

Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 7 and 10
A First Course in Artificial Intelligence
Lecture – 62
GSP: A Detailed Example

(Refer Slide Time: 00:14)

```
GSP(givenState, givenGoal, actions)
1  S ← givenState ; plan ← ( ) ; stack ← emptyStack
2  PushSet(givenGoal, stack)
3  while not Empty(stack)
4      do x ← Pop(stack)
5          if x is an action a
6              then plan ← (plan ◦ a)
7                  S ← Progress(S, a)
8          else if x a compound goal G and G is not true
9              then PushSet(G, stack)
10         else if x is a goal g and g ∉ S
11             then CHOOSE a relevant action a that achieves g
12                 if none then return FAILURE
13                 Push(a, stack)
14                 PushSet(Pre(a), stack)
15 return plan
```

Algorithm GSP

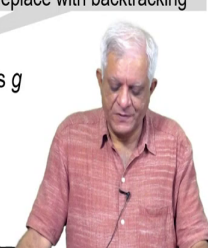
NPTEL

A planning problem

Backward search

Forward plan construction

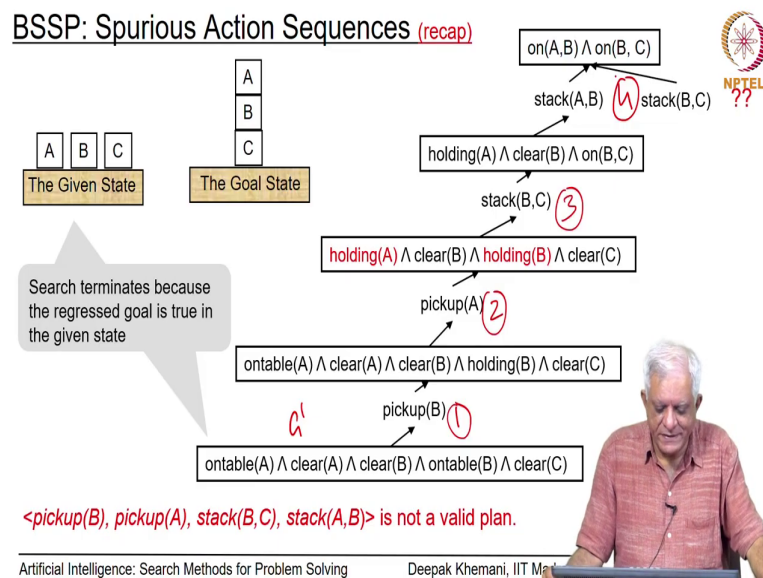
Non-deterministic choice - replace with backtracking



So, welcome back we have just finished studying the goal stack planning algorithm and just to do a very quick recap with the algorithm that is in front of you at this moment. You start off by pushing the goal that you want to achieve on to the goal stack thereby initiating search in a backward fashion then you add the individual goals from that goal set to the stack.

And then you start popping out the goals and the goal is true in the current state you do not have to do anything if the goal is not true then you have to in push in a action that is relevant to that particular individual action. When you push an action you also push the preconditions of the action after that so that the preconditions would be first checked before the action is popped out in which case the action would be applicable.

(Refer Slide Time: 01:11)

