Représentation des connaissances

La représentation des connaissances constitue un enjeu majeur en intelligence artificielle. En effet, la façon de représenter les connaissances associées à un problème considéré peut faire en sorte que sa résolution soit plus ou moins facile. De façon générale, l'utilisation de plusieurs modes de représentation des connaissances permet une mise en œuvre plus efficace des outils de traitement des connaissances dans le cadre de la résolution de problèmes spécifiques.

L'objectif de ce document est de décrire différents formalismes utilisés en intelligence artificielle pour la représentation des connaissances. À travers les exemples présentés, on devrait pouvoir avoir une bonne idée des types de question à se poser en ce qui a trait à la représentation des connaissances lors de la résolution de problèmes particuliers en intelligence artificielle.

Nous verrons plus particulièrement les réseaux sémantiques, la logique des prédicats, les graphes conceptuels, les schémas, les scripts et les objets. Plusieurs de ces formalismes de représentation seront repris dans le cadre du traitement automatique des langues naturelles.

2.1 CONNAISSANCES ET RÉPRÉSENTATION DES CONNAISSANCES

On définit la connaissance ou les connaissances comme ce qu'on a appris par l'étude ou par la pratique. Dans le but de résoudre des problèmes complexes qui relèvent de l'intelligence artificielle, il faut un bon bagage de connaissances et des outils de manipulation de ces connaissances. Les connaissances concernent des faits, considérés vrais dans un certain monde. Pour représenter ces faits on a recours à un formalisme ou mode de représentation. Au niveau des faits, on traitera des objets et des relations qu'ils entretiennent. Au niveau du formalisme, on définira des symboles et des opérations sur ces symboles.

Le langage naturel est un outil très informel pour représenter les connaissances. L'arithmétique et la logique, plus formels, sont aussi des outils de représentation.

Exemples :

Grisou et Garfield sont des chats	(en langage naturel français)
$a^2 + b^2 = c^2$	(en arithmétique)
$(\forall x)(\exists y)$ [x aime y]	(en logique)

Avant d'aller plus loin dans la formalisation des connaissances, nous allons voir à travers un exemple que cette dernière peut apparaître d'une certaine complexité et qu'un bon choix de représentation peut être déterminant pour la résolution d'un problème.

2.2 CARACTÉRISTIQUES DE LA REPRÉSENTATION DES CONNAISSANCES

Considérons le célèbre problème du fermier, du renard, de l'oie et du sac de grains :

> *Un jeune fermier, propriétaire d'un renard, d'une oie et d'un gros sac de grains se trouve, avec ses biens, sur la rive gauche de la rivière Jacques-Cartier. Le fermier désire passer avec ses biens sur la rive droite de la rivière. Il dispose d'une petite barque très étroite qui ne peut supporter que son poids et celui d'un autre objet. Le fermier peut traverser la rivière autant de fois qu'il le désire mais il ne peut laisser sur la même rive l'oie et le sac de grains ni le renard et l'oie. Évidemment, il est le seul à savoir ramer. Comment le fermier doit-il s'y prendre?*

Pour avoir une bonne perception du problème, on doit s'efforcer de trouver une représentation simple et complète. On commence par identifier les objets pertinents : le fermier, le renard, l'oie et le sac de grains ainsi que la rivière et la barque.

Par ailleurs, on note qu'à chaque instant il est important de connaître la position de chaque objet. À un instant donné, la situation peut être décrite par la position du fermier et de chacun de ses biens. On parlera alors d'un état.

En étudiant les mouvements des divers objets, on note que le parcours du fermier comprend des étapes consistant chacune, à partir d'une rive, à embarquer seul ou avec un de ses biens et ramer jusqu'à l'autre rive. On pourrait définir un état au moyen d'une triple liste :

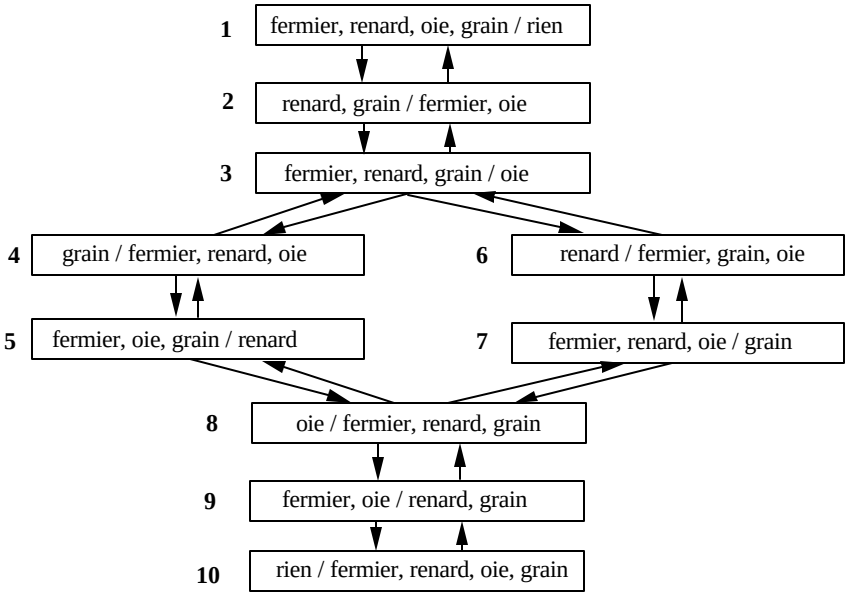
< liste rive-gauche, liste barque, liste rive-droite>

mais la double liste :

<liste rive-gauche, liste rive-droite>

est plus concise et suffit pour représenter les états pertinents.

On ne considérera qu'une partie de tous les états possibles. Par exemple, l'état < fermier, renard / oie, grain > n'a pas à être considéré puisque l'oie ne doit pas être laissée seule avec le grain. Le changement d'un état à un autre sera représenté par une flèche orientée reliant le premier au second. Le problème peut être complètement représenté au moyen du graphe suivant :



La recherche d'une solution devient alors très facile. Il est même possible de trouver plusieurs solutions.

Par exemple :

- 1, 2, 3, 4, 5, 8, 9, 10
- 1, 2, 3, 6, 7, 8, 9, 10
- 1, 2, 3, 4, 5, 4, 3, 6, 7, 8, 9, 10.

Idéalement, une bonne représentation est celle qui permet une résolution presque immédiate du problème. Elle doit donc être transparente, concise, complète et doit permettre un accès rapide à l'information, selon une démarche procédurale.

Une représentation doit comprendre quatre parties fondamentales :

- **lexicale :** Quels sont les symboles autorisés pour représenter objets et relations ?
- **structurelle :** Quelles sont les contraintes d'arrangement de ces symboles ?
- **procédurale :** Comment créer et modifier l'information ?
- **sémantique :** Comment associer un sens aux descriptions formelles ?

Dans le problème du fermier, la représentation comprenait :

- au niveau **lexical** : les nœuds et les flèches pour représenter le problème ;
- au niveau **structurel :** une flèche reliait deux nœuds ;
- au niveau **sémantique :** un nœud était associé à chaque couple de listes d'objets et une flèche à chaque changement d'état ;
- au niveau **procédural** : des mécanismes mentaux permettent de dessiner le graphe en fonction de la perception du problème ainsi que de l'interprétation qui serait faite de la représentation symbolique.

Nous allons maintenant étudier, comme premier formalisme de représentation des connaissances, les réseaux sémantiques. Ce formalisme servira en fait d'introduction à la logique clausale qui lui est semblable sous bien des rapports, mais qui offre plus de flexibilité pour la représentation des connaissances.

2.3 LES RÉSEAUX SÉMANTIQUES

Nous allons commencer par définir, de façon assez informelle, les éléments qui interviennent dans le domaine des connaissances : **objets, fonctions, relations, prédicats**.

2.3.1 Constituants de la connaissance

Les objets peuvent être de diverses natures. Il peut s'agir d'entités concrètes comme un livre, la lune, Napoléon, le capitaine Crochet, etc. ou encore de concepts abstraits comme l'amour, la peur, la justice. De plus, un objet peut être primitif comme les précédents ou composé comme un ordinateur ou la double liste dans le problème du fermier.

Soit A et B deux ensembles. Le produit cartésien de A et B, noté A x B, est défini de la façon suivante : $\{(x, y) \mid x \in A \text{ et } y \in B\}$. Tout sous-ensemble de A x B est appelé une relation de A vers B. Une fonction f de A vers B est une relation qui fait correspondre au plus un seul élément de B à un élément de A.

Exemples :

$A = \{a, b\}$, $B = \{c, d\}$, $A \times B = \{(a, c),(a, d),(b, c),(b, d)\}$

$R1 = \{(a, c), (b, d)\}$ $R2 = \{(a, d),(b, c),(b, c)\}$

$f1 = \{(a, d),(b, c)\}$

Une fonction n-aire f fait donc correspondre un seul objet $f(x_1, x_2, \ldots, x_n)$ à une liste de n objets $x_1, x_2, \ldots, x_n$.

Exemples de fonctions :

âge(Pierre) : l'âge de Pierre

père(Andrée) : le père d'Andrée

entier_entre(2, 4) : l'entier entre 2 et 4

Un prédicat exprime une relation entre objets. Les prédicats père, mère, homme, femme, étudiant, professeur expriment des relations évidentes. On peut associer une valeur de vérité (vraie ou fausse) à une relation. Par exemple, les relations suivantes sont vraies : 2<3, chat(Grisou). L'arité d'une relation est le nombre de ses arguments. Le choix de l'arité pour une relation est souvent une question d'interprétation personnelle. Par exemple, on pourrait écrire a_des_griffes(Garfield) ou muni(Garfield, griffes). La relation a_des_griffes a une arité de 1 alors que la relation muni a une arité de 2. Par ailleurs, l'arité est normalement fixe, cependant il est possible d'accepter une arité multiple dans certains contextes. Par exemple, fils(Andrée, Marcel) et fils(Andrée) pourraient coexister si Marcel est le mari d'Andrée.

Objets, fonctions et relations peuvent être combinés pour former des propositions de complexité variable. On distingue les propositions atomiques des propositions composées. Par exemple, la proposition chat(Grisou) est atomique alors que chat(Grisou) et chat(Garfield) est composée puisqu'elle utilise le connecteur logique et. Par contre, chats(Garfield, Grisou) est atomique. 3≤4 est atomique mais elle est équivalente à 3 < 4 ou 3 = 4 qui ne l'est pas.

2.3.2 Définition du réseau sémantique

- Au niveau lexical, un réseau sémantique comprend des nœuds (un nœud représente un objet primitif), des liens orientés ou arcs (un arc représente une relation entre deux objets) et des étiquettes d'arcs (les arcs sont étiquetés en fonction des relations qu'ils représentent).
- Au niveau structurel, un nœud est généralement représenté au moyen d'un rectangle, d'un cercle, d'une ellipse. Un arc relie un nœud source (queue de l'arc) à un nœud cible (tête de l'arc).
- L'interprétation sémantique du réseau dépendra de chaque application concernée.
- Au niveau procédural, on se donnera des mécanismes de construction, de lecture, d'écriture et d'effacement de réseaux ou parties de réseaux.

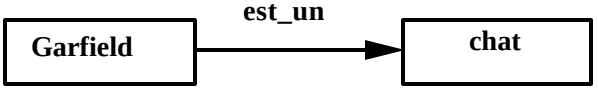
On voit déjà que le réseau sémantique est assez restrictif puisqu'il ne permet de représenter que des objets primitifs et des relations binaires (à la rigueur, des relations unaires en utilisant des boucles).

Les réseaux sémantiques utilisent généralement deux relations très particulières concernant des objets de l'un des types individu ou classe :

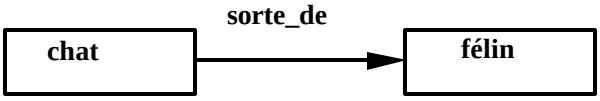
- **est_un :** relation entre un individu et une classe exprimant l'**appartenance** ;
- **sorte_de :** relation entre deux classes exprimant l'**inclusion**.

Exemples :

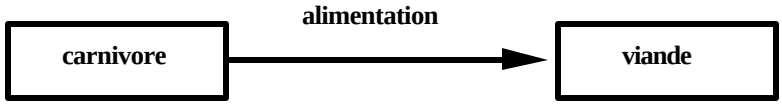
- Garfield est un chat.



- Les chats sont des félins.

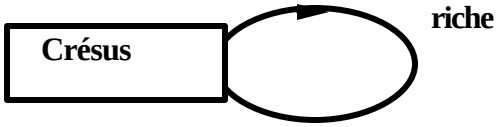


- *Les carnivores mangent de la viande.* (équivalent à *tout individu de la classe carnivore a la propriété de se nourrir de viande*).



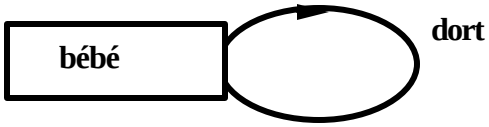
Une proposition atomique exprimant un prédicat d'arité 1 est représentée par un graphe comprenant un nœud auquel est attachée une boucle :

- *Crésus est riche.*

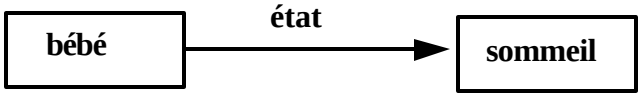


Soit l'équivalent binaire :

- *Le bébé dort.*



Soit l'équivalent binaire :



2.3.3 Construction du réseau sémantique

Dans les exemples précédents, on n'a représenté que des propositions atomiques. Le réseau sémantique peut permettre également de représenter tout un discours. Il s'agit alors de scinder ce discours en une suite de propositions atomiques.

Exemple :

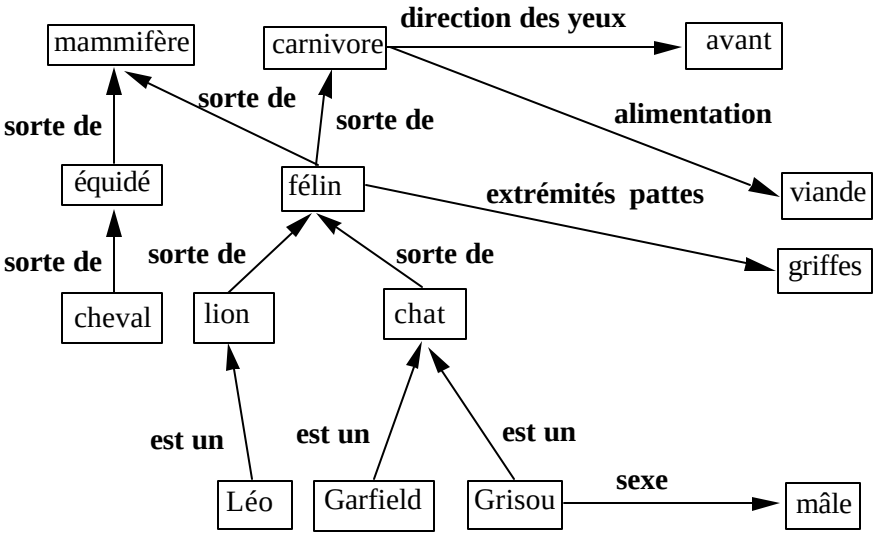
> *Un félin est un carnivore. Un carnivore est un animal qui a les yeux dirigés vers l'avant et qui mange de la viande. Les pattes d'un félin ont des griffes à leurs extrémités. Un félin est un mammifère. Grisou et Garfield sont des chats. Grisou est un chat mâle. Léo est un lion. Les chats et les lions sont des félins. Un cheval est un équidé. Un équidé est un mammifère.*

Propositions atomiques :

félin	sorte_de	carnivore
carnivore	direction des yeux	avant
carnivore	alimentation	viande
félin	extrémités pattes	griffes
félin	sorte_de	mammifère
Grisou	est_un	chat
Garfield	est_un	chat
Grisou	sexe	mâle
Léo	est_un	lion

chat	sorte_de	félin
lion	sorte_de	félin
cheval	sorte_de	équidé
équidé	sorte_de	mammifère

Réseau sémantique associé :



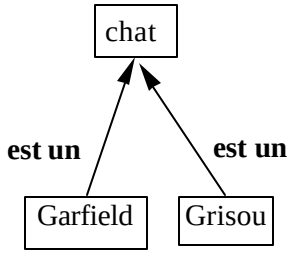
Pour construire ce réseau sémantique :

- On lit d'abord tout le texte pour repérer les objets et les relations. En effet, leur réutilisation d'une phrase à l'autre permet de mettre en évidence leur utilité.
- On sépare les propositions atomiques et on les met sous forme tabulaire.
- On dessine successivement les parties de réseau correspondant aux propositions atomiques.
- Éventuellement, on revient sur les choix effectués et on rectifie le réseau.

2.3.4 Énoncé conditionnel

Jusqu'à présent on n'a représenté, à l'aide d'un réseau sémantique, que des propositions atomiques. Cependant, les réseaux sémantiques permettent la représentation de propositions composées.

Dans l'exemple des animaux, on voit que la proposition composée *Grisou et Garfield sont des chats* est représentée par la partie du réseau suivante :



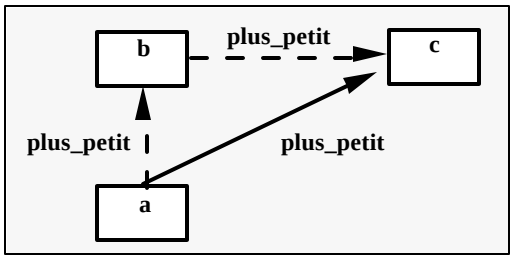
Mais comment représenter un **énoncé conditionnel**, soit une proposition composée de la forme :

Si p_1 et p_2 et ... p_n alors q où les p_i $(1 \geq i \geq n)$ et q sont des propositions atomiques.

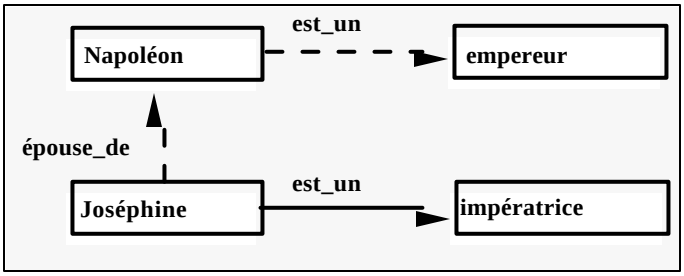
- Chacune des proposition $\mathbf{p_i}$ est un **antécédent** et la proposition $\mathbf{p_1}$ **et** $\mathbf{p_2}$ **et … et** $\mathbf{p_n}$ est dite **proposition de queue.**
- q est la proposition de **tête** ou **conséquent**.
- La proposition conditionnelle est représentée au moyen d'un réseau sémantique comprenant des arcs en **pointillé** (les relations antécédentes) et un arc **plein** (la relation conséquente).

