Représentation des connaissances et raisonnement

1

INTELLIGENCE ARTIFICIELLE

ECOLE NATIONALE POLYTECHNIQUE D'ORAN
Département Génie des systèmes
Filière IMSI : 4ème année ingénieur
Mme Si-Moussa. h

L’objectif du cours

- présenter les bases de l'intelligence artificielle,
- en particulier, les systèmes à base de connaissances, de leur conception à leur implémentation dans des domaines variés.

- **Connaissances préalables recommandées :**

Notions de base en logique mathématique

Plan

- **1. REPRÉSENTATION DES CONNAISSANCES**
- 1.1. LOGIQUE DES PROPOSITIONS
- 1.2. LOGIQUE DES PRÉDICATS
- **2. PROGRAMMATION LOGIQUE :PROLOG (LES FONDEMENTS THÉORIQUES DU LANGAGE PROLOG)**
- **3. SYSTÈMES MULTI-AGENTS**
- 3.1. MOTIVATION POUR LES AGENTS
- 3.2 DÉFINITIONS D'AGENTS (CARACTÉRISTIQUES, CLASSIFICATION)
- 3.3 SYSTÈMES MULTI-AGENTS
- 3.4 INTELLIGENCES DES AGENTS (INTERACTION ENTRE AGENTS)
- 3.5 LIENS AVEC D'AUTRES DISCIPLINES
- 3.6 DOMAINES DE RECHERCHE
- 3.7 EXEMPLES D'APPLICATIONS
- **4. RAISONNEMENT À PARTIR DE CAS**
- 4.1 LE RÀPC : NOTIONS DE BASE ET NOTATIONS
- 4.2 OBJETS DE BASE DU RÀPC
- 4.3 ÉTAPES DU RÀPC
- 4.4 CONNAISSANCES UTILISÉES POUR LE RÀPC
- **5. IMPRÉCISION ET INCERTITUDE DANS LES SYSTÈMES À BASE DE CONNAISSANCES**
- **6. APPRENTISSAGE AUTOMATIQUE, RÉSEAUX DE NEURONES**

Formes de représentation des connaissances

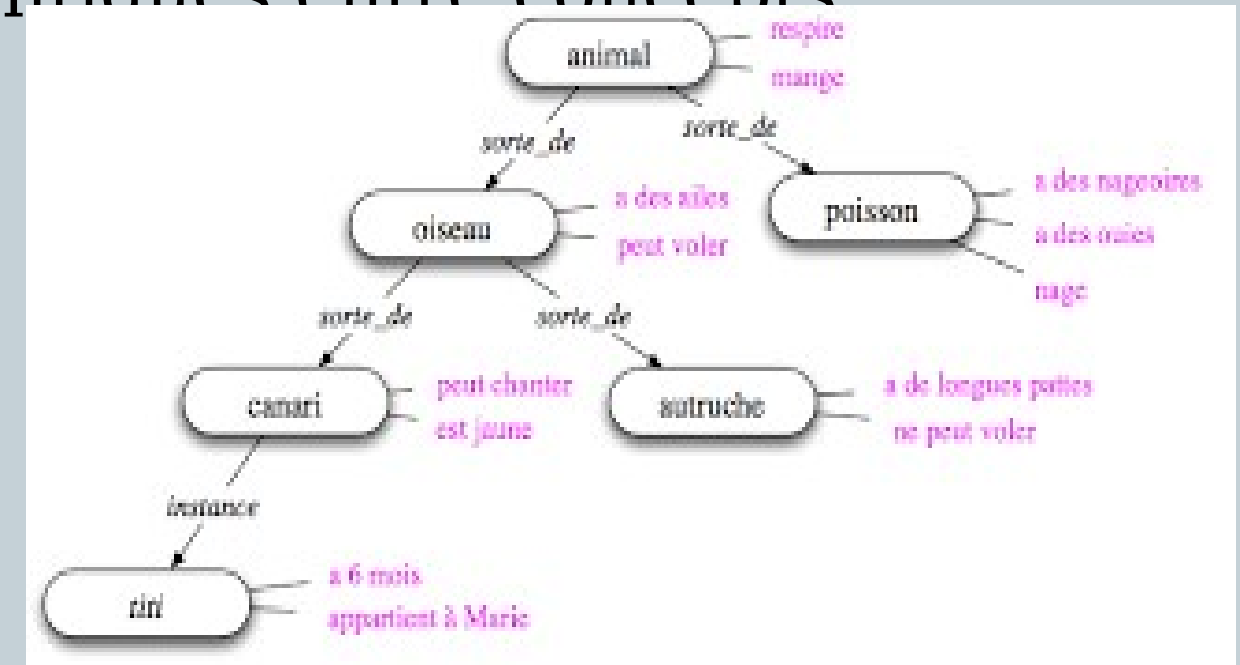
- Règles de production (basé sur la logique du premier ordre)

