1 Vocabulary Check

Define each of the following terms:

1. Linear Planning Algorithm

   This algorithm greedily picks a subgoal and finds a plan for it. It does not move onto the next subgoal until it has found an plan for the current one. This algorithm is not necessarily complete or optimal, it usually works well with goals whose subgoals require mostly disjoint actions

2. Non-Linear Planning

   The agent interleaves actions to solve multiple subgoals at a time. This will be complete and optimal.

3. Interference

   One action's effect deletes or negates a precondition of the other.

4. Inconsistent effects/Inconsistency

   One action's effect deletes or negates an effect of the other.

5. Competing Needs

   One action's precondition is the negation of a precondition of the other.

6. Sussman's Anomaly

   A reason for a linear planning algorithm to loop infinitely, this occurs when completing one subgoal undoes another subgoal, so the goal is never reached

2 Compare and Contrast

1. What is the difference between linear and non-linear planning? When are they the same?

   Linear planning: We keep a stack of unnachieved goals and solve each one, one at a time, adding back goals that we violate along the way.
   Non-Linear planning: Maintain a set of unnachieved goals and search all interleavings of these goals adding a goal back to the set if a later change makes it violated.

   Linear planning and non-linear planning are equivalent when there is one goal because there is only one possible "interleaving" of the goals so linear and non-linear planning will have the same approach.

2. What are some ways to find a plan using a classical planning environment model?

   Naive search (BFS), linear planning/non-linear planning, graph plan.

3. What classical planning assumptions are relaxed when using the GraphPlan heuristic? Why is this helpful compared to naive search?

   We are assuming we can take multiple non-mutex actions at the same time.

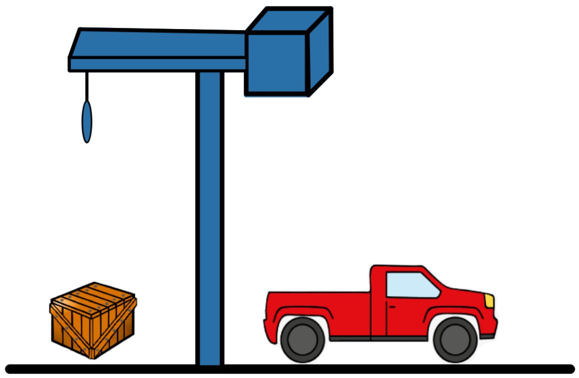
   This is useful since in this environment, taking multiple steps at a time will allow us to add multiple goals, finishing the search problem much quicker than the tradidional one action method

   (Also, if we return a plan that requires we take multiple actions at the same time, we can take them in any order with the same effect since they are non-mutex)