Exemples

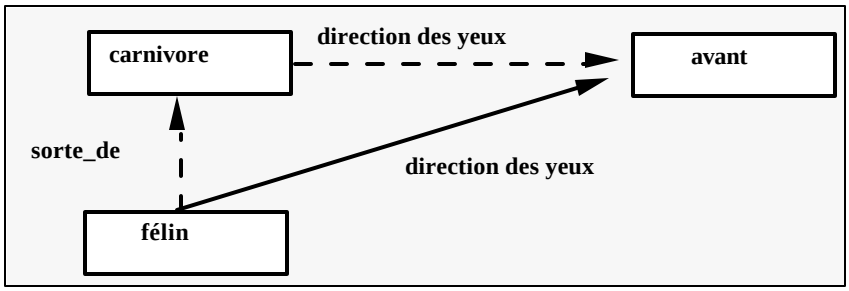
- *Si a < b et b < c alors a < c.*



- *Si Napoléon était empereur et Joséphine était son épouse alors Joséphine était impératrice.*



- *Si les félins sont des carnivores et les carnivores ont les yeux dirigés vers l'avant alors les félins ont les yeux dirigés vers l'avant.*



Remarque : On n'affirme pas dans le réseau précédent que les félins sont des carnivores ni que les carnivores ont les yeux dirigés vers l'avant. On affirme seulement que si ces deux relations sont vraies alors on en *déduit* que les félins ont les yeux dirigés vers l'avant. Ce réseau peut donc être utilisé comme outil pour faire de la déduction.

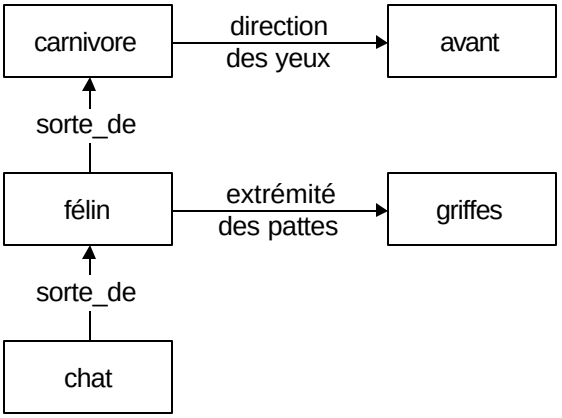
2.3.5 Réseau auxiliaire

Le réseau correspondant à une proposition conditionnelle est appelé **réseau auxiliaire.** Les réseaux auxiliaires sont utilisés comme outils de transformation des réseaux sémantiques afin d'établir des faits nouveaux (effectuer des déductions).

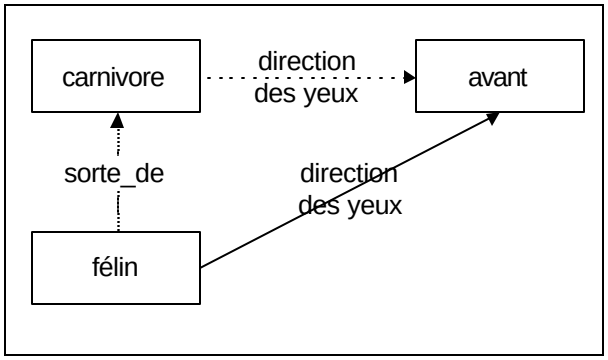
Définition : Soient un réseau sémantique R et A un réseau auxiliaire dont les nœuds et les arcs en pointillé constituent un sous-réseau identifiable à un sous-réseau de R. On dit alors que A *permet de déduire dans* R *un arc identifiable à l'arc en trait continu de* A.

Exemple :

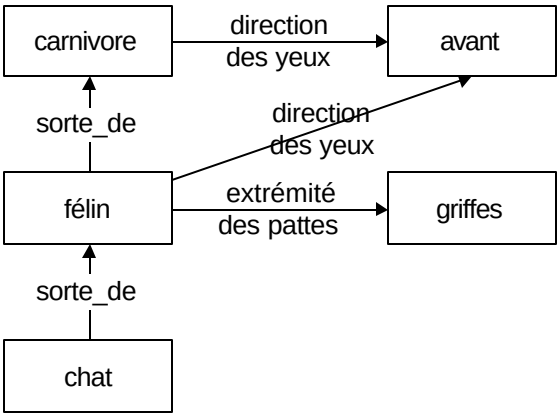
Réseau sémantique R



Réseau auxiliaire A



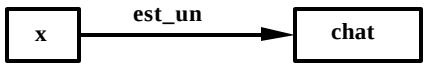
Déduction dans R de la relation *félin direction des yeux avant* à l'aide de A :



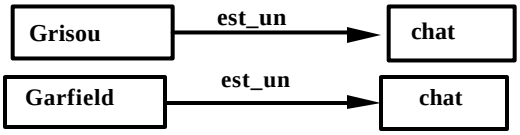
2.3.6 Utilisation de variables

Une **variable** est un symbole susceptible d'être remplacé par une valeur constante appartenant à l'ensemble des objets. Les variables ne sont pas utilisées pour représenter des prédicats. Utilisée à l'intérieur d'un réseau auxiliaire, la variable donne au nœud concerné, à tout arc incident à ce nœud et à tout le réseau un caractère de **généralité**. On utilisera, pour représenter les variables, les lettres de la fin de l'alphabet : …, W, X, Y, Z.

Par exemple, le réseau général suivant :



pourrait représenter les deux réseaux (constants) suivants :

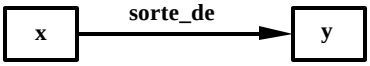


On dira qu'ici le nœud x peut être instancié avec les valeurs Grisou et Garfield. Par extension, on dit aussi que l'arc général x est_un chat est instancié aux arcs spécialisés correspondants.

La transformation du réseau général, par **substitution** d'une constante à la place de la variable, en réseau spécifique s'appelle **spécialisation** ou **instanciation.** On dit que la variable **x s'instancie** (ou encore est **valuée**) avec une constante (Grisou ou Garfield dans l'exemple précédent). On dit aussi que le réseau général est un **patron** et que les réseaux spécialisés correspondants sont ses **instances**.

Inversement, le passage d'un réseau ne contenant pas de variable (constant) à un réseau général, par remplacement d'un nœud constant par un nœud variable, s'appelle **généralisation**. Un réseau général peut comporter plusieurs nœuds variables tous susceptibles d'instanciation. Une **valuation** de ce réseau consiste à attribuer une valeur à chacun de ses nœuds variables.

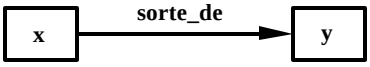
Exemple : Soit le monde des animaux, tel que décrit dans la section 2.3.3, le réseau suivant :



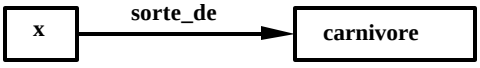
peut s'instancier de six façons :

x = chat		et	y = félin
x = lion	et	y = félin	
x = cheval		et	y = équidé
x = félin		et	y = carnivore
x = félin		et	y = mammifère
x = équidé		et	y = mammifère

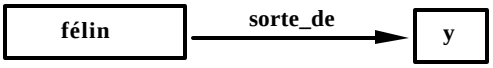
Un arc ayant ses deux nœuds identifiés par des variables peut s'instancier en un arc ayant un seul nœud identifié par une variable. Par exemple, l'arc suivant :



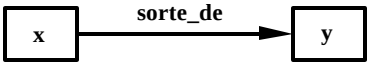
peut s'instancier en :



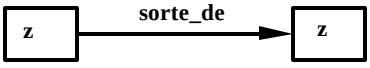
ou en :



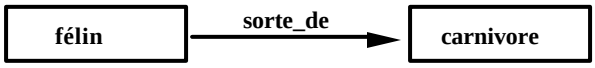
Un arc ayant ses deux nœuds identifiés par des variables différentes peut s'instancier en un arc ayant ses deux nœuds identifiés par des constantes différentes ou égales ou par la même variable. Par exemple, l'arc suivant :



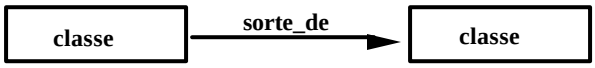
peut s'instancier en :



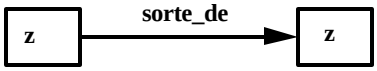
ou en :



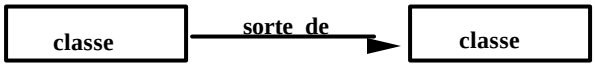
ou encore en :



Par opposition, un arc ayant ses deux nœuds identifiés par la même variable ne peut s'instancier qu'en un arc ayant ses deux nœuds identifiés par la même variable ou la même constante. Par exemple, l'arc suivant :



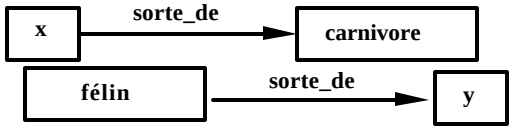
peut s'instancier en :



2.3.7 Unification (assortiment de patrons ou filtrage)

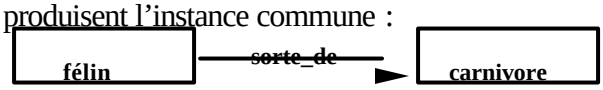
L'unification est le mécanisme permettant de réaliser des déductions à l'intérieur des réseaux sémantiques. Deux arcs sont dits **unifiables** s'ils peuvent s'instancier en un **même** arc. On parle alors d'assortiment de patrons ou d'unification des arcs.

Par exemple, les arcs suivants :

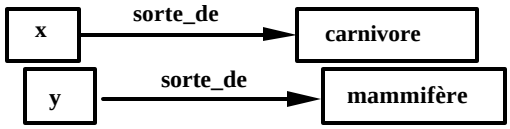


peuvent s'unifier, au moyen des substitutions : x = félin et y = carnivore

En s'unifiant, ces deux arcs produisent l'instance commune :



Par contre, les arcs suivants :



ne sont pas unifiables puisque carnivore et mammifère ne sont pas unifiables (On ne peut unifier deux constantes).

2.3.8 Conditionnelle universelle

On peut aussi faire intervenir les variables dans les propositions composées et en particulier dans les conditionnelles.

Par exemple, les deux conditionnelles suivantes :

> *C1 : Si les chats sont des félins et les félins sont munis de griffes alors les chats sont munis de griffes.*

C2 : Si les lions sont des félins et les félins sont munis de griffes alors les lions sont munis de griffes.

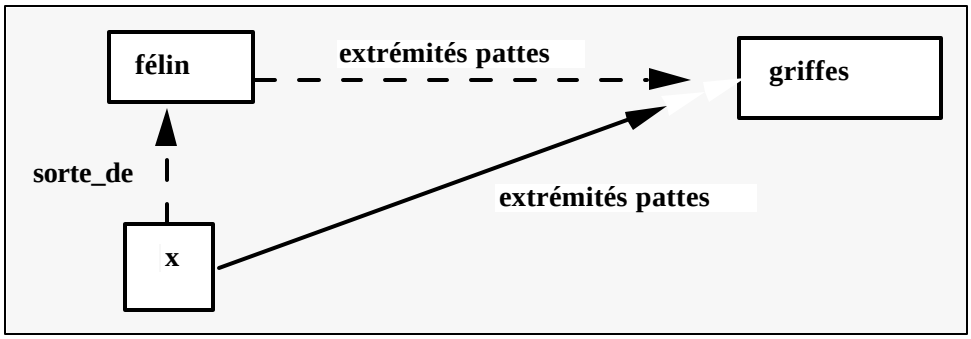
peuvent se généraliser en :

*C3 : Si **x** est un félin et les félins sont munis de griffes alors **x** est muni de griffes.*

où x est une variable valuée dans l'ensemble des félins.

Lorsque l'on substitue à **x** une de ses valeurs possibles, dans la conditionnelle universelle, on obtient une conditionnelle dite **instance** de l'universelle.

On agit de façon semblable avec les réseaux auxiliaires qui, rappelons-le, représentent les conditionnelles. Par exemple, le réseau auxiliaire universel suivant permettrait de produire, par substitutions successives, les réseaux auxiliaires correspondant aux conditionnelles précédentes C1 (x substituée par chat) et C2 (x substituée par lion).

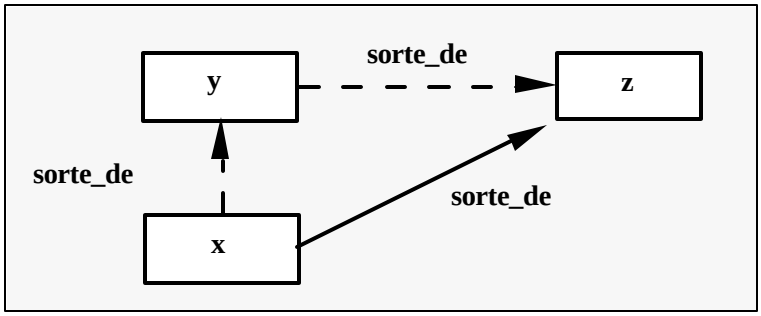


2.3.9 Déduction par application de réseau auxiliaire

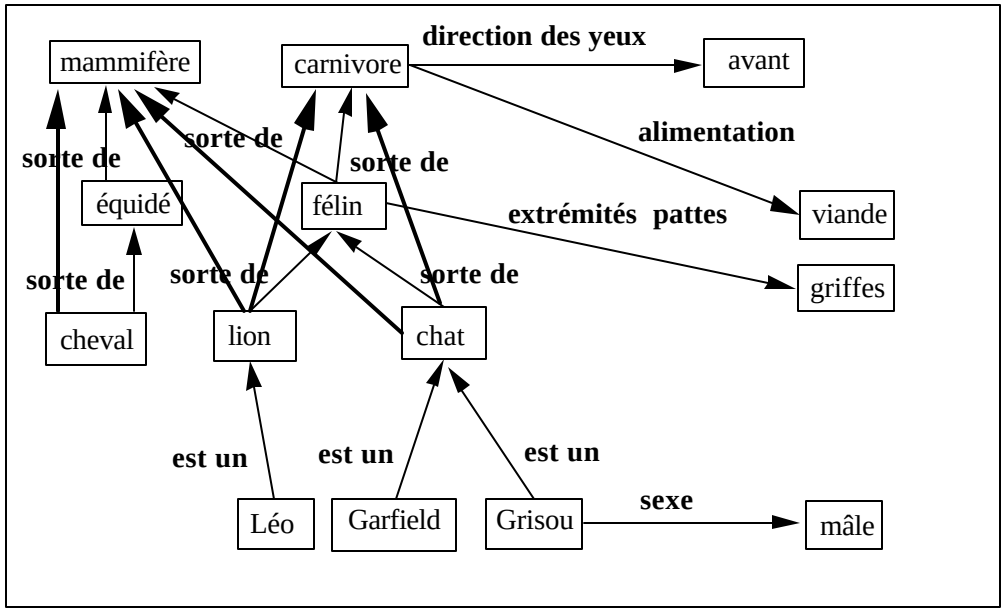
Soient un réseau sémantique R et un réseau auxiliaire A représentant une proposition conditionnelle P. Supposons maintenant que chacun des arcs en pointillé dans A (i.e. chacun des antécédents de P) puisse être instancié en un des arcs de R. Appliquer A à R consiste alors à ajouter à R l'arc en trait continu de A (correspondant au conséquent de P). Cette action permet de déduire de nouvelles connaissances dans le monde de R.

Pour résoudre un problème, on peut être amené à appliquer successivement plusieurs réseaux auxiliaires. Il faut donc être en mesure non seulement de concevoir ces réseaux auxiliaires (à partir de certains **principes de résolution,** ou encore à partir *du gros bon sens*, etc.), mais aussi de les appliquer dans le bon ordre.

Par exemple, considérons le réseau auxiliaire suivant, noté A10 :

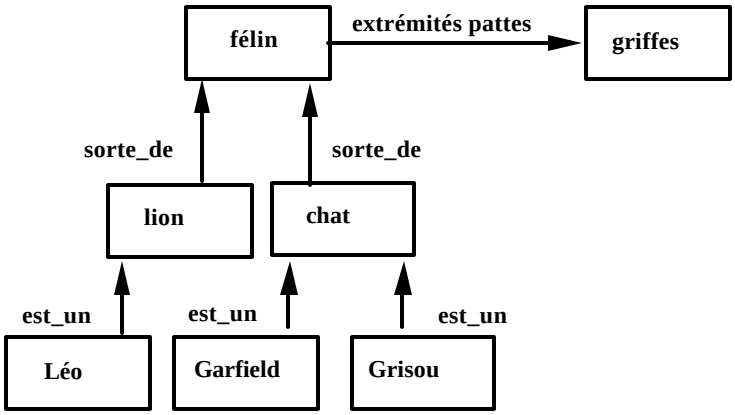


Dans le monde des animaux représenté par le réseau sémantique suivant, A10 permet de déduire les arcs en gras présents dans ce réseau :



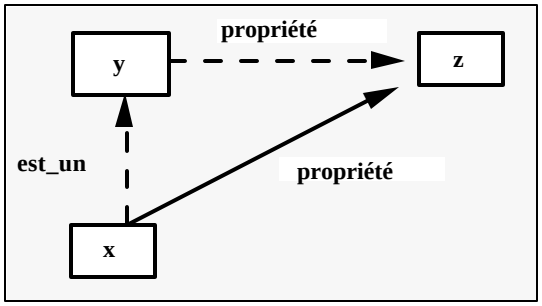
Maintenant, dans ce même monde des animaux, trouver : Q*ui est muni de griffes ?*

Pour résoudre ce problème, on commence par repérer dans l'énoncé les **mots-clés** qui vont permettre de situer, dans le réseau sémantique, les nœuds et arcs pertinents. Ici *muni* et *griffes* sont significatifs. Ils permettent d'induire le sous-réseau suivant comme étant la partie nécessaire à la résolution du problème :



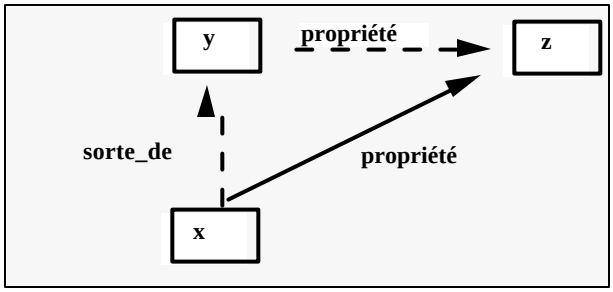
Dans cet exemple, nous allons identifier les réseaux auxiliaires qui permettront de résoudre le problème en fonction de notre bon sens. Ces réseaux auxiliaires peuvent également être construits à partir des connaissances du domaine. Notons d'abord que les arcs du sous-réseau sont étiquetés par les relations : est_un, sorte_de et extrémités pattes. Les deux premières ont des propriétés familières issues de la théorie des ensembles qu'on exprimera au moyen des réseaux auxiliaires suivants A1 et A2.

