David Pérez Carrasco | 25984360
Lucas Amar | 37082823
11/08/2022

CPSC 322 Assignment 3

Question 1 [30 points]: Heuristics in STRIPS: a video game example

a) Variables and domains:

The variables with their corresponding domains are the following:

Loc_i, $where\ i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $with\ dom(Loc_i) = \{T, F\}$
$Weapon\_charged$, $with\ dom(Weapon\_charged) = \{T, F\}$
$FreeMonster_3$, $with\ dom(FreeMonster_3) = \{T, F\}$,
$where\ T\ corresponds\ to\ the\ absence\ of\ the\ monster\ in\ Loc_3$
$FreeMonster_9$, $with\ dom(FreeMonster_9) = \{T, F\}$,
$where\ T\ corresponds\ to\ the\ absence\ of\ the\ monster\ in\ Loc_9$
$Weapon\_charge_1$, $with\ dom(Weapon\_charge_1) = \{T, F\}$,
$where\ T\ corresponds\ to\ the\ presence\ of\ a\ weapon\ charge\ in\ Loc_1$
$Weapon\_charge_4$, $with\ dom(Weapon\_charge_4) = \{T, F\}$,
$where\ T\ corresponds\ to\ the\ presence\ of\ a\ weapon\ charge\ in\ Loc_4$

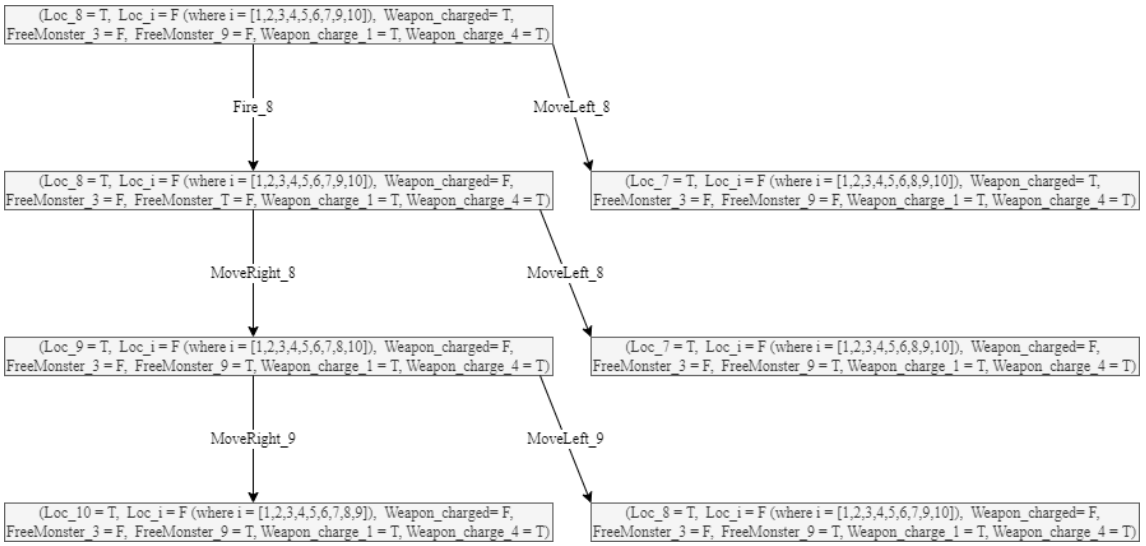
Note: We could also have defined the Monster variables with {monster, no monster} domain and the weapon charge with {available, unavailable} domain and weapon with {charged, uncharged} domain, but we have considered it easier and simpler to declare all of them boolean variables.

b) Actions in Location 4

Action	Preconditions	Effects
$moveRight_4$	$Loc_4 = T$	$Loc_4 = F$ $Loc_5 = T$
$moveLeft_4$	$Loc_4 = T$ $FreeMonster_3 = T$	$Loc_4 = F$ $Loc_3 = T$
$pickup_4$	$Loc_4 = T$ $Weapon\_charge_4 = T$	$Weapon\_charged = T$ $Weapon\_charge_4 = F$
$fire_4$	$Loc_4 = T$ $Weapon\_charged = T$	$FreeMonster_3 = T$ $Weapon\_charged = F$

c) Search Space

The optimal solution path will be $Fire_8$, $MoveRight_8$, $MoveRight_9$, and the part of the search space corresponding to the solution is shown in the next figure, where only the possible actions from each step of the optimal solution path are included.