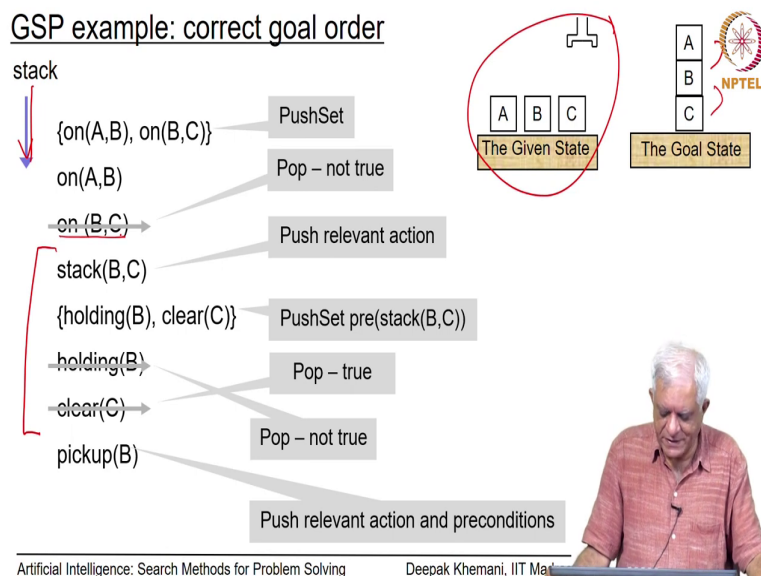


So, let us see this algorithm on the small example that we had studied earlier. So, if you remember we have this very tiny example where three blocks A B and C were on the table and you have to stack them on as to A on B and B on C and so the goal was simply that A is on B and B on C and we looked at backward stage say planning we choose the relevant action stack A on B because that achieves a goal on A, B and we regress to this goal in which you are holding A clear B and on B, C.

The backward state space algorithm could choose any of these goals to be achieve next. So, let us assume that it chooses the action on B, C and it chooses a goal on B, C and chooses a action on stack B on C and we had observed that point that what is regresses to is not something which is consistent and it nevertheless goes ahead and construct a plan and it constructs a plan in which it has reach the regress to a goal.

So, this is some goal prime that it has to regress to and that is happens to be true in the given state. So, it can terminate, but then we observe that the plan that it is produced in the first action being pick up B, second action being pick up A, third action being pick up stack B on C and the fourth action being stack A on B was not a valid plan in fact, it cannot be executed in the domain that we have defined.

(Refer Slide Time: 02:40)



So, let us see how goal stack planning addresses this problem. So, we said that goal stack planning serializes a goals sub goals and it will put them in some order. So, let us first look at the same example it is just the small example we can look at both possible orders there are two goals to be solved; one is that A is on B and the other is that B is on C and let us assume that it first choses the correct order.

Now, what is the correct order if you look at this problem, you should first solve B on C and then you should solve A on B that is the most logical thing to do. So, let us see how goal stack planning works just to illustrate the algorithm when it happens to choose the correct order.

So, it pushes the goal set, the goal set is on A, B and on B, C and we are assuming that the order is correct and it pushes the two goals on A, B and on B, C in that order, remember that this means that when we pop an element the goal on B, C will come out first.

So, goal stack planning will focus on solving that goal on B, C it will ignore the other goal meanwhile and simply wait till it has found some plan and to solve this goal on B, C.

So, this is the first phase when we have pushed the initial goal and individual goals on to the stack and then we pop the first thing on the stack, remember that our stack is growing in a downward direction here normally of course, we draw stacks going upward but it is easier for me to draw it like this.

So, the stack is growing in the downward direction and the top of the stack is this element on B, C and that is what we have pop and we check whether it is true in the given state. The given state is this one that A is on the table, B is on the table and C is on the table and on B, C is not true. So, it fix a relevant action what is the relevant action pop on B, C in our world it is a fully one action exist which is that you can to stack B on to C.

So, we push this action stack B on to C on to the stack then we push its preconditions which are that you must be holding B and there must be nothing else on top of C which means C

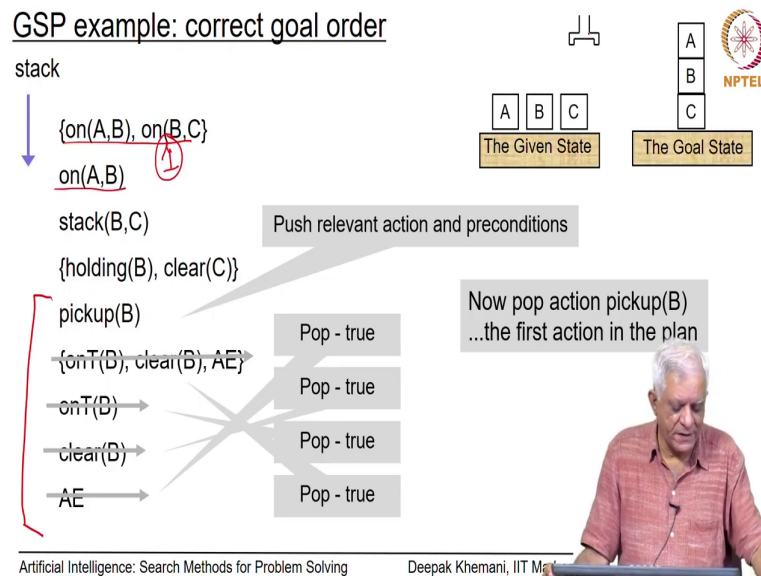
must be clear and then this two individual goals in some order. We have chosen the order holding B and clear C and this is a stage where you have pushed in an action and its preconditions on to the stack and that is been done to solve the goal that we just popped from the stack.

