Functional and Logic Programming Fall 2017

S10 : Prolog (DCG)

Professor : Le Peutrec Stephane
Assistant : Lauper Jonathan

Exercise 1

(1)

```
% DCG that recognize the language [0]*[1][0]*[1]
s --> b, b.
b --> [0], b.
b --> [1].
```

(2)

```
% DCG that recognize the language (ab)*
s2 --> [a], [b], s2.
s2 --> [].
```

(3)

```
% DCG that recognize the langauge of expression
% term, factor and parenthesis expression
e --> t.
e --> t, ['+'], e.
t --> f.
t --> f, ['*'], t.
f --> [Int], {integer(Int)}.
f --> ['('], e, [')'].
```

Exercise 2

```
% V is the evaluation of the expression given to the DCG
e(V) --> t(V).
e(V) --> t(V1), ['+'], e(V2), {V is V1 + V2}.
t(V) --> f(V).
t(V) --> f(V1), ['*'], t(V2), {V is V1 * V2}.
f(V) --> [Int], {integer(Int), V is Int}.
f(V) --> ['('], e(V), [')'].
```

Exercise 3

```
% Gender is unify with either male or female if sentence/4 succeed
% Number is unify with either singular or plural if sentence/4 succeed
sentence(Gender,Number) --> nominal_group(Gender,Number), verbal_group(Number).

nominal_group(G,N) --> article(G,N), substantive(G,N).

nominal_group(G,N) --> article(G,N), substantive(G,N), adjective(G,N).

nominal_group(G,singular) --> firstname(G).

verbal_group(N) --> verb(N).

% terminal symbols
% using member is simplier because we have to check for Gender and Number
firstname(G) --> [N], {member(N,[ruth, david]), gender(N,G)}.

article(G,N) --> [A], {member(A,[le, la, les, un, une, des]), gender(A,G), number(A,N)}.

substantive(G,N) --> [S], {member(S,[chien, chat, amis, voiture, souris]), gender(S,G), number(S,N)}.

verb(N) --> [V], {member(V,[chante, roule, marche, racontent, avancent]), number(V,N)}.

adjective(G,N) --> [A], {member(A,[rouge, rouges, blanc, blancs, blanche, blanches]), gender(A,G),
↪  number(A,N)}.

%%% Gender

% gender(?E,?G), succeed if E as the gender G
% G is unify with the atoms male or female
gender(A,male) :- is_male(A).
gender(A,female) :- is_female(A).

is_male(A) :- member(A,[rouge,rouges,blanc,blancs,chien,chat,amis,le,les,un,des,david]).

is_female(A) :- member(A,[blanche,blanches,voiture,souris,la,une,ruth,les,des]).

%%% Number

% number(?E,?N), succeed if E as the numbering N
% N is unify with the atoms singular or plural
number(A,singular) :- is_singular(A).
number(A,plural) :- is_plural(A).

is_singular(A) :-
↪  member(A,[le,la,un,une,chien,chat,voiture,souris,rouge,blanc,blanche,chante,roule,marche]).

is_plural(A) :- member(A,[les,des,souris,amis,racontent,avancent,rouges,blancs,blanches]).
```

Exercise 4

(a)

```
% myLast(+X,-L:list), succeed if X is the last element of L
myLast(X,L) :- myLast(X,L,[]).

myLast(X) --> [X].
myLast(X) --> [_], myLast(X).
```

(b)

```
% convertToDec(?X,+L:list[int]), succeed if X is the decimal value of the binary value L
% L is a list of 1 and 0, like L = [0,0,1,1,0,1,0]
% we use a big endian representation, for example :
% - [1,0,0] is the decimal value of 1
% - [0,0,1] is the decimal value of 4
convertToDec(X,L) :- convertToDec(X,0,L,[]).

convertToDec(0,_) --> [0].
convertToDec(X,Depth) --> [1], {X is 2**Depth}.
convertToDec(X,Depth) --> [0], {Depth1 is Depth+1}, convertToDec(X,Depth1).
convertToDec(X,Depth) --> [1],
                          {Depth1 is Depth+1},
                          convertToDec(Xsub,Depth1),
                          {X is Xsub + 2 ** Depth}.
```