Fundamentals of Artificial Intelligence

Partial Order Planning

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

Goal Stack Planning

- One of the earliest planners STRIPS considered actions by reasoning in a *backward* manner; but committed to actions only in a *forward* manner.
  - Recall the working through for the Blocks World Problem in STRIPS in the previous example.
- The planning algorithm discussed is referred to a Goal Stack Planning.
  - Pushes subgoals and actions into a stack; picks an action only when all its preconditions are satisfied.
  - It works with state descriptions that are always consistent for growing plans, and goal descriptions for growing the search tree.

Goal Stack Planning

- Goal Stack Planning is incomplete, in the sense that it could terminate without finding a plan.
  - GSP could commit to a wrong action at some time, reaching a state from which a plan cannot be found.
- Goal stack Planning breaks up a set of goal predicates into individual subgoals and attempts to solve them individually one after another.
- This approach is called *linear planning*.
  - This refers to the fact that the subgoals are attempted and solved in a linear order.
  - This may not be always possible!

Plan Space Planning

- Plan space planning approaches work with plan structures, and have the ability to modify or extend any part of the plan.
- Plan space planning is also referred to as nonlinear planning.
  - They can modify any part of a plan; the plan space planning approaches are not constrained to focus on any one subgoal continuously.
  - They can shift attention *midway*; and in the process often solve problems, like the Sussman anomaly correctly.

Plan Space Planning

- The planning approach described so far, reason with states.
  - The planner is looking at a state and a goal; if the state satisfies the goal, it terminates.
  - Else, it makes a search move over the state space looking for actions to add to the plan.
- Plan space planning is an alternative; wherein the idea is to consider the space of all possible plans; search in this space for a plan.
- Algorithms in this category represent a plan as actions arranged in a partial order, and hence also referred to as partial order planning.

Least Commitment

- Many planners use the principle of least commitment, which says that one should only make choices about things that you currently care about, leaving the other choices to be worked out later.
  - This is a good idea for programs that search; because making a choice about something you don't care about now, you are likely to make the wrong choice and have to backtrack later.
- A least commitment planner could leave the ordering of the two steps unspecified.
  - When a third step, RightSock, is added to the plan, we want to make sure that putting on the right sock comes before putting on the right shoe. But we do not care where they come with respect to the left shoe.

Linearization

- A planner that can represent plans in which
  some steps are ordered (before or after) with respect to each other and
  other steps are unordered
  is called a partial order planner.
- The alternative is a total order planner, in which plans consist of a simple list of steps.
- A totally ordered plan that is derived from a plan P by adding ordering constraints is called a linearization of P.

Fully Instantiated Plan

- Planners have to commit to bindings for variables in operators.
- For example, suppose one of your goals is Have(Milk), and you have the action Buy(item, store).
  - A sensible commitment is to choose this action with the variable item bound to Milk. No good reason to pick a binding for store; principle of least commitment says to leave it unbound and make the choice later.
  - By delaying the commitment to a particular store, we allow the planner to make a good choice later; strategy can also help prune out bad plans.
  - Suppose for some reason the branch that includes the partially instantiated action Buy(Milk, store) leads to a failure. If we had committed to a particular store, then the search algorithm would force us to backtrack and consider another store.
- Plans in which every variable is bound to a constant are called fully instantiated plans.

Components of a Plan

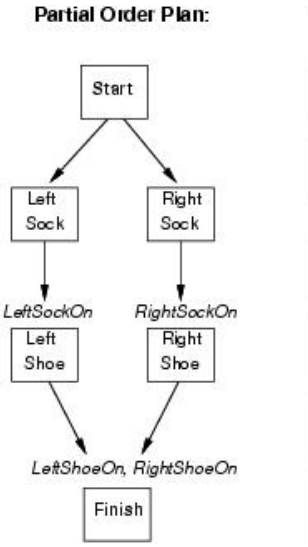
1. A set of actions; Each action is one of the operators for the problem.
2. A set of ordering constraints
   - A $\prec$ B reads "A before B" but not necessarily immediately before B
   - Alert: caution to cycles A $\prec$ B and B $\prec$ A
3. A set of causal links (protection intervals) between actions
   - A $\xrightarrow{p}$ B reads "A achieves *p* for B" and p must remain true from the time A is applied to the time B is applied
   - Example RightSock $\xrightarrow{RightSockOn}$ RightShoe
4. A set of open preconditions
   - Planners work to reduce the set of open preconditions to the empty set without introducing contradictions

Initial Plan

- The initial plan, before any refinements have taken place, simply describes the unsolved problem; consists of two steps, called Start and Finish.
  - Start $\prec$ Finish
  - Both Start and Finish have null actions associated with them, so when it is time to execute the plan, they are ignored.
  - Start step has no preconditions, and its effect is to add all the propositions that are true in the initial state.
  - Finish has the goal state as its precondition, and no effects.
- By defining a problem this way, the planners can start with the initial plan and manipulate it until they come up with a plan that is a solution.

Partial Ordered Plan

Partial Order Plan



Solution

- A solution is a plan that an agent can execute, and that guarantees achievement of the goal.
  - If we wanted to make it really easy to check that a plan is a solution, we could insist that only fully instantiated, totally ordered plans can be solutions.
  - But this is unsatisfactory for three reasons.
    - First, it is more natural for the planner to return a partial-order plan than to arbitrarily choose one of the many linearizations of it.
    - Second, some agents are capable of performing actions in parallel, so it makes sense to allow solutions with parallel actions.
    - Lastly, when creating plans that may later be combined with other plans to solve larger problems, it pays to retain the flexibility afforded by the partial ordering of actions.
- We allow partially ordered plans as solutions using a simple definition: a solution is a complete, consistent plan.

Solution

Complete Plan

- A complete plan is one in which every precondition of every step is achieved by some other step.

Consistent Plan

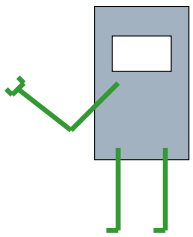
- A plan is consistent iff there are no cycles in the ordering constraints and no conflicts with the causal links.

- A consistent plan with no open preconditions is a solution.

An Illustrative Example

Consider the robot needs to solve the following

Get a quart of milk; a dark chocolate and a good book.



Start: At(Home), Sells(Store, Milk), Sells(Store, Chocolate), Sells(BStore, Book)

Goal: Have(Milk), Have(Chocolate), Have(Book), At(Home)

An Illustrative Example

Start

At(Home), Sells(Store, Milk), Sells(Store, Chocolate), Sells(BStore, Book)

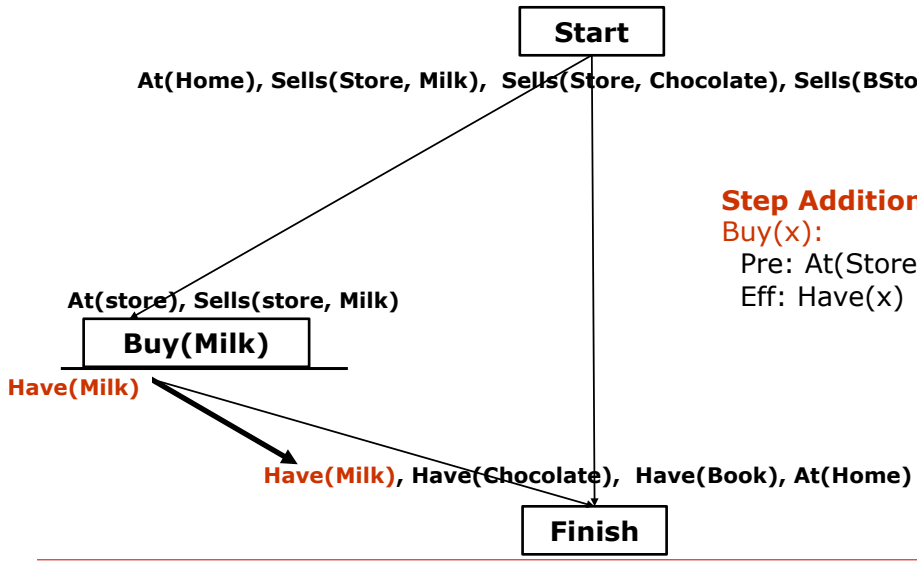
The planner starts with an initial plan representing the start and finish steps, and on each iteration adds one more step. If this leads to an inconsistent plan, it backtracks and tries another branch of the search space.

