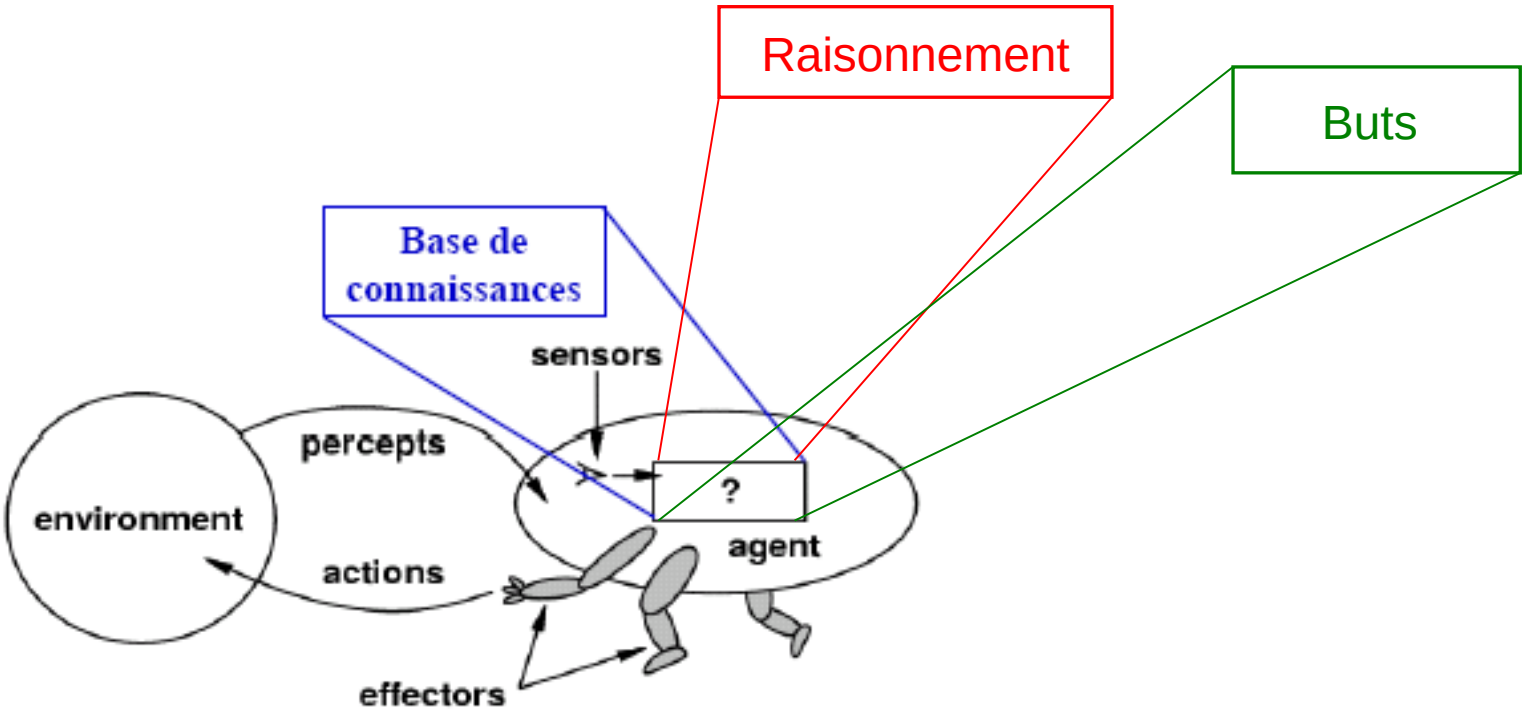
Intelligence Artificielle
Connaissances/Logique

Représentation des connaissances

Agents comme systèmes logiques

Agent comme système logique



- **Questions:**

– comment représenter des connaissances du monde réel?

– comment raisonner sur ces connaissances ?

– quelles représentations sont appropriées pour traiter du monde réel?

Connaissance

- Connaissance => capacité à mobiliser des informations pour agir
- Le passage de INFORMATION à CONNAISSANCE est lié à l'expérience de l'action => pas de frontière parfaitement définie
- Définition : Connaissance = Information (donnée) qui influence un processus.
- Pas de classement universel des différents types de connaissances

Différence entre donnée, information, connaissance.
- la donnée transporte l'information : ce sont des signaux non interprétés ;
- l'information est une interprétation de la donnée ;
- la connaissance utilise l'information dans le cadre d'actions, dans un but précis. Les actions peuvent être la prise de décisions, la création de nouvelles informations, etc...

La Représentation

Problème central en IA

- L'IA cherche à reproduire *l'intelligence humaine*
- L'intelligence humaine est basée sur la connaissance

Comment représenter la connaissance en informatique ?

Pour comprendre une scène ou une histoire, il faut trouver une correspondance entre les éléments perçus et les choses connus.

Les perceptions brutes (les phénomènes) sont compris par associations aux catégories mentales (les concepts).

Cela exige une représentation.

DONC IL FAUT MODELISER

• Comment définir (informatique) ce qu'est la connaissance ?
• Comment la traduire en structures manipulables par un logiciel ?

Types de la Représentation

➢ Représentation déclarative et Représentation procédurale

• Représentation déclarative

Représente ce que l'on sait dans une collection statique de faits et de règles d'inférence.

- Représentation déclarative → *QUOI*

• Représentation procédurale

représente la connaissance comme une collection de procédures qui indiquent comment utiliser la connaissance.

- Représentation procédurale → *COMMENT*

Avantages

- Avantages des représentations déclaratives

– chaque fait n'est stocké qu'une seule fois,
– Il est facile d'ajouter de nouveaux faits.

- Avantages des représentations procédurales

– facilite la représentation de la connaissance opératoire (qui dit comment faire les choses),
– permet d'accommoder la connaissance difficile à exprimer de façon simple dans le mode déclaratif,
– facilite la représentation de la connaissance *heuristique*.

Représentation structurée

Considérer la connaissance nécessaire pour lire le texte suivant:

« Moha décida d'aller rendre visite à Ali; il se rendit chez lui, mais il vit que toutes les lumières étaient éteintes, à la place il alla au cinéma"

et pour répondre aux questions suivantes:

est-ce que Moha a vu Ali?
de qui la maison a-t-elle les lumières éteintes?
qui est allé au cinéma?

Il faut avoir:
des structures de connaissances: “structures de données” dans lesquelles la connaissance d'un domaine peut être stockée, sous forme d’objets et relations entre objets:

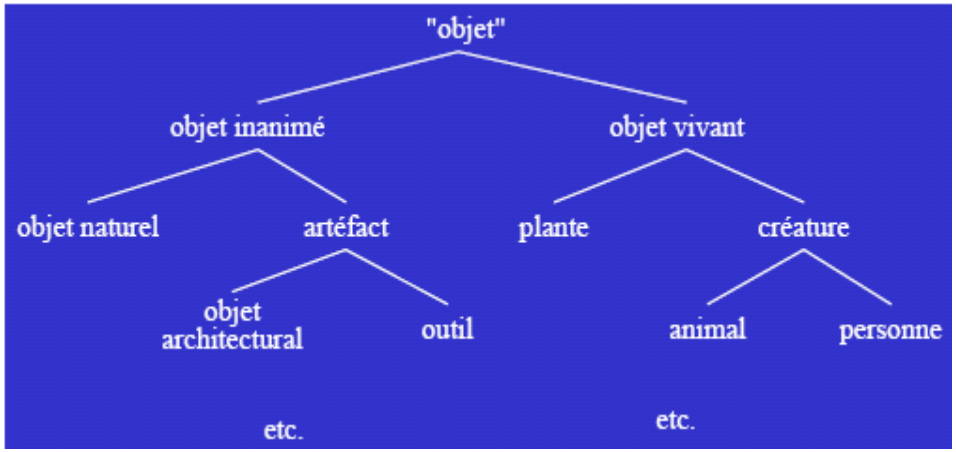
Réseaux sémantiques et *Frames* (AIMA, chap. 10.6).

