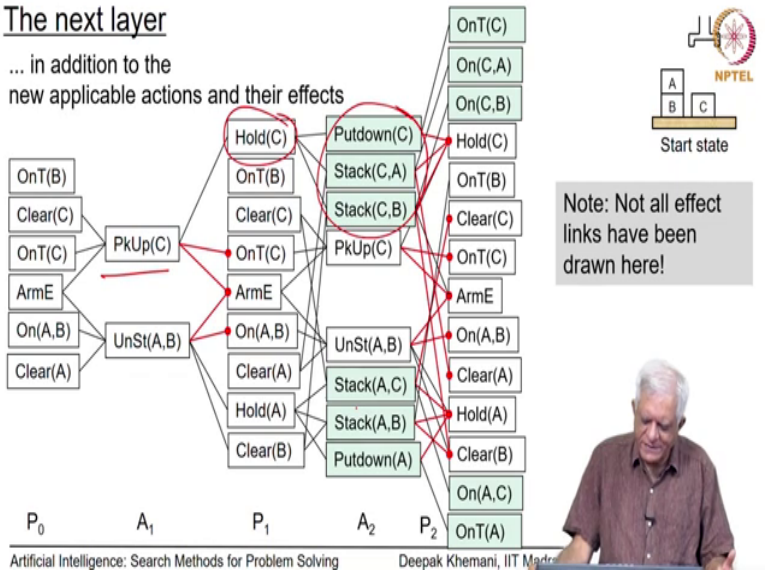
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter - 7 and 10
A First Course in Artificial Intelligence Lecture - 68
Algorithm Graphplan

(Refer Slide Time: 00:15)

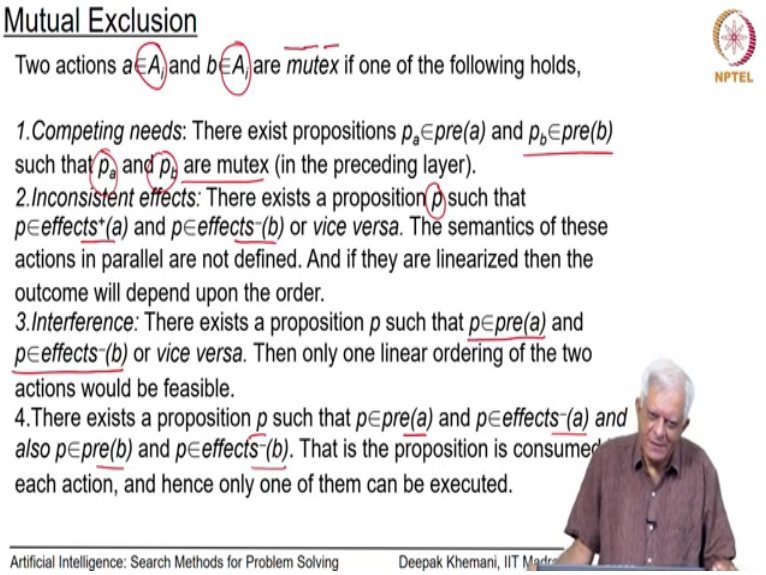


But this is kind of you know merging everything together into one structure and it is not clear as to what can be what can be a possible set of actions which will be a solution for the plan, but the advantage of constructing this planning graph is that it can be constructed in polynomial time and we 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. Be in B space complete which means it is in exponential time, but polynomial space.

So, you cannot say that you have found a solution to 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 proportion layer and how to identify actions which can in practice be executed in these action layers.

(Refer Slide Time: 01:40)



So, to do that we introduce another set of links and these are links which are within every layer these are called Mutual Exclusion Links. So, there are two kinds of mutual exclusion links. One is that between actions and the other is between propositions.

So, in the case of actions mutual exclusion link says that certain actions cannot be done in parallel. They cannot be done simultaneously. So, either you can do one action or you can do the other, but you cannot do both and this mutual exclusion is stored in the planning graph as a binary relation.

Though it is possible that you could have thought of something which is a higher order relation tertiary or whatever, but it is always a trade off between how much you represent, how much work you do to represent these kind of constraints and how much time you spend in solving those constraints essentially. So, maybe when we talk about constraints, we will get a little bit more flavor of that sort of a thing.

