HW2: Classical Planning and Constraint Satisfaction

CS6300: Artificial Intelligence, Spring 2018 **Tucker Hermans**
University of Utah

---

1 Classical Planning

1. **Define the preconditions and postconditions (add and delete lists) for the two actions. Assume the agent can only pick up a single object with nothing on it, that it can only hold a single object at a time, and that it can only place the object on a horizontally-flat, open surface.**

   We will introduce three new fluents. The first *flat(x)* is true if x is a horizontally-flat box. The second *in_hand(x)* is true if x is in the agent's hand. Finally *empty(hand)* indicates the agent's hand is empty.

   Additionally, *clear(x)* indicates there is nothing on top of x and it's not in the agent's hand, but it is not necessarily flat. Finally, *on(x, y)* will be used to indicate object x is on object y.

   Using these, we can define the preconditions, add list, and delete list for *pick_up(x, y)* and *place(x, y)*. Note we expanded the definition of *pick_up(x)* to *pick_up(x, y)* to specify the object under x.

PC	**D**	**A**
clear(x)	*on(x, y)*	*clear(y)*
empty(hand)	*empty(hand)*	*in_hand(x)*
on(x, y)	*clear(x)*	

Table 1: Preconditions, add list, and delete list for *pick_up(x, y)*.

PC	**D**	**A**
in_hand(x)	*in_hand(x)*	*empty(hand)*
clear(y)	*clear(y)*	*clear(x)*
flat(y)		*clear(table)*
		on(x, y)

Table 2: Preconditions, add list, and delete list for *place(x, y)*.

   Note that *clear(table)* is included in *place(x, y)'s* add list in the case that y is the table.

2. **How would you express a goal that there are two towers, each of height 3 blocks, with pyramids on top of each tower?**

Let B_1, B_2, B_3, B_4, B_5, and B_6 be blocks. The goal state can be defined generically as a conjunction of fluents (broken into two lines for readability):

$$on(B_1, Table) \ \wedge \ on(B_2, B_1) \ \wedge \ on(B_3, B_2) \ \wedge \ \neg flat(B_3)$$
$$\wedge \ on(B_4, Table) \ \wedge \ on(B_5, B_4) \ \wedge \ on(B_6, B_5) \ \wedge \ \neg flat(B_6)$$

It is excessive to check if B_1, B_2, B_4, and B_5 are *flat* because the *on(x, y)* operations aren't possible if y isn't flat.

3. **Suppose we introduce a new type of *plank* object into the environment, which is long and skinny and must be supported by either the table or two blocks. What necessary changes would you have to make to your planning description to accommodate these changes?**

   First we will assume the supporting blocks must be flat.

   To accommodate the plank, we will add new actions, fluents, and goal states.

   The fluent *clear(plank)* represents there being room to fit another block on the plank (similar to the table). We will add *plank_on(x, y, z)* which is true when plank instance x is on y and z. In the case of the plank being on the table, both y and z are the table to avoid making too many new fluents. To determine which actions are valid on which objects, a $plank(x)$ fluent is added that is true when x is a plank object.

   A variation of *pick_up(x, y)* and *place(x, y)* are added as actions that be performed on the plank. They are defined as *pick_up_plank(x, y, z)* where x is the plank instance and y and z are it's supports and *place_plank(x, y, z)* where x is the plank instance and y and z are the destination supports. For both of these actions, y and z can both be the table.

   The actions *pick_up(x, y)* and *place(x, y)* need slight modifications to restrict x from being a plank and we will define the preconditions, add list, and delete list for the new actions.

PC	**D**	**A**
clear(x)	*on(x, y)*	*clear(y)*
empty(hand)	*empty(hand)*	*in_hand(x)*
on(x, y)	*clear(x)*	
$\neg$*plank(x)*		

Table 3: Update preconditions, add list, and delete list for *pick_up(x, y)*.

PC	**D**	**A**
in_hand(x)	*in_hand(x)*	*empty(hand)*
clear(y)	*clear(y)*	*clear(x)*
flat(y)		*clear(table)*
$\neg$*plank(x)*		*on(x, y)*

Table 4: Updated preconditions, add list, and delete list for *place(x, y)*.

PC	D	A
clear(x)	*plank_on(x, y, z)*	*clear(y)*
empty(hand)	*empty(hand)*	*clear(z)*
plank_on(x, y, z)	*clear(x)*	*in_hand(x)*
plank(x)		

Table 5: Preconditions, add list, and delete list for *pick_up_plank(x, y, z)*.

PC	D	A
in_hand(x)	*clear(y)*	*empty(hand)*
plank(x)	*clear(z)*	*clear(x)*
clear(y)	*in_hand(x)*	*clear(table)*
clear(z)		*plank_on(y, z)*
flat(y)		
flat(z)		

Table 6: Preconditions, add list, and delete list for *place_plank(x, y, z)*.

Finally we will add three additional goal states to our goal state from part 2. The additional goals are: two towers with a plank on their bottom blocks, a plank on their middle blocks, or both. Let B_1, B_2, B_3, B_4, B_5, and B_6 be blocks and P_1 and P_2 be planks.

Two towers of height three with a plank on their bottom blocks:

$$on(B_1, Table) \ \wedge \ on(B_2, P_1) \ \wedge \ on(B_3, B_2) \ \wedge \ \neg flat(B_3)$$
$$\wedge \ on(B_4, Table) \ \wedge \ on(P_2, B_4) \ \wedge \ on(B_6, B_5) \ \wedge \ \neg flat(B_6)$$
$$\wedge \ plank\_on(P_1, B_1, B_4)$$

Two towers of height three with a plank on their middle blocks:

$$on(B_1, Table) \ \wedge \ on(B_2, B_1) \ \wedge \ on(B_3, P_1) \ \wedge \ \neg flat(B_3)$$
$$\wedge \ on(B_4, Table) \ \wedge \ on(B_5, B_4) \ \wedge \ on(B_6, P_1) \ \wedge \ \neg flat(B_6)$$
$$\wedge \ plank\_on(P_1, B_2, B_5)$$

Two towers of height three with a plank on both their bottom and middle blocks:

$$on(B_1, Table) \ \wedge \ on(B_2, P_1) \ \wedge \ on(B_3, P_2) \ \wedge \ \neg flat(B_3)$$
$$\wedge \ on(B_4, Table) \ \wedge \ on(B_5, B_4) \ \wedge \ on(B_6, P_2) \ \wedge \ \neg flat(B_6)$$
$$\wedge \ plank\_on(P_1, B_1, B_4) \ \wedge \ plank\_on(P_2, B_2, B_5)$$

2 Crossword Puzzles

1. **Formulate this problem as a CSP where the variables are words. List all the variables and constraints.**

2. **Formulate this problem as a CSP where the variables are letters. List all the variables and constraints.**

3. **Suppose we wish to make sure the the crossword puzzle we generate doesn't contain multiple words that are "too similar." We'll say that words w_1 and w_2 are too similar if *either* one can be obtained from the other by changing exactly one character *or* one is a substring of the other. For instance "dog" and "dogs" are too similar (by the second constraint) as are "dog" and "dog" (again, by the second constraint). Similarly, "cats" and "cots" are too similar (by the first constraint). How would you specify these additional constraints in both the by-word and by-letter formulations?**