

## Project 2

## Problem 1:

My constraints did not require like and dislike to be explicitly mentioned as symmetric relations.

## Representation:

In a set `seat(x,y)`, `x` represents table number and `y` represents guest number.

## Test case1:

```
#const n = 3.
#const m = 3.

%test case1
like(1,9).
like(2,8).
like(3,7).
like(7,6).
like(8,4).
dislike(1,7).
dislike(1,8).
dislike(7,8).

%generate (table,guest) sets
m{seat(I, 1..m*n)}m :- I = 1..n.
%one person should occupy only one table
:- seat(T1,G), seat(T2,G), T1!=T2.
%remove tables which has 2 persons who dislike each other
:- seat(T,G1), seat(T,G2), dislike(G1,G2).
%remove tables that do not have both persons who like each other
:- seat(T1,G1), seat(T2,G2), like(G1,G2), T1!=T2.

#show seat/2.
```

## Online clingo editor:

Advanced Examples

Matching Salesperson

```

1  #const n = 3.
2  #const m = 3.
3
4  %test case1
5  like(1,9).
6  like(2,8).
7  like(3,7).
8  like(7,6).
9  like(8,4).
10 dislike(1,7).
11 dislike(1,8).
12 dislike(7,8).
13
14
15 %generate (table,guest) sets
16 m{seat(I, 1..m*n)}m :- I = 1..n.
17 %one person should occupy only one table
18 :- seat(T1,G), seat(T2,G), T1!=T2.
19 %remove tables which has 2 persons who dislike each other
20 :- seat(T,G1), seat(T,G2), dislike(G1,G2).
21 %remove tables that do not have both persons who like each other
22 :- seat(T1,G1), seat(T2,G2), like(G1,G2), T1!=T2.
23
24 #show seat/2.
```

Configuration: reasoning mode enumerate all ▼

## Output:

```
clingo version 4.5.3
Solving...
Answer: 1
seat(2,1) seat(1,2) seat(3,3) seat(1,4) seat(2,5) seat(3,6) seat(3,7) seat(1,8) seat(2,9)
Answer: 2
seat(3,1) seat(1,2) seat(2,3) seat(1,4) seat(3,5) seat(2,6) seat(2,7) seat(1,8) seat(3,9)
Answer: 3
seat(1,1) seat(3,2) seat(2,3) seat(3,4) seat(1,5) seat(2,6) seat(2,7) seat(3,8) seat(1,9)
Answer: 4
seat(1,1) seat(2,2) seat(3,3) seat(2,4) seat(1,5) seat(3,6) seat(3,7) seat(2,8) seat(1,9)
Answer: 5
seat(2,1) seat(3,2) seat(1,3) seat(3,4) seat(2,5) seat(1,6) seat(1,7) seat(3,8) seat(2,9)
Answer: 6
seat(3,1) seat(2,2) seat(1,3) seat(2,4) seat(3,5) seat(1,6) seat(1,7) seat(2,8) seat(3,9)
SATISFIABLE

Models      : 6
Calls       : 1
```

clingo version 4.5.3

Solving...

Answer: 1

seat(2,1) seat(1,2) seat(3,3) seat(1,4) seat(2,5) seat(3,6) seat(3,7) seat(1,8) seat(2,9)

Answer: 2

seat(3,1) seat(1,2) seat(2,3) seat(1,4) seat(3,5) seat(2,6) seat(2,7) seat(1,8) seat(3,9)

Answer: 3

seat(1,1) seat(3,2) seat(2,3) seat(3,4) seat(1,5) seat(2,6) seat(2,7) seat(3,8) seat(1,9)

Answer: 4

seat(1,1) seat(2,2) seat(3,3) seat(2,4) seat(1,5) seat(3,6) seat(3,7) seat(2,8) seat(1,9)

Answer: 5

seat(2,1) seat(3,2) seat(1,3) seat(3,4) seat(2,5) seat(1,6) seat(1,7) seat(3,8) seat(2,9)

Answer: 6

seat(3,1) seat(2,2) seat(1,3) seat(2,4) seat(3,5) seat(1,6) seat(1,7) seat(2,8) seat(3,9)

SATISFIABLE

Models : 6

Calls : 1

Time : 0.237s (Solving: 0.01s 1st Model: 0.01s Unsat: 0.00s)

CPU Time : 0.000s

## Test case 2:

```
#const n =3.
#const m = 4.

%test case2
like(1,2).
like(1,4).
like(2,5).
like(10,5).
like(6,7).
like(5,7).
dislike(2,3).
dislike(4,6).
dislike(1,8).
dislike(8,9).

%generate (table,guest) sets
m{seat(I, 1..m*n)}m :- I =1..n.
%one person should occupy only one table
:- seat(T1,G), seat(T2,G), T1!=T2.
%remove tables which has 2 persons who dislike each other
:- seat(T,G1), seat(T,G2), dislike(G1,G2).
%remove tables that do not have both persons who like each other
:- seat(T1,G1), seat(T2,G2), like(G1,G2), T1!=T2.

#show seat/2.
```

