Prolog

Master I

Lakhdar SAÏS

sais@cril.fr

http://www.cril.fr/~sais/

PL : plan du cours

- Les origines de Prolog
- Deux exemples de programmes
- Syntaxe
  - Constantes, variables, termes, prédicats.
  - Assertions, Règles, Buts.
- Sémantique
  - Unification
  - Arbres de dérivation
- Le langage

PROGRAMMER EN LOGIQUE

Les origines :

- 1972 : création de Prolog par A. Colmerauer et P. Roussel à Marseille.
- 1980 : reconnaissance de Prolog comme langage de développement en Intelligence Artificielle
- Depuis plusieurs versions, dont une tournée vers la programmation par contraintes.

- Depuis, un grand nombre d'implémentations. Voir par exemple: http://fr.wikipedia.org/wiki/Prolog

La version que nous allons utiliser: SWI-Prolog

1er exemple : « biographie »

Un programme prolog: assertions et règles

```
/*bio(nom,sexe,ne en,dcd en,pere,mere)*/
bio(louis13, h, 1601, 1643, henri4, marie_medicis).
bio(elisabeth_france, f, 1603, 1644, henri4, marie_medicis).
bio(louis14, h, 1638, 1715, louis13, anne_autriche).

/*pere(pere, enfant)*/
pere(X,Y) :- bio(Y, _ , _ , _ ,X, _ ).
/*age(personne, age)*/
age(X,Y) :- bio(X, _ ,Z,T, _ , _ ),Y is T-Z.
```

1er exemple : « biographie »

On interroge le programme: buts

Quel est la date de naissance de Louis XIV?

bio(louis14, _ ,X,_, _ , _).

Qui est le père de Louis XIII?

pere(X, louis13)....

mais aussi...

bio(louis13, _ , _ , _ ,X, _).

Combien d'année Louis XIV a survécu à son père?

bio(louis14, _ , _ ,Y,Z, _), bio(Z, _ , _ ,T, _ , _), R is Y-T.

Parmi les personnes dont on dispose de la « biographie », qui sont les hommes?

bio(X, h, _ , _ , _ , _).

1er exemple : « biographie »

```
/* bio.pl */
bio(louis13, h, 1601, 1643, henri4, marie_medicis).
bio(elisabeth_france, f, 1603, 1644, henri4, marie_medicis).
bio(louis14, h, 1638, 1715, louis13, anne_autriche).
pere(X,Y) :- bio(Y,_,_,_,X,_).
age(X,Y) :- bio(X,_,T, Z,_,_), Y is Z-T.
# prolog
?- consult(bio).
yes
?-bio(louis14,_,X,_,_,_).
X = 1638 ?;
no
```

1er exemple : « biographie »

?- pere(X, louis13).

X = henry4 ?;

no

?-bio(louis13,_, _ , _ ,X, _).

X=henry4? ;

no

?- bio(louis14, _ , _ ,Y,Z, _),bio(Z, _ , _ ,T, _ , _),R is Y-T.

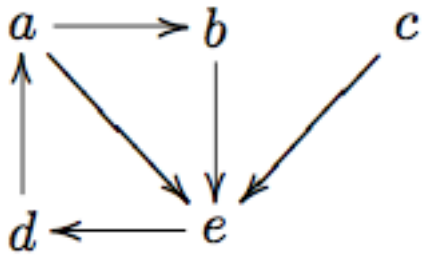
R = 72, T = 1643, Y = 1715, Z = louis13 ?;

no

1er exemple : « biographie »

```
?- bio(X,h,_, _ , _ , _ , _).
X = louis13 ?;
X = louis14 ?;
no
?- halt.
[ Prolog execution halted ]
#
```

2ème exemple : modélisation d'un graphe orienté



/* arc(source,destination) */

arc(a,b). arc(a,e). arc(b,e). arc(c,e). arc(e,d). arc(d,a).

/* connexe(source,destination) */

connexe(X,Y) :- arc(X,Y).

connexe(X,Y):- arc(X,Z),connexe(Z,Y).

