Assignment 4 | HY-487

Stivaktakis Giorgos | csd4300

Exercise 1

No we cant use just 1 rule. We will have an infinite loop because we dont have a ‘stop’ condition. It could work sometimes if we predefine all connections with interconnected/2 rather connected/2.

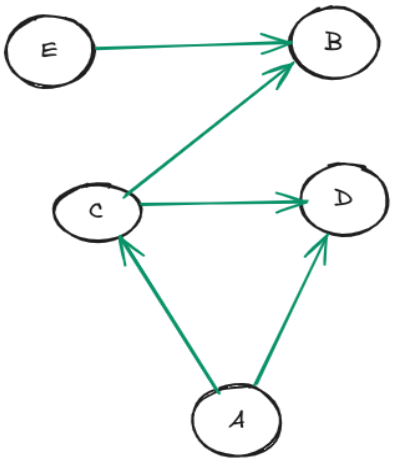
Exercise 2

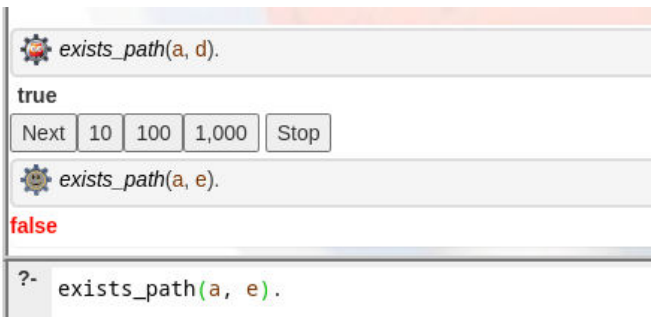
Exist_path code :

```
exists_path(X, Y) :- connect(X, Y).

exists_path(X, Y) :-
        connect(X, Z),
        exists_path(Z, Y).
```

Example using it in the following graph :



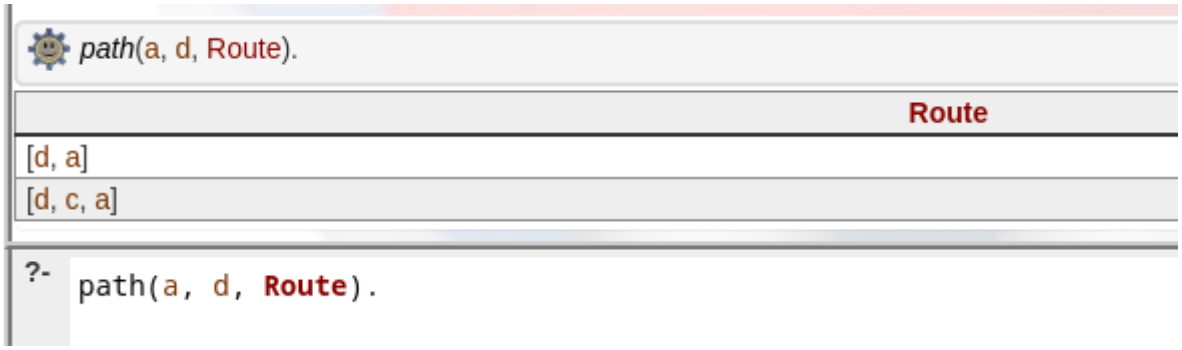


Exercise 3

Code for path (we add each visited node to a list in order to prevent cycles) :

```
path(Start, End, Path) :-
    path(Start, End, [Start], Path).
path(Start, End, Visited, [End|Visited]) :-
    connect(Start, End),
    \+ member(End, Visited).
path(Start, End, Visited, Path) :-
    connect(Start, Mid),
    Mid \== End,
    \+ member(Mid, Visited),
    path(Mid, End, [Mid|Visited], Path).
```

Results using it in graph from Exercise 2 :



Exercise 4

Same idea as exercise 3 ,although this time using the interconnected Rule (directed graph) and also calculate the current cost after visiting each node :

```
cost_path(Start, End, Path, Cost) :-
    cost_path(Start, End, [Start], 0, RevPath, Cost),
    reverse(RevPath, Path).

cost_path(Start, End, Visited, CurrCost, [End|Visited], TotalCost) :-
    interconnected(Start, End, Cost),
    TotalCost is CurrCost + Cost.

cost_path(Start, End, Visited, CurrCost, Path, TotalCost) :-
    interconnected(Start, Mid, Cost),
    Mid \== End,
    \+ member(Mid, Visited),
    NewCost is CurrCost + Cost,
    cost_path(Mid, End, [Mid|Visited], NewCost, Path, TotalCost).
```

Results example :