

Re: CLP(FD) Prolog: help needed with a simple problem

Source: <http://coding.derkeiler.com/Archive/Prolog/comp.lang.prolog/2005-03/0312.html>

From: Bart Demoen (*bmd_at_cs.kuleuven.ac.be*)

Date: 03/29/05

Date: Tue, 29 Mar 2005 12:42:54 +0200

Chema wrote:

- > *I got the point, but the problem is: I've to impose this kind of*
- > *condition for each subgraph. Consider an undirected graph $G=(V,E)$*
- > *with $E = (a,b), (b,c), (c,d), (d,h), (e,h), (a,e), (b,e), (b,h),$*
- > *$(c,h), (e,f), (f,h), (h,g), (d,g), (f,g)$*
- > *You see, I should include at least two constrains for each cycle.*
- > *This is an NP hard problem, and this constrains grow in an*
- > *esponential way... I can't do that "manually".*

Let's first get things straight: you are trying to solve the problem
"given a graph, find a spanning tree"

That problem is not NP hard. The algorithms of Kruskal and Prim give you a minimal spanning tree in a weighted graph in polynomial time. You can use these algorithms with weight 1 for each edge. Those algorithms avoid cycles, but not by trying to represent each of them explicitly.

You want to solve CLP for solving that problem – fine. Don't set up constraints for cycles. There is some simple piece of graph theory that says: for a connected graph with n nodes, the spanning tree contains $(n-1)$ edges and of course also the n nodes.

So the following will work: give a distinct variable name to every edge, say E_i , and keep track for each node V_i on which edges it lies;

now generate the constraints

```
all  $E_i$  in [0,1]
the sum of all  $E_i$  equals  $n-1$ 
for every  $V_i$  there is some  $E_j$  (which  $V_i$  lies on) which is 1
```

That's the generalisation of what I wrote for the very simple example.
The structure of your program could be something like:

```
st(Edges) :-
    all_edges(Edges),
    all_nodes_on_edges(Edges,NodesOnEdges),
```

comp.lang.prolog: Re: CLP(FD) Prolog: help needed with a simple problem

```
length(NodesOnEdges,N),  
Edges in 0..1,  
sum(Edges,N-1),  
foreach(Node-ItsEdges,member(Node-ItsEdges,NodesOnEdges),(sum(ItsEdges,M),M>0)),  
label(Edges).
```

Code not tested. You still need to keep track of the association between the edge variables and the actual edges (I ignored that completely above).

Cheers

Bart Demoen