To keep the search focused, the planner considers adding steps that serve to achieve a precondition that has not yet been achieved. The causal links are used to keep track of this.

Have(Milk), Have(Chocolate), Have(Book), At(Home)

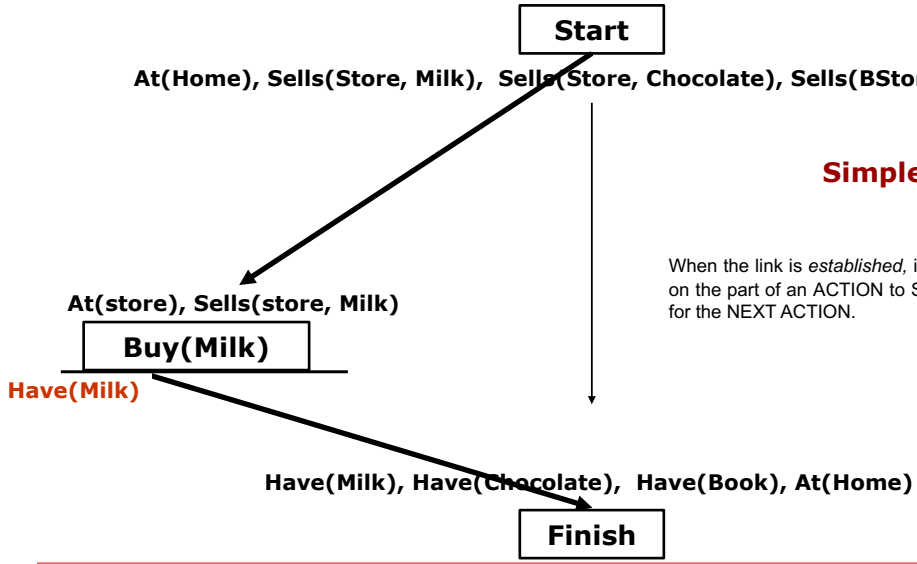
Finish

An Illustrative Example



Step Addition
Buy(x):
Pre: At(Store), Sells(Store,x)
Eff: Have(x)

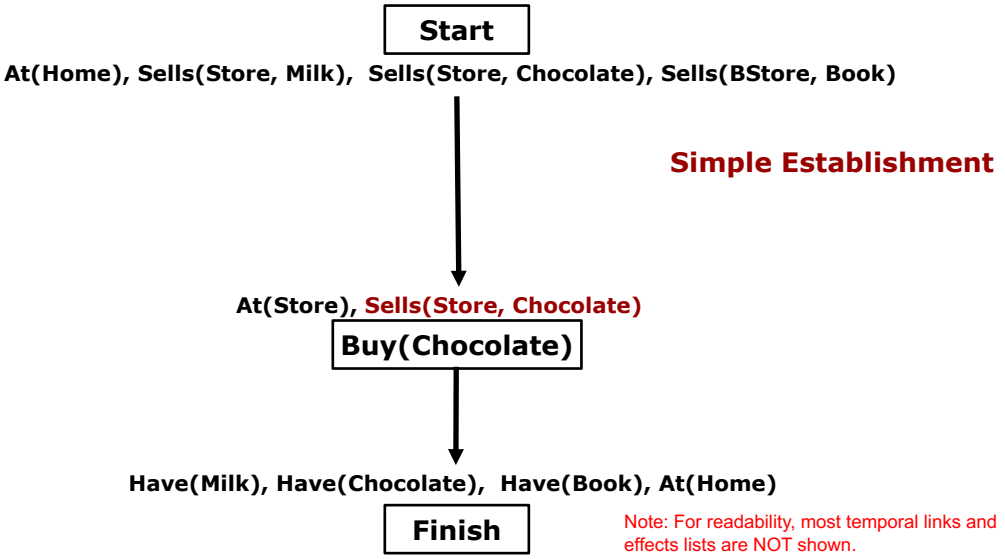
An Illustrative Example



Simple Establishment

When the link is *established*, it represents a commitment on the part of an ACTION to SUPPORT the precondition for the NEXT ACTION.

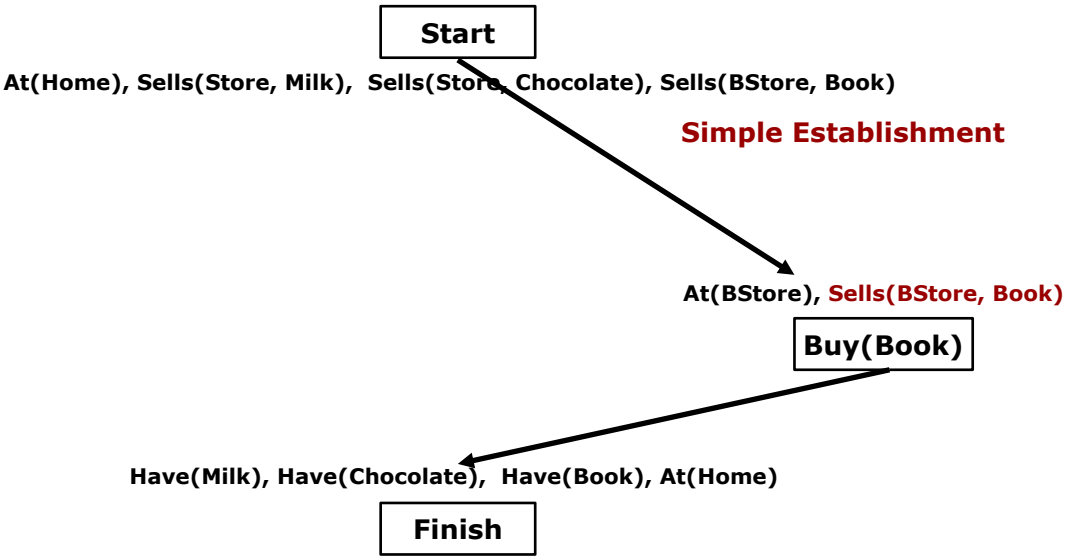
An Illustrative Example



Note: For readability, most temporal links and effects lists are NOT shown.

