Concepts of Programming Languages, CSCI 305, Fall 2021
Lab 6, Prolog – tracing programs
Section 12.2.4 Search/Execution Order, Oct. 22

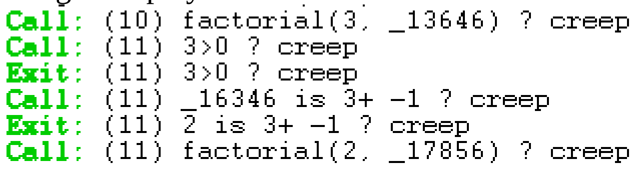
Commands:
```
?- trace.   % To turn off: ?- notrace.
```

Standard ports:

call, exit, redo, fail

creep - continue

The tracer displays the port name, the current depth of the recursion and the goal. For example, when tracing is on, at the query:
```
?- factorial(3,X).
```
the following is displayed:



```
Call: (10) factorial(3, _13646) ? creep
Call: (11) 3>0 ? creep
Exit: (11) 3>0 ? creep
Call: (11) _16346 is 3+ -1 ? creep
Exit: (11) 2 is 3+ -1 ? creep
Call: (11) factorial(2, _17856) ? creep
```

1. Enter and run the following program:

```
factorial(0,1).
factorial(N,Ans) :-
        N>0,
        K is N-1,
        factorial(K, SubAns),
        Ans is SubAns * N.
```

2. Turn on tracing and execute factorial(3,X). Make sense out of the “calls”, “exits”, levels and goals, so that given a snippet of code and a query, you could tell how Prolog would resolve the query.

3. The following database represents unidirectional edges in a graph.

```
% Specify edges.
edge(a, b).
edge(a, c).
edge(a, d).
edge(b, e).
edge(e, f).
edge(e, g).
edge(g, h).
edge(g, i).
edge(d, j).
```

4. Enter the above facts into the database. Define a path function which takes a source and destination vertex and returns true if a path exists between the source and destination, and false otherwise. (If source=destination, return true).

5. Resolving the query

```
?- path(a,j).
```

   will use the "fail" and "redo" ports, in addition to "calls", "exits". Understand the ports, levels and goals, so that given a snippet of code and a query, you could tell how Prolog would resolve the query.

6. Add code to the above so that the vertices traversed in the path is returned.

7. Towers of Hanoi: The objective of this famous puzzle is to move N disks from the left peg to the right peg using the center peg as an auxiliary holding peg. At no time can a larger disk be placed upon a smaller disk. The following diagram depicts the starting setup for N=3 disks.

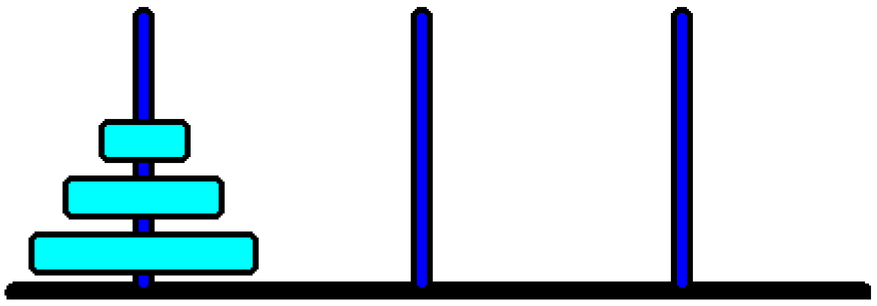


Fig. 2.3

This problem can be solved in Prolog using only two clauses:

1. A clause to move 1 disc from peg X to peg Y, using peg Z as a helper.
2. A clause to move more than one disc from peg X to peg Y, using peg Z as a helper.

Here is the first clause:

```
move(1,X,Y,_) :-
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
```

Complete this program.

8. Try your code for N=3 and verify that it works.

https://www.cpp.edu/~jrfisher/www/prolog_tutorial/2_3.html