CS 322 A3
Peter Chung 38777124
Mitchell Chan 10753135

Q1

a)

Variables: Monster, Agent, Weapon, WeaponCharge

dom(Monster_3): {T, F}
dom(Monster_9): {T,F}
dom(Agent_1) : {T,F}
dom(Agent_2) : {T,F}
dom(Agent_3) : {T,F}
dom(Agent_4) : {T,F}
dom(Agent_5) : {T,F}
dom(Agent_6) : {T,F}
dom(Agent_7) : {T,F}
dom(Agent_8) : {T,F}
dom(Agent_9) : {T,F}
dom(Agent_10) : {T,F}
dom(WeaponCharged): {T, F}
dom(WeaponCharge1): {T, F}
dom(WeaponCharge4): {T, F}

b)

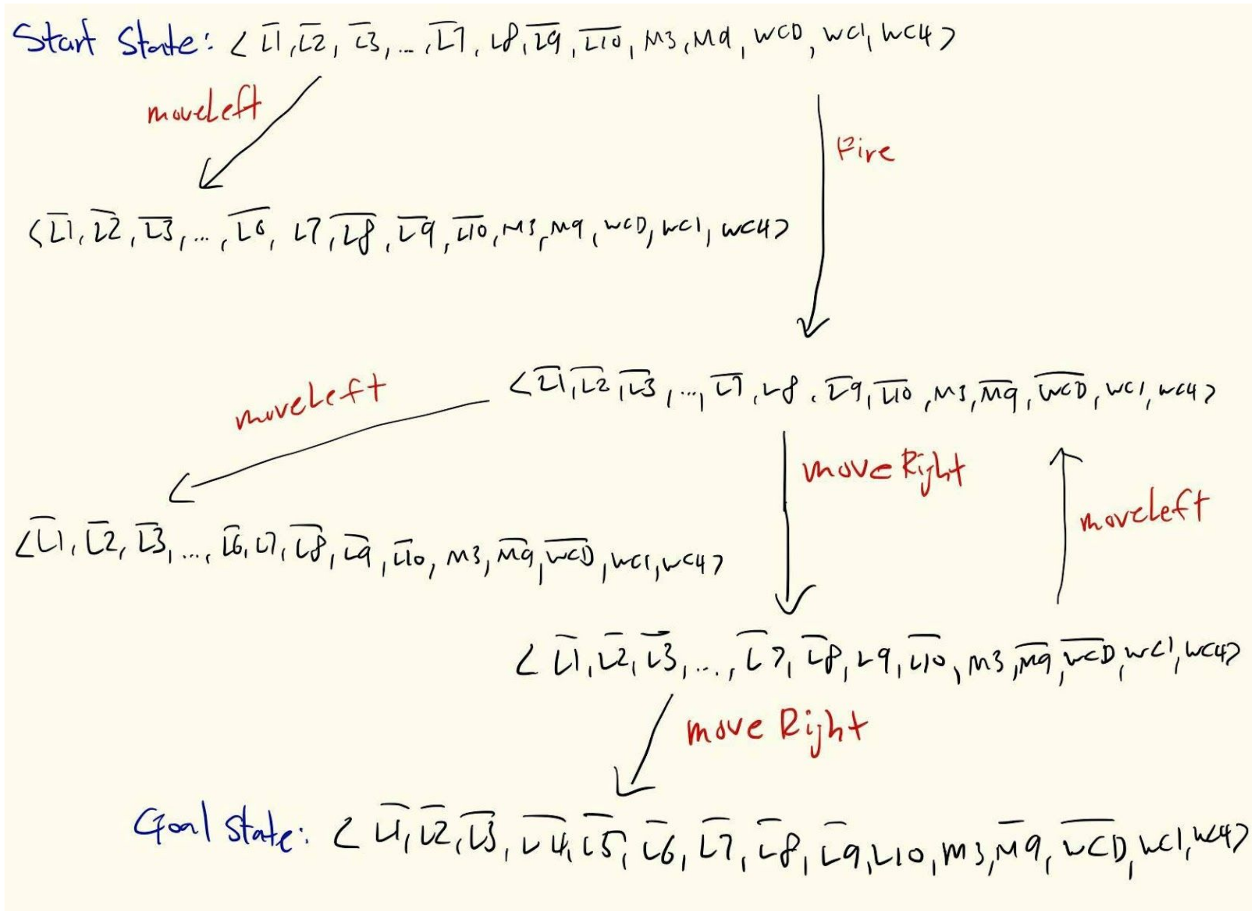
moveRight4
Precondition: Agent_4 = T
Effect: Agent_4 = F, Agent_5 = T

