe(x,y))$

5. Si x est parent de y, alors x est plus vieux que y

$\Leftrightarrow \forall x \forall y\ Parent(x,y) \Rightarrow PlusVieux(x, y)$

8. Si x est la mère de y, alors x est un parent de y"

$\Leftrightarrow \forall x\ \forall y\ Mère(x,y) \Rightarrow Parent(x, y)$

9. Chacun est loyal envers quelqu'un"

$\Leftrightarrow \forall x\ \exists y\ Personne(x) \wedge Personne(y) \wedge LoyalEnvers(x,y)$

Ou $\exists y\ \forall x\ Personne(x) \wedge Personne(y) \wedge LoyalEnvers(x,y)$?

10. Il existe quelqu'un envers qui chacun est loyal"

$\rightarrow$ ***phrase ambiguë !!***

10. Les gens tentent d'assassiner les dirigeants envers lesquels ils nesont pas loyaux

⇔∀x ∀y Personne(x) ∧ Dirigeant(y) ∧ TenterAssassiner(x,y) ⇒

¬LoyalEnvers(x,y)

11. La seule chose que les gens tentent de faire est d'assassiner ceuxenvers qui ils ne sont pas loyaux"

→ une autre phrase ambiguë !!

2.5 Égalité

Terme1 = Terme2 est vrai étant donné une interprétation si et seulementsi terme1 et terme2 font référence au même objet

Exemples:

1. Père(Omar) = Mohamed

2. Téléphone(Mourad) = 0770576344

Inférence en logique du 1er ordre

Prémisses:

1. Si x est un parent de y, alors x est plus âgé que y
2. Si x est la mère de y, alors x est un parent de y
3. Fatma est la mère de Zahra

Conclusion:

– Fatma est plus âgé que Zahra

Correspondance en logique du 1er ordre:

Prémisses:

1. ∀x∀y Parent(x,y) ⇒PlusAgé(x, y)
2. ∀x∀y Mère(x,y) ⇒ Parent(x, y)
3. Mère(Fatma, Zahra)

Conclusion:

PlusAgé(Fatma, Zahra)

L'inférence est obtenue par des axiomes et des règles (i.e. par destransformations syntaxiques) qui étendent ceux de la logique propositionnelle.

2.6 Unification

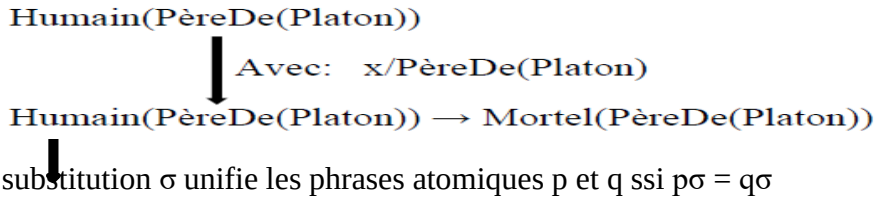
C'est le processus qui rend 2 expressions identiques.

Remarque: En logique propositionnelle 2 expressions sont les mêmes seulement si elles sont syntaxiquement identiques, la présence de variables en logique des prédicats complique ce fait.

Exemples :

1. Humain(x) = Humain(Socrate) **si et seulement si**x=SocrateIl faut parfois substituer une fonction à une variable:

2. Humain(x) → Mortel(x)



3. Une substitution σ unifie les phrases atomiques p et q ssi pσ = qσ

P	Q	σ	P=Q après unification
Connaît(Ali, x)	Connaît(Ali, Omar)	*{ x/Omar }*	Connaît(Ali, Omar)=Connaît(Ali, Omar)
Connaît(Ali, x)	Connaît(y, Karim)	*{ y/Ali, x/Karim}*	Connaît(Ali, Karim)=Connaît(Ali, Karim)
Connaît(Ali, x)	Connaît(y, Mère(y))	*{ y/Ali, x/Mère(Ali)}*	Connaît(Ali, Mère(Ali))=Connaît(Ali, Mère(Ali))
Connaît(Ali, x)	Connaît(x, Karim)	{échec}	

Figure 7.2 : Unification.

2.6.1 Principe de l'unification :

Unifier les prémisses des règles avec des faits connus, puis appliquer l'unificateur à la conclusion

Exemple:

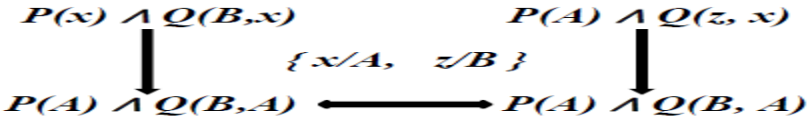
si on connaît q (dernière colonne) et on a la règle

Connaît(Ali, x) ⇒ Aime(Ali, x)

Alors on peut conclure :

- Pour q=Connaît(Ali, Omar)→ Aime(Ali, Omar)
- Pour q= Connaît(Ali, Karim) → Aime(Ali, Karim)
- Pour q= Connaît(Ali, Mère(Ali)) →Aime(Ali, Mère(Ali))

2.6.2 Règle : Une variable doit être substituée de manière consistante pourtoutes ses occurrences dans les expressions à unifier



2.6.3 Preuve par chaînage avant (Forward Chaining)

Trouver une preuve pour la conclusion ***PlusVieux(Fatma, Zahra)*** àl'aide d'un chaînage avant :

1. Mère(Fatma, Zahra) (donné)
2. Vivant(Fatma) (donné)
3. ∀x∀y Mère(x,y) ⇒ Parent(x, y) (donné)
4. ∀x∀y (Parent(x,y) ∧Vivant(x)) ⇒PlusVieux(x, y)
   (donné)

Par remplacement du fait (1) dans la règle (3) on obtient le nouveau faitsuivant :

5. Parent (Fatma, Zahra)

Remplaçons les faits (5), (2) dans la règle (4) on obtient le nouveau fait:

6. PlusVieux(Fatma, Zahra) (ce qui est demandé)

2.6.4 Principe du chaînage avant: effectuer une série de remplacement des faits donnés dans des règles aussi données jusqu'à arriver au fait à vouloir prouver (il s'agit d'un chaînage guidé par les données de départ Data-Driven)

2.6.5 Preuve par chaînage Arrière (Backward chaining)

a) Principe: Partir d'un objectif (à prouver) et dériver de nouveaux sous-objectifs jusqu'à n'avoir que des sous-objectifs connus comme vrais.

→ Semblable à la réduction de problème (chaînage guidé par l'objectifà prouver).

b) Exemple: Prenons le même exemple précédent

Preuve par chaînage arrière de ***PlusVieux(Fatma, Zahra)*** à partir desprémisses:

1. Mère(Fatma, Zahra)

2. Vivant(Fatma)

3. $\forall x \forall y$ Mère(x,y) $\Rightarrow$ Parent(x, y)

4. $\forall x \forall y$(Parent(x,y) $\wedge$ Vivant(x)) $\Rightarrow$PlusVieux(x, y)

objectif: ***(i) PLusVieux(Fatma, Zahra)***

on constate que l'objectif (i) correspond à la partie droite de (4)

⇨ Obtenir de nouveaux sous-objectifs:

(ii) Parent(Fatma, Zahra)

(iii) Vivant(Fatma)

(iii) correspond à (2) (déjà prouvé ou donné)

(ii) correspond à la partie droite de (3)

➔ On obtient un nouveau sous-objectif:

(iv) Mère(Fatma, Zahra)

(iv) correspond à (1) (qui est vrai)

On constate que tous les sous objectifs sont vrais, l'objectif de départest aussi vrai

Remarques :

- Le chaînage arrière est dirigé par les objectifs ("goal directed").
- Le chaînage arrière est à la base de la programmation logique(Prolog)

3. Autres Logiques

a) Logique de croyance

Ex : Croire(Ali, Père(Mourad, Ibrahim))

b) Logique temporelle

Permet de raisonner à propos du temps

c) Logique non-monotone

Permet aux valeurs de vérité associées aux faits de changer

d) Logique floue

Accompagne les faits de valeurs de vraisemblance

4. Règles de production

- Le formalisme le plus utilisé par les systèmes experts.
- Des formes simple et efficace.
- Base sur des règles de la forme « Si … Alors … »

Exemples :

1. S'il n'y a pas d'image mais que vous entendez le son, vérifiez leréglage d'intensité de l'écran"

Si *Non Image* ***et*** *Son* ***Alors*** *Intensité-Écran*

2. Un client est insolvable, si son compte bancaire est négatif, etqu'il ne possède aucun titre

Si ((*Solde-Compte < 0****)et (****Montant_Titre=0****)) Alors***

Client_Insolvable

4.1 Avantages

- Facilité d'expression
- Efficacité et clarté de représentation
- Développement aisé => large diffusion de logiciels
- Formation de programmeurs moins exigeante (Ing.Connais).
- Facilité de M.à.j
- Possibilité de vérification et de validation de connaissances.
- Implication des utilisateurs dans le développement.
- Possibilité d'appeler d'autres formalismes tels que les objetsstructurés.

Conclusion :

La logique est un langage formel pour représenter l'information et permettre d'en tirer des conclusions. Dans ce chapitre, les deux formalismes de représentation de connaissances basés sur la logique : la logique des propositions (pas de quantificateurs et pas de variables), la logique des prédicats de premier ordre (introduction de variables et de quantificateurs) et les règles de production.

Chapitre 08

Web sémantique, Ontologies et langages de description (RDF)

Introduction :

Web :

Les prémisses du Web

- Les origines :
  - ✓ Réflexions sur les documents électroniques.
  - ✓ Architectures et protocoles réseaux.
  - ✓ Web = couche applicative sur le réseau Internet
- Dates clés :
  - ✓ 90-92 : HTTP, HTML 1.0, NEXTSTEP
  - ✓ 92-97 : définition des URL, naissance du W3C, MOSAIC puis IE, HTML intégre images, tableaux, applets.

