

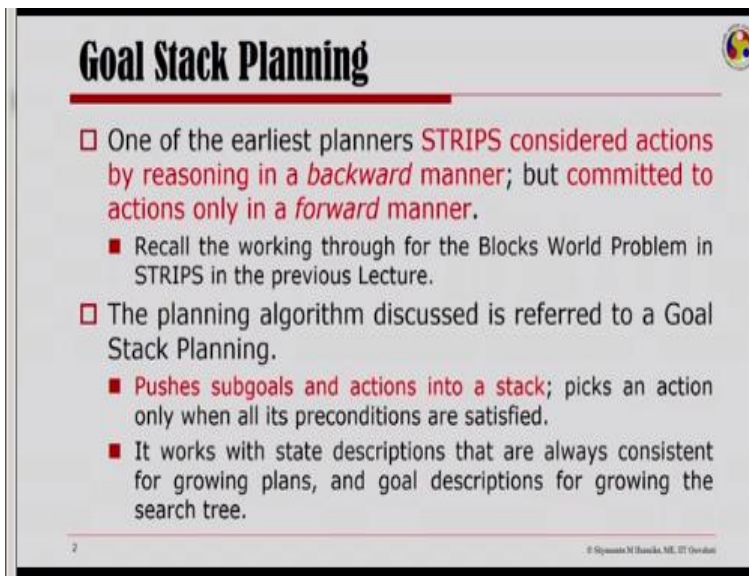
Fundamentals of Artificial Intelligence
Professor Shyamanta M. Hazarika
Department of Mechanical Engineering
Indian Institute of Technology- Guwahati

Lecture-22
Plan Space Planning

Welcome to fundamentals of artificial intelligence, we continue our discussion on planning, in the last lecture we had introduced the idea of a planning agent. We saw how planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to arrive at a solution. We had introduced situation calculus a variant of first order logic in which beliefs about the changing world can be represented.

Situation calculus enables a knowledge base agent to reason about actions and does look at the consequences of actions to finally arrive at a plan. We have looked at strips a representation of the situation calculus for planning, in strips every action can be seen as an operator that syntactically changes the world model and using such a representation we are in a position to arrive at a plan using what is called goal stack planning.

(Refer Slide Time: 02:13)



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 Lecture.
- 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.

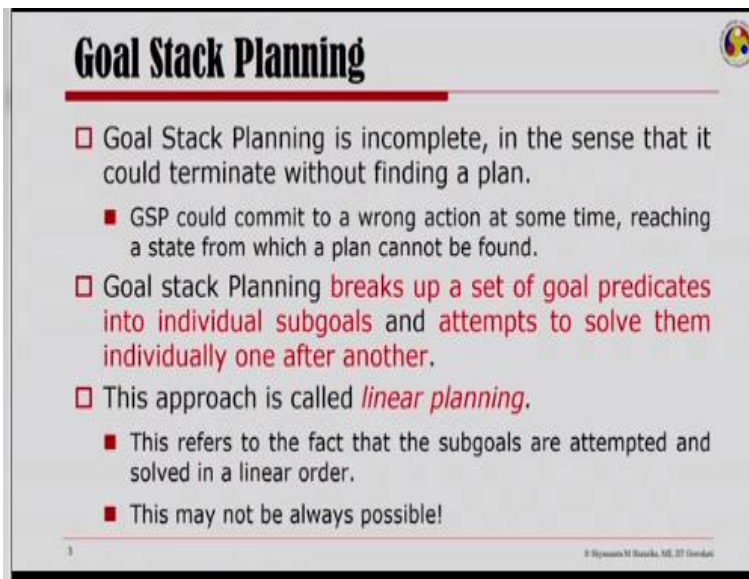
© Shyamanta M Hazarika, IIT Guwahati

The idea in goals stack planning is to push the sub goals and the actions into a stack and pick an action only when all it is preconditions are satisfied, recall that in the last lecture we had walked

through a blocks world problem in strips. And we saw how each of the sub goals and the actions where on a stack and planning was about picking an action as it is preconditions where being satisfied.

The planning algorithm referred to as goals stack planning works with state descriptions which are always consistent for growing plans and goal descriptions for growing the search tree.

(Refer Slide Time: 03:15)



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!

3 © Rajaraman M. Borra, IIT Guwahati

One needs to realize that goals stack planning is incomplete, in the sense that it could terminate without finding a plan. Now goal stack planning could commit to a wrong action at some time and thereby it could reach a state from which a plan cannot be found. Goal stack planning as already discuss in the previous lecture breaks up a set of goal predicates into individual sub goals and then attempts to solve them one after another.

Now when these sub goals are being encountered they come one after another in a sequence and therefore the approach is also called linear planning, linear planning for the sub goals are attempted and solve in a linear order. Now this may not be always possible as we saw in the blocks world problem, where it may not be possible to break the main goal into 2 sub goals which we refer to as the Sussman anomaly, also many of the planners in the real world would love to act not in sequence but operate in parallel.

(Refer Slide Time: 04:51)

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.

4 © Seymour M. Rosnick, M.S., Ph.D.

So we today look at another idea of planning which is referred to as the plan space planning, plan space planning works with plan structures that is you start with a plan and modify it to arrive at the final plan. Plan space planning is also referred to as nonlinear planning, for you can take any part of the plan and may change it to arrive at a goal. For example let us say you had made a plan to get some medicine that would required that you go to a pharmacy.

On the way you could then decide to bring some fruit as well, this would need that you would move from the pharmacy to a fruit shop next door. Now here the plan space approach as you can see does not constrain one to focus on only one sample continuously and one can shift attention midway and in the process often solve problems like the Sussman anomaly correctly.

(Refer Slide Time: 06:08)

Plan Space Planning

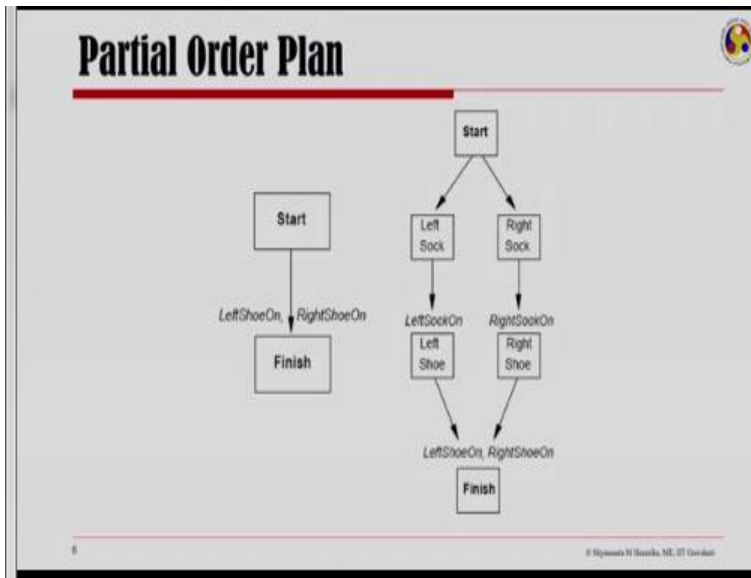
- The planning approach described so far, reason with states.
 - The planner is looking at a state and a goal; if the state satisfies the goal, it terminates.
 - Else, it makes a search move over the state space looking for actions to add to the plan.
- Plan space planning is an alternative; wherein 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.**

5 © Srinivasan M. Swaminathan, MS, ST, Ghent

The planning approach that we have describe so far actually at reason with states and the planner is looking at a state and a goal. If the state satisfies the goal we have got a plan and the planner terminates or else it makes a search move over the state space looking for actions to add to the plan. On the contrary in planned space planning, the idea is to consider the space of all possible plans and search is over the space for a plan.

Now algorithms in this category represent a plan as actions arrange in a partial order and hence these are also referred to as partial order plans. Take the example of you getting ready in the morning to go out and you have to put on your shoes, now you have to put the left shoe as well as the right shoe, whether putting the left shoe comes first or putting the right shoe comes first does not matter.

(Refer Slide Time: 07:20)



And therefore the only thing that matters is, that the final state I should have the left shoe on and the right shoe on, these type of plans are called partial order plans, for here the left socks and the left shoe has an order fixed, the right socks and the right shoe has an order fixed. If now I bring in a third step which is about the right socks to be added to the plan.