Réseau A1



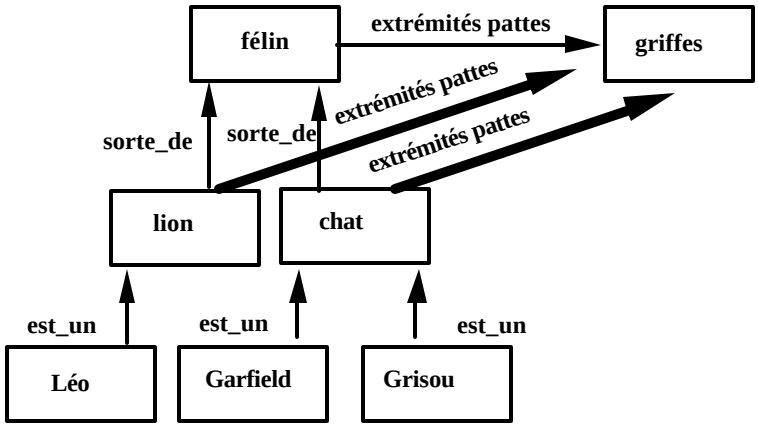
Si tous les membres de l'ensemble **y** ont la propriété **z** alors un objet **x** de cet ensemble a cette propriété.

Réseau A2

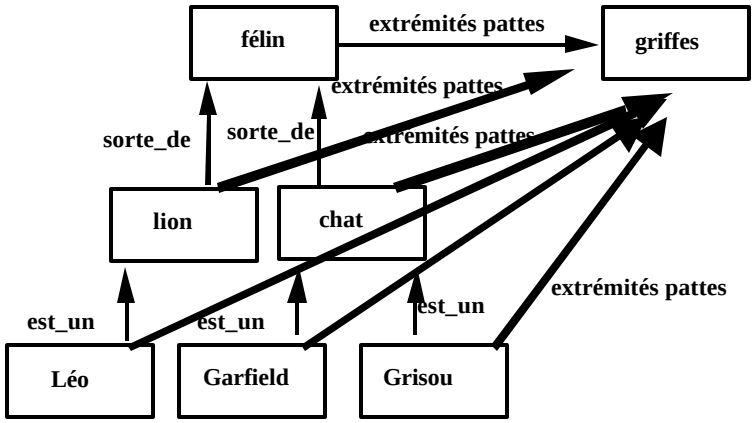


Si tous les membres de l'ensemble **y** ont la propriété **z** alors tous les membres d'un sous-ensemble **x** de **y** ont cette propriété

Poursuivons la résolution en appliquant tout à tour ces deux réseaux auxiliaires au réseau sémantique des animaux. En appliquant deux fois **A2**, on déduit que les lions et les chats sont munis de griffes.



En appliquant trois fois A1, on obtient la réponse : Léo, Garfield et Grisou sont munis de griffes.

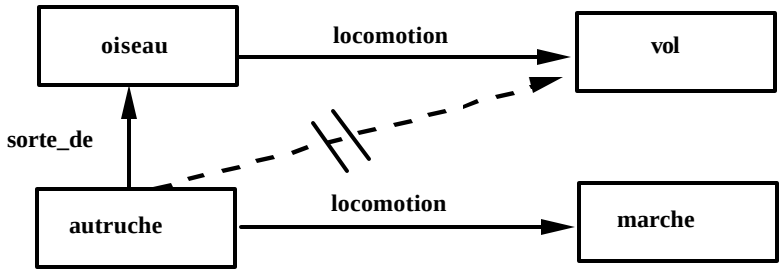


2.3.10 Héritage et propriété collective

Les relations **est_un** et **sorte_de** illustrent la notion d'**héritage** d'une propriété entre des objets spécifiés et des objets appartenant à une certaine classe. Cependant, il est important de noter que la

propriété dont il est question doit être une propriété attachée aux objets et non une propriété attachée à la classe elle-même.

Par exemple, dans le réseau suivant :



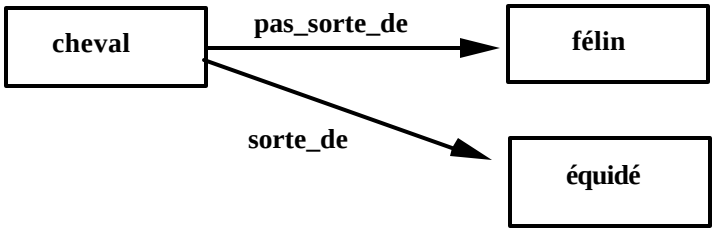
la relation locomotion est collective car elle concerne généralement l'ensemble des oiseaux, mais pas nécessairement chacun des oiseaux. En fait, la plupart des oiseaux volent, mais il existe des exceptions. La classe autruche n'hérite pas de la propriété voler de sa superclasse oiseau.

2.3.11 Preuve par réfutation

Le réseau sémantique exprime l'existence d'une relation entre deux objets au moyen d'un arc surmonté d'un prédicat. L'absence d'arc ne signifie pas que les objets ne sont pas liés par la relation. Elle signifie simplement que la relation n'a pas été établie. Alors comment affirmer la non-existence d'une relation entre deux objets ?

Non-existence d'une relation

On peut simplement créer un nouveau prédicat opposé. Par exemple, le réseau suivant qu'un cheval est un équidé mais qu'il n'est pas un félin.

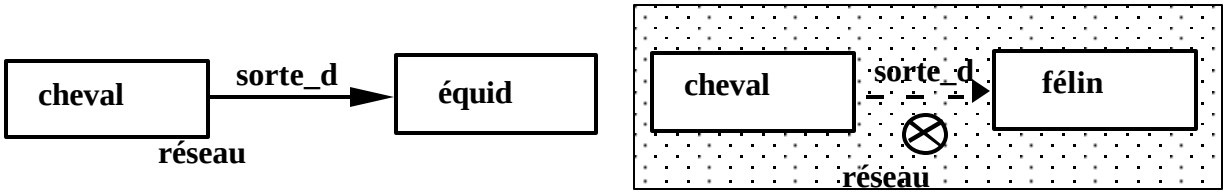


Cette approche offre divers inconvénients, à commencer par l'impossibilité de noter graphiquement le lien entre un prédicat et le prédicat contraire.

On peut aussi utiliser un réseau auxiliaire représentant une conditionnelle dont l'antécédent exprime la relation que l'on veut nier et le conséquent, appelé **falsum,** est un énoncé qui par définition est faux. Prouver qu'une relation est fausse dans un réseau consistera donc à faire apparaître la relation falsum dans ce réseau.

Falsum est noté graphiquement **:** ⊗

Dans l'exemple précédent, on aurait la représentation suivante :

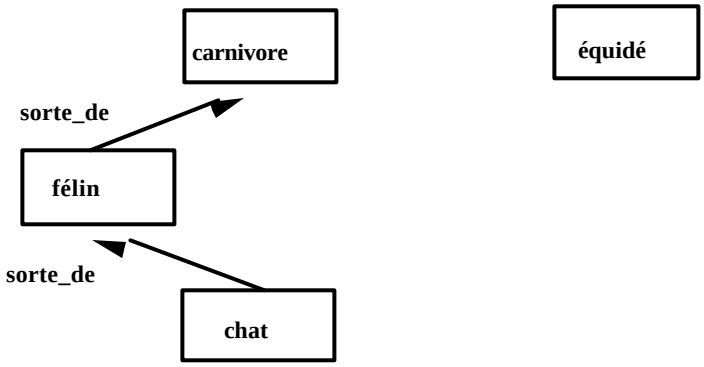


Le réseau auxiliaire permet de faire apparaître **falsum** dans tout réseau contenant l'arc cheval sorte_de félin. L'arc **falsum** sera essentiellement utilisé dans les preuves par réfutation.

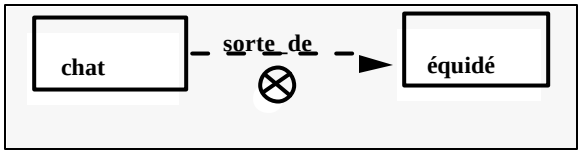
Principe de réfutation

Lorsque l'ajout d'une relation **v** entre deux objets d'un réseau R permet, suite à l'application d'un certain nombre de réseaux auxiliaires, de faire apparaître **falsum** dans ce réseau, on dit que ces deux objets ne vérifient pas la relation **v**.

Par exemple, supposons le sous-réseau du monde des animaux :



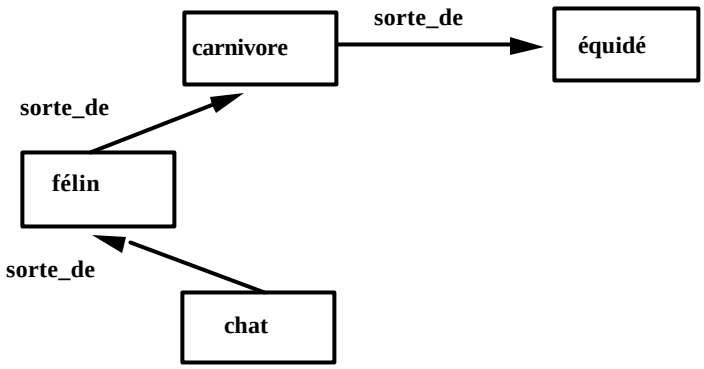
Donnons-nous maintenant le réseau auxiliaire suivant noté A11 :



Ce réseau exprime la propriété, que nous dirons connue de tous, que les chats ne sont pas des équidés, sous la forme conditionnelle : *si un chat est une sorte d'équidé alors* ***falsum****.*

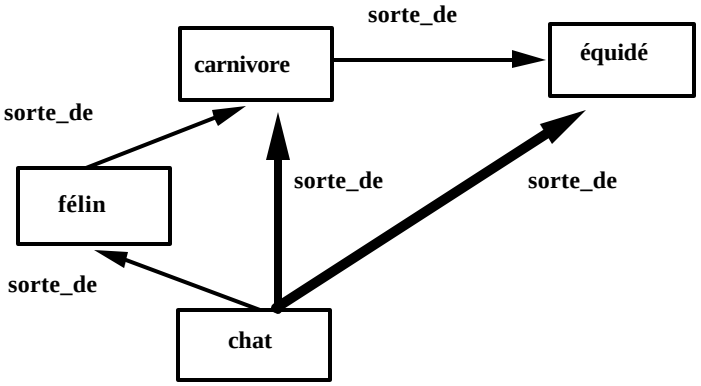
Le problème qu'on se pose maintenant consiste à démontrer **par réfutation** qu'un carnivore n'est pas un équidé.

1) On ajoute au réseau des animaux la relation dont on veut établir la **fausseté** : carnivore sorte_de équidé. (En fait, on suppose qu'un carnivore est un équidé et on va montrer qu'on obtient ensuite une contradiction).

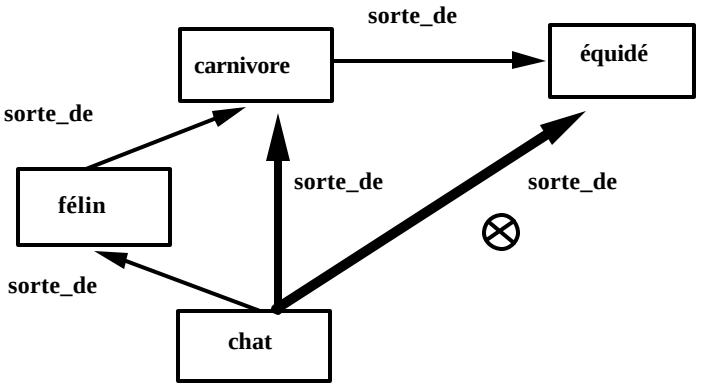


2) Par transitivité de **sorte_de**, décrite par le réseau auxiliaire A10, on déduit successivement les relations : chat sorte_de carnivore
chat sorte_de équidé

On obtient donc le réseau suivant :



3) Par application du réseau auxiliaire A11, on déduit **falsum** :

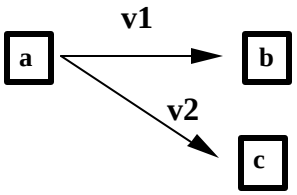


On obtient donc une contradiction (un chat est un équidé et un chat n'est pas un équidé). Cette contradiction provient de la relation ajoutée : carnivore sorte_de équidé. (Pourquoi ?) Nous en concluons que notre hypothèse était fausse, c'est-à-dire qu'il est faux qu'un carnivore est un équidé.

2.3.12 Représentation de relations alternatives

On vient de voir que pour prouver qu'une relation **v1** entre deux objets **a** et **b** est fausse, on peut procéder par réfutation en supposant que cette relation existe et en ajoutant un réseau auxiliaire approprié. On démontre ensuite que la relation **v1** entraîne une contradiction.

Par ailleurs, pour affirmer qu'un objet **a** est en relation **v1** avec un objet **b** et en relation **v2** avec un objet **c** on ajoute au réseau les arcs :

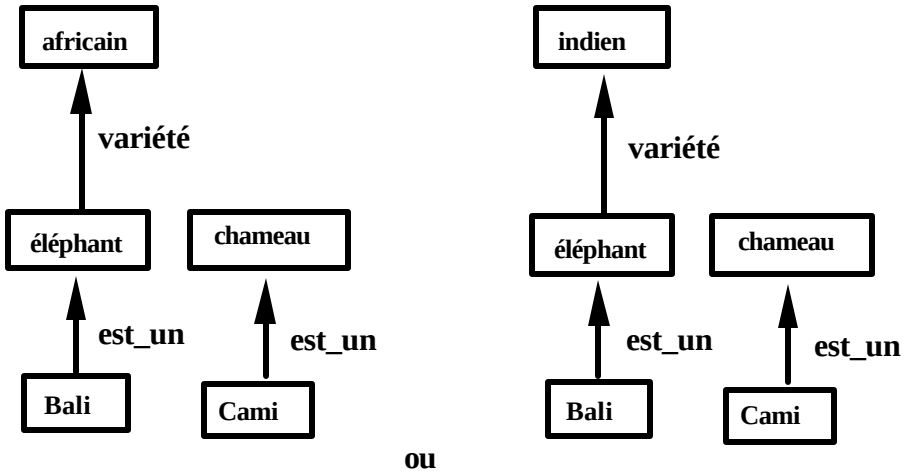


De façon générale, on peut exprimer le fait qu'un objet **a** est en relation **v1** avec un objet **b** ou en relation **v2** avec un objet **c** en produisant deux copies du réseau et en augmentant chaque copie de l'une des deux relations. Ceci signifie que toute opération sur le réseau sera faite sur chacune des copies et sera considérée complétée si elle l'est sur au moins l'une des copies.

Par exemple, les connaissances de l'énoncé :

Bali est un éléphant et Cami est un chameau.
Il y a deux variétés d'éléphants : africains et indiens

peuvent être représentées au moyen des réseaux alternatifs suivants :



Chacun de ces réseaux se dédoublerait à son tour si on ajoutait au texte la phrase : *un chameau peut avoir une ou deux bosses*.

Le besoin d'exprimer des alternatives est important dans la représentation des connaissances. Les dédoublements de réseau peuvent alourdir la mécanique de déduction déjà exposée.

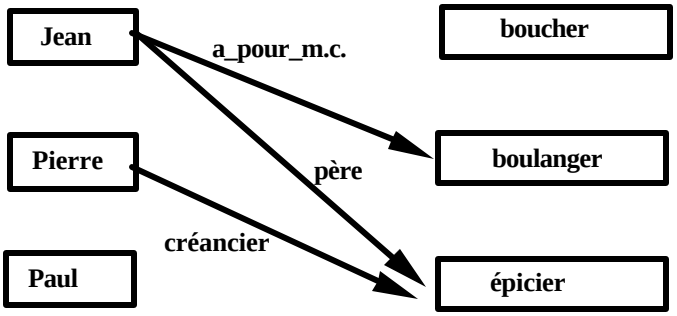
Exemple : Le problème des métiers

> *Jean, Pierre et Paul exercent chacun un métier différent. Ces métiers sont : boucher, boulanger et épicier. De plus, on sait que le boulanger est le meilleur client de Jean, que l'épicier doit $10 à Pierre et que Jean est le père de l'épicier. On demande d'associer correctement les métiers aux personnes.*

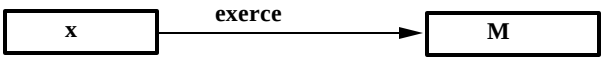
Dans ce texte, on identifie 6 objets :

3 personnes : Jean, Pierre, Paul,
3 métiers : boucher, boulanger, épicier.

Trois relations sont établies entre des personnes et des métiers (ou plus exactement entre des personnes identifiées et des personnes exerçant un métier). Ceci nous conduit au réseau R suivant :

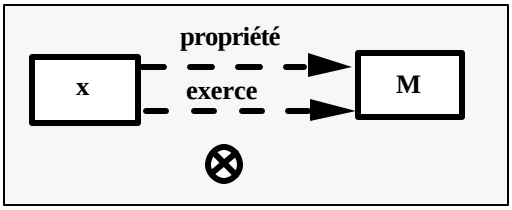


On doit ensuite se demander de quelle façon formuler la solution, soit avec l'arc général :



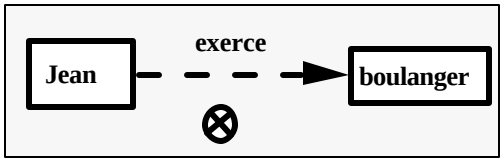
où x est une personne et M est un métier. Pour déduire ces relations, on va devoir construire quelques réseaux auxiliaires.

Notons d'abord que lorsqu'une personne est en relation avec un métier, elle ne peut exercer ce métier. Ceci nous donne des réseaux auxiliaires de la forme A1 suivante :

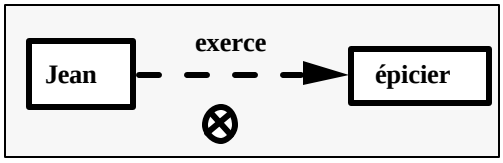


Ici propriété peut être l'une des relations : créancier, père, a_pour_m.c.

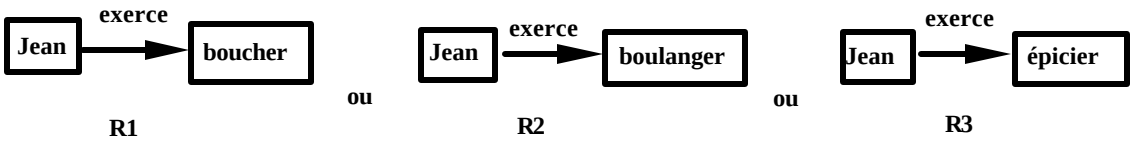
Appliquons A1 à R avec l'instanciation propriété = a_pour_m.c. Nous obtenons le réseau auxiliaire A2 :



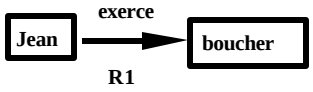
Appliquons A1 à R avec l'instanciation propriété = père. Nous obtenons le réseau auxiliaire A3 :



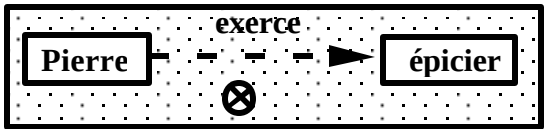
Par ailleurs, on sait que chaque personne exerce au moins l'un des trois métiers. On a donc trois alternatives pour Jean :



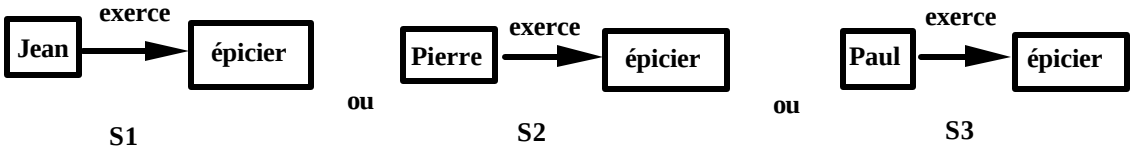
Si on applique A2 à R2 alors on obtient falsum, de même qu'en appliquant A3 à R3. La seule alternative possible est donc R1. On conclut que : Jean est boucher.