La maturité du Web

- Séparer forme et contenu : La nécessité de traiter et échanger les données du web font apparaître le besoin de séparer les données des traitements et de la présentation dans le navigateur.
- WEB = grande base de données de documents structurés.
- Dates clés :
  - ✓ 96 : Première définition des feuilles de style,
  - ✓ 98 : définition de xml 1.0,
  - ✓ 98-05 : langages de schémas DTD, XML SCHEMA, RELAX NG, langages XHTML, SVG, MATHML, ..., langages de manipulation XPATH, XLINK, XPOINTER, XSL, XSLT, XQUERY.

Le Web aujourd'hui :

- Information sur le Web essentiellement prévue pour être affichée (écran, imprimante) et lue par des humains
- Il est essentiellement syntaxique : contenu quasi inaccessible aux traitements machines.
- Seuls les humains peuvent interpréter ces contenus.
- BUZZWORD WEB 2.0 : généralisation du modèle à tous les systèmes d'information, clients légers (le navigateur), syndication et push, XHTML 5.0, petits objets portables communicants, réseaux sociaux.
- WEB =
  - ✓ Ensemble de ressources de très grande taille (WEB et WEB caché),
  - ✓ Outillé pour le traitement et la représentation de données et documents,

- ✓ Outillé pour l'adressage, la publication et l'échange de données,
- ✓ Outillé pour la recherche par l'humain de ressources,
- ✓ Permettant l'interaction des utilisateurs dans des réseaux.

Moteurs de recherche par mot clé :

- Les activités Web ne sont pas particulièrement adaptés aux outils logiciels
  - ✓ A l'exception des moteurs de recherche par mot clé :
  - ✓ Google, altavista Yahoo · · ·
- Le Web n'aurait pas eu autant de succès sans les moteurs de recherche.

Problèmes des moteurs de recherche par mot clé :

- Faible précision
- Résultats très sensibles au vocabulaire
- Résultats : seulement pages Web
- Intervention humaine pour interpréter et combiner les résultats
- Résultats des recherches pas lisibles par d'autres logiciels.

Les principaux problèmes du Web d'aujourd'hui

- Le sens des contenus Web n'est pas accessible aux machines : **manque de sémantique**
- Difficulté de distinguer le sens du verbe jouer :
  - ✓ Elle joue des cymbales.
  - ✓ Elle joue des coudes.
  - ✓ Elle joue de son influence.

Le futur du web :

- WEB = ? ? ?
  - ✓ Les évolutions sont très difficiles à prévoir car fortement dépendantes des réactions des communautés à des propositions d'évolutions technologiques ou applicatives.
  - ✓ Récemment les petits objets portables communicants, les données multimédia, les capteurs, les réseaux sociaux.
  - ✓ Demain la parole ? Nouvelles interfaces ? Nouveaux modèles de communautés ?
- Un besoin : des utilisateurs désireux de services de plus en plus sophistiqués. Alors pourquoi pas le WEB 3.0 = SEMANTIC WEB ?

1. Web sémantique

- *"The semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation"*

- Le Web actuel :

- ✓ Pas de structure explicite globale.
- ✓ Liens non exploitables sémantiquement.
- ✓ Travail limité sur les informations.
- ✓ La situation de RI sur le web :
  - Le mode d'interaction de l'utilisateur avec le web passe prioritairement par un moteur de RI,
  - L'interrogation et la recherche sont faites de façon syntaxique
  - L'utilisateur humain interprèteète les résultats, i.e. leur attribue une sémantique, et reformule sa requête au besoin.
- ✓ La situation de RI sur le WEB : Exemple de requête
  - "Hugo"
  - Liste ordonnéée de documents du Web contenant la chaîne de caractères Hugo sur des critères syntaxiques.
  - L'utilisateur affine sa requête selon qu'il cherche un titre de roman de Victor Hugo, la date de naissance de Victor Hugo, ou encore un calecon Hugo Boss.
- ✓ Services sur le web : La situation
  - La situation est identique car l'utilisateur humain doit lui-même décomposer sa demande en des recherches de services et appels de services WEB.
- ✓ Services sur le web : Exemple de service
  - "3 jours à Rome le week end de Toussaint"
  - Géolocaliser le point de départ, connaître les moyens de transport et d'hébergement, appeler les services correspondants de transport et d'hébergement, comparer les offres, ...
- ✓ Que faut-il ajouter au web actuel ?
  - Il faudrait que les programmes (les services) puissent interpréter les données : ce document correspond à un hôtel, un hôtel est un mode d'hébergement, dans un hôtel on peut réserver des chambres, une chambre ... Il serait alors possible d'appeler les services, de les combiner, d'ordonner les réponses à la demande de service, ...
  - Un prérequis est de représenter les connaissances liées aux données pour faire des inférences : cette page représente un hôtel, un hôtel est un mode d'hébergement donc cette page représente un mode d'hébergement.

- Le Web sémantique :

- ✓ Connaissances formalisées.
- ✓ Lien sémantique entre informations.
- ✓ Annotations plus riches.
- ✓ Standard à base d'XML, mais ouverture.

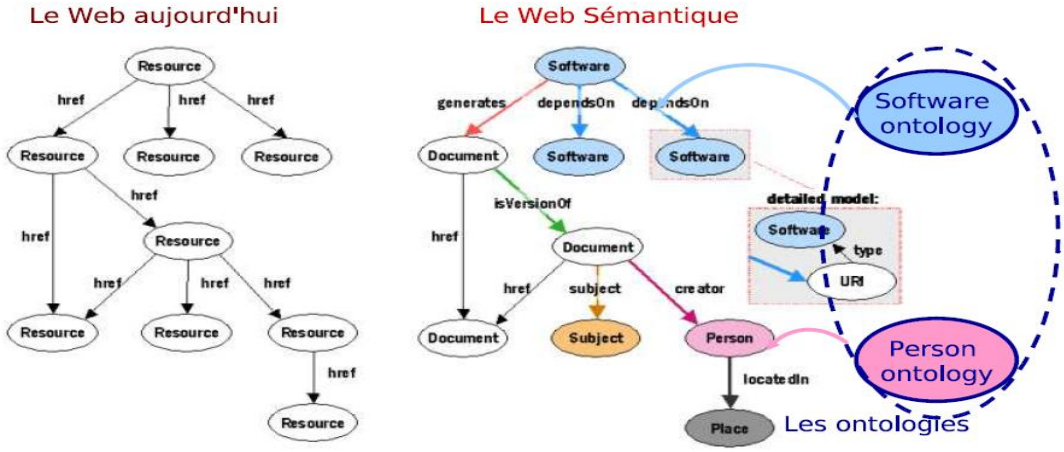


Figure 8.1 : Le Web aujourd'hui Vs Le Web sémantique.

- Le Web sémantique : impact sur gestion des connaissances

- ✓ Gestion des connaissances : acquisition, accès, maintien des connaissances dans une organisation.
- ✓ Activité importante dans l'industrie.
- ✓ Importance pour des organisation internationales dispersées géographiquement.
- ✓ La plupart des informations disponibles sont faiblement structurées (textes, sons, images, …).

1.1 Web sémantique Vs. Gestion des connaissances

- Limitation des technologies actuelles de gestion des connaissances :

- ✓ **Recherche d'information :** moteurs de recherche à base de mot-clé.
- ✓ **Extraction d'information :** intervention humaine nécessaire pour naviguer, chercher, interpréter, combiner.
- ✓ **Maintenance de l'information :** incohérences de terminologie, information dépassée.
- ✓ **Visualisation de l'information :** impossible de définir des vues sur la connaissance Web.

- Le Web sémantique adapté à la gestion des connaissances :

- ✓ Les connaissances sont organisées en espaces conceptuels selon leur signification.
- ✓ Outils automatiques pour la maintenance et la découverte de connaissances.
- ✓ Réponse à des questions sémantiques.

- ✓ Réponse à des questions sur plusieurs documents
- ✓ Possibilité de définir qui peut voir certaines parties de l'information.

1.2 Les challenges

- Représenter les connaissances dans un monde ouvert (tout type de connaissance), à l'échelle du web et avec une très large variété de protocoles, de langues, de culture, ... pour pouvoir manipuler les connaissances liées au contenu du web.

- Dates clés :

- **94 :** idées émises par Tim Berners Lee,
- **98 :** formalisation des idées au W3C,
- **98 -** : langage de description RDF99, langage de schéma RDFS04, et de raisonnement OWL, langage de requête SPARQL, ...

1.3 La pile du web sémantique

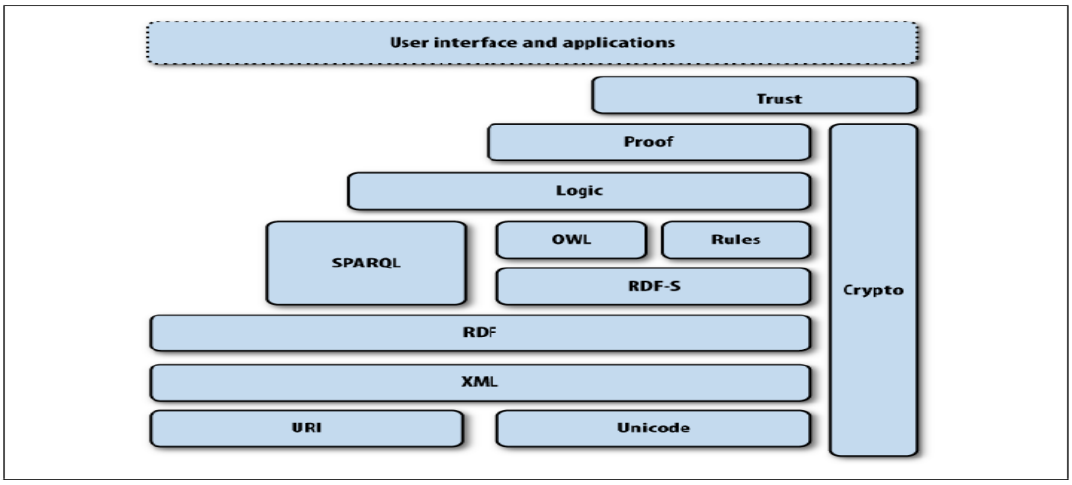


Figure 8.2 : Pile du web sémantique.