So, the mutual exclusion relations in planning graph or binary relations, so either two propositions are not feasible at the same time in the same layer or two actions are not feasible together in the same layer. So, we say the two actions in the same layer. So, observe that the layer is A i in this case.

So, in any layer two actions a and b we say we use a term mutex. The term mutex as you can imagine comes from mutual and exclusion. So, the first part of the word comes from mutual and the second part from exclusion.

So, we say that two actions a and b are mutex if one of the following holds that they have competing needs. So, if there is some proposition p a in the preceding layer which is a precondition of this action a and there is another proposition p b which is also in the preceding layer and it is a precondition of this action b and these proportions are such that the two propositions p a and p b, they are mutex essentially.

So if they are mutex, then the two actions cannot happen together because they are preconditions cannot be true at the same time. So, in general in the solutions returned to us by graph plan if there are two actions in the same layer and if they are not mutex, then in principle they can be executed in parallel essentially and in that sense this is very similar to what we studied for planned space planning that if there are two actions which in which there is no ordering link between them, then you could in principle execute them in parallel essentially.

But here we are saying we identifying which two actions cannot happen at the same time and these are the mutex actions. So, one the first one condition is that they are preconditions of mutex that is the competing needs. Another condition under which two actions are mutex that if there is a proposition p such that it is a positive effect of action a and it is a negative effect of action b.

So, imagine that p is being produced by one action and being deleted by the other action or vice versa. The semantics of these actions in parallel are clearly not defined because if you do both the actions in parallel, there is no way of saying whether p is true at the end of it or whether p a is false at the end of it.

And if you linearize them in some order, then the outcome will depend upon the order. If you do the adding action first, then at the end of it, it would be deleted because of deleting action will come later and if you do the deleting action first, then it will be true at the end of the this thing.

So, we want our planner to give us plans whose semantics are clearly defined and this was the case in partial order plans as well we said that any linearization of the partial order plan or any topological sort of a partial order plan should be a valid plan. Now if you allow these two actions such that one of them deletes p and the other one adds p, then clearly their linearizations may have different effects.

So, we just say that these two actions are mutex and they cannot be executed at the same time in that layer. Another condition is that there exists a proportion p, such that p is the precondition of action a and it is deleted by action b. So, in the previous case we were saying that both of them have conflicting effects. Now, we are saying that p is a precondition for a, but it is deleted by action b.

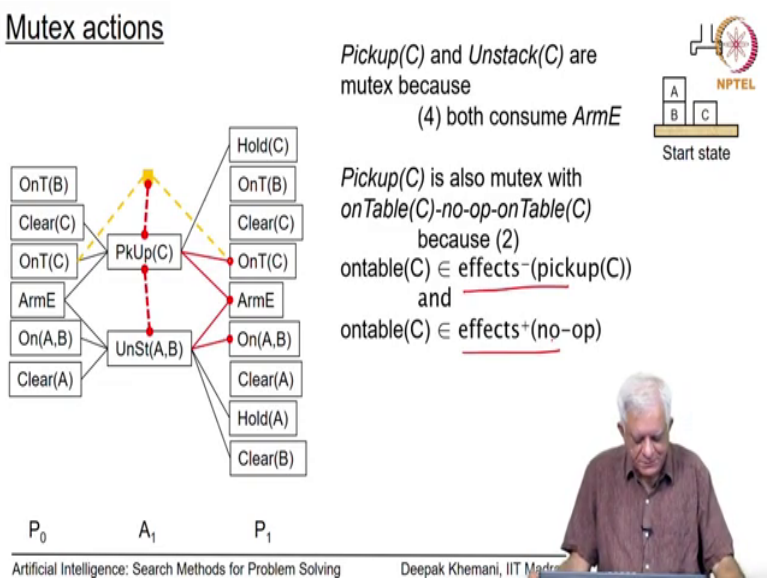
So, if you if you do actions a and b in parallel, then clearly the you can imagine that the semantics is defined, but when you do linear ordering, then you can see that these two actions will result into different states. So, in one state if you if you execute action a first and then you action b, then there is no problem. Both can be executed, but if you execute action b first because it is deleting the proposition p action, a cannot be executed after that.