(Refer Slide Time: 07:53)

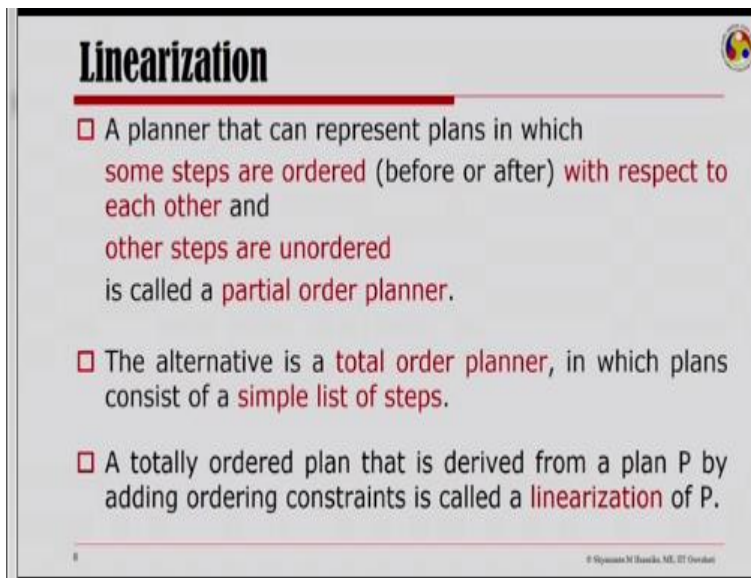
Least Commitment

- 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.

Now we only want to make sure that putting on the right socks comes before putting on the right shoe, we do not care where they come with respect to the left shoe. Now such planners when they use the idea of putting these steps they use the principle of least commitment, the least commitment principle states that one should only make choices about things that you currently care about, leaving the other choices to be worked out later.

Now this is a good idea because making a choice about something you do not care about now you are likely to make the wrong choice and have to backtrack later. So instead of making a lot of backtracks it would be best to commit to the least things now.

(Refer Slide Time: 08:51)



Linearization

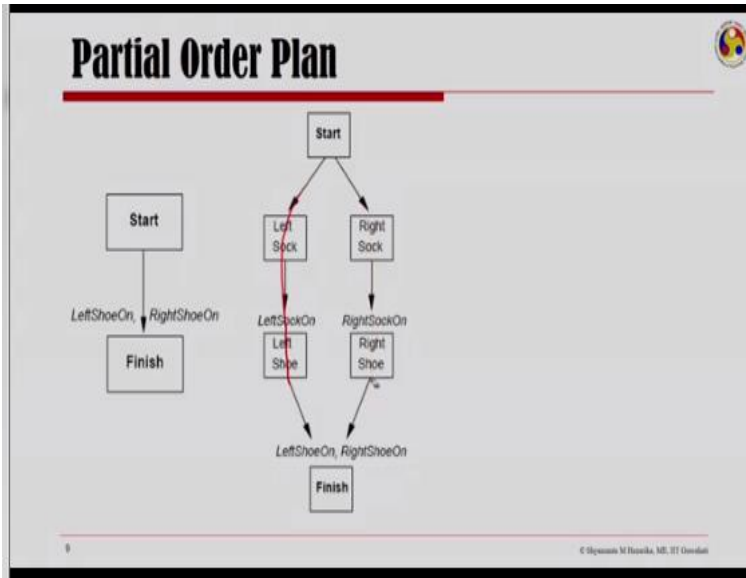
- A planner that can represent plans in which some steps are ordered (before or after) with respect to each other and other steps are unordered is called a **partial order planner**.
- The alternative is a **total order planner**, in which plans consist of a **simple list of steps**.
- A totally ordered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

© Srinivasan, M. Bhanu, M. E. Ouellet

A planner that can represent plans in which some steps are ordered 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. Like example of putting on the shoes, I put the left socks, then I put the left shoe, I put the right socks and then I put the right shoe. Here the only sequence that is to be maintain is that the left socks has to be put before the left shoe and the right socks need to be put before the right shoe.

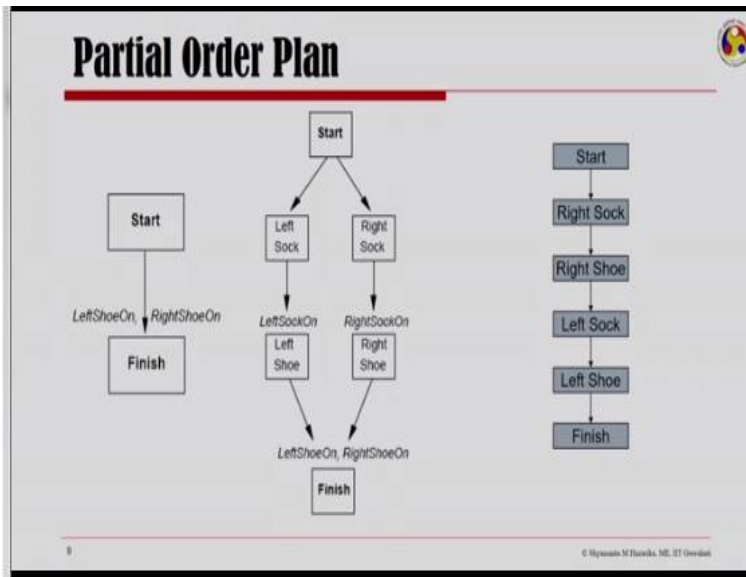
I can in a sense put the left socks and the right socks, thereafter put the right shoe and then the left shoe. So a partial order plan allows one to have steps which are unordered in a sense and only some steps are ordered. A total order plan is about a simple list of steps and now we can see that a totally ordered plan that is derive from a plan P by adding ordering constraints is called a linearization of P.

(Refer Slide Time: 10:27)



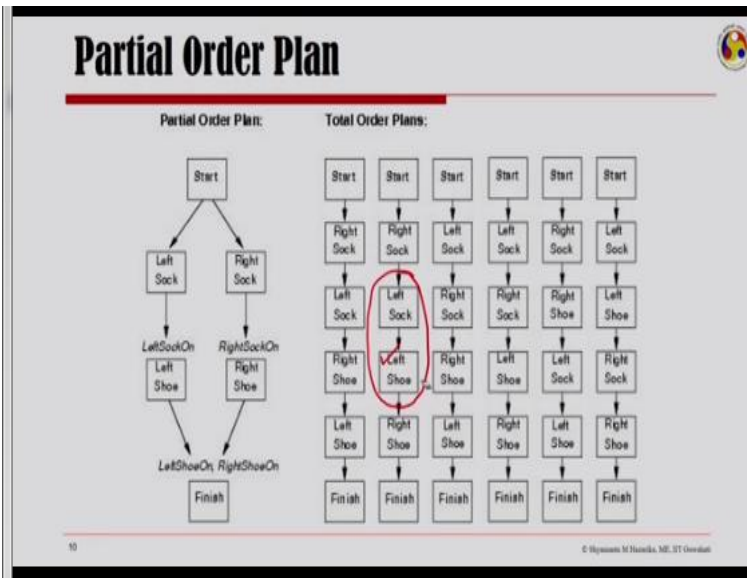
So let us come back to that example of putting the left shoe and the right shoe, now as we were discussing I could put the left socks and then the right socks or I need to put the left socks and the left shoe. So the only ordering that is required is the left socks and the left shoe and the right socks and the right shoe, given this partial order plan to me.

(Refer Slide Time: 11:04)



I could think of generating a number of total order plans like here on your right is another plan where I start with the right socks put the right shoe, then the left socks then the left shoe and finish putting on the shoes.

(Refer Slide Time: 11:24)



A little consideration will show, that I can have the following 6 total order plans out of the partial order plan of putting the left shoe and the right shoe. I could put the right socks, the left socks, the right shoe, the left shoe or I could do the right socks, left socks and then put the left shoe and then the right shoe. Because the only ordering that I need to maintain is about the socks and the shoe. These 6 total order plans that are possible to be obtain from the partial order plan on the left of the screen are called linearizations of this plan P.

(Refer Slide Time: 12:10)

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.