- Représentation structurée:

a. Réseau Sémantique
b. Frames
c. Logiques Terminologiques KL-ONE
d. Graphes conceptuels

Réseaux Sémantiques

- Un **réseau sémantique** est un graphe marqué destiné à la représentation des connaissances, qui représente des relations sémantiques entre concepts



Frames (Cadres sémantiques)

Un frame décrit une famille d'objets ou classe représenté par un schéma de classe. Un objet particulier de cette classe est décrit lui-même par un schéma, dit schéma d'instance, qui diffère du schéma de classe par la valeur donnée aux attributs. Une instance peut-être complète ou partielle.

Logique Terminologique (Logique de description)

les **logiques de description** (LD), une famille de langages de représentation de connaissances qui exploitent, en général, des sous-ensembles décidables (pour une logique, un problème de raisonnement est décidable si une machine de Turing peut le résoudre en un nombre fini d'étapes) de la logique de premier ordre. Ils ont été largement étudiés et utilisés dans plusieurs systèmes à base de connaissances.

Graphe Conceptuel

- Un **graphe conceptuel** est un formalisme de représentation de connaissances et de raisonnements.
- Un GC est un graphe bipartite orienté

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

Raisonnements

- Raisonnement Formel
- Raisonnement Procédural
- Raisonnement par Analogie
- Raisonnement par généralisation et Abstraction
- Raisonnement Géométrique

Raisonnement Formel

Un raisonnement est dit formel s'il est conduit en tenant compte seulement des relations entre objets ou propositions sans faire appel à des connaissances sur les objets mentionnés.

Raisonnement Procédural

- Le raisonnement procédural met l'accent sur les incapacités de la personne et sur les façons de les régler. Ce type de raisonnement est associé aux stratégies d'intervention visant l'amélioration des capacités de la personne.
- Déstiné aux applications temps réel.

Raisonnement Par Analogie

Le raisonnement par analogie consiste à mettre en relation un cas connu (la source) et un nouveau cas, moins bien connu (la cible) afin de faciliter la résolution ou la compréhension de la cible.

Raisonnement par Généralisation et Abstraction

L'abstraction de concepts à partir d'un ensemble d'objets spécifiques fait elle-même appel à une capacité de généralisation qui permet de s'affranchir des propriétés individuelles des objets pour ne retenir que l'information pertinente qui les unit et les différencie d'autres objets.

Raisonnement Géométrique

Il s'agit des transformations avec lesquelles on peut étudier les figures géométriques et les solides géométriques en mathématique élémentaire.

1. Représentation des connaissances

1.1 LOGIQUE DES PROPOSITIONS (PRINCIPES DE BASE)

Nombreuses utilisations de la logique en informatique

- intérêt immédiat en algo: écrire des conditionnelles et des conditions d'arrêt correctes;
- conception de circuits (portes "et" "ou" "non" réalisant une fonction de dans);
- preuves de programmes :

montrer qu'un programme satisfait une propriété (formule logique)

Exemple: fonction de recherche d'un entier dans un tableau trié qui rend "présent" ou "absent".

Correction du programme:

hypothèse: le tableau en entrée est trié

conclusion: on a "présent" si et seulement si la valeur est dans le tableau, et "absent" sinon.

