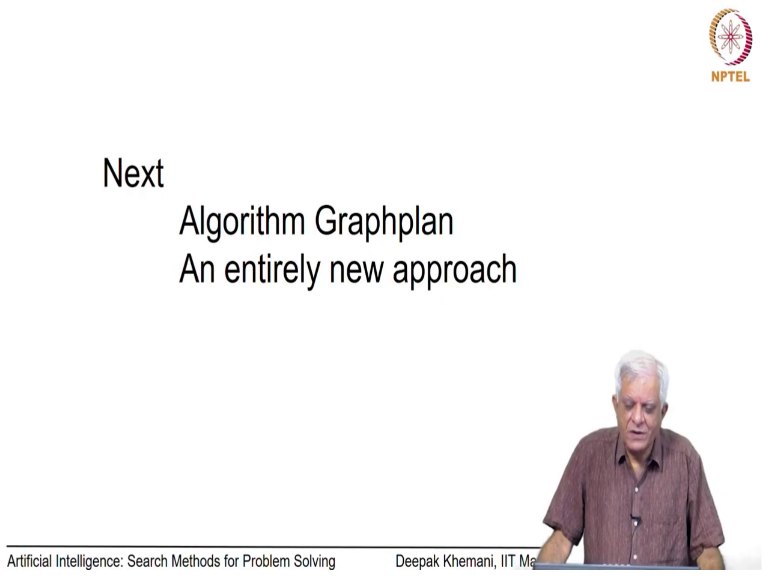
# Artificial Intelligence: Search Methods for Problem Solving

**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 – 67**
**The Planning Graph**

(Refer Slide Time: 00:14)



Next

Algorithm Graphplan

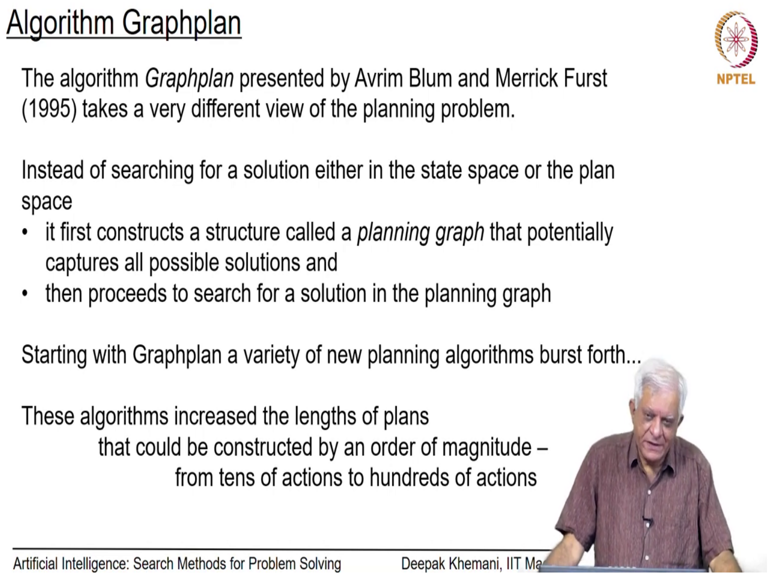
An entirely new approach

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

Welcome back. So, this is the last topic in Planning that we will consider. Actually, planning has become a vast subject in itself and one can offer a full course in planning and many people do. But since we are seeing it from the perspective of search methods in Ai, we are essentially trying to see what are the different approaches they use for constructing plans using search methods essentially.

So, what we want to now consider is this algorithm called graph plan which kind of revolutionize the whole world of planning because using techniques that kind of evolved around that time, people could construct much longer plans than earlier and that was gave a kind of a boost to research in planning.

(Refer Slide Time: 01:09)



## Algorithm Graphplan

The algorithm *Graphplan* presented by Avrim Blum and Merrick Furst (1995) takes a very different view of the planning problem.

Instead of searching for a solution either in the state space or the plan space

- it first constructs a structure called a *planning graph* that potentially captures all possible solutions and
- then proceeds to search for a solution in the planning graph

Starting with Graphplan a variety of new planning algorithms burst forth...