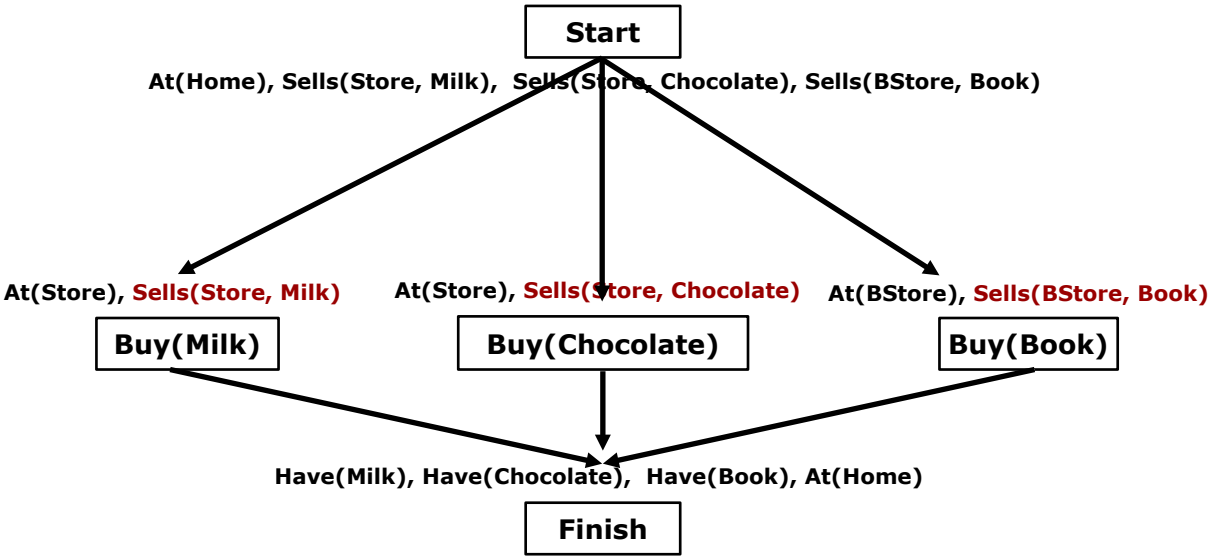
19

An Illustrative Example



20

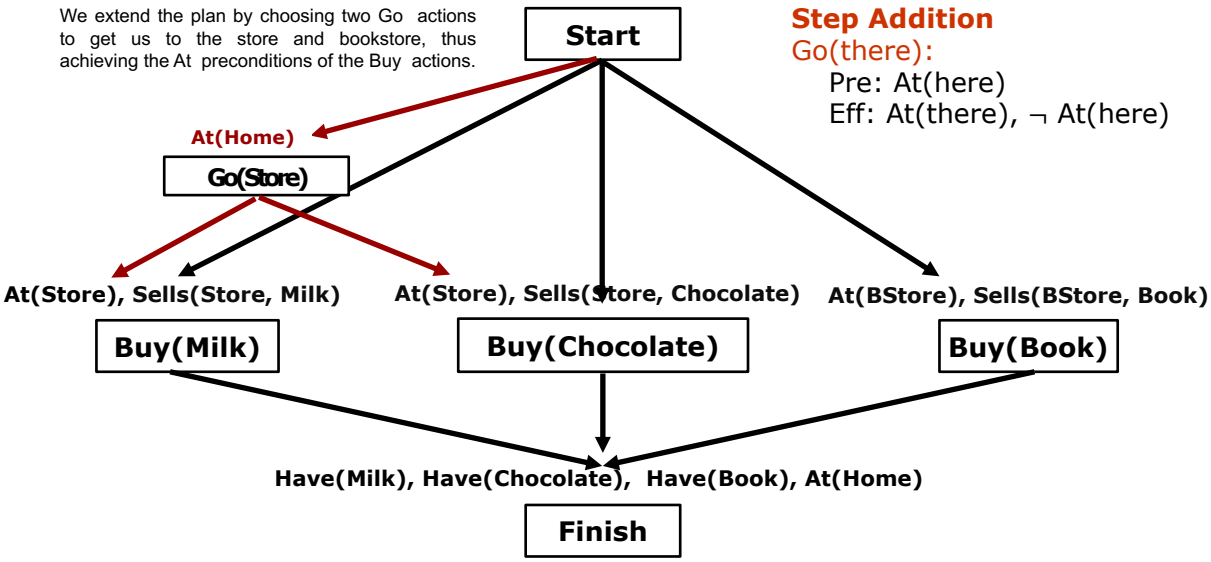
An Illustrative Example



An Illustrative Example

We extend the plan by choosing two Go actions to get us to the store and bookstore, thus achieving the At preconditions of the Buy actions.

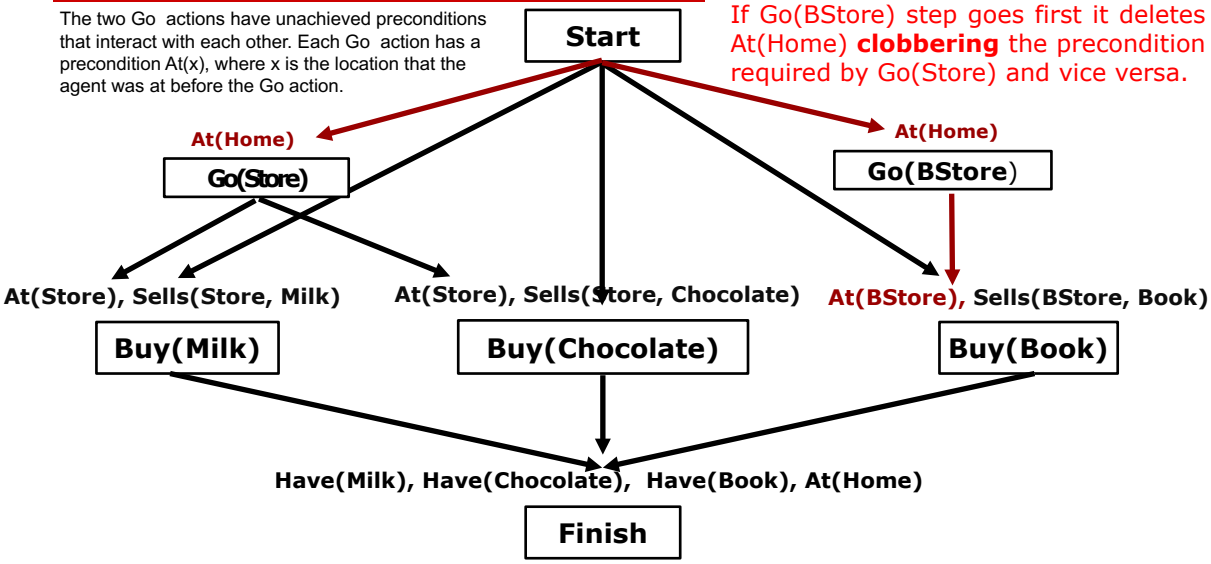
Step Addition
Go(there):
Pre: At(here)
Eff: At(there), ¬ At(here)



An Illustrative Example

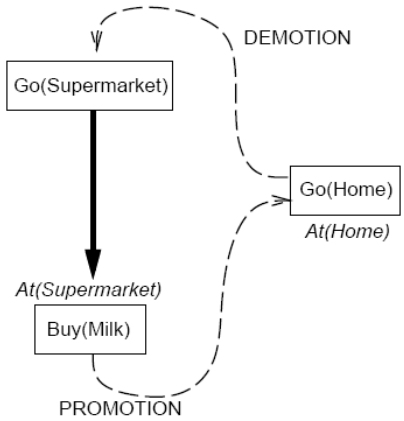
The two Go actions have unachieved preconditions that interact with each other. Each Go action has a precondition At(x), where x is the location that the agent was at before the Go action.

If Go(BStore) step goes first it deletes At(Home) **clobbering** the precondition required by Go(Store) and vice versa.



