CPSC 322 2022W1 Assignment 3

Make sure you follow all assignment instructions, including those in Canvas (in the assignment description or the syllabus). Failure to do so may result in heavy penalties.

Make sure that in your answers you clearly indicate the exact section you are answering.

Please start each question on a new page (eg. don't put your answer to Q3 on the same page as your answer to Q2).

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

Assume we want to write an automated planner for a very simple video game: we have a world of 10 different locations, arranged horizontally as shown below. There is a game agent that wants to end up in location 10.

1	2	3	4	5	6	7	8	9	10

Positions 3 and 9 may each contain a monster, and the agent cannot move to these locations when there is a monster there. The agent has a weapon that is either charged or uncharged (assume, unless stated otherwise, that it starts out charged), and at the start of the game there are new weapon charges at locations 1 and 4. The following actions are available to the agent:

1) the action "moveRight" increases our agent's location by 1 (up to 10),
2) the action "moveLeft" decreases the location by 1 (down to 1).
3) the action "pickUp" can be taken if there is a charge at the agent's location. If taken, this action will also charge the agent's weapon (whether or not it is already charged) and remove the charge from its location.
4) The action "fire" removes every monster whose location differs from the agent's current location by 1 and completely discharges the agent's weapon.

First, we will write this planning problem in STRIPS notation. STRIPS does not allow us to parameterize actions by the current state because the effects of actions need to be explicit assignments of values to variables; they cannot depend on the agent's state. Thus we cannot write an action "move-right" parameterized by location; instead, we have to make actions "moveRight$_i$", "moveLeft$_i$", "pickUp$_i$", and "fire$_i$" for i = 1, ..., 10, with each of these actions only being applicable at location *i* (similar to what was done for the actions *move-clockwise* and *move-counter-clockwise* in the delivery robot examples discussed in class).

(a) [4 points] Based on the problem description above, define the features (variables) and their domains for this problem.

(b) [4 points] Give the STRIPS representations of the actions the agent may take while in location 4.

(c) [5 points] Suppose that the agent's goal is to be in location number 10; and suppose its start state is at location 8, with monsters occupying locations 3 and 9, the weapon charged, and charges in locations 1 and 4. Draw a **part** of the search space that includes the optimal plan (i.e. the optimal solution path from start state to a goal state), similarly to the coffee delivery example in the lectures. Include all immediate successor states for each state along the optimal path.

For the remaining questions (i.e. part (d) onwards), let the goal be that the agent should be in location 10 and have a charged weapon.

(d) [3 points] Using the properties of this video game, come up with an admissible heuristic for this planning goal.

- As in previous assignments, h(n)=0 everywhere gets no credit.
- Note that this is a domain-dependent heuristic since you take the video game domain into account.
- To get full marks, you must also provide an explanation of why your proposed heuristic is good (i.e. informative) and is *admissible.*

For parts (e)-(g), consider a domain-independent heuristic: "ignore preconditions". For a state *n* this is defined as the cost of the optimal path from n to a goal using a relaxed problem whose actions have no preconditions.

(e) [2 points] What is the value of this heuristic for the state that has the agent in location 9, cells 3 and 9 free of monsters, the weapon charged and charges available in location 1 and 4? Represent this state using your features from (a), and justify your answer.

(f) [2 points] What is the value of this heuristic for the state that has the agent in location 8, monsters in both cell 3 and 9, the weapon charged and charges available in location 1 and 4? Represent this state using your features from (a), and justify your answer.

(g) [2 points] Is this heuristic useful? Why or why not?

For parts (h)-(j), consider the domain-independent heuristic covered in the lectures, namely "ignore delete list" (a.k.a. "empty delete list"). In order to apply this heuristic, all action preconditions need to be expressed as assignments of binary features to T (positive assertions). If the features you selected in **(a)** above do not allow you to satisfy this constraint for the actions defined in **(b),** please define new features accordingly (e.g. you may need to change a feature that expresses the fact that there is a monster in Cell 3 into a feature stating that Cell 3 is free of monsters). **You do not need to change your answer to (a);** simply give the new features here.
If you need to make any such changes, please also give here the new definition of the actions in **(b)** based on these new features.

(h) [3 points] What is the value of the *ignore-delete-list* heuristic for the state in (f)? Also give the optimal plan for the relaxed problem (starting at the state in (f)).

(i) [3 points] What is the value of the *ignore-delete-list* heuristic for the state in **(e)**? Also give the optimal plan for the relaxed problem (starting at the state in **(e)**).

(j) [2 points] Is this heuristic more useful than ignoring preconditions? Why or why not?

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

The DrinkUp Bartending school just opened, so it only has two students (Jane and Laura), one small cocktail shaker that can only make one cocktail at a time, and one cocktail glass. Both Jane and Laura need to practice making cocktails that require having a base of shaken vodka. Jane will use this to make a blue cocktail with Blue Curacao (Blue for short), Laura to make a pink one with Berry Liqueur (Berry for short). The cocktail glass needs to be clean when making a new cocktail. They will only make more shaken vodka base when there is no more left.

Figure 1 shows a STRIPS representation for this problem.