These algorithms increased the lengths of plans that could be constructed by an order of magnitude – from tens of actions to hundreds of actions

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

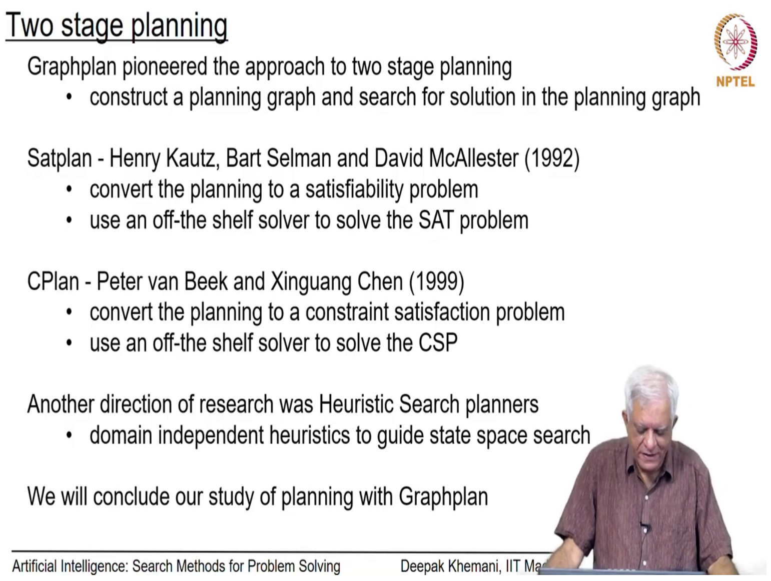
So, let us study this algorithm graph plan, it was presented by Blum and Furst in 1995 and as I said, it takes a very different view of the planning problem. Instead of searching for a solution either in the state space or the plan space, it constructs a structure which it calls which is called a planning graph that potentially captures all possible solutions and then, proceeds to search for a solution in the planning graph.

So, at the outset maybe I should clarify that it is not that it constructs the entire structure which contains all possible solutions. It does so incrementally and at certain stages, when there is a possibility that a plan might have been found, it stops constructing the planning graph and starts the plan extraction process or the search process within the planning graph and we will see this as we go along.

Now, starting with graph plan many new planning algorithms came in the mid nineties and early part of this century and planning as I have said has become richer. In the sense that with these better algorithms people have started addressing richer domains. For example, temporal planning, metric planning, soft constraints, the issues that we considered. But, we will study graph plan only in the context of states planning.

Now, as I said these new algorithms, they increase the lengths of plans that could be found from the or by an order of magnitude. So, earlier you could find plans which are 10, 15, 20, 30 steps long, but now you could find plans of 150 to 200, 250 steps long. Further, similar kind of problems and that is why there is great interest in planning.

(Refer Slide Time: 03:09)



## Two stage planning

Graphplan pioneered the approach to two stage planning
- construct a planning graph and search for solution in the planning graph

Satplan - Henry Kautz, Bart Selman and David McAllester (1992)
- convert the planning to a satisfiability problem
- use an off-the shelf solver to solve the SAT problem

CPlan - Peter van Beek and Xinguang Chen (1999)
- convert the planning to a constraint satisfaction problem
- use an off-the shelf solver to solve the CSP

Another direction of research was Heuristic Search planners
- domain independent heuristics to guide state space search

We will conclude our study of planning with Graphplan

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

Now, graph plan also kind of was the pioneer in this approach to two stage planning, the people had tried earlier also. For example, transforming planning problems into other problems; but it kind of got momentum after graph plan. So, what does graph plan do? Graph plan constructs a planning graph and then, searches for the solution in the planning graph.

So, so, the first stage is kind of construction of the graph and the backwards backward stage is searching for a solution there. There was an algorithm called Satplan which came around that same time 1992, a little bit earlier than graph plan. In fact, work on logic-based approach planning had been going on since the very beginning, till people realize that ah a strips like approaches kind of more simple and efficient.

But nevertheless, the work continued and what a SAT plan does is it converts a planning problem to a satisfiability problem and uses an of then you can use an off the shelf solver to sat problem. This has been our theme in this course throughout that we must have general purpose problem solving approaches and we take specific problems and solve them using those general purpose approaches.

Now, this SAT plan takes that one step further that even though no meaning in independent planning that we are considering is a general purpose approach, it even reduces this to something like a satisfiability problem because there has been a considerable amount of work in developing SAT solvers because SAT solvers are used not only for planning; but for other applications as well.

Unfortunately, we do not have time to go into these approaches in this course; but hopefully at some later point, we would have an opportunity to do. So, C plan was similar it converted the planning problem into what we called as a constraint satisfaction problem.

