Intelligence Artificielle

Pr. Anoual El kah

Faculté des Sciences et Techniques – Errachidia / 2022-2023

Plan

Chapitre 3

Programmation Logique

5 – Quelques prédicats prédéfinis de SWI-Prolog

1. Comparaison de 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 est inférieur strictement à T2
- ✓ T1 =< T2 réussit si T1 est inférieur ou égal à T2
- ✓ T1 > T2 réussit si T1 est supérieur strictement à T2
- ✓ T1 >= T2 réussit si T1 est supérieur ou égal à T2

5 – Quelques prédicats prédéfinis de SWI-Prolog

2. Affectation

- ✓ le prédicat très spécial **is**, prend deux arguments dont le premier doit être une variable et le second une expression arithmétique; il évalue l'expression et affecte sa valeur à la variable.
- ✓ Exemple :
  - X is 5.
  - Y is 3+2*5.
- ✓ Notation fonctionnelle :
  - Is(X,5).
  - is(X, +(3,*(2,5))).

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), etc.

5 – Quelques prédicats prédéfinis de SWI-Prolog

3. Exercice

Donner les réponses de Prolog aux requêtes suivantes :

```
?-X is 9 mod 4.
X = 1.
?-X = 9 mod4.
X = 9 mod 4.
?-X == 9 mod4.
False.
?- 1 =:= 9 mod 4.
True.
?-X is sqrt(25), Y=X.
X = Y, Y = 5.0.
```

```
?-f(X) == f(x).
False.
?-f(X) = f(x).
X=x.
?-f(X) = X+3.
Error.
?-f(X) ==f(X+3).
False.
?-f(X) = f(X+3).
X=X+3.
```

5 – Quelques prédicats prédéfinis de SWI-Prolog

3. Entrées/Sorties

- ✓ nl ➔ (sans argument) provoque un saut de ligne
- ✓ Tab(N) ➔ (terme à valeur numérique) provoque autant de tabulations
- ✓ Read(T) ➔ lit un terme (élémentaire ou composé) et l'unifie avec T.
- ✓ write(X) ➔ a pour effet d'envoyer en sortie l'écriture canoniqe du terme-argument
- ✓ writeln(X) ➔ comme write(X) mais finit sur une nouvelle ligne

6 – Les listes

1. Syntaxe

- ✓ [a,b,c]
- ✓ [ ] est la liste vide
- ✓ [Tête | Queue] est la liste où le premier élément est Tête et le reste de la liste est Queue
- ✓ [a,b,c] = [a|[b,c]] = [a|[b|[c]]] = [a|[b|[c|[]]]]
- ✓ [a,b,c,3,'Coucou'] est une liste de constantes

6 – Les listes

1. Syntaxe

Exemples :

[X|L] = [a,b,c] donne **True** {X=a,L=[b,c]}

[X|L] = [a] donne **True** {X=a,L=[]}

[X|L] = [] donne **False**

[X,L] = [a,b,c] donne **False**

[X,Y|L] = [a,b,c] donne **True** {X=a,Y=b,L=[c]}

[X|L] = [A,B,C] donne **True** {X=A, L=[B,C]}

6 – Les listes

2. Exercice

Dire si les expressions suivantes unifient et comment:

?-[X|Y] = [jean, marie, leo, lea].

?-[X|Y] = [].

?-[X|Y] = [a].

?-[X|Y] = [[], dort(jean), [2], [], Z].

?-[X,Y | W] = [[], dort(jean), [2], [], Z].

?-[_,X,_,Y | _] = [[], dort(jean), [2], [], Z].

?-[_,_,_,[_|X] | _] = [1,2, dort(jean), [2,3], [], Z].

6 – Les listes

3. Propriétés de liste

- is_list(Terme). ➔ Réussit si Terme est une liste
- is_set(Terme). ➔ Réussit si Terme est un set (liste ne comportant pas de doublons)
- length(Liste, Int) ➔ unifie Int avec la longueur de la liste

| ?-length([1,2,3,4,5], Long).

