CSE 410 - AI: Homework 4

This homework requires you submit multiple files. Please make a non compressed tar archive that contains the files and submit it. To run the planners we will use `pyperplan`, which is included in the handout. However, if you run test you are welcome to download and install the package in your python installation. Things that don't get parsed, don't get scored. Please check your files before submitting. You will get unhelpful error messages from the grader.

1) **(5) Basically free Points!** Download the files and run the example pddl file for state description of the 3x3 slider problem and submit the solution file, i.e. call `>>python3 pyperplan slide-domain.pddl task3x3.pddl` and sumbit `task3x3.pddl.soln`. (You'll need to figure out paths for the files)

2) **(15) Making Grid World** Looking at the 3×3 task file `task3x3.pddl` you should notice that the initial conditions specify a lot of adjacency (`adj`) predicates. The `slide-domain-grid.pddl` uses the increment and decrement trick from class to re-write the problem description and to make specifying the adjacency domains easier. The idea is that you can move in a direction if you increment or decrement in that dimension. Instead of locations you will have a pair of indexes representing the rows and columns. Translate the problem from `task3x3.pddl` to this new problem and submit the completed new domain, task (`tasks3x3-grid.pddl`), and solution files.

3) **(10) Written, compare hand-tuned and automatically generated heuristics.** The package `pyperplan` implements a number of the heuristics we had mentioned in class. Use the solver options in `pyperplan` to compute the effective branching factor on the slider problem. Specifically compute the branching factor for running Astar with the:

- Fast forward heuristic (`-H hff`), which gets rid of all the mutex relations).
- Level cost addition heuristic (`-H hdd`), which adds all the individual level costs for the goal states.
- Max level cost (`-H hmax), which uses the max level cost.`

Compute the effective branching factor for these heuristics. Which one is fastest? Make a plot like in homework 2 to check whether or not the fast-forward heuristic is admissible. Is it?

4) **(20) Written - Generate a new PDDL problem from scratch (paper version)** A robot needs to perform tasks in the environment shown on the next page. Specifically, it should bring the sleeping person coffee by dropping it in the room and and end up back its own room with all doors closed. Since the gripper needs to be empty (i.e. not holding coffee) in order to open and close doors. The robot can only perform actions when its charge is greater than 0.
You do not need use Lisp like syntax. You should submit a written description of your modeling choices. Specifically, you need to answer:

- What should be the objects?
- What should be the predicates?

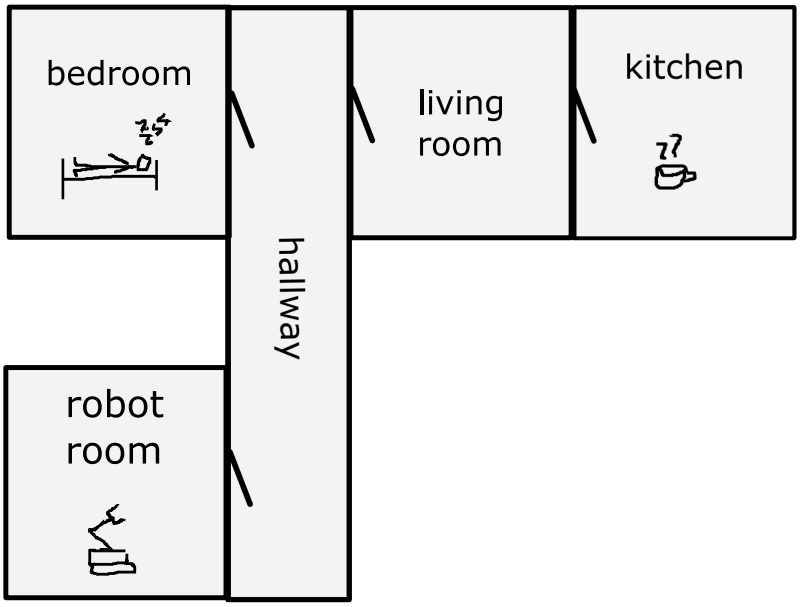

Figure 1: Layout of the environment. Assume that all doors are closed initially. The robot can open each door (if its hand is empty). In the kitchen it can pick up a coffee, which would fill up the hand.

- What should be the actions, and their necessary predicates to express the the precondition and effect?
- What is the initial state, and goal state?

5) **(50) Computer version:** Now write the pddl program, solve the problem and submit the domain file (`coffee-robot-domain.pddl`), task file (`coffee-deliver.pddl`), and solution file. In order to test your solution we will use our own task files, to ensure compatibility we provide an incomplete PDDL files (Note, you can make different modeling choices in your written submission. There are other ways to deal with doors.)

6) **(20) Extra Credit:** Extend the problem to include a the charge state of the robot. Assume that a robot has a battery capacity of 10 and that each action consumes one charge. The robot can recharge in its room and only perform actions if the charge is greater than 0.