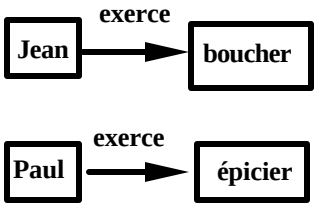
Appliquons A1 à R avec l'instanciation propriété = créancier. Nous obtenons le réseau auxiliaire A4 :



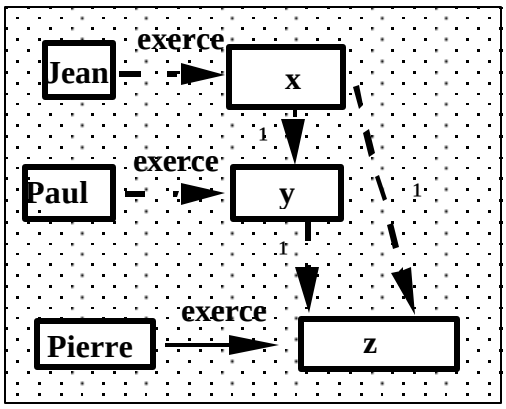
Par ailleurs, on sait que chaque métier est exercé par au moins l'une des trois personnes. On a donc trois alternatives pour le métier d'épicier :



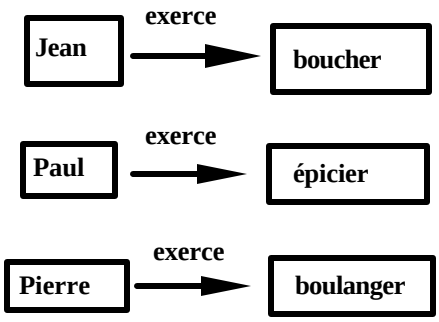
Si on applique A3 à S1 alors on obtient falsum, de même qu'en appliquant A4 à S2. La seule alternative possible est donc S3. On conclut que : Paul est épicier.



Il nous reste maintenant à établir, par le réseau auxiliaire A5, que lorsque deux des couples personne-métier sont construits le troisième est automatiquement déterminé. Cela revient à affirmer que chaque personne exerce un et un seul métier :



Dans A5, on instancie x = boucher, y = épicier, z = boulanger, et on l'applique au réseau S pour obtenir :



2.3.13 Limites des réseaux sémantiques

Étant donné un certain monde, un système formel est intéressant dans la mesure où il permet :

- la représentation des connaissances de ce monde ;
- la déduction de nouvelles connaissances dans ce monde.

On a senti, du moins partiellement, les limites des réseaux sémantiques quant à leurs possibilités pour la représentation des connaissances :

- les objets représentables sont primitifs ;
- les relations représentables sont au plus binaires ;
- les mécanismes de déduction peuvent être lourds à mettre en œuvre, en raison des difficultés de manipulation de la négation et des alternatives.

Cependant, pour les problèmes de classification (**taxonomie**), ils sont proches du langage courant et relativement faciles à mettre en œuvre sur ordinateur au moyen d'un langage déclaratif comme **Prolog** (si l'on se tient loin des alternatives).

On va maintenant aborder des outils de représentation des connaissances plus puissants mais aussi nettement plus complexes à mettre en œuvre sur ordinateur.

2.4 LE CALCUL DES PRÉDICATS

Le calcul des prédicats est un langage formel pour la représentation des connaissances. L'un des avantages du calcul des prédicats est sa capacité d'exprimer des relations complexes. Il permet d'utiliser une vaste gamme de symboles pour représenter des faits relevant de la théorie des ensembles, de l'arithmétique, du calcul relationnel et de la logique mathématique. On va donc bien au-delà des capacités d'expression des réseaux sémantiques, mais évidemment au prix d'un symbolisme beaucoup plus lourd. Un autre avantage du calcul des prédicats est sa puissance pour formaliser le raisonnement. Dans la section sur les règles d'inférence, on aura l'occasion de se familiariser avec toute une gamme d'outils pour la déduction.

2.4.1 Définition formelle du calcul des prédicats

Toutes les **phrases**, dans le calcul des prédicats, sont des suites finies de symboles disposés suivant des règles précises.

Pour décrire les structures de phrases autorisées, on aura recours à une BNF. Les symboles ::=, |, {, } jouent leur rôle habituel. Les non-terminaux sont en caractères gras. La structure des nombres et des identificateurs n'est pas précisée. On assumera simplement qu'on dispose de moyens pour distinguer les identificateurs de constantes, variables, fonctions et prédicats. La BNF suivante permet de définir formellement la notion de phrase dans le calcul des prédicats :

phrase ::= phrase_atomique | phrase_logique | phrase_quantifiée
phrase_atomique ::= nom_de_prédicat {(liste_de_termes)} | (terme opérateur _de_prédicat terme)
phrase_logique ::= ($\neg$ phrase) | (phrase $\vee$ phrase) | (phrase $\wedge$ phrase)
phrase_quantifiée ::= ($\forall$x: [phrase]) | ($\exists$x: [phrase])
liste_de_termes ::= terme | terme, liste_de_termes
terme ::= constante | variable | expression fonctionnelle
constante ::= nombre | nom_de_constante
variable ::= nom_de_variable
expression fonctionnelle ::= nom_de_fonction {(liste_de_termes)} | (terme opérateur_de_fonction terme) | (opérateur_de_fonction terme)
opérateur_de_fonction ::= + | - | * | / | $\cap$ | $\cup$ | C (complément)
opérateur_de_prédicat ::= = | $\neq$ | < | > | $\leq$ | $\geq$ | $\subseteq$ | $\supseteq$ | $\subset$ | $\supset$ | $\in$

De plus, les opérateurs logiques $\rightarrow$ et $\leftrightarrow$ seront employés, avec la signification suivante :

(terme$_1$ $\rightarrow$ terme$_2$) pour (($\neg$ terme$_1$) $\vee$ terme$_2$)

(terme$_1$ $\leftrightarrow$ terme$_2$) pour (terme$_1$ $\rightarrow$ terme$_2$) $\wedge$ (terme$_2$ $\rightarrow$ terme$_1$)

Dans la phrase $\forall x$: [Φ], Φ est appelée **portée** du quantificateur $\forall x$. De même, dans la phrase $\exists x$: [Φ], Φ est appelée **portée** du quantificateur $\exists x$.

On autorisera dans l'écriture des termes, expressions fonctionnelles, phrases, etc., pour les seules fins d'une lecture facilitée, une utilisation restreinte des parenthèses. Ainsi ((3+4)+5) pourrait éventuellement s'écrire 3+4+5. Cependant, afin d'éviter toute ambiguïté d'interprétation, on respectera les priorités décroissantes suivantes entre les opérateurs :

- unaire, C (complément)
- *
- /, $\cap$
- + , - dyadique, $\cup$
- =, $\neq$, <, >, $\leq$, $\geq$, $\subseteq$, $\supseteq$, $\subset$, $\supset$, $\in$
- $\neg$
- $\wedge$
- $\vee$
- $\rightarrow$
- $\leftrightarrow$
- $\forall$ et $\exists$

Exemples de termes :

age(capitaine)
log(2,64)
somme(un, deux)
somme(un,(2+trois))
((- b) + racine((b * b) - (4 * a * c))) / (2 * a)

Exemples de phrases :

lions $\subset$ carnivores $\wedge$ $\neg$ chats $\subset$ équidés
est_un(Grisou, chat) $\wedge$ sorte_de(chat, félin) $\rightarrow$ sorte_de(Grisou, félin)
$\forall x$: [est_un(x, chat) $\vee$ est_un(x, équidé) $\rightarrow$ sorte_de(x, mammifère)]
$\exists x$: [carnivore(x)] $\rightarrow$ $\forall x$: $\exists y$: [alimentation(x, y)]
$\forall s$: $\forall t$: $\forall x$: [$x \in s \wedge x \in t \leftrightarrow x \in s \cap t$]

2.4.2 Interprétation

Dans la section sur les réseaux sémantiques, nous avons volontairement confondu le monde « réel » et sa représentation sous forme de réseau. Les objets du monde ainsi que les nœuds du réseau, les relations dans le monde ainsi que les arcs dans le réseau, les conditionnelles dans le monde ainsi que les réseaux auxiliaires ont été respectivement identifiés. Ce qui était vrai dans le monde était identifié à ce qu'on dessinait dans le réseau.

Cependant, on construit plus généralement un système formel parce qu'il est plus facile à manipuler que le monde réel. Par ailleurs, un même système formel peut permettre d'appréhender des mondes différents. Par exemple, les mécanismes d'héritages peuvent se faire de la même façon pour n'importe quelle classification.

Au moyen de correspondances entre les entités (objets, relations, etc.) d'un système formel et les entités d'un monde, on peut faire en sorte que ce monde (ou plus précisément une certaine conceptualisation de ce monde) soit une **interprétation** du système formel. Ainsi, un système formel peut donner lieu à plusieurs interprétations dans le même monde ou dans des mondes différents.

Définitions : Un système formel est défini au moyen d'un certain nombre de symboles dont l'agencement, en respectant des règles précises de construction, permet d'obtenir des **phrases**. Lors d'une **interprétation** dans un certain monde, une phrase peut être mise en correspondance avec un fait reconnu vrai dans ce monde. On dit alors que la phrase est **satisfaite** par l'interprétation et que celle-ci est un **modèle** pour la phrase. Une phrase est dite **valide** si toute interprétation satisfait cette phrase. Un ensemble de phrases est dit **inconsistant** s'il n'existe aucune interprétation qui rende toutes ces phrases vraies en même temps.

Il est donc intéressant de pouvoir identifier les phrases valides dans un système formel. Malheureusement, dans le calcul des prédicats, ceci n'est pas possible, comme le précise le théorème d'indécidabilité de Church (1936) :

> *Il n'existe pas d'algorithme général qui permette de décider, en un temps fini, de la validité ou de la non-validité de n'importe quelle phrase de la logique des prédicats du premier ordre. On dit que celle-ci est indécidable .*

Comme la situation est la même dans le monde des réseaux sémantiques, on trouvera une issue satisfaisante en s'inspirant de la technique de **réfutation** utilisée dans les preuves. Nous reviendrons ultérieurement sur cette notion.

2.4.3 Procédures d'inférence

Dans tout système formel, certaines phrases sont désignées comme étant vraies. On les appelle des **axiomes**. Par exemple, tous les faits d'une base de connaissances sont des axiomes. L'**inférence** est le processus permettant de **dériver** des conclusions à partir d'hypothèses. Une **règle d'inférence** comprend donc un ensemble de prémisses (ou hypothèses) et un ensemble de conclusions. Une règle d'inférence peut être notée de deux façons :

$$\textit{prémisses} \vdash \textit{conclusions} \qquad \text{OU} \qquad \frac{\textit{prémisses}}{\textit{conclusions}}$$

Les phrases obtenues des axiomes par application des règles d'inférence s'appellent des **théorèmes.** La suite d'applications des règles d'inférence qui mène des axiomes au théorème constitue alors la **preuve du théorème**.

Le calcul des prédicats donne lieu à beaucoup de règles d'inférence. Nous en énumérerons les principales dans cette section. Elles sont en général applicables sans contrainte. Cependant, celles concernant les expressions quantifiées demandent quelques précautions. En effet, on doit vérifier si les variables sont libres. Soit Φ une phrase quantifiée contenant la variable **v** et soit Γ un terme quelconque. On dit ainsi que Γ est **libre** pour **v** dans Φ si **v** n'est dans la portée d'aucune variable de Γ.

Par exemple, considérons la phrase (que l'on peut considérer vraie) *il existe plus jeune que soi* :

$\Phi = \exists u$: [plus_jeune(u, v)]. (*Il existe un u plus jeune que v*).

Dans cette phrase, v est libre et u est liée. Si on remplace la variable v par le terme Γ = fils(u), on obtient alors la phrase :

$$\Phi_{v\,/\,fils(u)} = \exists\ u\text{: [plus\_jeune(u, fils(u))]. (v / fils(u) : on lit « v remplacé par fils(u))}$$

Cette phrase est évidemment fausse car cela voudrait dire qu'il existe un u plus jeune que le fils de u. On n'a donc pas le droit de remplacer v par fils(u). Ceci s'explique par le fait que Γ n'est pas libre pour v dans Φ. En effet, dans Φ, v est dans la portée de u qui est une variable de Γ.

Voyons maintenant quelques inférences importantes, qui comprennent toutes un certain nombre de prémisses et une seule conclusion :

Nom de la règle d'inférence	*Description*
Modus Ponens (MP)	$\Phi \rightarrow \Psi, \Phi \vdash \Psi$
Modus Tollens (MT)	$\Phi \rightarrow \Psi, \neg\Psi \vdash \neg \Phi$
Transitivité (T)	$\Phi \rightarrow \Psi, \Psi \rightarrow \Gamma \vdash \Phi \rightarrow \Gamma$
Simplification (S)	$\Phi \wedge \Psi \vdash \Phi$
Conjonction (C)	$\Phi, \Psi \vdash \Phi \wedge \Psi$
Spécification universelle (SU)	$\forall x{:}[\Phi] \vdash \Phi_{x/y}$ Si y est libre pour x dans Φ et Φ x/y est la phrase obtenue de Φ en remplaçant toutes les occurrences de x par y.
Spécification existentielle (SE)	$\Psi = \exists x{:}\,[\Phi] \vdash \Phi_{x/\pi(x_1, x_2, \ldots, x_n)}$ π est un nouveau nom de fonction et x_1, x_2, … et x_n sont les variables libres dans Ψ et $\pi(x_1, x_2, \ldots, x_n)$ est une expression fonctionnelle dépendant des variables x_1, x_2, … et x_n.
	$\exists x{:}\ \Phi \vdash \Phi_{x/a}$ a est une nouvelle constante.

Exemples : Questionnons-nous sur quelques inférences. On verra que certaines sont non valides parce qu'on passe outre les contraintes de substitutions indiquées dans les règles de spécification.

1) *Si Pierre aime tout le monde alors Pierre aime Jeanne.*

$$\forall x\text{: [aime(Pierre, x)]} \vdash \text{aime(Pierre, Jeanne)}$$

valide : la variable x a été remplacée par la constante Jeanne.

2) *Si Pierre aime tout le monde alors Pierre aime z.*

$$\forall x\text{: [aime(Pierre, x)]} \vdash \text{aime(Pierre, z)}$$

valide : la variable x a été remplacée par la variable z non présente dans la phrase quantifiée.

3) *Si tout le monde aime quelqu'un alors Pierre aime quelqu'un.*

$$\forall x\text{: } \exists z\text{ : [aime(x, z)]} \vdash \exists z\text{: [aime(Pierre, z)]}$$

valide : la variable x a été remplacée par la constante Pierre.

4) *Si tout le monde aime quelqu'un alors quelqu'un est aimé par son père.*

$\forall x: \exists z: [aime(x, z)] \not\models \exists z: [aime(père(z), z)]$

non valide : l'inférence n'est pas valide car père(z) n'est pas libre pour x.

5) *Si x et y sont parents de z alors x et y sont parents de leur enfant.*

$\exists z: [parents(x, y, z)] \vdash parents(x, y, enfant(x, y))$

valide : il faut supposer que enfant/2 est une nouvelle fonction.

6) *Si x £ y alors y+1 £ y.*

$\exists x: \exists y: [x \leq y] \not\models \exists y: [y+1 \leq y]$

non valide : car y + 1 n'est pas libre pour x.

7) *S'il existe quelqu'un à qui tout le monde obéit alors tout le monde obéit à Jacques.*

$\exists y: \forall x:$ obéit(x, y) $\vdash \forall x:$ obéit(x, Jacques)

valide : il faut supposer que Jacques est une nouvelle constante.

Lors de la suppression de quantificateurs, les contraintes sur les substitutions ne permettent que d'affirmer la validité de l'inférence. Le non-respect des contraintes ne permet pas d'affirmer la non-validité de l'inférence, comme dans l'exemple suivant :

Si tout le monde aime tout le monde alors tout le monde est aimé par son père.

$\forall x: \forall z:$ [aime(x, z)] $\vdash \forall z:$ [aime(père(z), z)]

valide : père(z) n'est pas libre pour x et cependant l'inférence est valide. Cette validité tient au fait que les deux variables sont quantifiées par l'universel ($\forall$).

Par contre :

$\exists x: \forall z:$ [aime(x, z)] $\not\models \forall z:$ [aime(père(z), z)] et $\forall x: \exists z:$ [aime(x, z)] $\not\models \exists z:$ [aime(père(z), z)]

Comment s'articulerait une preuve formelle utilisant les inférences présentées précédemment ? Étant donné un ensemble de règles d'inférence, une phrase Φ est dite **dérivable** d'un ensemble de **prémisses** Δ si et seulement si :

- Φ appartient à Δ ;
- Φ est obtenue par application d'une règle d'inférence à une phrase **dérivable** de Δ.

Cette définition récurrente permet donc la construction successive de phrases au moyen des règles d'inférence. Illustrons par un exemple :

Les constantes sont : Jeannot et Sauteur

Les prédicats sont : cheval, chien, lévrier, lapin (unaires) et plus_vite (binaire)

Δ, l'ensemble des règles est :

Δ_1 : $\forall x$: $\forall y$: [cheval(x) $\wedge$ chien(y) $\rightarrow$ plus_vite(x, y)]

Δ_2 : $\exists y$: [lévrier(y) $\wedge$ $\forall z$: [lapin(z) $\rightarrow$ plus_vite(y, z)]]

Δ_3 : $\forall y$: [lévrier(y) $\rightarrow$ chien(y)]

Δ_4 : $\forall x$: $\forall y$: $\forall z$: [plus_vite(x, y) $\wedge$ plus_vite(y, z) $\rightarrow$ plus_vite(x, z)]

$\Delta 5$: cheval(Sauteur)

$\Delta 6$: lapin(Jeannot)

On veut démontrer que : plus_vite(Sauteur, Jeannot). Pour cela, on dispose des règles d'inférence : MP(modus ponens), T(transitivité), SE(spécification existentielle), SU(spécification universelle), C(conjonction) et S(simplification).

