Centre Universitaire Adelhafid Boussouf Mila

Institut des Sciences et de la technologie
Département des mathématiques et de l'informatique
Filière Informatique

Intelligence Artificielle

Chapitre 4
L’arithmétique et la récursivité

I- Les opérateurs en Prolog

- Prolog comprend de nombreux opérateurs arithmétiques, parmi lesquels:

X+Y	Addition
X-Y	Soustraction
X*Y	Produit
X/Y	Division (flottant)
X//Y	Division entière
X mod Y	Modulo (reste de la division entière)
X==Y	Egal
X\=Y	Différent
X<Y	Inférieur
X>Y	Supérieur
X=<Y	Inférieur ou égal
X>=Y	Supérieur ou égal

II- Le prédicat IS

- L'unification = n'évalue pas les expressions arithmétiques, mais il existe un prédicat prédéfini pour les évaluer : IS

?- A=5+9.

A = 5+9.

?- A is 5+9.

A = 14.

?- N is 1+1, M is N+3, P = N+M.

N = 2
M = 5
P = 2+5 ;

- Le prédicat IS/2 évalue son deuxième paramètre, puis substitue le résultat à son premier paramètre. Le deuxième paramètre ne doit pas contenir d'inconnus lors de son évaluation.

?- B is 5, A is B+6.
B=5
A=11

- Exemple : écrire un programme qui pour le but *affiche(10).* affiche les nombres de 10 à 0 : (on utilise le prédicat prédéfini write) :

```
affiche(X) :- write(X), write(' '), X>0,  /* condition de fin */
                Y is X-1,  affiche(Y).

?- affiche(10).

10 9 8 7 6 5 4 3 2 1 0
```

- En prolog, on ne définit pas de fonction ou de procédures mais des prédicats, éventuellement avec 0 arguments.

- Pour faire l'équivalent d'une procédure, on dit qu'un certain prédicat est vrai à condition qu'un certain nombre d'actions soient faites, par exemple :

```
affiche :- write('bonjour'), nl , write('bonsoir').
?- affiche.
```

bonjour
bonsoir

- Pour définir une fonction f(X1,...,Xn)=Y, on définit un prédicat P(X1,X2,...,Xn,Y), par exemple :

```
a_pour_successeur(X,Y) :- Y is X+1.
?- a_pour_successeur(4,Y).
Y=5
?- a_pour_successeur(6,0).
no
```

- Pour calculer la somme de deux nombres ?

```
somme(X,Y,S) :- S is X+Y.
```

- Pour connaître le plus grand parmi 2 nombres ?

```
max2(X,Y,X) :- X>=Y.
max2(X,Y,Y) :- X<Y.
```

- Pour connaître le plus grand parmi 3 nombres ?

Sol1 :

```
max(X,Y,Z,X) :- X>=Y,X>=Z.
max(X,Y,Z,Y) :- Y>=X,Y>=Z.
max(X,Y,Z,Z) :- Z>=X,Z>=Y.
```

sol2 :

```
max3(X,Y,Z,M) :- max2(X,Y,MM),max2(MM,Z,M).
```

sol3 :

```
max3(X,Y,Z,MAX) :- max2(Z,max2(X,Y,M),MAX).
```

Cette dernière solution est incorrecte

Quelle est la réponse de Prolog à ces deux requêtes :

?- max2(X,1,3).

?- max2(1,X,3).

Remarques :

- A==B veut dire est ce que A et B sont identiques ?

  ?- 3==1+2.

  False.

  ?- X==1+2.

  False.

?- 1+2==1+2.

True.

?- 1+2==2+1.

False.

- A= :=B est ce que A et B ont la même valeur.

?- 3= :=1+2.

true.

?- 4+5= :=8+1.

true.

Exercice

- Comment Prolog répond-il ?
- ?- 2 is 6-4.
- ?- 3+2 is X.
- ?- X=3+2.
- ?- X is 3+2.
- ?- k(s(g),Y) = k(X,t(a)).

requête	Réponse
?- 2 is 6-4.	true
?- 3+2 is X	Erreur
?- X=3+2.	X=3+2
X is 3+2.	X=5
k(s(g),Y) = k(X,t(a)).	X=s(g) , Y=t(a).

Programmation récursive

- Rappel
- un programme récursif est un programme qui s'appelle lui-même
- Exemple : factorielle

✓ factorielle(0) = 1 (Cas de base)

✓ factorielle(n) = n * factorielle(n-1) si n≠0

- En Prolog, un prédicat est récursif si au moins une de ses règles associées réfère à lui-même

Exemple de récursivité simple

- digere(X,Y) :- vientDeManger(X,Y).
- digere(X,Y) :- vientDeManger(X,Z), digere(Z,Y).
- vientDeManger(moustique,sang(marie)).
- vientDeManger(grenouille,moustique).
- vientDeManger(marie,grenouille).

Exemple de récursivité simple

- ?- digere(marie,moustique).
- true.
- ?- digere(grenouille,marie)
- false
- ?- digere(marie,sang(marie)).
- true.

Intérêt de la récursivité

- Permet d'exprimer de manière succincte un processus réitéré n fois
- Expl : Exprimer la relation de descendance
- Non récursif : Nombre infini de règles
- descend(X,Y):- enfant(X,Y). descend(X,Y):- enfant(X,Z), enfant(Z,Y).
- descend(X,Y):- enfant(X,Z), enfant(Z,A), enfant(A,Y). ...

Intérêt de la récursivité

- Récursif : Deux règles
- descend(X,Y):- enfant(X,Y).
- descend(X,Y):- enfant(X,Z), descend(Z,Y).

Le cas de base

- Sans cas de base la récursivité ne s'arrète pas
- p :-p.
- ?- p.
- Il faut placer le cas de base avant la relation de récursivité sinon pas d'arret
- descend(X,Y):- descend(X,Z), enfant(Z,Y).
- descend(X,Y):- enfant(X,Y).
- enfant(marie,paul).
- ?- descend(marie,paul).
- ERROR : Out of local stack

conclusion

- Le chapitre qui suit concerne les liste, il repose sur la notion de récursivité