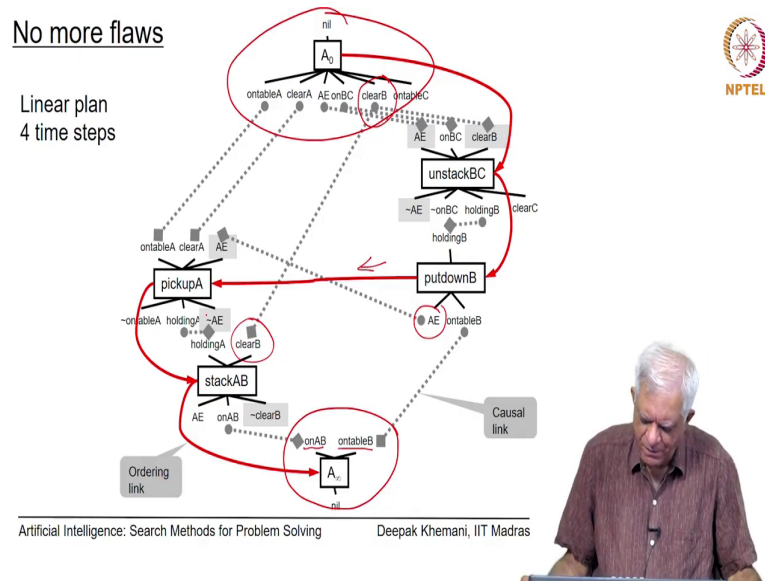


Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 07 and 10
A First Course in Artificial Intelligence
Lecture – 65
Multi – Armed Robots

(Refer Slide Time: 00:14)



Welcome back, let us resume our study of this new algorithm or new approach to planning which operates in the plan space and the only objects we reason with in it are plans.

And of course, actions have preconditions and postconditions which come from the predicates and we saw that every plan space planning algorithm begins with an action A_0 which has no preconditions and it produces this stack state predicates as its effects. And every

plan a partial plan has one more action which is A infinity or the last action and this consumes the goal state and has no effects.

Now we had said that there is a notion of a flaw in a plan and they were two kinds of flaws that we spoke about one is open goals. So, for example, when we start with A 0 and A infinity, the A infinity action has two open conditions onAB and ontableB. And what the planer needs to do? Is to resolve flaws or remove flaws.

And the way to remove the flaw which is an open condition is twofold one is that either you find an existing action which provides the support or the causal link needed for that we saw that as an example. For clearB we found that being produced by A 0 action and we just established a causal link with between the two.

If an existing action cannot provide this causal link then we add a new action and we did this in the case of both onAB and ontableB. For onAB we introduced in action stack AB and for ontableB we introduced the action putdownB.

Obviously, you can see that once you introduce this new actions they will have their own precondition. So, you would have new open goals. So, the planning algorithm would still have more work to do. The other kind of flaw that a partial plan may have is that some action may be threatening to undo the precondition of some other action.

So, we saw various examples there and in this particular example that we were looking at we saw three flaws and because of those three flaws we ended up introducing for example, these kind of ordering links which said that you must to putdownB before you pickupA because you wanted arm empty to be empty essentially and otherwise that condition was not made.

Then eventually when we have a partial plan which has no flaws then we call that a solution plan. And in this particular case that we saw of one arm robot working for us, there were only linear plans possible and it is not a surprise that for this problem that we had to solve there was a linear plan of four actions unstackBC putdownB pick up a and stack it on to B.

Now, we had said at the end of the last session that what if they were multiple arms or multiple agents for that matter? In which case many actions could happen in parallel.

