```
 1/*********************************************
 2 * OPL 12.4 Model
 3 * Author: alexander
 4 * Creation Date: Apr 23, 2014 at 4:36:54 PM
 5 Lets us specify a set of states (represented)
 6 by edges and guarded transitions (required sets of
 7 seen objects)
 8 *********************************************/
 9using CP;
10
11tuple Edge {
12    int pred;
13    int succ; // in minNodeId .. maxNodeId (consistency
  needs to be enforced elsewhere)
14    {int} additionalPredecessors; // a guard expression
15};
16
17// this will be the parameterized content
18range edgeIds = 1..6;
19Edge edges[edgeIds] = [<1,2, {}>, <1,3, {}>,  <2,3, {}>,
  <3,2, {}>, <2, 4, {3}>, <3,4, {2}>];
20{int} seen = {1, 2};
21
22// the current decision variable specifies which edge to
  use
23dvar int edgeId in edgeIds;
24
25// the actual next content
26dexpr int nextNode = edges[edgeId].succ;
27
28// these helping data structures need not be changed
29// int edgePredecessorSum[e in edgeIds] =  sum(i in
  edges[e].additionalPredecessors) (i in seen);
30int edgePredecessorSum[e in edgeIds] =  card(edges
  [e].additionalPredecessors inter seen) ;
```

```
31 int predCount[e in edgeIds] = card( edges
   [e].additionalPredecessors);
32 int edgePredecessorValid[e in edgeIds] = (predCount[e]
   == edgePredecessorSum[e]);
33 int edgeIsValid[e in edgeIds] = ((edgePredecessorValid
   [e] == true) && (edges[e].pred in seen));
34 int edgeLeadsToUnseen[e in edgeIds] = (edges[e].succ not
   in seen);
35
36 // maximize nextNode;
37 maximize edgeId;
38
39 subject to {
40    // consistency (did not work with decision
   expression)
41    // there exists an edge to nextNode from seen such
   that I may take it!
42     edgeIsValid[edgeId] == true;
43    //nextNode == 3;
44
45    // I should not have seen it already
46    edgeLeadsToUnseen[edgeId] == true;
47
48    // this does not work for some reason
49    // edges[edgeId].succ not in seen;
50 }
51
52 {int} a = {1,1,2};
53 {int} b = {1,2};
54 int aLen = card(a);
55 int bLen = card(b);
56
57 execute {
58   writeln(edgeLeadsToUnseen);
59   writeln(aLen);
```

```
60  writeln(bLen);
61  writeln('--------');
62  writeln(nextNode);
63  writeln(edgeId);
64  writeln(edgePredecessorValid);
65  writeln(edges[edgeId].succ);
66}
```