Clobbering

- A **clobberer** is a potentially intervening step that destroys the condition achieved by a causal link
  - Example
    Go(Home) clobbers At(Supermarket)
- Demotion
  - Put before Go(Supermarket)
- Promotion
  - Put after Buy(Milk)

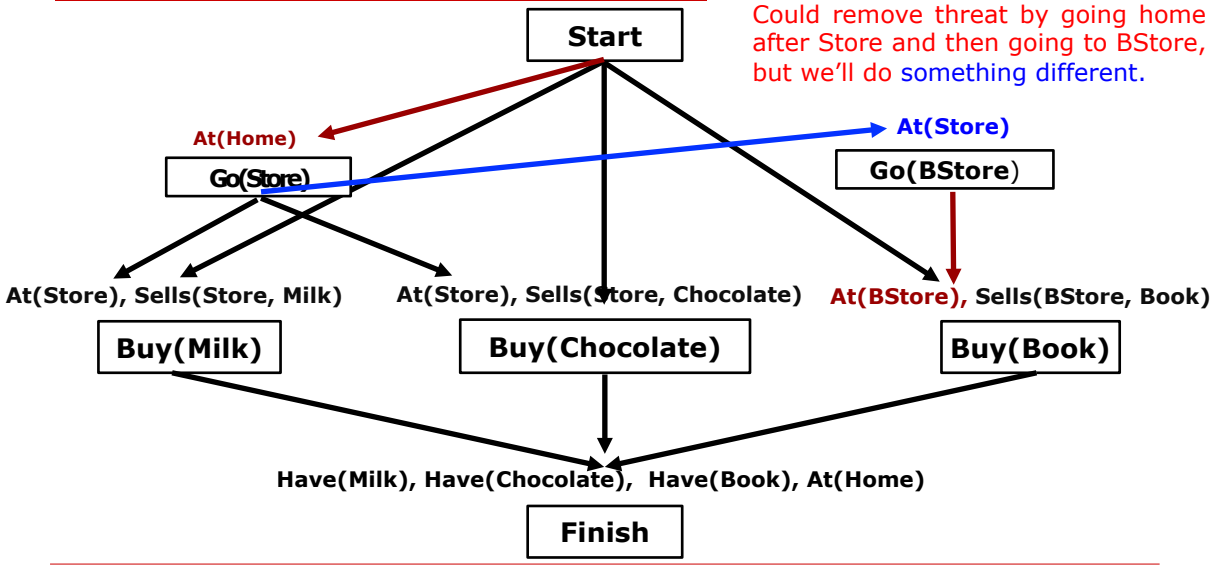


Declobbering – Threat Removal

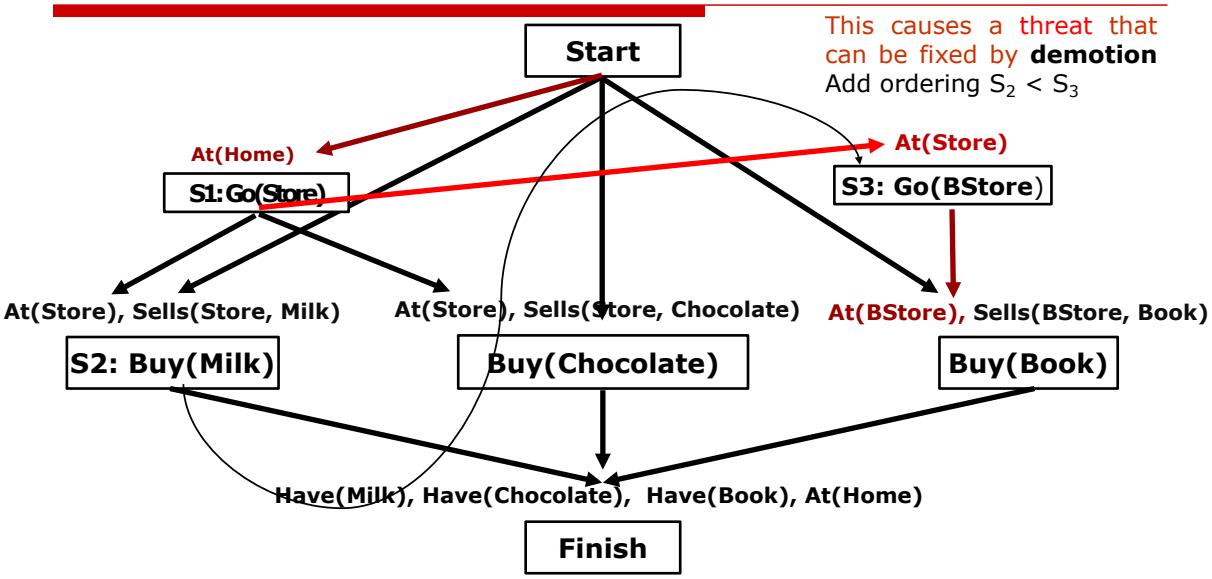
- Threat: a step that deletes (clobbers) a needed effect
  - S_2 requires an effect of S_1
    (i.e. there is a causal link between S_1 and S_2),
    but the effect of S_3 is to undo the effect S_2 requires

- Thus, S_3 can't occur between S_1 and S_2
  - it must occur either before S_1 (**promotion**)
    add link $S_3 < S_1$
  - or after S_2 (**demotion**)
    add link $S_2 < S_3$

