Apéndice C – Otros códigos

Aset-Solver.

Se le pidió a Veena Mellarkod si el problema del mínimo árbol abarcador podía ser codificado en Aset . A continuación se incluye la respuesta.

De: Veena Mellarkod [mailto:mellarko@redwood.cs.ttu.edu]
Enviado el: Viernes, 05 de Marzo de 2004 05:25 p.m.
Para: lopez chalini leticia
Asunto: Re: more examples for Aset-Solver
Hi Leticia,
Sorry for not replying all this time. It is a busy semester. The aggregates in ASet-Prolog involve only sets within a model. Finding the minimal of spanning trees involves generating all the spanning trees and looking between them to find the minimal one. Like the weak constraints in dlv and the minimize and maximize statements in smodels, ASet does not have any function which compares two models. CR-Prolog, can write such statements. CR-Prolog is A-Prolog with consistency restoring rules. You can find the syntax of the language on the KRLAB website. There is also a solver for it on the software section of the website. I wrote the problem of finding minimal spanning trees in cr-prolog:

The program works well when there are no conflict minimal spanning trees in the graph. So if the program returns models for the graph then they are the minimal spanning trees. If it doesn't return any models then it may OR may not have minimal spanning trees. I have been thinking of rewriting the preferences in the program to make it work for any graph.It is a little difficult, the preferences need to be written in a defeasible way. I can give you the program in cr-prolog when I am done with it, if you want.
Regards,
Veena.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%    MINIMUM SPANNING TREE %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The nodes in the graph.
node(a).
node(b).
node(c).
node(d).
node(e).

% edge(X,Y,C): there is an edge from node X to node Y
% with a cost C in the graph.
edge(a,b,4).
edge(a,c,3).
edge(c,b,2).
edge(c,d,3).
edge(b,e,4).
```

```
edge(d,e,5).

% The root of the spanning tree.
root(a).

% Used for grounding purposes
cost(S) :- edge(A,B,S).
#domain node(X;Y;X1;Y1).
#domain cost(C;C1).

% There cannot be two edges in the tree which
% go to the same node. (no cycles)
:- in_tree(X,Y,C), in_tree(X1,Y,C1), neq(X,X1), edge(X,Y,C), edge(X1,Y,C1).

% The root is reached.
reached(X) :- root(X).

% A node X is reached if there is a node Y which is
% reached and there is an edge in tree from Y to X.
reached(X) :- reached(Y), in_tree(Y,X,C), edge(Y,X,C).

% All nodes should be reached.:- node(X), not reached(X).
% If there is an edge from X to Y then it may be in tree
r(X,Y,C) : in_tree(X,Y,C) +- edge(X,Y,C).

% If there are two edges to Y, prefer the less costly one.
prefer(r(X,Y,C),r(X1,Y,C1)) :- edge(X,Y,C), edge(X1,Y,C1),C < C1.
```

DLV.

El siguiente código de Nicola Leone no pudo ser probado porque no ha sido liberada la versión que soporte #min aggregate function de la forma que la usa en este programa.

Date: Fri Feb 13 11:18:12 2004
From: penichet@mail.udlap.mx
To: osorio@cse.Buffalo.EDU
Subject: Fwd: Re: ACM problem in DLV...
Mauricio,
Aqui esta lo que me contesto Nicola.. esta interesante la forma en que el codifica el problema. espero que te sirva...
Saludos.

Nicola Leone <leone@mat.unical.it> wrote:
Yes, the ACM FastFood problem can be modeled rather nicely in DLV:

```
%=====================================
%Prof. Nicola Leone
% maxint must not be lower than numberOfRestaurants, numberOfDepots, and
%the maximum restaurant location
```

```
% (i.e., distance from the Head Quarter).
% PLEASE DONT SET IT HIGHER THAN NEEDED!!! (for the sake of efficiency)

#maxint=27.

numberOfRestaurants(6).
numberOfDepots(3).

% we identify each restaurant with each location
% restaurantLocation(r) means that there is a restaurant "r" kilometers far from the Headquarter
% one line (location) for each restaurant is to be specified.

restaurantLocation(5).
restaurantLocation(6).
restaurantLocation(12).
restaurantLocation(19).
restaurantLocation(20).
restaurantLocation(27).