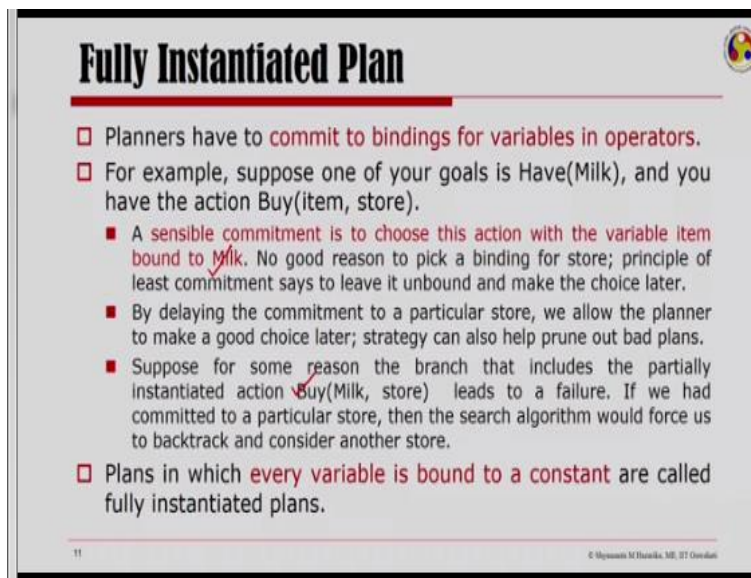
© Raymond M. Haralick, ME, ST, Georgia

Now planners even if I have not shown it in the total order or the partial order plans that I have shown in the previous slide have to commit to bindings for variables in operators. For example suppose our goal is to have milk and you have an action to buy an item from a store. Now a

sensible commitment for instantiation is to choose this action with the variable item bound to milk. Because my goal is to have milk therefore it is sensible to bind the item to milk.

But then there is no good reason to pick up a binding for store, this is what the principle of least commitment says, I would leave store unbound and make the choice later.

(Refer Slide Time: 13:13)



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.

11 © Srinivasan M. Thirumala, IIT Guwahati

Now, one should realize that when I am delaying the commitment to a particular store I am in a sense allowing the planner to make a good choice later. And using some strategy I may be able to also prune out bad plans, now suppose for some reason let us say the branch that includes the partially instantiated action of buying milk from a particular store would lead to a failure. Now if we had committed to a store as well.

Then my search algorithm would force me to backtrack and consider another search keeping myself least committed and only committing to the item of milk, keeping the store unbound would not generate this possibility. Now a fully instantiated plan is one in which every variable is bound to a constant plans in which every variable is bound to a constant are called fully instantiated plans.

(Refer Slide Time: 14:28)

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 < B reads "A before B" but ~~not~~ necessarily immediately before B
 - Alert: caution to cycles A < B and B < 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~~ $\xrightarrow{\checkmark\text{RightSockOn}}$ ~~RightShoe~~
4. A set of **open preconditions**
 - Planners work to reduce the set of open preconditions to the empty set without introducing contradictions

© Srinivas M. Harekula, M.E., IIT Chennai

We are now in a position to define what are the components of a plan in more formal terms, a plan has 4 components 1, a set of actions, where each action is one of the operators for the problem. Thereafter there are a set of ordering constraints A before B but then this before does not mean necessarily immediately before B. And one needs to be very careful when dealing with these ordering constraints to avoid cycles.

The third component of a plan is a set of causal links, causal links are also referred to as protection intervals between actions A to B over p read as A achieves p for B means that p must remain true from the time A is applied to the time B is applied. Now for the example that we have been talking off about putting on the socks and the shoes, right socks and the right shoe when I am talking of these 2 scenarios then right socks on should remain true, for me to go from right socks to right shoe.

The fourth component of a plan is a set of open preconditions, the idea is to reduce the set of open preconditions to the empty set without introducing contradictions to arrive at a plan.

(Refer Slide Time: 16:24)

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 < 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.

13 © Raymond M. Hullin, M.E., ET Gordon

Now the initial plan even before any refinements have taken place is one which simply describes the unsolved problem in essence it just contains 2 steps the start and a finish with the start before the finish. Both the start and the finish have null actions associated with them, so when it is time to execute the plan they are ignored. The start step has no preconditions all it specifies is its effect and the effects add all the propositions that are true in the initial state.

On the other hand finish has the goal state as its precondition and does not have any effects by defining a problem this way, in terms of just the start and the finish. The planner can start with the initial plan and manipulate it until they come up with a plan that is a solution.

(Refer Slide Time: 17:38)

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**.

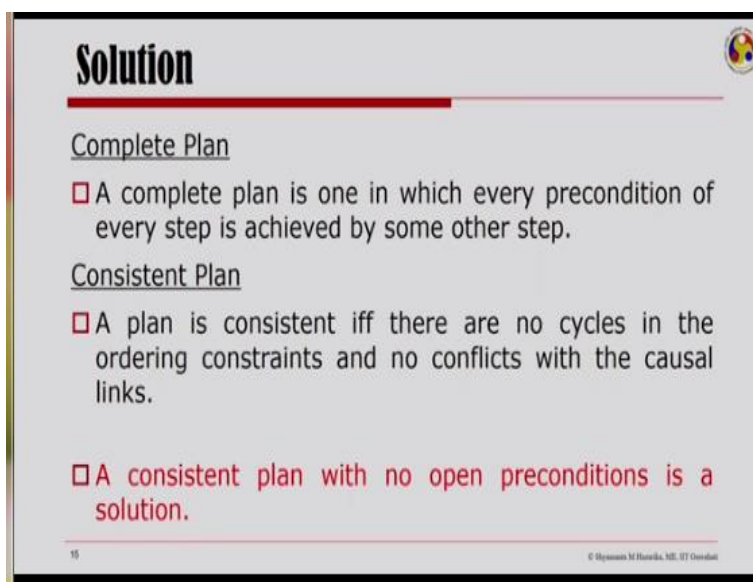
14 © Raymond M. Hullin, M.E., ET Gordon

Now a solution for a planning problem is a plan that an agent can execute and one that guarantees achievement of the goal. Consider a scenario in which we wanted to really make it easy to check that a plan is a solution, in that case we could insist that only fully instantiated totally ordered plans can be solutions. But one needs to understand that such a scenario is unsatisfactory for 3 reasons.

One it is natural for a planner to return a partial order plan than to arbitrarily chose one of the many linearizations of it. Like in the example of putting on the shoes I had 6 linearizations out of the partial order plan it is natural for planner to return the partial order plan rather than to commit to one of the 6 linearizations. Second, many agents are capable of performing actions in parallel and therefore it makes sense to allow solutions with parallel actions rather than just a sequence of steps.

Lastly when you are creating plans that may be combined with other plans to solve larger problems, one needs to retain the flexibility afforded by the partial ordering of actions. And therefore when we are looking for a solution to a planning problem a solution is a plan that is not fully instantiated or not a totally ordered plan. We want the solutions to be partially ordered and we use a very simple definition saying that such a partially ordered plan will be solution when it is complete and consistent.

(Refer Slide Time: 20:05)



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.

15 © Srinivasan M. Harshini, IIT Chennai

Now let us look at what we mean by complete and consistent plan, a complete plan is one in which every precondition of every step is achieved by some other step. A consistent plan is one in which there are no cycles in the ordering constraints and therefore no conflicts with the causal links. Now if I have a consistent plan with no open preconditions then I call it a solution or the plan that I am looking for.

(Refer Slide Time: 20:42)

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)

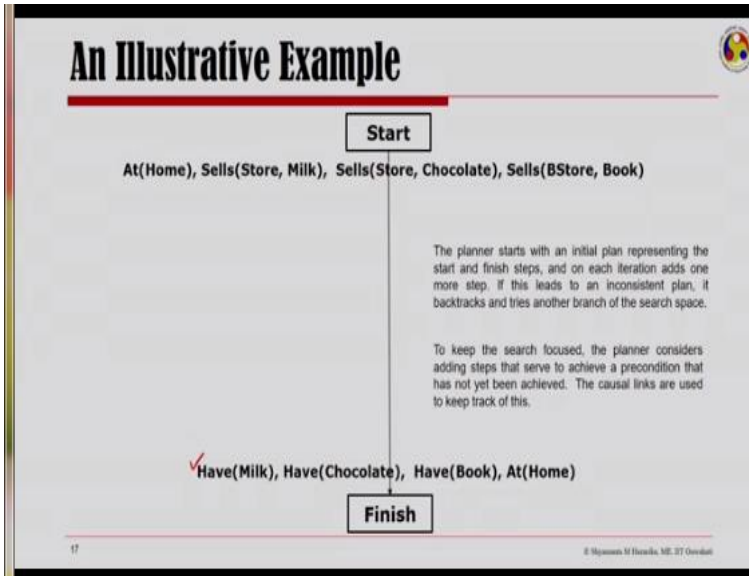
Goal: ✓ Have(Milk), ✓ Have(Chocolate), ✓ Have(Book), ✓ At(Home)

The slide features a small robot icon on the right. Red arrows indicate a path from the start state to the goal state, showing the sequence of actions: moving to the store to get milk and chocolate, then moving to the book store to get a book, and finally returning home.

