Planning and Optimization

M. Helmert, G. Röger
C. Büchner, R. Christen, S. Dold

University of Basel
Fall Semester 2023

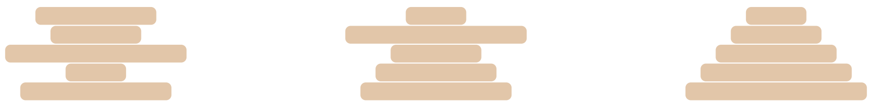
Exercise Sheet 6

Due: November 13, 2023

Important: For submission, consult the rules at the end of the exercise. Non-adherence to these rules might lead to a penalty in the form of a deduction of marks or, in the worst case, that your submission will not be marked at all.

Exercise 6.1 (0.5+0.5+2+1 marks)

In the *pancake problem* we have n pancakes with size $1, \dots, n$ on a pile. The goal is to order the pile by size, i.e., the largest pancake is on the bottom, the second largest on top of the largest and so on. The pile can be manipulated by removing a stack of i pancakes from the top and putting it back in reverse order. Hence, there is an action flip_i for all $1 \leq i \leq n$. The following figure shows an example instance:



Let $P = \{p_1, p_2, p_3, p_4, p_5\}$ be the set of pancakes where p_1 is the smallest pancake and p_5 is the largest, and let $N = \{1, 2, 3, 4, 5\}$ be the set of positions in the pile where 1 is on top and 5 is at the bottom. Consider the following formalization of the above instance of the pancake problem as a propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$:

- $V = \{p_i\texttt{-at-}j \mid p_i \in P, j \in N\}$;
- $I = \{v \mapsto \mathbf{T} \mid v \in V_I\} \cup \{v \mapsto \mathbf{F} \mid v \in V \setminus V_I\}$ with $V_I = \{p_1\texttt{-at-}4, p_2\texttt{-at-}2, p_3\texttt{-at-}1, p_4\texttt{-at-}5, p_5\texttt{-at-}3\}$;
- $O = \{\mathit{flip}_i \mid 1 \leq i \leq 5\}$ with $\mathit{flip}_i = \langle \top, \mathit{eff}_i, 1 \rangle$ such that

$$\mathit{eff}_i = \bigwedge_{p \in P} \bigwedge_{j=1}^{i} (p\texttt{-at-}j \rhd (\neg p\texttt{-at-}j \land p\texttt{-at-}k))$$

  where $k = i - j + 1$; and
- $\gamma = \bigwedge_{i=1}^{5} p_i\texttt{-at-}i$.

(a) Provide the invariant stating that at most one of p_1, p_2 and p_3 may be on the bottom of the pile at once.

Note: There is no good reason to leave out p_4 and p_5 from this invariant apart from reducing the effort needed to solve this exercise.

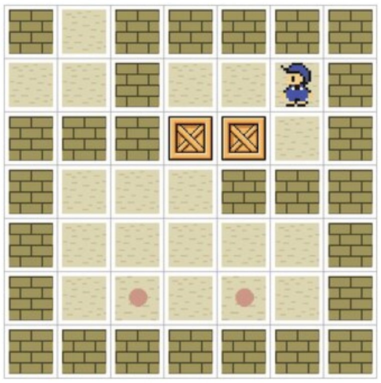
(b) Provide a mutex group stating that a certain pancake cannot be at multiple positions in the pile at once.

(c) Formalize the above pancake instance as a planning task $\Pi' = \langle V', I', O', \gamma' \rangle$ in finite-domain representation such that V' contains exactly one variable per pancake (i.e., $|V'| = 5$).

(d) Compare the two planning tasks Π and Π'. How many states does each of them have? How many of those states are reachable?