Long=5

Yes

|?- length(L,3).

L= [_G316, _ G319, _G322]

Yes

6 – Les listes

3. Accès aux éléments

- nth0(Ind, List, Elem).
  - Récupérer l'élément d'indice Ind
  - Vérifier si l'élément d'indice I est Elem
  - Enumérer les éléments d'une liste avec leurs indices
  - Insérer l'élément *E* à l'indice *I* dans la liste *L* si l'élément à l'indice *I* est une variable libre
- nth1(Ind, List, Elem).
- Last(List,Elem).

6 – Les listes

4. Enumération d'éléments

- member(Elem,List).

| ?- member(X, [1,2,3,5,8]).

X = 1 ;

X = 2 ;

X = 3 ;

X = 5 ;

X = 8 ;

No

- Select(Elem,List,Rest).

?- select(E, [1,2,3,4], R).
E = 1
R = [2, 3, 4] ;

E = 2
R = [1, 3, 4] ;

E = 3
R = [1, 2, 4] ;

E = 4
R = [1, 2, 3] ;
No

6 – Les listes

5. Appartenance d'éléments

- memberchk(Elem, List).

```
|?- memberchk(4, [1,2,3,4]).
```

- nextto(X, Y, List).

```
| ?- nextto(3, 4, [1,2,3,4,5]).
```

Permet :

- de vérifier si *X* et *Y* sont cote-à -cote dans *List*
- de déterminer quel élément suit *X* (ou précède *Y*)
- d'énumérer les éléments *X* et *Y* qui se suivent dans *List*

6 – Les listes

6. Manipulations sur les listes

- numlist(MIN,MAX,L).

```
| ?- numlist(2,8, Domaine).
Domaine = [2, 3, 4, 5, 6, 7, 8]
```

- reverse(List1, List2).

```
| ?- reverse([1,2,3,4,5], L2).
L2 = [5, 4, 3, 2, 1]
```

- append(List1, List2, List3).

```
| ?- append([1,2,3], [4,5,6], L3).
L3 = [1, 2, 3, 4, 5, 6]
```

- flatten(List1, List2).

```
| ?- flatten([[1,[2],3], [[4,5],[6,7]]], Flat).
Flat = [1, 2, 3, 4, 5, 6, 7]
```

- permutation(List1, List2).

```
| ?- permutation([1,2,3,4,5], [5,3,1,4,2]).
true
```

7. Exercices

Ecrire les prédicats suivants:

- affiche(L) est vrai si tous les éléments de la liste L sont écrits.
- affiche2(L) est vrai si tous les éléments de la liste L sont écrits en ordre inverse.
- premier(E,L) est vrai si E est le premier élément de L .
- premier2(L) est vrai si le premier élément de la liste L est affiché (et aucun autre).
- element(X,L) est vrai si X est élément de la liste L
- pair(L) qui est vrai si L a un nombre pairs d'éléments
- longueur(L,N) est vrai si N est la longueur de la liste L.

8. Exercices à rendre

Ecrire les prédicats suivants:

- dernier(E,L) est vrai si E est le dernier élément de L.
- avdernier(X,L) est vrai si X est l'avant-dernier élément d'une liste L.
- indice(X,L,N) est vrai si X appartenant à L et N est l'indice de la première occurrence de X dans L. Peut-on utiliser ce prédicat pour formuler une requête permettant de calculer le i ème élément d'une liste ?
- remplace(X1,X2,L1,L2) qui construit la liste L2 qui est la liste L1 dans laquelle X1 est remplacé par X2.
- somme(L,N) est vrai si N est la somme des éléments de la liste d'entiers L
- occurrence(L,X,N) est vrai si N est le nombre de fois où X est présent dans la liste L.
- palindrome(L) vrai si la liste L est sa propre image renversée. (exemple: [x,a,m,a,x].)