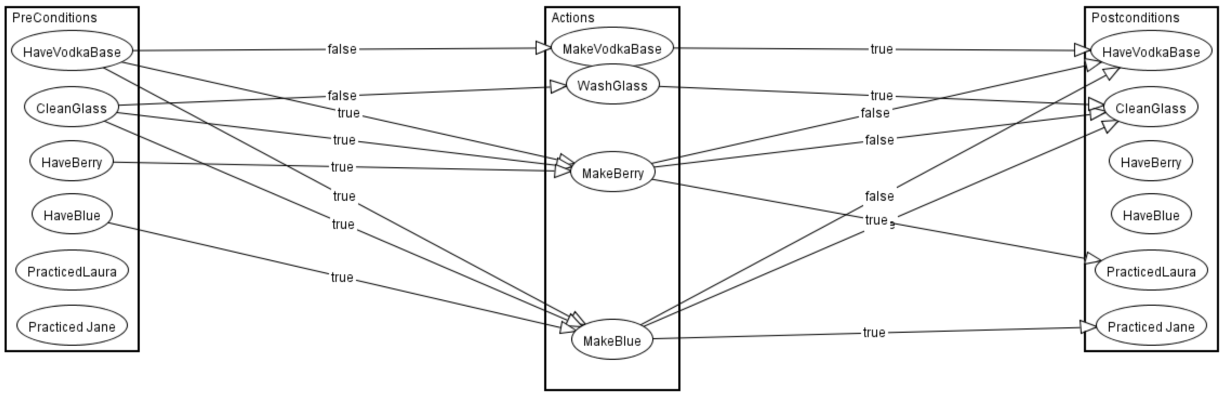


Figure 1 - STRIPS representation for the cocktail making problem

Load the applet at http://aispace.org/strips_to_csp/, and load the provided file cocktail.xml containing this STRIPS problem.

(a) [3 points] List the variables/features that define the states in the problem, as well as their domains.

(b) [4 points] List the actions in this problem, with their preconditions and effects.

(c) [1 point] Convert this STRIPS problem to a CSP planning problem with a horizon of 1. We suggest that you first try to set up the constraints network by yourself, but you can check your solution in the applet by clicking "Solve" and then "Solve with Arc Consistency".
Show a screenshot of the constraint network thus obtained. (**Note:** if the network appears cramped, you can enlarge the window and then select View->Autoscale to spread it out; you may still need to move around some of the nodes to make it easier to read.)

(d) [4 points] Show a screenshot of the truth table representing the constraint Pre_WashGlass_0 and explain what the different rows mean.

(e) [4 points] Show a screenshot of the truth table representing the constraint Effect_PracticedLaura_1 and explain what the different rows mean (for this part, you only need to report and discuss the rows checked as "true" in the applet)

(f) [4 points] Suppose neither Jane nor Laura have practiced making their cocktails. Also suppose that we are in a state with no vodka base, a dirty glass, but we have both Blue Curacao and Berry Liqueur. At what minimum horizon do we need to unroll the CSP if the goal is *PracticedLaura* = T and *PracticedJane* = T? Give a plan to reach the goal, listing the action(s) taken at each step.

Remember that to solve this problem as a CSP you need to define the features you want to hold in your start and goal states; in the AIspace STRIPS to CSP applet, you can do that via "create → set start state" and "create → set goal state". (You also set the horizon here.) After setting the start and goal state, when you click "solve → solve with arc consistency", the resulting CSP will have the desired constraints for start and goal states.

(g) [2 points] Explain why the plan you found in part (f) is inconsistent with the problem description given at the beginning of this question. *(**Note:** you may find a bug in one of the constraint tables in this problem. That is not relevant to this question, and you may ignore it.)*

(h) [2 points] Why does this issue happen with the CSP approach of solving this planning problem, and what can we do to the CSP to fix it?

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

Consider the following knowledge base KB.

1. $j \leftarrow q \land z$
2. k
3. $m \leftarrow w \land q \land p$
4. $w \leftarrow z$
5. $w \leftarrow u \land x$
6. $p \leftarrow x \land s$
7. $z \leftarrow s$
8. $z \leftarrow p$
9. $w \leftarrow k \land j$
10. $q \leftarrow s$
11. s
12. $q \leftarrow j \land w$
13. $u \leftarrow s$

(a) **[5 points]** Using the bottom-up proof procedure, list all atoms *v* such that KB $\vdash v$. Show your work, using the numbers provided above to indicate which clauses are involved in each step of your work.

(b) **[2 points]** Is it true that for all these atoms *v* (from part (a)), KB $\models v$? Why or why not?

(c) [4 points]

i. **[2 points]** Is it possible to pick an atom that is not a logical consequence of this KB, and give a model of that KB in which this atom is true? If yes, give the model. If no, explain why.

ii. **[2 points]** Provide an interpretation of the KB that is not a model (justify your answer).

(d) [9 points] Using the top-down resolution proof method, answer if the following queries are logical consequences of KB. If the answer is "yes" show the successful derivation; and if the answer is "no", show **one** of the failing derivations.