Exercise 6.2 (0.5+0.5+0.5+0.5 marks)

In the *Sokoban* domain, a worker has to push boxes to goal positions, but cannot pull them. The figure below illustrates an example problem. The goal is to push one box to each tile indicated by a red dot. (It does not matter which box is located at which position.) In any given state, the worker may move to an empty tile adjacent to its current location, where empty means that tile is neither a wall nor there is a box. If there is a box, the worker may still move there if the tile behind the box (from the worker's perspective) is empty, pushing the box to that empty tile.



In the following, we suggest four abstraction functions for Sokoban problems. While they might seem reasonable at a first glance, all of them come with different practical limitations. Point out and explain the problems with these suggestions in 2–3 sentences each.

(a) α_1: Each state is mapped to the number of boxes that are on a goal location.

(b) α_2: Each state is mapped to an abstract state by ignoring the position of the agent.

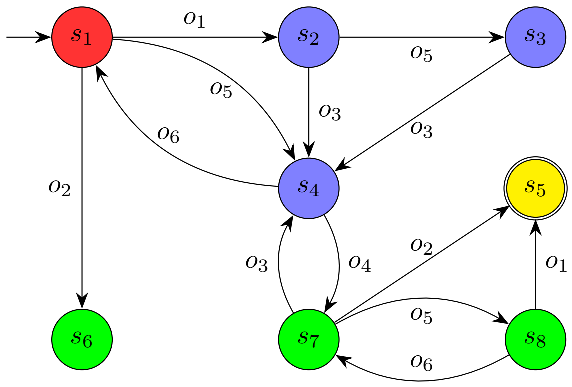
(c) α_3: Each state s is mapped to $f(s) \mod n$ where f is a bijection from the set of states S to the natural numbers from 1 to $|S|$ (i.e., $f\colon S \to \{1, \dots, |S|\}$) and $n = 10^6$ is used to limit the number of abstract states.

Hint: You may assume that it is possible to evaluate $f(s)$ for a given state s efficiently. In fact, we will discuss a specific function of this kind in lecture E6 (perfect hash function for Pattern Database heuristics).

(d) α_4: A state is mapped to s_1 if 5 or fewer moves are necessary to move all boxes to a goal location; it is mapped to s_2 if between 5 and 10 moves are necessary; and so on.

Exercise 6.3 (0.5+1+1.5+1 marks)

Consider the transition system $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$ with S, L, T, s_0 and $S_\star$ as depicted below and with $c(o_i) = i$ for all $1 \le i \le 6$.



(a) Consider the abstraction α that maps all states depicted in the same color to the same abstract state, i.e., $\alpha(s_1) = s_r$, $\alpha(s_2) = \alpha(s_3) = \alpha(s_4) = s_b$, $\alpha(s_5) = s_y$, and $\alpha(s_6) = \alpha(s_7) = \alpha(s_8) = s_g$. Graphically provide $\mathcal{T}^\alpha$ and give h^α.

(b) Assume you may change the abstraction α from part (a) by mapping one concrete state to another (already existing) abstract state. If you care about having some positive effect on the heuristic quality, which change do you make? Justify your answer. (There are multiple reasonable options.)

(c) Provide an abstraction β of $\mathcal{T}$ such that $|S^\beta| = 4$ and such that there is no abstraction $\beta' \neq \beta$ with $|S^{\beta'}| = 4$ and $h^{\beta'}(s_1) > h^\beta(s_1)$. Graphically provide the transition system $\mathcal{T}^\beta$.

(d) Consider the abstraction α from part (a). Provide an abstraction α' of $\mathcal{T}$ and a function α'' such that

- $\alpha' \neq \alpha$,
- α' is a coarsening of α,
- $h^{\alpha'}(s_1) = h^\alpha(s_1)$, and
- $\alpha' = \alpha'' \circ \alpha$.

Submission rules:

- Exercise sheets must be submitted in groups of 2–3 students. Create a team on ADAM including all members of your group and submit a single copy of the exercises per group.
- 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 text file(s) 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 text file(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly. After creating your zip file and before submitting it, open the file and verify that it complies with these requirements.
- 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.