So, only one linear ordering of the two would be feasible. So, again we say that the two actions are mutex and finally, one condition which is a kind of a special case of the of the third condition it says that if there is a proposition p such that p is a precondition of both a and action b and p is deleted by both actions a and b then we say then they are mutex.

So, you can see that one example of this would be the precondition arm empty ah. So, if arm empty is a precondition for pickup and unstack both and arm empty is deleted by both, then we are saying that a pickup action and an unstack actions or two pickup actions for that matter cannot be done simultaneously because they are consuming the precondition and deleting it.

So, both cannot consume it at the same time. So, these are the four conditions under which we say that the a set of actions is mutex.

(Refer Slide Time: 08:36)



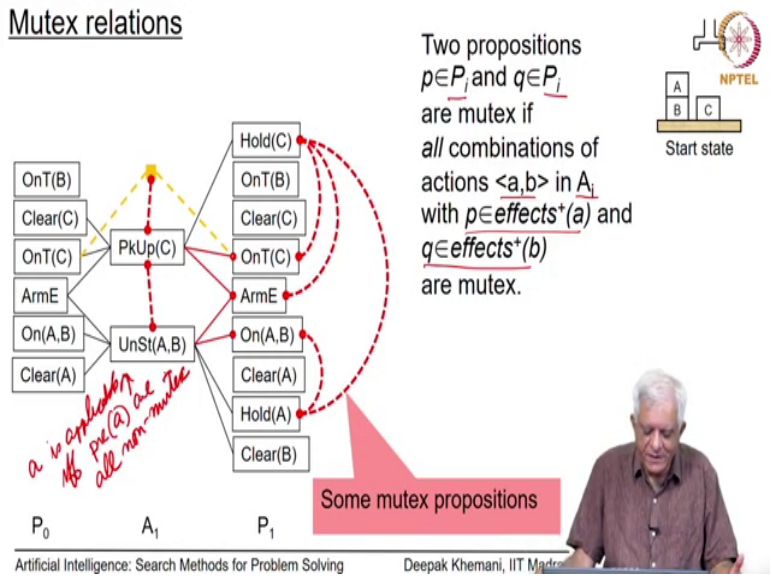
So, let us look at the example that we have been looking at. We had two actions and if we had a host of no-op actions out of which I have drawn only one just to illustrate this idea of mutex actions.

So, you can see that Pickup C and Unstack C on mutex because both consume arm and TC and that is depicted in this diagram by this link between the actions Pickup C and the red colored dashed link that you can see and the set of mutex links are links within a layer. So, in every layer you would identify what were the actions which are mutex and in every proposition layer as we will see shortly, you will identify the propositions which are mutex.

The action pickup C is also mutex with the no-op action that we have depicted here on which says that you know it carries forward on table C to the next layer because on table C is in the negative effects of pickup C and it is also on the positive effects of pickup C.

So, by condition two we say that these two actions cannot happen at the same time. Either one of these two happens. Now, this business about identifying mutex actions will be useful when we are extracting a plan out of that. So, we can in principle extract a parallel plan, but if two actions are mutex, then we cannot put them both into a plan.

(Refer Slide Time: 10:15)



So, when it comes to propositions we say the two propositions p and q both belonging to the same layer P i are mutex if all the ways of generating them all combinations of actions a and b which are in the preceding layer which is also called A i are mutex.