We are not so far studied constraint satisfaction problems in this course, but we will do a quick study towards the end which would for the sake of completeness. Because constraint satisfiability problems are also in their own right, a very interesting independent approach to general problem solving.

We will study a little bit of it towards the end of this course. Now, it turns out that SAT problems are just a special case of constraint satisfaction problems because both of them deal with having a set of variables, a set of domains for their variables and a set of constraints on some subset of variables.

Now, in the case of CSP, the variables can have any domains For example, classic example would be classroom scheduling because if there are many courses being offered and there are many teachers teaching those courses and many students have registered for those courses and so, constructing a timetable which may include assignment of courses to teachers.

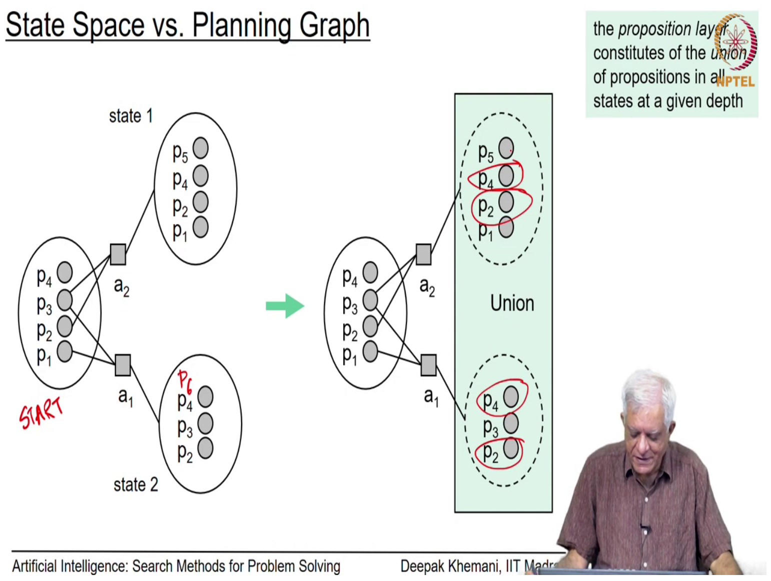
It can be seen as a constraint satisfaction problem, because there are very many variables a teacher, teaches a course; a classroom has a class timetable, courses have slots and so on. SAT is a general is a special case of constraint satisfaction problems in which the variables are all Boolean in nature. They can only take two values; typically, we call them true and false or 0 and 1 and the constraints in SAT problems are expressed as again Boolean constraints.

So, we have we can have you know ‘and or not’ and we have seen sat problems earlier in this course. Another approach which also took off at around that time was the use of domain independent heuristics.

We had not had time to look at it, but we had mentioned it in passing when we were looking at the 8 puzzle for example and the idea of domain independent heuristics is to reduce the planning problem to something called a relaxed problem and which can be solved in let us say polynomial time and use that relaxed problem to estimate a distance to the goal.

So, the heuristics that we are talking about do not come from the domain; they are not given to us by the user, but they are constructed by the algorithm themselves and this is also be the very profitable area of research for people working in planning. We will have our confine our study to graph plan in this course I think and in particular to graph plan with states like actions.

(Refer Slide Time: 08:23)


## State Space vs. Planning Graph

the *proposition layer* constitutes the *union* of propositions in all states at a given depth

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

So, let us see how graph plan is different from state space planning. If this was a problem, then if given that this was a start state which had four propositions; p 1, p 2, p 3, p 4 and there were two actions A 1 and A 2 that were applicable in the start state. Then, a search-based planner would construct a search space in which for example in forward state space planning, it would apply the applicable actions and move on to on progress over that action to the new state.

So, in this diagram, the new state have there are two states; state 1 and state 2. State 1 has some p 1, p 2, p 4 and a new proposition called p 5 and state 2 has three predicates p 2, p 3, p 4. Perhaps, I should have added another one p 6; but anyway, this is just to illustrate the idea.