Let us take an illustrative example to understand this idea, considered a robot needs to solve the following it is need to get a quart of milk, a dark chocolate and a good book, how does the robot go about solving this. Now as discuss we should note that we will start with just 2 states one the start the other the finish. So at start the robot is at home there is a store that sells milk, there is another store that sells books which we call the book store and the store that we have sells milks and chocolates.

Now the goal for the robot is to have milk, have chocolate, have a book and be at home, now a little reasoning will let you see that to start with the robot is at home to get these things it must go from home to the store collect these items and somehow finally return back home to satisfy my n conditions.

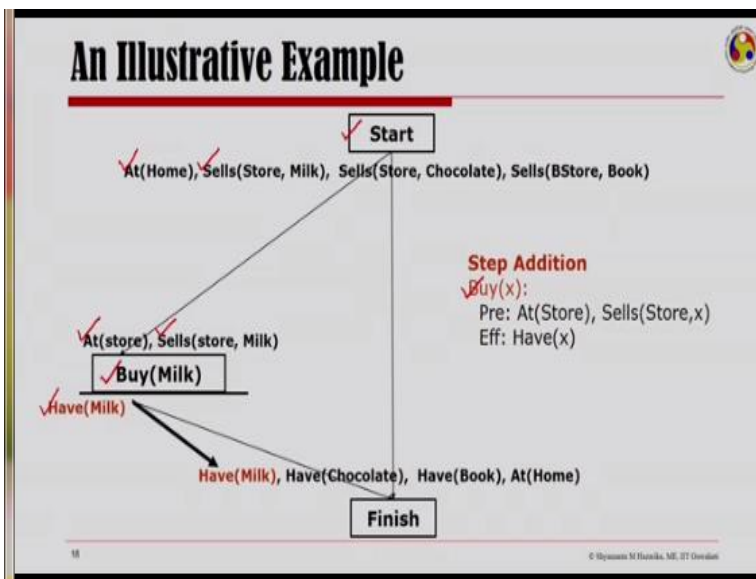
(Refer Slide Time: 22:15)



So how do we plan this whole thing in a partial order plan, so we start with a start state and a finish the planner has an initial plan representing the start in the finish steps. And now on every iteration I would add one or more step, if it leads to an inconsistent plan it would backtrack and try another branch of the search space. In order to keep the search focus, the planner would only add steps that serves to achieve the precondition that has not yet been achieved.

And the causal links are used to keep track of this, so as you see the first condition not satisfied is having milk.

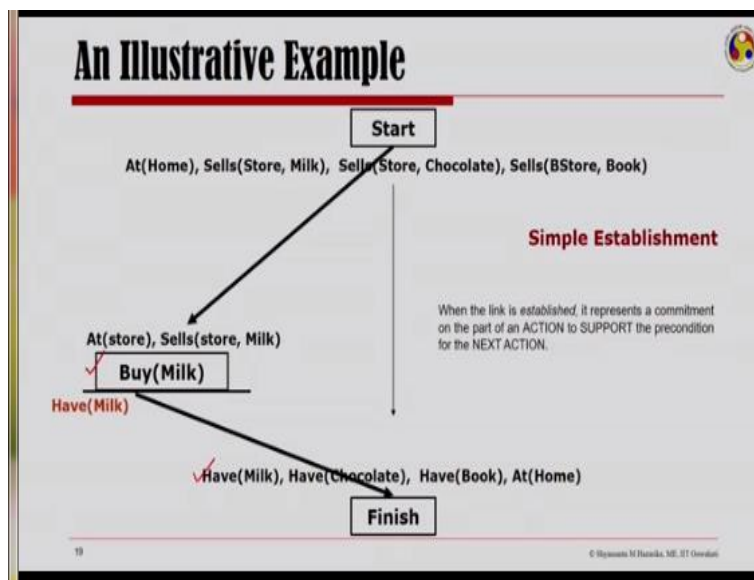
(Refer Slide Time: 23:11)



And therefore the first step would be an addition of an action of buying X, now any step about buying X there would be a set of preconditions and a set of effects. If you are buying X you must be at the store and the store must be selling X, now after you have done this action you would have X. So looking at this, you would love to add the step of buying milk here in your plan of things. Now you would see that the preconditions are that you must be at the store and the store must be selling milk.

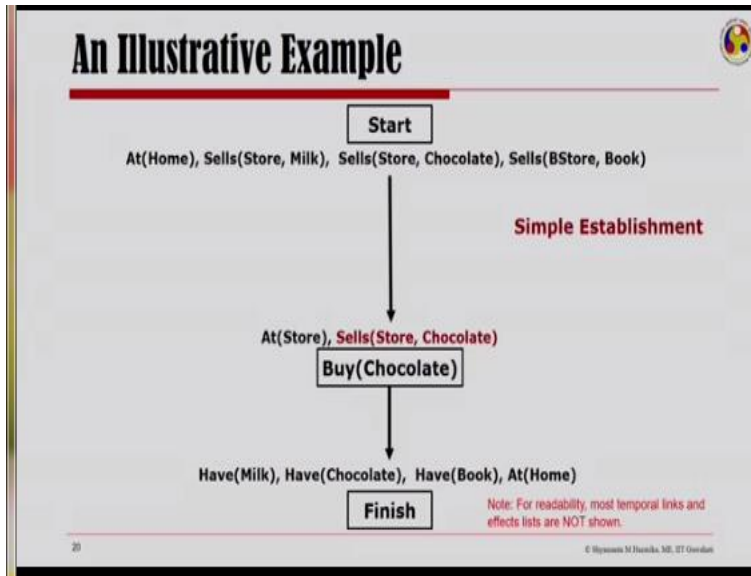
If you look back here in the start state that we have put the conditions, we do have that the store sells milk but we are at home. So somewhere down the line, we should have another action between buying milk and the start which will take me from home to the store we will come back to this in a little while. But first we are satisfying one condition of having milk by introducing an action of buying X.

(Refer Slide Time: 24:46)



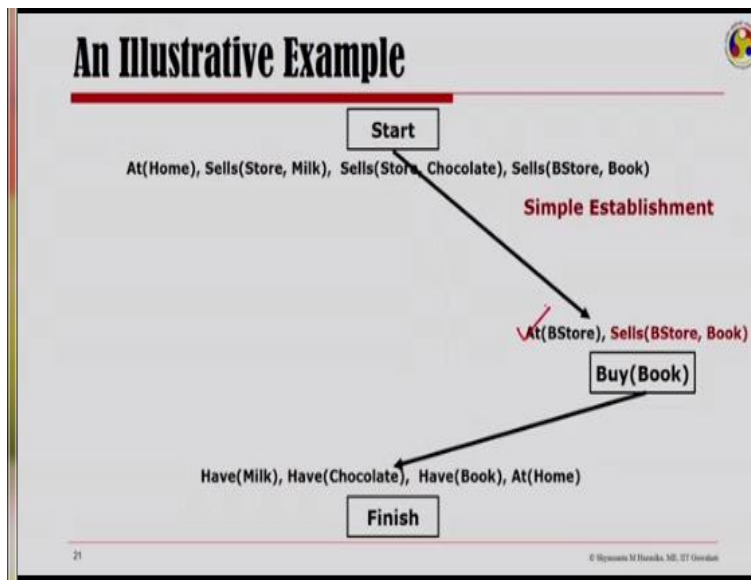
The next is about this simple establishment, now we say that the link is established, if it represents a commitment on the part of an action to support the precondition for the next action. So here there is a simple establishment of buying milk because it supports the precondition of have milk for my finish step.