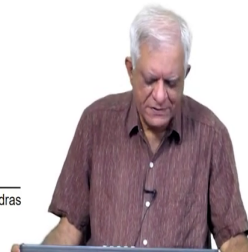
(Refer Slide Time: 04:02)



Versatile Multi Armed Robots doing

Plan Space Planning
Partial Order Planning
Non-linear Planning
Least Commitment Planning

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



So, we had post this problem in the last time that how can we reason or how can we model multi armed robots and we will see that we are talking about quite versatile robots. To do what we have been so, far calling as plan space planning, but its also called partial order planning because the plan that it produces is a partial order. And as you will see that if you have two arms you can indeed get a partial order which is not necessarily a linear order.

Its also called non-linear planning because you don't grow the plan from one end or the other you can add actions at any layer because depends on which flaws that you are trying to

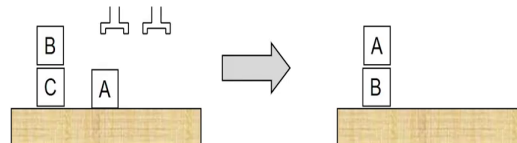
resolve you would add an action to or some other condition may be a causal link or maybe an ordering link to the plan.

It's also called least commitment planning because what it does is that it identifies actions that have to go into the plan, but it waits till it knows something more before it decides the order between actions essentially.

So, in that sense it does not commit itself to order as it finds the actions and we have seen that in the algorithm that we have seen so far especially forward state space search and backward state space search. The moment they pick an action they know where to place it in the plan, but plan space planning is more flexible on this matter and it waits for a need to induce some order onto the graph.

(Refer Slide Time: 05:36)

A two armed robot?



Given that we had a one armed robot, it was only possible to have a linear sequential plan, of 4 time steps

What if we had a two armed robot which could do actions in parallel? How long would the plan take to execute?

Exercise: Modify the STRIPS operators to cater to the domain with a two armed robot.

Will your solution extend easily to multiple arms?



So, this is a problem that we had looked at as a detail example and we had said that there is a single arm robot which is solving this. And we had asked the question as to what would happen if they were two arms for this robot.

So, for the single arm robot it was only possible to construct the sequential plan. If you remember the plan was pick up and stack B from C put it down pick up A and stack it on to B, but what if there were two robots as there is a case that we are investigating now.

So, now, we have a situation where we have two robots or we have to arm one robot with two arms as far as we are concerned the arm is the effective part of the robot that we are interested in then; what would should be the plan?.

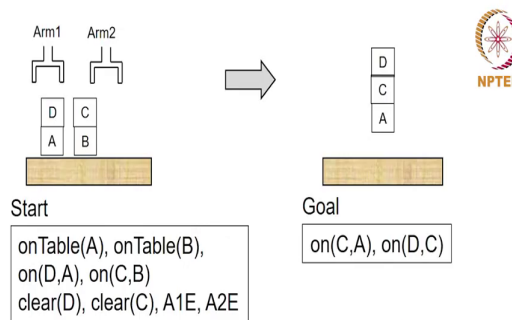
How do you model the domain first of all and how long would the plan take to execute? So, there are two kind of metrics that we use in evaluating plans. One is the number of actions in the plan and the other is the time taken to execute the plan which in the planning community we call as makespan.

Now, the makespan is equal to the number of actions in the plan if the plan is a linear plan because actions have to be done one after the other. But if actions can be done in parallel then the makespan can often be less than the number of actions in the plan and this is what we want to investigate now; essentially that if we had two armed robot here solving this problem; how fast could it solve it?

So, this is an exercise which I had left you with I hope you have done some work on this. That is how do you modify the strips operators to cater to a domain with two armed robots? And also what if he had a three armed robot or a four arm robot multiple armed robots or something; would your solution extend to that situation as well?

(Refer Slide Time: 07:35)

A Two Armed Robot



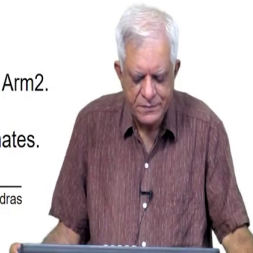
Exercise

Modify the STRIPS operators to work with two arms Arm1 and Arm2.