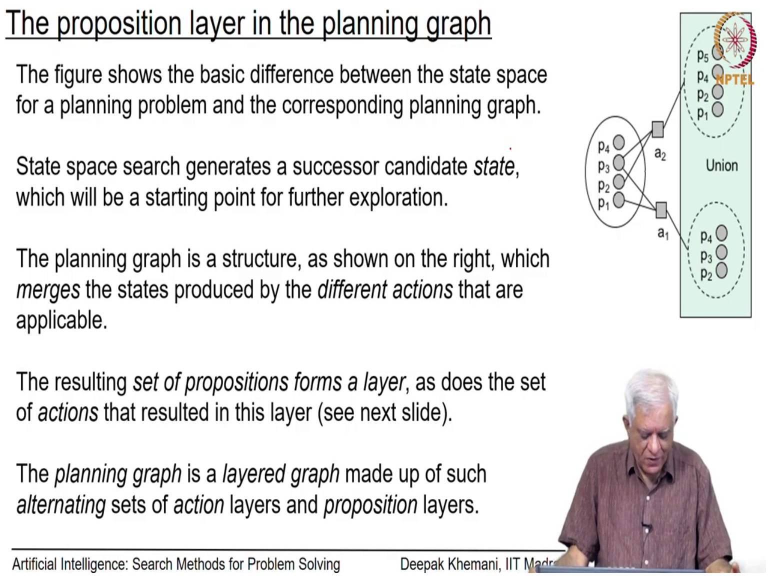
What the planning graph does and this is the most significant part is that it takes a union of all possible states that the planner could have moved to by applying one action and it constructs a layer of those proposition. So, that is called the proposition layer.

So, you can see that in this particular diagram on the right hand side, we have kind of merged the propositions in the two states and we have constructed one layer this thing. Of course, I have not shown you the union operation at this moment because as we can see there is p 2 here and p 2 here, but and also p 4 here and p 4 here; but in the union, there will be only one copy of that.

So, in a sense, this can give us some insight into what makes graph planning and it is that instead of considering all possible sequences of actions and all possible states that you can go through which can go exponentially, it kind of compresses them into a layer which somehow implicitly contains all possible states.

Of course, it has to do a considerable amount of work to sort out one state from another, but we will see how it does that. So, this is the basic idea in graph plan that you take the union of all possible states that can be reached in one step and call that a planning call that a proposition layer in the planning graph essentially.

(Refer Slide Time: 11:17)



## The proposition layer in the planning graph

The figure shows the basic difference between the state space for a planning problem and the corresponding planning graph.

State space search generates a successor candidate *state*, which will be a starting point for further exploration.

The planning graph is a structure, as shown on the right, which *merges* the states produced by the *different actions* that are applicable.

The resulting *set of propositions forms a layer*, as does the set of *actions* that resulted in this layer (see next slide).

The *planning graph* is a *layered graph* made up of such *alternating* sets of *action* layers and *proposition* layers.

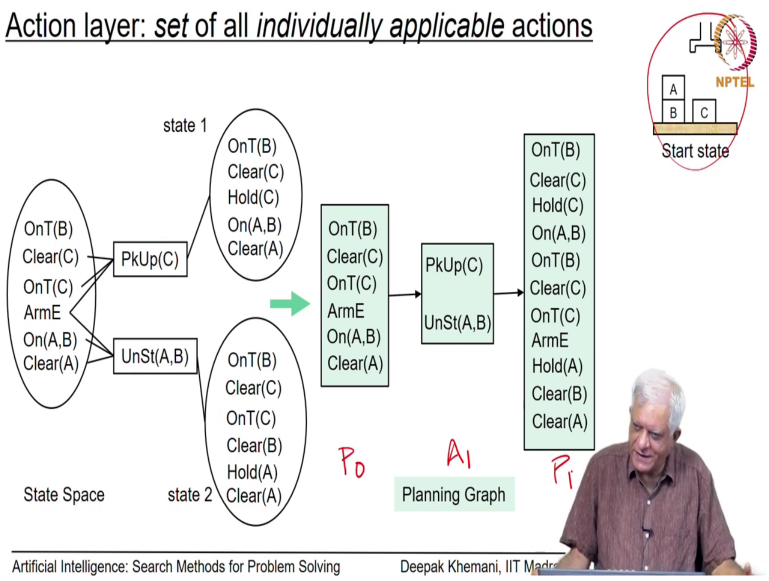
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madr

So, that is the proposition layer that we have been talking about and the figure shows the basic difference between state space for planning problem and the corresponding planning graph. In the state space each action is applied individually and produces a successor state individually. In the planning graph, all actions are applied simultaneously; not applied, one should say considered simultaneously and proposition layer is constructed which is the effects of all those actions.