d) Admissible heuristic

The easiest admissible heuristic could be the number of steps between the actual state and the goal state (i.e. the distance between the agent and the goal), but we could improve this including the number of monsters in the remaining steps, and that could be even improved including the number of required steps when the agent can charge his weapon (i.e. the number of steps in which there is a weapon charge available and the agent weapon is uncharged).

So the final heuristic would be: h(n) = #steps from the agent location to the goal location + #not-free monsters locations between the agent location and the goal location + #needed weapon charge steps between the agent location and the goal location.

This heuristic would be admissible since the distance between the agent and the goal is necessarily underestimating the needed number of steps, but also that amount plus the number of monsters must be equal or less than the final number of steps of the optimal solution, since the agent must take one extra step per each monster in his path in order to delete it. Finally, the number of weapon charges locations where the agent weapon would be uncharged is not overestimating neither the optimal solution, since the agent must necessarily take one extra step to charge his weapon at every location in which he has the weapon uncharged, there is a weapon charge and he would need that in order to reach the goal. Hence, this heuristic is not overestimating the cost of the solution and is the closest possible to the real cost, so it is an admissible heuristic.

e) State: ($Loc_1 = F$, $Loc_2 = F$, $Loc_3 = F$, $Loc_4 = F$, $Loc_5 = F$, $Loc_6 = F$, $Loc_7 = F$, $Loc_8 = F$, $Loc_9 = T$, $Loc_{10} = F$, $FreeMonster_3 = T$, $FreeMonster_9 = T$, $Weapon\_charged = T$, $Weapon\_charge_1 = T$, $Weapon\_charge_4 = T$)

The heuristic is 1 everywhere, except in the goal state, which is always 0, because if the precondition of MoveRight9 is empty, then we can reach the goal in just 1 step, it does not matter in which state we are.

f) State: ($Loc_1 = F$, $Loc_2 = F$, $Loc_3 = F$, $Loc_4 = F$, $Loc_5 = F$, $Loc_6 = F$, $Loc_7 = F$, $Loc_8 = T$, $Loc_9 = F$, $Loc_{10} = F$, $FreeMonster_3 = F$, $FreeMonster_9 = F$, $Weapon\_charged = T$, $Weapon\_charge_1 = T$, $Weapon\_charge_4 = T$)

The real cost of the solution path would be 3, since this is the same situation described in part c), but since we are considering the domain-independent heuristic, the heuristic would be 1 everywhere again, since the preconditions are deleted and then the MoveRight9 would be possible even though we are not even in Loc9, but since this heuristic frees all the preconditions, that action would be possible and then the heuristic will be just one step everywhere except in goal states.

g) This heuristic is not useful at all, since it will give us the same heuristic for all the states, so it would not do the heuristic function, which is helping to differentiate between the good and bad states, since with this heuristic all of them would get the same values. Hence, this is a too much relaxed heuristic, because it is underestimating too much the solution.

h) State: ($Loc_1 = F$, $Loc_2 = F$, $Loc_3 = F$, $Loc_4 = F$, $Loc_5 = F$, $Loc_6 = F$, $Loc_7 = F$, $Loc_8 = T$, $Loc_9 = F$, $Loc_{10} = F$, $FreeMonster_3 = F$, $FreeMonster_9 = F$, $Weapon\_charged = T$, $Weapon\_charge_1 = T$, $Weapon\_charge_4 = T$)

Optimal plan: Fire_8, MoveRight_8, MoveRight_9
Heuristic: h(n) = 3

i) State: ($Loc_1 = F$, $Loc_2 = F$, $Loc_3 = F$, $Loc_4 = F$, $Loc_5 = F$, $Loc_6 = F$, $Loc_7 = F$, $Loc_8 = F$, $Loc_9 = T$, $Loc_{10} = F$, $FreeMonster_3 = T$, $FreeMonster_9 = T$, $Weapon\_charged = T$, $Weapon\_charge_1 = T$, $Weapon\_charge_4 = T$)