## Online clingo editor:

```
1  #const n =3.
2  #const m = 4.
3
4  %test case2
5  like(1,2).
6  like(1,4).
7  like(2,5).
8  like(10,5).
9  like(6,7).
10 like(5,7).
11 dislike(2,3).
12 dislike(4,6).
13 dislike(1,8).
14 dislike(8,9).
15
16
17 %generate (table,guest) sets
18 m{seat(I, 1..m*n)}m :- I =1..n.
19 %one person should occupy only one table
20 :- seat(T1,G), seat(T2,G), T1!=T2.
21 %remove tables which has 2 persons who dislike each other
22 :- seat(T,G1), seat(T,G2), dislike(G1,G2).
23 %remove tables that do not have both persons who like each other
24 :- seat(T1,G1), seat(T2,G2), like(G1,G2), T1!=T2.
25
26 #show seat/2.
```

Configuration: reasoning mode

## Output:

```

clingo version 4.5.3
Solving...
UNSATISFIABLE

Models      : 0
Calls       : 1
Time        : 0.152s (Solving: 0.05s 1st Model: 0.00s Unsat: 0.05s)
CPU Time    : 0.000s

```

clingo version 4.5.3

Solving...

UNSATISFIABLE

Models : 0

Calls : 1

Time : 0.152s (Solving: 0.05s 1st Model: 0.00s Unsat: 0.05s)

CPU Time : 0.000s

## Problem 2:

### Program:

%assign codes to each of the firstnames

firstname(1,engles).

firstname(2,foster).

firstname(3,gunter).

firstname(4,halevy).

%assign codes to each of the lastnames

lastname(5,abner).

lastname(6,chuck).

lastname(7,duane).

lastname(8,bruce).

%assign codes to each of the pets

pet(9,iguana).

pet(10,jackal).

pet(11,kingcobra).

pet(12,llama).

%generate (firstname,lastname,pet) sets

4{owns(1..4,5..8,9..12)}4.

%2 persons can't have same 1st name

:-owns(I,J,K), owns(I,J1,K1), J!=J1, K!=K1.

%2 persons can't have the same last name

:-owns(I,J,K), owns(I1,J,K1), I!=I1, K!=K1.

%2 persons can't have same pet

:-owns(I,J,K), owns(I1,J1,K), I1!=I, J1!=J.

%a firstname-lastname combination can have only 1 pet

:-owns(I,J,K), owns(I,J,K1), K!=K1.

%a lastname-pet combination can have only one first name

:-owns(I1,J,K), owns(I,J,K), I1!=I.

%a firstname-pet combination can have only one lastname

:-owns(I,J1,K), owns(I,J,K), J1!=J.

%engles and abner are different persons

:-owns(1,5,K).

%foster and abner are different persons

```
:-owns(2,5,K).
%chuck doesn't own iguana
:-owns(1,6,9).
%duane doesn't own iguana
:-owns(1,7,9).
%foster doesn't own jackal
:-owns(2,J,10).
%foster doesn't own kingcobra
:-owns(2,J,11).
%duane doesn't own llama
:-owns(1,7,12).
%gunter and abner are different persons
:-owns(3,5,K).
%abner doesn't own kingcobra
:-owns(1,5,11).
%foster and bruce are different persons
:-owns(2,8,K).
%halevy doesn't own iguana
:-owns(4,J,9).
%engles and duane are different persons because their pet names are different
:-owns(1,7,K).
%decode numbers to names
own(I,J,K) :- owns(X,Y,Z), firstname(X,I), lastname(Y,J), pet(Z,K).
%show only own sets
#show own/3.
```