(Refer Slide Time: 25:18)



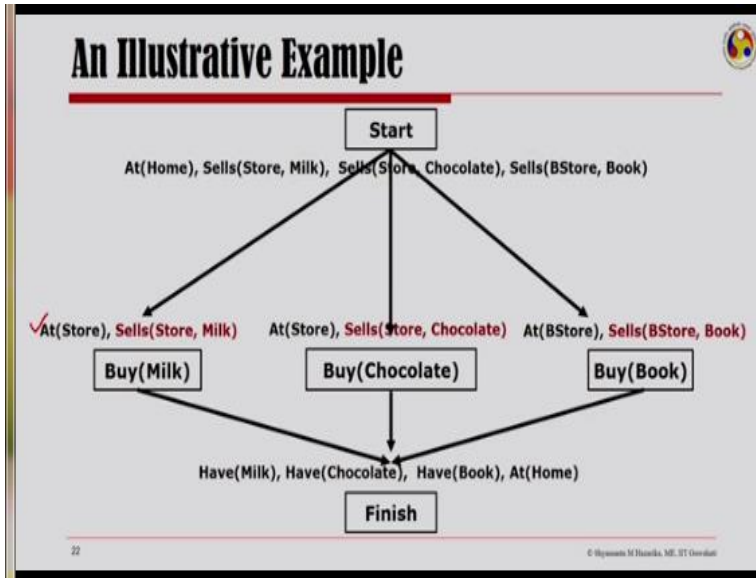
Similarly we could add simple establishments of getting a chocolate and another simple establishment of getting the book.

(Refer Slide Time: 25:31)



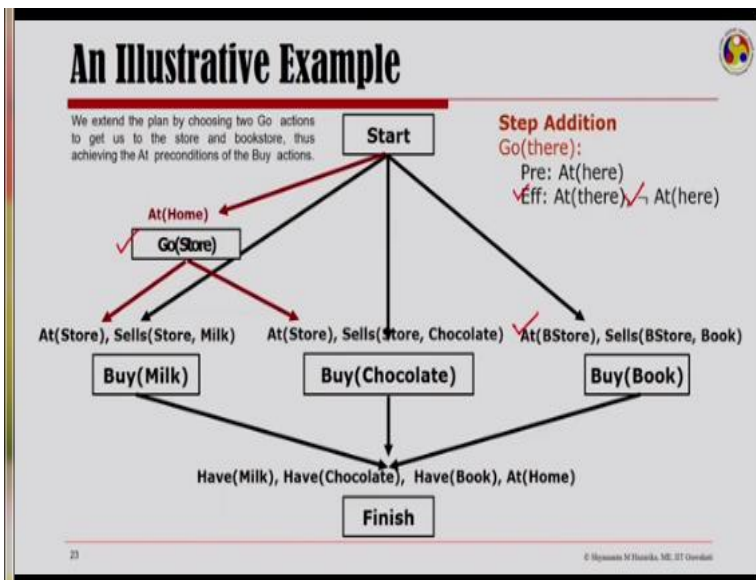
Now one need to realize at this point that getting the chocolate and the milk was about being at the store, whereas getting the book was about being at the book store.

(Refer Slide Time: 25:43)



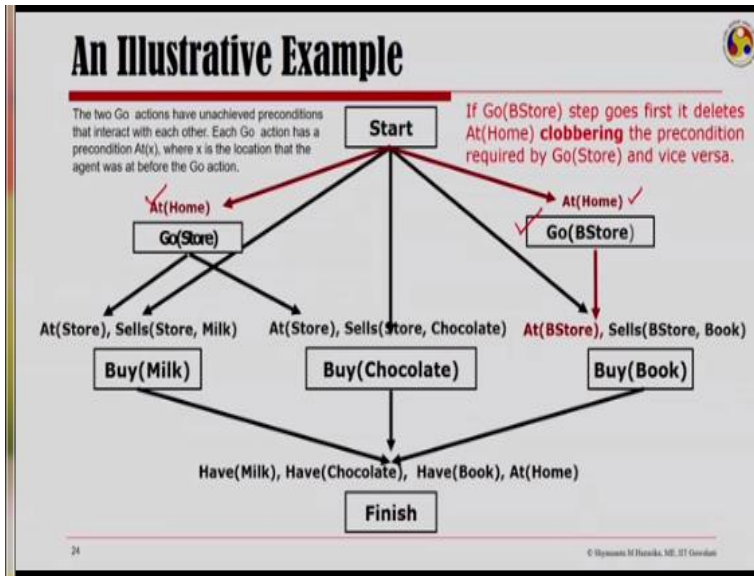
So we have 3 links here of actions, buy milk, buy chocolate and buy book that we have added, now in order to satisfy that I am at the store.

(Refer Slide Time: 26:00)



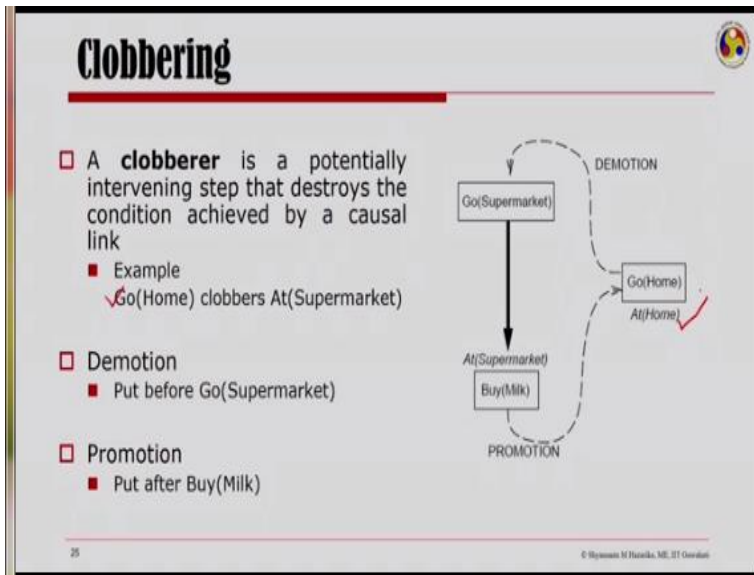
I would need to extend the plan by choosing to go actions, one that gets me to the store and another that get me to the book store. So I have another step addition of go there, so the precondition is that I must be here and the effect of that go there action is that I am there but I am no longer here, so not off at here. This I introduce in between my start and at store, so I have this action here, go store being introduce and the precondition would be that I am at home and its effect would be that I am at store, I could be at the store for a chocolate or for milk.

(Refer Slide Time: 27:06)



And I introduce here another go action which is about go to the book store, whose effect is that I am at the book store. Now we have arrived at a very interesting scenario now, if go book store step goes first then it will delete the at home thereby clobbering the precondition required for the go store, now that is called the threat.

(Refer Slide Time: 27:27)



So this idea of threat or clobbering is a potentially intervening step that destroys the condition achieved by causal link, like if I say go home then it clobbers at super market. So in order to avoid that threat I can either do it demotion that is put before go super market or I could do a promotion put after buy milk the idea of going home.

(Refer Slide Time: 28:12)

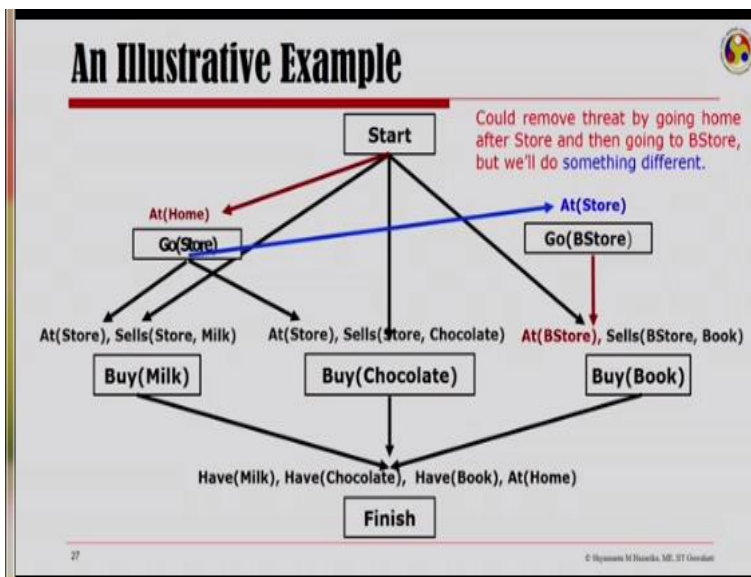
Declobbering – Threat Removal

- **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 $S_2 < S_3$

© Srinivasan M. Harshika, M.E., ST. Chennai

So declobbering is that I have to do for removing the threat, now a threat as I have seen it in the original planning example that I am talking of is a step that deletes or clobbers a needed effect. So S_2 requires an effect of S_1 that is there is a causal relation between S_1 and S_2 but the effect of S_3 is to undo the effect S_2 requires. So then we say that it is a threat, now S_3 therefore cannot occur between S_1 and S_2 it must either be before S_1 in which case I say I have promoted S_3 or it could be after S_2 in which case I say I have done a demotion.

