CSC4504/Prolog. : Formal Languages & Applications

J Paul Gibson, A207

paul.gibson@telecom-sudparis.eu

http://www-public.it-sudparis.eu/~gibson/Teaching/CSC4504/

Prolog - Answers
~/gibson/Teaching/CSC4504/L8-Prolog-Answers.pdf
Check if there is a path between two nodes in a directed graph

Encoding

edge(n,m) to represent an arc from n to m.
1 ?- ['/Users/jpaulgibson/Documents/MyPrograms/MyProlog/pathsingraph.pl'].
% /Users/jpaulgibson/Documents/MyPrograms/MyProlog/pathsingraph.pl compiled 0.00 sec, 11 clauses true.

2 ?- listing(edge).
edge(1, 2).
edge(2, 5).
edge(4, 5).
edge(2, 3).
edge(3, 1).
true.

3 ?- listing(path).
path(A, B, D) :-
    travel(A, B, [A], C),
    reverse(C, D).
true.

4 ?- listing(travel).
travel(C, A, B, [A | B]) :-
    edge(C, A).
travel(A, C, D, E) :-
    edge(A, B),
    B\=:=C,
    \+ member(B, D),
    travel(B, C, [B | D], E).
true.

Question: why reverse the list here?

Question: do you understand \+ ???
6 ?- path(X,Y,Z).
X = 1,
Y = 2,
Z = [1, 2]
X = 2,
Y = 5,
Z = [2, 5]
X = 4,
Y = 5,
Z = [4, 5]
X = 2,
Y = 3,
Z = [2, 3]
X = 3,
Y = 1,
Z = [3, 1]

X = 1,
Y = 5,
Z = [1, 2, 5]
X = 1,
Y = 3,
Z = [1, 2, 3]
X = Y, Y = 1,
Z = [1, 2, 3, 1]
X = 2,
Y = 1,
Z = [2, 3, 1]
X = Y, Y = 2,
Z = [2, 3, 1, 2]
X = 3,
Y = 2,
Z = [3, 1, 2]
X = 3,
Y = 5,
Z = [3, 1, 2, 5]
X = Y, Y = 3,
Z = [3, 1, 2, 3]
false.
2 ?- listing(arc).
   :- dynamic arc/2.

   arc(1, 2).
   arc(2, 3).
   arc(3, 1).
   arc(2, 5).
   arc(5, 5).
   arc(4, 5).

   true.

3 ?- listing(chemin).
   chemin(A, B) :-
      arc(A, B), !.
   chemin(A, B) :-
      A \= B,
      arc(A, C),
      retract(arc(A, C)),
      chemin(C, B).

   true.

/**
 * A solution using retract to
 * dynamically alter the database of
 * facts
 * (code from Arthur Barnoi)
 */

Do you understand this solution:

the cut?

the retract?
Testing the solution

11 ?- listing(arc).
:- dynamic arc/2.

arc(1, 2).
arc(2, 3).
arc(3, 1).
arc(2, 5).
arc(5, 5).
arc(4, 5).
true.

12 ?- chemin(X,Y).
X = 1,
Y = 2.

Question: why does it find only 1 path?
More tests

14 ?- chemin(1,2).
true.

15 ?- listing(arc).
:- dynamic arc/2.

arc(1, 2).
arc(2, 3).
arc(3, 1).
arc(2, 5).
arc(5, 5).
arc(4, 5).
true.

16 ?- chemin(2,3).
true.

17 ?- listing(arc).
:- dynamic arc/2.

arc(1, 2).
arc(2, 3).
arc(3, 1).
arc(2, 5).
arc(5, 5).
arc(4, 5).
true.

18 ?- chemin(1,3).
true.

19 ?- listing(arc).
:- dynamic arc/2.

arc(2, 3).
arc(3, 1).
arc(2, 5).
arc(5, 5).
arc(4, 5).

Question: do you see the problem?
A Possible Fix: Copy Arcs Into Temporary Store?

TO DO: can you fix/improve the code
Cuts, negation and negation as failure: “\+”

2 ?- listing(drunk).
  drunk(paul).
  drunk(fred).

true.

3 ?- listing(sober).
  sober(A) :-
   \+ drunk(A).

true.

4 ?- drunk(X).
  X = paul
  X = fred.

Solve the goal sober(paul) by trying to solve drunk(paul).

This fails, so sober(paul) fails.

But, sober(jim) succeeds

5 ?- sober(paul).
  false.

6 ?- sober(jim).
  true.

Question: what will we get if we query ?- sober(X).
Cuts, negation and negation as failure: “\+”

**Question:** what will we get if we query `- sober(X).`?

?- sober(X).
false

So “\+” is not the same as logical negation, where not(not)(bool) = bool

\+(\+(drunk(X)))) is not identical to drunk(X)
since the former will succeed with **X unbound**
while the latter will succeed with **X bound**, in the first instance, to **jim**.

Thus, eg, only use `\+ drunk(X)` if **X** is bound when it is called