## Online clingo editor:

```
1 %assign codes to each of the firstnames
2 firstname(1,engles).firstname(2,foster).firstname(3,gunter).firstname(4,halevy).
3 %assign codes to each of the lastnames
4 lastname(5,abner).lastname(6,chuck).lastname(7,duane).lastname(8,bruce).
5 %assign codes to each of the pets
6 pet(9,iguana).pet(10,jackal).pet(11,kingcobra).pet(12,llama).
7 %generate (firstname,lastname,pet) sets
8 4{owns(1..4,5..8,9..12)}4.
9 %2 persons can't have same 1st name
10 :-owns(I,J,K), owns(I,J1,K1), J!=J1, K!=K1.
11 %2 persons can't have the same last name
12 :-owns(I,J,K), owns(I1,J,K1), I!=I1, K!=K1.
13 %2 persons can't have same pet
14 :-owns(I,J,K), owns(I1,J1,K), I1!=I, J1!=J.
15 %a firstname-lastname combination can have only 1 pet
16 :-owns(I,J,K), owns(I,J,K1), K!=K1.
17 %a lastname-pet combination can have only one first name
18 :-owns(I1,J,K), owns(I,J,K), I1!=I.
19 %a firstname-pet combination can have only one lastname
20 :-owns(I,J1,K), owns(I,J,K), J1!=J.
21 %engles and abner are different persons
22 :-owns(1,5,K).
23 %foster and abner are different persons
24 :-owns(2,5,K).
25 %chuck doesn't own iguana
26 :-owns(I,6,9).
27 %duane doesn't own iguana
28 :-owns(I,7,9).
29 %foster doesn't own jackal
30 :-owns(2,J,10).
31 %foster doesn't own kingcobra
32 :-owns(2,J,11).
33 %duane doesn't own llama
34 :-owns(I,7,12).
35 %gunter and abner are different persons
36 :-owns(3,5,K).
37 %abner doesn't own kingcobra
38 :-owns(I,5,11).
39 %foster and bruce are different persons
40 :-owns(2,8,K).
41 %halevy doesn't own iguana
42 :-owns(4,J,9).
43 %engles and duane are different persons because their pet names are different
44 :-owns(1,7,K).
45 %decode numbers to names
46 own(I,J,K) :- owns(X,Y,Z), firstname(X,I), lastname(Y,J), pet(Z,K).
47 %show only own sets
48 #show own/3.
49
```

## Output:

Configuration: reasoning mode  ☐ project ☐ statistics

```
clingo version 4.5.3
Solving...
Answer: 1
own(engles,bruce,iguana) own(halevy,abner,jackal) own(gunter,duane,kingcobra) own(foster,chuck,llama)
SATISFIABLE

Models      : 1
Calls      : 1
Time       : 0.032s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time   : 0.000s
```

clingo version 4.5.3

Solving...

Answer: 1

own(engles,bruce,iguana) own(halevy,abner,jackal) own(gunter,duane,kingcobra)

own(foster,chuck,llama)

SATISFIABLE

Models : 1

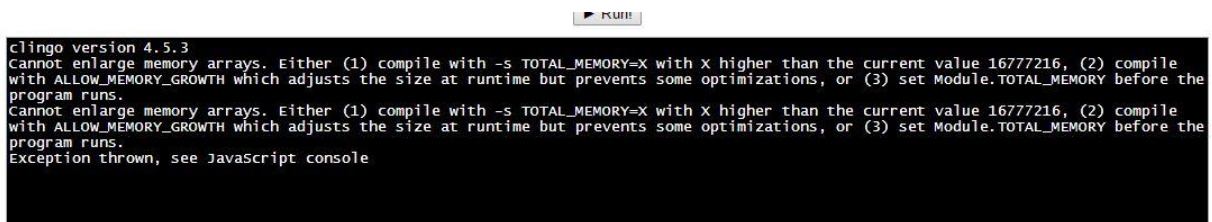
Calls : 1

Time : 0.032s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)

CPU Time : 0.000s

### Bonus credit problem:

I first wrote the program for 4x4 chessboard and extended it to 8x8 board. I have attached the program for both. I got the output for 4x4 whereas 8x8 is still running and I could not attach the output at the moment. I will email it to you as soon as the program completes. Running these online gave the following error. So I switched to desktop version of the tool.



```
clingo version 4.5.3
Cannot enlarge memory arrays. Either (1) compile with -s TOTAL_MEMORY=X with X higher than the current value 16777216, (2) compile
with ALLOW_MEMORY_GROWTH which adjusts the size at runtime but prevents some optimizations, or (3) set Module.TOTAL_MEMORY before the
program runs.
Cannot enlarge memory arrays. Either (1) compile with -s TOTAL_MEMORY=X with X higher than the current value 16777216, (2) compile
with ALLOW_MEMORY_GROWTH which adjusts the size at runtime but prevents some optimizations, or (3) set Module.TOTAL_MEMORY before the
program runs.
Exception thrown, see JavaScript console
```

### Representation:

A set tiles(x,y,z) means that it's a 3x1 tile and spans box x, box y, box z on the chess board. tile(i) means that it is a 1x1 tile and occupies box i on the board.

### 4x4 program:

%generate 3x1 tiles exactly 5 numbers

5{tiles(1..16,1..16,1..16)}5.

%generate 1x1 tile exactly 1 number

1{tile(1..16)}1.

%remove duplicates like (1,2,3), (3,2,1) etc

:-tiles(I,J,K), J<I.

:-tiles(I,J,K), K<I.

:-tiles(I,J,K), K<J.

%1x1 tile should not be among any 3x1 tile cells

:-tiles(I,J,K), tile(I).

:-tiles(I,J,K), tile(J).