- bases de données

On utilise des propriétés logiques, par exemple, si les prédicats de base sont "être le grand-père de", "être l'oncle de", etc. :

chercher "un garçon de 9 ans ayant un oncle de 32 ans et un grand père paternel de 70 ans"

revient à chercher "un homme de 32 ans ayant un père de 70 ans et un neveu de 9 ans"

La logique permet de produire toutes ces formulations équivalentes, étant donné les équivalences de base "x est le neveu de y" "y est l'oncle de x"

Logique

- Ce chapitre est une introduction à la formalisation des modèles logiques et du raisonnement dans ces modèles;
- Le calcul propositionnel est un premier modèle logique limité à des raisonnements sans variables;
- Le calcul des prédicats étend le modèle propositionnel à des individus et à des propriétés sur ces individus,
- Il est complété par un algorithme de démonstration automatique , dont une application majeure est le langage de programmation logique « prolog ».

Calcul propositionnel

- Le calcul propositionnel est un ***système formel*** pour la représentation de propositions logiques et des raisonnements.
- Le système formel fournit une description générique de systèmes à partir d'un noyau minimal et de règles d'extension.
- La réalisation d'un système formel fournit un cadre de représentation d'une réalité donnée où des règles automatiques produisent tous les éléments vrais.

1. Avoir un petit nombre de vérités initiales
2. Disposer de mécanismes de raisonnements
3. Pour inférer des vérités cachées.

Calcul propositionnel (exemples)

Si le feu est rouge ou orange et si je respecte le code de la route alors je ne passe pas.

Je suis en infraction si et seulement si je ne respecte pas le code .

Le feu est rouge et je passe

donc je suis en infraction.

Connecteurs logiques

- La négation qui est unaire (« ne pas » ¬)
- La conjonction « et » , la disjonction « ou », l'implication « si alors » et l'équivalence « si et seulement si » qui sont binaires.
- Est-ce qu'il existe une démonstration formelle de la conclusion à partir des hypothèses?

Langage propositionnel (syntaxe)

1. Les propositions atomiques sont des propositions.

Les propositions élémentaires (ou atomiques) sont des phrases simples dont on peut: déterminer dans un contexte (ou interprétation) si elles sont vraies ou fausses.

Pour nous les propositions élémentaires ou atomiques seront des lettres: a, b, c, ..A, B, C,….

2. Si X est une proposition, alors (¬X) ("non X", la négation de X) est une proposition.

3. Si X est une proposition alors (X) est une proposition.

4. Si X et Y sont des propositions, alors

(a) ($X \wedge Y$) ("X et Y ") conjonction

(b) ($X \vee Y$) ("X ou Y ", ou inclusif) disjonction

(c) ($X \Rightarrow Y$) ("si X alors Y ") implication

(d) ($X \Leftrightarrow Y$) ("X si et seulement si Y ") équivalence

sont des propositions.

Langage propositionnel (syntaxe)

En l’absence de parenthésage complet , les propriétés sont les suivantes:

- La négation est prioritaire sur La conjonction , la disjonction, l’implication et l’équivalence.

- La conjonction et la disjonction sont prioritaires sur l’implication et l’équivalence.

Langage propositionnel (syntaxe)

«proposition» --> «atome» |¬ «atome»
«proposition» -->(«proposition»)
«proposition» --> «atome» «connecteur» «atome»
«proposition» --> («proposition») «connecteur» «atome»
«proposition» --> «atome» «connecteur» («proposition»)
«proposition» --> («proposition») «connecteur»
(«proposition»)
«atome» --> a|b|c|..... |A|B|C|...

«connecteur» --> $\wedge|\vee|\Rightarrow|\Leftrightarrow$

Calcul des propositions (sémantique)

- La valeur de vérité d'une proposition du calcul propositionnel dépend de la valeur de vérité des atomes qui la composent.
- La valeur de vérité d'une proposition comportant *n atomes* distincts nécessite une table de vérité de 2^n lignes.
- On appel ***valuation*** d'un ensemble d'atomes, toute attribution de valeur de vérité à chacun de ces atomes.

 déduire inductivement la valuation de toute proposition.