1.4 La pile réelle du web sémantique

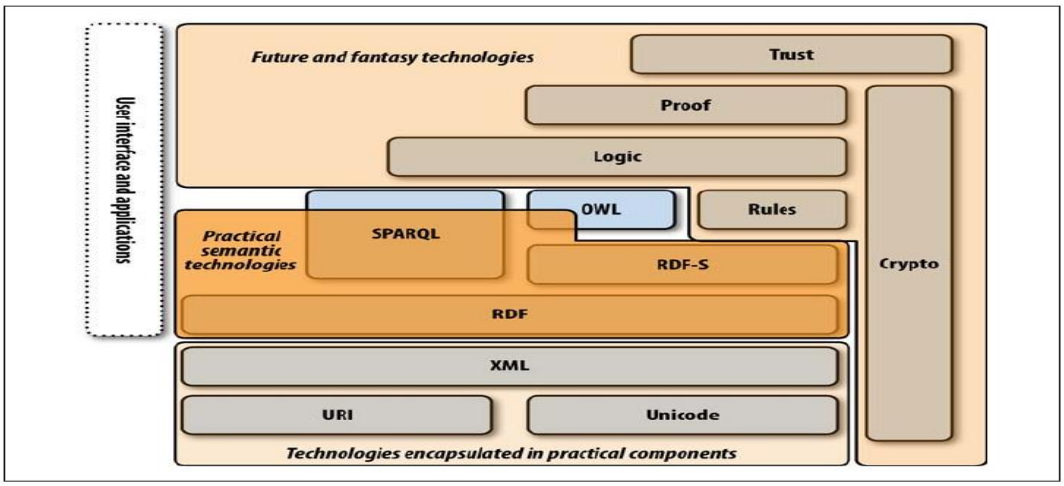


Figure 8.3 : Pile réelle du web sémantique.

1.5 Technologies du Web sémantique

- Méta données explicites
- Ontologies
- Logique et raisonnement
- Agents

1.5.1 Méta données explicites

- Représentation plus facilement exploitable par les machines.
- Méta données : données sur les données
- Méta données capturent une partie de la signification des données.
- Le Web sémantique ne repose pas sur des manipulations basées sur du texte mais plutôt sur des méta données exploitables par des machines.

Exemple HTML (G. Antoniou , F. van Harmelen)

```
<h1>Centre de kinésithérapie Agilitas  </h1>
Bienvenue à la page d'accueil du Centre de kinésithérapie
   Agilitas. Ressentez-vous de la douleur? Avez-vous eu un
   accident? Notre personnel
Lise Davanport,
Josiane Bouville (notre charmante secrétaire) et
Etienne Matthieu vont prendre soin de vous.
<h2>Horaire des consultations</h2>
Lun 11.00 - 19.00<br>
Mar 11.00 - 19.00<br>
Mer 15.00 - 19.00<br>
Jeu 11.00 - 19.00<br>
Ven 11.00 - 15.00<p>
Veuillez noter que nous n'avons pas de consultations les
   semaines de
<a href=". . .">State Of Origin</a> games.
```

EXEMPLE XML (G. Antoniou , F. van Harmelen)

```
<société>
   <traitementProposé>Kinésithérapie</traitementPro
   posé>
   <nomSociété>Centre de Kinésithérapie
   Agilitas</nomSociété>
   <personnel>
       <kiné>Lise Davanport</kiné>
       <kiné>Etienne Matthieu</kiné>
       <secrétary>Josiane Bouville</secrétaire>
   </personnel>
</société>
```

1.5.2 Ontologies

- **En philosophie :** étude de la nature de l'existence.

- **En informatique :**
  - Spécification formelle et explicite d'une conceptualisation.
  - Structurées en termes de concepts et de relations entre concepts.
  - Formalisation partagée sur un domaine.
  - Utiles pour l'organisation et la navigation sur les sites Web.

1.5.2.1 Langages de représentation des ontologies

- RDF Schéma :
  - RDF : modèle de données pour les objets et leurs relations.
  - RDF schéma : langage de description du vocabulaire.
  - Décrit les propriétés et les classes des ressources RDF.
  - Fournit une sémantique
- OWL
  - Langage plus riche.
  - Relations entre classes.
  - Contraintes de cardinalité.
  - Propriété de typage plus riches.

1.5.3 Logique et raisonnement

- Logique : étude des principes et formes du raisonnement.
- Langages formels de représentation des connaissances (ex Logiques de description).
- Sémantique formelle bien définie.
- Déduction automatique : outils de raisonnement (ex PELLET).
- Compromis entre expressivité et complexité calculatoire.

1.6 Le Web sémantique : Approche par couches

- Le développement du web sémantique s'opère par étapes. Chaque étape construit une couche au-dessus d'une autre.

- principes :

- Compatibilité descendante
- Compréhension partielle ascendante

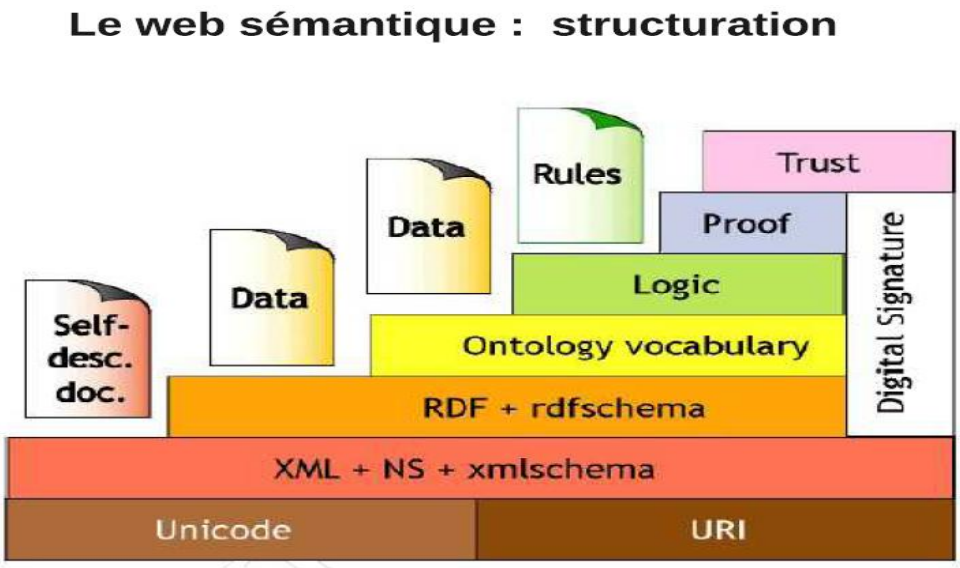


Figure 8.4 : Approche par couches.

Couche XML :

- Base syntaxique

Couche RDF

- RDF : modèle de données basique pour les faits.
- RDF Schéma : langage pour les ontologies.

Couche Ontologie

- Langage plus expressif que RDF Schéma.
- Standard courant pour le web : OWL.

Couche Logique

- Évolution des langages pour les ontologies
- Applications spécifiques pour des connaissances déclaratives

Couche Contrôle

- Génération de contrôles, validation

Couche Sécurisation

- Signatures numériques
- Recommandations, …

1.7 Le Web sémantique : Approche par couches alternative

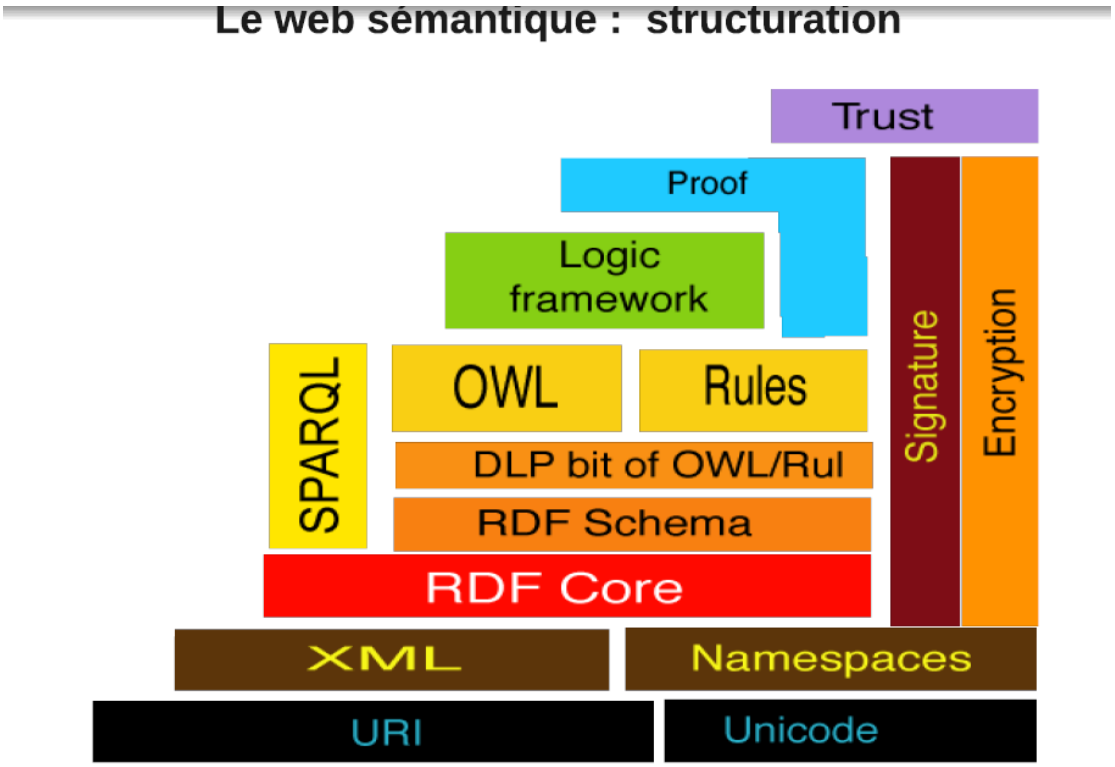


Figure 8.5 : Approche par couches alternative.