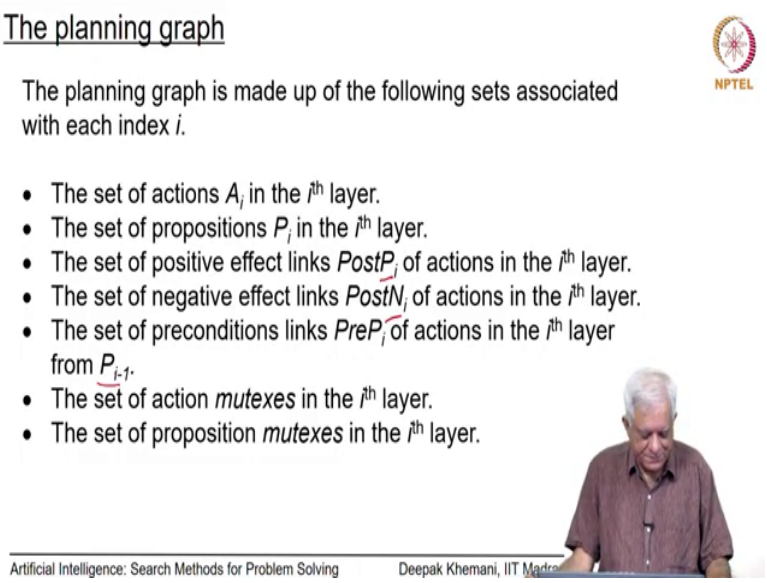
So, if p is a effect of a and q is the effect of v positive effect and if a and b you cannot find a pair of actions a and b which are non-mutex, then clearly you cannot produce p and q at the same time. So, p and q are mutex if all possible ways of generating them are mutex and they would be non-mutex if there is at least one pair of actions which can generate them which is non-mutex essentially.

So, here in this some example here are some mutex links that have identified not all. There are quite a few if you look at it carefully, you will be able to identify many more, but here is some example of a mutex p. Now, there is one thing which I have not mentioned here is that an action a is applicable. So, we are talking about actions here. We say that action a is applicable if its preconditions are all non-mutex in the preceding layer.

So, we cannot add all actions in general, but we can only add those actions whose preconditions are non-mutex. Now, if you observe this the initial state p 0 will always be non-mutex because it is a given state and hopefully it has been it is a consistent state given to us by the user. So, it can exist in practice. So, it is non-mutex and therefore, the actions that we have identified were already non-mutex in the preceding states.

But as we construct the planning graph further, we have to be careful that we only pick those actions whose preconditions are mutually non-mutex from each other which means that the action has to be applicable to be added to individually applicable to be added to the action layer essentially.

(Refer Slide Time: 12:41)



So, the planning graph now is can be seen is a made up of following sets associated with each index i, the set of actions A i in the ith layer, the set of propositions P i in the ith layer. So, the set of propositions are the effects of the set of actions in the previous layer including no-op actions remember.

Then the set of positive effects which connect the, so we will call this link as post effects of actions and p stands for positive here in the ith layer. So, we must know as to which propositions are produced by which actions.

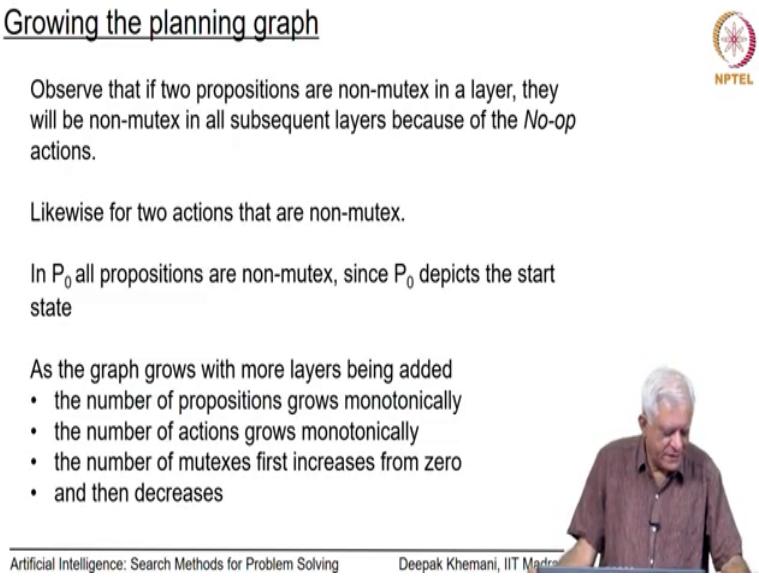
So, remember that when we wanted to identify whether through propositions are mutex or not, we needed to identify the actions from where which produced them. So, that is captured

by the positive effect links and likewise we have the negative effect links which tell you that this is a negative effect of that action.