Calcul des propositions (sémantique)

- Exemple: $v(((z \wedge y) \Rightarrow (x \vee y)))$

x	y	z	$z \wedge y$	$x \vee y$	$(z \wedge y) \Rightarrow (x \vee y)$
1	1	1	1	1	1
0	1	1	1	1	1
1	0	1	0	1	1
0	0	1	0	0	1
1	1	0	0	1	1
0	1	0	0	1	1
1	0	0	0	1	1
0	0	0	0	0	1

Calcul des propositions (sémantique)

- Une proposition P est ***satisfaisable*** si et seulement si il existe une valuation *v, v(P)=1*.
- Une proposition P est ***réfutable*** si et seulement si il existe une valuation *v, v(P)=0*.
- Une proposition P est ***tautologie*** si et seulement si il elle n'est pas réfutable: pour toute valuation *v, v(P)=1*.
- Une proposition P est une ***contradiction*** si et seulement si elle n'est pas satisfaisable: pour toute valuation *v, v(P)=0*.
- Deux propositions P et Q sont ***équivalentes sémantiquement*** si et seulement si pour toute valuation *v, v(P)=v(Q)*.

Calcul des propositions (sémantique) simplification de formules

$$\neg(\neg P) \Leftrightarrow P$$

$$P \vee (Q \vee R) \Leftrightarrow (P \vee Q) \vee R \quad et \quad P \wedge (Q \wedge R) \Leftrightarrow (P \wedge Q) \wedge R$$

$$P \vee P \Leftrightarrow P \quad et \quad P \wedge P \Leftrightarrow P$$

$$P \vee \neg P \Leftrightarrow 1 \quad et \quad P \wedge \neg P \Leftrightarrow 0$$

$$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R) \quad et \quad P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$$

$$P \vee (P \wedge Q) \Leftrightarrow P \quad et \quad P \wedge (P \vee Q) \Leftrightarrow P$$

$$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q \quad et \quad \neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$$

$$P \Rightarrow Q \Leftrightarrow \neg P \vee Q$$

$$P \Leftrightarrow Q \leftrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P) \leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$$

Calcul des propositions (sémantique)
Formes conjonctives et Formes disjonctives

- Pour toute proposition P, il existe une proposition équivalente sémantiquement à P *sous forme* ***conjonctive****:*

$$(a_1 \vee \ldots \vee a_n \vee \ldots \vee \neg a_{n+p}) \wedge (b_1 \vee \ldots \vee b_q \vee \ldots \vee \neg a_{q+r}) \wedge \ldots$$

où les a_i, b_j *...sont des atomes*

- Pareillement, pour toute proposition P, il existe une proposition équivalente sémantiquement à P *sous forme* ***disjonctive***:

$$(a_1 \wedge \ldots \wedge a_n \wedge \ldots \wedge \neg a_{n+p}) \vee (b_1 \wedge \ldots \wedge b_q \wedge \ldots \wedge \neg a_{q+r}) \vee \ldots$$

où les a_i, b_j *...sont des atomes*

Ensemble de formules démontrant une formule

- Pour démontrer qu'une formule est une conséquence logique d'un ensemble de formules, on se ramène à la démonstration de l'inconsistance d'un nouvel ensemble contenant les formules initiales et la négation du conséquent attendu.

- Pour montrer qu'un ensemble de formules est contradictoire, on peut utiliser la méthode de résolution de Robinson

Méthode de résolution de Robinson

- Permet de démontrer qu'un ensemble de formules est inconsistant.
- Permet de montrer qu'une formule est universellement valide en montrant que sa négation est contradictoire.

- Il faut premièrement transformer la ou les formules en ensemble de clauses,

Méthode de résolution de Robinson

- Littéraux et clauses:

1. Un littéral est une formule composée d'un atome ou de la négation d'un atome.
2. Une paire opposée de littéraux est composée d'un atome et de sa négation.
3. Une clause est une disjonction de littéraux. Une clause est une tautologie si et seulement si elle contient une paire opposée de littéraux.