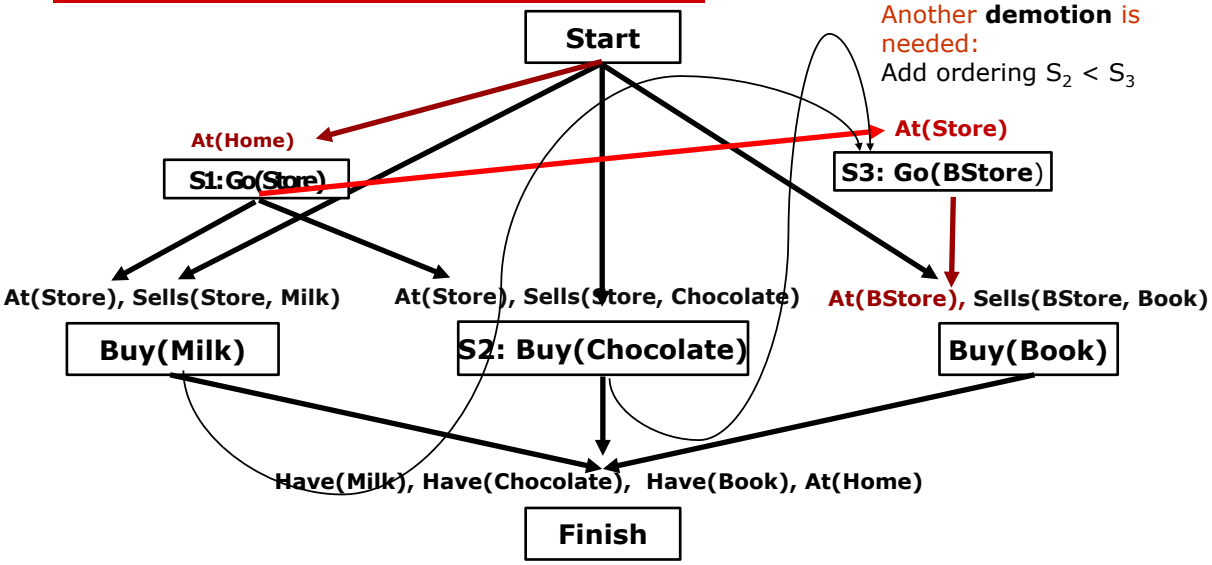
An Illustrative Example



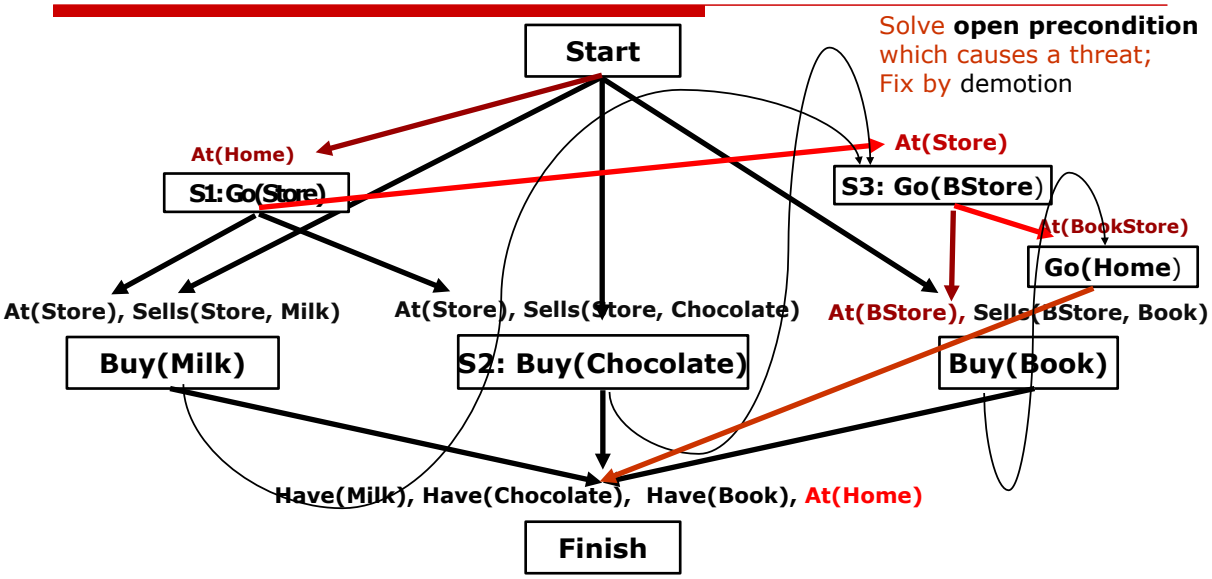
An Illustrative Example



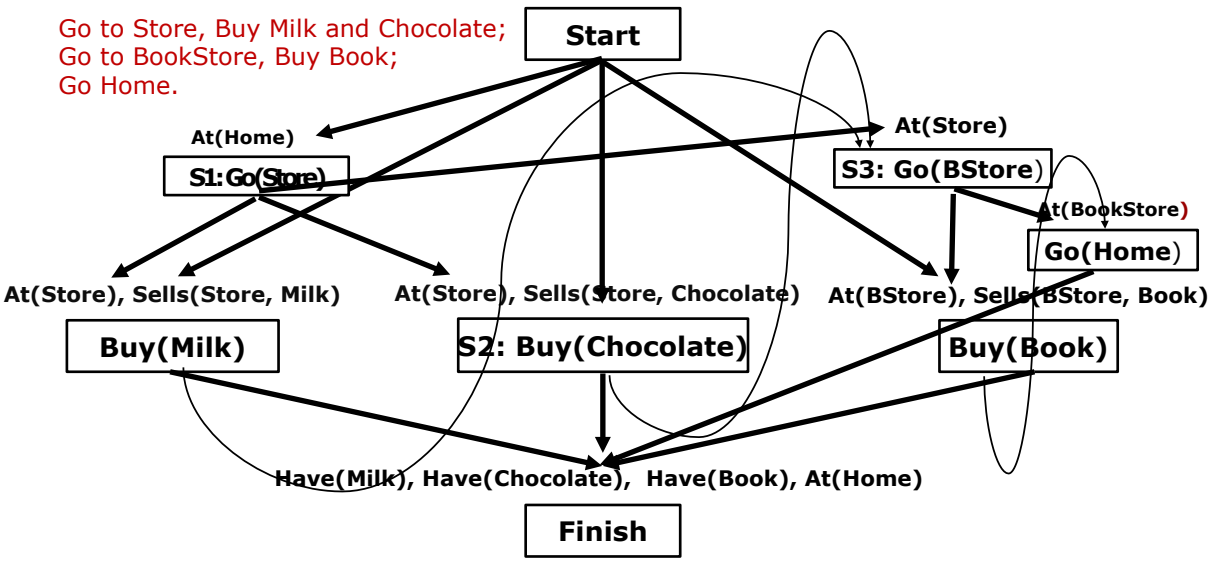
An Illustrative Example



An Illustrative Example

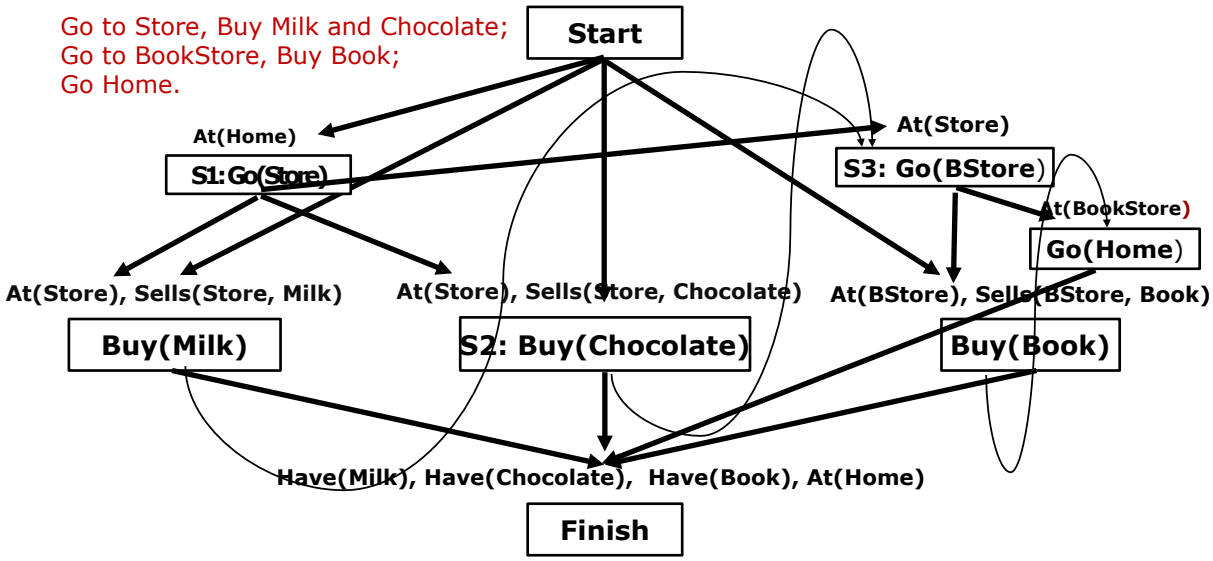


An Illustrative Example

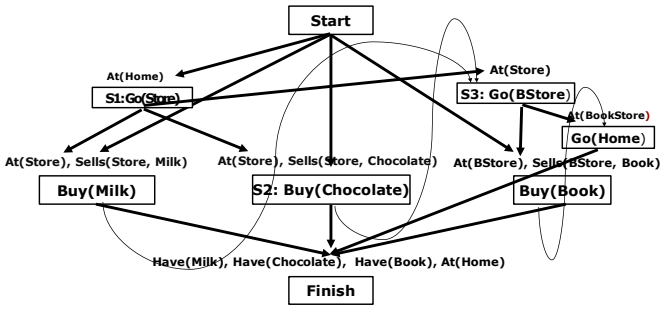


An Illustrative Example

Go to Store, Buy Milk and Chocolate;
Go to BookStore, Buy Book;
Go Home.