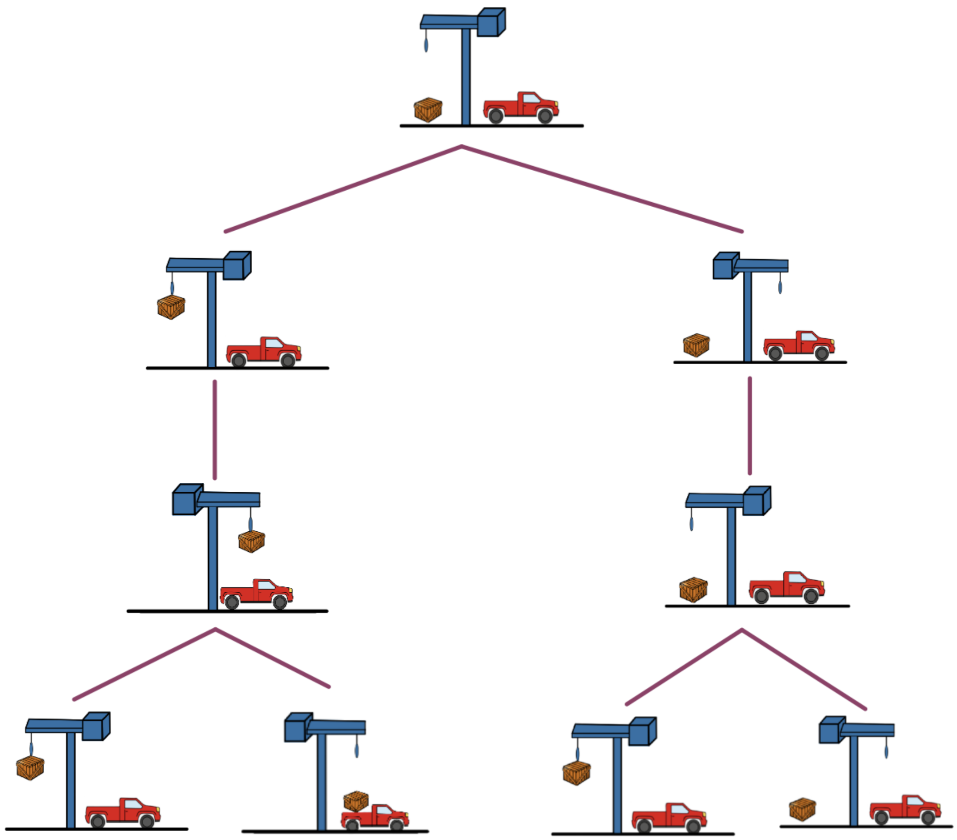
3 Symbolic Planning - Crate Problem

In the Crane problem, you are given a crane, a package and a truck. The package starts on the left, the truck on the right, and the crane faces the left. The goal of this is to load the package onto the truck and have the crane be facing the left.

The crane can swing between left and right, with or without a payload, and it can pick up the crate if it is on the same side. The crate can only be loaded onto the truck using the crane.



(a) Draw the planning graph for the first 3 moves. You may use pictures instead of propositions.

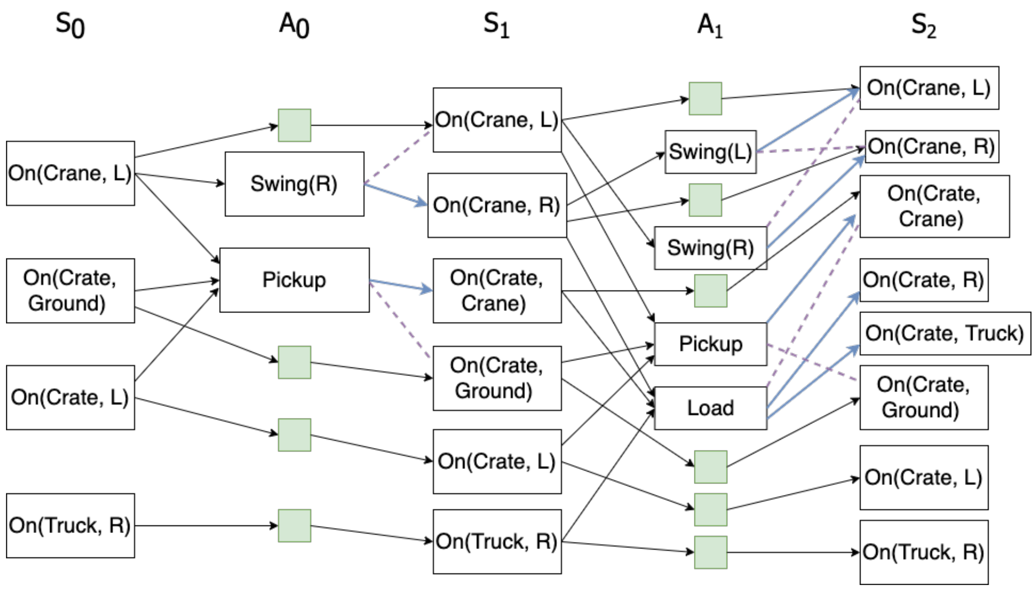


(b) Formulate the crate problem as a symbolic plan. You will need to define your variables, instances, start/goal states, and operators.

See provided sample code

(c) Draw the first two levels of the Graph Plan graph.

In the following diagram, the blue lines represent the propositions added as the result of an action and the dotted purple lines represent the propositions deleted at the result of that action. The green squares in the action levels represent no-op's.



(d) Identify the exclusive actions in your graph and determine which type of mutex each is.

