Laborator 14

Pro(log(os))

1. Fie urmatorul exemplu care determinarea factorilor primi pentru un numar de intrare pozitiv:

```
% prime_factors(N, L) :- N is the list of prime factors of N.
%    (integer,list) (+,?)
prime_factors(N,L) :- N > 0,  prime_factors(N,L,2).
prime_factors(1,[],_) :- !.
prime_factors(N,[F|L],F) :-                          % N is multiple of F
   R is N // F, N =:= R * F, !, prime_factors(R,L,F).
prime_factors(N,L,F) :-
   next_factor(N,F,NF), prime_factors(N,L,NF).       % N is not multiple of F
next_factor(_,2,3) :- !.
next_factor(N,F,NF) :- F * F < N, !, NF is F + 2.
next_factor(N,_,N).
```

```
Pentru testarea se va scrie in linia de comanda
?- ['exemplu.pl'].
?- prime_factors(20,L).
```

Sa se modifice astfel incat sa putem obtine numere prime dintr-un interval dat

2. Fie urmatorul exemplu de construire a codului GRAY

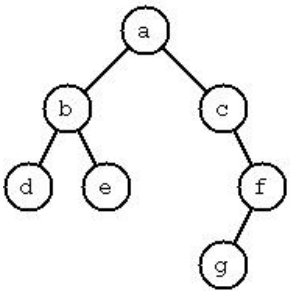
```
% gray(N,C) :- C is the N-bit Gray code
gray(1,['0','1']).
gray(N,C) :- N > 1, N1 is N-1,
   gray(N1,C1), reverse(C1,C2),
   prepend('0',C1,C1P),
   prepend('1',C2,C2P),
   append(C1P,C2P,C).
prepend(_,[],[]) :- !.
prepend(X,[C|Cs],[CP|CPs]) :- atom_concat(X,C,CP), prepend(X,Cs,CPs).
:- dynamic gray_c/2.
gray_c(1,['0','1']) :- !.
gray_c(N,C) :- N > 1, N1 is N-1,
   gray_c(N1,C1), reverse(C1,C2),
   prepend('0',C1,C1P),
   prepend('1',C2,C2P),
   append(C1P,C2P,C),
   asserta((gray_c(N,C) :- !)).
```

```
Pentru testarea se va scrie in linia de comanda
?-['ex2.pl'].
?-gray(2,C).
```

Sa se genereze si sa se scrie intr-un fisier codul gray pentru 256 de biti (cate un cod pe linie)

3. In prolog un arbore vid este reprezentat prin atomul ‘nil’ si arborele nevid prin termenul t(X,L,R), unde X – radacina, L – subarbore stang, Y – subarbore drept. De exemplu pentru arborele din figura

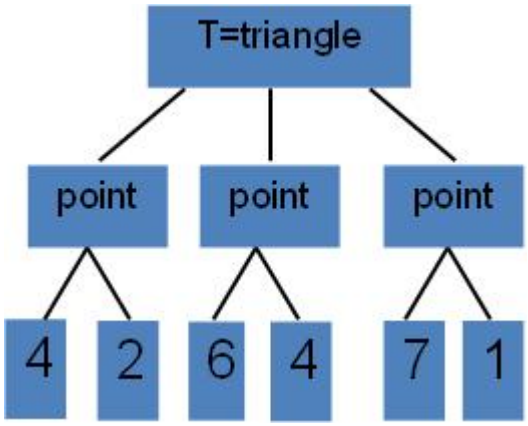
putem scrie T1 = t(a,t(b,t(d,nil,nil),t(e,nil,nil)),t(c,nil,t(f,t(g,nil,nil),nil))) si sa se testeze cu acest arbore exemplul de mai jos care extrage frunzele dintr-un arbore si le pune intr-o lista




```
istree(nil).
istree(t(_,L,R)) :- istree(L), istree(R).

% leaves(T,S) :- S is the list of the leaves of the binary tree T
leaves(nil,[]).
leaves(t(X,nil,nil),[X]).
leaves(t(_,L,nil),S) :- L = t(_,_,_), leaves(L,S).
leaves(t(_,nil,R),S) :- R = t(_,_,_), leaves(R,S).
leaves(t(_,L,R),S) :- L = t(_,_,_), R = t(_,_,_),
    leaves(L,SL), leaves(R,SR), append(SL,SR,S).

leaves1(nil,[]).
leaves1(t(X,nil,nil),[X]) :- !.
leaves1(t(_,L,R),S) :-
    leaves1(L,SL), leaves1(R,SR), append(SL,SR,S).

% nnodes(T,N) :- T is a binary tree with N nodes (o,i)
nnodes(nil,0) :- !.
nnodes(t(_,L,R),N) :- N > 0, N1 is N-1,
   between(0,N1,NL), NR is N1-NL,
   nnodes(L,NL), nnodes(R,NR).

% leaves2(T,S) :- S is the list of leaves of the tree T (o,i)
leaves2(T,S) :- leaves2(T,S,0).

leaves2(T,S,N) :- nnodes(T,N), leaves1(T,S).
leaves2(T,S,N) :- N1 is N+1, leaves2(T,S,N1).
```

```
Pentru testarea se va scrie in linia de comanda
?-['ex3.pl'].
?- istree(t(x,t(x,nil,nil),t(x,nil,t(x,nil,nil)))).
?-leaves2(t(x,t(x,nil,nil),t(x,nil,t(x,nil,nil))),S).
```

