**Εργασία 4**

Θεωρήσατε το παρακάτω Prolog πρόγραμμα.

Το κατηγόρημα monopati(X, Y, Monopati) είναι αληθές εάν Monopati είναι το μονοπάτι μεταξύ κορυφών Χ και Υ σ’ ένα κυκλικό κατευθυνόμενο γράφο.

monopati(X, Y, Monopati) :- monopati1(X, Y, [X], Monopati).

monopati1(X, X, Monopati, Monopati).

monopati1(X, Z, Monopati, Teliko\_monopati) :- akme(X, Y), \+ member(Y, Monopati), append(Monopati, [Y], Neo\_monopati), monopati1(Y, Z, Neo\_monopati, Teliko\_monopati).

member(X,[H|T]) :- X=H.

member(X,[H|T]) :- X \= H, member(X,T).

append([],L2,L2).

append([H1|T1], L2, [H1|T3]) :- append(T1, L2, T3).

akme(a, c).

akme(c, a).

akme(c, e).

akme(c, f).

akme(b, c).

akme(b, d).

akme(d, f).

akme(d, g).

Να αναπαραστήσετε το παραπάνω πρόγραμμα σε μη-βασικούς όρους. Κάθε πρόταση του προγράμματος σας θα είναι μια πρόταση της Prolog στη εξής μορφή. clause2(ΚεφαλήΠροτασης, ΣώμαΠρότασης).

Να γράψετε ένα μεταπρόγραμμα το οποίο θα κάνει τα εξής: Αρχικά θα δίνει στο χρήστη τον εξής κατάλογο επιλογών:

1. Για ενημέρωση του προγράμματος-αντικείμενο.

2. Για κλήση του διερμηνευτή/μεταφραστή solve/1 για να τρέξει κάποιους στόχους του προγράμματος-αντικείμενο, π.χ. monopati/3, τους οποίους θα δίνει ο χρήστης.

3. Για έξοδο από το μεταπρόγραμμα.

Εάν ο χρήστης επιλέξει

1 θα μπαίνει στον εξής κατάλογο επιλογών.

a. Για αλλαγής κάποιας πρόταση του προγράμματος.

b. Για εισαγωγή κάποιας νέας πρότασης στο πρόγραμμα- αντικείμενο.

c. Για διαγραφή κάποιας πρότασης από το πρόγραμμα-αντικείμνεο.

d. Για έξοδο από τη διαδικασία ενημέρωσης τους προγράμματος αντικείμενο. Εάν ο χρήστης επιλέξει

2 θα μπαίνει στον εξής κατάλογο επιλογών

a. Για εισαγωγή στόχου τον οποίο θα διερμηνεύσει ο διερμηνευτής solve/1. Ο χρήστης θα δίνει το στόχο μέσω της διεπικοινωνίας του μεταπρογράμματος σας και το μεταπρόγραμμα θα καλεί τον διερμηνευτή solve/1 ο οποίος θα διερμηνεύει το στόχο.

b. Για έξοδο από τη διαδικασία εκτέλεσης στόχων του προγράμματος-αντικείμενο.

Για διερμηνευτή/μεταφραστή να χρησιμοποιήσετε τον διερμηνευτή του προγράμματος 9.19 του βιβλίου σας και εφόσον χρειάζεται να τον τροποποιήσετε ή και να τον επεκτείνετε

Λυση

Το αρχείο Exercise\_4.pl περιεχει το προγραμμα που δημιουργει το μενου επιλογων.

:- dynamic clause2/2 ,solve/1 .

:- include('original\_programm\_exercise4.pl'). % My main programm

% MENU

menu:-

save,

write('Main Menu'),nl,

write('Select 1: Program Update.'),nl,

write('Select 2: Run Goals.'),nl,

write('Select 3 or other : Exit.'),nl,

read(Num),nl,

choice(Num).

choice(1) :- choice1. %Actions

choice(2) :- choice2.

choice(Num) :- write('EXIT').

%Choice 1

choice1:-

write('Program Update'),nl,

write('Select a: Change a clause.'),nl,

write('Select b: Insert new clause.'),nl,

write('Select c: Delete a clause'),nl,

write('Select d: Main menu'),nl,

read(Ch),nl,

choice1\_1(Ch).

