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

Principles of AI Planning
Exercise Sheet 4

22.11.2019

Exercise 4.1: Example for general regression

a) Consider the STRIPS planning tasks, Solve this problem with a breadth-first search.
for simplicity we call the atoms the following way:

atoms	name
juliet-dancing	jd
juliet-at-home	jh
romeo-at-home	rh
romeo-dancing	rd
romeo-at-work	rw

$$A = \{romeo, juliet\}$$
$$I = \{rh \mapsto 1, jh \mapsto 1\}$$
$$\gamma = jd \wedge rh$$
$$O = \{o_1, o_2, o_3\}$$

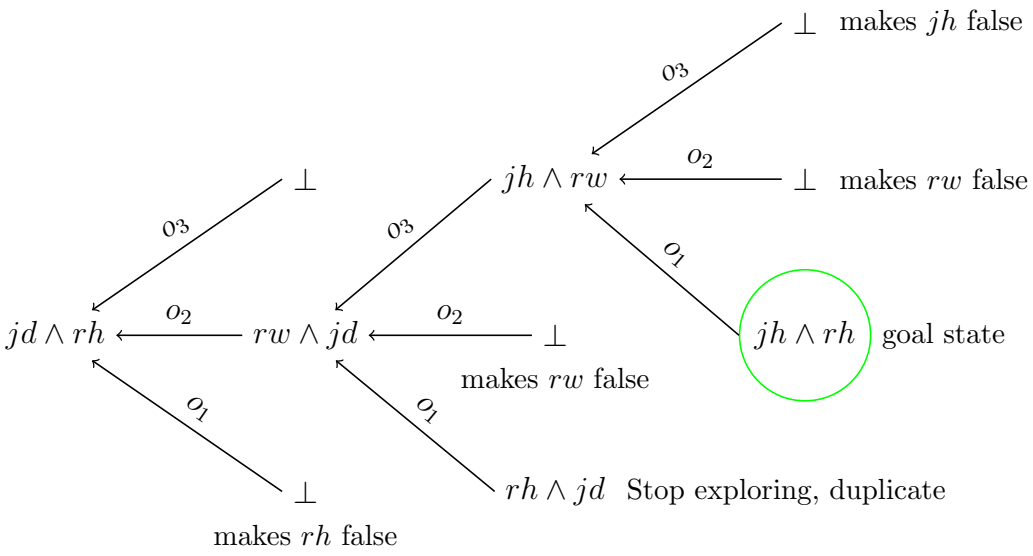
Go-work:	$o_1 = \langle rh, rw \wedge \neg rh \rangle$
Go-home:	$o_2 = \langle rw, rh \wedge \neg rw \rangle$
Go-dancing:	$o_3 = \langle jh, jd \wedge \neg jh \wedge (rh \triangleright (rd \wedge \neg rh)) \rangle$'

Operations:

$$\begin{aligned} regr_{o_3}(jd \wedge rh) &= jh \wedge (\top \vee (jd \wedge \neg\bot)) \\ &\wedge(\bot \vee (rh \wedge \neg rh)) \\ &= jh \wedge \top \wedge \bot = \bot \end{aligned}$$

$$regr_{o_2}(jd \wedge rh) = rw \wedge jd$$

$$\begin{aligned} regr_{o_3}(rw \wedge jd) &= jh \wedge (\top \vee (jd \wedge \neg\bot)) \\ &\wedge(\bot \vee (rw \wedge \neg\bot)) \\ &= jh \wedge rw \end{aligned}$$



Then the plan would be : o_1, o_3, o_2. This means: romeo goes to work, then juliet goes dancing but as romeo is at "work" he cannot go, finaly romeo returns happily to home.

Exercise 4.2 (Exponential plan length)

Show that for all $n \in N$ there is a conditional effect free planning task $\prod = \langle A, I, O, \gamma \rangle$ with $|A| = |O| = O(n)$ such that $|\pi| = O(2^{|n|})$ where $|\pi|$ is the length of a plan.

To prove it we will construct a planning task that works $\forall n \in N$, the task will be modeled as a binary counter, this counter will start in 0 and go to $2^n - 1$, and as we know in a binary counter you must pass from every state until reaching the goal state, therefore the optimal plan will have a length $\pi = 2^n$.

$$A = \{A_i, A_{i-1}, ..., A_1, A_0\}$$

Where A_i is the most significant bit, and A_0 is the least significant bit.

$$I = \{A_i \mapsto 0, A_{i-1} \mapsto 0, ..., A_1 \mapsto 0, A_0 \mapsto 0\}$$

The counter initial state starts with all bits in 0.

$$\gamma = \{A_i \mapsto 1, A_{i-1} \mapsto 1, ..., A_1 \mapsto 1, A_0 \mapsto 1\}$$

The goal will be all bits set to 1 as this represents the number $2^n - 1$.

$$O = \{O_0, O_1, ..., O_{i-1}, O_i\}$$
$$O_0 = \langle \neg A_0, A_0 \rangle$$
$$O_1 = \langle \neg A_1 \wedge A_0, A_1 \wedge \neg A_0 \rangle$$
$$O_2 = \langle \neg A_2 \wedge A_1 \wedge A_0, A_2 \wedge \neg A_1 \wedge \neg A_0 \rangle$$
$$...$$

$$O_i = \langle \neg A_i \wedge A_{i-1} \wedge ... \wedge A_1 \wedge A_0, A_i \wedge \neg A_{i-1} \wedge ... \wedge \neg A_1 \wedge \neg A_0 \rangle$$

The operators are the same size as the amount of variables and don't have conditional effects, which satisfies the problems conditions. Furthermore the operators flip all bits to the right to 0, and change the current bit to 1, which is what happens in a normal binary counter.

Describing the solution is similar to describing a recursive solution, we know that to set A_j to 1 we must use the operator O_j, then when A_j is turned to 1, the atoms/bits to the right are set to 0, and they must be set to 1 to reach the goal, to do these we must repeat the previous operators sequence, when they are all set to 1, then, O_{j+1} is the only operator than can be used and again all bits to the right are empty and we must repeat this process until setting all atoms to 1 and reaching the goal. Therefore this task work as the desired binary counter, always having an exponential length to reach the goal $\forall n \in N$. Q.E.D.