- Prend en compte les développements récents.
- Les différences essentielles :
  - Couche ontologie : 2 langages standard pour les ontologies pour le web : OWL et un langage basé sur des règles
  - DLP : intersection de OWL et la logique basée sur les clauses de Horn
- L'architecture du web sémantique est en débat : elle subir des modifications et évoluer dans le futur

2. Ontologies

éthymologie : **ontos** (l'existant) + **logos** (l'étude)

Philosophie :

- Étude de l'être en tant qu'être.
- Étude de l'existence en général

Informatique :

- Représentation de ce qui existe dans un formalisme permettant un traitement rationnel.
- Spécification explicite et formelle d'une conceptualisation

2.1 Spécification explicite d'une conceptualisation

- Spécification explicite : avec un langage
- Conceptualisation : structuration en concepts

2.2 Ontologies : concept

Désigné de 3 façons :

- Nom
- Signification (définition en intension)
- Objets dénotés (définition en extension)

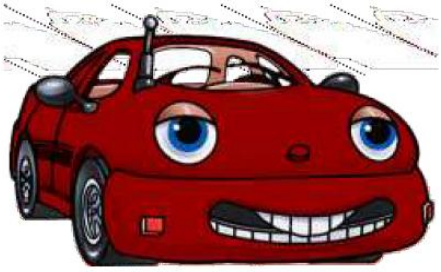


Figure 8.6 : Exemple de concept.

- Voiture, automobile, auto, tacot · · ·
- Véhicule automobile conçu pour le transport d'un petit nombre de personnes
- La 2CV 1945 RS 83, la punto 678 RS 13, la clio 999 ABC 13, …

2.2.1 relations entre concepts

- Généralisation (subsomption)
- Spécialisation
- Composition
- Est fabriqué, possède, …
- S'exprime de 3 façons par : son nom, son intention, son extension.

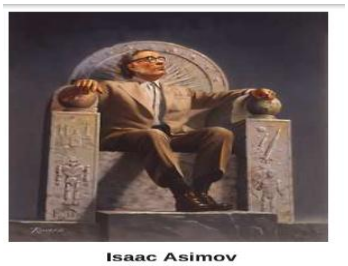


Figure 8.7: Relations entre concepts

- Auteur
- Personne qui crée une œuvre
- Homère est l'auteur de LOdyssée, Isaac Asimov est l'auteur des Robots …

2.3 Ontologies : Exemples

- Exemple schématique d'ontologie : le monde des cubes

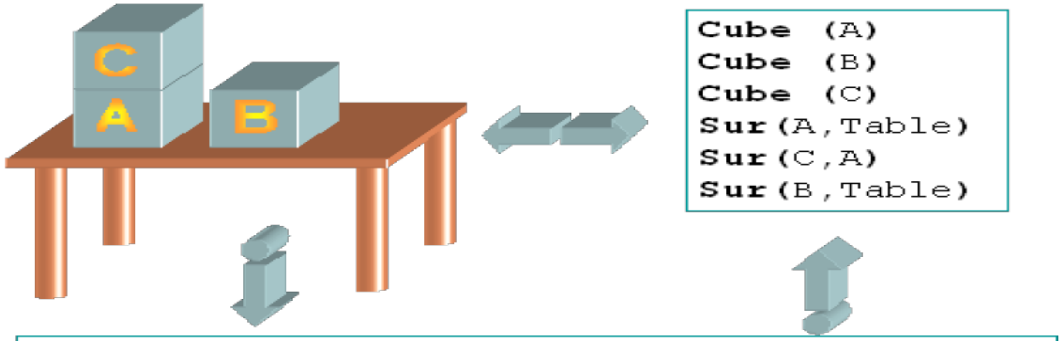


Figure 8.8: Exemple schématique d'ontologie.

Le monde des cubes : description de la scène

- Vocabulaire non ambigu (vocabulaire de l'ontologie)
- Énonciation des faits de la scène reposant avec le vocabulaire de l'ontologie

2.4 Ontologies : hiérarchie de concepts

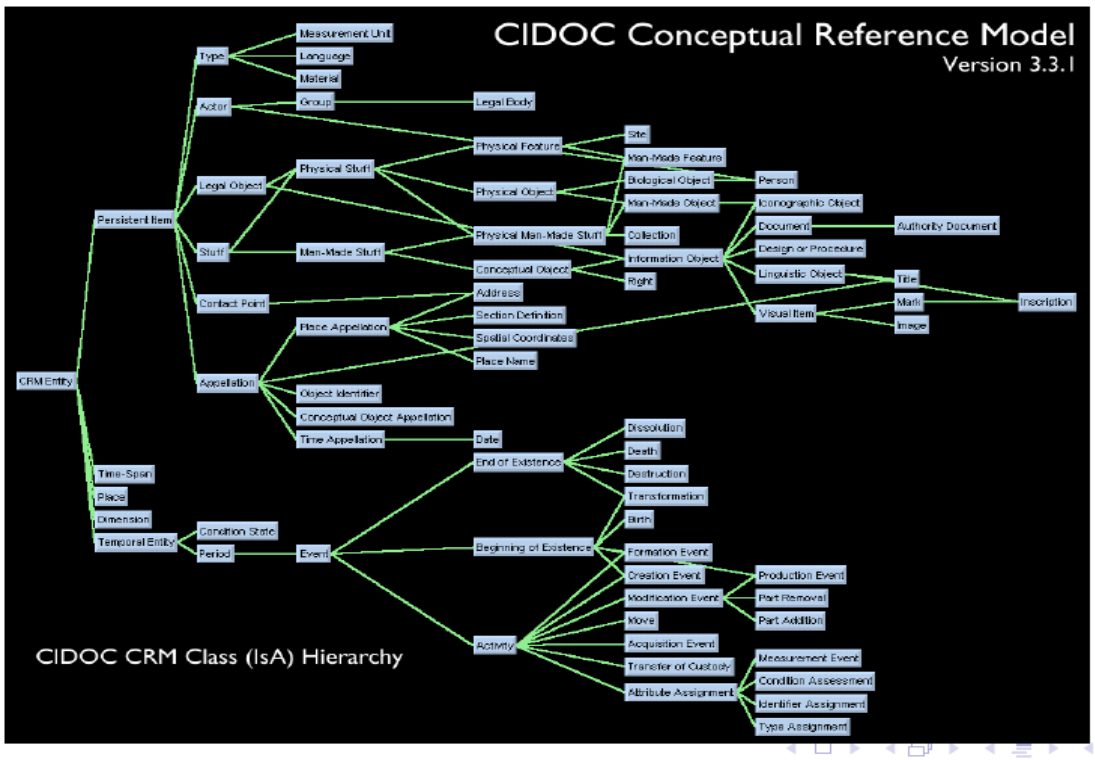


Figure 8.9: Hiérarchie de concepts.

2.5 Ontologies : Axiomes

Les contraintes :

- Contraintes de domaines
- Contraintes conditionnelles
- Contraintes d'intégrité

2.6 Ontologies : CRM CIDOC

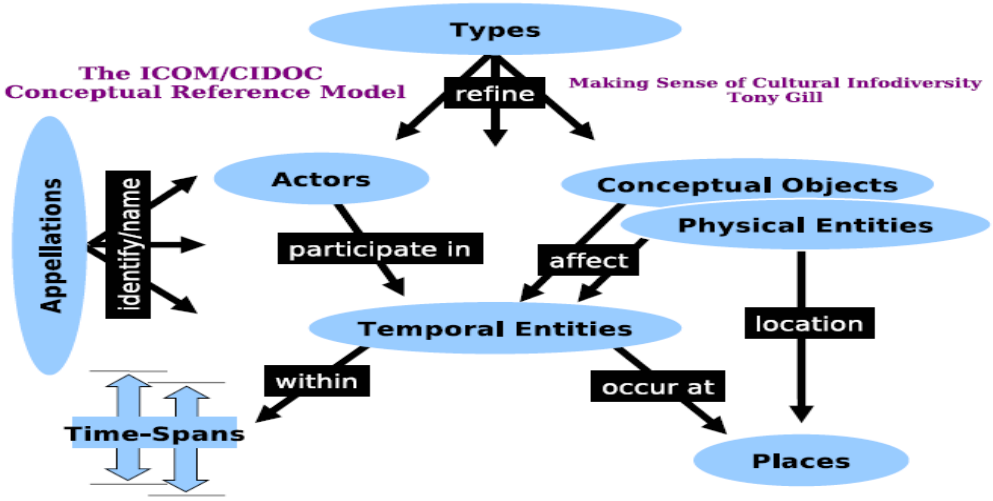


Figure 8.10: CRM CIDOC.