Optimal plan: MoveRight_9
Heuristic: h(n) = 1

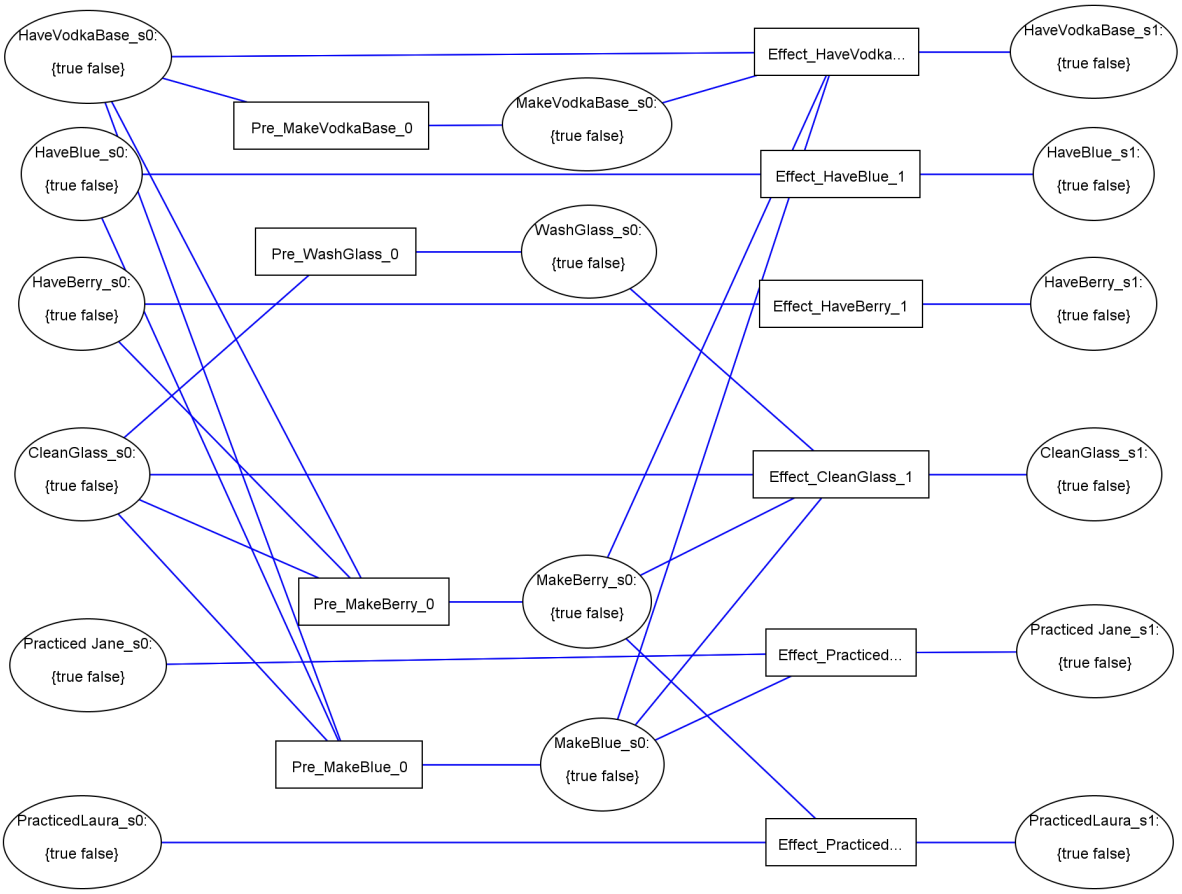
j) This is a more useful heuristic since it gives us more information about the actual closeness of each state to the optimal solution, differentiating between bad states and good states, so it would lead us to the closer states to the solution.