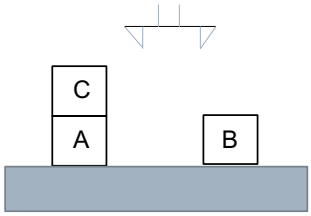
Final POP Plan



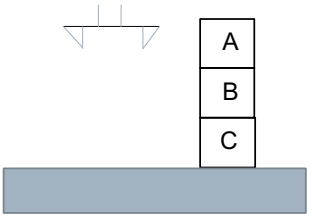
1. It can take a problem that would require many thousands of search states for a problem-solving approach, and solve it with only a few search states.
2. Moreover, the least commitment nature of the planner means it only needs to search at all in places where sub-plans interact with each other.
3. The causal links allow the planner to recognize when to abandon a doomed plan without wasting a lot of time expanding irrelevant parts of the plan.

Sussman Anomaly

on(C,A)
clear(C)
clear(B)
onTable(A)
onTable(B)
handEmpty



Initial State



Final State

on(A,B)
on(B,C)
clear(A)
onTable(C)
handEmpty

Noninterleaved planners typically separate the goal into sub-goals:

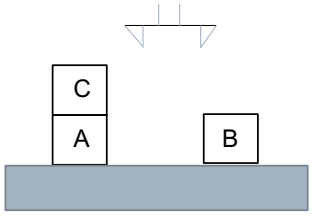
a. on(A,B)
b. on(B,C)

This is the Sussman anomaly; which illustrates a weakness of noninterleaved planning algorithms.

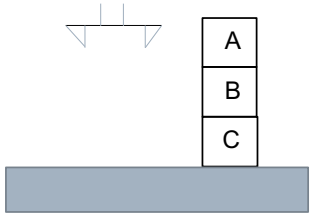
1. Suppose the planner starts by pursuing Goal 1. The basic step is to move C out of the way, then move A atop B. But while this sequence accomplishes Goal 1, the agent would be left with no option but to undo Goal 1 in order to pursue Goal 2.
2. If instead the planner starts with Goal 2, the most efficient solution is to move B. But again, the planner cannot pursue Goal 1 without undoing Goal 2:

Sussman Anomaly

on(C,A)
clear(C)
clear(B)
onTable(A)
onTable(B)
handEmpty



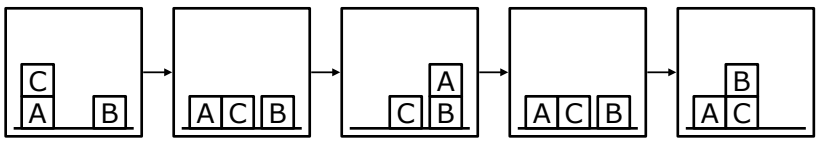
Initial State



Final State

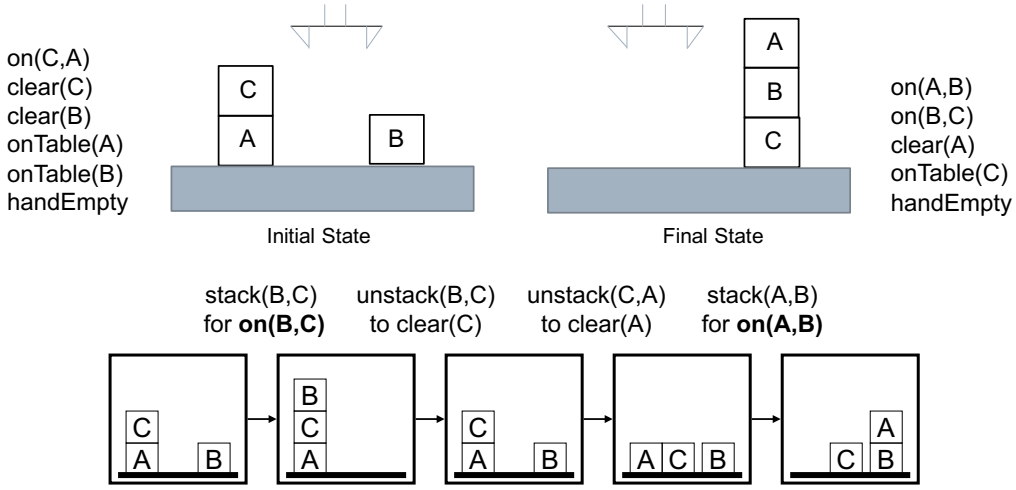
on(A,B)
on(B,C)
clear(A)
onTable(C)
handEmpty

unstack(C,A) to clear(A) | stack(A,B) for **on(A,B)** | unstack(A,B) to clear(B) | stack(B,C) for **on(B,C)**



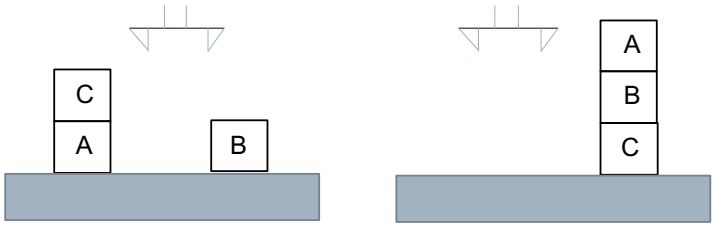
on(A,B) is undone to clear(B) when solving **on(B,C)**

Sussman Anomaly



on(B,C) is undone to clear(C) when solving **on(A,B)**

Sussman Anomaly



There is NO ORDERING of subgoals where when each of the subgoals solved individually, the goal is solved as a consequence.

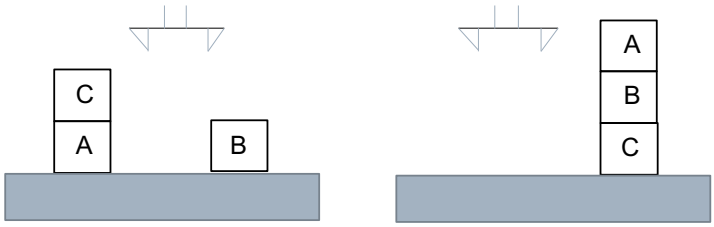
1. Begin work on ON(A,B) by clearing A
   i.e., putting C on table.
2. Achieve ON(B,C) by stacking B on C
3. Achieve ON(A,B) by stacking A on B.

We couldn't do this using a stack within STRIPS; but interleaving!

Interleaving vs. Non-interleaving Planner

- Non-interleaving planner
  - $G_1 \wedge G_2$:
    either all the steps for achieving G_1 occur before G_2,
    or all of the steps for achieving G_1 occur after G_2
  - all of the steps for a sub/goal must occur "atomically"
  - e.g. STRIPS
  - can't solve the Sussman Anomaly
- Interleaving planner
  - can intermix order of sub/goal steps
  - can solve the Sussman Anomaly by interleaving steps:
    unstack(C,A), Pickup(B), Stack(B,C), Stack(A,B)

Sussman Anomaly



Initial State | Final State

clear(x) on(x,z) clear(y)

stack(x,y)

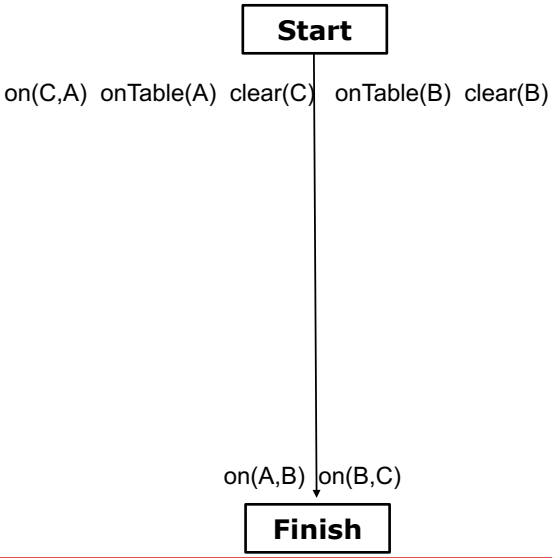
¬ on(x,z) ¬ clear(y)
clear(z) on(x,y)