Méthode de résolution de Robinson

- Résolvante d'une paire de clauses

Soit deux clauses *C1* et *C2*, elles forment une paire résoluble si et seulement si il existe un et un seul littéral ***t*** tel que ***t*** appartient à *C1* et ***¬t*** appartient à *C2*.

On appel résolvante de *C1* et *C2*, la clause notée *res(C1, C2)* obtenue en prenant la réunion des littéraux de *C1* et *C2* moins cette parties opposée.

Méthode de résolution de Robinson

- Exemple

Soient les quatre clauses P, Q, R et S définies par:

$$P = x \vee \neg y \vee \neg z \quad Q = \neg x \vee u \vee \neg z \quad R = \neg y \quad S = x \vee \neg u$$

La seule paire résoluble est (P,Q) de résolvante res (P,Q)= $u \vee \neg y \vee \neg z$

Méthode de résolution de Robinson

- Montrer que :

$$\{C_1 = a \vee b \vee c, \quad C_2 = \neg a \vee b \vee c, \quad C_3 = \neg b \vee c, \} \vdash \quad c$$

On cherche à démontrer que :

$$\{C_1, C_2, C_3, C_4 = \neg c\} \vdash \quad \textit{clause vide}$$

- Montrer que la résolution de cet ensemble de clauses contient la clause vide.

Méthode de résolution de Robinson

Parents	Résolvantes
C1C2	C5= $b \vee c$
C3C5	C6= *c*
C4C6	C7= clause vide

Méthode de résolution de Robinson

- Démontrer à l'aide de la méthode de résolution, le raisonnement suivant :

Si Jeaun n'a pas rencontré Pierre l'autre nuit c'est que Pierre est le meurtrier ou Jean un menteur. Si Pierre n'est pas le meurtrier, alors Jean n'a pas rencontré Pierre l'autre nuit. Le **crime a eu lieu après minuit**. Si le crime a eu lieu après minuit, alors Pierre est le meurtrier ou Jean n'est pas un menteur. Donc Pierre est le meurtrier.

1.2 Logique des Prédicats (premier ordre)

Logique des propositions/prédicats

- *Logique des propositions* étudies des énoncés qui sont vraies ou faux.
- *Logique des prédicats* a pour but de généraliser des propositions par l'introduction des variables.

Proposition: Oran est à l'ouest de l'Algérie.

x est une ville, *x* est à l'ouest de l'Algérie.

x: représente un objet (les choses dont on parle).

est à l'ouest de l'Algérie: prédicat (ce qu'on dit).

est à l'ouest(*x*, Algérie).

Logique des prédicats

- Un programme en Logique est un ensemble d'énoncés.
- Les énoncés sont des formules du calcul du premier ordre ne contenant pas de variables libres.
- Tout problème calculable peut être formulé dans ce langage.
- Le langage le plus représentatif est PROLOG(1971) basé sur deux mécanismes : *unification et résolution.*

Syntaxe

- La logique du premier ordre introduit deux *quantificateurs qui portent sur des variables :*

∀, le quantificateur *universel (« pour tout »)*

∃, le quantificateur *existentiel (« il existe »)*

Les énoncés universels: les variables représentants tous les objets du domaine, exemple: <(x, x+1).

Les énoncés existentiels: la variable sert à exprimer l'existence de quelque chose, exemple: 3x-22=8

Egal(moins (multipl(x,3),22),8) .

moins,multpl: fonctions

Syntaxe (exemple)

- Considérons par exemple l'énoncé suivant :

(1) tout chat est un animal

(2) Félix est un chat

donc Félix est un animal.

- On pourra le formaliser en logique du premier ordre de la manière suivante :

(1) $\forall x\ (chat(x) \rightarrow animal(x))$ *(tout x qui est un chat est un animal)*

(2) *chat(Félix),*

De (1) et (2), on peut déduire : *animal(Félix)*

Syntaxe

- Félix est une *constante,*
- *x une variable,*
- *chat(.) et animal(.) sont des prédicats unaires.*

- Pour désigner des entités (ou « objets »), on dispose des constantes (« entités connues ») et des variables (« entités inconnues »).