Types de Relations:

- Relation: Est-un (is-a)

– relations entre objets dans une taxonomie hiérarchique

• exemple



Hiérarchies et héritage

• Les hiérarchies permettent d'utiliser l'héritage de propriétés,réduisant significativement le nombre de faits explicitement représentés:
– toutes les propriétés de *Créature* sont aussi les propriétés de *Chien* (à travers *Animal*),

• elles complètent la LPO en permettant des inférences très rapides par parcours dans le graphe des hiérarchies.

- Relation : Partie-de

• relations entre objets qui sont constitués d'un ensemble de composantes:
– main *Partie-de* corps
– doigt *Partie-de* main
– phalange *Partie-de* doigt
Héritage en LPO soit:
– Est-un(Marcus, Homme)
– Est-un(Marcus, Pompéien)
– Est-un(Pompéien, Romain)
• on peut ajouter la règle:

$$\forall x \forall y \forall z \ \text{Est-un}(x,y) \wedge \text{Est-un}(y,z) \Rightarrow \text{Est-un}(x,z)$$

et ainsi prouver que Marcus est un Romain

Réseaux sémantiques

Idées principales des réseaux sémantiques:

1. La signification d'un concept vient de ses relations avec d'autres concepts et
2. L'information est stockée en interconnectant des nœuds par des arcs étiquetés.

Définition IA

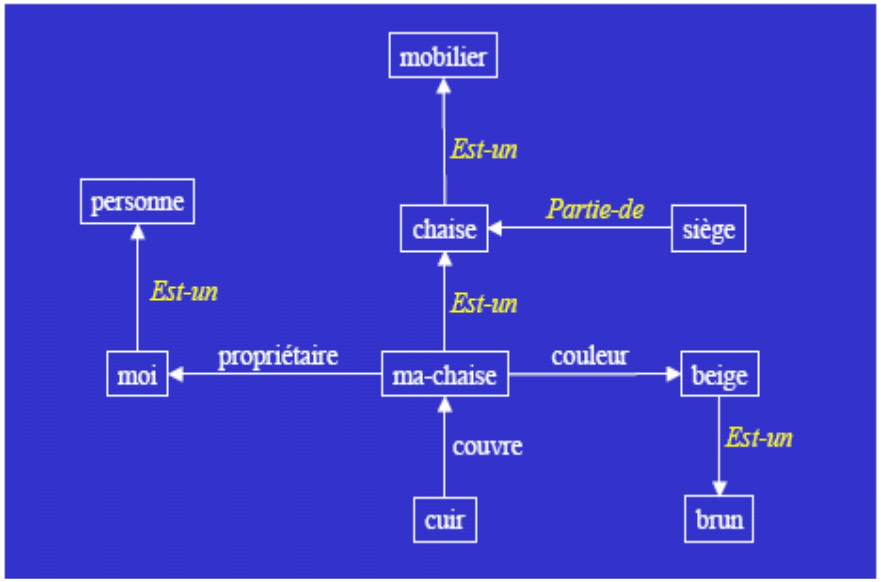
- Un réseau sémantique est un graphe orienté et étiqueté (un multi-graphe en fait car rien n'empêche deux noeuds d'être reliés par plusieurs arcs).

- Une « sémantique » (au sens de la logique) est associée par le biais des relations.

- Réseau = conjonction de formules logiques associées à chacun des arcs

Réseaux sémantiques: Exemple

- Représentation de "ma chaise"



Inférence dans les réseaux sémantiques

Mécanisme de base: suivre les arcs entre les noeuds.

2 méthodes pour réaliser l'inférence:

1. par recherche d'intersection: on propage l'activation à partir de 2 noeuds et en trouvant les intersections des activations on trouve les relations entre objets.

2. par héritage: les relations "Est-un" et "Partie-de" permettent de suivre les liens d'héritage dans une taxonomie hiérarchique. L'héritage permet aussi de faire du raisonnement par défaut.

Frames

Représentation proposée par M. Minsky, T. Winograd, R. Schank

Frames

– “Paquets d'informations” représentant des entités et leurs instances,

– les *Frames* sont une représentation des faits essentiels d'une entité structurelle quelconque,

– utiles pour:
• classifier de nouvelles instances d'entités connues (objets/événements/places/tâches),
• représenter les attributs des instances,
• inférer l'existence et les propriétés des entités et des instances,

– structure dynamique: adjonction, modification, suppression.

Structure de frames

• Exemple:

[TWEETY Est-un CANARI avec
couleur JAUNE
sexe MALE
age 2
santé EXCELLENT
propriétaire ANNE-SOPHIE
]

• en LPO:

– canari (TWEETY)
– couleur (TWEETY,JAUNE)
– sexe (TWEETY,MALE)
– …

• forme générale:

[< nom frame > Est-un < type >
< nom slot > < valeur slot >
< nom slot > < valeur slot >
…]

Structure de frames

Attributs - Facettes – Valeurs

• aux attributs ("slots") sont associées des valeurs, éventuellement modifiées par des facettes,

< nom slot > < facette > < valeur slot >

• facette, information exprimant:
– modalités descriptives ou comportementales de l'attribut et/ou de sa valeur:

• valeurs par défaut, exceptions, info. incomplètes/redondantes, type de données, etc.
– différents points de vue sur l'attribut,
– comment utiliser l'information représentée par l'attribut,
– etc. …

• attachement procédural (démon): facette peut être une procédure,
– permet de spécifier un comportement particulier:

- si-besoin: méthode de calcul de la valeur de l'attribut,
- si-ajout: que faire si la valeur est ajoutée (pour un attribut multi-valué),
- si-élimination: que faire si la valeur est supprimée.

Comparaison

• systèmes de **frames** et **réseaux sémantiques** comparés à la **logique du premier ordre**:

– plus efficaces (utilisation de procédures spéciales),

– plus faciles à interpréter,

– sémantique parfois "floue",

– plus facile à implanter,

– difficultés avec l'héritage multiple de propriétés incompatibles,

– moins expressifs (problèmes avec la négation, la disjonction et la quantification),

– plus expressifs (héritage avec exceptions, attachements procéduraux).

D'autres types de représentations

1. Extension de la Logique Classique (à 3 valeurs: pour exprimer vrai, faux et "ne sait pas", Floue, Modale, Temporelle, …)

2. Logique du 2ème ordre: pour exprimer la quantification aussi bien sur des objets individuels que sur des classes d'objets, (ou sur les relations)

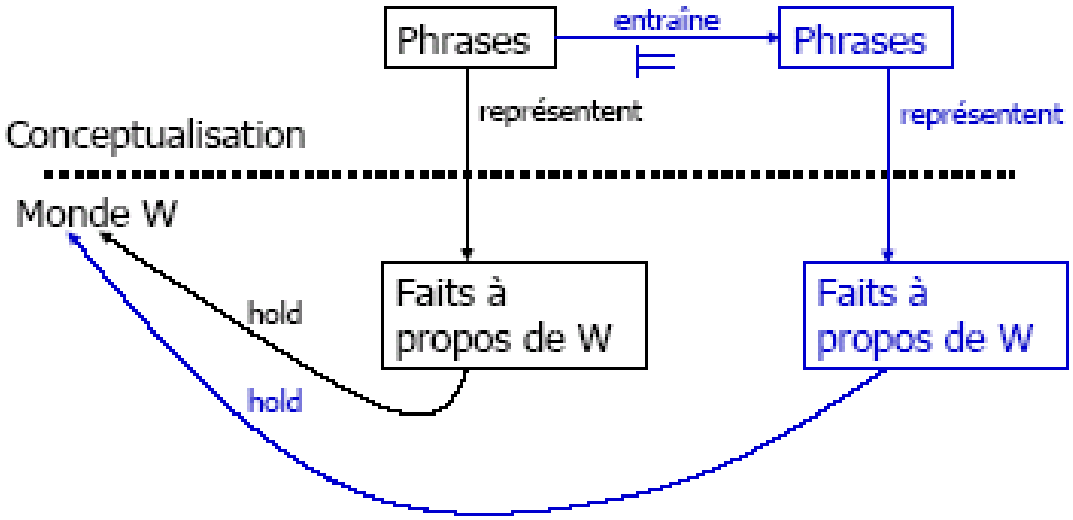
3. Logiques avec états: la valeur de vérité d'une expression dépend de l'état du monde,

4. Calcul de situations: pour exprimer des changements dans l'environnement:
– À(Agent, location) → À(Agent, location, situation)

Etc...

