Vivian Do
CSC 4330
Oct 09 2023

Homework 5

1.

Consider the (directed) network depicted here. We could represent this network with the following Prolog statements:

```
link(a,b).
link(a,c).
link(b,c).
link(b,d).
link(c,d).
link(d,e).
link(d,f).
link(e,f).
link(f,g).
```

Now, given this network, we say that there is a "connection" from a node "X" to a node "Y" if we can get from "X" to "Y" via a series of links, for example, in this network, there is a connection from "a" to "d", and a connection from "c" to "f", etc.

a. (30 points). Formulate the appropriate Prolog rule "connection(X,Y)" which is true if (and only if) there is a "connection" from "X" to "Y" as described above --- note that this rule will be recursive. Test this rule out on the above network, to see if it is working correctly.

Once it is working correctly, you will note that, e.g., the query "connection(a,e)." will give "true" multiple times. This means something, actually.

b. (10 points). How many times is "true" given for the query "connection(a,e).", and what does this mean?

a)

```
connection(X, Y) :- link(X, Y).
connection(X, Y) :- link(X, Z), connection(Z, Y).
```

b)

"true" is given 3 times for the query "connection(a, e)"

- this query returns "true" multiple times

This means that there are at least 3 different paths linking "a" to "e"

- each path represents a connection between these two nodes

2. (30 points).

Remember the "init" function of Haskell? It can be defined in Haskell as:

```
init :: [a] -> [a]
init [_] = []
init (x:xs) = x : init xs
```

Formulate the appropriate Prolog rule "init(Xs,Ys)" which is true if (and only if) "Ys" is the initial part of "Xs". That is, the query "init([1,2,3,4,5],[1,2,3,4])." would evaluate to "true", or the query "init([1,2,3,4,5],Ys)." would give "Ys = [1,2,3,4]".

Hint: this rule will be recursive, and will have a corresponding base case and recursive case, analogous to the base case and recursive case, respectively, of the above Haskell function --- the main difference being that the Prolog rule "init(Xs,Ys)" is a predicate, not a function.

```
init([], []).
init([_], []).
init([X | Xs], [X | Ys]) :- init(Xs, Ys).
```

3. (30 points).

Remember the "zip" function of Haskell? It can be defined in Haskell as:

```
zip :: [a] -> [b] -> [(a,b)]
zip _ [] = []
zip [] _ = []
zip (x:xs) (y:ys) = (x,y) : zip xs ys
```

Formulate the appropriate Prolog rule "zip(Xs,Ys,Zs)" which is true if (and only if) "Zs" is the result of "zipping" together "Xs" and "Ys". That is, the query "zip([1,2,3],[4,5],[(1,4),(2,5)])." would evaluate to "true", or the query "zip([1,2],[4,5,6],Zs)." would give "Zs = [(1,4),(2,5)]". Hint: again, this rule will have two base cases and a recursive case analogous to the above Haskell function.

```
zip([], _, []).
zip(_, [], []).
zip([X | Xs], [Y | Ys], [(X, Y) | Zs]) :- zip(Xs, Ys,
Zs).
```