SAARLAND UNIVERSITY AUTOMATIC PLANNING
Dr. Álvaro Torralba
Dr. Cosmina Croitoru, Daniel Gnad, Marcel Steinmetz

Exercise Sheet 1.
Solutions due Wednesday, **November 7**, 23:59, in Moodle. [1]

Exercise 1. (10 Points)

Encode an arbitrary domain of your choice in PDDL, along with at least two example instances. You may take as inspiration different types of (1-player) games, some real world inspired application, or PSPACE complete problem[2]. If some characteristics of the problem are two hard to model you may simplify them in your model.

You have to:

- Model the domain (domain.pddl) and two problem instances (p1.pddl and p2.pddl).
- Describe the problem that you are modelling (by adding a comment on top of your domain.pddl file), describing any differences or simplifications with respect to the original problem (if you based your model on some existing problem).
- Use a planner (e.g. FF, or the solver in `http://editor.planning.domains`) to solve these instances, and report its output.

Note that, apart of whether the model is correct, we will value the **originality** and **complexity** of the problem you are modelling. As a general guidance on how complex a domain should be to get maximum points, we will value if some actions have interesting preconditions. For example, one could get the Nurikabe puzzle (`https://en.wikipedia.org/wiki/Nurikabe_(puzzle)`) and modify it (e.g. a robot needs to paint a floor with a pattern that corresponds to the solution of a nurikabe puzzle) with some simplifications (e.g. ignoring the rule "There must be only one sea" of the nurikabe puzzle).

[1] At most 3 authors per solution. Include the names and matriculation numbers of all authors in the solution. Submission of solutions is via the course Moodle pages. Planning models must be submitted in PDDL format (domain.pddl and problem.pddl), everything else in a single PDF (sheet01.pdf). Put all solutions into a folder named "name1-name2-...-sheet01" with the names of the students on the team and create a zipped tar of it. To submit this file, go to the Assignment in Moodle, click "Add submission" and upload it. Do not forget to click "Save changes" after uploading the file. Until the deadline you can always update your submission by clicking "Edit submission". Note that only one student of each group should submit the solutions.
Please note that you are *not* allowed to copy from other groups!

[2] `https://en.wikipedia.org/wiki/List_of_PSPACE-complete_problems`

Also, please keep in mind that we are aware of most existing domains in Internet so you may use them to take inspiration for ideas on what to model but do not copy them, you must write your own PDDL files.

Exercise 2. (10 = 2 + 2 + 2 + 2 + 2 Points)

Consider the following planning task where a ghost wants to carve some patterns ($CarveTools = \{smiley, cat, bat\}$) on two pumpkins ($Pumpkins = \{Hokkaido, Muskat\}$) using his two hands ($Hands = \{left, right\}$). The ghost uses his hands only to carve the pumpkin and not to lift them up. The variable L represents to which of the pumpkins the robot is currently looking at (or none of them). There is one variable A_c for each tool to carve $c \in CarveTools$ that represent whether the carving tool c is available to be picked-up. There is one variable H_h for $h \in Hands$ that represent whether the ghost has hand h empty or has something in it. Finally, there is a variable $D_{p,c}$ for each $c \in CarveTools$ and $p \in Pumpkins$ that represents whether pumpkin p has the design c or not (i.e. a pumpkin can be carved with more than one thing at the same time). Actually, we want to have a pumpkin with 2 designs.

Formally, the task $\Pi = \langle V, A, I, G \rangle$ is defined as follows:

- $V = \{L, D_{p,c}, A_c, H_h\}$ with
  $D(L) = \{None\} \cup Pumpkins$
  $D(D_{p,c}) = D(A_c) = \{True, False\}$
  $D(H_h) = \{None\} \cup CarveTools$

- $A = \{look\_at(i), pick\_up(h, c), drop(c, h), carve(p, c, h)\}$, with
  - $look\_at(i)$ for $i \in Pumpkins$:
    $pre : \{\}$
    $eff : \{L = i\}$
  - $pick\_up(h, c)$ for $h \in Hands, c \in CarveTools$:
    $pre : \{A_c = True, H_h = None\}$
    $eff : \{H_h = c, A_c = False\}$
  - $drop(c, h)$ for $c \in CarveTools, h \in Hands$
    $pre : \{H_h = c\}$
    $eff : \{H_h = None, A_c = True\}$

- *put_stick*(p, c, h) for $p \in Pumpkin, c \in CarveTools, h \in Hands$:
  $pre : \{H_h = c, L = p\}$
  $eff : D_{p,c} = True$

- $I = \{L = None, H_{left} = None, H_{right} = None, A_{smiley} = True, A_{cat} = True, A_{bat} = True\} \cup \{D_{p,c} = False \mid p \in Pumpkin, c \in CarveTools\}$.
- $G = \{D_{Muskat,smiley} = True, D_{Muskat,cat} = True, D_{Hokkaido,bat} = True\}$

(i) Translate the planning task to STRIPS by defining the facts, actions, initial state and goal. You may use the sets as above to avoid writing down all combinations.

(ii) Draw the Causal Graph of the planning task.

(iii) Draw the DTG for Variables L and $D_{Muskat,cat}$. Annotate the transitions with their (outside) conditions. Which of those transitions are invertible according to the definition given in the lecture (Chapter 5)?

(iv) Explain which task simplification techniques from Chapter 5 can be applied in this planning task. Explain for which (sub-)problem(s) do we need to run a search algorithm. Give a plan for the simplified task and explain how will you recover the solution plan for the original task afterwards?

(v) Suppose that we pre-assign what to carve to the hands so that we only allow the left hand to carve the cat and the right hand the bat. Re-draw the causal graph. How does this affect to your answer above in part (iv)? Justify your answer.

Exercise 3. (2 Extra Points)

In the table on the slide 15 in Chapter 2, the number of distinct states in Blocksworld with n blocks are listed. This corresponds to the no-arm version where the arm is not modeled in the state and blocks are directly moved from one block (or the table) to another (or left on the table). In this version, two states are equal if they have the same number of piles on the table and the piles from the two states can be paired such that they are equal – have the same blocks in the same order. Prove that for $n = 4$ the number of reachable states with this formulation is indeed 73.

Exercise 4. (3 Extra Points)

Prove that PlanOpt for STRIPS is **PSPACE**-hard. For that purpose, use a polynomial reduction from PlanSat to PlanOpt.

Note: We assume here (and throughout the course) that the input STRIPS task $\Pi = (P, A, c, I, G)$ is given in explicit form, i.e., each of F, I, and G is given as a finite list of facts, and A is given as a finite list of actions.