Show the plan when the Plan Space Planning algorithm terminates.

Artificial Intelligence: Search Methods for Problem Solving

Deepak Khemani, IIT Madras



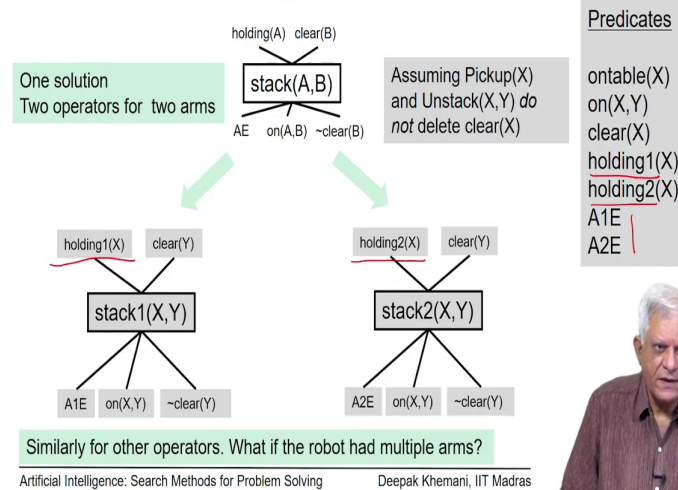
I had also introduced a new problem which was part of an exam that I had given to students here in IIT Madras. And in this problem we have on the left hand side the start state where D is on A and C is on B and we have two arms.

And we have to reduce the stack which is D should be on C and C should be on A we do not particularly care whether A is on the table or whether A is on B for example, its only block left and we don't even care whether B is on D for that matter.

The only thing we care about is that that C is on A and D is on C essentially. And how would the two armed robot how would you model the operators for a two arm robot? And we would like to see what the solution plan looks like essentially.

(Refer Slide Time: 08:28)

Modified STRIPS Planning Domain: Stack(X,Y)



We had also very quickly said that the simplest way to do this is to introduce two actions for each of the arms. So, instead of saying for example, that we have stack A, B and instead of saying that you are holding X or you are holding or we specify as to which arm is holding X.

So, you could modify the predicate to say that holding1 and holding2 are two predicates holding1 means that arm 1 is holding X holding2 means at arm 2 is holding X and likewise for the arm empty conditions. And so, that would be a gain of a very trivial solution where you just say two operators for two arms and what you have is an operator called stack one and another operator called stack2.

But they are identical in other ways except that the predicates that they use are slightly different. Stack1 as a precondition stack2 as a precondition that holding 2 X and stack1 has a precondition that holding 1 X essentially.

Now obviously, this will work for a two armed robot, but it would be difficult to replicate with multi armed robots essentially. Especially if you allow more flexible actions that we will see in a moment ok. So, again the same question what if the robot had multiple arms?

(Refer Slide Time: 09:55)

To delete or not to delete clear(X)

AE on(A,B) clear(A)

unstack(A,B)

~~~AE ~on(A,B) holding(A) clear(B) ~clear(A)~~

AE on(A,B) clear(A)

**unstack(A,B)**

~~~AE ~on(A,B) holding(A) clear(B)~~

holding(A) clear(B)

stack(A,B)

AE on(A,B) ~clear(B) clear(A)

holding(A) clear(B)

stack(A,B)

AE on(A,B) ~clear(B)


An alternative... after unstack(A,B) should clear(A) be deleted?


The operators presented so far.

In this one armed robot domain the *only* thing that can happen is stack(A,?X) or putdown(A)

Predicates

- ontable(X)
- on(X,Y)
- clear(X)
- holding(X)
- AE





Now, one question that we have kind of not addressed in detail is that when you pick up an object X for example, or when you unstack X on something else then should you delete clear X or should you let it be essentially?

So, the operators that we have introduced so far did not do anything to clear X and for example, when you unstack A from B, A was clear, but it does not show anywhere in the effects of the action which means that we are not deleting it essentially.