Then we had the set of preconditioned links going from the set of actions in the ith layer to the propositions in the i minus first layer which tell you what the preconditions are and we can add the action only if the preconditions are mutually non-mutex and then, there are of course the mutex layers in the ith layer the action mutex in the iths action layer and the proposition mutex in the ith action layer.

So, each layer has this different sets of information stored alongside the set of actions, the set of propositions, the set of positive effects, the set of negative effects of actions, the set of preconditions of those actions and the set of mutex relations in each layer saying which two actions cannot be done in parallel and which two propositions cannot be true at the same time.

(Refer Slide Time: 14:43)



So, the planning graph is grown as we have said is grown from left to right. Now if two propositions are non-mutex in a layer, they will be non-mutex in all subsequent layers because of the no-op actions essentially and just imagine that it you do not introduce any action here.

And because the no-op actions will produce those non-mutex predicates, anyway they will be the set of actions which produce those propositions and therefore, because no-op actions are always non-mutex amongst themselves, the propositions also will be non-mutex.

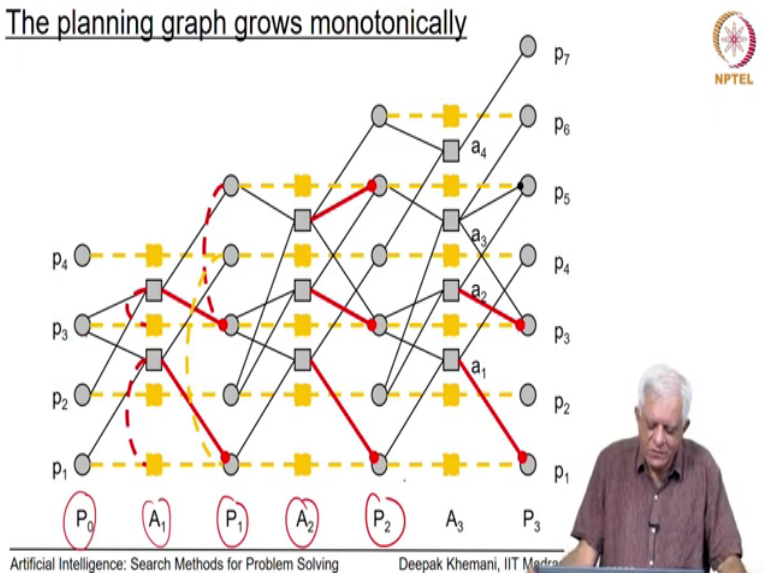
So, we had said that in the initial layer P 0, the setup propositions are non-mutex because you know that is a start layer. So, with this means that all the propositions which are true in the start layer will always continue to exist in all layers and in all layers they will be non-mutex.

Likewise if two actions are non-mutex, then they will continue to surface in every layer that is constructed after that essentially. So, in P 0 all propositions are non-mutex. In P 0 it depicts a start state and as the graph grows more layers get added and the number of propositions as we have observed grows monotonically.

The number of actions grows also monotonically, but the number of mutex first increases from 0 and then it may decrease later essentially. So, clearly we saw that in the first layer we had all in the 0th layer, all the propositions were non-mutex and the all the actions were introduced were between them non-mutex, but then we identified or the actions that we introduced were mutex essentially.

So, in the first proposition layer they were new mutex relationship, but in the first proposition layer we saw that initially they were no mutex relations in P 0, but in P 1 they were many essentially. So, mutex layers can appear and then increase and then maybe they will get reduced later.

(Refer Slide Time: 17:16)