2ème exemple : modélisation d'un graphe orienté

?- consult(graph).

yes

?- connexe(a,b).

yes

?- arc(a,X).

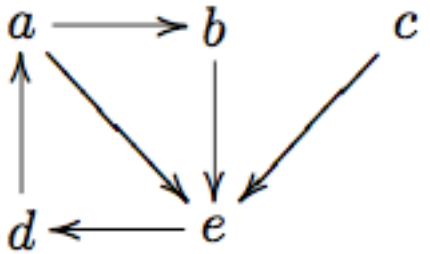
X=b? ;

X=e? ;

no

?- connexe(a,c).

boucle!



Prolog : syntaxe

Les éléments de base d'un programme Prolog sont
les prédicats et les termes.

Dans
connexe(a, X).
connexe est un (symbole de) prédicat.
a est un terme (une constante).
X est un terme (une variable).

Il n y'a pas de différence syntaxique entre prédicats et termes:
les termes aussi peuvent avoir des arguments:
membre(X, node(X,_,_)).
ici **node(X,_,_)** est un terme à trois arguments
(un arbre binaire étiqueté, par exemple).

Prolog : syntaxe

le lexique (simplifié)

Constantes : chaînes de caractères dont le premier est minuscule

variables : chaînes de caractères dont le premier est majuscule ou _

nombres : en notation décimale

les ponctuations : :- , . ( ) ; ...

Prolog : syntaxe

la grammaire (simplifiée)

programme-prolog ::= clause { clause }

clause ::= assertion | regle

assertion ::= predicat.

predicat ::= constante [(liste-termes)]

regle ::= predicat :- corps.

corps ::= predicat { , predicat }

liste-termes ::= terme { , terme }

terme ::= terme-simple | terme-complexe

terme-simple ::= constante | variable | nombre

terme-complexe ::= constante (liste-termes)

Prolog : sémantique

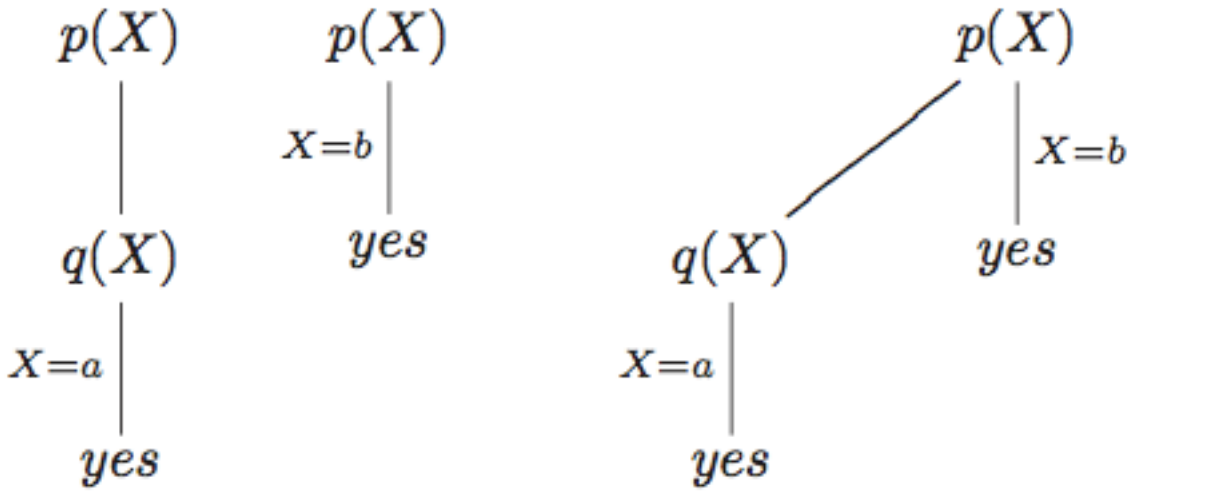
Sémantique opérationnelle: un exemple

q(a).

p(X):-q(X).

p(b).

but: p(X)



Arbre de dérivation de p(X)

Prolog : sémantique

Sémantique opérationnelle: l'unification

Une substitution est une fonction partielle qui associe des termes à des variables (qu'on peut noter comme un ensemble de couples (Var, terme). Par exemple σ0={ (X,zero), (Y,succ(T)) }
est une substitution).

