Authors:
Erick Rosete Beas — er165@uni-freiburg.de
Jessica Lizeth Borja Diaz — jb986@uni-freiburg.de

Principles of AI Planning

Exercise Sheet 11

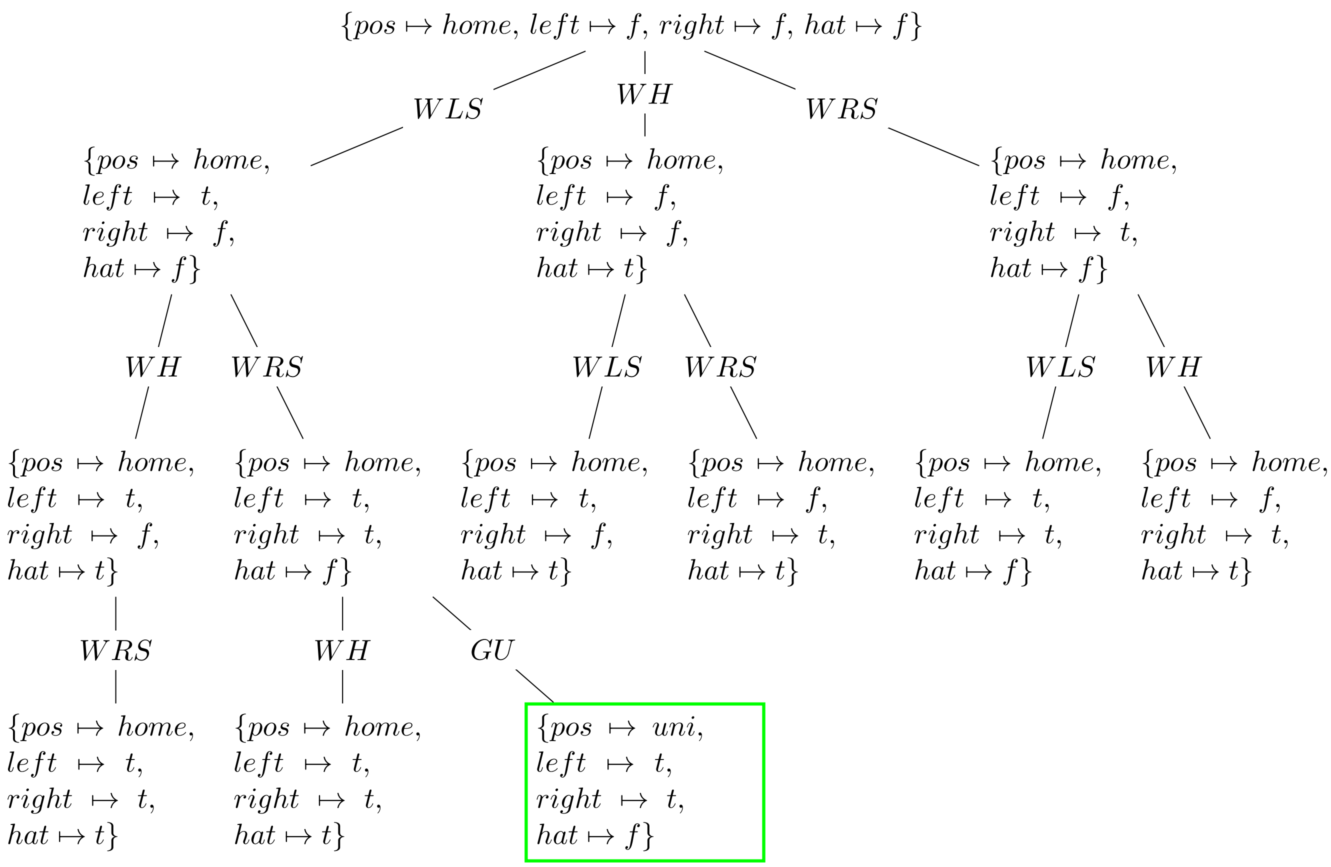
24.01.2020

Exercise 11.1 - Strong stubborn sets

Consider the SAS^+ planning task Π with variables $V = \{pos, left, right, hat\}, \mathcal{D}_{pos} = \{home, uni\}$ and $\mathcal{D}_{left} = \mathcal{D}_{right} = \mathcal{D}_{hat} = \{t, f\}$. The initial state $I = \{pos \mapsto home, left \mapsto f, right \mapsto f, hat \mapsto f\}$ and the goal specification is $\gamma = \{pos \mapsto uni\}$. There are four operators in O, namely

$$\begin{aligned}
\text{wear-left-shoe (WLS)} &= \langle pos = home \wedge left = f, left := t\rangle \\
\text{wear-right-shoe (WRS)} &= \langle pos = home \wedge right = f, right := t\rangle \\
\text{wear-hat (WH)} &= \langle pos = home \wedge hat = f, hat := t\rangle \\
\text{go-to-university (GU)} &= \langle pos = home \wedge left = t \wedge right = t, pos := uni\rangle
\end{aligned}$$