So, the state space search would have constructed a successor state from which search would have proceeded forward; but in the planning graph, all the states are kind of merged into one representation and it will be a SAT representation as we will see we will call it as a layer and all the states that can be produced by different applicable actions are merged into one layer essentially.

So, the resulting set of propositions forms a layer and in fact, as just the actions that resulted in this layer. So, this actions A 1 and A 2, they are being considered separately in state space planning. In the planning graph, they would go into an action layer as you will see in the next slide. So, the planning graph is essentially a layered graph which is made up of alternating layers or alternating sets of action layers and proposition layers.

(Refer Slide Time: 12:51)


Action layer: *set* of all *individually applicable* actions

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

So, this representation of the state space search with a specific example that we are looking at just to illustrate this. So, in this example, the start state it has been shown on the right top corner of this slide, A is on B and B is on the table C is on the table and the arm is empty.

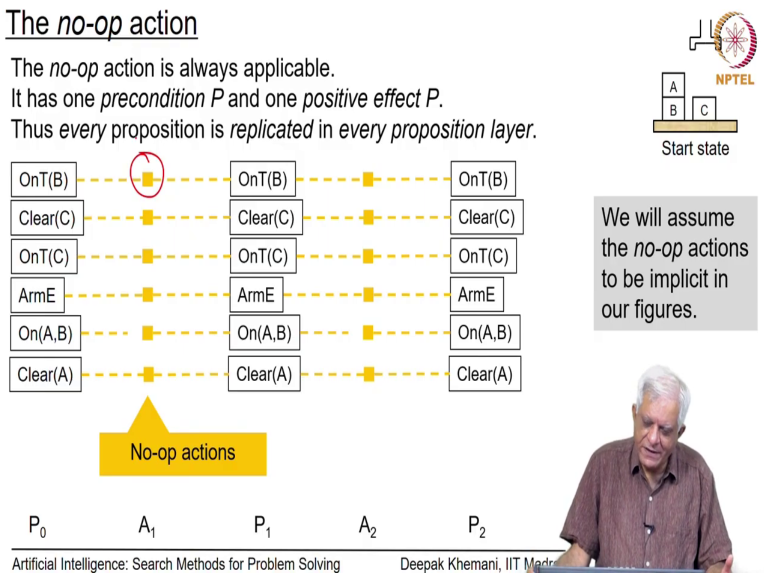
While constructing the planning graph, we do not really consider the goal state. It is like power state space search in some sense that you keep going forward and at some point, if you find the goal state, then it is terminate and we the planning graph also does something a little bit which is similar to that.

Now, in the state space as we have seen, we have had two applicable actions in this thing, you could either pick up block C or you could unstack block A from block B and you would go into two different states; state 1 or state 2 essentially. What happens in the planning graph representation? Is that all these three stages the start state; so, this is the start state which is often called as P 0 or the zeros planning layer.

Then, the first action layer and the first proposition layer which follows after that and there are these layer by layer and after the in the first action, there are both the actions that were applicable in the start state have been included and in the first proposition layer P 1, all the propositions that were true in the different states, when those actions were applied have been included essentially.

So, the action layer is a set of all individually applicable actions and this is the basic structure of the planning graph and after this, we will have more layers. So, the second action layer A 2 followed by the second proposition layer p 2 and so on and so forth.

(Refer Slide Time: 14:45)



## The *no-op* action

The *no-op* action is always applicable.
It has one *precondition P* and one *positive effect P*.
Thus *every* proposition is *replicated* in *every proposition layer*.

We will assume the *no-op* actions to be implicit in our figures.

Now, there is a special action that we consider in graph plan and in the planning graph and we need to consider this action because it is possible that there are certain actions which are applicable, but we want to apply them later on in the plan; but we want to carry forward the states forward.

So, in a way, since we are keeping a union of all propositions, we should be careful that any actions which were applicable earlier will also be applicable layer and this is done in the planning graph by using an action called the no-op actions.

So, there is no operator there and the no-op action basically is always applicable and for every proposition in your planning graph in a planning layer, you can include no-op action. The no-op action takes any predicate any proposition P as a precondition and it has only one positive effect which is the same proposition P essentially.

