INFf3
Programmation logique

Cours 2 : Mécanisme de résolution

Benoît Lemaire

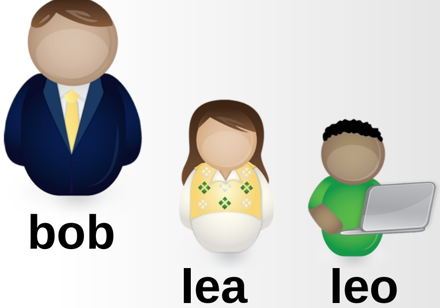
Université Grenoble Alpes
L2 - MIASHS
Grenoble – France

Ces diapos se sont inspirées de celles créées par Jean-Michel Adam

Unification

- Procédé par lequel on essaie de rendre deux clauses identiques par des substitutions (en donnant des valeurs aux variables qu'elles contiennent).
- Classique en maths :
  - f(x,b)=f(a,y) succès avec {x=a, y=b}
  - f(x,y,z)=g(x,y,z) échec
  - f(x)=f(x,y) échec
  - f(x)=f(y) succès avec {x=y}

Unification



- **L'unification réussie se propage :**

```
?- frere(X,Y).
```

```
frere(X,Y):-homme(X),parent(Z,X),parent(Z,Y),X\==Y.
frere(leo,Y):-homme(leo),parent(Z,leo),parent(Z,Y),leo\==Y.
frere(leo,Y):-homme(leo),parent(bob,leo),parent(bob,Y),leo\==Y.
frere(leo,lea):-homme(leo),parent(bob,leo),parent(bob,lea),leo\==lea.
```

instanciée

non instanciée

Unification (symbole "=")

- une variable non-instanciée peut être unifiée avec une autre variable (instanciée ou non) ou avec une constante
- une variable instanciée peut être unifiée avec une variable instanciée de même valeur ou une constante de même valeur
- une constante peut être unifiée seulement avec elle-même
- un prédicat peut être unifié avec un autre prédicat si les noms sont les mêmes, les arités identiques et les arguments s'unifient.

Parcours de résolution

- **Parcours du programme du haut vers le bas**
- **Parcours des règles de gauche à droite**
- **Quand il y a plusieurs manières d'unifier, on a un point de choix :**
  - Différentes règles définissant un même prédicat
  - Disjonction dans une règle
  - Instanciation d'une variable (plusieurs possibilités)

