PROLOG - TP 1

Licence 3 SPI Parcours informatique
Faculté Sciences et Techniques

Prise en main

Même si la plupart des noyaux Prolog sont multi-plateformes, les TP de Prolog se dérouleront exclusivement sous linux.
Nous utiliserons le noyau Prolog SWI-Prolog. Cet interpréteur est appelé par la commande *swipl*.

Vos programmes Prolog seront réalisés à l'aide de l'éditeur de texte de votre choix (gedit, vi, emacs, ...). L'extension des fichiers contenant le code source d'un programme Prolog est généralement *.pl*.
Le chargement de votre programme s'effectuera avec le prédicat consult/1.
Exemple pour un programme Prolog écrit dans le fichier test.pl :
consult ('test').
Il est également possible d'utiliser les crochets :
[test].

Le prédicat listing/0 permettra d'afficher les faits et les règles chargés dans le noyau Prolog.

Le prédicat trace/0 permet de suivre l'exécution de l'algorithme de résolution de buts : ce prédicat est très utile pour le déboggage.

Exercices.

1.Copiez la base de connaissances 2 du cours 1 et enregistrez le code dans le fichier bc2.pl :

```
cool(yolanda).
ecoute2LaMusique(mia).
ecoute2LaMusique(yolanda):- cool(yolanda).
joueAirGuitar(mia):- ecoute2LaMusique(mia).
joueAirGuitar(yolanda):- ecoute2LaMusique(yolanda).
```

2. Chargez le programme dans l'interpréteur et testez la commande listing/0.

3. Testez également la commande listing/1. Par exemple : listing(ecoute2LaMusique).

4. Faites quelques essais de requêtes :
?- cool(yolanda).
?- cool(hop).
?- `ecoute2LaMusique`(butch).
?- cool(X).
?- `joueAirGuitar`(X).

5. Testez les exemples d'unification vus en cours avec le prédicat =/2 :

```
?- mia = mia.
?- mia = vincent.
?- mia = X.
```

```
?- X=mia, X=vincent.
?- k(s(g),Y) = k(X,t(k)).
?- k(s(g),t(k)) = k(X,t(Y)).
```

6. Copiez et enregistrez le programme suivant :

```
f(a).
f(b).

g(a).
g(b).

h(b).

k(X):-  f(X),  g(X),  h(X).
```

Ensuite, activez le mode de trace :

```
trace.
```

et demandez la requête suivante :

```
k(X).
```

Essayez de comprendre le déroulement de la résolution de ce but, pas à pas.

7. Soit le labyrinthe défini par le prédicat connecte/2. :

```
connecte(1,2).
connecte(3,4).
connecte(5,6).
connecte(7,8).
connecte(9,10).
connecte(12,13).
connecte(13,14).
connecte(15,16).
connecte(17,18).
connecte(19,20).
connecte(4,1).
connecte(6,3).
connecte(4,7).
connecte(6,11).
connecte(14,9).
connecte(11,15).
connecte(16,12).
connecte(14,17).
connecte(16,19).
```

Dans cette représentation, le lien indiqué ne fonctionne que dans un sens (gauche->droite).

a) Ecrivez le prédicat chemin/2 qui indique de quel point à quel point il est possible d'aller : est-il possible d'aller du point 5 vers le point 10 ?
b) Quels sont les points, à part le point 2, accessibles à partir du point 1 ?
c) Et à partir du point 13 ?

8. Soit l'ensemble de faits suivant :

```
byCar(auckland,hamilton).
byCar(hamilton,raglan).
byCar(valmont,saarbruecken).
byCar(valmont,metz).

byTrain(metz,frankfurt).
byTrain(saarbruecken,frankfurt).
byTrain(metz,paris).
byTrain(saarbruecken,paris).

byPlane(frankfurt,bangkok).
byPlane(frankfurt,singapore).
byPlane(paris,losAngeles).
byPlane(bangkok,auckland).
byPlane(singapore,auckland).
byPlane(losAngeles,auckland).
```

a) Ecrire le prédicat voyage/2 qui détermine si il est possible d'aller d'une ville à une autre en utilisant une combinaison de différents moyens de transport. Par exemple, votre programme doit répondre yes à la requête `voyage(valmont,raglan).`

b) Ecrire le prédicat voyage/3 qui permet de connaitre les différents étapes. Par exemple, votre programme devrait répondre :

```
X  =  go(valmont,metz,
                go(metz,paris,
                      go(paris,losAngeles)))
```

à la requête :

```
travel(valmont,losAngeles,X).
```

c) Améliorer votre programme de manière à connaître également le moyen de transport utilisé à chaque étape.