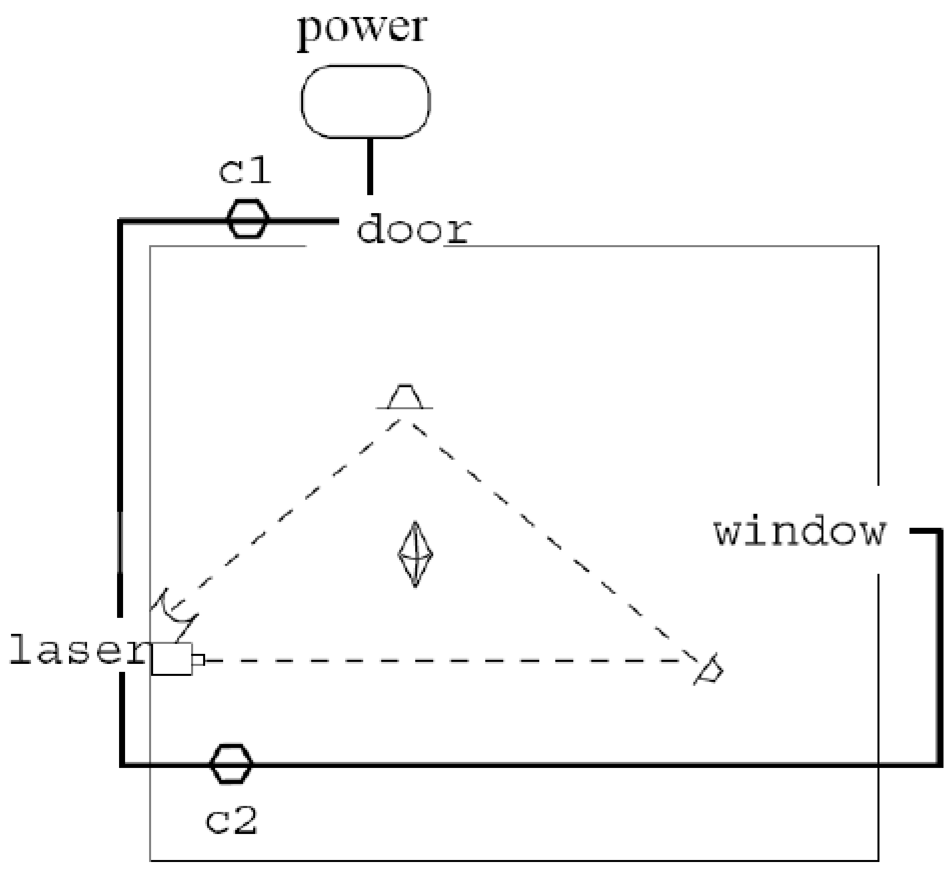
i. [5 points] ? $m \land j$.

ii. [5 points] ? $j \land w$.

4 [15 points]: Predicate Logic

A priceless diamond, the Orange Ocelot, will be displayed at the **Totally-Secure Museum of Really Expensive Stuff** *(trademark pending)*. You have been hired to write the software for the security system that will be used to monitor the room holding the diamond for display. If some dastardly jewel thief attempts to steal the diamond, the system alarm must trigger. The system, ***museum***, shown in the figure below, has three sensors:

- a motion sensor, ***laser***
- a window sensor, ***window***
- a door lock sensor, ***door***



The three sensors are wired in one circuit around the room as shown. The door sensor is connected directly to the power. The laser and the window are each preceded by circuit breakers, c1 and c2, which stop the power at that point if the fuse has blown. If a circuit breaker is “ok" the power flows through it to the next sensor.

You will use the Definite Clause Deduction applet available at http://www.aispace.org to implement the system. This will involve writing a short set of axioms to define the problem and running some queries. First, take a look at the sample knowledge base called “An Electrical System" and understand how it works. You can use the “Move Subtree" button to get a better view of the answer to queries. Your program will use some of the same predicates used here to represent the flow of electricity through a set of switches and devices. **You should use only (and all of) the following predicates for your system:**

alarm_triggered(X)	system X has a triggered alarm
circuit_ok(X)	circuit breaker X is ok
connected_to(X,Y)	X is connected to Y
door_open(X)	there is an open door for X to detect
hasSensor(X,Y)	system X has sensor Y
laser_interrupted(X)	there is an interrupted laser for X to detect
live(X)	X is live (i.e. has electrical power flowing through it)
system(X)	X is a system
triggered(X)	sensor X is triggered
window_broken(X)	there is a broken window for X to detect

Test out your system by stating different facts such as door_open(door) and performing queries such as live(laser) or triggered(door). Remember that uppercase letters denote variables and lowercase denote particular values or instances. Make sure disabling a circuit breaker causes any dependent sensors to fail to trigger an alarm.

Implement the system using the Definite Clause applet in AIspace, and paste the contents of the .pl file you created into your submission. In addition to any facts that are needed in order to implement the system itself, include only the following facts:

- the power is on
- all circuits are ok
- the window is broken

Also, provide a screen capture of the resulting proof deduction tree for the query alarm_triggered(X) for this scenario. You can get this by clicking on “View Proof Deduction" and then selecting the *true* node from the tree.