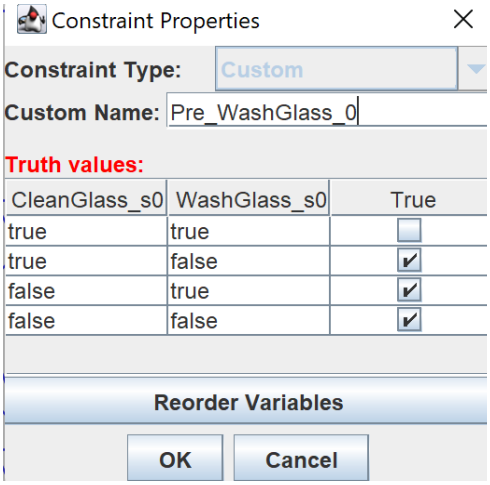
Question 2 [24 points]: STRIPS-to-CSP

(a) Variables and domains
(i) HaveVodkaBase → Domain = {True, False}
(ii) CleanGlass → Domain = {True, False}
(iii) HaveBerry → Domain = {True, False}
(iv) HaveBlue → Domain = {True, False}
(v) Practiced Jane → Domain = {True, False}
(vi) PracticedLaura → Domain = {True, False}

(b) Actions
(i) MakeVodkaBase
    (1) Preconditions → HaveVodkaBase = False
    (2) Effects → HaveVodkaBase = True
(ii) WashGlass
    (1) Preconditions → CleanGlass = False
    (2) Effects → CleanGlass = True
(iii) MakeBerry
    (1) Preconditions →
        a) HaveVodkaBase = True
        b) CleanGlass = True
        c) HaveBerry = True
    (2) Effects →
        a) HaveVodkaBase = False
        b) CleanGlass = False
        c) PracticedLaura = True
(iv) MakeBlue
    (1) Preconditions →
        a) HaveVodkaBase = True
        b) CleanGlass = True
        c) HaveBlue = True
    (2) Effects →
        a) HaveVodkaBase = False
        b) CleanGlass = False
        c) PracticedJane = True

(c) STRIPS to CSP:





(d)

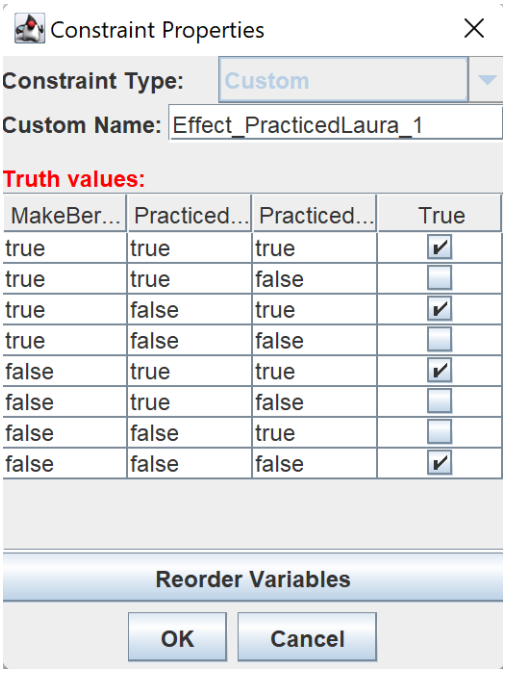
(i) The first row has values true for CleanGlass_s0 and WashGlass_s0, and the constraint is false, meaning that it is not satisfied. In order to WashGlass we need CleanGlass to be false, however in this case both are true making it impossible.

(ii) The second row has values true for CleanGlass_s0 and false for WashGlass_s0. In this case the constraint is true, given that it does not matter what the value of CleanGlass is.

(iii) The third row shows the preconditions to perform WashGlass, as CleanGlass_s0 is false and WashGlass_s0 is true. Therefore the constraint has to be true.

(iv) For the fourth row, just like the second row, if WashGlass is false the value of CleanGlass does not matter, and the constraint will be true.

(e)



(i) The first row is true given that Laura makes her cocktail and she had already practiced. Therefore the constraint is true and the value for PracticedLaura will not change in the next step.