moveLeft4
Precondition: Agent_4 = T, Monster_3 = F
Effect: Agent_4 = F, Agent_3 = T

pickUp4
Precondition: Agent_4 = T, WeaponCharge4 = T
Effect: Agent_4 = T, WeaponCharged = T, WeaponCharge4 = F

Fire4
Precondition: Agent_4 = T, WeaponCharged = T
Effect: Agent_4 = T, WeaponCharged = F, Monster_3 = F

c)



d)
Domain dependent heuristic is running an empty-delete-list on current state. (e.g. if at l2, can either mR or mL or f.) Enforce deleting effects that are F. Heuristic is number of features that are assigned T.

A good admissible heuristic for this planning goal is that you have a charged weapon but you don’t have to worry about being blocked if there is a monster present in either 3 or 9 - you can move wherever you want regardless. It is good and admissible because the goal is for the agent to be in location 10 and have a charged weapon.

e)
When we are in location 9, we can only perform either of moveLeft or moveRight.
Heuristic value of moveLeft : 4 (L8,wcd, wc1, wc4)
m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = T l9 = F l10 = F wcd = T wc1 = T wc4 = T
Heuristic value of moveRight : 4 (l10, wcd, wc1, wc4)
m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = F l9 = F l10 = T wcd = T wc1 = T wc4 = T

f)
When we are in location 8, we can only perform either of moveLeft or fire.
Heuristic value of moveLeft : 6 (m3,m9,L7,wcd, wc1, wc4)
m3 = T m9 = T l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = T l8 = F l9 = F l10 = F wcd = T wc1 = T wc4 = T
Heuristic value of fire : 4 (m3, L8, wc1, wc4)
m3 = T m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = T l9 = F l10 = F wcd = F wc1 = T wc4 = T

g)
As you can see from the results of e), when you are free to moveLeft or moveRight, there is no difference in heuristic values when in fact, you should be awarded for getting closer to the goal state when the agent performs moveRight

h)
When the agent is at location 8, we can perform the following actions : moveLeft, moveRight, Fire, Pickup
Heuristic value of moveLeft : 6 (m3, m9, L7, wcd, wc1, wc4)
m3 = T m9 = T l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = T l8 = F l9 = F l10 = F wcd = T wc1 = T wc4 = T
Heuristic value of moveRight : 7 (Fire (precondition to remove monster at location 9), m3, m9, L9, wcd, wc1, wc4)
m3 = T m9 = T l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = F l9 = T l10 = F wcd = T wc1 = T wc4 = T
Heuristic value of Fire : 4 (m3,L8, wc1, wc4)
m3 = T m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = T l9 = F l10 = F wcd = F wc1 = T wc4 = T
Heuristic value of Pickup: 6 (m3, m9, L8, wc1, wc4, wcd)
m3 = T m9 = T l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = T l9 = F l10 = F wcd = T wc1 = T wc4 = T

i)
When the agent is at location 9, we can perform the following actions : moveLeft, moveRight, Fire, Pickup
Heuristic value of moveLeft : 4 (L8, wcd, wc1, wc4)
m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = T l9 = F l10 = F wcd = T wc1 = T wc4 = T
Heuristic value of moveRight : 4 (L10, wcd, wc1, wc4)
m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = F l9 = F l10 = T wcd = T wc1 = T wc4 = T
Heuristic value of Fire : 4 (wcd (precondition),L9, wc1, wc4)

m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = F l9 = T l10 = F wcd = F wc1 = T wc4 = T
Heuristic value of Pickup: 4 (L9, wc1, wc4, wcd)
m3 = F m9 = F l1 = F l2 = F l3 = F l4 = F l5 = F l6 = F l7 = F l8 = F l9 = T l10 = F wcd = T wc1 = T wc4 = T

j)
Yes by taking preconditions into the consideration of heuristic values, actions requiring other actions as a precondition result in higher heuristic value.