So, it's a question of semantics that if you are holding that object A can you say that it is clear or can you say that it's not clear essentially? So, obviously, if you take a more physical perspective then it would depend whether how the robot is holding the object.

If it is holding from the top; then clearly that object will not be clear, but if it is somehow holding it from the side, then you can imagine that it's clear essentially and as you will see in practice or in principle we can imagine stacking another block onto that essentially.

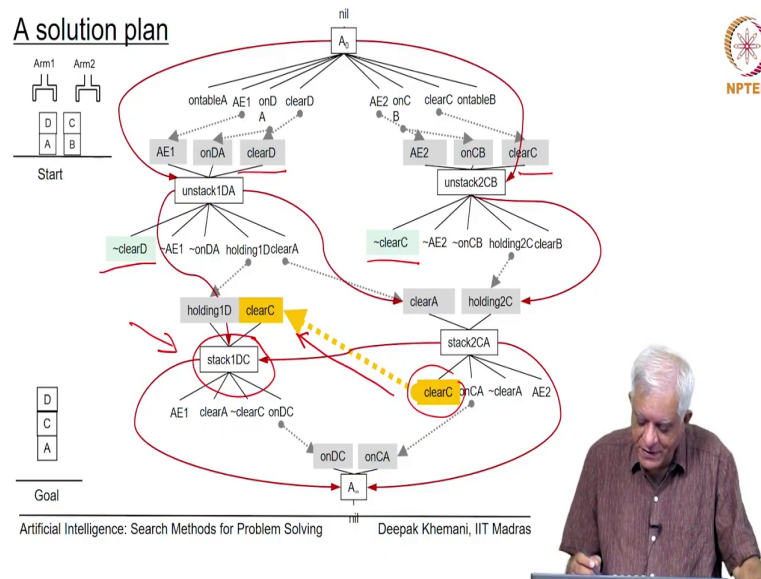
But the algorithms that we had written we had not deleted clear B and deleted clear A from this thing. And therefore, we had to do nothing for that for in the case of stacking that object.

But now supposing we do delete this predicate. So, for example, when we unstack A from B, we say that A is no longer clear as shown here in red. So, blue so in blue the it's a precondition and in red it's the post condition that we are saying is no longer clear.

So, if it's no longer clear in this one armed robot case it does not really matter whether you delete this clear or not because the only thing you can do with an object. If you are holding it you can either put it down or you can stack it on to another object. So, if you were to stack A on to B for some reason again then we would have to add clear A again.

So, if you unstack it you have to delete clear A if you stack it you have to add clear A. And in a one arm robot it does not really matter because it simply is going to not change anything and you might as well say that this you know let us not delete clear A because it is less work in deleting it and adding it again essentially.

(Refer Slide Time: 12:43)



Now, look at this plan in this plan we have assumed that we have going to add this delete action. So, as you can see when you unstack D from A D had to be clear and then we have added as an effect not clear D or in other words we have deleted clear D from its effect. The same thing for clear c on the right hand side and we have deleted clear essentially.

Now, because of the fact that we are deleting clear C, action stack D 1 to C needs clear C as you can see and we have to wait for stack C on A to finish because that would add clear C and that would provide this causal link here for this action.

So, this is a partial plan that is the plan without flaws there are two actions; one is to do with unstacking D from A and afterwards tacking D 1 to C and the other right hade side is to do

with unstacking C from B and then stacking it on to A essentially and this is a partial plan which has no flaws.

The maroon coloured links are the ordering links the dotted arrows are the causal links as we have seen before. So, in this plan there are no flaws and so, its a solution plan. I encourage you to take this up and work on it a little bit and see that it is indeed the plan that you will generate.