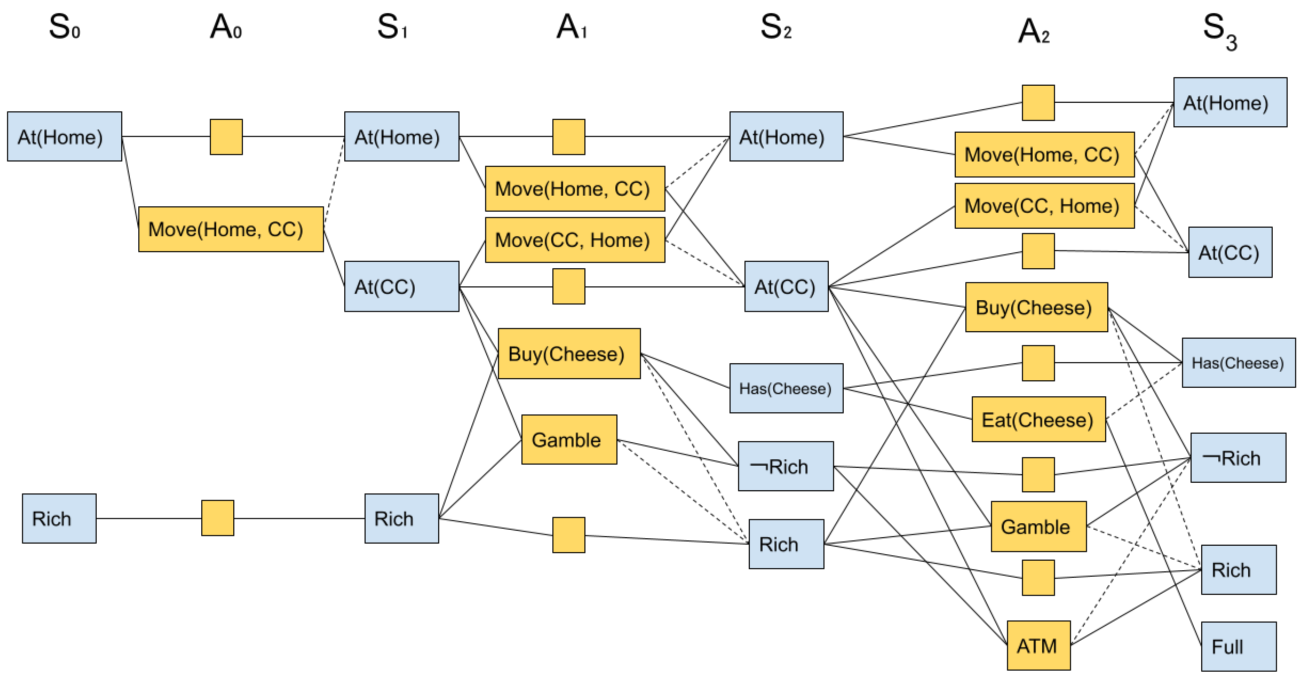
In the level A_0, Swing(R) and Pickup interfere with each other. In level A_1, one example would be Swing(L) and Swing(R) being inconsistent.

4 Mutex relation? I don't even know her!

Pinky is getting food from a Chuck E. Cheese. Pinky has the following actions:

- Move(A,B):
  - Preconditions: At(A)
  - Add list: At(B)
  - Delete list: At(A)
- Buy(Cheese):
  - Preconditions: At(ChuckyCheese), Rich
  - Add list: Has(Cheese), ¬ Rich
- Gamble
  - Preconditions: At(ChuckyCheese),Rich
  - Add list: ¬ Rich
  - Delete list: Rich
- ATM
  - Precondition: At(ChuckyCheese),¬ Rich
  - Add list: Rich
  - Delete list: ¬ Rich
- Eat(Cheese):
  - Preconditions: Has(Cheese)
  - Add list: Full
  - Delete list: Has(Cheese)

The start state contains the predicates Rich and At(Home).
The goal state is any state containing Full.
Below is the corresponding GraphPlan graph:



(a) Based on the above graph, list two actions that are mutex via inconsistent effects in level A_0.

No-op of At(Home) and Move(Home, ChuckyCheese)

(b) Based on the above graph, list two actions that are mutex via Interference in level A_1

Buy(Cheese) and Gamble()

(c) Based on the above graph, list two actions that are mutex via Competing needs in level A_2.

No-op of Rich and ATM