#	*Phrase (prémisse ou déduction)*	*Justification*
1	cheval(Sauteur)	$\Delta 5$
2	lapin(Jeannot)	$\Delta 6$
3	$\forall x$: $\forall y$: [cheval(x) $\wedge$ chien(y) $\rightarrow$ plus_vite(x, y)]	$\Delta 1$
4	$\forall y$: [cheval(Sauteur) $\wedge$ chien(y) $\rightarrow$ plus_vite(Sauteur, y)]	3, SU {x/Sauteur}
5	$\exists y$: [lévrier(y) $\wedge$ [$\forall z$: lapin(z) $\rightarrow$ plus_vite(y, z)]]	$\Delta 2$
6	lévrier(Élan) $\wedge$ ($\forall z$: lapin(z) $\rightarrow$ plus_vite(Élan, z))	5, SE {y/Élan}
7	lévrier(Élan)	6, S
8	$\forall y$: [lévrier(y) $\rightarrow$ chien(y)]	$\Delta 3$
9	lévrier(Élan) $\rightarrow$ chien(Élan)	8, SU {y/Élan}
10	chien(Élan)	7, 9, MP
11	cheval(Sauteur) $\wedge$ chien(Élan) $\rightarrow$ plus_vite(Sauteur, Élan)	4, SU {y/Élan}
12	cheval(Sauteur) $\wedge$ chien(Élan)	1, 10, C
13	plus_vite(Sauteur, Élan)	11, 12, MP
14	$\forall z$: [lapin(z) $\rightarrow$ plus_vite(Élan, z)]	6, S
15	lapin(Jeannot) $\rightarrow$ plus_vite(Élan, Jeannot)	14, SU {z/Jeannot}
16	plus_vite(Élan, Jeannot)	2, 15, MP
17	$\forall x$: $\forall y$: $\forall z$: [plus_vite(x, y) $\wedge$ plus_vite(y, z) $\rightarrow$ plus_vite(x, z)]	$\Delta 4$
18	plus_vite(Sauteur, Élan) $\wedge$ plus_vite(Élan, Jeannot) $\rightarrow$ plus_vite(Sauteur, Jeannot)	17, SU {x/Sauteur, y/Élan, z/Jeannot}
19	plus_vite(Sauteur, Élan) $\wedge$ plus_vite(Élan, Jeannot)	13, 16, C

20	plus_vite(Sauteur, Jeannot)	18, 19, MP

N.B. : Pour les besoins de la preuve, la constante Élan a été créée.

Dans cet exemple, on a vu comment déduire une certaine phrase à partir de prémisses, en utilisant des règles d'inférence. Il n'est pas dit comment on doit procéder à chaque étape pour produire une nouvelle phrase ni comment on peut s'assurer du progrès vers la phrase cherchée.

Un mécanisme qui permettrait, à partir de n'importe quel groupe de phrases, de construire successivement de nouvelles phrases par application de règles d'inférence s'appelle une **procédure d'inférence**. Si on suit de près une preuve, on peut considérer en fait qu'à chaque ligne du raisonnement on ajoute à nos connaissances une nouvelle phrase. On parle alors procédure d'inférence de type **monotone**. Une procédure de type **non monotone** aurait la caractéristique de permettre, à chaque étape, l'ajout mais aussi le **retrait** de phrases.

Ayant toujours le souci d'une mise en œuvre efficace de mécanismes de déduction, on cherche à réduire le nombre de règles d'inférence, tout en perdant le moins possible la capacité de démontrer ce qui est « démontrable ». Par exemple, bien que très puissante, la seule règle de Modus Ponens est insuffisante pour démontrer tout ce qui est démontrable dans le calcul des prédicats. La phrase $\Phi = \forall x: [F(x) \vee G(x)]$ est une conséquence logique de $\Phi 1 = \forall x: [F(x)]$ et $\Phi 2 = \forall x: [G(x)]$ mais n'est pas déductible de $\Phi 1$ et $\Phi 2$ par la seule utilisation de Modus Ponens.

2.4.4 Forme clausale

La **forme clausale** est une forme standard permettant de représenter les propositions de façon à pouvoir appliquer sans difficulté la règle d'inférence que l'on appellera **principe de résolution (pour la réfutation).** On verra, par la suite, que certaines formes clausales sont appropriées pour une implantation simple sur ordinateur du raisonnement logique.

On appelle **littéral** un terme atomique (littéral **positif**) ou la négation d'un terme atomique (littéral **négatif**). Par exemple, chat(léo) et $\neg$ femme(félix) sont des termes atomiques respectivement positif et négatif.

Une **clause** est une **disjonction** de littéraux. Voici trois exemples de clauses :

alimentation(carnivore, viande)
a_une_bosse(Cami) $\vee$ a_deux_bosses(Cami)
alimentation(x, viande) $\vee \neg$ sorte_de(x, carnivore).

Une **clause de Horn** est une clause dont **au plus** un littéral est positif. Son utilité se justifie facilement si on considère la représentation en clause de Horn de la conditionnelle typique :

$$p_1 \wedge p_2 \wedge \ldots p_n \rightarrow q$$

est la clause suivante : $\neg p_1 \vee p_2 \vee \ldots \vee \neg p_n \vee q$.

Nous allons voir maintenant que toute proposition logique peut être mise sous forme d'un **ensemble** de clauses. La transformation s'effectue en un certain nombre d'étapes (9 au maximum). Selon le contenu de l'expression à transformer, certaines étapes peuvent être omises.

1. Éliminer les connecteurs ® et «

Rappellons que :

$\Phi \rightarrow \Psi$ est équivalent à $\neg \Phi \vee \Psi$

$\Phi \leftrightarrow \Psi$ est équivalent à $(\neg\Phi \vee \Psi) \wedge (\neg\Psi \vee \Phi)$

2. Distribuer les Ø pour les juxtaposer aux littéraux et au besoin simplifier.

Rappelons que :

$\neg\neg\Phi$ est équivalent à Φ

$\neg (\Phi \vee \Psi)$ est équivalent à $\neg\Phi \wedge \neg\Psi$ (lois de Morgan)

$\neg (\Phi \wedge \Psi)$ est équivalent à $\neg\Phi \vee \neg\Psi$ (lois de Morgan)

$\neg\forall x: \Phi$ est équivalent à $\exists x: \neg\Phi$

$\neg\exists x: \Phi$est équivalent à $\forall x: \neg\Phi$

3. Dans le but de préparer l'élimination des quantificateurs, renommer les variables liées de façon qu'à chaque quantificateur soit associée une variable distincte des autres variables.

Par exemple : $\forall x: [P(x, x)] \wedge \exists x: [Q(x)] \vee R(x, y, z)$

devient : $\forall w: [P(w, w)] \wedge \exists t: [Q(t)] \vee R(x, y, z)$

Donc, le premier x a été substitué par w (x/w) et le deuxième x a été substitué par t (x/t).

4. Préfixer la phrase de tous ses quantificateurs, sans en changer l'ordre.

Par exemple : $\forall x: [\exists y: [P(x, y)] \wedge \forall z: [Q(x, z)]]$

devient : $\forall x: \exists y: \forall z: [P(x, y) \wedge Q(x, z)]$

5. Éliminer les quantificateurs existentiels.

Deux cas se présentent :

5.1 Si un quantificateur existentiel ne se trouve pas dans la portée d'un universel, enlever l'existentiel et remplacer toutes les occurrences de sa variable par une nouvelle constante dite constante de skolem.

Par exemple : $\exists x:\forall y: [P(x, y) \wedge Q(x)]$

devient : $\forall y: [P(a, y) \wedge Q(a)]$ où **a** est une constante de skolem.

Notons que l'on ne peut substituer **x** par une constante de skolem dans l'expression suivante $\forall y: \exists x: [P(x, y) \wedge Q(x)]$ (Pourquoi ?).

5.2 Si un quantificateur existentiel se trouve dans la portée de quantificateurs universels, éliminer le quantificateur existentiel et remplacer toutes les occurrences de sa variable par une fonction nouvelle (dite de skolem) appliquée aux variables quantifiées par les quantificateurs universels.

Par exemple : $\forall x: \forall y: \exists z: [P(x, y, z)]$

devient : $\forall x: \forall y: [P(x, y, F(x, y))]$ où F est une nouvelle fonction.

Par exemple : $\forall z: \exists x: \forall y: [Q(x, y, z)]$

devient : $\forall z: \forall y: [Q(G(z), y, z)]$ où G est une nouvelle fonction.

6. Éliminer les quantificateurs universels.

Par exemple : $\forall x: \forall y: [P(x, y)]$

devient : P(x, y).

7. Mettre la phrase sous forme conjonctive.

Par exemple : $\Phi \vee (\Psi \wedge \omega)$
devient : $(\Phi \vee \Psi) \wedge (\Phi \vee \omega)$, soit un ensemble de deux clauses : $\{\Phi \vee \Psi, \Phi \vee \omega\}$

8. Transformer en clause chaque facteur de la forme conjonctive.
Par exemple : $(\Phi \vee \Psi) \wedge \omega$
devient : $\{\Phi \vee \Psi, \omega\}$, un ensemble de deux clauses:

9. Renommer de façon distincte les variables d'une clause à l'autre.
Par exemple : $\{ P(x, y) \vee \neg Q(y), R(x, y) \}$
devient : $\{P(x_1,y_1) \vee \neg Q(y_1) , R(x_2, y_2) \}$
Cette dernière opération est nécessaire pour affirmer l'indépendance des deux clauses.

Revoyons maintenant ces étapes de transformation avec les deux exemples suivants.

Exemple #1 : *Quiconque est romain et connaît Marcus déteste César ou bien croit fou tout individu qui déteste au moins une personne.*

$\forall x: [(romain(x) \wedge connaît(x,Marcus)) \rightarrow (déteste(x,César) \vee \forall y:[\exists z: [déteste(y,z)] \rightarrow croitfou(x,y)])]$

1. Éliminer les connecteurs $\rightarrow$:

$\forall x: [\neg(romain(x) \wedge connaît(x, Marcus)) \vee (déteste(x, César) \vee \forall y: [\neg\exists z: [déteste(y, z)] \vee croitfou(x, y)])]$

2. Distribuer les $\neg$:

$\forall x: [\neg romain(x) \vee \neg connaît(x, Marcus) \vee déteste(x, César) \vee \forall y: [\forall z: [\neg déteste(y, z)] \vee croitfou(x, y)]]$

3. Renommer les variables liées : rien à faire.

4. Préfixer les quantificateurs :

$\forall x: \forall y: \forall z: [\neg romain(x) \vee \neg connaît(x, Marcus) \vee déteste(x, César) \vee \neg déteste(y, z) \vee croitfou(x, y)]$

5. Éliminer les quantificateurs existentiels : aucun.

6. Éliminer les quantificateurs universels :

$\neg romain(x) \vee \neg connaît(x, Marcus) \vee déteste(x, César) \vee \neg déteste(y, z) \vee croitfou(x, y)$

7. Mettre la phrase sous forme conjonctive : rien à faire.

8. Transformer chaque facteur en clause (ici, une seule clause) :

$\neg romain(x) \vee \neg connaît(x, Marcus) \vee déteste(x, César) \vee \neg déteste(y, z) \vee croitfou(x, y)$

Solution : $\{\neg romain(x) \vee \neg connaît(x,Marcus) \vee déteste(x,César) \vee \neg déteste(y,z) \vee croitfou(x,y)\}$

Remarque : ce n'est pas une clause de Horn puisqu'elle contient deux littéraux positifs.

Exemple #2 : $\forall x: [\forall y: [P(x, y)] \rightarrow \neg\forall y: [Q(x, y) \rightarrow R(x, y)]]$

1. Éliminer les connecteurs $\rightarrow$:

$\forall x: [\neg\forall y: [P(x, y)] \vee \neg\forall y: [\neg Q(x, y) \vee R(x, y)]]$

2. Distribuer les $\neg$:

$\forall x: [\exists y: [\neg P(x, y)] \vee \exists y: [Q(x, y) \wedge \neg R(x, y)]]$

3. Renommer les variables liées :

$\forall x: [\exists y: [\neg P(x, y)] \vee \exists z: [Q(x, z) \wedge \neg R(x, z)]]$

4. Préfixer les quantificateurs :

$\forall x: \exists y: \exists z: [\neg P(x, y) \vee (Q(x, z) \wedge \neg R(x, z))]$

5. Éliminer les quantificateurs existentiels :

d'abord y :

$\forall x: \exists z: [\neg P(x, F_1(x)) \vee (Q(x, z) \wedge \neg R(x, z))]$ (F_1 fonction de skolem, on pose y / $F_1(x)$)

puis z :

$\forall x: [\neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x))]$ (F_2 fonction de skolem, on pose z / $F_2(x)$)

6. Éliminer le quantificateur universel :

$\neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x))$

7. Mettre la phrase sous forme conjonctive :

$(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge (\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))$

8. Transformer chaque facteur en clause (ici, deux clauses) :

$\{ \neg P(x, F_1(x)) \vee Q(x, F_2(x)), \neg P(x, F_1(x)) \vee \neg R(x, F_2(x)) \}$

9. Renommer de façon distincte les variables, d'une clause à l'autre :

$\{ \neg P(x_1,F_1(x_1)) \vee Q(x_1, F_2(x_1)), \neg P(x_2,F_1(x_2)) \vee \neg R(x_2, F_2(x_2)) \}$

Solution : $\{ \neg P(x_1,F_1(x_1)) \vee Q(x_1, F_2(x_1)), \neg P(x_2,F_1(x_2)) \vee \neg R(x_2, F_2(x_2)) \}$

2.4.5 Le principe de résolution

Nous allons voir quel mécanisme utiliser pour faire de la déduction à partir de la forme clausale. Pour cela, rappelons la règle de transitivité du calcul propositionnel : $p \rightarrow q , q \rightarrow r \vdash p \rightarrow r$

qui s'écrit sous forme clausale : $\{\neg p \vee q , \neg q \vee r\} \vdash \{\neg p \vee r\}$

On constate donc que l'une des clauses prémisses contient le littéral positif q tandis que l'autre contient le littéral négatif $\neg q$. Ces deux clauses, dites complémentaires, permettent de déduire une clause réunissant tous les littéraux des prémisses sauf q et $\neg q$.

Définition : Si A et B sont deux clauses complémentaires, c'est-à-dire contenant respectivement les littéraux Φ et $\neg\Phi$ alors on peut déduire la nouvelle clause C, dite résolvant, obtenue en réunissant tous les littéraux de A et B sauf Φ et $\neg\Phi$. Ce mécanisme s'appelle une résolution binaire entre deux clauses.

Le principe de résolution permet de combiner ce mécanisme de construction de nouvelles clauses à la technique de réfutation pour faire de la déduction. Dans le calcul propositionnel, le principe de résolution s'applique ainsi :

Soit Δ un ensemble de clauses et soit Φ une proposition à démontrer.

1. Mettre $\neg\Phi$ sous forme clausale et l'ajouter à Δ.
2. Répéter jusqu'à obtention de la clause vide ou jusqu'à ce qu'aucun progrès ne soit perceptible :

   Choisir deux clauses complémentaires A et B et construire leur clause résolvante C :
   - si C est la clause vide alors $\Delta \vdash \Phi$,
   - si C n'est pas vide alors ajouter C à Δ.

Exemple : $\Delta = \{ p , \neg p \vee q , \neg q \vee r \}$ et $\Phi = r$. on veut donc démontrer Φ.

PREUVE : Appelons respectivement Δ_1, Δ_2 et Δ_3 les trois clauses contenues dans Δ ($\Delta_1 = p$, $\Delta_2 = \neg p \vee q$, Δ_3 $\neg q \vee r$). Appliquons maintenant le principe de résolution.

	Affirmations	Justifications
1.	p	Δ_1
2.	$\neg p \vee q$	Δ_2
3.	$\neg q \vee r$	Δ_3
4.	$\neg r$	$\neg\Phi$
5..	$\neg q$	résolution binaire de 3 et 4
6.	$\neg p$	résolution binaire de 2 et 5
7.	{ }	résolution binaire de 1 et 6

Ayant obtenu la clause vide, cela signifie que nous avons généré une contradiction, d'où nous ne pouvions affirmer $\neg\Phi$. Par conséquent $\Delta \vdash \Phi$, c'est-à-dire que la clause r est déductible de $\Delta = \{ p, \neg p \vee q, \neg q \vee r\}$.

Le principe de résolution peut se généraliser pour le calcul des prédicats. Dans ce cas, il faut tenir compte des variables qui sont susceptibles de s'unifier avec des constantes.

Exemple : Soit $\Delta = \{ P(x) \vee Q(x, y), \neg P(a) \vee R(b, z) \}$ et $\Phi = Q(a, w) \vee R(b, t)$. Prouver Φ par résolution.

PREUVE :

Posons : $\Delta_1 = P(x) \vee Q(x, y)$ et $\Delta_2 = \neg P(a) \vee R(b, z)$.

$\neg\Phi = \{\neg Q(a, w), \neg R(b, t) \}$. Nommons $\Phi_1 = \neg Q(a, w)$ et $\Phi_2 = \neg R(b, t)$.

	Affirmations	Justifications

1. $P(x) \vee Q(x, y)$	$\Delta 1$
2. $\neg Q(a, w)$	Φ_1
3. P(a)	Par résolution binaire de 1 et 2 et la substitution { x/a, y/w }.
4. $\neg P(a) \vee R(b, z)$	Δ_2
5. R(b, z)	Par résolution binaire de 3 et 4.
6. $\neg R(b, t)$	Φ_2
7. { }	Par résolution binaire de 5 et 6 et la substitution { z/t }.

Ayant obtenu la clause vide, cela signifie que nous avons généré une contradiction, d'où nous ne pouvions affirmer $\neg\Phi$. On conclut donc que Δ |- Φ, c'est-à-dire que Φ peut être déduite de Δ. La clause $Q(a, y) \vee R(b, z)$ est donc déductible de $\Delta = \{P(x) \vee Q(x, y), \neg P(a) \vee R(b, z)\}$.

Remarque : On aura noté à la ligne 3 le remplacement de la variable x par la constante a. De façon générale, on appelle **substitution** une suite finie d'associations entre des variables et des expressions :

{ substitué = variable / substituant = expression} où :

- chaque variable est associée à au plus une expression ;
- aucune variable associée à une expression n'apparaît dans une des autres expressions.

Par exemple :

{x/a, y/F(b), z/w} est une substitution,

{x/G(y), y/F(x)} n'est pas une substitution car x apparaît dans l'expression substituante de y.

Définition : Si Φ est une phrase et **s** est une substitution s'appliquant à Φ, on note Φ**s** la phrase résultante.

Par exemple, si $\Phi = P(x, x, y, z)$ et s = {x/a, y/F(b), z/w} alors $P(x, x, y, z)s = P(a, a, F(b), w)$.

On peut appliquer deux substitutions consécutives à une phrase. On parle alors de substitution composée.

Par exemple, soient s_1 = {w/G(x, y)} et s_2 = {x/a, y/b, z/c} alors s_1s_2 = {w/G(a, b), x/a, y/b, z/c } est une substitution composée. On obtient donc s_1s_2 en appliquant les substitutions de s_2 dans les substituants de s_1 et en ajoutant à s_1 les substitutions de s_2.