LOGIQUE

Relation Monde-Représentation



La logique en général

• C'est un formalisme de représentation des connaissances de type "langage" applicable dans de nombreux domaines,

• les logiques sont des langages formels pour représenter l'information et permettre d'en tirer des conclusions,

• 2 notions importantes:
syntaxe: définit les expressions possibles du langage (i.e suite de mots et de symboles formant une phrase),
sémantique: définit la signification d'une expression (sa valeur de vérité) dans un "monde" donné,

• offre un mécanisme de manipulation et de raisonnement sur l'information sémantique contenue dans une phrase en ne manipulant que les symboles qui la constituent,

- système informatique de raisonnement par manipulation de symboles

Exemples de logiques

Logique propositionnelle

- une suite de symboles séparés par des conjonctions (et), des disjonctions (ou) ou des négations (non)
- exemples:

"Socrate est un homme" → HommeSocrate

"Platon est un homme" → HommePlaton

"Platon et Socrate sont des hommes"
→ HommeSocrate ∧ HommePlaton

Logique des prédicats du 1er ordre

- une suite de symboles, de variables et de relations avec des quantificateurs universels et existentiels
- exemples:
  - Homme(Socrate) $\forall x$ Homme(x) $\Rightarrow$ Mortel(x)

Autres logiques

Logique de croyance
- Croire(Jean, Père(Zeus, Chronos))

Logique temporelle
- permet de raisonner à propos du temps

Logique non-monotone
- permet aux valeurs de vérité associées aux faits de changer

Logique floue
- accompagne les faits de valeurs de vraisemblance

Logique des prédicats

Motivations

La logique des propositions est bien trop limitée pour décrire des situations réelles. Elle ne peut pas par exemple exprimer le fait qu'une propriété est vraie pour un ensemble d'objets ou d'individus : « l'homme est mortel ».

Syntaxe

Language

Définition1 (alphabet). L'alphabet de la logique des prédicats est constitué de :

- un ensemble dénombrable de symboles de prédicats à 0, 1, ou plusieurs arguments, notés p, q, r, ..., homme, mortel, père, ...

- un ensemble dénombrable de variables d'objets (ou variables d'individu), notées x, y, z, x_1, x_2, ...

- un ensemble dénombrable de fonctions à 0, 1, ou plusieurs arguments, notées f, g, ... , père-de, ...

- les quantificateurs $\sqsupset$ et $\curlywedge$

- les connecteurs ainsi que les parenthèses de la logique propositionnelle

Les fonctions à 0 arguments sont appelées constantes (souvent notées a, b, ..., Socrate, ...) définies dans un domaine D.

Les prédicats à 0 arguments ne sont rien d'autre que des variables propositionnelles.

Définition2 (terme). L'ensemble des termes est le plus petit ensemble de mots construits sur l'alphabet de la logique des prédicats tel que :

- toute variable est un terme

- $f(t_1,...,t_n)$ est un terme si f est une fonction à n arguments et $t_1,...,t_n$ sont des termes

Définition3 (formule atomique). Si p est un prédicat à n arguments et $t_1,...,t_n$ sont des termes alors p(t1,...,tn) est une formule atomique.

L'ensemble des phrases (ou formules bien formées) de la logique des prédicats est alors défini de la même manière qu'en logique propositionnelle, en rajoutant une clause pour les quantificateurs :

(Q x A) est une formule si Q est un quantificateur, x une variable et A une formule
Une expression est un terme ou une formule.

Définition plus explicite des formules
L'ensemble des phrase (ou formules bien formées) de la logique des prédicats est le plus petit ensemble de mots construits sur l'alphabet tel que

- si p est un prédicat à n arguments et $t_1,...,t_n$ sont des termes alors $p(t_1,...,t_n)$ est une formule (aussi appelée formule atomique)
- (Q x A) est une formule si A est une formule, Q un quantificateur et x une variable
- VRAI et FAUX sont des formules atomiques
- (~A) est une formule si A est une formule
- (A & B), (A v B), (A ⊔ B) sont des formules

***Traduction* Exemples :**

Prédicats utilisés : champignon(x), violet (x) et empoisonne (x), egal(x,y).

Tous les champignons violets sont empoisonnés.
∀ x, (champignon(x) ∧ violet(x)) -> empoisonne(x)

Aucun champignon violet n'est empoisonné.
∀ x, (champignon(x) ∧ violet(x)) -> ~empoisonne(x)

Tout champignon est violet ou empoisonné.
∀ x, champignon(x) -> (violet(x) ∨ empoisonne(x))

Tout champignon est soit violet ou empoisonné mais pas les deux (ou exclusif).
∀ x, champignon(x) -> ((violet(x) ∧ ~empoisonne(x)) ∨ (~violet(x) ∧ empoisonne(x)))

Tous les champignons violets sauf un sont empoisonnés
∧ x, (violet(x) ∧ champignon(x) ∧ ~empoisonne(x)) ∧ (∀ y, (violet(y) ∧ champignon(y) ∧ ~égal(x,y)) -> empoisonne(y))

Non-unicité de la traduction
Souvent, il y a plusieurs manières de traduire :
Exemple :
Chacun aime sa mère :

∀ x, y, mère(y,x) -> aime(x,y) **ici, mère est prédicat**

∀ x, aime(x, mère(x)) **ici, mère est fonction**

Quelle est la bonne représentation ? Les deux ! C'est l'application qui pourrait en rendre une plus appropriée que l'autre.

Logique des prédicats : formes normales

Forme normale prénexe

Définition (forme normale prénexe).

Une formule A est en forme normale prénexe si elle est de la forme $Qx_1 \ldots Qx_n$ B , où B est une formule sans quantificateurs, Q un quantificateur et les x_i sont des variables.

La suite des quantificateurs est appelée préfixe, et B est appelée matrice.

Exemples

La formule p(x) & ~p(y) est en forme normale prénexe.

La formule ∀ x ∃y ((p(x) & q(x,y)) v r(z)) est en forme normale prénexe.
∀ x ∃ y est le préfixe, et (p(x) & q(x,y)) v r(z) est la matrice.

La formule ∃ x ((p(x) -> ∀ x p(x)) n'est pas en forme normale prénexe.