(ii) The third row is also true given that MakeBerry is true and Laura hasn't practiced. Therefore the constraint is satisfied as Laura can practice and in the next step PracticedLaura will become true.

(iii) The fifth row is also true as MakeBerry is false and PracticedLaura is true. Therefore we are not taking MakeBerry action so PracticedLaura will stay false in the next step.

(iv) The eighth row is also true given that when everything is false nothing will happen and thus the constraint is satisfied.

(f) Min_Horizon to find a solution is 2. The solution found yields steps:

(i) MakeVodkaBase_s0 → WashGlass_s0 → MakeBerry_s1 → MakeBlue_s1

(g) The plan from part f is inconsistent given that 2 actions are being taken simultaneously. It is making both cocktails at the same time and therefore a solution is found at a lesser horizon.

(h) As there are no constraints for mutually exclusive actions the CSP approach is not the right one. This can be solved with a mutex constraint, making it impossible for the CSP solver to carry out 2 actions simultaneously.

Question 3 [20 points]: Propositional Definite Clauses - Proof Procedures

a) All atoms by KB
   i) List: {k,s} → lines 2 and 11
   ii) Updated List: {k,s,z} → line 7
   iii) Updated List: {k,s,z,q} → line 10
   iv) Updated List: {k,s,z,q,u} → line 13
   v) Updated List: {k,s,z,q,u,j} → line 1
   vi) Updated List: {k,s,z,q,u,j,w} → line 9
   vii) Final list: {k,s,z,q,u,j,w}

b) Yes, it is true for all the atoms since bottom up is sound.

c)

   (i) If we pick atom x on top of the previous v, the KB now becomes {k,s,z,q,u,j,w,x,p,m}. Therefore, since the entire KB is included in this KB, it is a model.

   (ii) Picking atoms p or m would not have given a model, given that we would not have been able to get x. Therefore we would not have all atoms.

d)

   (i) Yes ← m^j
      (1) Yes ← m^j^w^q^p → line 3
      (2) Yes ← m^j^w^q^p^z → line 1
      (3) Yes ← m^j^w^q^p^z^u^x → line 5
      (4) Yes ← m^j^w^q^p^z^u^x^s → line 6
      (5) Yes ← m^j^w^q^p^z^u^x^s^k → line 5
      (6) With the top down method, m^j is a logical consequence of KB

   (ii) Yes ← j^w
      (1) Yes ← j^w^q^z → line 1
      (2) Yes ← j^w^q^z^u^x → line 5
      (3) Yes ← j^w^q^z^u^x^s → line 7
      (4) Yes ← j^w^q^z^u^x^s^k → line 9
      (5) Yes ← j^w^q^z^u^x^s^k^p → line 8
      (6) Yes ← j^w^q^z^u^x^s^k^p^m → line 3
      (7) With the top down method, j^w is a logical consequence of KB

4 [15 points]: Predicate Logic

File content:

```
% file: A3.pl

%Assigment 3, Q4

%the door is connected with the power

connected_to(door,power).

%The laser is connected to the door is c1 is ok

connected_to(laser,door) <- circuit_ok(c1).

%The window is connected to the laser is c2 is ok

connected_to(window,laser) <- circuit_ok(c2).

%X has power if is connected to Y and Y has power

live(X) <- connected_to(X,Y) & live(Y).

%Window sensor is triggered if window is broken and window has power

triggered(window) <- window_broken(window) & live(window).

%Door sensor is triggered if door is open and has power

triggered(door) <- door_open(door) & live(door).

%Laser sensor is triggered if laser is interrupted and has power

triggered(laser) <- laser_interrupted(laser) & live(laser).

%System M has a triggered alarm if M is a system, has a sensor X and X sensor is triggered

alarm_triggered(M) <- system(M) & hasSensor(M,X) & triggered(X).

%We have a system s with sensors laser, window, door

system(s).

hasSensor(s,laser).

hasSensor(s,window).

hasSensor(s,door).


%Included facts

live(power).

circuit_ok(c1).

circuit_ok(c2).

window_broken(window).
```

Resulting proof deduction tree for the query alarm_triggered(X):