Q2

a)

HaveVodkaBase

{T, F}

HaveBlue

{T, F}

HaveBerry

{T, F}

CleanGlass

{T, F}

PracticedJane

{T, F}

PracticedLaura

{T, F}

b)

MakeVodkaBase

Preconditions: HaveVodkaBase = False

Postconditions: HaveVodkaBase = True

WashGlass

Preconditions: CleanGlass = F

Postconditions: CleanGlass = T

MakeBerry

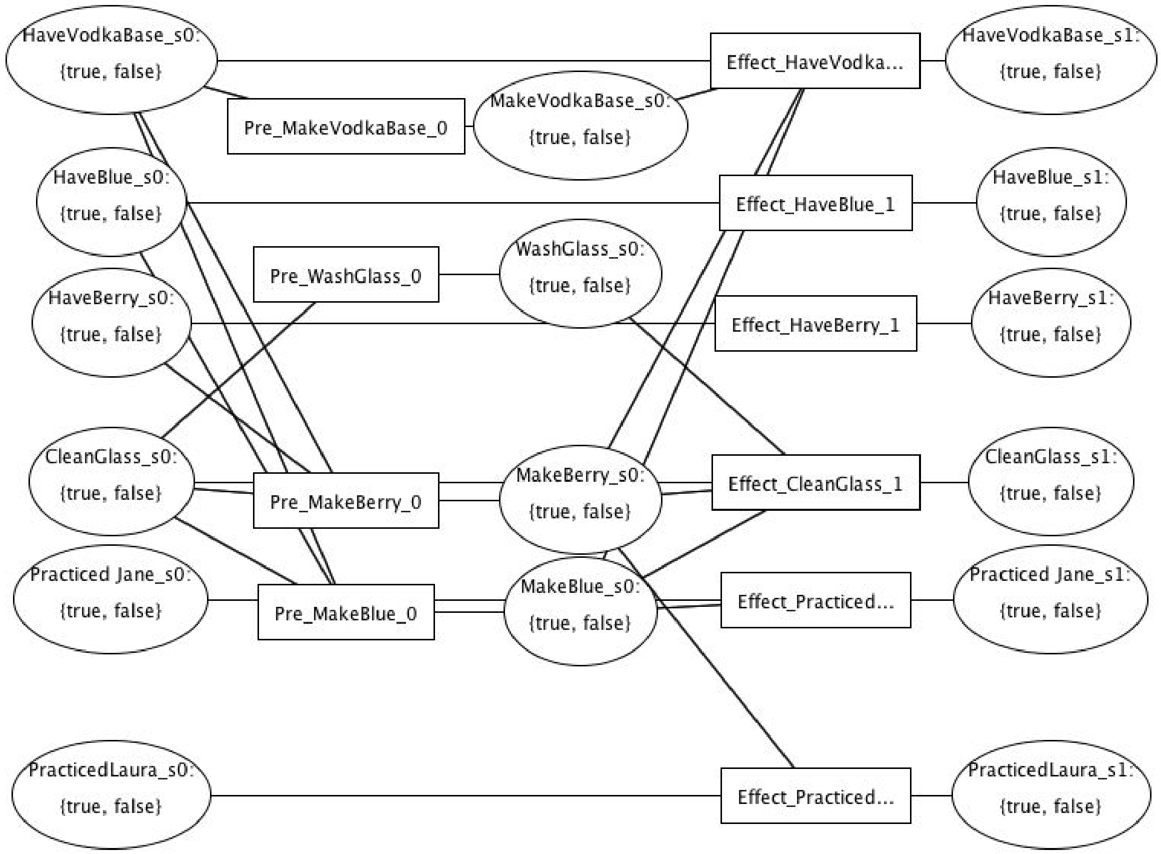
Preconditions: HaveVodkaBase = T; CleanGlass = T; HaveBerry = T