% REPLACE A CLAUSE

choice1\_1(a):-

write('Give clause to replace'),nl,

read(Old\_clause),nl,

write('Give new clause like : [Goal , Body]'),nl,

read(New\_clause),nl,

retract(clause2(Old\_clause , \_)),

assert\_new(New\_clause),

save,menu.

assert\_new([H|[H1|T]]) :- asserta(clause2(H,H1)).

% INSERT A CLAUSE

choice1\_1(b):-

write('Give the new clause'),nl,

read(New\_clause),nl,

assert(clause2(New\_clause,true)),nl,

save,menu.

% DELETE A CLAUSE

choice1\_1(c):-

write('Give clause to delete'),nl,

read(Del\_clause),nl,

retract(clause2(Del\_clause,\_)),nl,

save,menu.

choice1\_1(d):- menu.

%Choice 2 RUN GOALS

choice2:-

write('Run Goals'),nl,

write('Select a: Run a clause.'),nl,

write('Select b: Main Menu.'),nl,

read(Ch),nl,

choice2\_1(Ch).

choice2\_1(a):-

write('Give a goal'),nl,

read(Goal),

solve(Goal),nl,

fail,

menu.

choice2\_1(b):- menu.

save:- tell('C:/Users/sfina/OneDrive/Desktop/final\_programm\_exercise4.pl'),

retractall(solve),

retractall(clause2),

listing(solve),

listing(clause2),

Told.

Το αρχειο original\_programm\_exercise4.pl περιεχει το αντικειμενο προγραμμα με το μεταπρογραμμα ( δηλ. Οι εντολες που πιθανόν να τροποποιηθουν)

solve(true) :- !.

solve((A, B)) :- !,

solve(A),

solve(B).

solve(Goal) :-

\+(predicate\_property(Goal,built\_in)),

clause2(Goal,Body),

solve(Body).

solve(Goal) :-

predicate\_property(Goal,built\_in),

call(Goal).

clause2(monopati1(X,X,Monopati,Monopati),true).

clause2(monopati1(X,Z,Monopati,Teliko\_monopati),(akme(X,Y),\+(member(Y,Monopati)),append(Monopati,[Y],Neo\_monopati),monopati1(Y,Z,Neo\_monopati,Teliko\_monopati))).

clause2(monopati(X, Y, Monopati), (monopati1(X, Y, [X], Monopati),write(Monopati))).

clause2(member(X,[H|T]),(X=H)).

clause2(member(X,[H|T]),(X\=H , member(X,T))).

clause2(append([],L2,L2),true).

clause2(append([H1|T1], L2, [H1|T3]),(append(T1,L2,T3))).

clause2(akme(a, c), true).

clause2(akme(c, a), true).

clause2(akme(c, e), true).

clause2(akme(c, f), true).

clause2(akme(b, c), true).

clause2(akme(b, d), true).

clause2(akme(d, f), true).

clause2(akme(d, g), true).

Και το αρχειο final\_programm\_exercise4.pl περιεχει το original\_programm μετα την τροποποιηση του.

solve(true) :- !.

solve((A,B)) :- !,

solve(A),

solve(B).

solve(A) :-

\+predicate\_property(user:A,built\_in),

clause2(A, B),

solve(B).

solve(A) :-

predicate\_property(user:A, built\_in),

call(user:A).

clause2(monopati1(A,A,B,C), C=B).

clause2(monopati1(A,B,C,D), (akme(A,E),\+member(E,C),append(C,[E],F),monopati1(E,B,F,D))).

clause2(monopati(A,B,C), (monopati1(A,B,[A],C),write(C))).

clause2(member(A,[B|\_]), A=B).

clause2(member(A,[B|C]), (A\=B,member(A,C))).

clause2(append([],A,A), true).

clause2(append([A|B],C,[A|D]), append(B,C,D)).

clause2(akme(a,c), true).

clause2(akme(c,e), true).

clause2(akme(c,f), true).

clause2(akme(b,c), true).

clause2(akme(b,d), true).

clause2(akme(d,f), true).

clause2(akme(d,g), true).

clause2(akme(e,a), true).

Παρακατω η υλοποιηση των ερωτηματων απο το SICStus ( οι στοχοι που υλοποιούνται ξεκινάνε απο τη μέση της 2ης φωτογραφιας)