Th: Toute formule (phrase) admet une forme prénexe

Algorithme de mise en forme normale prénexe

entrée : une formule A

sortie : une formule en forme normale prénexe

début
éliminer -> , <->;
appliquer autant que possible les équivalences suivantes, dans n'importe quel ordre (en remplaçant le membre gauche par le membre droit) :

~~A <-> A
~(A v B) <-> ~A & ~B
~(A & B) <-> ~A v ~B
~(∧ x A) <-> ∨ x ~A
~(∨ x A) <-> ∧ x ~A

tant que il existe une sous-formule Q x B telle que x apparaît en dehors de B dans A faire
remplacer Q x B par Q y (B), où y est une nouvelle variable si x existe déjà (pour ne pas confondre les variables) ;
fin tant que;
appliquer autant que possible les équivalences suivantes, dans n'importe quel ordre (en remplaçant le membre gauche par le membre droit) :

(Q x A) & B <-> Q x (A & B)
(Q x A) v B <-> Q x (A v B)
A & (Q x B) <-> Q x (A & B)
A v (Q x B) <-> Q x (A v B)

fin

Exemple

Soit la formule $\forall x\ (p(x) \rightarrow \exists x\ p(x))$
Éliminer -> : $\forall x\ (\neg p(x) \vee \exists x\ p(x))$
Renommer $\exists x\ p(x)$ en $\exists y\ p(y)$: $\forall x\ (\neg p(x) \vee \exists y\ p(y))$
Sortir $\exists y$: $\forall x\ \exists y\ (\neg p(x) \vee p(y))$
N.B. : le renommage est essentiel ici ;
par ex. $\forall x\ (\neg p(x) \vee \exists x\ p(x))$ n'est pas équivalent à $\forall x\ \exists x\ (\neg p(x) \vee p(x))$

Forme normale de Skolem

Définition (forme normale de Skolem).
Une formule est en forme normale de Skolem si elle est en forme normale prénexe et ne contient pas de quantificateur existentiel.

Remplacer toute variable de $\exists$ par une fonction dont les arguments sont les variables de $\forall$ qui précèdent la variable en question.

Exemple.

Soit la formule $\forall x\ \exists y\ p(x,y)$:

remplacer y par la fonction f(x) : $\forall x\ p(x,f(x))$ (si pas de variable avec $\forall$ alors f est une constante)

Forme normale Conjonctive (forme standard)

On peut remarquer que les variables restantes dans la forme skolem sont précédées par $\forall$ qu'on peut alors l'omettre.
La forme Normale conjonctive se construit de la même manière que la logique propositionnelle à partir d'un skolem.

Logique des prédicats : *Unification*

C'est le processus qui rend 2 expressions identiques.

En logique propositionnelle, 2 expressions sont identiques ssi elles sont syntaxiquement identiques.
La présence de variables en logique des prédicats complique ce fait :

Humain(x) = Humain(socrate) ssi x=socrate

Parfois, il faut substituer une fonction à une variable :
Humain(x) -> Mortel(x)
Humain(Père-de(platon))
Instanciation de x par Père-de(platon) : **Humain(Père-de(platon)) -> Mortel(Père-de(platon))**

Une substitution q est un ensemble fini de couples (t_i,V_i) où chaque V_i est une variable, chaque t_i est différent de V_i et où aucun couple d'éléments de l'ensemble n'a la même variable V_i.
Ce couple est écrit: t_i/V_i **ou** $\{ V_i \sqsupset t_i \}$

Exemple :
$q_1 = \{f(Z)/X,t/Y\}$
$q_2 = \{a/X,f(g(a))/Y,g(a)/Z\}$

Définition d'instance
Soit q une substitution, E une expression.

L'expression obtenue à partir de E en remplaçant simultanément toute occurrence de V_i par t_i pour tout i est **notée** E.q et est **appelée instance de** E.
E={P(X,a,f(Z))} alors E.q = {P(a,a,f(g(a)))}

Composition de substitutions

Soit $q = \{t_1/X_1, t_2/X_2, ..., t_n/X_n\}$ et $s = \{u_1/Y_1, u_2/Y_2, ..., u_m/Y_m\}$

la composition de q **et** s **(notée** $s°q$**)** est la substitution obtenue à partir de

$\{t_1.s/X_1, t_2.s/X_2, ..., t_n.s/X_n, u_1/Y_1, u_2/Y_2, ..., u_m/Y_m\}$

en éliminant les couples :

a) u_i/Y_i si $Y_i \in \{X_1, X_2, ..., X_n\}$

b) $t_j.s/X_j$ si $X_j = t_j.s$

Définition de l'unifieur :

Une substitution q est un unifieur de l'ensemble W des expressions $\{E_1, ..., E_k\}$

<==> $E_1.q = E_2.q = ... = E_k.q$

W est dit unifiable

Définition de l'unifieur le plus général ou upg :

Un unifieur s de $W = \{E_1, ..., E_k\}$ est dit **upg** (unifieur le plus général)
si et seulement si pour tout unifieur q **de W, il existe** d une substitution **telle que :** $q = d°s$

$W = \{P(X,Y), P(f(T),Z)\}$
$s_1 = \{f(T)/X, Z/Y\}$ et $s_2 = \{f(T)/X, Y/Z\}$ **upgs**
$q = \{f(a)/X, g(g(a))/Y, g(g(a))/Z, a/T\}$ **unifieur**
d_1 tel que $q = d_1°s1$ --------- ▭ $d_1 = \{a/T, g(g(a))/Z\}$
d_2 tel que $q = d_2°s2$ --------- ▭ $d_2 = \{a/T, g(g(a))/Y\}$

L'algorithme d'unification consiste en la recherche d'un upg d'un ensemble d'expressions.

l'algorithme d'unification :

Soit W l'ensemble fini à unifier

Etape 1 : $k = 0$; $W_k = W$; $s_k = e$ (identité)

Etape 2 : SI W_k est un singleton

ALORS s_k upg de W

SINON trouver D_k l'ensemble de discordance de W_k ;

Etape 3 : SI il existe des éléments V_k et t_k de D_k tels que :

_ V_k soit une variable

_ t_k soit un terme ne contenant pas V_k

ALORS aller à l'étape 4

SINON W non unifiable

Etape 4 : $s_{k+1} = \{t_k/V_k\} \circ s_k$;

$W_{k+1} = W_k.\{t_k/V_k\}$

Etape 5 : $k = k+1$;

Aller à l'étape 2

SI W est ensemble fini non vide unifiable alors l'algorithme s'arrête toujours à l'étape 2 et s_k est upg

Exemple :

W = {P(a,X,f(g(Y))), P(Z,f(Z),f(U))}

1. k = 0 ; W_0 = W ; s_0 = e ;
2. D_0 = {a,Z} avec V_0=Z variable et t_0=a terme ne contenant pas Z ;
3. s_1 = {a/Z}°s_0 = {a/Z}

W_1 = W_0.{a/Z} = {P(a,X,f(g(Y))), P(a,f(a),f(U))}

4. D_1 = {X,f(a)}, V_1 = X, t_1 = f(a) terme ne contenant pas X ;
5. s_2 = {f(a)/X}°{a/Z} = {f(a)/X, a/Z}

W_2 = W_1.{f(a)/X} = {P(a,f(a),f(g(Y))), P(a,f(a),f(U))}

6. D_2 = {g(Y),U}, V_2 = U, t_2 = g(Y) terme ne contenant pas U ;
7. s_3 = {g(Y)/U}°s_2 = {g(Y)/U, f(a)/X, a/Z}

W_3 = W_2.{g(Y)/U } = {P(a,f(a),f(g(Y)))}

8. W_3 est un singleton ==> s_3 = {g(Y)/U, f(a)/X, a/Z} est l'upg de W

donc W est unifiable.

EXERCICE

Donner les upgs (s'ils existent) des ensembles d'expressions suivants :

a) W = {P(T,T), P(f(V),V)}
b) W = {P(a,T), Q(X,Y)}
c) W = {P(f(X),Y,X), P(Z,X,g(T))}
d) W = {P(f(X),X), P(Y,g(Y))}

Principe de la résolution

Clause résolvante
(Li : littéral, Ci: Clause)

Si C1 et C2 sont 2 clauses et si L1 = ⫟L2 et L1 est dans C1 et L2 est dans C2
C est la disjonction des clauses restantes après suppression des littéraux L1 et L2.
Elle est appelée **clause résolvante** et ou **résolvant** de C1 et de C2.
L1 et L2 sont les **littéraux résolus**.