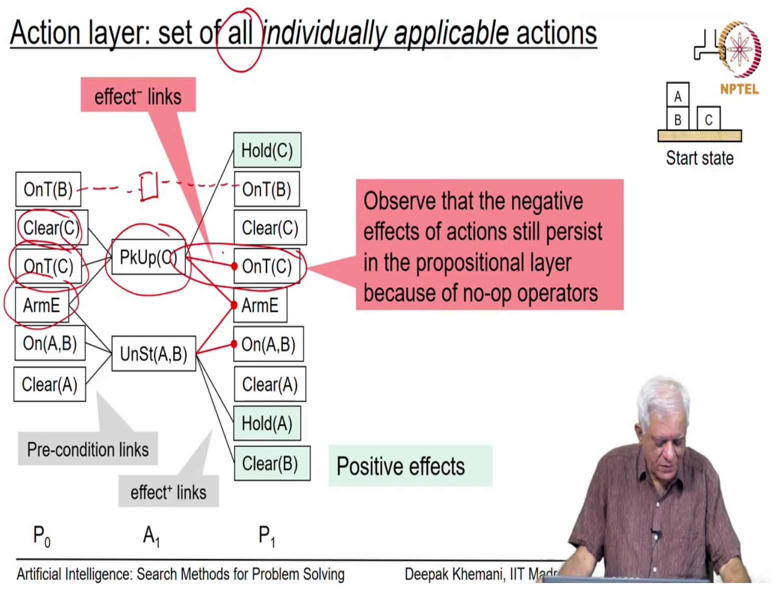
So, you can see here that the top most no-op action has preconditioned that B is on the table, on the table B and it has an effect on table B itself essentially. So, if this will be basically used to carry forward the state so that actions can be applied at a later stage as well.

Now, since a no-op action is always applicable, it is going to replicate the proposition layer at every stage essentially. So, every layer as we proceed forward, we will have the same proposition carry forward again. So, this is the first observation you must make about the planning graph, that if any proposition occurs in any layer i, it will also occur in any layer j which is greater than i essentially and the planning graph is constructed.

In our diagram that we will draw, we will assume that we will not draw the no-op actions. We have we have shown it here and except for one specific place, we will not show it again you must assume that these no-op actions are there as part of the planning graph essentially.

As we said the planning graph is made up of layers, proposition layers and action layers alternate alternating with each other and the action layers must include all no-op actions as well essentially.

(Refer Slide Time: 17:20)



## Action layer: set of all *individually applicable* actions

Observe that the negative effects of actions still persist in the propositional layer because of no-op operators

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

Now, let us consider the actions which are different from no-op actions, basically which do something which change something in the domain. Now, we saw that in this particular state, there are two actions which are applicable that you can unstack A from B or you can pick up C. So, we must add those two actions to the first action layer.

Remember that all the no-op actions are still present in this action, there we are just not drawn them here essentially. So, implicitly there is an action which is the no-op action, which carries forward the predicate or proposition on table B and that is why all the propositions layer P 0 have been replicated into the action layer P 1 essentially.

So, we have just two actions pick up C and unstack A from B, but we have to also include other information about these actions and that information pertains to the preconditions for the actions, we must know when the action is applicable. In this case, somehow we have said that they are applicable, but we have not depicted it in the planning graph.

So, the planning graph explicitly contains sets of edges in different layers and the sets of edges are the first one is the preconditioned links. So, we have preconditioned links going connecting to propositions and they identify the preconditions for every action here.

So, you can see here for example, that for pickup C, it must be true that that C is clear, it must be true that C is on the table and it must be true that arm is empty. So, the three pre conditioned links make this connection and store it in the planning graph essentially. Likewise for the other action unstack A, B.

Now, the propositions that we have shown in the first action layer P in the first proposition layer P 1, they have been generated by the no-op actions. So, for example, on table B is replicated in the first proposition layer, but we also have to include that the propositions which are the effects of these actions that we are added to the action layer and we must add all applicable actions to the action layer.

So, this all is important and that is for the sake of completeness, we do not want to miss out on any possible plan. So, the effects of these actions is the next thing that we have to think about and you can see that these positive effects will appear as new propositions in the planning graph in layer P 1 and this fact that they are the effects of these actions is captured by storing the positive effect links which tell you what are the positive effects of each of the actions.

So, unstack B A for example, has a positive effect that we are holding A and the fact that B has become clear; likewise pick up C has action that you are holding C. What remains is negative effects and the negative effects are stored in a different kind of a link which tells you that certain thing certain propositions are going to be deleted by this action.