2.7 Ontologies : Définition formelle

- structure d'une ontologie

structure d'une ontologie

$$O = \{C,\ R,\ H^C,\ rel,\ A\}$$

- C et R : ensembles disjoints des concepts et des relations
- H^C hiérarchie de concepts : $H^C \subseteq C \times C$
- rel : relation rel: $R \rightarrow C \times C$ (définit des relations sémantiques non taxonomiques) avec 2 fonctions associées :
  - $dom : R \rightarrow C$ avec $dom(R) = \Pi_1(rel(R))$
  - $range : R \rightarrow C$ avec $range(R) = \Pi_2(rel(R))$ co-domaine

2.8 Ontologies : exemple de fragment d'ontologie

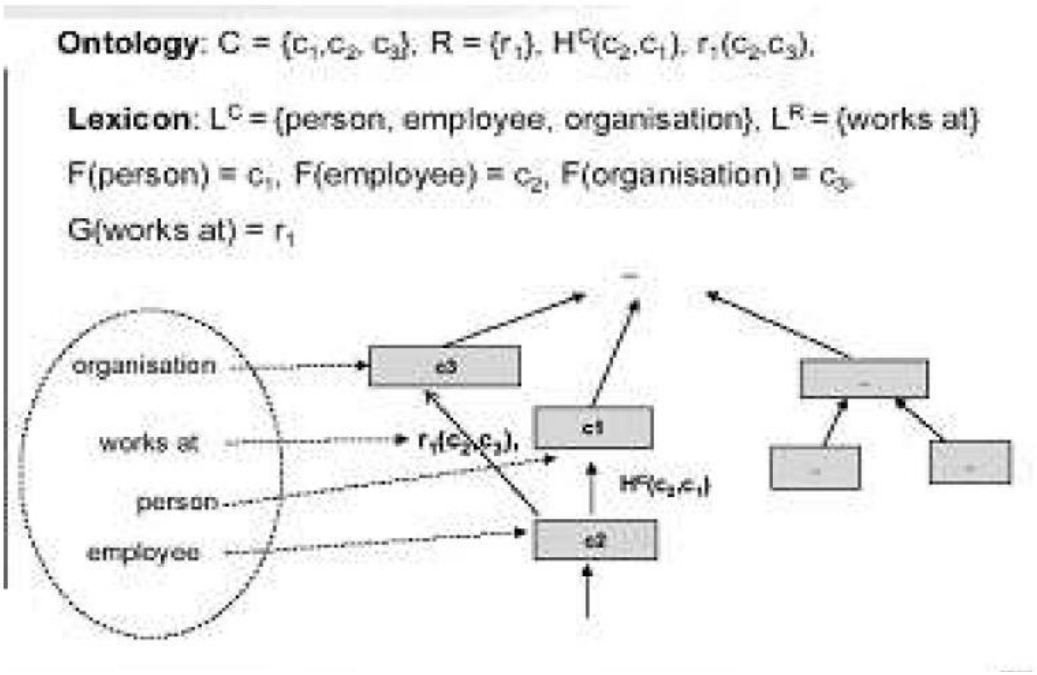


Figure 8.11: exemple de fragment d'ontologie.

Exercice : le monde des blocs

C = ?

R = ?

H^C = ?

dom(R) = ?

range(R) = ?

A = ?

- ontologies lourdes : $O = \{C,\ R,\ H^C,\ rel,\ A\}$
- ontologies légères : $O = \{C,\ R,\ H^C,\ rel\}$

Ontologie versus bases de connaissances

$$BC = \{O,\ I,\ inst,\ instr\}$$

- $O = \{C,\ R,\ H^C,\ rel,\ A\}$: une ontologie
- I : un ensemble d'instances
- $inst$: $C \rightarrow 2^I$ fonction d'instanciation de concept
- $instr$: $R \rightarrow 2^{I \times I}$ fonction d'instanciation de relation

2.9 Ontologies versus bases de connaissances

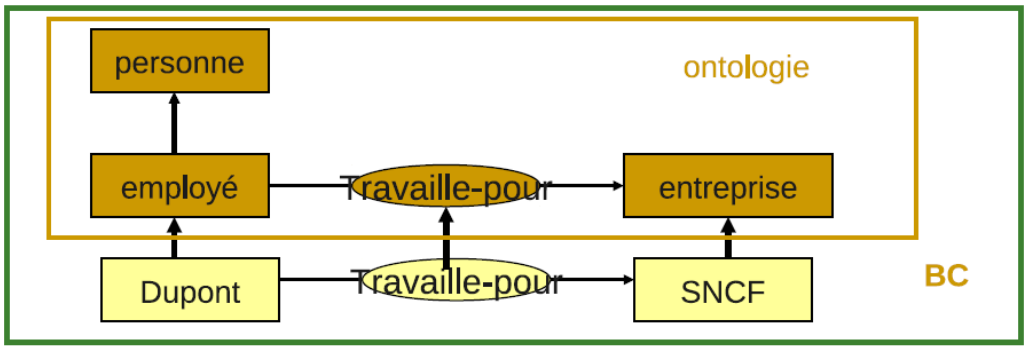


Figure 8.12: Ontologies Vs. Bases de connaissances.

Exercice : l'ontologie des repas

Ontologie

Un repas est constitué d'un hors-d'œuvre, d'un plat et d'un dessert, par ailleurs, un plat peut être soit de la viande, soit du poisson.

Donner l'ontologie des repas

Base de connaissances

La carte d'un restaurant qui comporte pour les hors-d'œuvre :

Cèleri, melon; pour les viandes : rôti, steak; pour les poissons :

sole; rouget; pour les desserts : flan, fruit.

Donner la base de connaissances du restaurant

2.10 Objectifs et rôles des ontologies

- Permettre un traitement symbolique des connaissances (le Web sémantique rejoint l'Intelligence artificielle)
- Faire des traitements automatiques à des logiciels au sein du Web pour faire interopérer des machines ou des machines et des humains.
- Vocabulaire, structuration et exploitation des méta-données
- Représentation pivot pour l'intégration de données de sources hétérogènes
- D´écrire les services web.

N.B :

Motivations : réutilisation et partage des connaissances et communication

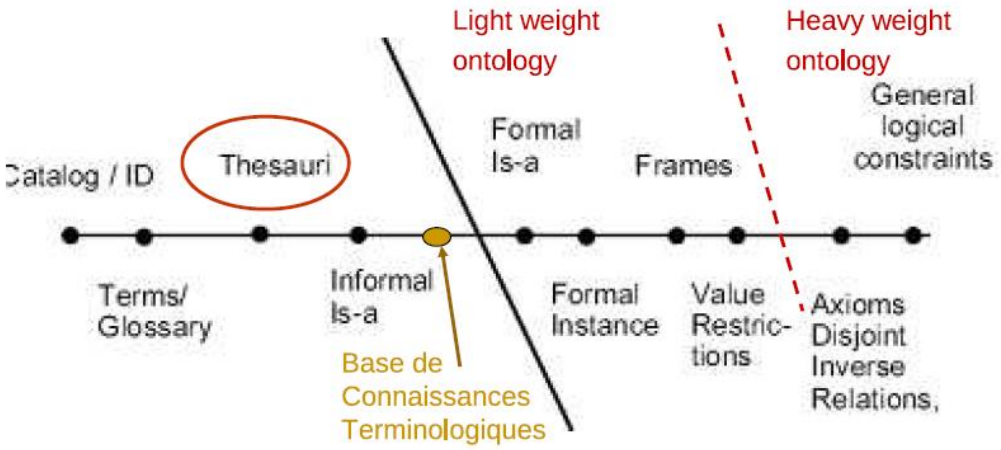


Figure 8.13: Réutilisation et partage des connaissances et communication.

2.11 Différentes ontologies

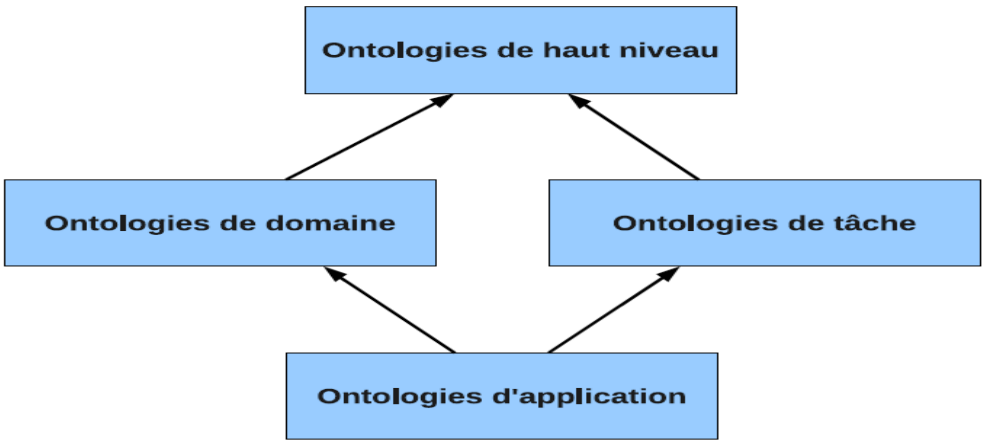


Figure 8.14: Différentes ontologies.

- Ontologies de haut niveau

- Concepts très généraux indépendants du problème

- Ontologies de domaine

- Concepts spécifiques `a un domaine

- Ontologie de tâche

- Concepts spécifiques à une application

- Ontologie d'application

- Concepts très spécifiques à un domaine et une tâche particulière.