Syntaxe

- Un *langage du premier ordre L est formé :*
- Constantes : a, b, c,...
- Variables : x, y, z,...
- Fonctions : f, g, h, ...
- Relation(prédicat ou fonctions booléennes): P, Q, R
- A chaque prédicat p_i est associé un entier ≥ 0, son arité, qui fixe son nombre d'arguments.

Syntaxe: parenthèses et priorités

- Les ordres de priorité en l'absence de parenthèses sont les suivants:

$\exists$, $\forall$, $\neg$, *sont prioritaires devant* $\vee$, $\wedge$, *qui sont prioritaires devant* $\rightarrow$, $\leftrightarrow$

Syntaxe

- Toute constante est un terme.
- Toute variable est un terme.
- Si t1, t2, ..tn sont des termes et si f est une fonction n-aire, alors f(t1, t2, ..,tn) est un terme.
- Si t1, t2, ..., tn sont des termes et si P est un prédicat n-aires alors P(t1, t2, .., tn) est un atome.

formule bien formée – fbf

- Les *formules bien formées, fbf, construites sur le langage du premier ordre L peuvent être définies* inductivement de la manière suivante :
- *(base) la base est constituée par les formules atomiques (ou* ***atomes) : p(t1, ..., tn) où p est un*** prédicat n-aire et t1, ..., tn sont des termes ;
- *(règle de construction) si A et B sont des fbf et si x est une variable alors les expressions suivantes* sont des fbf :
- **¬A, (A ∧ B), [(A ∨ B), (A → B), (A ↔ B)]**
- **∀xA [∃xA]**

Syntaxe (tout simplement)

- Tout atome est une formule.
- Si P et Q sont deux formules alors :

Non P,

P ∧Q, P V Q,

P => Q, P <=> Q,

∀ x P(x), ∃x P(x) :

Sont des formules.

Arborescence syntaxique d'une fbf

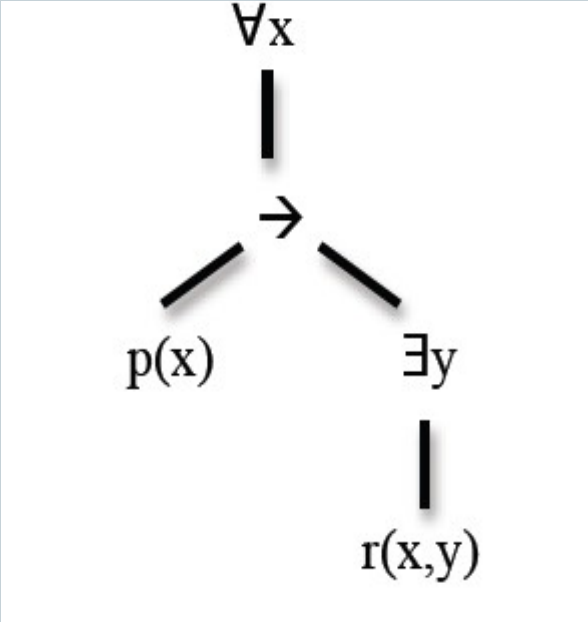
L'arborescence syntaxique d'une fbf **f (notation : ARBO(f)) se définit inductivement de la façon** suivante :

- si f est un atome, **ARBO(f) est une arborescence réduite à un sommet étiqueté par f**
- si f = ¬A, ARBO(f) est une arborescence dont la racine est étiquetée par ¬ et a un seul fils qui est la racine de ARBO(A)
- si f = Q xA, où Q est un quantificateur, ARBO(f) est une arborescence dont la racine est étiquetée par Qx et a un seul fils qui est la racine de ARBO(A)
- si f = (A op B), où op est un opérateur binaire, ARBO(f) est une arborescence dont la racine est étiquetée par op et qui a deux fils : le fils gauche est la racine de ARBO(A) et le fils droit est la racine de ARBO(B).

