CMPUT620: Topics In Knowledge Representation And Reasoning: Assignment 1

Cody Rosevear
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
rosevear@ualberta.ca

February 2nd 2018

1 Q 1 & 2

Both question 1 and 2 were completed. Code and a custom small test case are included in the appendix. The resultant output was correct.

2 Q 3

Code for the question is provided below:

```
%GENERATE
room(r1). room(r2). room(r3).
#const firstDayAm = 0. #const firstDayPm = 1. #const secondDayAm = 2.
#const secondDayPm = 3.
time(firstDayAm). time(firstDayPm). time(secondDayAm). time(secondDayPm).
session(a). session(b). session(c). session(d). session(e). session(f). session(g).
session(h). session(i). session(j). session(k).

%We want to schedule exactly 11 sessions
11{schedule(X, Y, Z): session(X), time(Y), room(Z)}11.

%DEFINE
occurs_at(S, X):- session(S), time(X), room(Z), schedule(S, X, Z).
occurs_in(S, Z):- session(S), time(X), room(Z), schedule(S, X, Z).

simultaneous(X, Y) :- session(X), session(Y), time(A), time(B), A == B,
occurs_at(X, A), occurs_at(Y, A).
simultaneous(X, Y, Z) :- session(X), session(Y), session(Z), time(A), time(B), A == B,
occurs_at(X, A), occurs_at(Y, A), occurs_at(Z, A).
```

```
simultaneous(X, Y, Z, W) :- session(X), session(Y), session(Z), session(W), time(A),
time(B), A == B, occurs_at(X, A), occurs_at(Y, A), occurs_at(Z, A), occurs_at(W, A).

%CONSTRAINTS
%No session can be scheduled at more than one time
:- time(X), time(Y), X != Y, session(S), occurs_at(S, X), occurs_at(S, Y).

%No session can be scheduled in more than one room
:- session(S), room(X), room(Y), X != Y, occurs_in(S, X), occurs_in(S, Y).

%Precedence constraint
:- session(A), session(B), precede(A, B), time(X), time(Y), X >= Y,
occurs_at(A, X), occurs_at(B, Y).

%Constraints on sessions that cannot occur together
:- simultaneous(a, j).
:- simultaneous(j, i).
:- simultaneous(i, e).
:- simultaneous(c, f).
:- simultaneous(f, g).
:- simultaneous(d, h).
:- simultaneous(b, d).
:- simultaneous(k, e).
:- simultaneous(a, g, e).
:- simultaneous(d, f, j).
:- simultaneous(b, i, h, j).
:- simultaneous(b, c, d, h).

%DISPLAY
#show schedule/3.
```

2.1 Q3 Test cases

The program was tested on 4 instances, all of which are reproduced below. The first two instances yielded the correct output. The third one was taking too long to solve, so the search process was terminated by the user. Therefore, I added an additional test case, whose output was also correct.

2.1.1 Q3 Test Case 1

```
precede(e,j).
precede(f,k).
precede(d,k).
schedule(a, firstDayAm, r2).
```

2.1.2 Q3 Test Case 2

```
precede(b,j).
precede(k,a).
precede(c,d).
precede(e,f).
schedule(b, firstDayAm, r3).
schedule(c, secondDayAm, r3).
```

2.1.3 Q3 Test Case 3

```
precede(j,i).
precede(a,e).
precede(e,f).
precede(d,e).
schedule(c, firstDayAm, r2).
schedule(a, secondDayAm, r1).
```

2.1.4 Q3 Test Case 4

```
precede(j,i).
precede(a,e).
precede(e,f).
precede(a,c).
precede(k,a).
```

3 Q 4

Code for the question is provided below:

```
%GENERATE
1{in(N,X) : num(N,X)}.

%TEST
:- 0 != #sum {X,N : in(N,X)}.

%DISPLAY
#show in/2.
```

3.1 Q4 Test cases

The program was tested on 4 instances, 3 of which are shown below. The 3rd is larger and so located in a separate file titled Q4test3.lp.

3.1.1 Q4 Test Case 1

```
Represents a problem instance that is satisfiable: answer is {-3, -2, 5}
num(-7). num(-3). num(-2).
num(5). num(8).
```

3.1.2 Q4 Test Case 2

```
Represents a problem instance that is not satisfiable
num(-7). num(-3). num(-2).
num(20). num(8).
```

3.1.3 Q4 Test Case 2

Represents a problem instance that is satisfiable, and exhibits the multiset property.

```
num(0,-7). num(1,-3). num(2,-3).
num(3,6). num(4,8).
```

3.1.4 Q4 Test Case 3

This instance is larger and took a little longer to solve, so the output was not checked by hand.

```
Answer: 1
in(10,13) in(17,149) in(30,17) in(37,40) in(69,168) in(99999,-387)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 71.949s (Solving: 69.67s 1st Model: 69.66s Unsat: 0.00s)
CPU Time     : 69.899s
```

4 Q5

Results for my local experiments for each problem are reproduced below. The experiments were run using problem encodings from the ASP-2 encoding column of the system track only table on the ASP competition 2013 website. All of the tested problem instances are also from the above mentioned website: they were taken from the sample instances column of the same table. The table can be found here: https://www.mat.unical.it/aspcomp2013/OfficialProblemSuite