So, the goal is gone from the stack and the actions and the preconditions remain. So, now, we again go back to popping. So, first we pop this element as the top of the stack clear C it is true. So, we do not have to do anything about it.

Then we pop holding B, holding B is not true now remember that when individual goal is popped and it is not true in the given state you want to push an action to make that goal true. Now holding B can be done in two ways you can either unstack B from something or you can pickup B from the table.

And let us since we have working the non - determinants action and in as human beings we can also see that what is given in the start state the pickup B action is applicable, but unstack B from something is not applicable. So, somehow we have chosen this action pickup B and so once we have chosen this action we will now push it is relevant this is the relevant action and we will push its preconditions also on to the stack.

(Refer Slide Time: 06:40)



So, I have deleted the elements which have been popped from the stack and this is what the stack looks like we had two sub goals to start with compound goal, only one remains that will be address after we have finished solving the first one which is on B, C and we are looking for ways to solve that and we algorithm figured out that what you have to do is to stack B on to C to achieve that goal and then it figured out that.

We are able to stack B on C you must be holding B and to hold B it chose the action pickup B and this is where we again take up the action. And we push the relevant actions and the preconditions for pickup B, what are the actions preconditions for pickup B that it must be on the table because that is how the action is defined it must be clear there must be nothing else on top of B and the arm must be empty.

And then we push in the three individual goals one by one in this same order I have put them, but it does not have to be in the same order, may be you can device an algorithm slightly more cleverly which in the general case will work better, but it not so easy to do so as if you try it out yourself.

So, having pushed so again we have pushed an action and its preconditions on to the stack and then we start popping, first we stack the last element the top of the stack arm empty it is true in the given state which is still the start state then we pop clear B which is also true in the given state when we pop on table B which is also true in the given state and then we pop the compound state compound goal that is also true.

So, all the three preconditions for the action through so, once they have been popped away and out of the way we can now pop the action pick up B and this will become the first action in our plan.

(Refer Slide Time: 08:44)

GSP example: correct goal order


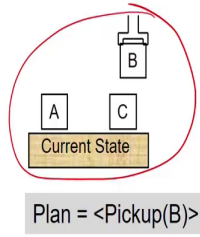
stack

↓ {on(A,B), on(B,C)}

on(A,B)

→ ~~stack(B,C)~~


~~{holding(B), clear(C)}~~



Pop compound action - true

Next pop action stack(B,C)

Plan = <Pickup(B)>, Stack(B,C)>



Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us do that. This is a situation now, observe that the current state has changed. The current state is now where you have picked up B and you can see that this action pickup B is part of the plan already and the new state is that you are holding B.

Now, the preconditions required for the stack action that we originally thought of the original relevant action the last action of a plan in this case, not the last action the last action for achieving the sub goal on the C in the plan.

The preconditions for that way holding B and clear C essentially pop the compound action as we can see in the current state you are holding B and there is nothing on top of C. So, now, we can pop the second action.

So, plan which is stack B on C and add it to the plan. So, the plan as you can see now is constructed in a forward manner, the first action is that you pick up B which is true in the initial state then you are holding B and the second action is stack B on to C.

So, that is also possible and so we have achieved the first sub goal that we wanted to achieve which is that you want to make on B, C true. So, at the end of these two actions on B, C will become true and this remember is the correct goal order. So, everything is going smoothly here the algorithm will now shifts his attention to this other goal which is on A, B.