So, for example, this negative link which says that if you pick up C, then on table C will no longer be true essentially. So, instead of deleting it, like we did in forward state space planning, we add a different kind of a link which is called a negative effect link.

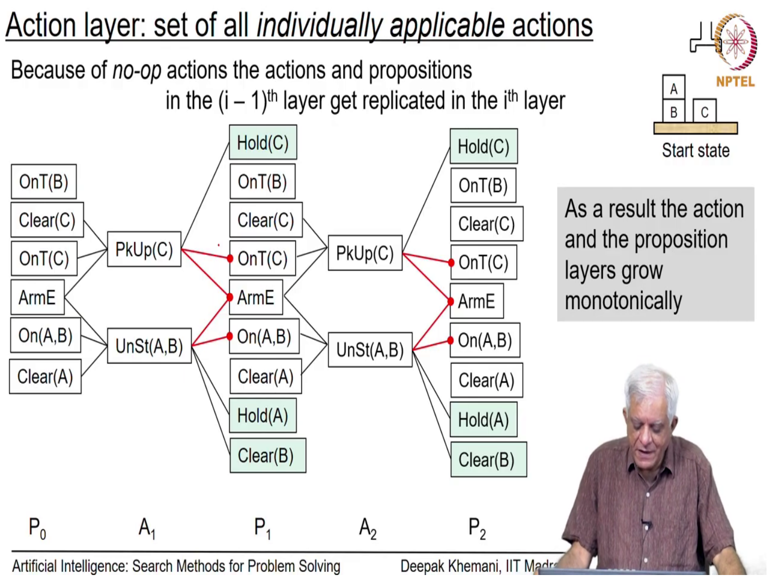
So, you can see that in this particular example, there are three or there are four negative links. If you pick up C, then C is no longer on the table or if you unstack A from B, then A is no longer on B and in both cases, the arm is no longer empty essentially. So, this kind of depicts the different kinds of layers that we are storing in the planning graph.

We have propulsion layers, we have action layers, we have succeeding proposition layers. So, P 0, P a, A 1 and P 1 in this case, then we have preconditioned links which link actions in A 1 into the preceding proposition layer and then, we have positive effect links and negative effect links which link action. So, they effects in the following layer essentially.

So, even though for example, when you do pick up C, forward state space planning would have deleted on table C. Graph plan does not do so, but it adds a new link from pickup C to on table C and this is a negative link, it says that this is a negative effect of pickup C. On table C survives because remember that we have those no-op actions and they will continue to be yeah continue to apply.

So, you can imagine that if you have to construct a plan in which the first action was unstack A from B, then you should be able to do this pick up C action sometime later and the fact that you are carrying it forward in every proposition layer will enable you to do that. The precise conditions, we will talk about as we go along.

(Refer Slide Time: 22:57)



## Action layer: set of all *individually applicable* actions

Because of *no-op* actions the actions and propositions in the (i – 1)th layer get replicated in the ith layer

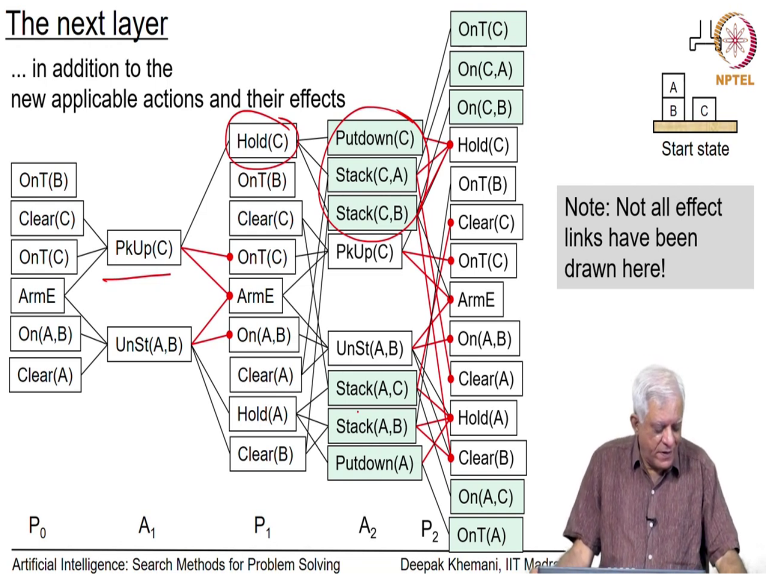
As a result the action and the proposition layers grow monotonically

Now because of no-op actions the actions and the propositions in any i minus one layer gets replicated in the ith layer. So, just as we replicated the start state actions into P 1, the start state propositions into P 1, we will also replicate the propositions in P 1 into the layer P 2 and because of the fact that the actions that were applicable in P 0 will be applicable in state P 1 also. We will also include those actions in the second action layer.