Exemples :
Soient C1 et C2 les deux clauses suivantes :
C1 : E1 ⫟ E2
C2 : ⫟E2 ⫟ E3
Le résolvant de **C1** et de **C2** est **C : E1 ⫟ E3**

Soient C1 et C2 les deux clauses suivantes :
C1 : P
C2 : ⫟P
Le résolvant de **C1** et de **C2** est **C** : Ø **la clause vide**

La clause « Faux » est notée Ø c'est la **clause vide**

Le résolvant C de deux clauses C1 et C2 est une conséquence logique de C1 et de C2

On appelle **déduction** (ou résolution) d'une clause C à partir d'un ensemble de clauses S
= séquence finie R1, R2,... Rn = C de clauses telle que chaque Ri est:
- soit une clause de S
- soit un résolvant de clauses le précédant

On appelle **réfutation** la déduction de la clause vide Ø à partir de S

Montrer qu'une formule bien formée **F est valide** :
- C'est équivalent à montrer que ⊬**F est inconsistante**
- C'est aussi équivalent à monter que $\mathbf{S}_{\vdash F}$ **insatisfiable** ($S_{\vdash F}$: ensemble de clauses de ⊬F)
- C'est aussi équivalent à monter qu'**il existe une déduction de la clause vide** Ø à partir de $\mathbf{S}_{\vdash F}$

Algorithme de résolution
Début
Ecrire la négation de F ;
Mettre ⊬F sous forme d'un ensemble de clauses ;

Tant que la clause vide n'est pas rencontrée et qu'il existe des paires réductibles **faire**
Début
Chercher des clauses résolvantes ;
Ajouter ce résultat à la liste des clauses ;
Fin Tant que ;

Si on trouve la clause vide **alors** F est valide
sinon F est invalide
Finsi ;

Fin ;

Exemple

Soit $S_{\neg F} = \{\neg P \vdash \neg Q \vdash R,\ \neg R,\ P,\ \neg T \vdash Q,\ T\}$
C1 C2 C3 C4 C5

$S_{\neg F}$ est l'ensemble des clauses d'une fbf $\neg F$.

Montrons que cet ensemble est insatisfiable

C1 = $\neg P \vdash \neg Q \vdash R$ donc (par résolution) C6 = $\neg P \vdash \neg Q$
C2 = $\neg R$

C6 = $\neg P \vdash \neg Q$ donc C7 = $\neg Q$
C3 = P

C7 = $\neg Q$ donc C8 = $\neg T$
C4 = $\neg T \vdash Q$

C8 = $\neg T$ donc C9 = Ø
C5 = T

A partir de l'ensemble des clauses de la fbf $\neg F$ on a déduit la clause vide donc

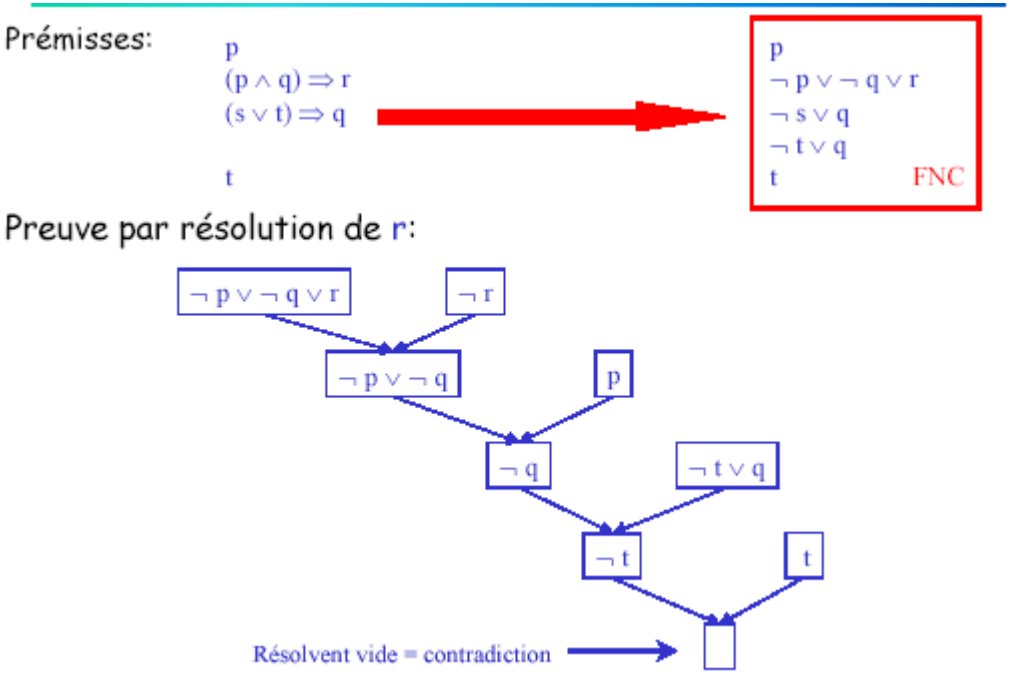
on peut conclure que **$\neg F$ est insatisfiable** et donc que **F est valide**.

Application du Principe de résolution:

Pour dériver une formule F à partir d'un ensemble de formules S, il suffit de montrer l'inconsistance de S U {¬F}

(On dit que S démontre F ou F est déduite de S)

Exemple : on veut montrer que r dérive de S={p, (p∧q) ⊏ r, (s∨t) ⊏ q, t}



Principe de la résolution: CAS DE LPO

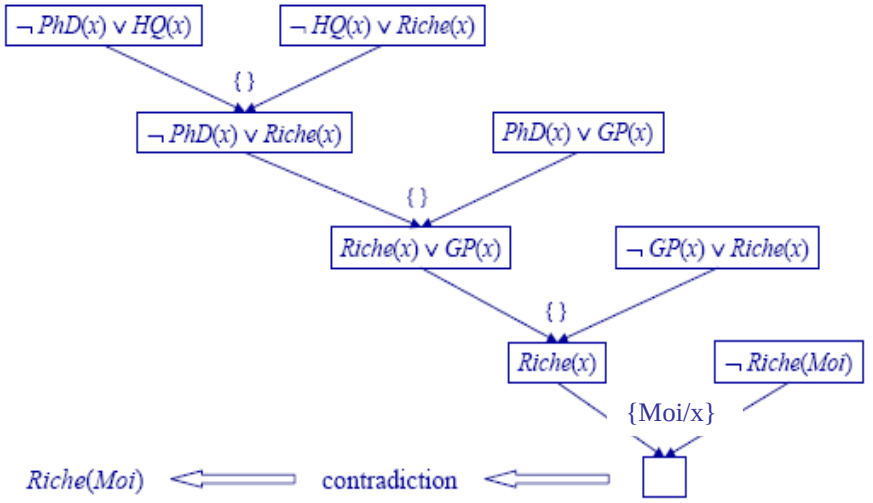
On calcule la résolvante après avoir unifié les littéraux mis en jeux.

Exemple: pour prouver Riche(Moi), on ajouter ⊣Riche(Moi) à la BC en forme FNC
Soit la BC suivante (déjà en FNC):

1. ⊣PhD(x) ⊣ HautementQualifié(x)
2. PhD(x) ⊣ GainsPrécoces(x)
3. ⊣HautementQualifié(x) ⊣ Riche(x)
4. ⊣GainsPrécoces(x) ⊣ Riche(x)

Clause rajoutée :

5. ⊣Riche(Moi)



APPLICATIONS

Bases du langage Prolog

Les symboles du langage sont les constantes (a,b,c,…), les variables (X,Y,Z,…) à quantification universelle, les fonctions (f,g,h,…), les prédicats (p,q,r,…) et les opérateurs { :- , . ?- }.