(Refer Slide Time: 10:25)

GSP example: correct goal order

stack

{on(A,B), on(B,C)}

↓

~~on(A,B)~~ → Pop – not true

Stack(A,B)

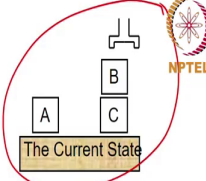

{holding(A), clear(B)}

✓ holding(A) → ✓ clear(B) → Push relevant action and preconditions

Plan = <Pickup(B)>, Stack(B,C)>

The Current State

B is clear,
GSP will next find the action Pickup(A) to make holding(A) true
This will be the third action, and Stack(A,B) the last action
Final plan = <Pickup(B)>, Stack(B,C), Pickup(A), Stack(A,B)>

So, it will pop on A, B from. So, this is how the stack looks like after you have popped the second action stack B, C. The only one goal remains which is on A, B and the algorithm now proceeds to address that it pops that goal from the stack it is not true.

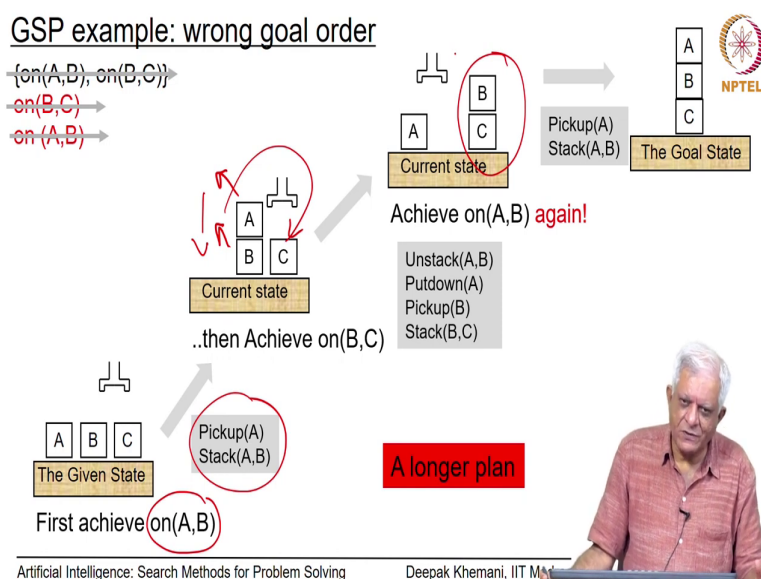
So, it will push the relevant action and its preconditions the relevant action for achieving on A, B is stack A on to B, the preconditions for stack A on to B are holding A and clear B and then the individual goals are holding A and clear B.

Now remember that the given state is will progress to in this world B is on C the arm is empty, B is clear, A is clear essentially. So, B is clear and holding A is not true. So, one of the things is true and the other one is not true, we will not go through the details again you can work it out yourself, but you will just make the observation that B is clear. So, when it is popped.

So, when you pop B you do not have to do anything from it, but when you pop holding A then you will have to add the action pickup A to make it true essentially that. So, it will do; it will do that pickup A is applicable in the given state as you can see the arm is empty and A is on the table and there is nothing on top of A. So, pickup A would we apply it and you would progress to a state where you are holding A.

And then it will find the last action which is already there in the stack as you can see which is to say stack A, B and once holding A has become true and clear. B is already true as you can see from the diagram it can apply the last action and that will be the last action and it would construct the plan which is a valid plan for doing this which is first pick up B stack it on to C then pickup A and stack it on to B essentially. So, this was the situation where it chose the right order.

(Refer Slide Time: 12:31)



What happens if we choose the wrong order? So, the wrong order is depicted here on the top it is saying that I have these two goals on A, B and on B, C to be achieved and it is saying first achieve on A, B then achieve on B, C. So, as an intelligent human being of course, you will realize that this is not the best way of doing things, but we have said that you know we are putting them in some order and we yet do not have a heuristic of how to do this thing.

And so, let us see what happens when it chooses a wrong order. So, what will this algorithm now do it will pop the first goal which is on A, B and it will focus on achieving the sub goal which is on A, B.

