

5 Négation et manipulation de termes

Exercice 1

Négation par l'échec $\backslash +$.

1. Soit le programme :

```
p(X) :- q(X), \+ r(X).  
r(X) :- w(X), \+ s(X).  
q(a). q(b). q(c).  
s(a). s(c).  
w(a). w(b).
```

Ecrire l'arbre de recherche résultant du but $p(a)$. Que rend le but $p(b)$?

2. On considère les faits et les règles suivants :

```
q(1). q(2). q(3).  
r(2).  
s(2,a). s(2,b). s(3,a). s(3,b).  
p1(X,Y) :- q(X), r(X), !, s(X,Y).  
p2(X,Y) :- q(X), \+ r(X), s(X,Y).  
p3(X,Y) :- \+ q(X), r(X), s(X,Y).
```

Donner l'arbre de dérivation et les réponses de Prolog pour chacune des requêtes suivantes :

- (a) $p1(X,Y)$
 - (b) $p2(X,Y)$
 - (c) $p3(X,Y)$
3. On considère la base de données parentales de la feuille 1, qui est représentée par des faits **homme/1**, **femme/1** et **enfant/2**. Ecrire les requêtes pour chercher :
 - (a) les hommes sans enfants mentionnés dans la base,
 - (b) les personnes X et Y dans la base telles que X n'est pas le père de Y .

Exercice 2

Prédicat : **univ** et **call**

1. Ecrire un prédicat **filtre(+L,+NP,?R)** qui met dans la liste R la liste L purgée de ses éléments X pour lesquels le but $NP(X)$ échoue. On suppose que le code Prolog du prédicat NP est fourni.

Par exemple on possède les prédicats **pair(X)** et **positif(X)** défini comme suit :

```
pair(X) :- 0 is X mod 2.  
positif(X) :- X >= 0.
```

Si l'utilisateur souhaite purger de sa liste tous les nombres négatifs, il peut poser :

```
?- filtre([-5,2,3,-4,6],positif,R).  
R = [2,3,6]
```

S'il veut enlever les nombres impairs de sa liste :

```
?- filtre([-5,2,3,-4,6],pair,R).  
R = [2,-4,6]
```

2. Ecrire les prédicats `addition(+X,+Y,?Z)`, `soustraction(+X,+Y,?Z)`, `multiplication(+X,+Y,?Z)` et `division(+X,+Y,?Z)` qui effectuent les opérations arithmétiques correspondant. Ecrire un prédicat qui réalise, au choix, la somme ou la multiplication (ou autre) des éléments deux à deux d'une liste. Par exemple :

```
?- appliquer([1,2,3,4,5,6],addition,R).  
R = [3,7,11]
```

Comment faire pour appeler ce prédicat avec : `appliquer('+', [1,2,3,4,5,6],L)` ?

Exercice 3

Ecrire un prédicat `eval(+E,+L,?V)` qui prend un terme E qui représente une expression, une liste L de valeurs associées aux symboles de E et qui évalue la valeur V de E à ces valeurs. L'expression E est écrite avec des opérations `+`, `-`, `*`, `/` et des parenthèses (et). Exemple :

```
?- eval(x * (y + 2), [(x,2),(y,3)],V).  
V = 10  
?- eval(x * (y / 2), [(x,2),(y,3)],V).  
V = 3.0
```

Modifier le prédicat pour accepter la liste L sous le format `[x=2, y=3]` :

```
?- eval(x * (y + 2), [x=2, y=3],V).  
V = 10
```