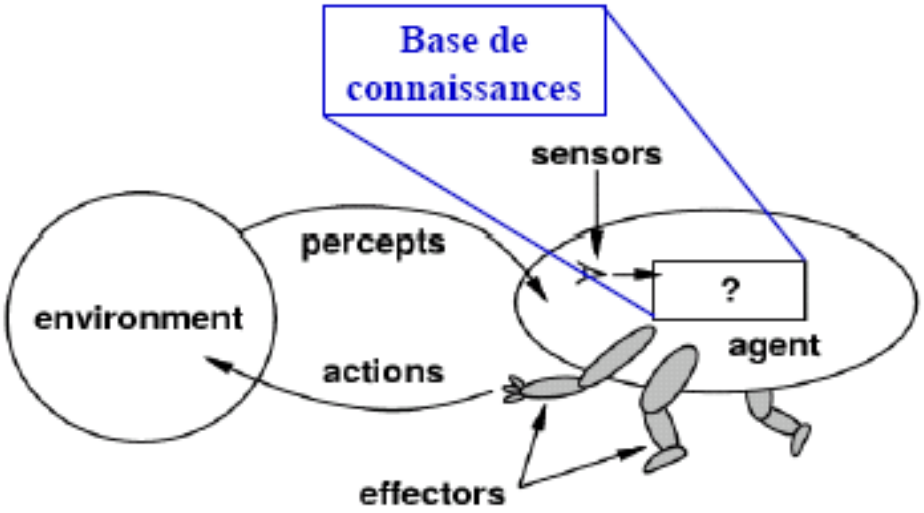
Un énoncé est une clause de Horn stricte, c'est-à-dire qui comporte un littéral positif. La syntaxe de Prolog impose que, dans tous les énoncés, le littéral positif soit séparé des littéraux négatifs par le symbole ":- ".

Au cas où l'énoncé ne comporte pas de littéraux négatifs, le symbole :- est omis.

La virgule sépare les littéraux négatifs dans les clauses ($\neg$)
:- est interprété comme l'implication de la droite vers la gauche
Ainsi, l'énoncé Prolog :

above(X,Z) :- on(X,Y), above(Y,Z). correspond aux fbfs :
above(X,Z) V -on(X,Y) V -above(Y,Z) ou
above(X,Z) $\leftarrow$ (on(X,Y) & above(Y,Z)).
Le point termine les énoncés.
Une requête est représentée par :
?- above(X, table), above(X,b).
PROLOG

Application : Agent basé sur la connaissance



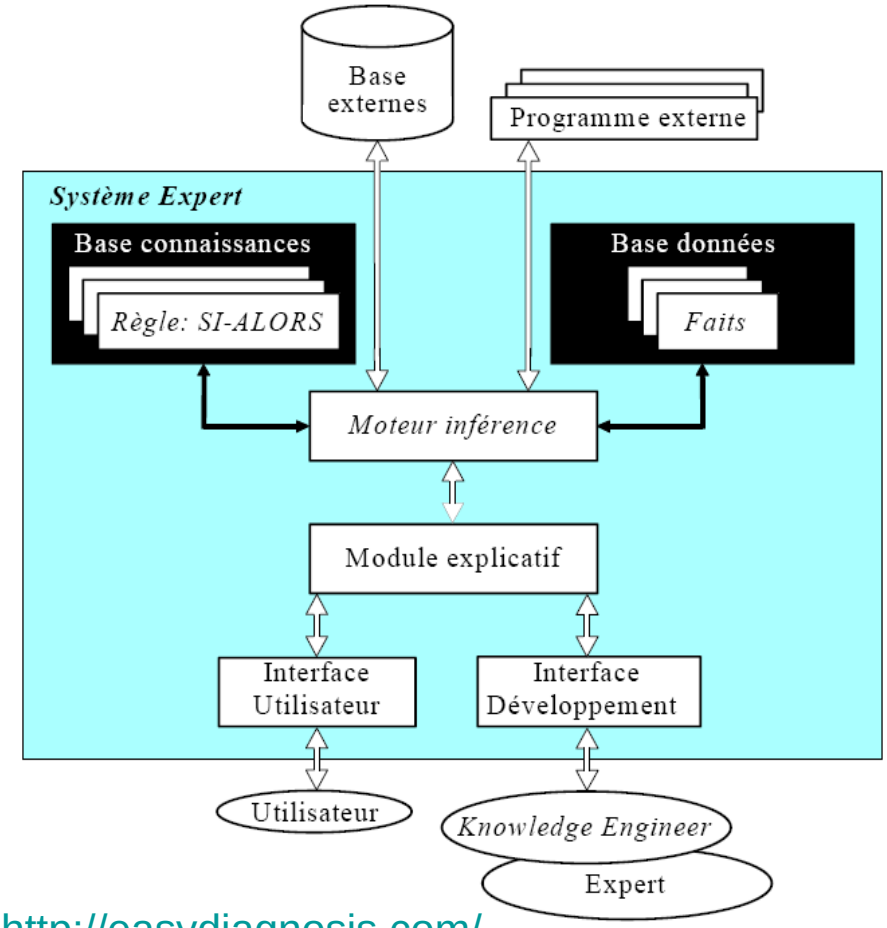
Composantes de tel Agent (système expert) à base de règles

• Un système expert à base de règles est constitué de:

– un ensemble de règles stockées dans une base de connaissances
– une mémoire de travail ou base de données (ou de faits)
– un interpréteur de règles- ou moteur d'inférence
– une interface utilisateur
– un module d'explication

• Ces 5 composantes sont essentielles à n'importe quel système à base de règles

Constituants d'un système expert à Base de Règles



http://easydiagnosis.com/

http://www.expertise2go.com/

• Le moteur d'inférence:
– modélise le raisonnement
– met en relation les règles de la base de connaissances avec les faits de la base de données

• Le module d'explication:
– permet à l'utilisateur d'interroger le SE pour savoir comment une conclusion particulière a été atteinte et pourquoi un fait spécifique a été utilisé,
– un SE doit être capable d'expliquer son raisonnement et de justifier ses conseils, son analyse ou ses conclusions

• L'interface avec l'utilisateur:
– C'est le module de communication entre l'utilisateur cherchant une solution à un problème et un SE.

Composantes additionnelles
• En plus des composantes essentielles, il peut y avoir:
• des bases de données externes
• des programmes externes
• des outils de développement
comprenant:
    • des éditeurs de base de connaissances et de bases de faits,
    • des outils de mise au point.

Représentation à base de règles

Règles de production

Prémisses ⊨ Conclusions

- **Exemples**

❑ **Si** la température du réacteur dépasse 800°C
alors descendre les barres de contrôle

❑ **Si** champignon à lames séparables et sporée rose
alors champignon du genre leucopaxillus

❑ **Si** *X* est un chien **alors** *X* est un mammifère

Expressivité des règles

- **Ordre 0** : logique des propositions
  - ❑ **Si Ferrari et Michael alors rapide**

- **Ordres 0+** : logique des propositions typée (attribut-valeur)
  - ❑ **Si voiture = ferrari et pilote=michael alors vitesse=rapide**

- **Ordre 1** : logique des prédicats

  ▭ **X,Y : Si voiture(X) et X=ferrari et pilote(X,Y) et Y=michael alors rapide(X)**

