Foundations of Artificial Intelligence

M. Helmert
S. Eriksson
Spring Term 2022

University of Basel
Computer Science

Exercise Sheet 10

Due: May 8, 2022

Important: for submission, consult the rules at the end of the exercise. Non-adherence to the rules will lead to your submission not being corrected.

Exercise 10.1 (1 mark)

Formally define a state space with four states, seven transitions, two actions and two goal states. Pay attention to define *all* aspects of the state space.

Exercise 10.2 (1+0.5+1+1 marks)

In this exercise we consider a PDDL representation of a variant of *Blocks World*, where moving blocks requires to first pick them up with your hand and then put them down again. Download the archive `blocks-world.zip` from the course website or ADAM. The domain file `blocks-world.pddl` describes the variables and actions of Blocks World, while the concrete problem instance is described in `blocks-world-problem.pddl`.

Parts (b) to (d) also require the domain-independent planning system *Fast Downward*. You can download either the sources or one of various container files from

`http://www.fast-downward.org/Releases/21.12`

The page also provides links to instructions for compilation and container usage. If you encounter technical problems, please let us – the tutor or assistant – know *sufficiently ahead of the due date.*

(a) Provide a graphical representation of the problem in a similar way as depicted in part (d).

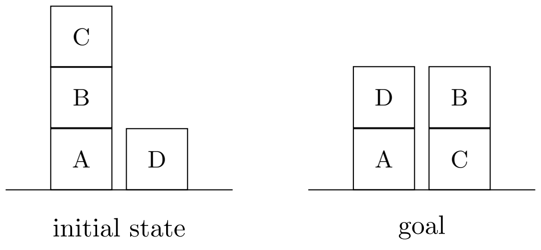
(b) Use Fast Downward with a configuration that performs greedy best-first search with the delete relaxation heuristic FF to solve the problem. To do so, invoke the planner with

```
./fast-downward.py blocks-world.pddl blocks-world-problem.pddl
--search "eager_greedy([ff()])"
```

Provide the number of expanded states and the found plan.

(c) Modify the PDDL files such that you can pick up an unlimited amount of blocks, and solve the resulting problem with the Fast Downward configuration from part (b). Provide the number of expanded states as well and found plan.

(d) Formalize the following Blocks World instance in a new file `blocks-world-problem-d.pddl` and solve it with the Fast Downward configuration from part (b). Use the original domain `blocks-world.pddl`, where only one block may be held at a time. Provide the number of expanded states and the found plan.



For part (c), add the modified files `blocksworld.pddl` and `blocksworld-problem.pddl`, and for (d) the new file `blocks-world-problem-d.pddl` to your submission.

Exercise 10.3 (1+1 marks)

Specify a STRIPS task with the following properties or justify why such a task cannot exist.

(a) An unsolvable task Π whose relaxation Π^+ is solvable.

(b) A parameterized planning task Π_n with n variables where the longest *simple path* in the state space of Π_n^+ has length $n+1$. For a directed graph $G = (V, E)$, a simple path of length n is a sequence $\langle v_0, \ldots, v_n \rangle$ with $(v_{i-1}, v_i) \in E$ for all $1 \leq i \leq n$ and $v_i \neq v_j$ for all $i \neq j$.

Hint: Consider how the size of the set of variables representing a state changes when following a simple path in the state space of Π^+. Can it get smaller, stay equal, get bigger?

Exercise 10.4 (1.5+0.5+0.5+0.5+0.5 marks)

Consider the STRIPS planning task $\Pi = \langle V, I, G, A \rangle$ with $V = \{a, b, c, d, e, f\}$, $I = \{a\}$, $G = \{d\}$, and $A = \{a_1, a_2, a_3, a_4, a_5\}$ with $cost = \{a_1 \mapsto 1, a_2 \mapsto 1, a_3 \mapsto 10, a_4 \mapsto 2, a_5 \mapsto 1\}$ and

$$\begin{aligned}
&pre(a_1) = \{a\} && add(a_1) = \{b\} && del(a_1) = \{\} \\
&pre(a_2) = \{a\} && add(a_2) = \{c\} && del(a_2) = \{\} \\
&pre(a_3) = \{b\} && add(a_3) = \{d\} && del(a_3) = \{b\} \\
&pre(a_4) = \{b, c\} && add(a_4) = \{e, f\} && del(a_4) = \{c\} \\
&pre(a_5) = \{e, f\} && add(a_5) = \{d\} && del(a_5) = \{c, e\}.
\end{aligned}$$

(a) Provide the relaxed planning graph for Π up to depth 3 (i.e., the resulting graph should have four variable layers and three action layers). You can omit all goal vertices except G^3.

(b) Compute $h^{\max}(I)$. Provide the values for all nodes in the RPG.

(c) Compute $h^{\text{add}}(I)$. Provide the values for all nodes in the RPG.

(d) Compute $h^{\text{FF}}(I)$. Provide the marked RPG.

(e) Why do we need to build the graph to depth 3 for computing $h^{\max}$ and h^{add}, even though depth 2 already reaches all goal variables?

Submission rules:

- Exercise sheets must be submitted in groups of two students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending .pdf) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Either use page numbers on all pages or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, only create those code textfiles required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded.
- For the submission: if the exercise sheet does not include programming exercises, simply upload the single PDF. If the exercise sheet includes programming exercises, upload a ZIP file (ending .zip, .tar.gz or .tgz; *not* .rar or anything else) containing the single PDF and the code textfile(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to ADAM, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.