Postconditions: HaveVodkaBase = F; CleanGlass = F; PracticedLaura = T

MakeBlue

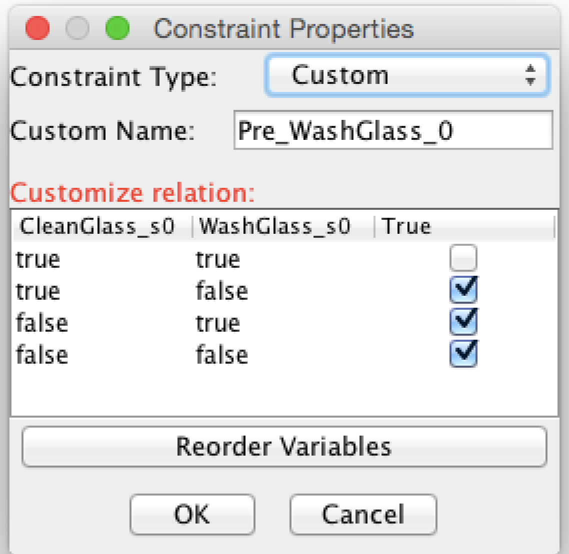
Preconditions: HaveVodkaBase = T; CleanGlass = T; HaveBlue = T

Postconditions: HaveVodkaBase = F; CleanGlass = F; PracticedJane = T c)



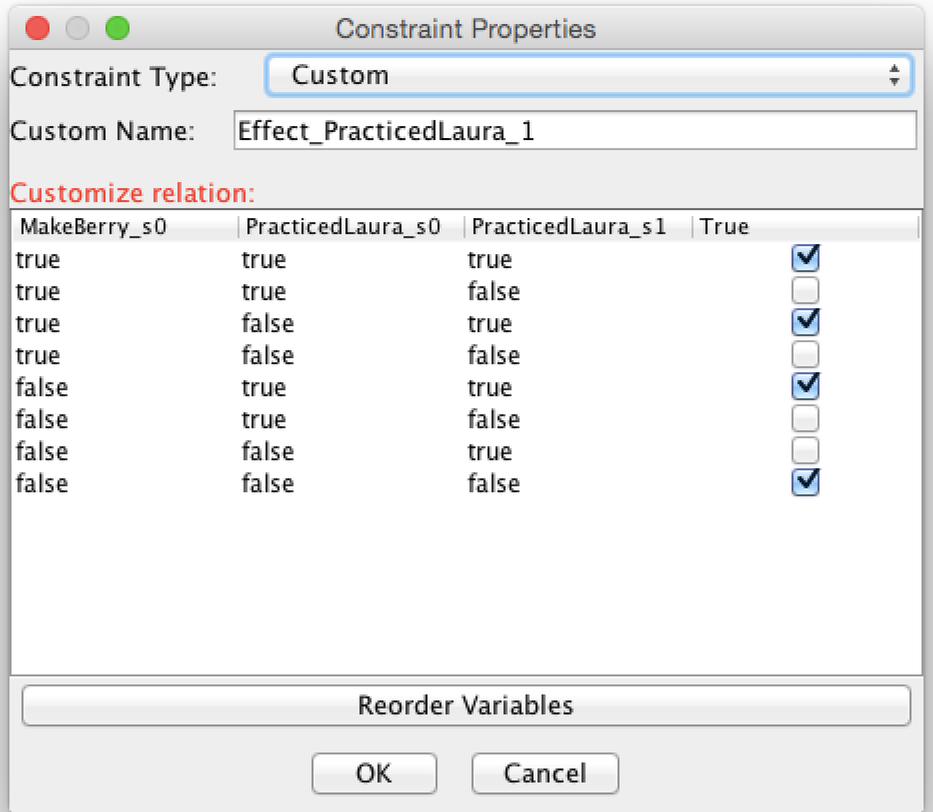
HaveVodkaBase_s0:
{true, false}
Pre_MakeVodkaBase_0
MakeVodkaBase_s0:
{true, false}
Effect_HaveVodka...
HaveVodkaBase_s1:
{true, false}
HaveBlue_s0:
{true, false}
Effect_HaveBlue_1
HaveBlue_s1:
{true, false}
Pre_WashGlass_0
WashGlass_s0:
{true, false}
HaveBerry_s0:
{true, false}
Effect_HaveBerry_1
HaveBerry_s1:
{true, false}
CleanGlass_s0:
{true, false}
Pre_MakeBerry_0
MakeBerry_s0:
{true, false}
Effect_CleanGlass_1
CleanGlass_s1:
{true, false}
Practiced Jane_s0:
{true, false}
Pre_MakeBlue_0
MakeBlue_s0:
{true, false}
Effect_Practiced...
Practiced Jane_s1:
{true, false}
PracticedLaura_s0:
{true, false}
Effect_Practiced...
PracticedLaura_s1:
{true, false}

d)