2.4.6 Unification de littéraux

L'unification, déjà étudiée dans le cadre des réseaux sémantiques, sera l'outil permettant l'application du principe de résolution, dans le calcul des prédicats. Un ensemble de littéraux $\{\Phi_1, \Phi_2, ..., \Phi_n\}$ est dit **unifiable** s'il existe une substitution **s** telle que :

$$\Phi_1\mathbf{s} = \Phi_2\mathbf{s} = ... = \Phi_n\mathbf{s}$$

s est appelée substitution unificatrice (ou simplement **unificateur**) de $\{\Phi_1, \Phi_2, ..., \Phi_n\}$.

Exemple :

Soient les littéraux $\Phi_1 = P(a, y, z)$ et $\Phi_2 = P(x, b, z)$.

Soient les substitutions : $s_1 = \{x/a, y/b, z/c\}$ et $s_2 = \{x/a, y/b\}$.

En appliquant les substitutions, on obtient :

$\Phi_1 s_1 = \Phi_2 s_1 = P(a, b, c)$

$\Phi_1 s_2 = \Phi_2 s_2 = P(a, b, z)$

s_1 et s_2 sont deux unificateurs de Φ_1 et Φ_2. Cependant, s_2 est plus générale que s_1 car il existe une substitution s telle que $s_1 = s_2 s$ ($s = \{z/c\}$).

Quand il existe, l'unificateur **le plus général** (UPG) est unique, aux noms des variables près. Soit une clause A, considérons un sous-ensemble B de littéraux de A pour lequel existe **s**, l'unificateur le plus général. A' = A**s** est alors appelé **facteur** de A.

Mais comment définir **l'unification de deux clauses** dans le calcul des prédicats ? Considérons les deux suivants.

Soient $A = P(x) \vee P(F(y)) \vee R(x, y)$ et $B = \{P(x), P(F(y))\}$

$s = \{x/F(y)\}$ est l'UPG des littéraux de B.

$A' = As = P(F(y)) \vee R(F(y),y)$.

Si A et B sont deux clauses et A' est un facteur de A contenant le littéral Φ et B' est un facteur de B contenant le littéral $\neg\Psi$ et si **s** est l'unificateur le plus général de Φ et de Ψ alors on définit **la clause unificatrice** ω par le **résolvant** de A**s** et de B**s**.

Si $A' = A = \neg I(x) \vee F(b, y)$ et $B' = B = \neg F(x, a) \vee K(x, y)$

alors $\Phi = F(b, y)$ et $\Psi = F(x, a)$

s = {x/b, y/a} est donc l'unificateur le plus général de Φ et Ψ

avec $\omega = \neg I(b) \vee K(b, a)$

Maintenant, nous sommes prêts à généraliser le principe de résolution dans le cadre du calcul des prédicats, ce qui nous donne :

Soient Δ un ensemble de clauses et ω la proposition à démontrer.

Pour démontrer que $\Delta \vdash \omega$:

1. Mettre $\neg\omega$ sous forme clausale et l'ajouter à Δ.

2. Répéter jusqu'à l'obtention de la clause vide ou jusqu'à ce qu'aucun progrès ne soit perceptible :

Choisir deux clauses A et B qui s'unifient en C :

- Si C est vide alors $\Delta \vdash \omega$.

- Si C n'est pas vide ajouter cette clause à Δ.

2.4.7 Interrogation d'un ensemble de clauses

Considérons l'ensemble de clauses (la base de connaissances) Δ suivant :

Δ = { Père(Charles, Marie) , Parent(Claude, Julie) , ¬Père(x, y) ∨ Parent(x, y) }

On veut prouver par réfutation que Parent(Charles, Marie).

PREUVE :

Notons respectivement Δ1, Δ2 et Δ3 les clauses de Δ (de gauche à droite). Notons ω la conclusion et ajoutons sa négation à Δ, soit ¬ω = ¬Parent(Charles, Marie).

Affirmations	Justifications
1. Père(Charles, Marie)	Δ1
2. Parent(Claude, Julie)	Δ2
3. ¬Père(x, y) ∨ Parent(x, y)	Δ3
4. ¬Parent(Charles, Marie)	¬ω
5. ¬Père(Charles, Marie)	Résolution binaire de 3 et 4 avec la substitution : {x/Charles, y/Marie}
6. { }	Résolution binaire de 1 et 5

Ayant obtenu la clause vide, cela signifie que nous avons généré une contradiction, d'où nous ne pouvions affirmer ¬ω. On conclut donc que Δ |- ω, c'est-à-dire que ω peut être déduite de Δ.

La clause Parent(Charles, Marie) est donc déductible de Δ :

Δ = { Père(Charles, Marie), Parent(Claude, Julie) , ¬Père(x, y) ∨ Parent(x, y) }

On peut aussi poser la question « Qui est Parent de Marie ? » en prouvant Parent(z, Marie).

Pour cela, on commence par définir un littéral nouveau Rép d'arité n (où n est le nombre de variables dans la question à poser, ici n = 1) et on rajoute ensuite à la base de connaissances la clause ¬Parent(z, Marie) ∨ Rép(z). (Pourquoi ?)

Reprenons : ω = Parent(x, Marie), donc ¬ω = ¬Parent(x, Marie)

et ¬ω ∨ Rép(z) = ¬Parent(z, Marie) ∨ Rép(z).

1. Père(Charles, Marie)	Δ1
2. Parent(Claude, Julie)	Δ2
3. ¬Père(x, y) ∨ Parent(x, y)	Δ3
4. ¬Parent(z, Marie) ∨ Rép(z) ¬ω ∨ Rép(z)	
5. ¬Père(x, Marie) ∨ Rép(x)	Résolution binaire de 3 et 4, avec {y/Marie, z/x}

6. Rép(Charles) Résolution binaire de 1 et 5, avec {x/Charles}

Utilisons maintenant cette technique pour résoudre un problème plus complexe « le meurtre de Victor » :

> *On sait que Victor a été tué. Arthur, Bernard et Charles sont suspects. Arthur dit qu'il n'a pas tué. Il dit que Bernard était ami de la victime mais que Charles détestait la victime. Bernard dit qu'il n'était pas en ville le jour du meurtre et de plus qu'il ne connaissait pas la victime. Charles dit qu'il est innocent et qu'il a vu Arthur et Bernard avec la victime juste avant le meurtre.*

On suppose que tout le monde dit la vérité, sauf peut-être le meurtrier. Pour résoudre ce problème, on définit les prédicats : innocent/1, ami/2, déteste/2, hors_ville/1, connaît/2, avec/2.

On note (a)rthur, (b)ernard, (c)harles, (v)ictor comme constantes.

On écrit ce monde sous forme d'assertions en logique prédicative. On transforme ensuite ces assertions en clauses. On ajoute la négation de la conclusion à la base. Puis, on applique le principe de résolution par réfutation.

Certaines assertions viennent directement à l'esprit en inspectant le texte. Sachant qu'un innocent ne ment pas, on écrit les assertions suivantes :

$\Delta 1$: Innocent(a) $\rightarrow$ Ami(b, v)

$\Delta 2$: Innocent(a) $\rightarrow$ Déteste(c, v)

$\Delta 3$: Innocent(b) $\rightarrow$ Hors_ville(b)

$\Delta 4$: Innocent(b) $\rightarrow \neg$Connaît(b, v)

$\Delta 5$: Innocent(c) $\rightarrow$ Avec(a, v)

$\Delta 6$: Innocent(c) $\rightarrow$ Avec(b, v)

Les trois assertions suivantes viennent du sens commun :

- *Si on était avec Victor on ne pouvait être en dehors de la ville.*

$\Delta 7$: Avec(x, v) $\rightarrow \neg$Hors_ville(x)

- *Si on est ami de quelqu'un on le connaît.*

$\Delta 8$: Ami(x, y) $\rightarrow$ Connaît(x, y)

- *Si on déteste quelqu'un on le connaît.*

$\Delta 9$: Déteste(x, y) $\rightarrow$ Connaît(x, y)

Les trois clauses suivantes viennent de l'hypothèse du monde clos (deux des trois suspects sont innocents). On pourrait aussi dire que lorsque x est coupable alors y est innocent ($y \neq x$) :

$\Delta 10$: $\neg$Innocent(a) $\rightarrow$ Innocent(b)

$\Delta 11$: $\neg$Innocent(a) $\rightarrow$ Innocent(c)

$\Delta 12 : \neg Innocent(b) \rightarrow Innocent(c)$

La question est : « Qui a tué Victor? », ce qui revient à prouver qu'il existe quelqu'un de coupable :

But = $\exists x : \neg Innocent(x)$
$\neg But = \neg(\exists x: \neg Innocent(x))$
$\forall x: Innocent(x)$
$Innocent(x)$

Notons $\Delta 13$ cette clause ($\Delta 13$: Innocent(x)). Justifier les étapes de la transformation du but.

Transformons les autres assertions en clauses et appliquons le principe de résolution par réfutation :

1. $\neg Innocent(a) \vee Ami(b, v)$	$\Delta 1$	
2. $\neg Innocent(a) \vee Déteste(c, v)$		$\Delta 2$
3. $\neg Innocent(b) \vee Hors\_ville(b)$		$\Delta 3$
4. $\neg Innocent(b) \vee \neg Connaît(b, v)$		$\Delta 4$
5. $\neg Innocent(c) \vee Avec(a, v)$		$\Delta 5$
6. $\neg Innocent(c) \vee Avec(b, v)$		$\Delta 6$
7. $\neg Avec(x, v) \vee \neg Hors\_ville(x)$		$\Delta 7$
8. $\neg Ami(x, y) \vee Connaît(x, y)$		$\Delta 8$
9. $\neg Déteste(v, w) \vee Connaît(v, w)$		$\Delta 9$
10. $Innocent(a) \vee Innocent(b)$	$\Delta 10$	
11. $Innocent(a) \vee Innocent(c)$	$\Delta 11$	
12. $Innocent(b) \vee Innocent(c)$	$\Delta 12$	
13. $Innocent(R) \vee Rép(R)$		$\Delta 13 \vee Rép(R)$, ajout de Rép(R) au but
14. $\neg Innocent(a) \vee Connaît(b, v)$		Résolution binaire de 1 et 8, {x/b, y/v}
15. $\neg Innocent(c) \vee \neg Hors\_ville(b)$		Résolution binaire de 6 et 7, {x/b}
16. $\neg Innocent(a) \vee \neg Innocent(b)$		Résolution binaire de 4 et 14
17. $\neg Innocent(c) \vee \neg Innocent(b)$		Résolution binaire de 3 et15
18. $\neg Innocent(b) \vee Innocent(c)$		Résolution binaire de 11 et 16
19. $\neg Innocent(b)$		Résolution binaire de 17 et18
20. Rép(b)		Résolution binaire de 13 et 19, {x/b}

L'argument du prédicat Rép/1 donne directement la réponse, donc b est coupable. Or, b désigne Bernard, donc Bernard est l'assassin.

Remarques :

- Le principe de résolution est **sain**, c'est-à-dire que lorsque le principe de résolution permet de déduire une clause à partir d'un ensemble de clauses alors toute interprétation satisfaisant l'ensemble de clauses satisfait aussi la clause.
- Par contre, certains principes de résolution tels que le Modus Ponens ne sont pas **complets** pour la **déduction**. Cependant, le principe de résolution est **complet pour la réfutation**, c'est-à-dire que si un ensemble quelconque de clauses est **inconsistant**, il existe une suite finie d'applications du principe de résolution permettant de construire la clause vide.

Illustrons ces résultats sur la complétude au moyen d'un exemple. Soient les assertions prémisses A et $A \rightarrow B$ et soit le but $A \vee B$. Le principe de résolution avec Modus Ponens ne permet pas de déduire ce but :

1. A prémisse
2. $A \rightarrow B$ prémisse
3. B Modus Ponens avec 1 et 2
4. ?

Par contre, la négation du but $\neg(A \vee B)$ donne les deux clauses $\neg A$ et $\neg B$ et on peut démontrer l'inconsistance de l'ensemble de clauses $\{A, \neg A \vee B, \neg(A \vee B)\}$:

1. A prémisse
2. $\neg A \vee B$ clause équivalente à la prémisse $A \rightarrow B$
3. $\neg A$ simplification de la négation de la conclusion
4. { } résolution binaire de 1 et 3

On obtient donc une contradiction dans la base de clauses. On en conclut que la négation de la conclusion était fausse. On peut donc affirmer que $A \vee B$.

Nous allons maintenant voir qu'il existe d'autres formes de représentation des connaissances permettant d'aborder certaines classes de problèmes de façon plus naturelle qu'en utilisant le calcul des prédicats.

2.5 LES GRAPHES CONCEPTUELS

Ce mode de représentation des connaissances a été défini par Sowa (1984) pour traduire la logique sous forme graphique. L'idée de base consiste à représenter les connaissances par les nœuds et les flèches d'un graphe.

Il y a 2 types de nœuds :

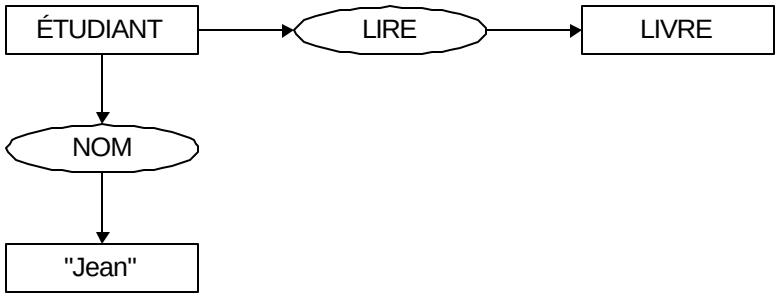
- nœud conceptuel : un rectangle représente une entité ;
- nœud relationnel : une ellipse représente une propriété ou une relation ;

Les flèches relient toujours deux nœuds de type différent.

On peut les noter sous forme graphique ou textuelle (dite aussi forme linéaire).

Exemple : Jean lit un livre.

- Représentation graphique :



- Représentation textuelle :

[« Jean »] <- (NOM) <- [ÉTUDIANT] -> (LIRE) -> [LIVRE].

Dans cet exemple, on a 3 concepts : ÉTUDIANT, LIVRE et le nom « Jean ». On a 2 relations : LIRE et NOM.

2.5.1 Définition

Un **graphe conceptuel** est un graphe biparti connexe orienté[1]. Les deux types de nœuds sont les concepts et les relations. Un arc relie toujours 2 nœuds de type différent.

Un **concept** est identifié par son type. L'identificateur de ce type est inscrit dans le concept. Un type peut être concret ou abstrait. Un concept est dit concret si on peut s'en former une image mentale (un chien, une maison, une université, etc.). Dans le cas contraire, un concept est dit abstrait (l'amour, la beauté, la réussite, la peur, etc.). Tous les types sont structurés par une hiérarchie.

Soit E l'ensemble des types de concepts :

$E \neq \varnothing$

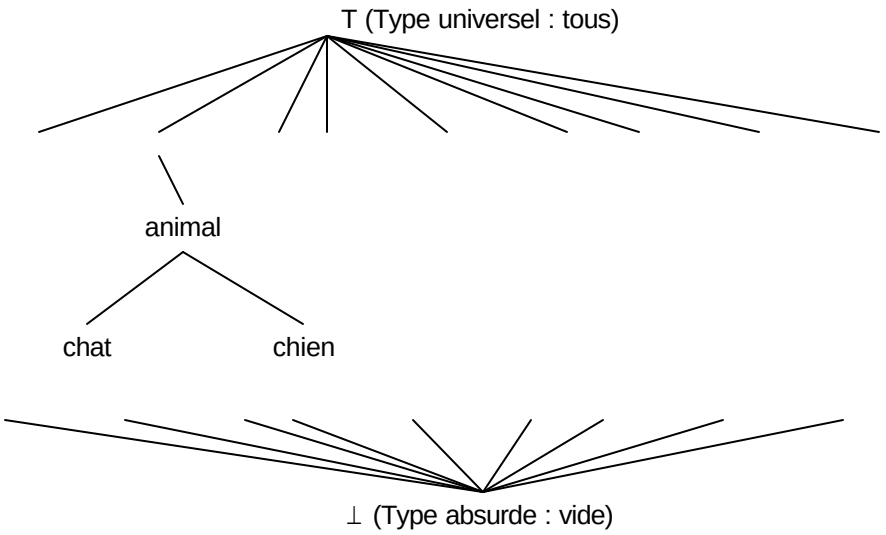
E est un ordre partiel ($\subseteq$)

($E, \subseteq$) est un treillis borné, c'est-à-dire que toute partie finie de E possède une borne inférieure dans E et une borne supérieure dans E. En d'autres termes, il existe un plus petit élément et un plus grand élément.

Tous les types existant dans notre monde son donc présents dans E (d'après les catégories d'Aristote).

Extrait de la représentation de l'ensemble E :

[1] Certains auteurs n'orientent les arcs que lorsque le sens de lecture du graphe n'est pas clair. Toutefois, dans le cadre de ce cours, nous utilisons toujours des graphes orientés.



On utilise également un **référent** ou **marqueur**, qui est ajouté après le type suivi de deux points. Ce référent est une étiquette qui permet de caractériser le concept. Il peut nommer le concept, fournir de l'information sur sa cardinalité ou encore préciser le niveau de connaissance disponible. Il existe plusieurs types de référents, dont voici les plus courants (Sowa, 1984) :

Référent	*Représentation*	*Interprétation*
universel	LIVRE : ∀	tous les livres
générique	LIVRE : * LIVRE	un livre (ce référent est implicite)
particulier	LIVRE : #45	le livre
singulier	LIVRE : 'Les misérables'	un livre intitulé 'Les misérables'
ensemble générique	LIVRE : {*}	des livres
ensemble particulier	LIVRE : {#45, #89}	les livres
ensemble singulier	LIVRE : {'Les misérables', 'Les trois mousquetaires'}	un livre intitulé 'Les misérables' et un livre intitulé 'Les trois mousquetaires'
mesure générique	Température : @20°	une température de 20 degrés
mesure particulière	Longueur : #67 @30m	la longueur de 30 mètres
cardinalité d'un ensemble	LIVRE : {*} @2	deux livres

On note I l'ensemble des marqueurs de singularité : I = {#1, #2, #3, #4, etc.}

L'exemple précédent complété avec les marqueurs d'après Sowa (1984) devient :

Chaque **relation** a aussi un type qu'on inscrit aussi dans la représentation d'un nœud relationnel. Dans l'exemple précédent, la seule relation est de type LIRE. Sowa (1984) a défini un certain nombre de types de relation (voir tableau suivant). Cette définition permet d'uniformiser les représentations sous forme de graphes conceptuels.