Exemples :

- ontologies de haut niveau
  - DOLCE (http://www.loa-cnr.it/DOLCE.html)
  - Wordnet (http://www.cogsci.princeton.edu/ wn/index.html)

- ontologies de domaine
  - UMLS (Unified Medical Language System) http://umlsks.nlm.nih.org
  - Dublin Core http://dublincore.org

- ontologie de tâche
  - ONTOLINGUA (http://ksl.stanford.edu/software/ontolingua/)

entrepôt d'ontologies : tones : http://owl.cs.manchester.ac.uk/repository/

2.12 Cycles de vie d'une ontologie

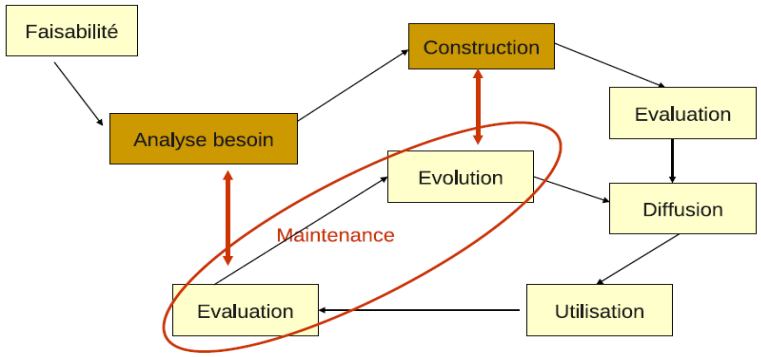


Figure 8.15: Cycles de vie d'une ontologie

Etude de faisabilité

- Rôle du système visé
- Situation du système dans l'organisation
- Situation de l'ontologie dans le système
- Identification des acteurs concernés
- Approche ergonomique, démarche participative
- Ontologie : domaine couvert
- Capacité trouver les connaissances requises

2.13 Principe de construction d'ontologie

Clarté : objectivité, documentation

Cohérence : pas de contradiction

Extensibilité : spécialisation incrémentale

Minimiser le biais du codage : niveau connaissance

Minimiser l'engagement ontologique : partage

- Modéliser les connaissances avec les experts du domaine

- Exploiter les différentes ressources avec techniques appropriées
- Réutilisation d'ontologies
- Adaptation de terminologies
- Analyse de données
- Analyse automatique de documents
- Entretiens avec les experts
- Modéliser avec un langage conceptuel d'ontologie
- Raffiner concepts et relations
- Identifier les axiomes
- Formaliser
- Parvenir l'ontologie ciblée

2.13.1 Processus de construction

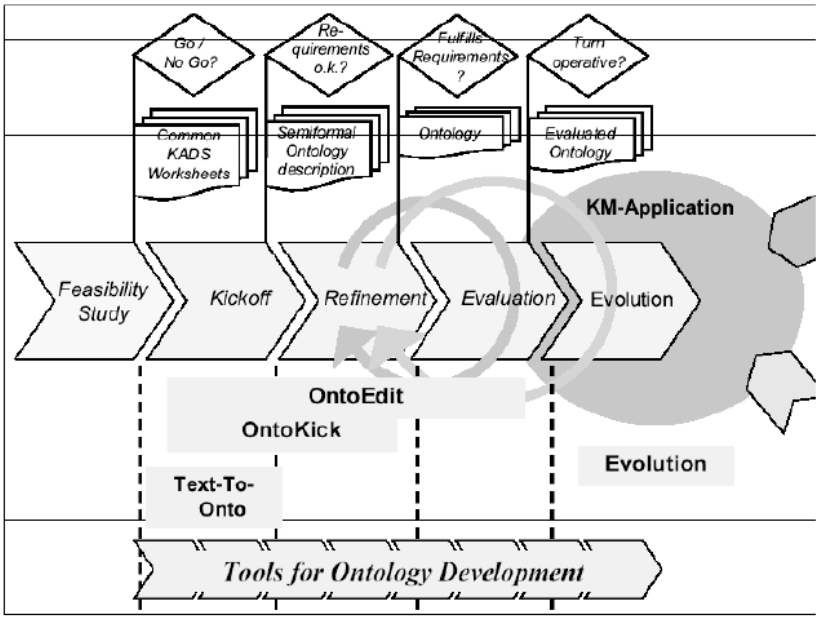


Figure 8.16: Processus de construction.

- Faisabilité

- Les systèmes de gestion de connaissances ne fonctionnent Correctement que s'ils sont intégrés dans l'organisation
- Plusieurs facteurs (autres que technologiques) déterminent la réussite
- Bien délimiter le domaine
- Identifier les personnes impliquées

- Démarrage

- Etablir un document de spécifications :
  - Domaine, objectif, sources de connaissances disponibles, utilisateurs potentiels, cas d'utilisations, applications
- Analyser les sources de connaissances
- Où sont les compétences ? Quels sont les concepts pertinents ? Y a-t-il d'autres ontologies utilisables ? …
- Prototype
  - Concepts et relations les plus importants

- Raffinement

- Acquisition de la connaissance auprès des experts du domaine et de leurs documents.
- Formalisation (Logique de Description, RDF, OWL, … :
  - Choix certaines entités sont des concepts ? des attributs ? · · ·
- Développement et raffinement de l'ontologie cible.

- Evaluation

- Vérifier le document de spécification
- Tester l'application cible
- Déployer l'ontologie

- Maintenance et évolutions

- Évolution : les conditions et les spécifications de l'ontologie peuvent changer :
  - Qui s'occupe de la maintenance ?
  - Comment est-ce fait ?
- Comment évoluent les applications qui utilisent l'ontologie ?

- Réutilisations

- C'est l'idée de départ
  - En réalité difficile :
  - Ontologies de haut niveau
- inter-opérabilité des ontologies

N.B

- Des outils existent:
- OntoEdit
- PROTEGE (utilisé pour les séances de TP)
- http://protege.stanford.edu/

2.14 utilisations des ontologies

- Web sémantique
- E−commerce
- Gestion des connaissances
- Extraction d'informations, recherche d'informations
- E−learning
- Ingénierie des bases de données
- Traduction
- ….

2.15 Langages de représentation pour les ontologies

- RDF et RDF Schéma : ontologies simples
- OWL le standard du W3C :
- Basé sur les logiques de description
- Sémantique

3. RDF : langage de description du Web sémantique

- RDF pour Resource Description Framework développé par le W3C et bas_e sur le RDF data model qui est le modèle introduit précédemment, c'est-_à-dire le modèle d'entrepôts de triplets de la forme (s,p,o).
- La norme RDF spécifié également la façon de d'écrire les ressources (sujets et objets) et les prédicats,
- La norme RDF spécifié également les formats d'échange de fichiers de ressources sur le web (sérialisation)
    - **RDF : un langage pour rendre les machines intelligentes**

3.1 RDF sur un exemple

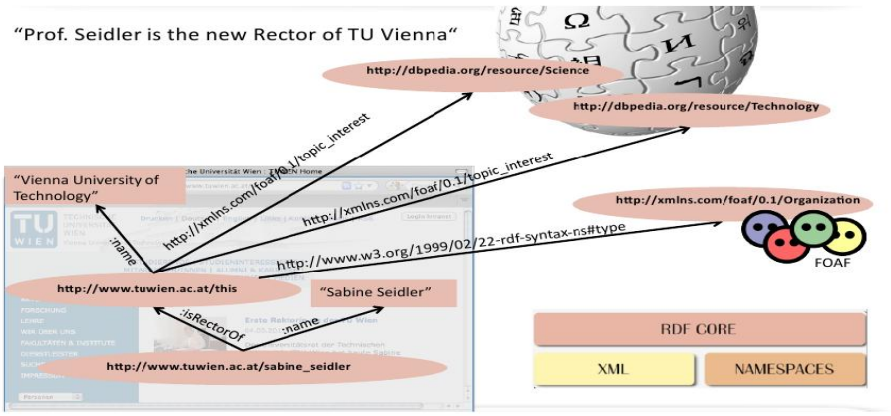


Figure 8.17: Exemple 1 de RDF.

3.2 RDF : identification des ressources

La norme RDF conçoit toute chose de l'univers comme une ressource qui peut être identifiée par une URI

Les URIS

- Une URI (Uniform Resource Identifier) identifie toute ressource qu'elle soit accessible _électroniquement ou pas. Exemple :
  http ://fr.dbpedia.org/page/Lille,
- Toute URL est une URI,
- L'URI n'est pas l'objet mais l'identifiant d'un objet qui peut être représenté dans un graphe RDF
- Les URIS sont faites pour les machines, cependant les espaces de nom (éventuellement locaux) vont permettre de simplifier les écritures.

3.3 RDF : et si la ressource n'a pas d'URI ?

Il se peut que l'on ne dispose pas d'URI pour la ressource dont on souhaite parler. Par exemple, dans les réseaux sociaux, on ne dispose pas d'URI pour les membres. On crée alors un nœud vide (nœud blanc, nœud anonyme, \blank node").

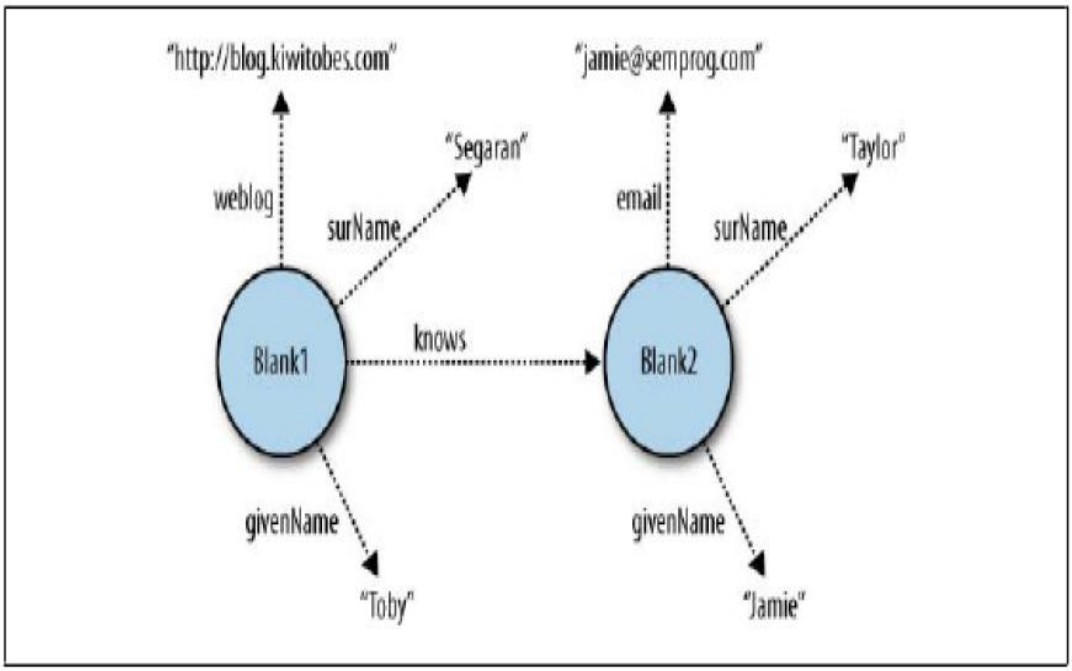


Figure 8.18: Exemple 2 de RDF.