:-tiles(I,J,K), tile(K).

%only unique elements are allowed within sets

:-tiles(I,J,K), I=J.

:-tiles(I,J,K), J=K.

:-tiles(I,J,K), I=K.

%unique numbers across sets



```

:-tiles(I,J,K), tiles(_,I,_).
:-tiles(I,J,K), tiles(_,_,I).
:-tiles(I,J,K), tiles(J,_,_).
:-tiles(I,J,K), tiles(_,_,J).
:-tiles(I,J,K), tiles(K,_,_).
:-tiles(I,J,K), tiles(_,K,_).
:-tiles(I,J,K), tiles(I,J1,K1), J!=J1, K!=K1.
:-tiles(I,J,K), tiles(I1,J,K1), I!=I1, K!=K1.
:-tiles(I,J,K), tiles(I1,J1,K), I!=I1, J!=J1.

```

%horizontal right end boundary.

```

:-tiles(I,J,K), I>=2+(F*X), I<=4+(F*X), K=I+2, X=0..3, F=4.
:-tiles(I,J,K), I>=2+(F*X), I<=4+(F*X), J=J+1, X=0..3, F=4.

```

%vertical constraint for tiles and also boundary

```

:-tiles(I,J,K), J!=I+1, J!=I+4.
:-tiles(I,J,K), K!=I+2, K!=I+8.
:-tiles(I,J,K), K!=J+1, K!=J+4.

```

**Command line:**

```

C:\WINDOWS\system32\cmd.exe

D:\SEM1\AI\clingo-4.5.3-win64>clingo tiles.txt 1
clingo version 4.5.3
Reading from tiles.txt
Solving...
Answer: 1
tiles(1,2,3) tiles(4,8,12) tiles(5,9,13) tiles(6,10,14) tiles(7,11,15) tile(16)
SATISFIABLE

Models      : 1+
Calls       : 1
Time        : 9.077s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 9.047s

D:\SEM1\AI\clingo-4.5.3-win64>

```

**Output:**

```

D:\SEM1\AI\clingo-4.5.3-win64>clingo tiles.txt 1
clingo version 4.5.3
Reading from tiles.txt
Solving...
Answer: 1
tiles(1,2,3) tiles(4,8,12) tiles(5,9,13) tiles(6,10,14) tiles(7,11,15) tile(16)
SATISFIABLE

```

```

Models      : 1+
Calls       : 1
Time        : 9.077s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 9.047s

```

D:\SEM1\AI\clingo-4.5.3-win64>

**8x8 program:**

%generate 3x1 tiles exactly 21 numbers



```

21{tiles(1..64,1..64,1..64)}21.
%generate 1x1 tile exactly 1 number
1{tile(1..64)}1.

%remove duplicates like (1,2,3), (3,2,1) etc
:-tiles(I,J,K), J<I.
:-tiles(I,J,K), K<I.
:-tiles(I,J,K), K<J.

%1x1 tile should not be among any 3x1 tile cells
:-tiles(I,J,K), tile(I).
:-tiles(I,J,K), tile(J).
:-tiles(I,J,K), tile(K).

%only unique elements are allowed within sets
:-tiles(I,J,K), I=J.
:-tiles(I,J,K), J=K.
:-tiles(I,J,K), I=K.

%only unique elements are allowed across sets
:-tiles(I,J,K), tiles(_,I,_).
:-tiles(I,J,K), tiles(_,_,I).
:-tiles(I,J,K), tiles(J,_,_).
:-tiles(I,J,K), tiles(_,_,J).
:-tiles(I,J,K), tiles(K,_,_).
:-tiles(I,J,K), tiles(_,K,_).
:-tiles(I,J,K), tiles(I,J1,K1), J!=J1, K!=K1.
:-tiles(I,J,K), tiles(I1,J,K1), I!=I1, K!=K1.
:-tiles(I,J,K), tiles(I1,J1,K), I!=I1, J!=J1.

%horizontal right end boundary.

:-tiles(I,J,K), I>=6+(E*X), I<=8+(E*X), K=I+2, X=0..7, E=8.
:-tiles(I,J,K), I>=6+(E*X), I<=8+(E*X), J=J+1, X=0..7, E=8.

%vertical constraint for tiles and also boundary
:-tiles(I,J,K), J!=I+1, J!=I+4.
:-tiles(I,J,K), K!=I+2, K!=I+8.
:-tiles(I,J,K), K!=J+1, K!=J+4.

```

### Command line:

 C:\WINDOWS\system32\cmd.exe - clingo tilesfinal.txt 1

```

D:\SEM1\AI\clingo-4.5.3-win64>clingo tilesfinal.txt 1
clingo version 4.5.3
Reading from tilesfinal.txt

```

And it's still running as there is a combinatorial explosion due to  $64 \times 64 \times 64$ .