(Refer Slide Time: 14:33)

Multi-armed robot domain 1

Since A is not clear nothing can be stacked on A


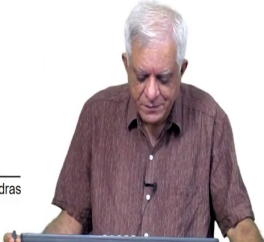
Predicates

- ontable(X)
- on(X,Y)
- clear(X)
- holding(N,X)
- AE(N)

The diagram illustrates the causal and ordering links for four actions in a multi-armed robot domain:

- pickup(N,A):**
 - Preconditions: ontable(A), clear(A), AE(N)
 - Effects: ~ontable(A), holding(N,A), ~AE
 - Ordering link: ~clear(A) (circled in maroon)
- unstack(N,A,C):**
 - Preconditions: AE(N), on(A,C), clear(A)
 - Effects: ~AE(N), ~on(A,C), holding(N,A), clear(C)
 - Ordering link: ~clear(A) (circled in maroon)
- stack(N,A,B):**
 - Preconditions: holding(N,A), clear(B)
 - Effects: AE(N), on(A,B), ~clear(B), clear(A)
- putdown(N,A):**
 - Preconditions: holding(N,A)
 - Effects: AE(N), ontable(A), clear(A)

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

Now, we are also talked about multi armed robots. So, let us look at this question again. If we want to have multi armed robots instead of saying there are two separate actions we can introduce another parameter in the predicates corresponding predicates which is as you can see holding an arm empty and introduce an arm number or something like that. So, we have

arm 1 arm 2 arm 3 you can have any number of arms and we can simply extend the whole thing to that.

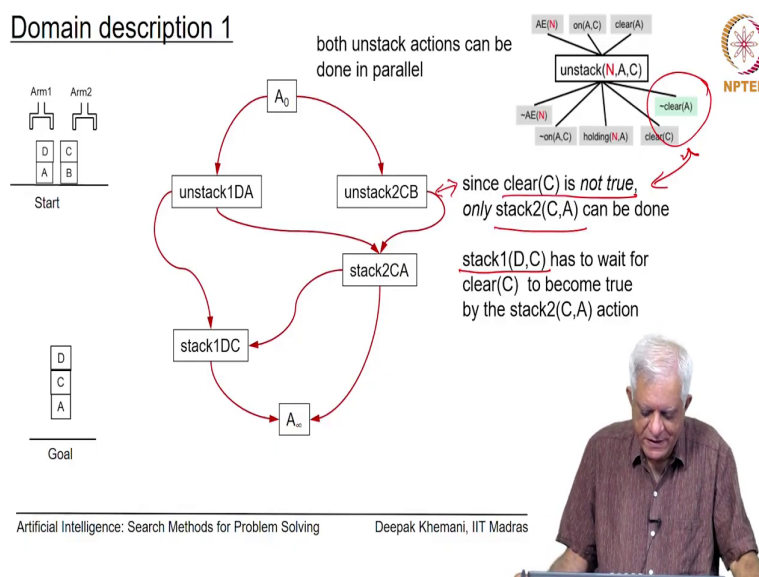
So, this will of course, carry forward to the predicates that are used because now we have revised. Predicate we have seen here, but if you also carry forward to the actions that we just need to define the arm number or arm name as the case may be as a parameter for pickup unstack put down and stack essentially. So, we are simply saying that this arm is picking up or this arm is stacking something onto that and so, on and so, forth.

And that is a scene that we have used when we produce this partial plan that as a solution plan we have deleting this that is not the scene we have used, but this other feature that I wanted to talk about is that you are adding this you are deleting clear whenever you are unstacking or picking up. And that is a feature that we had used in the in the partial plan that we produce.

In this in this example partial plan we had still assume that there are two actions one is unstack 1 and the other is unstacked 2, but we can easily modify it to adding 1 and 2 as a parameter instead. But the point that we were talking about also was that if you are going to delete the fact that their object you have picked up is clear, then you can actually no longer stack anything on to that.