Une personne de nom Toby Segaran auteur du blog kiwitobes connaît une personne nommée Jamie Taylor dont l'email est jamie@semprog.com

3.3.1 Sémantique et syntaxe pour les nœuds vides

Les noeuds vides

- La sémantique : $\exists X, \exists Y, (X, \text{givenName}, \text{“Toby”}), \ldots, (X, \text{knows}, Y), \ldots, (Y, \text{email}, \text{“jamiesemprog.com”})$
- La syntaxe : noeuds vides abstraits par un nom de variable `_:id`. Ce qui donne (`_:person1`, givenName, “Toby”), ..., (`_:person1`, knows, `_:person2`), ..., (`_:person2`, email, “jamiesemprog.com”)

- Autre utilisation des nœuds vides

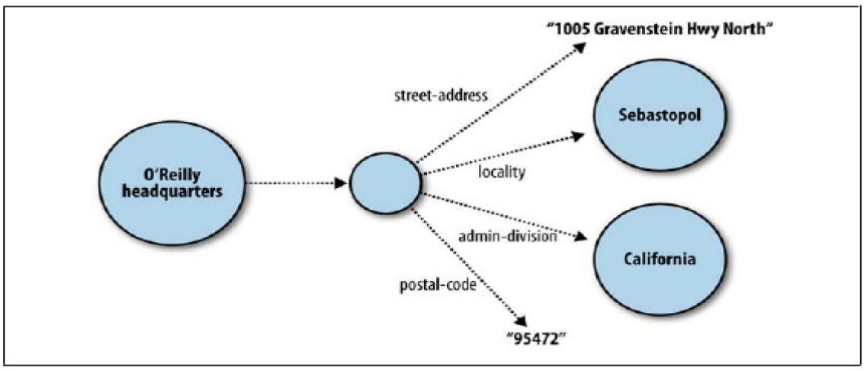


Figure 8.19: Autre utilisation des nœuds vides.

3.4 Les valeurs en RDF

Les valeurs littérales

- Peuvent être typées
- Avec la spécification de XML schema. Voir par exemple

  http ://www.w3.org/2001/XMLSchema#int,
- Elles peuvent être exprimées dans une langue, en général,
- En respectant souvent la norme ISO 639 (en, fr, ja, ...).

Résumons

Un modèle de données en triplets de la forme spo : (sujet, prédicat, objet) représenté avec le langage RDF dans lequel

- Les ressources sont identifiées par des URIS, les uris peuvent être simplifiées par des espaces de nom, les URIS peuvent référer à des vocabulaires (rdf, foaf, dbpedia, ...),
- Ou représentées par des nœuds vides avec des variables locales _ :id,
- Les valeurs peuvent être typées et associées à une langue.

3.5 Sérialisation en RDF

Pour échanger des données de triplets, il faut les sérialiser dans des fichiers en respectant des contraintes communes pour émetteur et receveur. Il existe de nombreux formats de sérialisation.

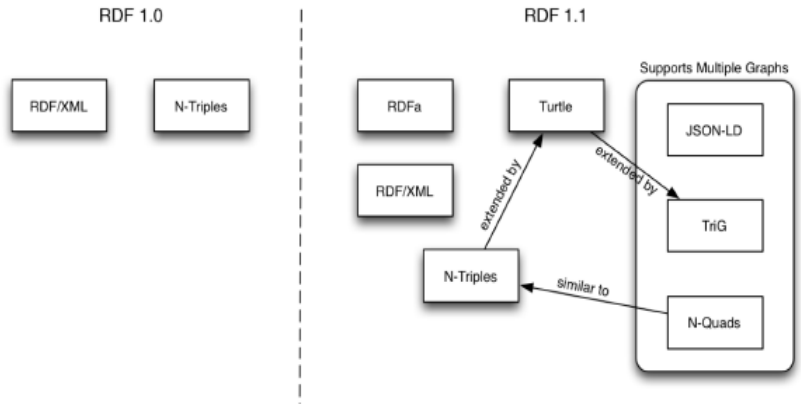


Figure 8.20: Sérialisation en RDF.

- Les principaux formats

- N-triples : les triplets sont échangés dans leur entièreté. Tout est représenté par URI sauf les valeurs et les nœuds vides. Il existe plusieurs sérialisations sous la forme de triplets. I La plus basique est la syntaxe N-triples, qui écrit chaque triplet sous la forme

  <IRI du sujet> <IRI du prédicat> <IRI de l'objet ou littéral>

  Par exemple :

```
<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows>
<http://example.org/alice#me> .

<http://example.org/bob#me> <http://schema.org/birthDate>
"1990-07-04"^^<http://www.w3.org/2001/XMLSchema#date> .
```


- N3, Turtle : on condense l'écriture avec l'utilisation de préfixés, des factorisations et des raccourcis de notation. La syntaxe Turtle reprend celle des N-triples en y ajoutant des facilités syntaxiques pour rendre le code plus lisible:

```
BASE   <http://example.org/>
   PREFIX foaf: <http://xmlns.com/foaf/0.1/>
   PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
   PREFIX schema: <http://schema.org/>
   PREFIX dcterms: <http://purl.org/dc/terms/>
   PREFIX wd: <http://www.wikidata.org/entity/>

   <bob#me>
       a foaf:Person ;
       foaf:knows <alice#me> ;
       schema:birthDate "1990-07-04"^^xsd:date ;
       foaf:topic_interest wd:Q12418 .

   wd:Q12418
       dcterms:title "Mona Lisa" ;
       dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .

   <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D6
       dcterms:subject wd:Q12418 .
```

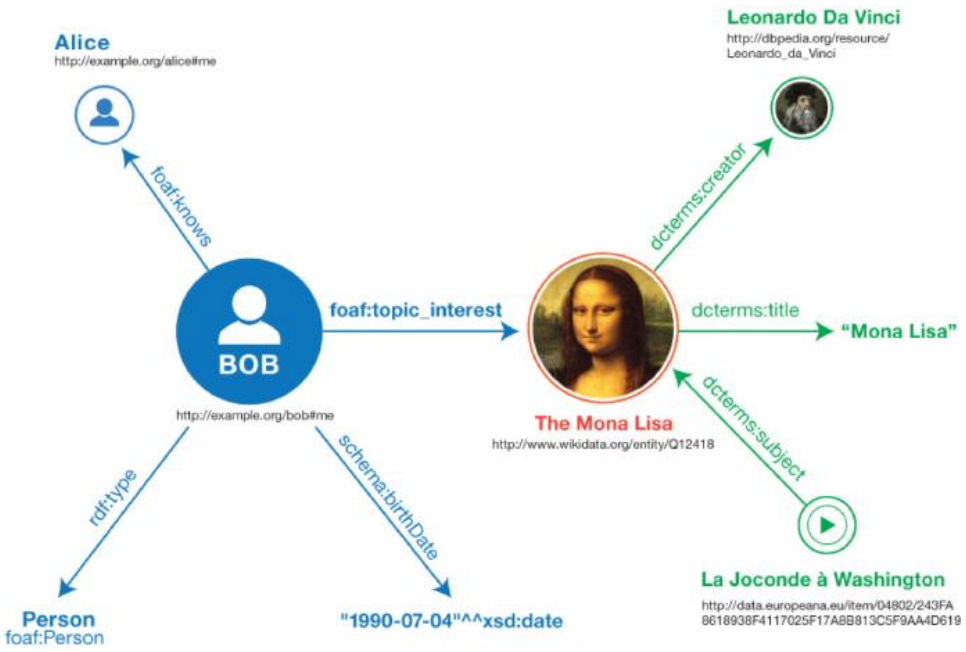


Figure 8.21: Forme de triplets.