L'application d'une substitution σ à un terme t, notée tσ, est le terme t dans lequel les variables de σ sont remplacées par les termes correspondants.

Par exemple add(X,Y)σ0 = add(zero, succ(T))

Deux termes t et s sont unifiés par la substitution σ si tσ ≡ sσ (c.à.d. si les deux termes résultants de la substitution sont identiques).

La substitution σ est l'unificateur le plus général (mgu) de t et s S'il unifie t et s et si tout autre unificateur de t et s s'obtient par instanciation de σ.

Prolog : sémantique

Sémantique opérationnelle: <u>l'unification</u>

Le prédicat binaire infixe = de Prolog calcule les unificateurs (mgu)

?- f(X)=f(g(a,Y)).
X = g(a,Y)

?- f(X,X)=f(g(a,Y),g(Z,b)).
X = g(a,b),
Y = b,
Z=a

?- X=f(X).
X=
f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(
f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f... Segmentation fault

Prolog : sémantique

Sémantique opérationnelle: Arbre de dérivation

On considère un programme P=c1,...ck et un but G=g1,...,gl.

Soit $c_i = t_i :- b_i^1, b_i^2, ..., b_i^{li}$.

(si $l_i = 0$ alors c_i est une assertion, sinon c'est une règle).

On définit noeuds et arcs de l'arbre de dérivation de G pour P par induction (sur la hauteur) :

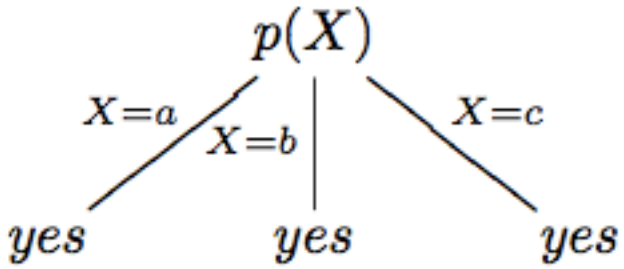
- la racine est G.
- Soit $H=h_1,...h_l$ un noeud de hauteur n, et soit c_s une clause de P dont la tête t_s s'unifie avec h_1, avec mgu σ.
  Alors on crée le noeud, de hauteur n + 1, $H' = b_s^1\sigma,...,b_s^{ls}\sigma, h_2\sigma,...h_l\sigma$ et on étiquette l'arc de H à H' par σ.

Une feuille est soit un but vide (succes), soit un but dont le premier prédicat ne s'unifie avec aucune tête de clause (echec).

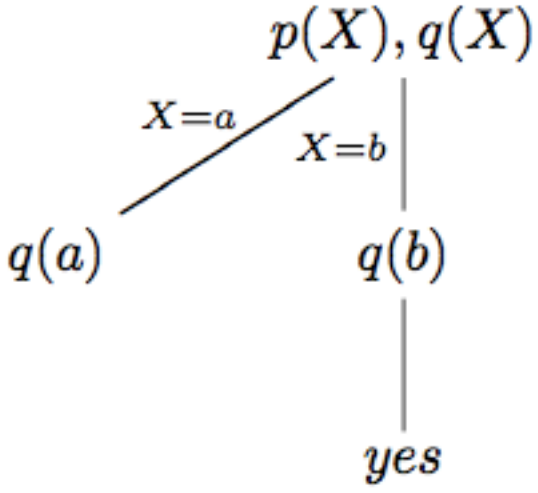
Prolog : sémantique

Sémantique opérationnelle: exemples d'arbres de dérivation

```
P = p(a).  p(b).  p(c).
G= p(X)
```




```
P= p(a).  p(b).  q(b).
G= p(X),q(X)
```



Prolog : sémantique

Sémantique opérationnelle: arbres de dérivation

L'exécution d'un goal G pour un programme P a comme résultat l'ensemble de feuilles succès de l'arbre de dérivation correspondant.