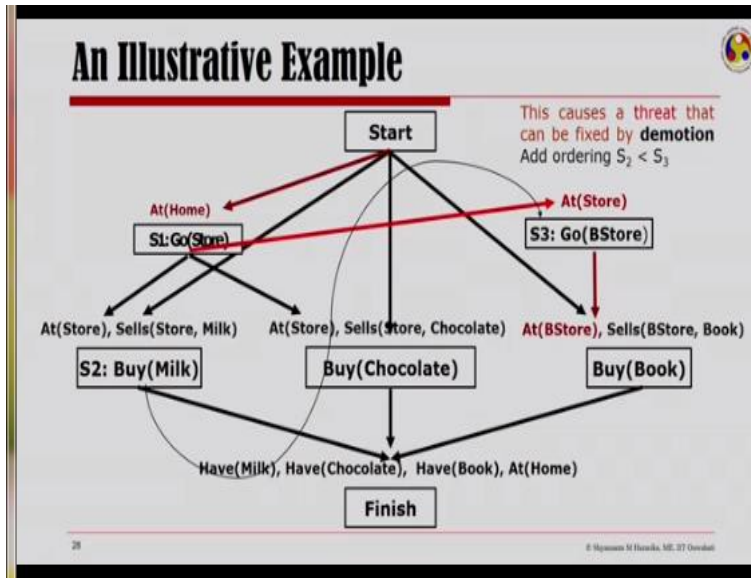
(Refer Slide Time: 29:09)



Let us get back to that illustrative example and now we could see that the go store and the go book store they are clobbering each other. So I could remove the threat by going home after going to the store and then going to the book store but we would do something different. So after

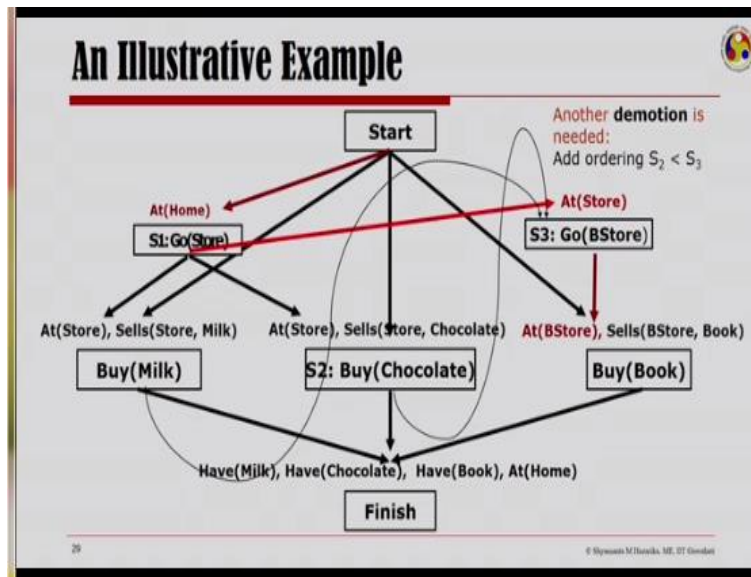
visiting the store if you could visit the book store then this idea of at home being clobbered by book store will not be there.

(Refer Slide Time: 29:46)



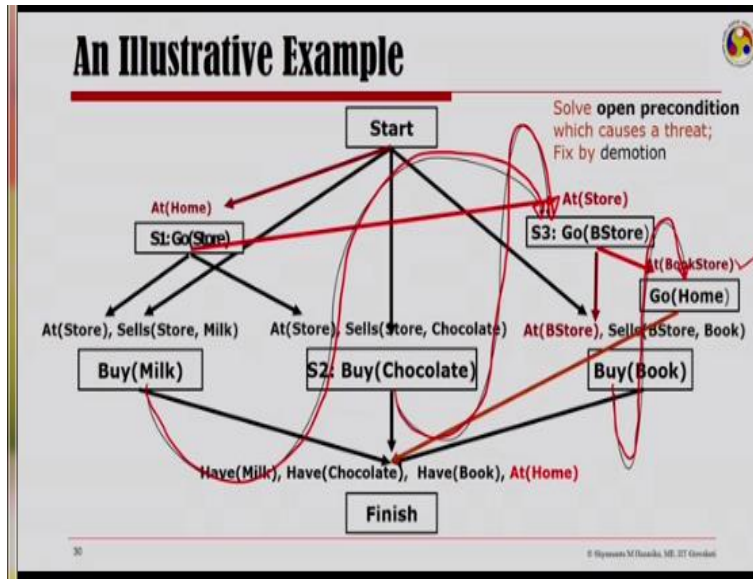
So this is what I would do, so basically the threat would be fix by a demotion I would add an ordering that I should buy the milk and thereafter go to the book store and demotion would be brought in, so an ordering would be made between S2 and S3.

(Refer Slide Time: 30:11)



Another demotion is needed between buying the chocolate and going to the book store, so that is what is added. Now once I am at the book store after getting the milk and the chocolate I can buy the book, but then it would require that at I return back home.

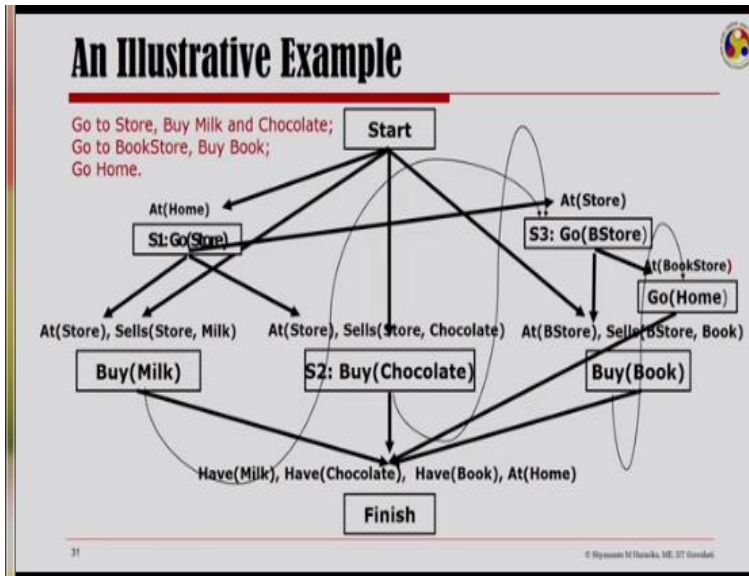
(Refer Slide Time: 30:33)



So that would mean that I would have to add a go home action here at the book store, now that open precondition that I am at home and adding the action go home also causes of threat. Because if you are at the book store and if you go home from there then you would not be at the book store any longer to buy the book. So I have to introduce this demotion, so there are couple of demotions that are introduce, that are very important to realize.

