NATALIA PATTARONE
MARIA MANCHENO

# Homework 3 - Deterministic Planning

Autonomous Systems

2021-03-10

## Exercise C.1

### (a) Monkey and Bananas - Describe the Model.

For the coding solution, see attached pddl files.

For this particular problem, we model it as follows:

1. 3 `:constants` that do not change: monkey, bananas and chair
2. 5 `:predicates` defining our parametrized (or not) variables: location `?x`, on-floor (true or false), at `?m ?x` (to indicate monkey position), on-chair `?x` (if it is in the chair, where is positioned) and hasbananas (true or false).
3. 4 :actions describing the possible movements in the environment where the effect is presented as the final result of performing each action, for example GET-BANANAS would have the effect that `hasbananas` turn to be True if all preconditions are met:
   - MOVE-MONKEY, has 2 parameters to know from where to where do the moving, preconditioned by the 2 given positions, that the monkey is on-floor and not in the final position

   - CLIMB, has 1 parameter to allow the monkey climb over the chair to grab the bananas, preconditioned by that monkey, chair and location are all in the same location parameter

   - MOVE-CHAIR, has 2 parameters to know from where to where move the chair, preconditioned by the 2 positions and that neither the monkey nor the chair are already at the final position parameter, and that the monkey is on-floor

   - GET-BANANAS, has 1 parameter indicating the location of the grabbing action, preconditioned to that monkey is on-chair, chair is in location of the parameter as well as the bananas

## Exercise C.2

### (a) Describe how would you model in PDDL a variation of the standard Sokoban where the agent can teleport herself to any empty location of the grid, but can do that only once. Teleporting into a wall or in the same location as a box is not allowed; teleporting into a goal location is allowed. What predicates are you going to need to encode grid cells, wall, player and box positions, and also be able to describe the goal situation with a logical formula?

For the coding solution, see attached pddl files.

For this particular problem, we model it as follows:

1. 1 `:player`
2. 7 `:predicates` defining our parametrized (or not) variables: clear ?l, at ?t, at-goal ?s ?x (to indicate position), IS-GOAL ?l (if position is in goal), IS-NONGOAL ?l (if position is not in goal), MOVE-DIR ?from ?to and teleported.
3. 4 :actions describing the possible movements in the environment where the effect is presented as the final result of performing each action. The possible actions are:
   - Move, has 1 parameter to know from where to where do the moving, preconditioned by the position and the direction it should move on.

- Teleport, has 1 parameter to allow the player to teleport from one location to another.
- Push-to-nongoal, has 3 parameters so that the player is able to move the box to a tile that is not a goal position.
- Push-to-goal, has the same 3 parameters to indicate when the boxes are moved to a goal position.

## (c) How many of the problems can you solve in each case? How much do the number of steps in the non-optimal solutions differ from the optimal ones? Are the Sasquatch problems too easy or too difficult for your compilation-to-PDDL approach? If they are too easy, what do you think would make them more difficult? If they are too difficult, what could make them easier to solve?

As we can see from the file levelssatis.txt the optimal and satisficing techniques were not too efficient, meaning that the Sasquatch problems were too difficult for our compilation-to-PDDL approach. In fact, when using lama as the satisficing technique no level was solved in less than 60 seconds. However, due to this reason, we changed the limit time to $180s$ and it did not do significantly better, but at least it solved 2 levels.Level 14, for example, took $139.65s$ to solve and did it in 12 steps. It was only able to solve one more level, which in this case was level 2. For this level the technique took $87.29s$ and it took 9 steps to do so. Considering that the problems were too hard for the PDDL compilation, one thing that could work to make it easier would be decreasing the field size as the number of predicates would be smaller and so easier to find solutions to it.

As for the optimal technique we used as the planner LM-CUT. It didn't solve more levels than the satisficing tecniques but it did solvem in less time. For example, level $2$ was solved in $66.98s$ which was almost in the $60s$ limit we wanted. For level 14, it took $118.33s$ so it didn't do significantly better.