Université d'Ottawa · University of Ottawa

Faculté de Génie – EECS

CSI2520 : PARADIGMES DE PROGRAMMATION

Hiver 2014 – Solution Tutorat 2

Exercice 1

1. Ecrire un prédicat qui convertit une température en degrés Celsius à son équivalent en température en degrés Fahrenheit. La formule de conversion est donnée par l'expression suivante :

F = C * 9/5 + 32

```
c_to_f(C,F) :- F is C * 9 / 5 + 32.
```

2. Ecrire un prédicat qui détermine si une température (ºF) est au dessous du point de congélation (32ºF).

```
freezing(F) :- F =< 32.
```

Exercice 2

L'heure du jour est exprimée en heures, minutes et secondes, comme *c(H, M, S)*. On veut écrie un prédicat qui va produire exactement une seule solution : l'heure après une seconde. Voici quelques exemples:

```
?- heureSuivanteS(c(21, 54, 25), N).
N = c(21, 54, 26) .

?- heureSuivanteS(c(21, 54, 59), N).
N = c(21, 55, 0) .

?- heureSuivanteS(c(21, 59, 59), N).
N = c(22, 0, 0) .
```

?- heureSuivanteS(c(23, 59, 59), N).

N = c(0, 0, 0).

Solution:

```
clockNextS(c(H, M, S), c(H, M, SN)) :- S < 59,  SN is S + 1.
clockNextS(c(H, M, S), c(H, MN, 0)) :- S =:= 59,  M < 59,  MN is M +
1.
clockNextS(c(H, M, S), c(HN, 0, 0)) :- S =:= 59,  M =:= 59,  H < 23,
HN is H + 1.
clockNextS(c(H, M, S), c(0, 0, 0)) :- S =:= 59,  M =:= 59,  H =:= 23.
```

Exercice 3

Soit le prédicat suivant :

```
factorial(0,1).
factorial(N,F) :-
   N>0,
   N1 is N-1,
   factorial(N1,F1),
   F is N * F1.
```

Combien de fois le prédicat factorial/2 sera appelé par la requête suivante :

```
?- factorial(3,W).
```

Justifier votre réponse en montrant l'arbre d'exécution.

Arbre d'exécution:

```
?- factorial(3, W).
(2)   {N = 3}
?- N1 is N-1, factorial(N1, W1), W is N *W1
(2)   {N1 = 2}
?-N2 is N1-1,factorial(N2,W2),W1 is N1*W2,W is N*W1.
(2)   {N2 = 1}
?-N3 is N2-1, factorial(N3, W3), W2 is N2 * W3, W1 is N1*W2,W is N*W1.
(1)   {N3 = 0, W3 = 1}
?- W2 is 1 * W3 , W1 is N1 * W2, W is N * W1.
Succeeds {W2 = 1, W1 =  2, W = 6}

Donc le prédicat factorial/2 sera appelé est 3 fois.
```

Exercice 4

Soit la base de faits suivante :

regne(philippeIV,1285,1314).
regne(philippeIII,1270,1285).

Et le prédicat suivant :

est_roi_en(Nom,Annee):- regne(Nom,A,B), Annee>=A, Annee=<B.

Définissez un opérateur *est_roi_en* qui peut être utilisé de la façon suivante :

philippeIII est_roi_en 1275
R est_roi_en 1290

Solution :

```
:- op(500, xfy, est_roi_en).
```

Exercice 5

Donnez l'expression d'un prédicat qui détermine si un nombre positif est pair ou impair.

Solution :

```
pair(0).
pair(N) :- N=\=0, M is N-1, impair(M).
impair(N) :- N=\=0, M is N-1, pair(M).
```

ou encore

```
pair(0).
pair(X) :- X>0, X2 is X-2, pair(X2).
impair(X):- \+pair(X).
```

Exercice 6

.

Écrivez un programme Prolog pour calculer la fonction d'Ackerman définit comme suit:

$$A(x, y) \equiv \begin{cases} y + 1 & \text{if } x = 0 \\ A(x - 1, 1) & \text{if } y = 0 \\ A(x - 1, A(x, y - 1)) & \text{otherwise.} \end{cases}$$

Notez bien que la valeur résultante, ainsi que le temps de calcul, grandit très vite, même pour des nombres petits. Par exemple, A(4,2) est un entier constitué de 19729 chiffres décimaux.

Solution :

```
ack(0,N,A) :- A is N + 1.
ack(M,0,A) :- M > 0, M1 is M - 1, ack(M1,1,A).
ack(M,N,A) :- M > 0, N > 0, M1 is M - 1, N1 is N - 1, ack(M,N1,A1),
ack(M1,A1,A).
```

Exercice 7

L'inspecteur Maigret veut connaître les suspects qu'il doit interroger pour un certain nombre de faits : il tient un individu pour suspect dès qu'il était présent dans un lieu, un jour où un vol a été commis et s'il a pu voler la victime. Un individu a pu voler, soit s'il était sans argent, soit par jalousie. On dispose de faits sur les vols, par exemple, Marie a été volée lundi à l'hippodrome, Jean, mardi au bar, Luc, jeudi au stade. Il sait que Max est sans argent et qu'Eve est très jalouse de Marie. Il est attesté par ailleurs que Max était au bar mercredi, Eric au bar mardi, et qu'Eve était à l'hippodrome le lundi (on ne prend pas en compte la présence des victimes comme possibilité qu'ils aient été aussi voleurs ce jour là).

Ecrivez le programme Prolog qui, a la question suspect(X), renverra toutes les réponses possibles et représenter l'arbre de recherche de Prolog.

Solution :

```
suspect(X) :- present(X, L, J), vol(L, J, V), apuvoler(X, V).
apuvoler(X, _) :- sansargent(X).
apuvoler(X, Y) :- jaloux(X, Y).
vol(hipp, lundi, marie).
vol(bar, mardi, jean).
vol(stade, jeudi, luc).
sansargent(max).
jaloux(eve, marie).
present(max, bar, mercredi).
present(eric, bar, mardi).
present(eve, hipp, lundi).

?- suspect(X).
X = eve
```