(a) Draw the breadth-first search graph (with duplicate detection) for planning task Π without any form of partial-order reduction.



(b) Draw the breadth-first search graph (with duplicate detection) for planning task Π using strong stubborn set pruning. For each expansion of a node for a state s, specify in detail how T_s (and thus $T_{app(s)}$) are computed, i.e., explain how the initial disjunctive action landmark is chosen and how operators are iteratively added to T_s as part of necessary enabling sets or interfering operators, respectively. Break ties in favor of *wear-left-shoe* over *wear-right-shoe.*

How many node expansion do you save with strong stubborn sets compared to plain breadth-first search? What about the lengths of the resulting solutions?

$$
\begin{array}{c}
S_1 = \{pos \mapsto home,\, left \mapsto f,\, right \mapsto f,\, hat \mapsto f\} \\
| \\
WLS \\
| \\
S_2 = \{pos \mapsto home,\, left \mapsto t,\, right \mapsto f,\, hat \mapsto f\} \\
| \\
WRS \\
| \\
S_3 = \{pos \mapsto home,\, left \mapsto t,\, right \mapsto t,\, hat \mapsto f\} \\
| \\
GU \\
| \\
\boxed{\{pos \mapsto uni,\, left \mapsto t,\, right \mapsto t,\, hat \mapsto f\}}
\end{array}
$$

The strong stubborn set in S_1 is determined by taking into account the disjunctive action landmark, we know the operator *go-to-university* must be inside this set.
As *go-to-university* is not applicable in the initial state, then we need to add a neccessary enabling set for this operator, a possible NES is $\{wear\text{-}left\text{-}shoe\}$ or $\{wear\text{-}right\text{-}shoe\}$, we broke ties by adding the operator *wear-left-shoe* to the strong stubborn set.
As *wear-left-shoe* is applicable, we must add all operators that interfere, the only operator that interferes with this operator is *go-to-university* but it is already in the set, therefore:

$$T_{S_1} = \{go\text{-}to\text{-}university, wear\text{-}left\text{-}shoe\}$$
$$T_{S_1} \cap T_{app(S_1)} = \{wear\text{-}left\text{-}shoe\}$$

The strong stubborn set in S_2 is determined using again $\{go\text{-}to\text{-}university\}$ as DAL.
As *go-to-university* is not applicable in S_2, then the smallest NES is $\{wear\text{-}right\text{-}shoe\}$
As *wear-right-shoe* is applicable, we must add the operator *go-to-university* as it interferes, but it is already in the set, therefore:

$$T_{S_2} = \{go\text{-}to\text{-}university, wear\text{-}right\text{-}shoe\}$$
$$T_{S_2} \cap T_{app(S_2)} = \{wear\text{-}right\text{-}shoe\}$$

The strong stubborn set in S_3 is determined using again $\{go\text{-}to\text{-}university\}$ as DAL.
As *go-to-university* is applicable in S_3, then we need to add all operators that interfere, in this case *go-to-university* disables $\{wear\text{-}left\text{-}shoe, wear\text{-}right\text{-}shoe, wear\text{-}hat\}$

$$T_{S_3} = \{go\text{-}to\text{-}university, wear\text{-}left\text{-}shoe, wear\text{-}right\text{-}shoe, wear\text{-}hat\}$$
$$T_{S_3} \cap T_{app(S_3)} = \{go\text{-}to\text{-}university, wear\text{-}hat\}$$

In this case the first visited node was a goal state, after that we stopped the search.

By pruning with the strong stubborn set we only visited 4 states, in contrast without pruning we visited 13 states, 9 states more, which represents approximately a 30% of the previous graph.

Exercise 11.2 - Weak vs. strong stubborn sets

Show that *weak* stubborn sets admit exponentially more pruning than *strong* stubborn sets.