Tema 1. Propuneti si scrieti un program pentru o reprezentare pentru dreptunghi, patrat si cerc ca o structura Prolog. (De exemplu un dreptunghi poate fi reprezentat de patru puncte)

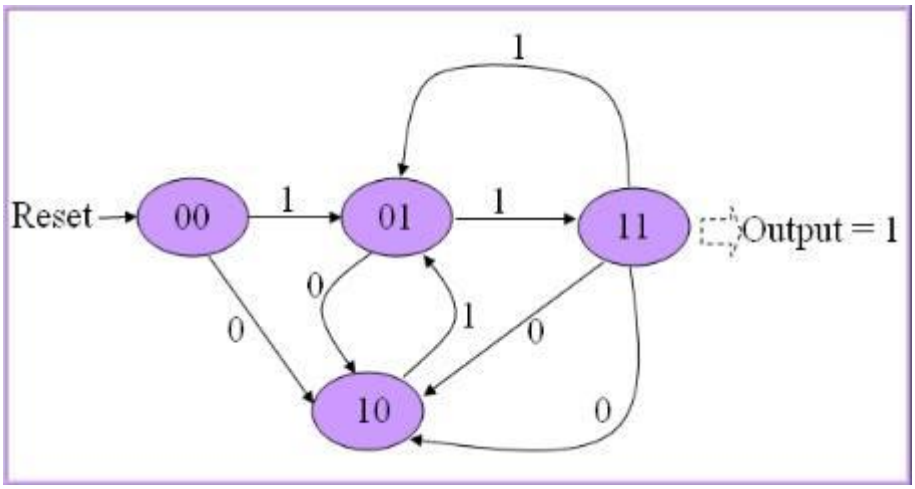


Tema 2. Fie urmatoarele:

```
big( bear).
big( elephant).
small( cat).
brown( bear).
black( cat).
gray( elephant).
dark(Z):- black(Z).
dark(Z):- brown(Z).
```

In care din cele doua cazuri Prolog trebuie sa munceasca mai mult inainte de a gasi raspunsul ? **?- big(X), dark(X). sau ?- dark(X), big(X).**

Tema 3. Sa se scrie un program prolog pentru implementarea urmatoruui automat



Tema 4. Sa se scrie un program Prolog care sa poata gasi ultimul element dintr-o lista (?- my_last(X,[a,b,c,d]). X = d)

Tema 5. Sa se scrie un program Prolog care sa poata elimina duplicatele dintr-o lista fara a schimba ordinea acestora (?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X). → X = [a,b,c,a,d,e])

Tema 6. Sa se scrie un program Prolog care sa poata elimina fiecare al n-lea element dintr-o lista (?-drop([a,b,c,d,e,f,g,h,i,k],3,X). → X = [a,b,d,e,g,h,k])

Referinte

http://www.ic.unicamp.br/~meidanis/courses/mc336/problemas-prolog/

Ivan Bratko, Programming for Artificial Intelligence, Three edition 2001, Addision Wesley