So, the first thing that we observed is that any action that has introduced in any layer will continue to be introduced in subsequent layers and any proposition that is introduced in any layer will continue to be introduced in subsequent proposition layers.

And this is all because we have these no-op actions, which means that the action layers and the proposition layers, they will grow monotonically as we grow the graph. We are growing the graph from left to right, it is a layered graph. But apart from this, replicating this thing, we also have new actions which are become applicable.

(Refer Slide Time: 24:16)



## The next layer

... in addition to the new applicable actions and their effects

Note: Not all effect links have been drawn here!

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

And so, in the next layer those will come as well. So, in addition to the no-op actions, the applicable actions and their effects have to be included. And now, you can see that in the next layer three more actions have been, six more actions have been included that because if you had done for example, pick up C and then you would be holding C and then, you could do any of these three actions that you put down C or you stack C onto A and you stack C onto B..

Likewise, if you had unstuck A from B, you could have done this something similar with A that you could you could have stacked it onto C or stacked it onto B or put it down essentially. And obviously, as we can see the number of propositions is growing the number of links is growing and in the diagram that I have drawn, I have not included all possible links because it is already becoming quite cluttered.

But I hope you get the idea that this is how the planning graph is grown essentially. So, we have spoken about the following sets or the following sets that are stored; one is a setup proposition layers. So, P 0, P 1, P 2 and so on. Then, there is a set of action layers A 1 between P 0 and P 1, A 2 between P 1 and P 2 and so on.

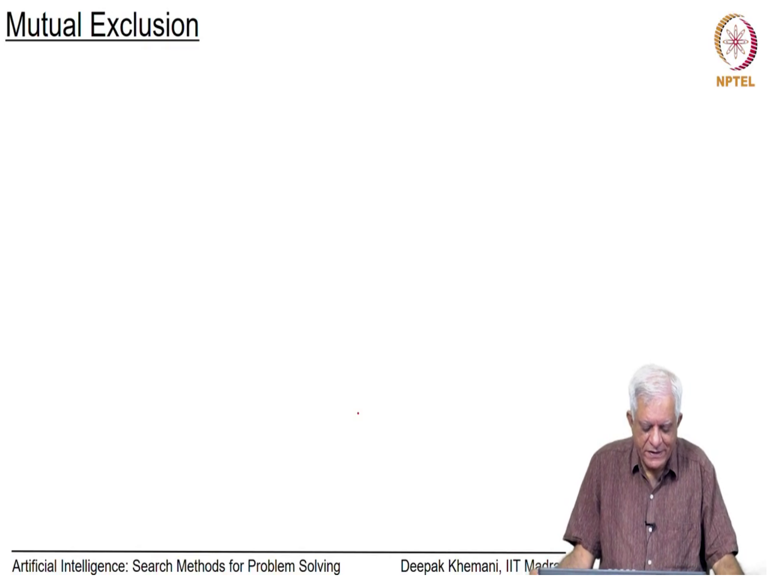
Then, each of the actions layers has a connection to the previous layer which is the preconditioned links and connection to the following layer, which is the effect links and there are two kinds of effect links; positive effects and negative effects.

But this is kind of you know merging everything together into in to one structure and it is not clear as to what can be what can be a possible set of actions which will be a solution for the plan.

But the advantage of constructing this planning graph is that it can be constructed in polynomial time and we kind of defer the act of searching for a plan onto the planning problem. Now, as I have said often that the planning problem has been shown, it was shown by Gupta and now in 1992 or so to be in P space complete which means it is in exponential time; but polynomial space.

So, you cannot say that you have found a solution to a problem which is cheaper than that; but within those bounds, you can find algorithms which are faster than other algorithms and that is what these new approaches tend to do. But now, we must also figure out as to how to identify states in this proposition layer and how to identify actions which can in practice, we executed in these action layers.

(Refer Slide Time: 27:16)



## Mutual Exclusion

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

So, to do that, we introduced another set of links and these are links which are within every layer, these are called mutual exclusion links.