- **Ordre 2** : logique d'ordre 2

  ▭ **R, X,Y : Si type(R)=symétrique et R(X,Y) alors R(Y, X)**

Le raisonnement : règles d'inférence

- **Modus ponens**

$$\frac{A,\ A \rightarrow B}{B}$$

- **Modus tollens**

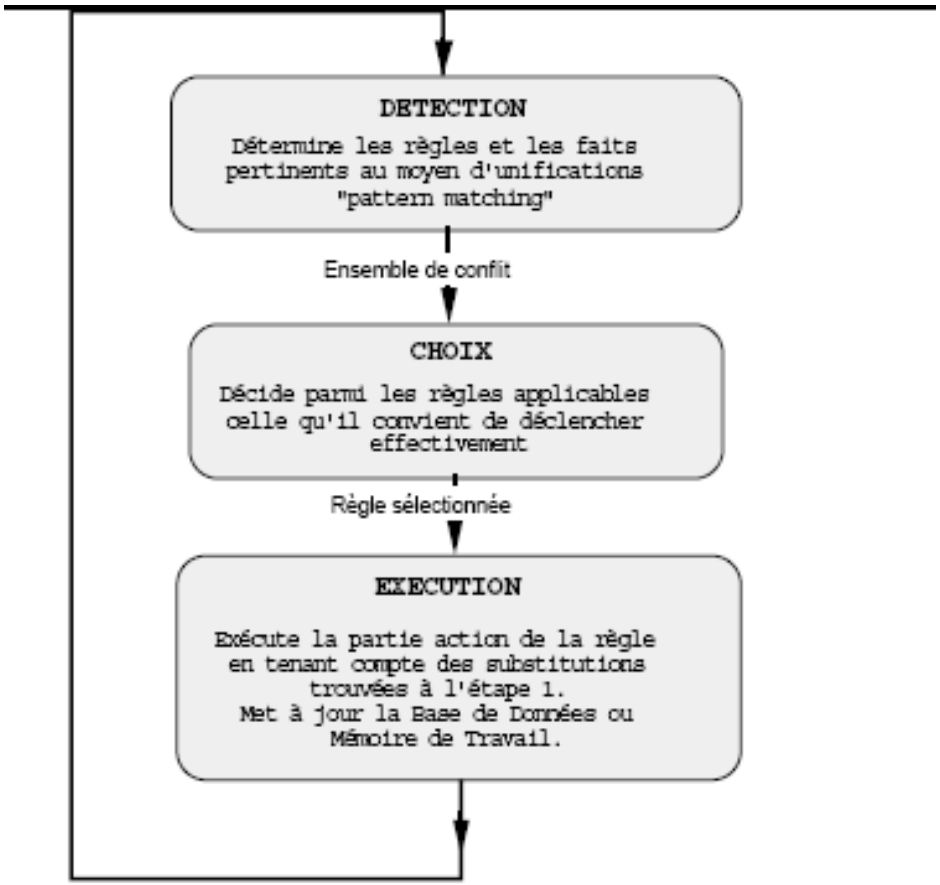
$$\frac{\neg B,\ A \rightarrow B}{\neg A}$$

- **Résolution**

$$\frac{A \vee B,\ \neg B \vee C}{A \vee C}$$

- **…**

Le cycle de production



Les deux formes de raisonnement: exemple

SI Marocain ET plus_de_18_ans ALORS Il est éligible

1) **Progressif**
« Je sais qu'il est marocain et a plus de 18 ans. Donc j'infère qu'il est éligible »

→ **Chaînage-avant** : des données aux buts

2) **Régressif**
« Je veux savoir s'il est éligible. Donc je me demande s'il est marocain et s'il a plus de 18 ans »

→ **Chaînage-arrière** : des buts aux données

La sélection

- Sélectionne une règle parmi l'ensemble de conflit
- Méthodes :
  - La première règle applicable (e.g. Prolog)
  - La règle la plus spécifique (spécificité)
    – Les règles les plus spécifiques sont préférées. Celles avec le plus grand nombre de conditions sont les plus difficiles à satisfaire, et prennent en compte le plus d'éléments de la situation courante.
  - La règle la plus utile (selon une valeur d'utilité à calculer)
    – Appliquer la règle de **plus haute priorité**
    – Pour des applications simples, la priorité peut être établie en plaçant es règles dans un ordre approprié dans la base de connaissances
  - Récence ("Recency«)
    – On préfère les règles qui utilisent les données les plus récemment ajoutées à la base de données

  …

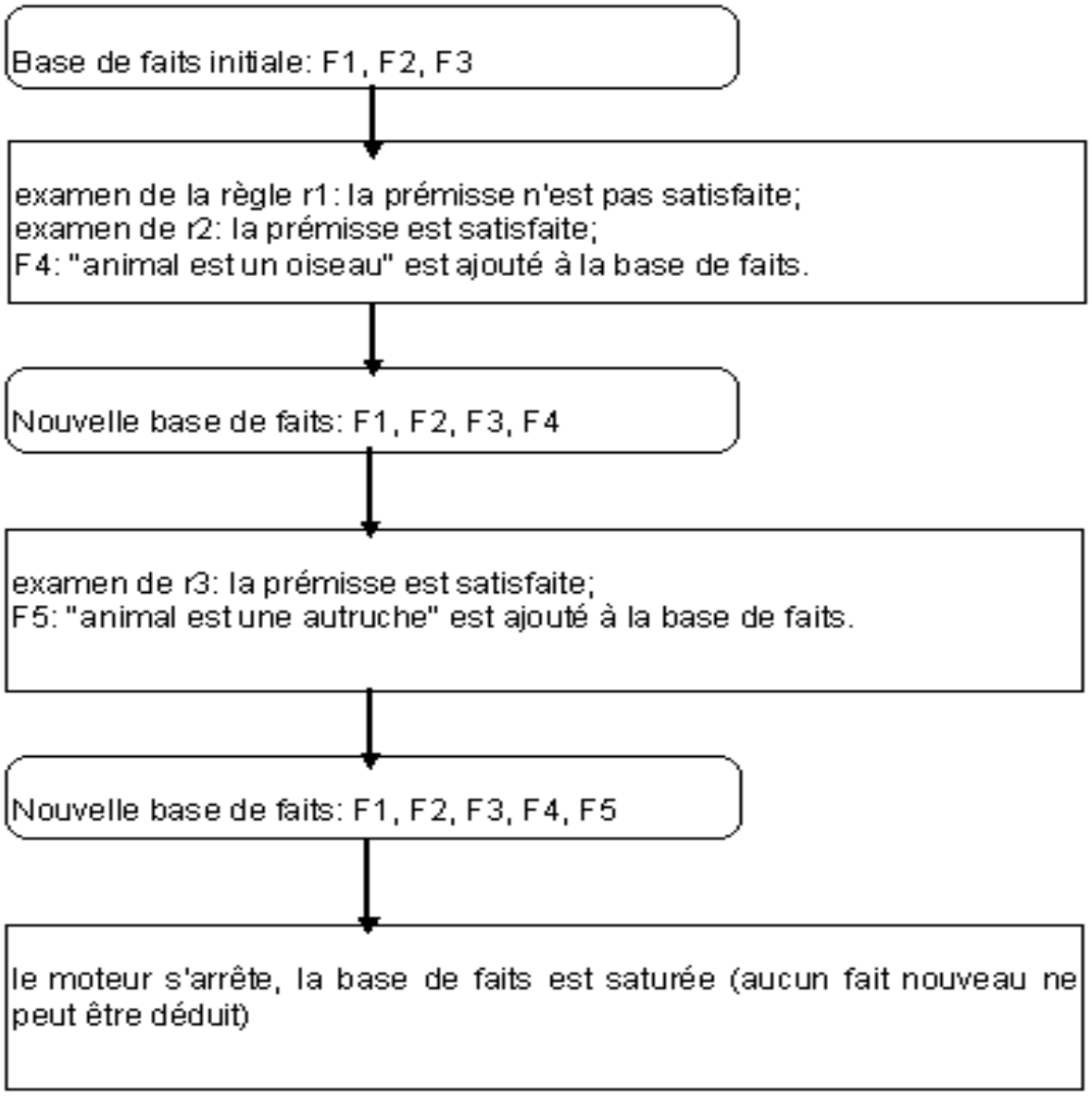
- Les métarègles
  - Des règles contrôlant la sélection des règles à appliquer

Exemple : les règles

- REGLE r1
  SI animal vole ET animal pond des oeufs
  ALORS animal est un oiseau
- REGLE r2
  SI animal a des plumes
  ALORS animal est un oiseau
- REGLE r3
  SI animal est un oiseau ET animal a un long cou ET
  animal a de longues pattes
  ALORS animal est une autruche