Arborescence syntaxique d'une fbf

Exemple : arborescence de la formule:

$\forall x(p(x) \rightarrow \exists y\ r(x,y))$



formule fermée

une variable x est une variable libre (resp. liée) d'une formule fbf si et seulement si x a une occurrence libre (resp. liée) dans la formule.

Une fbf est dite **fermée (ou close) lorsqu'elle n'a aucune variable libre** ***(et pas lorsque toutes les*** *variables sont liées, puisqu'une variable peut être libre et liée ...).*

Formules propres

Une fbf est dite propre lorsque l'ensemble des variables libres est disjoint de l'ensemble des variables liées.

Pour transformer une formule impropre en une formule propre, il suffit de standardiser les variables en les renommant ainsi:

dans toute sous-formule de liaison ayant une variable *x* liée de même nom qu'une variable libre, renommer *x*

Formule propre

- Exemple

Soit la formule impropre P définie par:

$$\forall x(\exists y\ P(x,y) \rightarrow \forall z\ Q(x,y,z) \wedge \forall y\ \exists x\ S(f(x),y))$$

Elle se transforme en la formule propre suivante:

$$\forall x(\exists v\ P(x,v) \rightarrow \forall z\ Q(x,y,z) \wedge \forall u\ \exists t\ S(f(t),u))$$

Exercice (1)

- Modéliser cette figure par la logique des prédicat (prédicat, objet):

Exercice (2)

- Formaliser les phrases suivantes:

A. Quiconque sait lire est instruit.
B. les dauphins ne sont pas instruits
C. Certains dauphins sont intelligents.

Substitution

- Substitution d'une variable par un terme:

Soit ø une formule propre, *x* une variable et *t* un terme, la substitution de *t* à *x*, ø[t/x] est la formule obtenue en remplaçant toutes les occurrences libres de *x* dans ø par *t*.

Soit ø une formule propre, *x* une variable et *t* un terme, *t* est substituable à *x*, [libre pour *x*] si et seulement si aucune occurrence libre de x dans ø ne devient une occurrence liée dans ø[t/x]. Dans le cas contraire il faut renommer soit les variables liées, soit les termes.

Sémantique

- La valeur de vérité d'une fbf (vrai ou faux) dépend de l'**interprétation du langage L sur lequel elle** est construite. On peut voir une interprétation de L comme :

un monde constitué d'entités (ou individus, ou objets, …) : l'ensemble des entités de ce monde s'appelle le *domaine de l'interprétation ;*

Interprétation

- Il faut un ensemble de vocabulaire utilisé

W(const; var; fonct; pred), il faut un domaine d'interprétation D.

Une interprétation I est définie comme suit:

- Pour chaque constante C de W un élément C_I dans D.
- Pour chaque symbole de fonct à n arguments, une fonction f_I dont les arguments sont interprétés dans D.
- Pour l'interprétation des variables, on assignes (affectation) une valeur à chaque variable (un élément de D).
- Pour chaque symbole de prédicat à n arguments une relation à n arguments dans D.

Interprétation

- exemple:

W(a, b, c, d; z; h; Q), D=(1, 2, 3, 4, rouge, bleu, vert)

a_I= 4; b_I=2; c_I=rouge; d_I=bleu

h_I=(« 1,1 »bleu; « 1,2 »rouge; «3,4 »bleu)

Q_I=(«rouge, rouge,1 »; «rouge, vert,2 »; «rouge, bleu,4 »).

I(Q(c, d, a))=Q(rouge, bleu,4)=V

Exercice

- On considère les symboles des prédicats suivants:

Personne(unaire); Machine(unaire); Autorisation(binaire); égal(binaire).

Les constantes: a, b,c, s1, s2, m1, m2

D=(toto, albert, lili, sun1, sun2, mac1, mac2)

a_I = toto, b_I = albert, c_I = lili, $s1_I$ = sun1, $s2_I$ = sun2,

$m1_I$ = mac1, $m2_I$ = mac2.