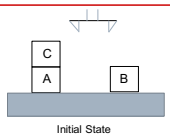
clear(x) on(x,z)

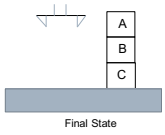
putOnTable(x)

¬ on(x,z) clear(z)
on(x,Table)

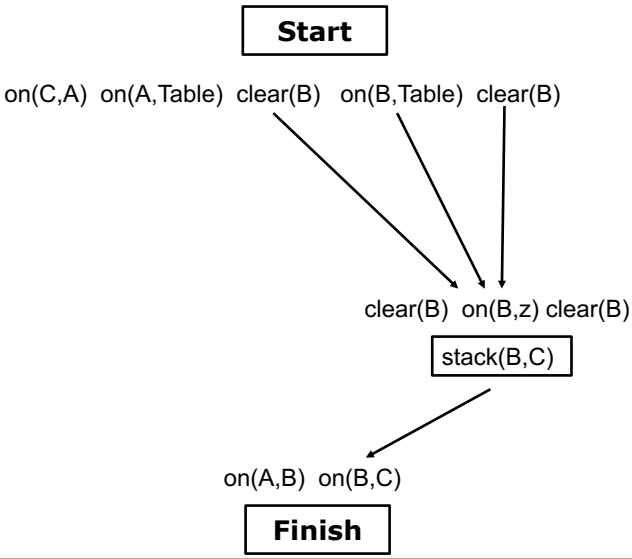
Sussman Anomaly

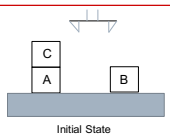


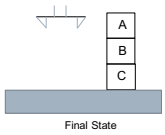




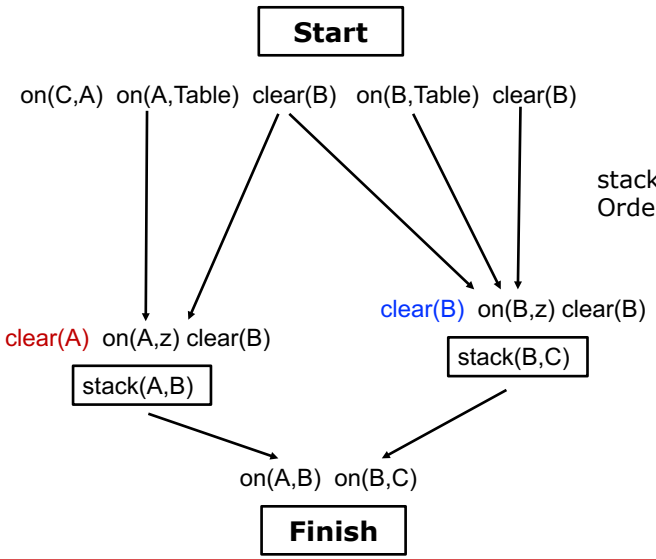
Sussman Anomaly

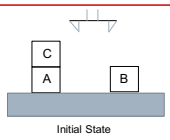




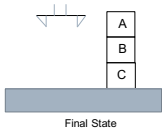


Sussman Anomaly

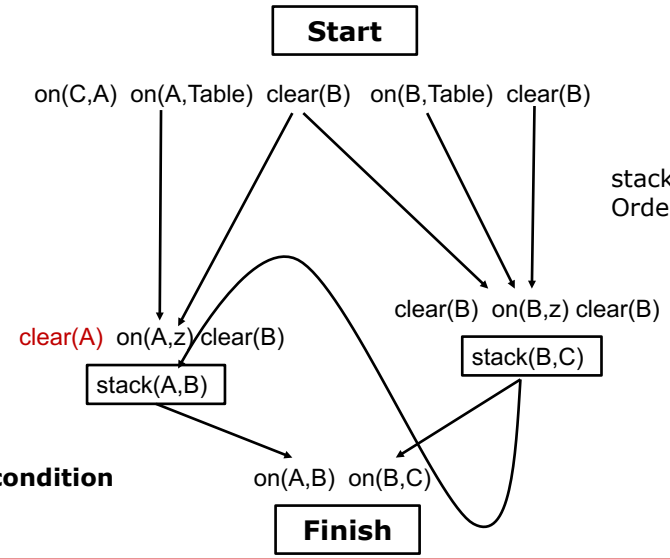


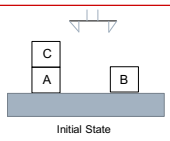


stack(A,B) clobbers clear(B)
Order: After stack(B,C)



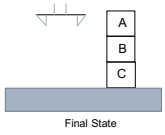
Sussman Anomaly



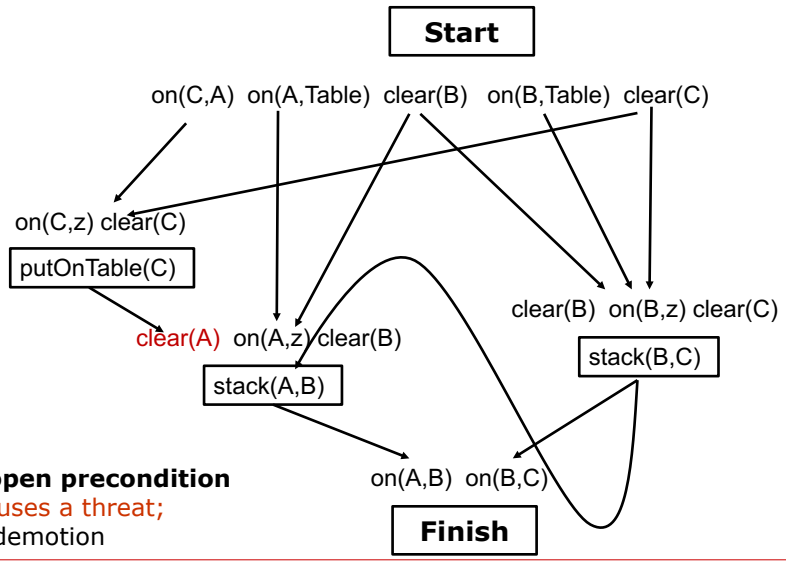


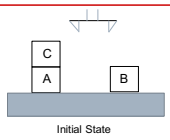
stack(A,B) clobbers clear(B)
Order: After stack(B,C)

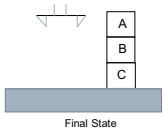
Solve **open precondition**



Sussman Anomaly

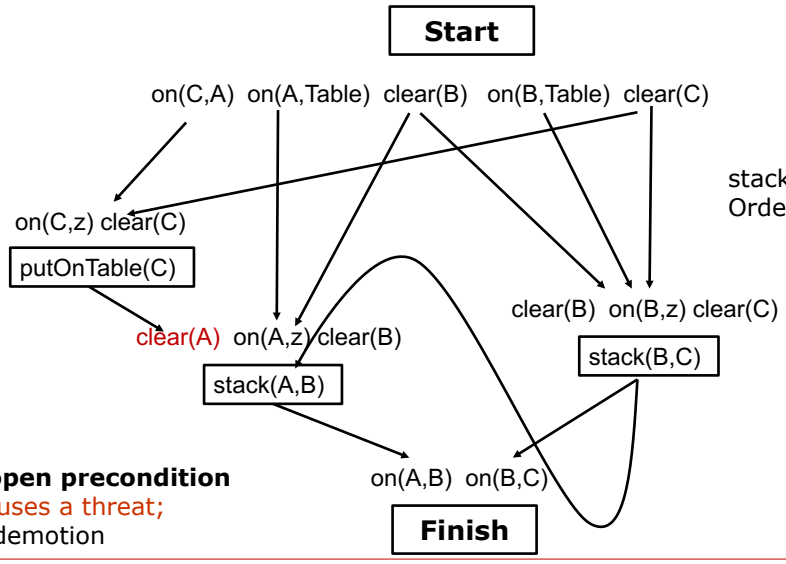


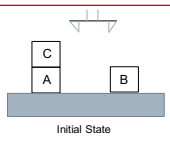




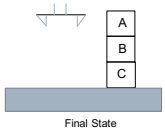
Solve **open precondition**
This causes a threat;
Fix by demotion