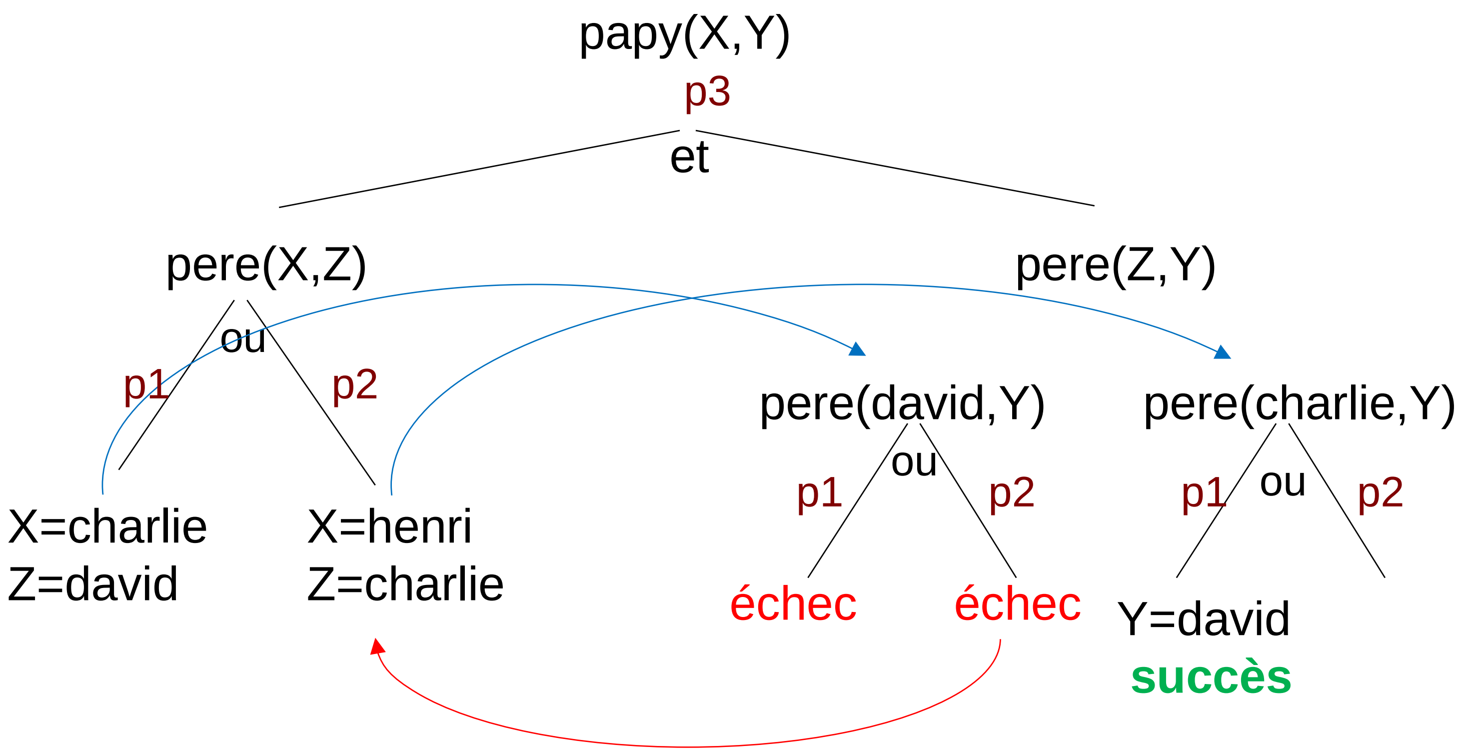
Exemples

```
?- p(X,Y)=p(1,2).
X = 1,
Y = 2.

?- p(X,a)=p(b,a).
X = b.

?- p(X,a)=p(b,Y).
X = b,
Y = a.

?- p(X,X)=p(a,b).
false.

?- p(X,Y)=p(a,a).
X = Y, Y = a.

?- X=Y,p(X,Y)=p(a,b).
false.
```

Exemple

pere(charlie, david). (p1)
pere(henri, charlie). (p2)
papy(X,Y) :- pere(X,Z), pere(Z,Y). (p3)

Requête : papy(X,Y).



Parcours de résolution

- En cas d'échec, Prolog revient en arrière (*backtrack*) sur le dernier choix.
- toutes les unifications faites depuis ce dernier point de choix sont défaites.
- Ce mécanisme garantit que toutes les combinaisons vont être explorées.

  p(X) :- a(X), b(Y), c(Z).

Parcours de résolution

p(X,Y,Z) :- a(X), b(Y), c(Z).

première solution de a → première solution de b → toutes les solutions de c

solution suivante pour b → toutes les solutions de c

toutes les solutions de c

...

...

dernière solution pour b

solution suivante pour a → première solution de b → toutes les solutions de c

...

Exercice

- Soit la base de connaissance et la requête suivante. Construire l'arbre de recherche Prolog.

```
rectangle(a).
rectangle(b).
losange(b).
carre(X) :- rectangle(X),losange(X).
```

```
?- carre(X).
```

Expressions arithmétiques

- Opérateur **is** : ...N **is** 5*X… (N = 5*X ne fait pas le calcul!)
- Opérations habituelles : addition (+), soustraction (-), multiplication (*), division entière (//), division flottante (/), modulo (mod), puissance(^)
- Fonctions mathématiques prédéfinies :

  abs(X), log(X), sqrt(X), exp(X), sign(X), random(X), sin(X), cos(X), tan(X), min(X,Y), max(X,Y), pi, etc.

```
?- X is 2^20, Y is exp(1)
X = 1048576,
Y = 2.718281828459045,

?- Z is random(100)+100.
Z = 151.
```

```
?- X is sin(pi/2), Y is cos(pi).
X = 1.0,
Y = -1.0.

?- S1 is sign(20), S2 is sign(-12).
S1 = 1,
S2 = -1.
```