Nom de la relation	*Graphe conceptuel*	*Explication*
AGENT	[C2] -> (**AGNT**) -> [C1]	C1 est l'acteur de l'action C2.
ATTRIBUT	[C2] -> (**ATTR**) -> [C1]	C1 est l'attribut de C2.
CAUSE	[C2] -> (**CAUS**) -> [C1]	C1 est la cause de C2.
CARACTÉRISTIQUE	[C2] -> (**CHCR**) -> [C1]	C1 est une caractéristique de C2.
BUT	[C2] -> (**GOAL**) -> [C1]	C1 est le but atteint par C2.
DURÉE	[C2] -> (**DUR**) -> [C1]	C2 persiste pendant le temps C1.
FUTUR	(**FUTR**) -> [C1]	C1 se réalisera dans le futur.
LIEU	[C2] -> (**LOC**) -> [C1]	C1 est l'endroit où se passe C2.
MANIÈRE	[C2] -> (**MANR**) -> [C1]	C2 se réalise de la manière C1.
NÉGATION	(**NEG**) -> [C1]	C1 est faux.
OBJET	[C2] -> (**OBJ**) -> [C1]	C1 est l'objet de C2.
PARTIE	[C2] -> (**PART**) -> [C1]	C1 est une partie de C2.
PASSÉ	(**PAST**) -> [C1]	C1 s'est réalisé dans le passé.
QUANTITÉ	[C2] -> (**QTY**) -> [C1]	C1 indique le nombre d'entités de C2.
RÉCIPIENT	[C2] -> (**RCPT**) -> [C1]	C1 reçoit l'objet de C2.
RÉSULTAT	[C2] -> (**RSLT**) -> [C1]	C1 est le résultat de C2.

Convention sur l'orientation des arcs : Plusieurs flèches peuvent entrer dans un nœud relationnel, par contre il ne peut en sortir qu'une seule. Lorsqu'il y a plusieurs flèches qui entrent dans une relation, on indique parfois l'ordre des flèches si cela s'avère important.

Représentation textuelle et convention d'écriture

Représentation graphique	*Représentation textuelle*
concept	[TYPE : référent]
relation	(TYPE)
flèche	→ ou ←

2.5.2 Les quatre opérations définies pour les graphes conceptuels

Copie : À partir d'un graphe g, on peut construire un nouveau graphe g', copie de g.

Restriction : Dans un graphe, on peut remplacer un concept par l'instance de ce concept.

- [TYPE : *] devient alors [TYPE : #n]

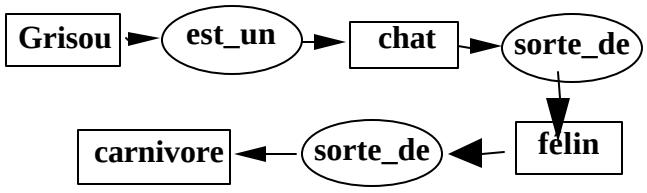
Par exemple, ceci revient à remplacer CHAT par « Grisou » (le nom du chat).

- [TYPE1 : marqueur] devient alors [TYPE2 : marqueur] si TYPE2 ⊆ TYPE1.

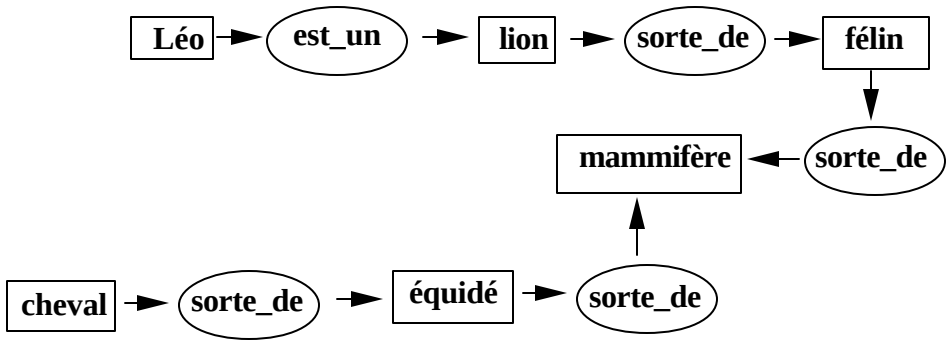
Par exemple, ceci revient à remplacer FÉLIN par CHAT.

Jointure ou somme : On peut joindre deux graphes conceptuels qui possèdent un concept identique. Ainsi, si le concept C1 dans le graphe g1 est identique au concept C2 dans le graphe g2, on peut construire un nouveau graphe g à partir de g2 en y remplaçant C2 par C1 et en ajoutant tout le sous-graphe de g1, issu de C1.

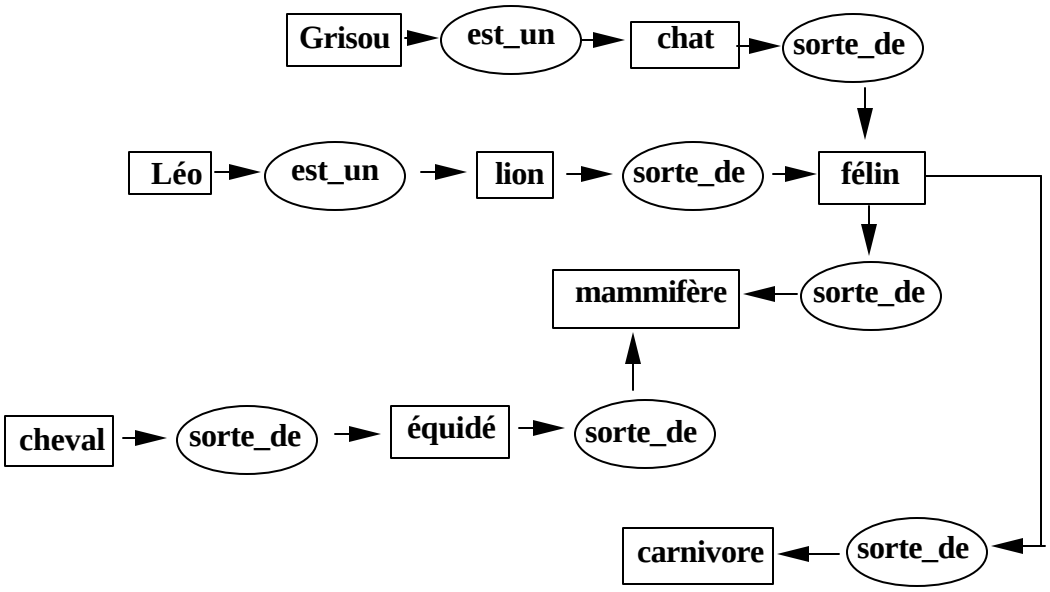
Exemple : Soit le graphe g1 :



Soit le graphe g2 :



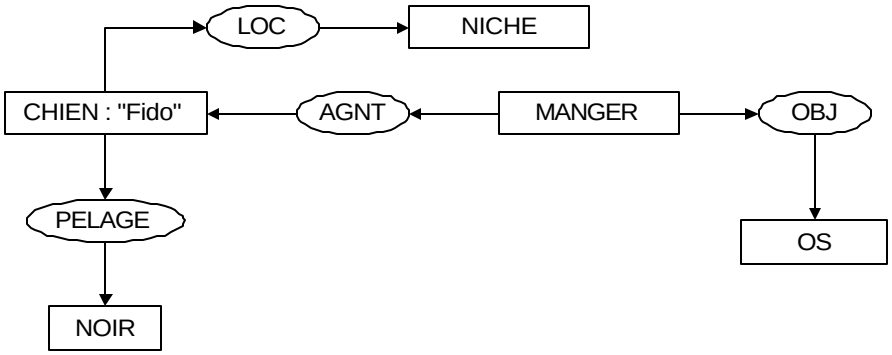
On fait la jointure ou somme de g1 et de g2 qui contiennent le sous-graphe constitué du seul concept félin.



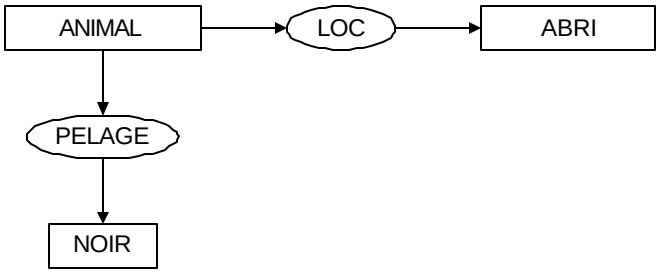
Simplification : Si on a deux relations de même type avec les mêmes arguments, alors on peut supprimer un des deux nœuds relationnels et tous les arcs qui sont connectés.

Exemple récapitulatif : Nous appliquerons successivement les règles de restriction, de jointure et de simplification.

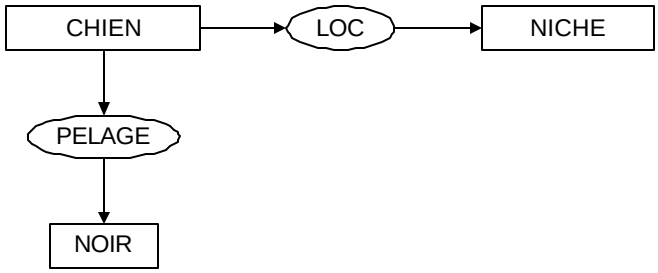
Soit le graphe g1 :



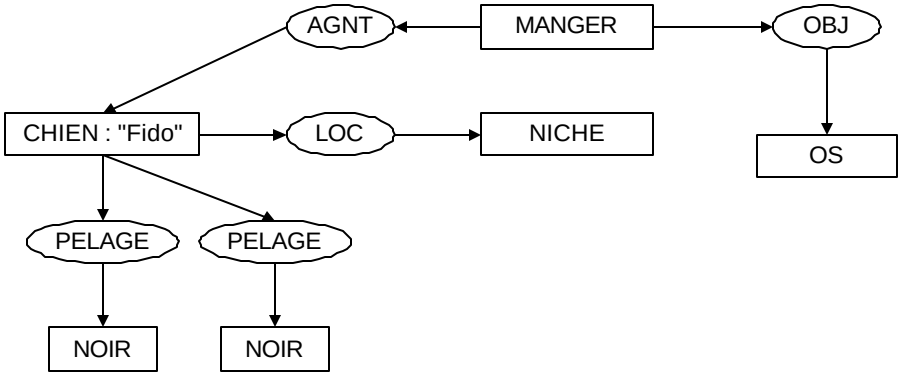
Soit le graphe g2 :



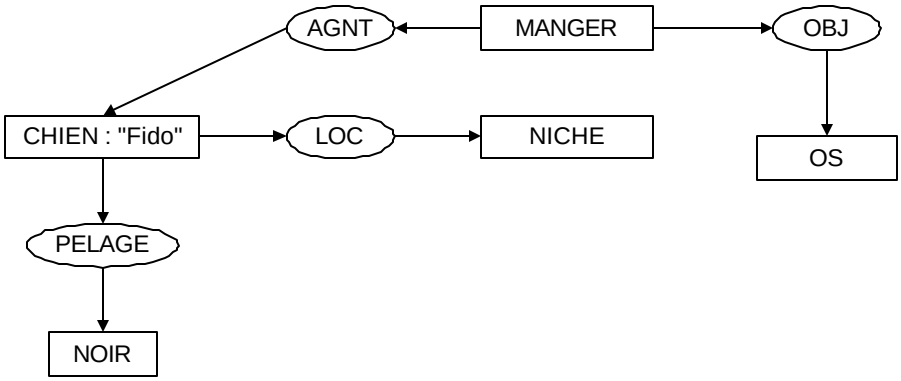
Nous effectuons d'abord deux restrictions sur le graphe g2. Le concept chien est un sous-type du concept animal et le concept niche est un sous-type du concept abri. On obtient alors le graphe g3.



Nous appliquons maintenant une jointure entre g1 et g3 en spécifiant l'abri de Fido, ce qui est traduit par le graphe suivant g4.



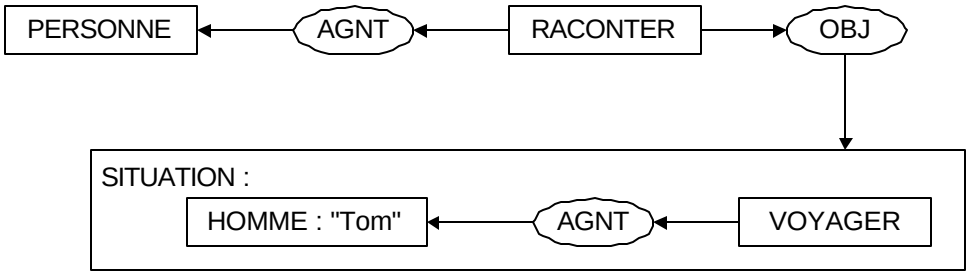
Lors de cette dernière opération de jointure, on n'a repris qu'une occurrence du concept noir. Ce n'est pas une application de la règle de simplification. Toutefois, on constate que le sous-graphe [chien : « Fido »] -> (pelage) -> noir est représenté deux fois dans g4. On applique alors la règle de simplification pour obtenir le graphe final g5.



Remarque : Ces règles de construction ne sont pas des règles d'inférence. En d'autres termes, elles permettent de construire des phrases sémantiquement plausibles mais pas nécessairement vraies.

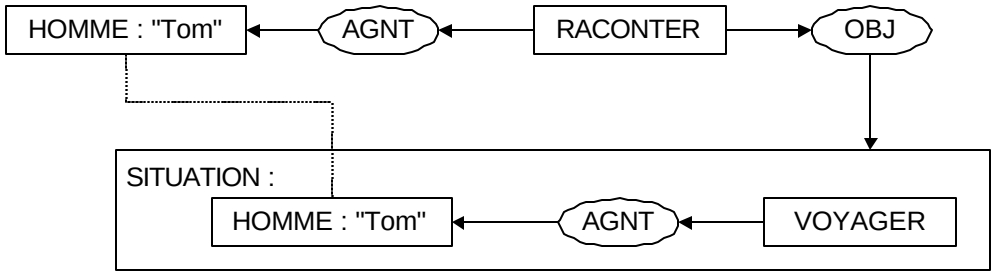
2.5.3 Les contextes et les liens de coréférence

Un contexte est un concept dont le référent est défini par un graphe conceptuel complet. Ceci permet de représenter des propositions ou des situations et de travailler avec la logique de premier ordre. Par exemple, pour représenter la phrase : « Une personne raconte le voyage d'un homme Tom », on utilise un contexte de type SITUATION, qui décrit ce qui est raconté. La figure suivante illustre graphiquement cet exemple.



De plus, en liant la relation monadique NEG à un contexte, il est alors possible de nier une proposition. De plus, en regroupant plusieurs contextes sous un même contexte de niveau supérieur, on introduit la notion de conjonction. La négation et la conjonction permettent à elles seules de représenter tous les opérateurs booléens et offrent ainsi la possibilité de travailler en logique du premier ordre.

Si on reprend l'exemple précédent et qu'on suppose que c'est Tom qui raconte lui-même son voyage, alors on doit faire deux fois appel au concept [HOMME : «Tom »], à l'intérieur et à l'extérieur du contexte. Aussi, lorsqu'un contexte réutilise un concept apparaissant déjà à l'extérieur de celui-ci, un lien de coréférence est nécessaire pour indiquer qu'il s'agit bel et bien du même concept. Les liens de coréférence sont graphiquement représentés par un arc en pointillé non orienté. La figure illustre le fait que Tom raconte lui-même son voyage.



Dans la forme linéaire, on utilise une variable (x, y, z, etc.) pour préciser qu'on fait référence au même concept.

[HOMME : x « Tom »] <- (AGNT) <- [RACONTER] -> (OBJ) -> [SITUATION : [HOMME : x « Tom »] <- (AGNT) <- [VOYAGER]]

2.6 LES SYSTÈMES DE SCHÉMAS

2.6.1 Définition

Étant donné un domaine, un bon système de représentation des connaissances devrait avoir les propriétés suivantes :

- la capacité de représenter toutes les sortes de connaissances du domaine ;
- la capacité de manipuler les constructions formelles de façon à pouvoir dériver de nouvelles constructions formelles correspondant aux connaissances déductibles des connaissances disponibles ;

- la capacité d'intégrer dans la structure de connaissances de nouvelles informations permettant d'orienter le mécanisme d'inférence dans des directions profitables ;
- la facilité de recevoir de nouvelles connaissances.

Par exemple, la représentation relationnelle est simple et pratique lorsqu'on a affaire à des données présentant une certaine homogénéité.

Joueur	*Taille (m)*	*Poids (kg)*	*Frappe/Lance*
T. Wallach	1.90	85	droite/droite
M. Grissom	1.85	82	droite/droite
D. DeShield	1.80	83	gauche/droite

Dans cet exemple, la taille, le poids, etc., sont les attributs des objets <Joueur>. Cette représentation est limitée quant à sa capacité de faire des déductions. Une simple consultation permet, par exemple, de connaître la taille d'un joueur ou encore sa position au bâton mais ne permet pas de déduire directement qui est le plus grand ou le plus lourd. Elle ne nous renseigne pas non plus beaucoup sur ce qu'est un joueur de base-ball ni sur tout ce qui concerne son univers.

Étant donné un réseau sémantique, on appelle **schéma** (en anglais *frame*) le sous-réseau sémantique identifié par un nœud accompagné de ses attributs et de leurs valeurs (éventuellement déterminées par des attachements procéduraux).

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

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

Dans les systèmes de schémas, il peut être intéressant de faire jouer à la fois le rôle d'un objet et celui d'une classe à un schéma.

Par exemple :

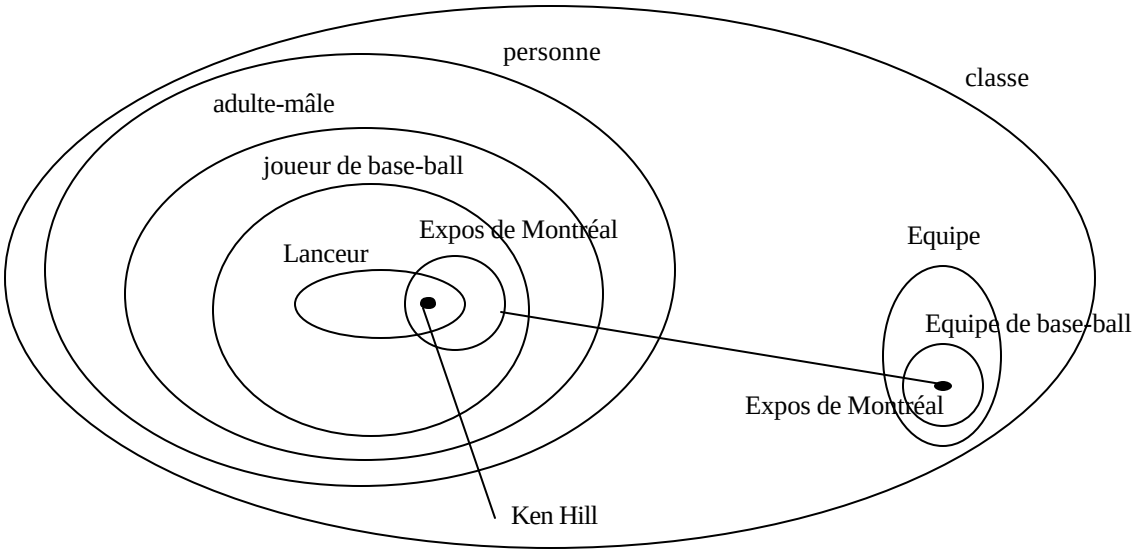
- L'équipe de base-ball des Expos de Montréal est un regroupement de joueurs (classe) qui constitue une sous-classe (relation sorte_de) de la classe de tous les joueurs de base-ball. Par ailleurs, cette même équipe (objet) fait partie (relation est_un) de l'ensemble des équipes de base-ball majeur (classe).
- Si classe représente l'ensemble (abstrait) de toutes les classes alors les deux relations suivantes tiennent : classe est_un classe et classe sorte_de classe.