Plus précisément, pour chacune de ces feuilles, le résultat est l'ensemble des instanciations des variables de G qui se trouvent sur le chemin qui mène de la racine à la feuille en question.

L'arbre de dérivation est produit par un parcours en profondeur d'abord. Donc l'ordre des clauses est important.
Par exemple, pour p(X) :- p(X). p(a) le but p(X) ne donne aucun résultat (boucle), mais pour p(a). p(X) :- p(X) le résultat X=a est atteint.

Le langage : termes et prédicats

Comment modéliser un problème

Les entités, les objets dont on parle: les termes, construits à l'aide des symboles de fonction.

Les relations entre termes: les formules, construites à l'aide des symboles de prédicats.

exemple:

Adam aime les pommes. Clara aime les carottes.
Olivier aime les oranges. Les pommes sont des fruits.
Les oranges sont des fruits. Les carottes sont des légumes. Ceux qui aiment les fruits sont en bonne santé.

Les objets dont on parle (symboles de fonction):
adam/0, pommes/0, clara/0, carottes/0, olivier/0, oranges/0

Les relations entre objets (symboles de pŕ edicat):
fruit/1, legume/1, aime/2, bonne sante/1

Le langage : termes et prédicats

Comment modéliser un problème

Le programme correspondant à ce choix de termes et prédicats:

```
aime(adam, pommes).
aime(clara, carottes).
aime(olivier, oranges).
fruit(pommes).
fruit(oranges).
legumes(carottes).
bonne_sante(X) :- aime(X,Y),fruit(Y).
```

Le langage : termes et prédicats

Un choix alternatif

Les objets dont on parle (symboles de fonction):

adam/0, pommes/0, clara/0, carottes/0, olivier/0, oranges/0, fruit/0, legume/0

Les relations entre objets (symboles de prédicat):

aime/2, bonne sante/1, est instance de/2

Le programme correspondant:

```
aime(adam, pommes).
aime(clara, carottes).
aime(olivier, oranges).
est_instance_de(pommes, fruit).
est_instance_de(oranges, fruit).
est_instance_de(carottes, legumes).
bonne_sante(X):- aime(X,Y), est_instance_de(Y,fruit).
```

Le langage : termes et prédicats

Comment modéliser un problème (2)

Un deuxième exemple, où les termes ne sont pas forcement des constantes.

Voici ceux qu'il faut modéliser:

Un arbre binaire est soit une feuille ,
soit un noeud dont les deux fils sont des arbres binaires.

La hauteur d'une feuille est 0,
et la hauteur d'un noeud est le max des hauteurs de ses fils plus 1.

La taille d'une feuille est 0, et la taille d'un noeud est la somme des tailles de ses fils plus 1.

Le langage : termes et prédicats

Comment modéliser un problème (3)

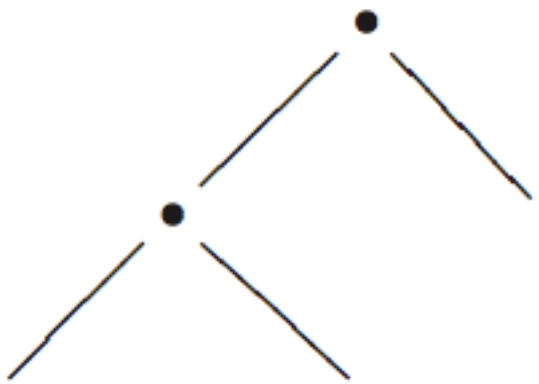
Les symboles de fonction:

feuille/0, noeud/2

Un nombre infini de termes. Par exemple

noeud(noeud(feuille,feuille),feuille)

qui désigne l'abre :



Le langage : termes et prédicats

Comment modéliser un problème (4)

Les symboles de prédicat:

hauteur/2, taille/2

Assertions et règles:

```
hauteur(feuille,0).
hauteur(noeud(T1,T2), R):-
        hauteur(T1,H1),
        hauteur(T2,H2),
        max(H1,H2,S),
        R is S+1.
taille(feuille,0).
taille(noeud(T1,T2), R) :-
        taille(T1,R1), taille(T2,R2), R is R1 + R2 + 1.
```