Exemple : les faits

- F1 : animal a des plumes
- F2 : animal a un long cou
- F3 : animal a de longues pattes



- F1 : animal a des plumes
- F2 : animal a un long cou
- F3 : animal a de longues pattes

F4 : animal est un oiseau

F5 : animal est une autruche

- REGLE r1
  SI animal vole ET animal pond des oeufs
  ALORS animal est un oiseau
- REGLE r2
  SI animal a des plumes
  ALORS animal est un oiseau
- REGLE r3
  SI animal est un oiseau ET
  animal a un long cou ET
  animal a de longues pattes
  ALORS animal est une autruche

Chaînage arrière

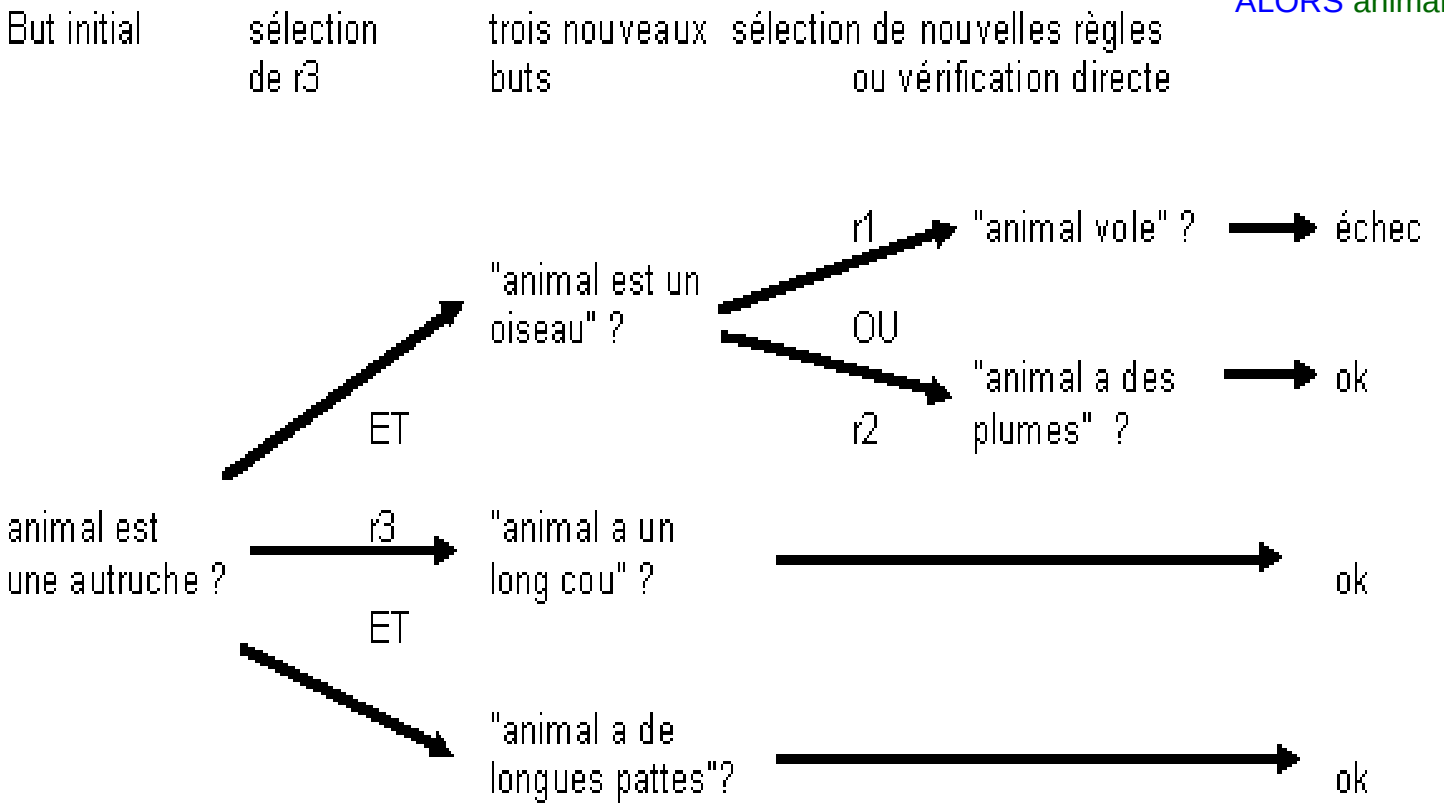
– orienté vers l'objectif, tentative d'atteindre un but spécifique,

– s'applique en "marche arrière" à partir d'une **conclusion** et recherche un ensemble de conditions qui induisent cette conclusion,

– décomposition de l'objectif initial en un ensemble de sous-objectifs,

– continuer jusqu'à ce que le but initial ait été prouvé (ou nié) ou qu'aucune condition n'est disponible.

- REGLE r1
  SI animal vole ET animal pond des oeufs
  ALORS animal est un oiseau
- REGLE r2
  SI animal a des plumes
  ALORS animal est un oiseau
- REGLE r3
  SI animal est un oiseau ET
  animal a un long cou ET
  animal a de longues pattes
  ALORS animal est une autruche



Évaluation

Systèmes Experts

- Avantages:
  - Représentation naturelle de la connaissance
    - Règles de production de la forme "Si ... Alors ..."
    - Chaque règle est un fragment indépendant de connaissance
    - Représentation auto-documentée
  - Séparation entre connaissance et traitement
    - Le moteur d'inférence est totalement séparé de la base de connaissance
    - Possibilité de développer différentes applications à partir d'un seul noyau de système expert
  - Traitement de l'information incomplète et incertaine

- Inconvénients:
  - Relations opaques entre les règles
    - Interaction logique entre les règles se fait à travers le moteur d'inférence (boîte noire),
    - Comment une règle influence le processus de raisonnement?
  - Méthode de recherche inefficace
    - Le moteur d'inférence applique une recherche exhaustive à chaque cycle,
    - De grands ensembles de règles (plus de 1000) rendent le processus assez lent.
  - Pas d'apprentissage possible (systèmes classiques)
    - Impossibilité de modifier les règles existantes ou d'en ajouter de nouvelles,
    - Utilisateur responsable des modifications du système.

EXERCICE

Soit *BF* = {*B*,*C*}, *BR* composée des règles :

Si *B* **et** *D* **et** *E* **alors** *F*
Si *G* **et** *D* **alors** *A*
Si *C* **et** *F* **alors** *A*
Si *B* **alors** *X*
Si *D* **alors** *E*
Si *X* **et** *A* **alors** *H*
Si *C* **alors** *D*
Si *X* **et** *C* **alors** *A*
Si *X* **et** *B* **alors** *D*

- Chainage Avant
- Prouver D.

INFERENCES

Références

• George F. Luger (2002)
Artificial Intelligence: Structures and Strategies for Complex Problem Solving
Addison-Wesley

• Durkin. J. (1998)
Expert Systems Design and Development
Prentice Hall

• Hopgood A. A. (1992)
Knowledge Based Systems for Engineers and Scientists
CRC Press

• Forsyth. R. (1989)
Expert Systems: Principles and Case Studies (2ème édition)
Chapman & Hall

Références Web

- CLIPS (outil de développement)
  - http://www.ghg.net/clips/CLIPS.html

- American Association for Artificial Intelligence (AAAI):
  - http://www.aaai.org/Pathfinder/html/expert.html

- Démos systèmes experts de Acquired Intelligence, Inc.
  - The Whale Watcher expert system combines artificial intelligence, marine biology and web technology to produce an interactive system that helps you with whale identification.
  - The Graduate Admissions Screening System demonstrates the use of ACQUIRE¨ with an administrative screening task - the categorization of student applications for admission to graduate school.

- Démonstrations (amusantes) en ligne à l'adresse:
  - http://www.expertise2go.com/webesie/
    (clicker sur "Demo Knowledge Bases" à gauche)