If you go through the details which we will not go through here, you will see that the two actions are needed for that which is pickup A and stack A on B. Goal stack algorithm will construct these two actions in this particular order first you pickup A and then you stack A on

B and I will strongly encourage you to work this out yourself when you pop this goal on A, B you would push in the action stack A on to B, but for stacking A on to B you would need holding A.

And for holding A you would have to push the action pickup A and pickup A would be applicable in the given state so that would be the first action and stack A on to B would be the next action and this is the world that you would reach.

Then it will go to the second goal like in the previous case having solve the first sub goal it is now saying let us achieve on B, C and to achieve on B, C again you should go through the process that you will have to first push in the action stack B on to C and then the preconditions and so on and so forth.

And you will see that the sequence of action that achieves on B in the given state is that you first unstack A from B. Then you put it down on to the table, then you pickup B and put it on to stack it on to C essentially. So, these four actions in this particular order we will achieve the situation where the second goal on B, C is true. Now notice that first you achieved on A, B here you found a sequence of actions and then you reach the state where on A, B was true.

Then you achieve the second goal on B, C indeed on B, C has been achieved here, but the original goal which was that you must achieve both on A, B and on B, C has not been achieved and that is the reason why we push in the compound goal as well. So, now, when it pops the compound goal it will see that it is not true and it will then go on to achieve on A, B again. So, it will pickup A and stack A on to B.

And you can see that it is not the most optimal plan it has a plan of eight steps whereas, you could have produced the plan which was of four steps which was done when we chose a right order of doing things.

(Refer Slide Time: 15:42)

Goal ordering matters

We saw in the tiny example that goal order matters
In this domain the wrong goal order resulted in a longer plan
In other domains a wrong goal order may lead to a dead-end

- for example during cooking
- requiring backtracking to a different goal order

So, does it mean that choosing a correct goal order
will always result in an optimal plan?

No!

Certain problems have *non-serializable subgoals*!

- there is no optimal order for solving them
- solving one may *undo* a previously achieved goal
- for example in the 8-puzzle and the Rubik's cube
- Gerald Sussman gave us a tiny example!

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Madras



So, the lesson to be learnt is that goal ordering matters it is indeed important we saw that in a tiny example that this is true and in this domain the wrong goal order that we just finished only resulted in a longer plan, but in other domains a wrong goal order may lead to a dead end. For example, if you are doing cooking then you have to do certain things in certain order you know. So, for example, you first boil the water and then you add tea, if you want to make tea and typically you do not do it in a reverse order.

So, I know people who first add tea leaves and then boil the water, but that is not the recommended way of making tea and you may reach the dead end of something. So, you have chopped something before peeling for example, if you are making some vegetable peeling is to be done first and then chopping is to be done later. But if you do them in the wrong order you can imagine that you have to peel every small piece of vegetable that you have chopped.

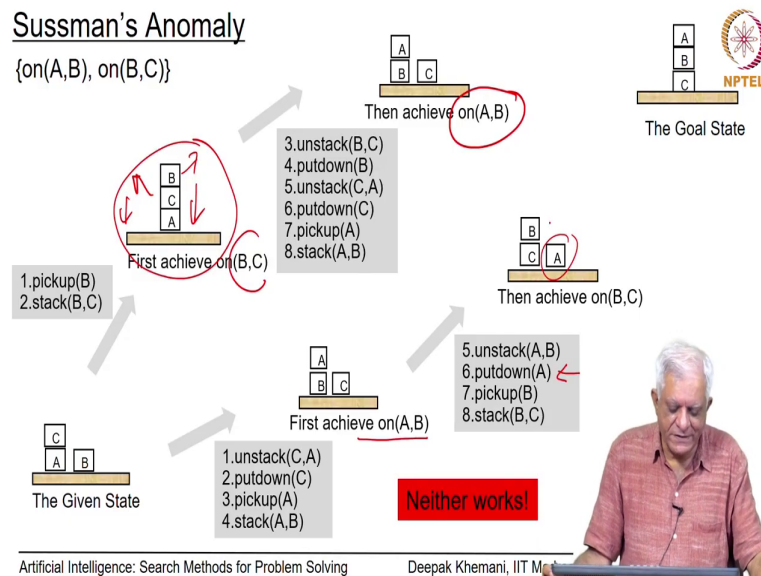
So, essentially it may lead to a dead end and the algorithm may actually want to backtrack to a different place and try a different goal order, but the question you want to ask now is that, does choosing a correct goal order always result in an optimal plan?