- RDF/XML : syntaxe XML pour décrire les fichiers RDF. Cette syntaxe est très utilisée mais pas facile à lire (par l'humain).
  - Elément *Description* pour décrire une ressource (attribut about pour le sujet, sous-élément pour la propriété, contenu du sous-élément pour la propriété (qui peut être parfois simplifié en attribut).
  - On peut regrouper dans un même élément *Description* toutes les propriétés dont cette ressource est sujet.
  - Exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
       xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:foaf="http://xmlns.com/foaf/0.1/"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:schema="http://schema.org/">
<rdf:Description rdf:about="http://example.org/bob#me">
  <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  <schema:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1990-07-04</schem
  <foaf:knows rdf:resource="http://example.org/alice#me"/>
  <foaf:topic_interest rdf:resource="http://www.wikidata.org/entity/Q12418"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.wikidata.org/entity/Q12418">
  <dcterms:title>Mona Lisa</dcterms:title>
  <dcterms:creator rdf:resource="http://dbpedia.org/resource/Leonardo_da_Vinci"/>
</rdf:Description>
<rdf:Description rdf:about="http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B
  <dcterms:subject rdf:resource="http://www.wikidata.org/entity/Q12418"/>
</rdf:Description>
</rdf:RDF>
```

Illustration par l'exemple

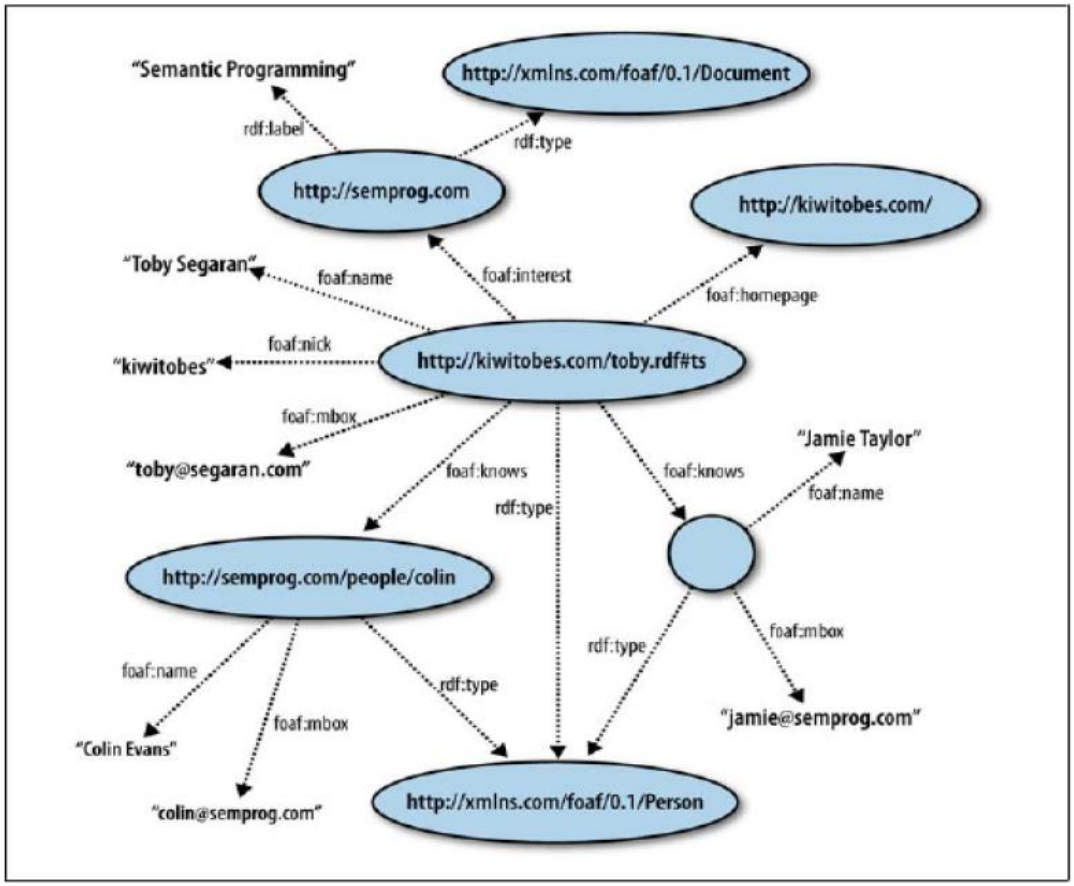


Figure 8.22: RDF/XML.

3.6 Réification

- En informatique, la réification consiste à transformer un concept en un objet informatique.
- Par exemple : langage orienté objet avec mécanisme de réflexion, on peut réifier une classe, qui devient instance d'une (méta-)classe.
- En RDF, la réification permet de considérer un triplet comme un nœud.

Soit le graphe G_1 :

```
<ex:un_sujet> <ex:une_propriete> <ex:un_objet> .
```

Le graphe G_2 ci-dessous est une réification de G_1 :

```
_:xxx rdf:type rdf:Statement .
_:xxx rdf:subject <ex:un_sujet> .
_:xxx rdf:predicate <ex:une_propriete> .
_:xxx rdf:object <ex:un_objet> .
```

3.7 Collections

En RDF, une collection est une liste "`a la LISP" :

- de type *rdf:List*, avec un premier élément *rdf:first* et une suite *rdf:rest*. La liste vide a la valeur *rdf:nil*.

- Exemple :

```
_:c1 rdf:first <ex:aaa> .
_:c1 rdf:rest _:c2 .
_:c2 rdf:first <ex:bbb> .
_:c2 rdf:rest rdf:nil .
```

- Une collection est une liste fermée : elle forme un groupe qui ne contient que les membres spécifiés lors de la déclaration de la collection.
- Exemple de collection :

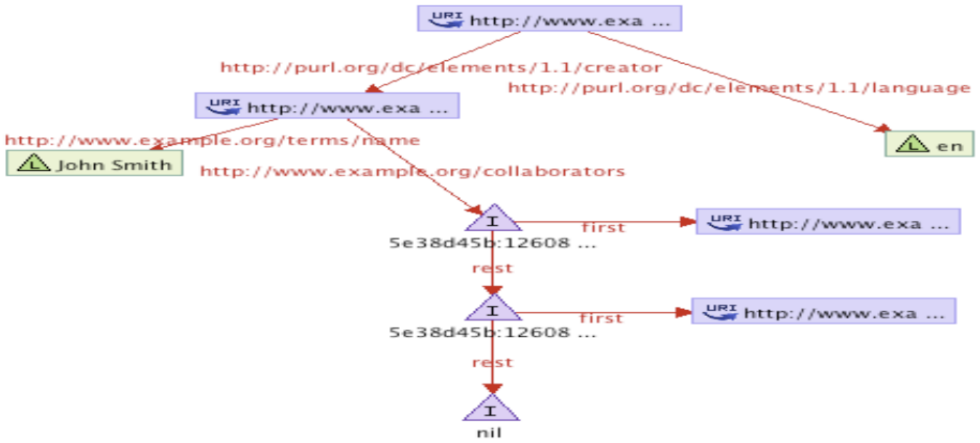


Figure 8.23: Exemple de collection.

3.8 Les ”Containers”

- Ils permettent de décrire des groupes. Les choses contenues dans un container sont appelées membres du groupe.
- Il existe 3 types de containers prédéfinis :
  - *rdf:Bag* : multi-ensemble de ressources ou littéraux
  - *rdf:Seq* : Séquence de ressources ou littéraux (l'ordre est important)
  - *rdf:Alt* : alternatives entre plusieurs ressources ou littéraux
  - Pour indiquer qu'une ressource est un container, on utilise la propriété *rdf:type*.
- On dit que les containers sont ouverts, i.e. il peut exister d'autres membres du container que ceux indiqués par la description dont on dispose.

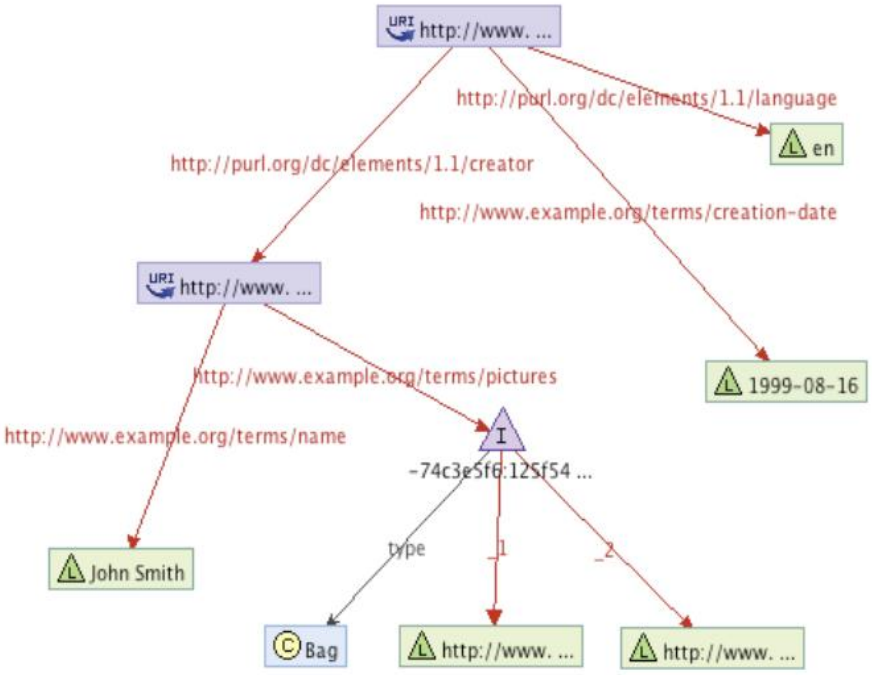


Figure 8.24: Les ”Containers”.

Conclusion :

Les formalismes de représentation des connaissances ont été introduits dans ce chapitre. Un aperçu général du web sémantique a d'abord été fourni. Les ontologies et leurs idées fondamentales ont ensuite été introduites. Ce chapitre s'est terminé par une explication approfondie des langages de description (RDF).

Références

Références

1. Allan M. Allan M. Collins et Elizabeth F. Loftus, "A spreading-activation theory of semantic processing", Psychological Review, vol. 82, no 6, p. 407–428, 1975.
2. Diana Kalibatiene et Olegas Vasilecas, "Survey on Ontology Languages", dans Perspectives in Business Informatics Research, vol. 90, Springer Berlin Heidelberg, p. 124–141, 2011.
3. F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, P. F. Patel-Schneider : The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, 2003.
4. Gerber, Aurona ; Van der Merwe, Alta ; Barnard, Andries, "A Functional Semantic Web architecture", European Semantic Web Conference 2008 ESWC'08, Tenerife, juin 2008.
5. J.F. Sowa, "Semantic Networks, in Stuart C Shapiro", Encyclopedia of Artificial Intelligence, 1987.
6. James Hendler, Nigel Shadbolt, Wendy Hall, Tim Berners-Lee, Daniel Weitzner, "Web science: an interdisciplinary approach to understanding the web", Communication ACM 51, 7, 2008.
7. John F. Sowa and John Zachman, "Extending and Formalizing the Framework for Information Systems Architecture" In: IBM Systems Journal, Vol 31, no.3, p. 590-616, 1992.
8. Marite Kirikova, "Information Systems Development: Advances in Methodologies", Components, and Management. p.194, 2002.
9 Nkambou Roger, Gauthier Gilles, Frasson Claude, "Un modèle de représentation des connaissances relatives au contenu dans un système tutoriel intelligent", In: Sciences et techniques éducatives, volume 4 n°3, pp. 299-330, 1997.
10 Paquette, G., "Modélisation des connaissances et des compétences : un langage graphique pour concevoir et apprendre", Sainte-Foy: Presses de l'Université du Québec, 2002.
11 Paquette, G., "Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies", Hershey, PA: IGI Global, 2010.
12 R. Jackendoff, "Semantic Structures", The MIT Press, Cambridge Mass, 1990.
13 Russel &Norvig, "Artificial Intelligence a Modern approach", Prentice Hall Series in Artificial Intelligence, 1995.
14 W.A. Woods, "What's in a Link: Foundations for Semantic Networks", Bolt, Beranek and Newman, 1975.