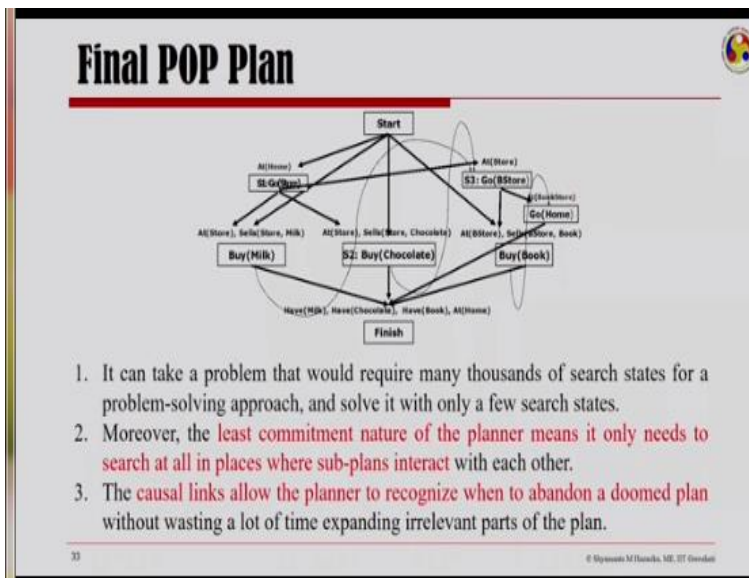
I will buy milk and then I have to be sure that then only I go to the book store or buy chocolate and then only go to the book store. So, these 2 and the third one which is about buying the book and then only going to this state about going home.

(Refer Slide Time: 31:38)



So here is the final plan, I go to the store buy milk and chocolate, go to the book store buy the book and go home.

(Refer Slide Time: 31:50)



Now if you look at this final partial order plan 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. Because of the fact that I had done a least commitment nature of planning, meaning it only needs to search at all places, where sub plans interact with each other and the causal links allow the planner to recognize when to abandon a doom plan.

(Refer Slide Time: 32:28)

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:

- ✓~~1.~~ on(A,B)
- ✓~~2.~~ 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:

34 © Stephen M. Edelkamp, MIT, 2004

Let us quickly recall our discussion of the Sussman anomaly from the blocks world problem. Now having looked at a plan generated using a partial order planning algorithm, it would be nice to see if we could solve the Sussman anomaly in a similar manner. Now quickly recalling the Sussman anomaly, here is my initial state where I have C on A and B on the table and I want A on B and B on C.

Now any non interleaved planner would typically separate the goal into sub goals that is I would have A on B and B on C. If I am talking of pursuing the first goal, then the basic step is to move C out of the way and put A on top of B. While this sequence accomplishes goal 1, now the agent would have no option but to undo this step in order to achieve goal 2.

(Refer Slide Time: 33:51)

Sussman Anomaly

Initial State

Final State

Initial State Goals: `on(C,A)`, `clear(C)`, `clear(B)`, `onTable(A)`, `onTable(B)`, `handEmpty`

Final State Goals: `on(A,B)`, `on(B,C)`, `clear(A)`, `onTable(C)`, `handEmpty`

Noninterleaved planners typically separate the goal into sub-goals:

- `on(A,B)`
- `on(B,C)`

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

- 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.
- If instead the planner starts with Goal 2, the most efficient solution is to move B onto C. But again, the planner cannot pursue Goal 1 without undoing Goal 2:

© Sijmen M. Harste, SE, ST, Ghent

Now in a similar manner if instead the planner starts with goal 2, then the most efficient solution is to move B onto C but in order to achieve goal 1 now, it has to undo this path.

(Refer Slide Time: 34:07)

Sussman Anomaly

Initial State

Final State

Initial State Goals: `on(C,A)`, `clear(C)`, `clear(B)`, `onTable(A)`, `onTable(B)`, `handEmpty`

Final State Goals: `on(A,B)`, `on(B,C)`, `clear(A)`, `onTable(C)`, `handEmpty`

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

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

© Sijmen M. Harste, SE, ST, Ghent

So on A, B is undone to clear B when solving B on C as can be seen here.

(Refer Slide Time: 34:19)

Sussman Anomaly

Initial State

Final State

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

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

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

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

And similarly on B, C is undone to clear C when solving A on B.

(Refer Slide Time: 34:30)

Sussman Anomaly

Initial State

Final State

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!

So if you begin work on A, B by clearing a that is putting C on the table then achieve on B, C by stacking B on C and achieve on A, B by stacking A on B then we could achieve the final result. But we cannot do this using a stack within strips interleaving is required.

(Refer Slide Time: 34:56)

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_2 occur after G_1
- all of the steps for a sub/goal must occur "atomically"
- e.g. STRIPS
- can't solve the Sussman Anomaly

□ Interleaving planner

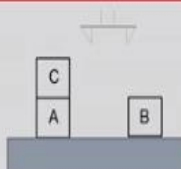
- can intermix order of sub/goal steps
- can solve the Sussman Anomaly by interleaving steps:
unstack(C,A), Pickup(B), Stack(B,C), Stack(A,B)

38 © Srinivasan M. Bhatnagar, MIT, EE-6.034

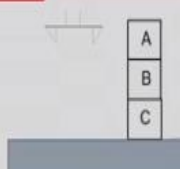
So non interleaving planners where either all the steps for achieving G_1 occur before G_2 or all the steps for achieving G_2 occur after G_1 cannot solve the Sussman anomaly. If I am in a position to intermix the order of the sub goals or what are called interleaving planners I can solve the Sussman anomaly.

(Refer Slide Time: 35:24)

Sussman Anomaly



Initial State



Final State

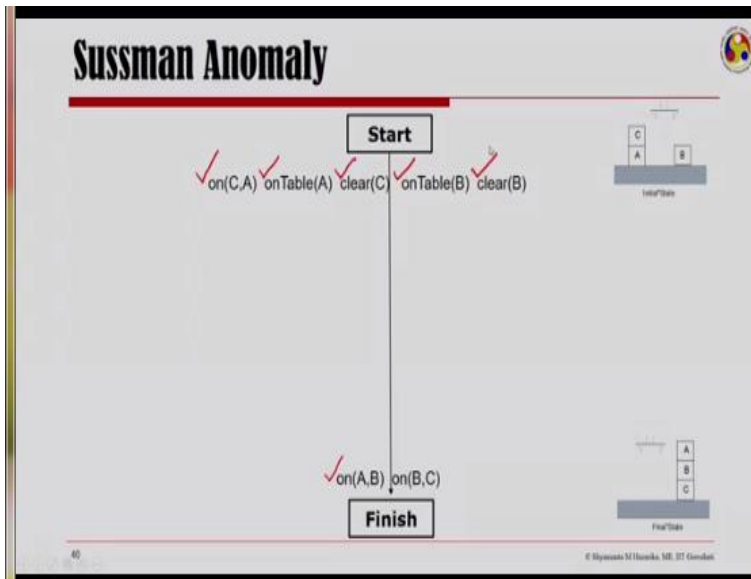
\checkmark clear(x) \checkmark on(x,z) \checkmark clear(y) \checkmark stack(x,y) \checkmark on(x,z) \checkmark clear(y) \checkmark clear(z) \checkmark on(x,y)	\checkmark clear(x) \checkmark on(x,z) \checkmark putOnTable(x) \checkmark on(x,z) \checkmark clear(z) \checkmark on(x,Table)
---	--

39 © Srinivasan M. Bhatnagar, MIT, EE-6.034

And the partial order planner is one planner that can solve the Sussman anomaly, so let us look at that. We have the initial state here where B is on the table and C is on A, we have the final state where I have B on C and A on B with C on the table. Let me take help of just 2 actions one which is stacking x on y under a scenario where x needs to be clear and x needs to be on some z and y needs to be clear, so these are its preconditions.

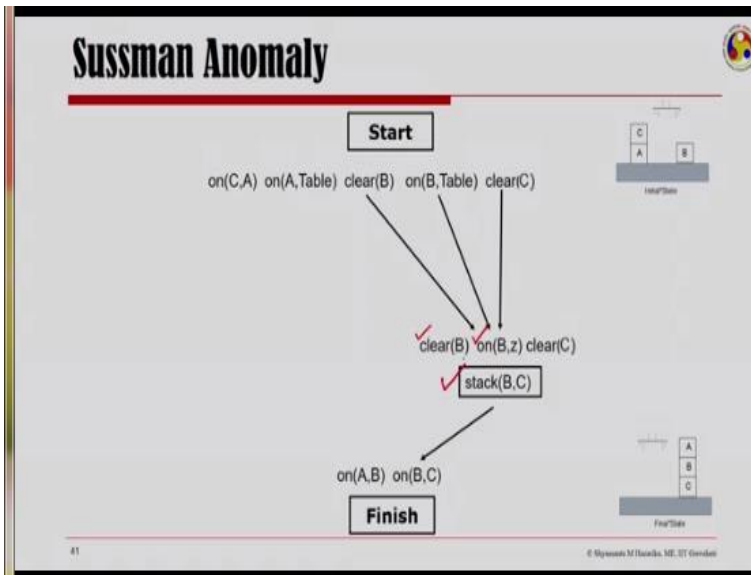
And the x on scenario I can stack x on y, once I have x on y, x would not be on z and y would not be clear, whereas z would be clear and stacking x on y would finally give me x on y. Similarly I have another action which is about putting x on the table, the preconditions are that x must be clear there must not be anything on top of x and x must be over some z. So if I am putting x on the table then x is no longer on z and the final result is that x is on the table.