Last column determines whether or not this is a valid action. Not a valid action when CleanGlass_s0 and WashGlass_s0 are both TRUE. When CleanGlass_s0 is TRUE, WashGlass_s0 must be FALSE and action cannot be committed as the precondition of WashGlass is that CleanGlass must be FALSE. When CleanGlass_s0 is FALSE, action WashGlass_s0 can be done since precondition of action is met. When both CleanGlass_s0 and WashGlass_s0 are FALSE, means no action is taken - option of not taking an action despite it being valid.

e)



Constraint Properties
Constraint Type:
Custom
Custom Name:
Effect_PracticedLaura_1
Customize relation:
MakeBerry_s0 PracticedLaura_s0 PracticedLaura_s1 True
true true true
true true false
true false true
true false false
false true true
false true false
false false true
false false false
Reorder Variables
OK
Cancel

PracticedLaura_s0	MakeBerry_s0	PracticedLaura_s1	Explanation
T	T	T	Valid action because only preconditions of MakeBerry is ⇒ *HaveVodkaBase = T; CleanGlass = T; HaveBerry = T* and as long as these are met, nothing wrong. Therefore, doesnt matter if PracticedLaura is already T
F	T	T	Valid action because only preconditions of MakeBerry is ⇒ *HaveVodkaBase = T; CleanGlass = T; HaveBerry = T* and as long as these are met, nothing wrong. Effect of MakeBerry is that PracticedLaura becomes T
T	F	T	Valid action because only preconditions of MakeBerry is ⇒ *HaveVodkaBase = T; CleanGlass = T; HaveBerry = T* and as long as these are met, nothing wrong. Since PracticedLaura already T and MakeBerry action not taken, PracticedLaura still T.
F	F	F	Option of not taking an action despite it being valid

f)

Plan

Horizon

1	WashGlass
2	MakeVodkaBase
3	MakeBerry
4	WashGlass
5	MakeVodkaBase
6	MakeBlue

Need to unroll the CSP to a minimum horizon of 6 for the goal *PracticedLaura* = T and *PracticedJane* = T

g)

According to CSP, there is no solution as there are multiple empty domains

h)

We would need to include duplicate domain values so that we can account for the other drink being made after the first is completed.

Q3

a) Atoms without bodies : k,s where k = T and s = T
   Given k,s, we can further derive u from u ← s,q from q ← s and z from z ← s where u = T, q = T and z = T
   Given k,s,u,q,z we can further derive j from j ← q ^ z and w from w ← z where j = T and w = T
   Given k,s,u,q,z,j,w we can no longer derive.
   Hence k,s,u,q,z,j,w can be proved by KB
b) False, KB is missing x and p that makes all atoms entailed.
c)
   i) Yes. If you have additional model x (by itself without a body), this is not a logical consequence of this KB but still can be true.
   ii) a ← b where a = F and b is also F. If b was assigned as false and a was also assigned as false, then all the clauses are false hence cannot be a model.
d) 1
   i) No, m ← w ^ q ^ p but p is not provable
   ii) Yes,
      Yes ← j ^ w
      Yes ← q ^ z ^ w by j ← q ^ z
      Yes ← q ^ z ^ z by w ← z
      Yes ← q ^ s ^ z by z ← s
      Yes ← q ^ s ^ s by z ← s
      Yes ← q ^ s by s
      Yes ← q by s
      Yes ← s by q ← s
      Yes ← by s
      Hence yes.

Q4

a)

```
system(museum).

hasSensor(museum,door).
hasSensor(museum,laser).
hasSensor(museum,window).

connected_to(door,power) <- live(door) & live(power).

triggered(laser) <- laser_interrupted(laser).
triggered(door) <- door_open(door).
triggered(window) <- window_broken(window).

live(door) <- live(power).
live(laser) <- live(power) & circuit_ok(c1).
live(window) <- live(power) & circuit_ok(c1) & circuit_ok(c2).

alarm_triggered(laser) <- live(laser) & triggered(laser).
alarm_triggered(door) <- live(door) & triggered(door).
alarm_triggered(window) <- live(window) & triggered(window).
```

b)