And which is why this action of stack 1D action of stacking D onto C had to wait before C became clear and C became clear after you had stacked c onto a on the right hand side and that is why we had this causal link.

(Refer Slide Time: 16:53)



So, how can we execute these plans? So, in this description and by domain description 1 I mean this description where you are deleting clear A or clear X as the case may be from the description of the effects. So, in that description initially of course, we have two actions. So, we are just trying to see how will that plan B executed the plan that we have just seen here.

I have deleted the preconditions and effects and causal links and everything and I have only kept the four actions that are part of the plan and I have only kept the ordering links essentially.

Now, given that plan how can it be executed and we are interested in parallel execution to begin with. So, the first two actions which were unstack actions arm 1 is going to unstack D

from A and arm 2 for unstack C for B they can clearly be done in parallel irrespective of whether that deleting of clear this happening or not. So, there is no controversy here.

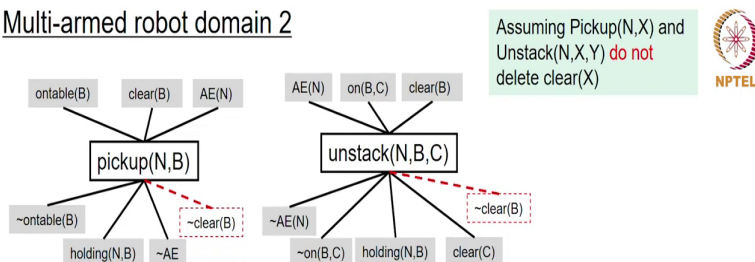
But if you are using an action format in which we are deleting this thing that we are picking up. Then at the next stage you can see that because you have unstacked C from A it is a case that clear C will not be true because it is deleted as depicted in this action.

So, you cannot put anything on to C you can do stack C onto A clearly because that that only requires this condition that A be clear, but you cannot do stack D on to C at this point because of the fact that clear C is not true and that is not true because of the fact that we have deleted it we have interviewed we have added it to the delete list of that action. So, the only action you can do is stack C on to A and that is consistent with the ordering links that we have and in the next step.

So, now, we have finished two time steps. In the first time step is unstack both the blocks in the seconds time step is stack C onto A and into the in the third time step we can now add stack D on to C essentially. So, let us look at the other possibility; the possibility is when clear a is not deleted essentially.

(Refer Slide Time: 19:50)

Multi-armed robot domain 2



Assuming Pickup(N,X) and Unstack(N,X,Y) do not delete clear(X)



Since holding(N,B) and clear(B) are both true the action $stack(M,A,B)$ is applicable

- robot can hold a stack of blocks!
- also putdown(N,B) and stack(M,A,B) can be concurrent!
- many arms can create a stack in one time step!

Exercise: Modify the domain so that an arm can hold only 2 blocks



So, the action schemas look like this that we are not deleting that object that we are picking up we are not deleting clear B of the public that we are picking up. So, clearly B could have been there in these two actions, but in this variation which we call as the domain to description we do not delete clear C.

What is the impact of that? The impact of this is that; in this case if arm N is holding B either because of the pickup action or because of the unstack action and clear B is true. Why is it true? Because we have not deleted it its not part of the delete list of the action.

If there is another arm M which is holding B which is holding A, it can stack A onto B. So, what is this situation? Arm 1 arm 1 or arm N is holding B then arm M can stack A onto B. So, we can stack one block on top of another block.

Not only can be stack one block we can imagine in this situation that on arm 1 which is arm N in this case you had B first then you stack A onto B then you can stack C onto A then you can stack D on to C and so, on.