$Personne_I$= (toto, albert, lili),

$Machine_I$= (sun1, sun2, mac1, mac2),

$Autorisation_I$= (« toto,sun1 », « toto, mac1 », « lili, mac1 », « albert,mac2 »),

$égal_I$= (« toto, toto », « mac1, mac1 », …).

Exercice

- Donner une valeur de vérité aux formules suivantes:

1) Personne(s1)
2) Autorisation (s2, m1) ou machine (s2)
3) ∃x Autorisation(x, m2)
4) ∀ x (Personne(x) --> ∃y Autorisation (x,y))
5) ∀ m (Machine(m) --> ∃P Autorisation (p,m))
6) ∀ x(∃y Autorisation (x,y))

- Modifier l'interprétation des prédicats pour rendre l'expression 5 vraie.

Exercice

- Formaliser les expression suivantes:

1) Il ne peut y avoir plus d'une personne autorisée par machine.
2) Pour toute machine il y'a au moins une personne autorisée à l'employer.
3) Aucune personne n'est autorisé à employer toutes les machines.

Normalisation des formules

- Forme Prénex: Mettre les quantificateurs devant la formule
- Skolémisation: permet d'éliminer le quantificateur ∃
- FNC
- Forme clausales: permet d'éliminer le quantificateur ∀

Forme Prénex

- Méthode 1:

Se débarrasser de $\leftrightarrow$ $\vee$ $\wedge$ en utilisant:

$$A \vee B = \neg A \rightarrow B$$

$$A \wedge B = \neg(A \rightarrow \neg B)$$

$$A \leftrightarrow B = \neg((A \rightarrow B) \rightarrow \neg(B \rightarrow A))$$

Changement de certains variables liées de manière à n'avoir plus d'une variable quantifier deux fois.

Forme Prénex

- Faire remonter les quantificateurs:

$$C \rightarrow \forall x A(x) \Leftrightarrow \forall x (C \rightarrow A(x))$$

$$C \rightarrow \exists x A(x) \Leftrightarrow \exists x (C \rightarrow A(x))$$

$$\forall x A(x) \rightarrow C \Leftrightarrow \exists x (A(x) \rightarrow C)$$

$$\exists x A(x) \rightarrow C \Leftrightarrow \forall x (A(x) \rightarrow C)$$

$$\forall x A(x) \rightarrow \exists x B(x) \Leftrightarrow \exists x (A(x) \rightarrow B(x))$$

Forme Prénex

- Méthode 2:

Se débarrasser de$\leftrightarrow$ $\rightarrow$ en utilisant:

$$A \rightarrow B = \neg A \vee B$$

$$A \leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$$

Changement de certains variables liées de manière à n'avoir plus d'une variable quantifier deux fois.

Forme Prénex

- Faire remonter les quantificateurs:

$$C \wedge \vee \forall x A(x) \Leftrightarrow \forall x (C \wedge \vee A(x))$$

$$C \wedge \vee \exists x A(x) \Leftrightarrow \exists x (C \wedge \vee A(x))$$

$$\forall x A(x) \wedge \vee C \Leftrightarrow \forall x (A(x) \wedge \vee C)$$

$$\exists x A(x) \wedge \vee C \Leftrightarrow \exists x (A(x) \wedge \vee C)$$

$$\forall x A(x) \wedge \vee \forall x B(x) \Leftrightarrow \forall x (A(x) \wedge \vee B(x))$$

$$\exists x A(x) \wedge \vee \exists x B(x) \Leftrightarrow \exists x (A(x) \wedge \vee B(x))$$

Forme clausale

- exemple: mettre sous forme clausale

$$\forall x A(x) \rightarrow (\exists t B(t) \vee \exists t c(t))$$

Résolution d'un ensemble de clauses

- exemple: démontrer que l'ensemble de clauses est contradictoire:

$$C1 = S(a,y) \vee T(x,y), C2 = \neg T(z,t) \vee U(a,z),$$
$$C3 = \neg S(a,b), C4 = \neg U(u,v)$$