In this example that we saw that was true, but unfortunately that is not the case in general we have observed the planning is a hard problem that it was shown by (Refer Time: 17:23) and Gupta that long time ago in the 90s that it is in piece space complete which means basically it is a hard problem to solve so do not expect a free lunch.

Now there are certain problems which have non-serializable sub goals; that means, there is no optimal order of solving them. If you solve sub goal 1 and then you solve sub goal 2 then sub goal 2 may undo sub goal 1 we saw this in the wrong order that we saw just now, but if you have been solving the Rubik's cube or the 8 puzzle you would know that if you achieve some partial goal the top layer or something like that if you do the next layer you have to undo what you had done earlier.

But a very nice example was given to us by the scientist Gerald Sussman who is still working at MIT in Boston and this was something that he gave in the 70s.

(Refer Slide Time: 18:19)



So, he is been a long time worker in AI. So, let us look at this problem which is called Sussman's Anomaly, it is a very simple problem you have the given state where C is on A and B is on the table. And you want to achieve the same goal say that we have been talking about which is A on B and B on C. Now clearly there are two ways of solving this that either you do on A, B first or we do on B, C first.

In the previous example when all A, B, C were on the table we saw that there was one correct order of doing things which produce the shortest plan what Sussman's showed was that with this small problem which is called as Sussman's Anomaly.

There is no correct order that we cannot find the optimal plan by choosing either sub goal as the first goal to achieve. So, let us look at this, supposing we say that let us first achieve on B,

C and if you look at the given state on B, C can be achieved quite simply by picking up B and stacking it on to C.

So, we have achieved the first sub goal on B, C then we achieve the second sub goal which is on A, B. So, given this state you want to achieve sorry on A, B here. So, there is the sequence of these six actions you first unstack B, then you put it down, then you unstack C and then you put it down and then you pickup A and stack it on to B, you have put down B on the table. So, this is a state that you end up the.

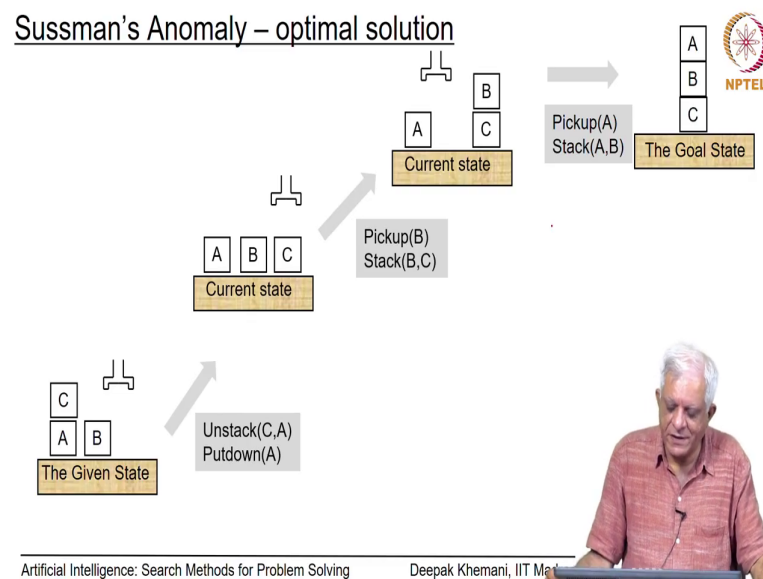
So, we can see that first achieving on B, C and then achieving on A, B did not work essentially. The interesting thing about this particular problem given to us by Sussman is that the other order in which you first achieve on A, B also does not work. So, let us go through that we can see that if you want to achieve on A, B, you have to first unstack C from A put it down then pickup A and put it on to B and you will achieve the state where A is on B and C is on the table.