Les réseaux sémantiques n'offraient pas de facilité pour distinguer les attributs propres à une classe des attributs possédés par toutes les instances de cette classe. Par exemple, la classe des étudiants inscrits à un cours donné possède l'attribut **cardinalité** (le nombre d'inscrits) en propre alors que chacun des étudiants possède l'attribut **note_obtenue**. Dans un système de schémas, on distinguera les attributs héritables par les instances d'une classe en leur adjoignant le symbole *.

Voyons l'exemple suivant, inspiré de Rich et Knight (1991).

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

Dans le but de représenter ces connaissances, après avoir analysé le texte, distingué les objets et les classes (tenant compte que certaines entités peuvent jouer les deux rôles), on construit le diagramme suivant exprimant la plus grande partie des liens.



Ce diagramme est une représentation ensembliste permettant d'identifier les objets, les classes et les relations **est_un** et **sorte_de**.

On complète la description du système de schémas au moyen des données suivantes :

Ken Hill	
est_un	*Lanceur*
est_un	*Expos de Montréal*
bras lanceur	*droit*

Lanceur	
sorte_de	*Joueur de baseball*
cardinal	*280*
*moyenne	*.180*

Joueur de base-ball	
sorte_de	*Adulte-mâle*
cardinal	*672 /* nombre total de joueurs */*
*taille	*1,85m*
*frappe	*idem dextérité*
*moyenne	*.250*

Adulte-mâle	
sorte _de	*personne*
cardinal	*2 milliards*
*taille	*1.75m*

Personne	
sorte _de	*classe*
cardinal	*6 milliards*
*dextérité	*droite*

Expos de Montréal	
est_un	*Équipe du base-ball majeur*
sorte_de	*Joueur de base-ball*
gérant	*Felipe Alou*
effectifs	24
*uniforme	blanc à rayures bleues
*nom_équipe	Expos

Équipe du base-ball majeur		
sorte_de	*équipe sportive*	
est_un	*classe*	
cardinal	28	*/* nombre d'équipes ...*/*
*effectifs	24	*/* nombre de joueurs par équipe */*
*gérant		

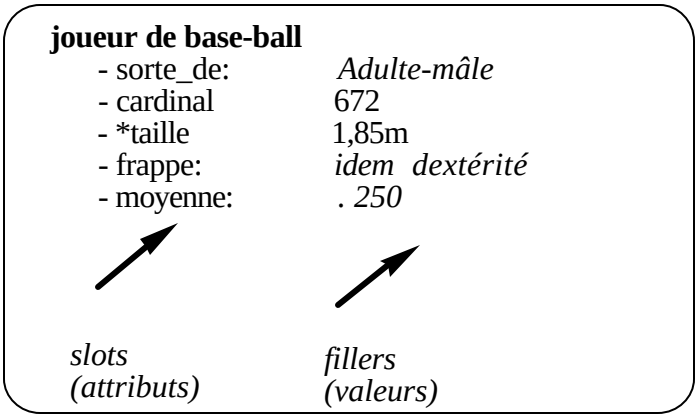
Équipe sportive	
sorte_de	*classe*
est_un	*classe*
*gérant	
*uniforme	

Classe	
est_un	*classe*
sorte_de	*classe*

De ce tableau, il ressort que chaque schéma est identifié par :

- son nom,
- une liste d'attributs éventuellement accompagnés d'une valeur.

Exemple :



Les attributs accompagnés du symbole * sont ceux héritables par toute instance de la classe.

2.6.2 Héritage dans les systèmes de schémas

La propriété d'**héritage** est un mécanisme d'inférence par lequel les objets d'une classe héritent des attributs (et de leurs valeurs) de superclasses. Pour rechercher les attributs et les valeurs correspondantes d'un schéma, il faut disposer d'un **algorithme de recherche d'attributs.**

Voici un exemple d'algorithme de recherche d'attributs :

Soit l'objet O dont on cherche la valeur V de l'attribut A dans un système de schémas. Repérer O dans le système :

Si O a l'attribut A et celui-ci a la valeur v alors **rapporter :** V = v et le **succès** ;
Sinon,
Si O n'a pas de valeur pour l'attribut **est_un** rapporter **l'échec**,
Sinon, aller au nœud ainsi désigné et rechercher une valeur pour l'attribut A :

- en cas de succès **rapporter** la valeur ;
- en cas d'échec chercher de nœud en nœud jusqu'au **succès** ou jusqu'à ce qu'il n'y ait plus de valeur pour l'attribut **sorte_de** (dans ce cas, **échec**) :
  - obtenir le nœud désigné par l'attribut **sorte_de** du nœud courant et aller au nœud ainsi identifié ;
  - s'il existe une valeur pour l'attribut A, **la rapporter.**

Exemple : Recherchons tous les attributs de Ken Hill :

- attribut direct ⇒ bras lanceur = droit
- Ken Hill **est_un** lanceur ⇒ héritage : moyenne = .180
- un Lanceur est une **sorte_de** Joueur de base-ball ⇒ héritage : taille = 1,85m
  frappe = idem dextérité #
  moyenne = .250

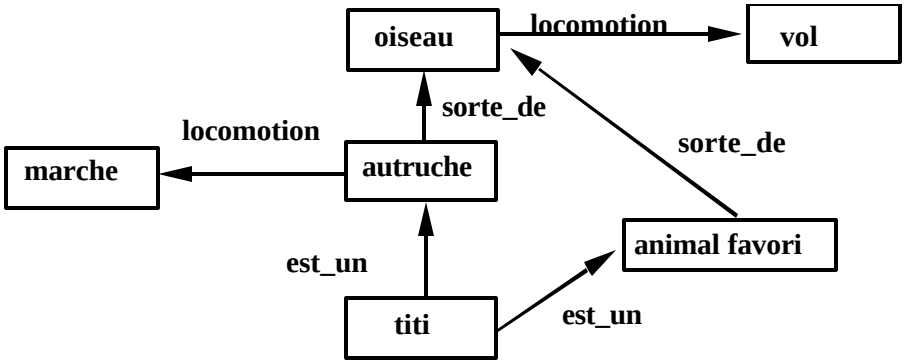
- un Joueur de base-ball est une **sorte_de** Adulte-mâle ⇒ héritage : taille = 1,75m
- un Adulte-mâle est une **sorte_de** Personne ⇒ héritage : dextérité = droite
- Ken Hill **est_un** Expos de Montréal ⇒ héritage : uniforme = blanc à rayures bleues
  nom_équipe = Expos
- un Expos de Montréal est une **sorte_de** Joueur de base-ball ⇒ héritage : déjà traité
- un Expos de Montréal **est_un** Équipe du baseball majeur (L'héritage pour Ken Hill ne peut se faire puisque Expos de Montréal est traité ici comme un objet).

Remarque : #frappe = idem dextérité est interprété comme un calcul à effectuer pour trouver la valeur de l'attribut frappe. Cette valeur est obtenue par héritage de la classe Adulte-mâle (dextérité = droite). Pour Ken-Hill, l'attribut frappe prend la valeur droite. Cette façon de calculer la valeur d'un attribut est appelée calcul par **attachement procédural**. Ce calcul peut aussi prendre la forme d'une vraie procédure.

2.6.3 Traitement des conflits d'héritage

Le système de schémas permet de laisser cohabiter d'apparentes ambiguïtés. Dans l'exemple, on voit que comme lanceur Ken Hill hérite d'une moyenne de .180 alors que comme joueur de base-ball, il hérite d'une moyenne de .250. Comme joueur de base-ball, il mesure 1,85 m alors que comme adulte mâle il mesure 1,75 m.

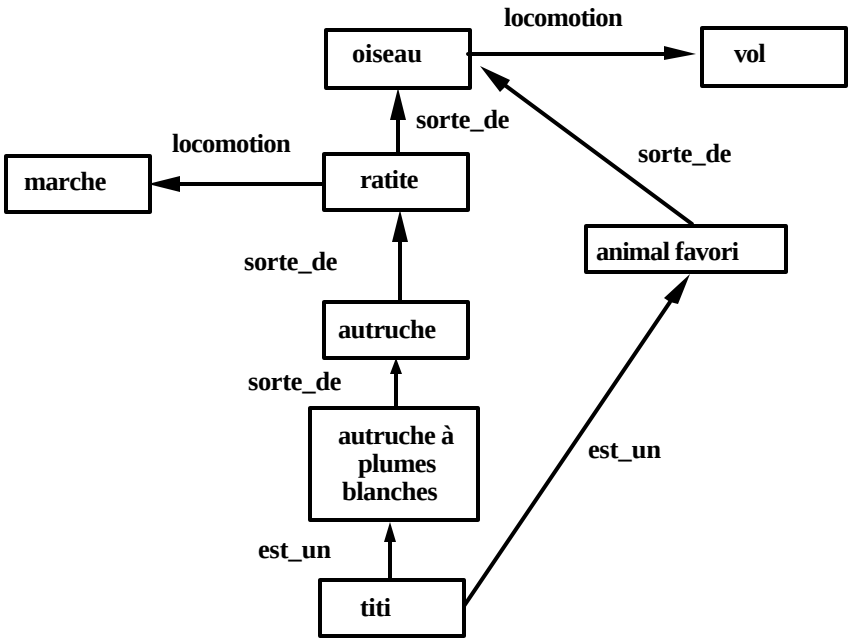
Considérons l'exemple concernant le moyen de locomotion de l'autruche :



Il y a conflit de valeurs pour le calcul de l'attribut locomotion pour l'objet titi. Le chemin passant par autruche donne priorité à la valeur marche pour l'attribut locomotion tandis que celui qui passe par animal favori lui donne la valeur vol. On pourrait résoudre l'ambiguïté en choisissant le plus court chemin entre titi et les classes dont titi hérite et qui ont une valeur fixée pour l'attribut locomotion :

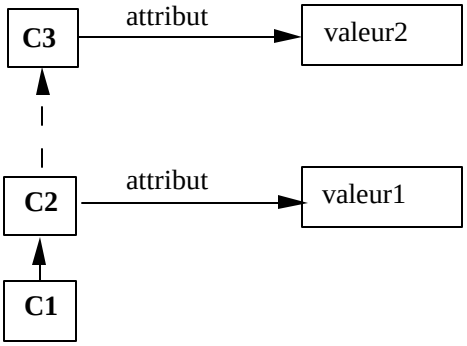
Chemin	*Valeur de locomotion*	*Longueur du chemin*
titi → autruche	marche	1
titi → autruche → oiseau	vol	2
titi → animal favori → oiseau	Vol	2

Cependant, la méthode est imparfaite comme le montre l'exemple suivant.



Chemin	*Valeur de locomotion*	*Longueur du chemin*
titi → autruche à plumes blanches → autruche→ ratite	marche	3
titi → animal favori → oiseau	vol	2

La longueur n'est plus ici le bon critère ; on peut toutefois trouver une solution en utilisant le critère de **proximité inférentielle**. Supposons en effet qu'une classe C1 hérite un attribut (directement ou non) de deux classes C2 et C3. On dit que C2 est de proximité inférentielle (par rapport à C1) supérieure à celle de C3 si C2 hérite l'attribut de C3.



Selon le critère de proximité inférentielle, C1 héritera de attribut = valeur1.

Dans l'exemple de l'autruche, on voit que titi hérite de l'attribut locomotion (valeur = marche) de la classe ratite et du même attribut de la classe oiseau (valeur = vol). Cependant, la classe ratite est de proximité inférentielle (par rapport à titi) supérieure à celle de la classe oiseau. Donc, titi a la marche comme moyen de locomotion.

Les systèmes de schémas constituent un moyen très naturel de représenter les connaissances, surtout lorsqu'elles concernent des objets aux structures complexes. Il existe une gamme de langages spécialisés permettant le traitement de systèmes de schémas.

Dans les sections suivantes, nous allons étudier d'autres systèmes de représentation particulièrement adaptés pour le traitement de la langue naturelle.

2.8 LES SCRIPTS

Pour **comprendre** un texte en langage naturel, un programme doit avoir une bonne connaissance du monde environnant (le contexte). Cet environnement peut changer brusquement, sans avertissement, provoquant ainsi d'éventuelles difficultés d'interprétation.

Un **script** est une représentation structurée d'une suite d'événements, dans un contexte particulier. Un exemple classique de script est celui du restaurant, publié par Shank en 1977. Le script du restaurant, même s'il concerne l'assimilation de nourriture, est différent de celui du Fast-Food ou de celui du repas familial.

Les composantes d'un script sont :

- les **conditions d'entrée**, soit l'ensemble des conditions requises pour que le script puisse être appelé (Exemples : le restaurant est ouvert, le client a faim) ;
- le **résultat**, soit l'ensemble de faits après la réalisation du script. (Exemples : le client s'est appauvri, le client n'a plus faim, le restaurateur a plus d'argent) ;
- les **objets spécifiques** (props), soit l'ensemble des entités pour lesquelles sont affirmés des faits. (Exemples : tables, chaises, garçon, client, chef, argent) ;
- les **rôles**, soit l'ensemble des actions que chaque entité doit effectuer (ou subir) dans le cadre de sa participation au script. (Exemples : le garçon sert à table et présente le menu, le client mange et paye) ;
- les **scènes**, soit l'ensemble des suites d'événements concernant les objets et les rôles. Le script est divisé en une suite ordonnée de scènes. Chaque scène est présentée au moyen d'un formalisme de dépendance conceptuelle. (Exemples : l'entrée au restaurant, la commande, la consommation, la sortie du restaurant).

Tout programme, basé sur un script, devant effectuer l'interprétation d'un texte en langue naturelle :

- vérifie si le texte satisfait les conditions d'entrée du script ;
- lie les objets et les personnes du texte aux variables du script ;
- utilise les rôles des individus à l'intérieur des scènes afin de combler les informations manquantes ;
- produit une représentation structurée du texte, la plus complète possible.

Cette représentation structurée facilite ensuite d'éventuels questionnements sur l'histoire représentée par le texte. Dans le tableau de la page suivante, nous avons succinctement représenté le script du restaurant.

Script	**Scène 1 : entrée au restaurant**
Restaurant	
	C PTRANS C au restaurant
Props	C ATTEND yeux aux tables
Tables	C MBUILD où s'asseoir
Menu	C PTRANS C à une table
N = nourriture	C MOVE C position assise
Facture	*(passer à la scène 2)*

Argent

Rôles :
C = client
G = garçon
Ch = chef
Ca = caissier
P = propriétaire

Conditions d'entrée
C a faim
C a de l'argent

Résultat
C a moins d'argent
P a plus d'argent
C n'a plus faim
C est content
(optionnel)

Scène 2 : la commande (3 cas)

menu sur la table	*C demande menu*	*G apporte menu*
C ATRANS menu à C	G PTRANS G à table G ATRANS menu à C	C MTRANS signe à G G PTRANS G à table C MTRANS « menu ? » à G G PTRANS G à menu G PTRANS G à table G ATRANS menu à C

Choix de la nourriture

C MTRANS liste de nourriture au conscient de C
* C MBUILD choix de N
C MTRANS signe à G
G PTRANS G à table
C MTRANS « je désire N » à G
G PTRANS G à Ch
G MTRANS (ATRANS N) à Ch

commande non satisfaisante	*commande acceptable*
Ch MTRANS « pas disponible » à G G PTRANS G à C G MTRANS « pas disponible » à C *(revenir en * ou aller en scène 4, issue : rien à payer)*	Ch DO (prépare script à N) *(passer à la scène 3)*

Scène 3 : la consommation

Ch ATRANS N à G
G ATRANS N à C
C INGEST N
(*option : retour à scène 2 pour une commande supplémentaire, sinon : aller en scène 4*).

Scène 4 : la sortie du restaurant (2 cas)

le garçon amène la facture	*le client réclame la facture*
G MOVE (écrire facture) G PTRANS G à C G ATRANS facture à C C ATRANS pourboire à G C PTRANS C à Ca C ATRANS argent à Ca C PTRANS C à sortie restaurant.	C MTRANS G (*appel de* G) G ATRANS facture à C C PTRANS C à Ca C ATRANS argent à Ca C PTRANS C à sortie restaurant.

issue : rien à payer

Imaginons maintenant que nous disposions d'un programme capable de lire et d'utiliser ce script et pouvant également interpréter de petites histoires en français. Voyons quelques questions auxquelles le lecteur pourrait répondre mais qui peuvent causer des difficultés au programme.

Histoire 1 : *Julie est allée au restaurant la nuit dernière. Elle a commandé un steak. Au moment de payer, elle s'est rendue compte qu'elle n'avait pas assez d'argent. Elle a couru à la maison puisqu'il s'est mit à pleuvoir.*

Questions : Est-ce que Julie a mangé la nuit dernière ?

Est-ce que Julie a utilisé de l'argent ou une carte de crédit ?

Comment Julie pouvait-elle obtenir un menu ?

Qu'est-ce que Julie a acheté ?

Histoire 2 : *Henri est sorti pour manger. Il s'est assis à une table et a appelé le garçon qui lui a apporté le menu. Il a commandé un sandwich.*

Questions : Pourquoi le garçon a-t-il apporté un menu à Henri ?

Est-ce que Henri était dans un restaurant ?

Qui a payé ?

Qui a commandé le sandwich ? (*ambiguïté du dernier « il »*).

Histoire 3 : *Marie est allée au restaurant. On l'a conduite à une table et elle a fait la commande d'un steak au garçon. Elle a attendu un grand moment. Finalement, elle s'est fâchée et a quitté le restaurant.*

Questions : Qui a attendu ?

Pourquoi a-t-elle attendu ?

Qui s'est fâché et a quitté le restaurant ?

Pourquoi cette personne s'est-elle fâchée ?

A noter que le programme ne pourrait pas répondre à une question comme «*Pourquoi les gens se fâchent-ils quand le garçon ne vient pas rapidement ?* »

Quelle est donc l'utilité du script ?

On sent intuitivement que le script est surtout utile pour aider à compléter l'interprétation sémantique d'une histoire. Le script permet de substituer des constantes (valeurs par défaut) à certaines variables des phrases analysées. Il permet d'ajouter certains schémas conceptuels non explicites dans l'histoire. Son but premier est donc de favoriser la production d'une image du texte de l'histoire mieux intégrée dans son contexte d'utilisation. On devine qu'un script peu ou trop détaillé ne favorisera pas la production d'une interprétation concise et complète. Par ailleurs, des précautions devront être prises pour identifier correctement le script du contexte courant. Par exemple, le texte suivant est acceptable en français :

Jean s'est arrêté à son restaurant favori, sur le chemin du Grand-Théâtre. Il a bien accepté la facture puisqu'il adore Mozart.

Mais à quel moment faut-il déclencher le script du restaurant et le script du théâtre ? La facture dont on parle, relève-t-elle du premier ou du second ?

