Foundations of Artificial Intelligence

M. Helmert, M. Wehrle
T. Keller
Spring Term 2016

University of Basel
Computer Science

Exercise Sheet 2

Due: March 11, 2016

Exercise 2.1 (1.5+1.5 marks)

Characterize the following environments by describing if they are *static / dynamic*, *deterministic / non-deterministic / stochastic*, *fully / partially / not observable*, *discrete / continuous*, and *single-agent / multi-agent*. Explain your answer.

(a) Klondike Solitaire (the Solitaire game that comes with every Windows distribution, see e.g. https://en.wikipedia.org/wiki/Klondike_(solitaire))

(b) Soccer Robot

Exercise 2.2 (3 marks)

Determine if the following statements about state spaces $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ are correct or not. Explain your answer.
Remark: The cardinality of a set X is denoted by $|X|$.

(a) If all actions have equal costs, each solution for $\mathcal{S}$ is optimal.

(b) There is no solution for $\mathcal{S}$ if $T = \emptyset$.

(c) From $|S| < \infty$ and $|A| < \infty$ it follows that $|T| < \infty$.

(d) If T is finite, the set of paths in $\mathcal{S}$ is finite as well.

(e) If $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ is a minimal cost path from state $s \in S$ to state $s' \in S$ and $\pi' = \langle \pi'_1, \ldots, \pi'_m \rangle$ is a minimal cost path from s' to state $s'' \in S$, then $\pi'' = \langle \pi_1, \ldots, \pi_n, \pi'_1, \ldots, \pi'_m \rangle$ is a minimal cost path from s to s''.

(f) Let π, π', and π'' be minimal cost paths from $s \in S$ to $s' \in S$, s' to $s'' \in S$ and s to s'', respectively. Then $cost(\pi'') \leq cost(\pi) + cost(\pi')$.

Exercise 2.3 (0+4+2 marks)

The task in this exercise is to write a software program. We expect you to implement your code on your own, without using existing code you find online. If you encounter technical problems or have difficulties understanding the task, please let us – Patrick Buder or Thomas Keller – know *sufficiently ahead of the due date*.

(a) You can find sample Java code for the state space of the blocks world problem that was presented in the lecture on the website of the course. It implements the provided `State` and `Action` interfaces as well as the black box interface for state spaces that was discussed in the lecture. Test the implementation by invoking the `StateSpaceTest` class, which creates a set of random successor states starting from the initial state.

(b) Implement the state space of a blocks world variant where

- the maximum number of table positions (and hence towers) is limited by a constant
- the maximal height of each tower is limited by a constant (separately for each table position)

- the goal is to create towers of specific blocks in specific table positions

(c) Implement the `buildFromCmdline` function for the blocks world variant such that it parses an input file with the following syntax:

- first line: total number of blocks and maximum number of table positions
- one line for each table position: maximum number of blocks in the tower at this table position, followed by the IDs of the blocks in the initial state, starting from the surface of the table (end with `-1`)
- one line for each table position: IDs of the blocks in the goal state, starting from the surface of the table (end with `-1`)

You can find the sample input file `alt_bw_inst_1` on the website of the course. It encodes an instance with 4 blocks and 3 table positions. The tower in the first table position has a maximum height of 1 and is empty initially, the tower in the second table position has a maximum height of 4 and contains the tower of blocks with IDs 1, 2, and 4 and the tower in the third table position has a maximum height of 2 and contains a tower that consists only of block 3. The goal is to have a tower of blocks with increasing IDs in the second table position.

The exercise sheets can be submitted in groups of two students. Please provide both student names on the submission.