Then you go to the second goal which is to achieve on B, C now you would unstack A from B put it down pickup B and stack it on to C number we have put down A in this step here and therefore, A is on the table and the goal has not been achieved.

So, you can see that in this problem neither goal neither order of choosing the sub goal works and in fact, this shows that this kind of problems are non-serializable, you cannot say that I will break it down into sub goals one by one and solve them independently.

What you will do of course, in the next session is look at an algorithm will allow us to solve such problems where there is as the community says there is goal interaction that the two goals that you are trying to achieve two or more goals that you are trying to achieve may interact with each other and then we need more sophisticated algorithm.

(Refer Slide Time: 21:25)



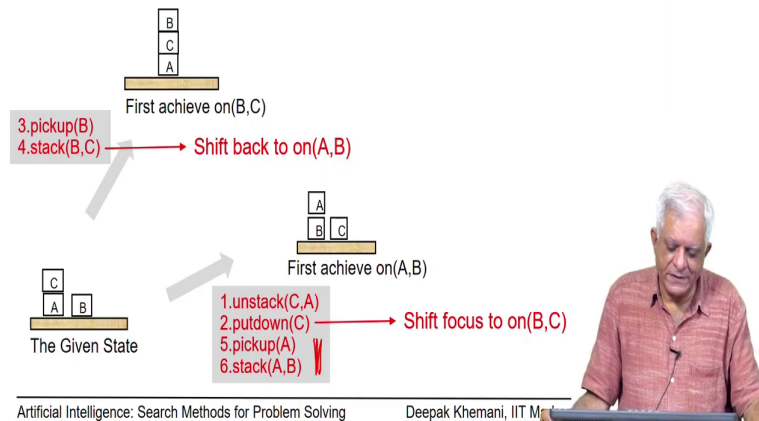
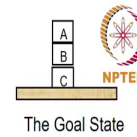
What is a solution that we are looking for in Sussman's Anomaly it is not too hard to find you can see that first you unstack C from A and put it down you will reach the state that was the starting point for us in the previous example and from this starting point you know that there is a correct order of doing things which is that you first pickup B and stack it on to C and then you pickup A and stack it on to B and we have a optimal plan which is of six steps.

Unfortunately goal stack planning will not find this plan because goal stack planning is committed to linear planning it is committed to solving the goals one by one in a linear order and we will need to do something different. So, what is at that we would need to do? Let us just get a flavor of it now and in the next session we will look at a different approach to planning which will enable us to do that.

(Refer Slide Time: 22:24)

Sussman's Anomaly – non-serializable subgoals

{on(A,B), on(B,C)}
on(B,C)
on(A,B)



So, let say that we have these two sub goals to solve on B, C or on A, B. We have seen just now that either way if you take up a sub goal and pursue it lengthily let us solve and then only revert to the other goal you will not find the optimal kind of course, you will find the plan which is the longer plan, but you will not find the optimal plan. So, to do the optimal plan it is basically says that you must in some ways switch between the two goal essentially. So, just to illustrate that.

So, let us say you start by solving the first goal on A, B to be true and to achieve that what you do is that you first do unstack C from A and you put down C. This is an example that we had seen in that this is how you could achieve the first that goal, but before having achieve the first sub goal you shift your attention. If you could somehow shift your attention that I will achieve I will come back to on A, B later let me focus on B, C first.

So, shift your attention to on B, C in some sense shift to the other goal to the other branch that goal stack planning would have pursued achieve that goal which is just to pickup B and stack it on to B and then you come back to on A, B. Then you can see that when you come back to on A, B this is the remaining part of the plan that we had embarked upon, but kind of cut off mid way we come back to this plan and in this six steps we have this nice optimal solution.

Goal stack planning unfortunately will not be able to do it and in the next session we will look at an algorithm which is part of what people call as non-linear planning or partial order planning or plan space planning and we will look at that algorithm in the next class which will be on non-linear planning. So, see you in the next session.

(Refer Slide Time: 24:30)



Next

Non-linear planning