% we identify each restaurant with each location
% restaurantLocation(r) means that there is a restaurant "r" kilometers far from the Headquarter

%Guess a (restaurant) location for each depot
depotLocation(R) v -depotLocation(R) :- restaurantLocation(R).

% The number of depots equals precisely to the specified "numberOfDepots"
:- numberOfDepots(N), not #count{D: depotLocation(D)} = N.

% serves(D,R,Dist) means that depot "D" serves Restaurant "R",
%which is "Dist" kilometers far away from "R".
serves(D,R,Dist) :- restaurantLocation(R), depotLocation(D),
                    distance(R,D,Dist),
                Dist = #min {Y : depotLocation(D1), distance(D1,R,Y) }.

% Minimize the sum of the shipping distances
:~ serves(D,R,Dist). [Dist:]

%distance(X,L,Y) is true if Y = |X-L|
distance(X,L,Y):- #int(X), #int(L), X>L,  X=L+Y.
distance(X,L,Y):- #int(X), #int(L), X<=L, L=X+Y.

To compute the best assignements for the depot locations
and the restaurants they serve, please run

>dlv instance fastfood.dlv -pfilter=depotLocation,serves
>
>DLV [build DEV/Jan 20 2004 gcc 3.2.2 (Mandrake Linux 9.1 3.2.2-3mdk)]
>
```

>Best model: {depotLocation(6), depotLocation(19), depotLocation(27), serves(6,5,1), serves(6,6,0), serves(6,12,6), serves(19,19,0), serves(19,20,1), serves(27,27,0)}
>Cost ([Weight:Level]): <[8:1]>
>
>Best model: {depotLocation(6), depotLocation(20), depotLocation(27), serves(6,5,1), serves(6,6,0), serves(6,12,6), serves(20,19,1), serves(20,20,0), serves(27,27,0)}
>Cost ([Weight:Level]): <[8:1]>

It is worthwhile noting that our encoding uses the #min aggregate function, which is still experimental in the old DLV release (and does not work properly there).We'll release a new version of DLV with more stable aggregate functions within the next month.
However, if you tell us the operating system you are using, we can give you immediately the new version incorporating the improvements on the aggregates to test the encoding of the fastfood ACM problem.

Best Regards
Nicola

==========
Prof. Nicola Leone
Department of Mathematics
University of Calabria
87030 Rende (CS), Italy
phone: +39 0984 496433
fax: +39 0984 496410
email: leone@unical.it
http://www.mat.unical.it/leone/~home/
==========

Smodels.

Las siguientes indicaciones son para modificar el código de Smodels y permitir la salida de todos los modelos óptimos del problema 3.1 página 46.

>Subject: Re: knapsack`s example -- test stress
>From: Patrik Simons <patrik.simons@neotide.fi>
>To: Leticia López Chalini <ci098386@mail.pue.udlap.mx>
>Date: Fri, 30 Apr 2004 08:55:24 +0300
On Thu, 2004-04-29 at 20:36, Leticia López Chalini wrote:
Dear Patrik,
One last question,
I have tested knapsack`s example in Smodels and dlv with 49 objects. the Smodels output was 37 models but only one is the best with one profit = 381 and cost=100.
The dlv´s output was 6 models and all of them are optimized.
Is there any way to Smodels gives the 6 optimize models and not only one ?

Not without changing the source.
If you remove the lines:
```
   else if (r->next == 0)
```

```
      return true;
```
from Smodels::conflict () in smodels.cc (lines 216-217),
then smodels will output all optimal models.
--
Patrik
```
--- smodels.cc.orig     2004-04-30 08:54:27.000000000 +0300
+++ smodels.cc  2004-04-30 08:54:30.000000000 +0300
@@ -213,8 +213,8 @@
       return true;
     else if (r->minweight < r->minoptimum)
       return false;
-    else if (r->next == 0)
-      return true;
+  //    else if (r->next == 0)
+  //      return true;
   return false;
 }
```