4.1 Maximum Clique Problem Results

```
instance1
Optimization: 180
```

```
OPTIMUM FOUND

Models       : 46
  Optimum    : yes
Optimization : 180
Calls        : 1
Time         : 473.781s (Solving: 473.47s 1st Model: 0.00s Unsat: 4.05s)
CPU Time     : 452.062s

instance 2
Models       : 45
  Optimum    : yes
Optimization : 181
Calls        : 1
Time         : 1436.476s (Solving: 1436.19s 1st Model: 0.00s Unsat: 10.64s)
CPU Time     : 1360.179s

instance 3
Optimization: 181
OPTIMUM FOUND

Models       : 45
  Optimum    : yes
Optimization : 181
Calls        : 1
Time         : 1032.757s (Solving: 1032.43s 1st Model: 0.00s Unsat: 1.15s)
CPU Time     : 1011.235s

instance 4
Optimization: 187
OPTIMUM FOUND

Models       : 39
  Optimum    : yes
Optimization : 187
Calls        : 1
Time         : 986.242s (Solving: 985.92s 1st Model: 0.00s Unsat: 9.15s)
CPU Time     : 975.955s

instance 5
Optimization: 201
OPTIMUM FOUND

Models       : 50
  Optimum    : yes
Optimization : 201
```

```
Calls        : 1
Time         : 955.745s (Solving: 955.39s 1st Model: 0.00s Unsat: 27.95s)
CPU Time     : 942.816s
```

4.1.1 Discussion

All of the problem instances for the maximum clique problem took significantly longer to run than most of the other problem instances. This is likely due to the addition of an optimality constraint which requires greater search time to find a suitably optimal answer set for the given problem instance.

4.2 Tower Of Hanoi Results

```
instance 1
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 3.378s (Solving: 3.10s 1st Model: 3.09s Unsat: 0.00s)
CPU Time     : 3.370s

instance 2
SATISFIABLE
Models       : 1+
Calls        : 1
Time         : 11.552s (Solving: 11.30s 1st Model: 11.30s Unsat: 0.00s)
CPU Time     : 11.495s

instance3
SATISFIABLE
Models       : 1+
Calls        : 1
Time         : 23.324s (Solving: 22.99s 1st Model: 22.99s Unsat: 0.00s)
CPU Time     : 23.247s

instance 4
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 1998.913s (Solving: 1998.49s 1st Model: 1998.49s Unsat: 0.00s)
CPU Time     : 1333.716s

instance 5
SATISFIABLE
```

```
Models       : 1+
Calls        : 1
Time         : 959.298s (Solving: 958.85s 1st Model: 958.84s Unsat: 0.00s)
CPU Time     : 935.051s
```

4.2.1 Discussion

All of the problems were solved in a fairly short time frame. Also, all programs were satisfiable, which is to be expected, since the tower of hanoi always has a solution and what changes with the addition of more disks is simply the time taken to find the solution.

4.3 Knights Tour Results

```
instance 1
UNSATISFIABLE

Models       : 0
Calls        : 1
Time         : 4.556s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 4.434s

instance 2
UNSATISFIABLE

Models       : 0
Calls        : 1
Time         : 5.632s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 5.471s

instance 3
UNKNOWN

INTERRUPTED  : 1
Models       : 0+
Calls        : 1
Time         : 32645.062s (Solving: 32638.24s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 17916.086s

instance 4
UNSATISFIABLE

Models       : 0
Calls        : 1
Time         : 3.981s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 3.868s
```

```
instance 5
UNSATISFIABLE

Models       : 0
Calls        : 1
Time         : 4.634s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 4.619s
```

4.3.1 Discussion

All of the problem instances were solved in a short amount of time, with the exception of problem instance 3. The program was halted manually as it was taking too long to finish.

5 Appendix

5.1 Q1 & 2 Code

```
%GENERATE
#const n = 7.
#const m = 9.
#const size = 3.
size{in(X) : vertex(X)}size.

%DEFINE
%Arcs go both ways
arc(X, Y) :- vertex(X), vertex(Y), X != Y, arc(Y, X).

%Set membership
in(X) :- vertex(X), vertex(Y), X != Y, connected(X, Y), in(Y).
out(X) :- vertex(X), not in(X).

%Connections can be direct or transitive through other arcs
connected(X, Y) :- vertex(X), vertex(Y), X != Y, arc(X, Y).
connected(X, Z) :- vertex(X), vertex(Y), vertex(Z), X != Y, Y != Z, X != Z,
connected(X, Y), connected(Y, Z).

%TEST
:- vertex(X), vertex(Y), X != Y, in(X), in(Y), not connected(X, Y).
:- vertex(n), vertex(m), n != m, in(n), out(m).

%DISPLAY
#show in/1.
```

5.2 Q1 Test Case

```
vertex(0).
vertex(1).
arc(0,1).

vertex(2).
vertex(3).
vertex(4).
arc(2,3).
arc(3,4).

vertex(5).

vertex(7).
vertex(8).
arc(7,8).

vertex(9).
vertex(10).
vertex(11).
arc(9,10).
arc(10,11).
```