Automated Planning (TDDD48)

Jendrik Seipp
Mika Skjelnes

Linköping University

Lab 2

Important: For submission, consult the rules at the end of this document. Non-adherence to these rules might lead to a penalty in the form of a deduction of points. Some points are *bonus points*. These can help you reach the point quota per lab (4/12 points) and the overall point quota ($50\% \cdot 7 \cdot 12 = 42$ points).

Exercise 2.1 (0.5+0.5+1 points)
Let $s = \{a \mapsto \mathbf{T}, b \mapsto \mathbf{F}, c \mapsto \mathbf{T}, d \mapsto \mathbf{T}\}$ be a state and o_1, o_2 be operators for a planning task Π, with

$$o_1 = \langle a \wedge \neg b, \neg a \wedge \neg d \wedge ((b \vee c) \rhd a) \rangle \text{ and } o_2 = \langle b \wedge d, ((a \wedge b) \rhd (c \wedge d)) \rangle.$$

(a) What is $\mathit{effcond}(\neg d, \mathit{eff}(o_1))$?

(b) What is $\mathit{effcond}(c, \mathit{eff}(o_2))$?

(c) Is o_1 applicable in s? Is o_2 applicable in s? If yes, provide $s[\![o_1]\!]$ and/or $s[\![o_2]\!]$.

It is sufficient to give the final results (after applying equivalence transformations for simplification). In LaTeX, you can use `\top` for $\top$ and `\bot` for $\bot$.

Exercise 2.2 (2 points)
Consider a simplification of the logistics task from chapters B2 and B3: Let $\Pi' = \langle V', I', O', \gamma' \rangle$ be a planning task with

- $V' = \{i, p, t\}$;
- $I' = \{i \mapsto \mathbf{F}, p \mapsto \mathbf{T}, t \mapsto \mathbf{F}\}$;
- $O' = \{m', l', u'\}$ where
  - $m' = \langle \top, (t \rhd \neg t) \wedge (\neg t \rhd t), 5 \rangle$,
  - $l' = \langle \neg i \wedge (p \leftrightarrow t), i, 1 \rangle$, and
  - $u' = \langle i, \neg i \wedge (t \rhd p) \wedge (\neg t \rhd \neg p), 1 \rangle$; and
- $\gamma' = \neg i \wedge \neg p$.

Note: Variable p denotes the position of the package when it is not inside the truck. Otherwise, it denotes the location where it was loaded, which is irrelevant in this problem.

Provide a graphical representation of $\mathcal{T}(\Pi')$. Label states to clearly express the value of all variables in each state, make sure to indicate initial and goal states, and label the transitions.

Exercise 2.3 (2+2 points)
Consider the propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$ with

$$
\begin{aligned}
V &= \{a, b, c, d\} \\
I(v) &= \mathbf{F} \quad \text{for all } v \in V \\
O &= \{o_1, o_2, o_3, o_4\} \\
\gamma &= d
\end{aligned}
$$

and

$$
\begin{aligned}
o_1 &= \langle \neg b, c \rangle \\
o_2 &= \langle \top, b \rangle \\
o_3 &= \langle \neg a, a \rangle \\
o_4 &= \langle a \wedge b, \neg c \wedge d \rangle
\end{aligned}
$$

(a) Plot the search tree explored by a progression breadth-first search of Π. Generate successors in the order of increasing operator indices (i.e., o_1 before o_2 etc.). Mark states that are duplicates of previously generated states as such and prune them (i.e., do not expand them further).

(b) Plot the search tree explored by a regression breadth-first search of Π. Generate and expand nodes in the order described in part (a). Simplify the state formula as much as possible at every node of the search tree. Mark nodes with an unsatisfiable formula or one that logically entails the state formula of a previously expanded node and prune them.

In the regression search space the search nodes are formulas that represent sets of world states. Note that here the search should not only prune nodes that are exact duplicates (i.e., they have the same formula) but also nodes where the set of represented world states is a subset of the set represented by a previously expanded node (e.g., the parent node). For example, if we have already expanded a node n_1 with formula x and later generate a node n_2 with formula $x \wedge \neg y$, then we can prune n_2, because $x \wedge \neg y$ entails x, i.e., all states represented by n_2 are already represented by n_1, so we don't need to handle them again.

Exercise 2.4 (0.5+0.5+2+1 points)
Consider the propositional planning task $\Pi = \langle V, I, O, \gamma \rangle$ with

- $V = \{x, y, z\}$;
- $I = \{x \mapsto \mathbf{T}, y \mapsto \mathbf{T}, z \mapsto \mathbf{F}\}$;
- $O = \{o_1, o_2, o_3\}$ where
  - $o_1 = \langle x, \neg x \wedge (y \rhd z), 2 \rangle$,
  - $o_2 = \langle z, x \wedge \neg y \wedge \neg z, 1 \rangle$, and
  - $o_3 = \langle \top, (\neg y \rhd y) \wedge (y \rhd \neg y), 2 \rangle$; and
- $\gamma = \neg(x \vee z)$.

In this exercise, your task is to define a sequential SAT-encoding for Π.

(a) Provide the clauses that encode the initial state I.

(b) Provide the clauses that encode the goal γ at horizon T.

(c) Provide all clauses that encode the transitions induced by o_1 for some time step i. Simplify the clauses and omit those that simplify to $\top$. (You do not need to provide intermediate results for the simplification.) Annotate each remaining clause as precondition clause, positive or negative effect clause, or positive or negative frame clause.

Note that even though o_1 is not a STRIPS operator, the resulting transition formula is in conjunctive normal form and thus does not need an extra conversion step to CNF.

(d) Assuming the precondition, effect and frame clauses (positive and negative) induced by o_2 and o_3 are given, which clauses are yet missing to obtain a complete SAT-encoding? Provide them explicitly, again parametrized for some time step i.

Bonus Exercise 2.5 (2+1 **bonus** points)

(a) Model a binary counter with three bits as a propositional planning task $\Pi_1 = \langle V_1, I_1, O_1, \gamma_1 \rangle$. The initial value of the counter is 0 and the goal is to obtain a value of 7. The only allowed operation is to increment the current value by 1, so your model should have a single operator.

(b) Replace O_1 in your previously defined model with a set of operators O_2 without conditional effects. Your model should still resemble a binary counter.

Hint: For (b) you need more than one operator to implement the single logical operation of incrementing the counter by 1.

Submission rules:

- Lab sheets must be submitted in groups of 2–3 students. Clone the labs repo (`https://github.com/mrlab-ai/tddd48-labs`) and push it to a repo at the University GitLab instance `https://gitlab.liu.se`. Make sure the repo is **private** and give read access to Mika Skjelnes (`mika.skjelnes@liu.se`).
- For non-programming exercises, create a single PDF file at the location `labX/solution.pdf`. 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. 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, directly edit the code in the cloned repository and only create those code text file(s) required by the lab. 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.