Comparaison et unification de termes

- Comparer deux termes :
  - T1==T2   réussit si T1 est identique à T2
  - T1\==T2 réussit si T1 n'est pas identique à T2
  - T1=T2   réussit si T1 s'unifie avec T2
  - T1\=T2   réussit si T1 n'est pas unifiable à T2

  Exemples :

```
?- f(X)==f(x).
false.

?- f(X)=f(x).
X = x.

?- f(X)\=f(x).
false.

?- f(X)\==f(x).
true.
```

Les prédicats de comparaison

- Comparaison des expressions arithmétiques
  - X =:= Y   se traduit par X est égal à Y (il peut y avoir des expressions de chaque côté)
  - X =\= Y   se traduit par X est différent de Y (il peut y avoir des expressions de chaque côté)
  - X < Y
  - X =< Y
  - X > Y
  - X >= Y

  Il y a évaluation puis comparaison.

```
?- 5 = 3 + 2.
false
?- 5 =:= 3 + 2.
true
```

Entrées/sorties

- Affichage
  - write : write("hello !"), write(X), ...
  - nl (saut de ligne)
- Lecture
  - read :
    - `?- read(A).`
    - `|: toto.`
    - `A = toto.`

Listes

- Exemple : [4, toto, 12]
- liste vide : `[]`
- liste de listes : [[1,2],[3,4],[[5]],toto]
- S’unifie avec les variables : notes([12,6,14,17,11]) s’unifie avec notes(X) et X vaut [12,6,14,17,11]

Le symbole |

Le symbole | permet de séparer les premiers éléments du reste de la liste.

- [X|L] : X est le premier, L est le reste.
- Attention, X est un élément et L est une liste d'éléments !!!
- [X|L] = [1,2,3]   donne X=1 et L=[2,3]
- [X,Y|L]=[1,2,3]   donne X=1, Y=2, L=[3]
- [X|L] = [] échoue
- [X|L] = [1] donne X=1 L=[]
- [[X,Y]|L] = [[a,1],[b,2],[c,3]] donne X=a, Y=1, L=[[b,2],[c,3]]

Exercice

- Dire si les expressions suivantes unifient et comment :

```
?- [X|Y] = [jean, marie,  leo,  lea].
X = jean, Y = [marie, leo, lea].
?- [X|Y] = [].
false.
?- [X|Y] = [a].
X = a, Y = [].
?- [X|Y] = [[], dort(jean), [2], [], Z].
X = [], Y = [dort(jean), [2], [], Z].
?- [X,Y|W]=[[], dort(jean), [2], [], Z].
X = [], Y = dort(jean), W = [[2], [], Z].
?-  [_,X,_,Y|_] = [[], dort(jean), [2], [], Z].
X = dort(jean), Y [].
?- [_,_,_,[ _|X ]|_] = [1,2, dort(jean), [2,3], [], Z].
X = [3].
```

Parcours d’une liste

- Cas de base :

parcours([],v).   v = valeur associée à la liste vide

- Cas général :

Parcours([T|Q],R):- parcours(Q,RI), R *construit à partir de* RI et T.

Exemple : somme des éléments d’une liste

```
somme([], 0).
somme([T|Q], S) :- somme(Q, Sq), S is T + Sq.
```

Exercice

Ecrire le prédicat nbpairs/2 qui prend en premier argument une liste d'entiers et produit en second argument le nombre d’entiers pairs de la liste.

Exemple d'utilisation:

```
?- nbpairs([5,4,1,12,3],N).
N = 2.
```

Solution de l'exercice

```
nbpairs([], 0).
nbpairs([T|Q], N) :- T mod 2 =:= 0, nbpairs(Q, N1), N is
  N1+1.
nbpairs([T|Q], N) :- T mod 2 =:= 1, nbpairs(Q, N).
```