Sussman Anomaly



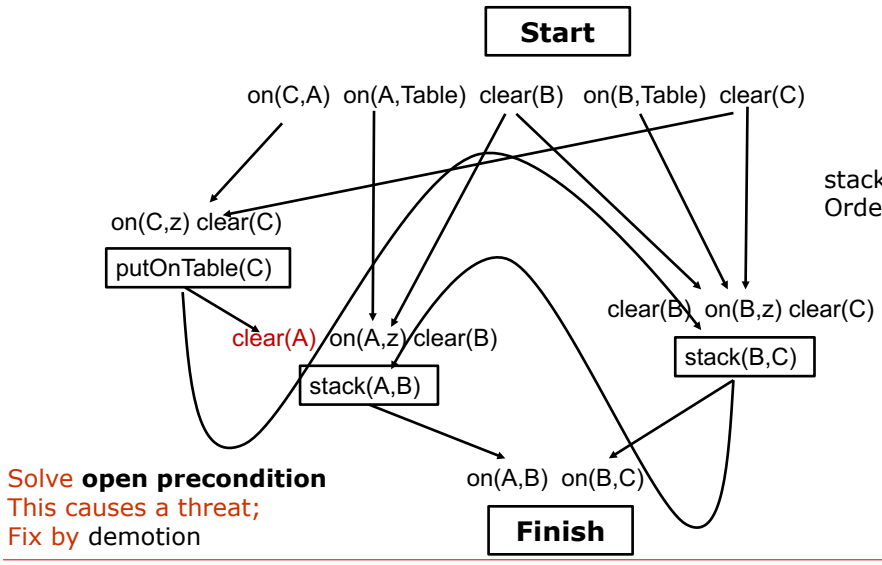


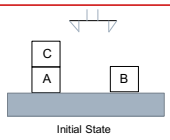
stack(B,C) clobbers clear(C)
Order: After putOnTable(C)



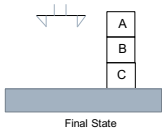
Solve **open precondition**
This causes a threat;
Fix by demotion

Sussman Anomaly



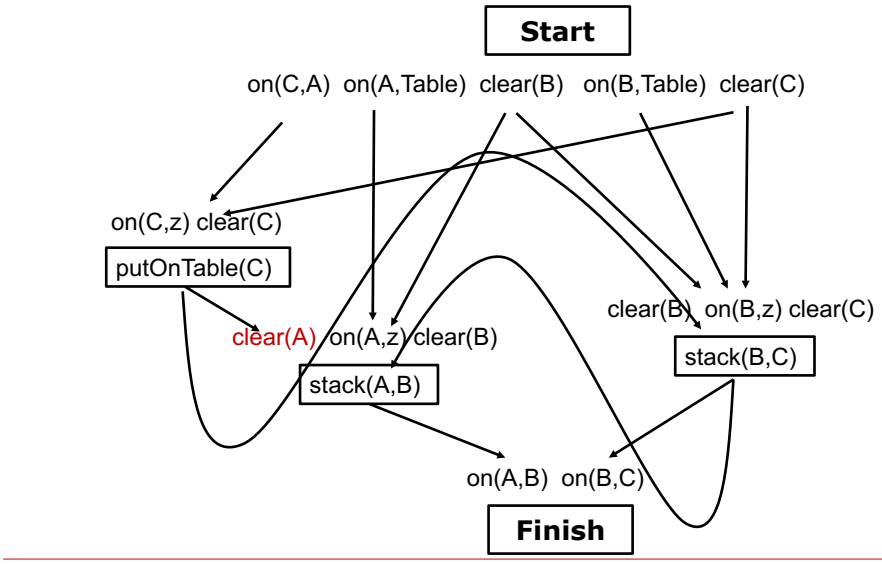


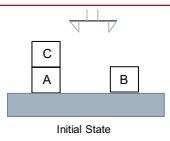
stack(B,C) clobbers clear(C)
Order: After putOnTable(C)

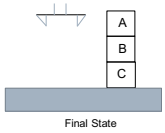


Solve **open precondition**
This causes a threat;
Fix by demotion

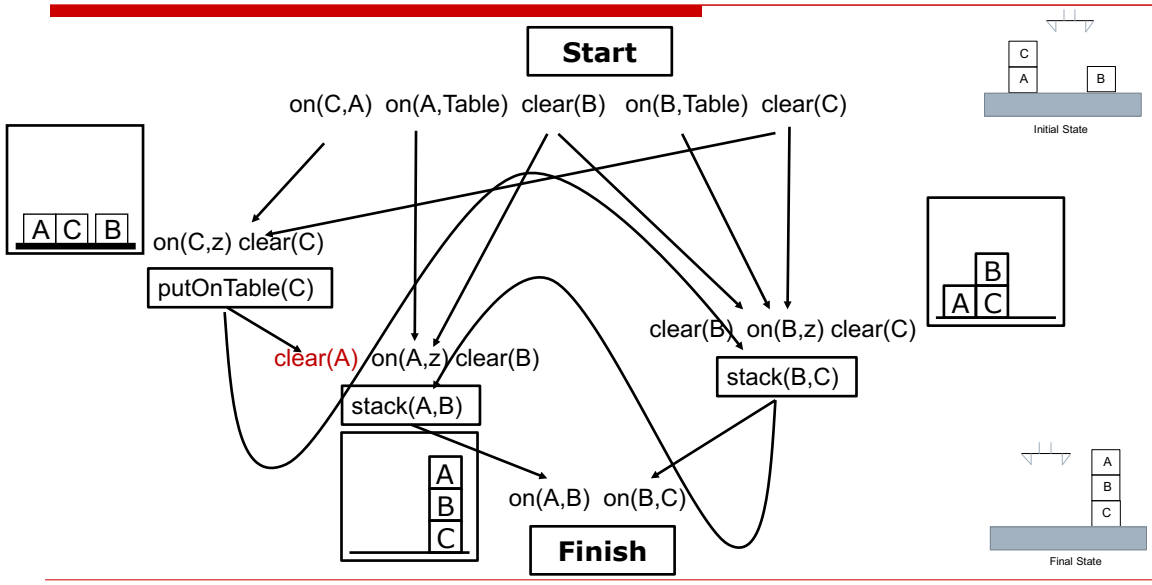
Sussman Anomaly







Sussman Anomaly: Final Plan



Summary

- Planning agents search to find a sequence of actions to achieve a goal using a flexible representation of states, operators, goals, plans
  - STRIPS language describes actions in terms of their preconditions and effects
- It isn't feasible to search through the entire space as was done with problem-solving search agents
  - regression planning focuses the search
  - STRIPS assumes sub-goals are independent
  - POP uses Principle of Least Commitment, declobbering to arrive at the plan.
- Given the fact that even the simplest planning problems are hard, we need to look for methods to speed up search.