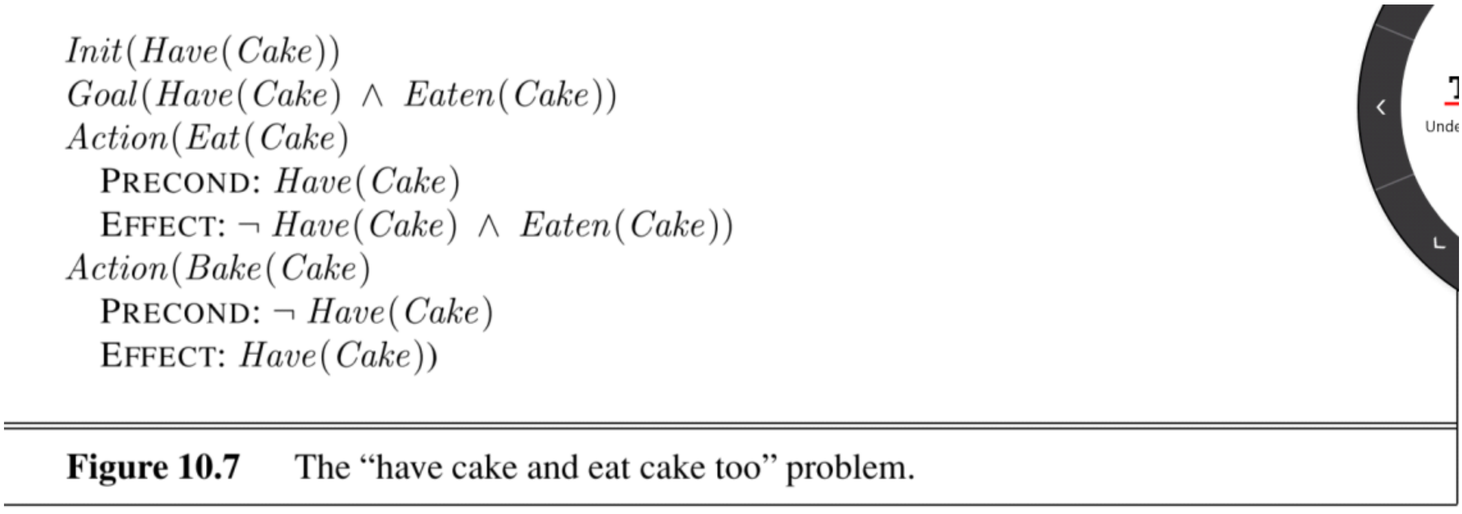
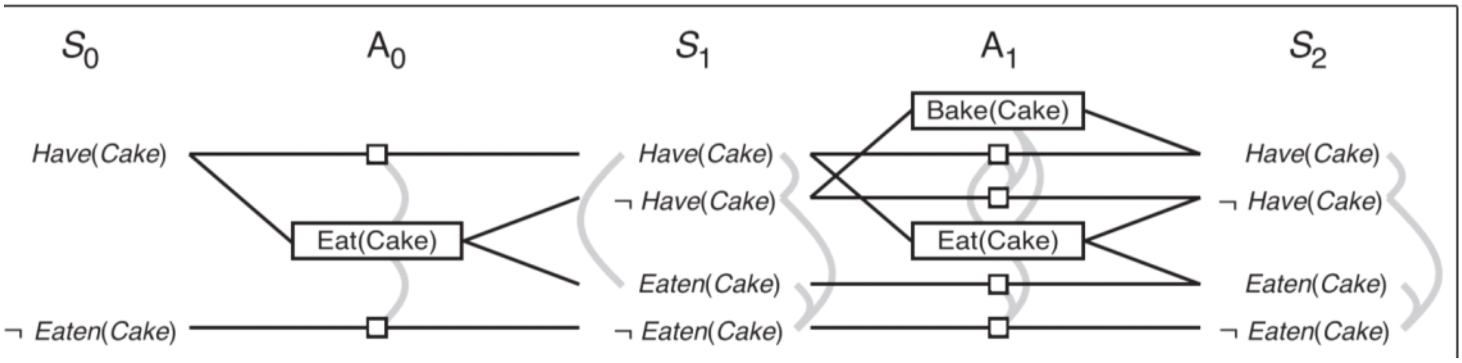
Thursday, December 21, 2017 1:12 AM



$Init(Have(Cake))$
$Goal(Have(Cake) \wedge Eaten(Cake))$
$Action(Eat(Cake)$
  $\textsc{Precond}: Have(Cake)$
  $\textsc{Effect}: \neg\, Have(Cake) \wedge Eaten(Cake))$
$Action(Bake(Cake)$
  $\textsc{Precond}: \neg\, Have(Cake)$
  $\textsc{Effect}: Have(Cake))$

Figure 10.7 The "have cake and eat cake too" problem.



How to build action level.

In A_0, loop through self.all_actions (this all_action includes no op), if action's precondition is a subset of the S_0, then add the action to A_0, even if the preconds are mutex. We should use PgNode_a.prenodes to do the job, it's a set, so it has issubset function, we just need to pass S_0 as args. If true, we add this action to A_0.

When we add a new action, we should connect them, the PgNode has two 3 attributes: parents, children and mutex (all are sets). We loop through all nodes in S_0, add action to its children set, and add nodes in S_0 to action's parent set.

How to build state level

In S_1, every node is the effect node of action in A_0, so add every effect node to S_1, but we should use sets, since, unlike all_actions, there are no repeat action, there might be repeat effect node. Then we should connect them.

How to build mutex for action

Inconsistent effect

For any two actions, if their +effect and -effect cancel each other out, then we say they are mutex.

Interference

For any two actions, if one's +effect is another's -precond or other way around, then they are mutex. I think this one is for persistent action since its precondition and effect are the same, so we go back to inconsistent effect.

PS, I do not sure when two actions are not persistent action, if interference still holds. But I guess it's still holds, because I think precondition and effect is closely relate to each other, at least in PDDL world.

Competing needs

For any two actions, if their preconditions are mutex, they are mutex. We could utilize PgNode.is_mutex and PgNode.parent, since precondition is one of the nodes in S_i. In this case, if we found one pairs of node_s that are mutex, we return True

How to build mutex for states.

Inconsistent support

Two literals are mutex, if no two actions (they are either mutex or negate each other) can achieve them.
To get this done, again, we use PgNode.mutex and PgNode.parent to check if two actions are mutex. But in this case, we have to scan all pairs of node_a since we have to make sure no two action can achieve them. And if one pairs of action can achieve the literal, we return False.

Negation

Literals that are negate each other, e.g. have(cake) and ~have(cake)

Level sum heuristic

We are given a goals, we then search through self.s_levels, if goal 1 is found at level 2, goal 2 is found at level 3, then level_sum will return 2+3=5. We should use sefl.problem.goal (goal is list of expr object, and self.s_levels[level] consists of node objects, to compare, use expr == node.symbol (symbole is Expr object)