Consider the family of planning tasks $(\Pi_n)_{n\in\mathbb{N}}$, where $\Pi_n = \langle V_n, I_n, O_n, \gamma\rangle$ is the planning task with the following components:

- $V_n = \{a, x, y, b_1, ..., b_n\}$ with variable domains $\mathcal{D}_a = \mathcal{D}_x = \mathcal{D}_y = \{0, 1\}$ and $\mathcal{D}_{b_i} = \{0, 1, 2\}$ for all $i \in \{1, ..., n\}$
- $O_n = \{o, o', o_d, \overline{o_d}, o_1, ..., o_n, \overline{o_1}, ..., \overline{o_n}\}$
- $pre(o) = \{a \mapsto 0\}, eff(o) = \{x \mapsto 1\}$
- $pre(o') = \{a \mapsto 0\}, eff(o') = \{y \mapsto 1\}$
- $pre(o_d) = \{a \mapsto 0\}, eff(o_d) = \{a \mapsto 1, b_1 \mapsto 1, ..., b_n \mapsto 1\}$
- $pre(\overline{o_d}) = \{a \mapsto 1\}, eff(o_d) = \{a \mapsto 0, b_1 \mapsto 1, ..., b_n \mapsto 1\}$
- $pre(o_i) = \{b_i \mapsto 1\}, eff(o_i) = \{b_i \mapsto 2\}$ for $1 \leq i \leq n$
- $pre(\overline{o_i}) = \{b_i \mapsto 2\}, eff(\overline{o_i}) = \{b_i \mapsto 1\}$ for $1 \leq i \leq n$
- $I_n = \{a \mapsto 0, x \mapsto 0, y \mapsto 0, b_1 \mapsto 0, ..., b_n \mapsto 0\}$
- $\gamma = \{x \mapsto 1, y \mapsto 1\}$

Let's obtain the strong stubborn set for the given initial state of a planning task Π_j.
The disjunctive action landmark must contain operators o or o' as this operators are the only ones that can change the value of the variables x and y to 1.
We will consider both cases and arrive to the same stubborn set.
Case 1: DAL containing o
As the operator o is applicable we must add all operators that interfere with o, in this case operator o_d disables o, therefore it must be added to T_{I_j}.
As operator o_d is applicable we add all operators, as every operator interferes with o_d.
Explanation: we can see that o_d disables operators o, o' (because of a assignment) and $\overline{o_i}$ for $1 \leq i \leq j$ (because of b_i assignment), furthermore it conflicts with $\overline{o_d}$ as it assigns a different value to a and it conflicts with every o_i for $1 \leq i \leq j$ as it has a distinct b_i assignment.
Case 2: DAL containing o'
As o' is applicable we need to add o_d as it interferes with o', o_d is applicable and it interferes with all other operators, therefore the stubborn set is the set of operators

$$T_{I_j} = O_j = \{o, o', o_d, \overline{o_d}, o_1, ..., o_j, \overline{o_1}, ..., \overline{o_j}\}$$

In fact the same behavior will happen for future states as if we haven't reach the goal state we will still have the DAL set including operators o or o'.
We can see that if operator o (or o') is applicable we can follow the same logic presented before and if it is not, then the operator $\overline{o_d}$ must be included as part of the NES and as it is applicable we need to add all operators o_i, $\overline{o_i}$ and o_d as they interfere with it. This will result in no order reduction at all.

On the other hand the weak stubborn set for the initial state is the following:
The smallest DAL contains either o or o'.
Case 1: DAL containing o
As the operator o is applicable we search for operators that have conflicting effects with o or that are disabled by o. But we can see no precondition includes x therefore o doesn't disable any operator, and no other operator affects x then there are no conflicting operators.

$$T_{I_j} = \{o\}$$

Case 2: DAL containing o'
As the operator o' doesn't have conflicting effects or disables another operator.

$$T_{I_j} = \{o'\}$$

We can conclude that with the given family of planning task there is no way to make smaller the strong stubborn set as it may always contain all the applicable operators, but in contrast the weak stubborn set can contain only 1 operator in every planning task of this family.

Furthermore solving the planning task using the weak stubborn set will only take two steps, by using operator o followed by o' or viceversa. In contrast as there is in fact no pruning of operators in the strong stubborn set, we will need to explore exponentially more states.