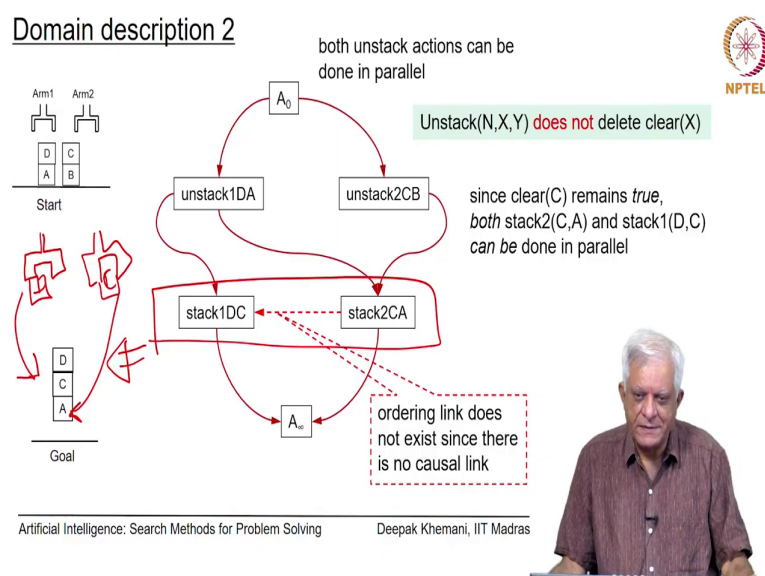
You can hold the entire stack of blocks in one arm essentially because difficult to imagine, but you can see that that is possible. Also if you are putting down B you are holding B in arm N and you want to execute putdownB action and you also want to achieve a onto b then this can be done concurrently.

Now, it is a again it kind of you have to extend your imagination a little bit to imagine that both the arms are putting down both the blocks that they are holding in such a way that one block ends up being on top of another block. So, that is why we called over multi arm robot as versatile robots with this kind of condition they can do all these kinds of things.

In fact, not just two blocks you can if you had many arms holding let us say N arms holding N blocks, then they could create a stack of N blocks in one time step essentially you know clearly that is mind boggling.

As a small exercise I would ask you to redefine the domain so, that a robot can only hold a stack of two blocks. So, it can hold B for example, and you can stack A onto B, but you cannot stack a third one on to that. So, its a small exercise that you should take up to see how that can be implemented.

(Refer Slide Time: 22:52)



But at the moment we have a very versatile robot and with this domain description how does the plan look like ok. So, as before both the initial actions which are to unstack the two blocks D from A and C from B they can be done in parallel.

So, there is there is no change here because there was no interaction between those two actions, but now we are assuming that when you unstack X from Y, it does not delete clear X; which means in these two cases at the end of these two actions D is going to remain clear because that is a precondition for unstacking B from A and C is also going to remain clear because that was a precondition for unstacking C from B and we have said that this unstack action does not delete that particular clear goal from the this thing.

So, now, since C remains clear or C clear C remains true both the actions that we were talking about they can be done in parallel. So, we can stack D on to C and we can stack C onto A. So,

just imagine that these two actions stacking D on to C and stacking C onto A they are happening at the same time. So, both the robot sorry this its not from here, but its from the robot which is handling it.

So, we have two robot arms one is holding D let me draw this a bit neatly. So, this arm is holding D and it is going to stack it on to C and this other arm is going to be holding C and it could stack it on to A.

So, you can imagine that both these actions these two arms have one stack C and D respectively and they are going to stack it simultaneously. So, D will be stack on the C and A will be stack C will be stack onto A and this is what it means by saying that both these actions are happening at the same time essentially.

Now, it also this is possible because in this new domain this stack1DC did not have to wait for C to become clear and therefore, this causal link that was there in the in domain 1 no longer exists essentially.

Ok so this is what we have seen so, far that the main key takeaway is that in plan space planning you can have plans which are partial orders that is why its also called partial order planning also that ordering of the plans is left to as late as possible. So, its also called least commitment planning.

Will, come back to this description in the next session and wind up the section on plan space planning. So, see you next time when we will look at plan space planning and we will also look at where are the origins of plan space planning they in fact, go quite far back into the history of AI ok.

So, see you the next time.