(Refer Slide Time: 37:04)



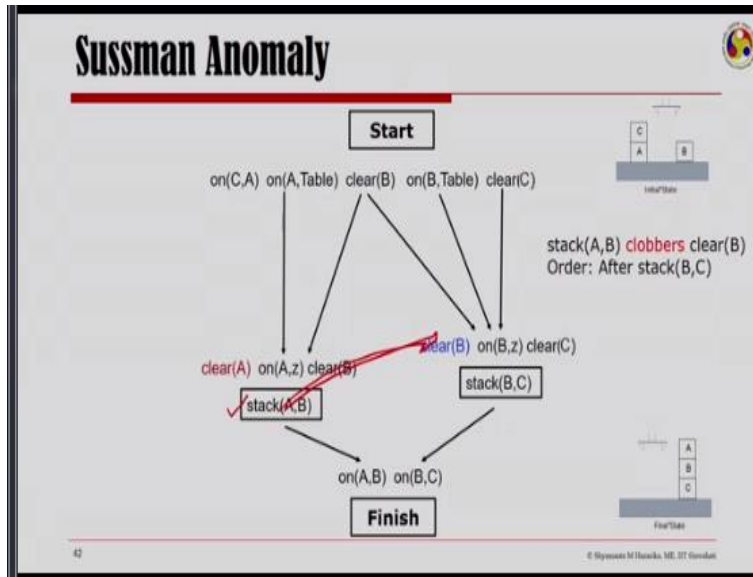
Now let us get our start and finish states, so finally it must be that A is on B and B is on C to start with I have C on A, A on the table nothing on top of C, B on the table and nothing on top of B.

(Refer Slide Time: 37:34)



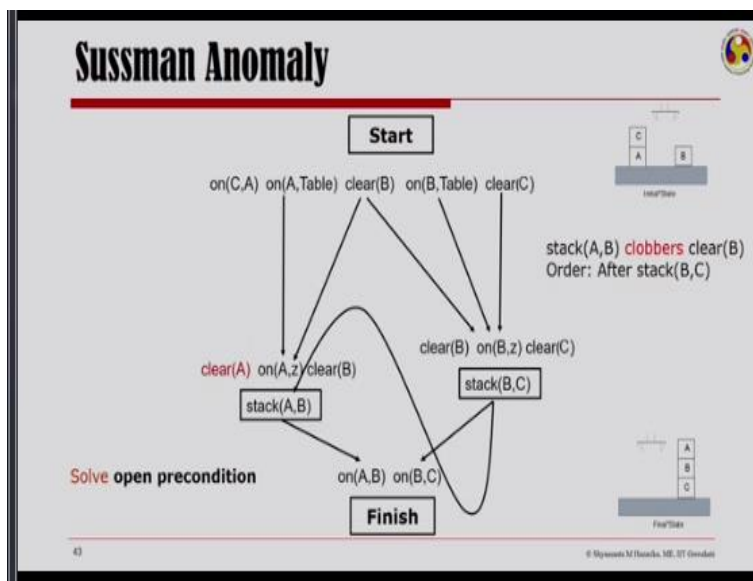
So the first action that I take is about stacking B on C, now the preconditions B must be clear, B must B on some z and these conditions are clearly visible here. So I could have this would be a C clear C, so I could clearly have a stack operation happening.

(Refer Slide Time: 38:05)



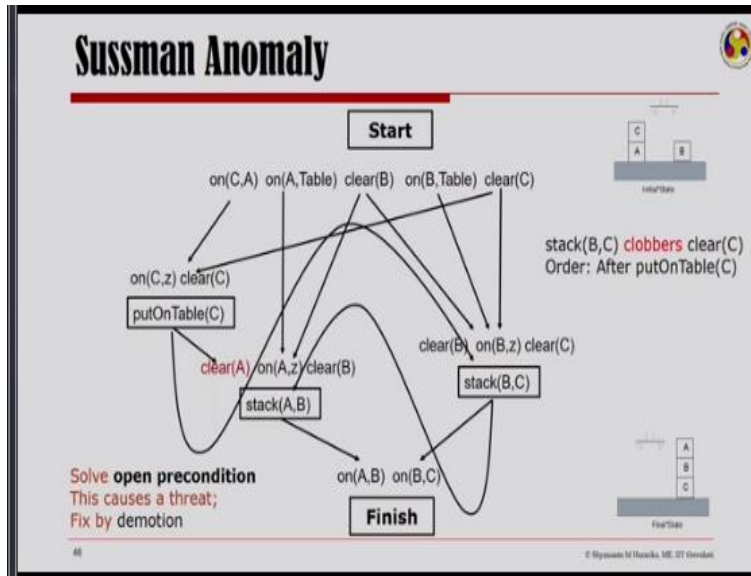
Similarly I could think of having stack A, B where A on z and B is clear, so I could do stack A, B but then you need to realize that the stack A, B clobbers the clear B here so that is a threat to stack B, C.

(Refer Slide Time: 38:37)



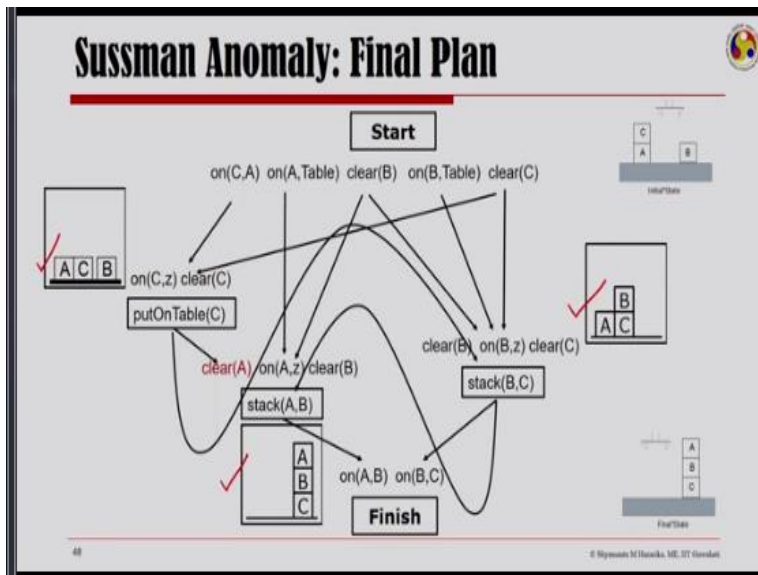
And therefore I can solve this by putting after B, C.

(Refer Slide Time: 38:45)



And then I put on the table the block C that operation which solves this precondition about clear A and now this one you could see is again causing a threat and that I must solve by put on table C before stack B, C.

(Refer Slide Time: 39:13)



So here is our final plan which is about first putting C on the table then putting B on C and then finally putting A on B. So we saw that a partial order planner can solve the Sussman anomaly very easily.

(Refer Slide Time: 39:38)

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.

49 © Benjamin M. Bonet, SE, ST Gordon

So to summarize we have looked at planning agents which search to find a sequence of actions to achieve a goal. Now they have use a very flexible representation of states, operators, goals and plans and we have looked at the STRIPS representation which describes actions in terms of their preconditions and effects. Now it is not feasible to search through the entire space as was done with problem solving search agents.

So regression planning focuses the search STRIPS assume sub goals are independent and we could do what is called the goal state planning. Partial order planners uses the principle of least commitment and we have seen how declobbering or threat removal is used to arrive at the plan, given the fact that even the simplest of planning problems are hard. We need methods to speed up search.

And this is what we would take up in our final lecture on planning, where we will see some relaxation to the constraints that we have posed to the planning problems here and also use some heuristic information to do planning thank you.