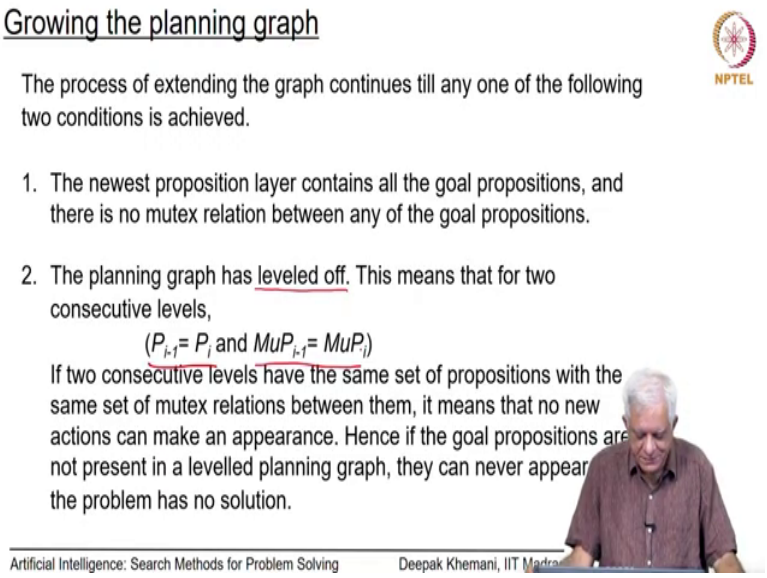


So, in general this is how our planning graph looks. We have ah set of proposition layers P 0 here, then A 1, then P 1, then A 2, then P 2, then we have the preconditioned the links which are shown in black arrows from P 0 to A 1 and from P 1 to A 2 and so on.

And then we have the positive effect links which are also shown in black arrows or lines. So, from A 1 to P 1 and from A 2 to P 2 and so on. We have negative effects which are shown as red thick lines which are going from A 1 to P 1 and A 2 to P 2 and so on.

Then within each layer we have mutex links between actions and between proposition essentially. So, the planning graph keeps growing and as we said it keeps growing monotonically.

(Refer Slide Time: 18:12)



And there are two conditions under which we stop growing the planning graph. These are as follows that one of the following two conditions is achieved.

The first condition is in the latest proposition layer that we have constructed, all the goal propositions are present and there is no mutex relation between any of the goal propositions. So, if all the goal propositions are present and there is no mutex layer mutex link, then it is possible that there may be a plan of that length.

So, we stopped growing the planning graph and start searching for the plan in the backward fashion. Now, I am saying that it is possible that there may be a plan. It is not necessary that there would be a plan because even though the there are no mutex links in the goal propositions remember that we have said that the mutex links are binary in nature.

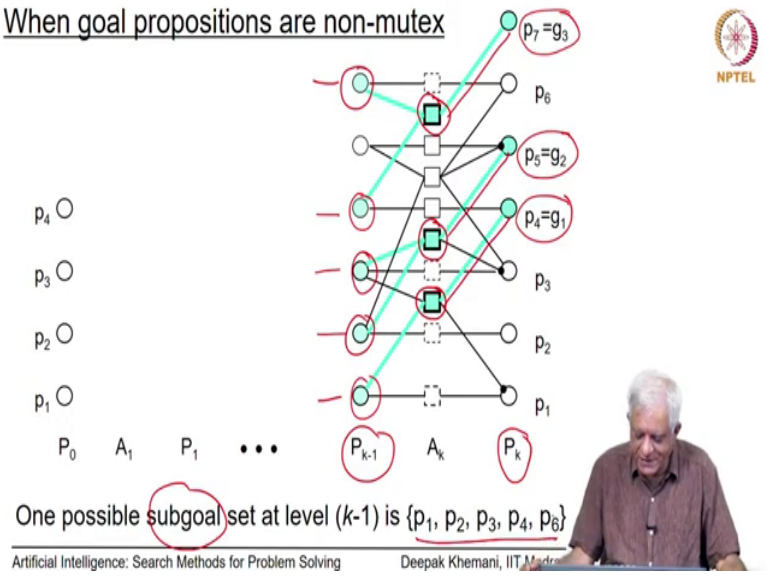
So, supposing they are three goal propositions, it is possible that there is no mutex between any two of them, but somehow between all three of them cannot happen to be true simultaneously in that layer, but it is possible that they might be true. So, we have to start searching backwards because one of the things that graph plan does is that it returns a shortest ah make span or the shortest path plan of the or the smallest plan to the this thing.

The other condition is becomes true when there is no plan in fact and this has to be identified in some way and we say that the other condition is that the planning graph has as we say leveled off that there is no more changes that are going to happen in the planning graph and we identify this by saying that if two consecutive proposition layers have the same propositions and also if two consecutive layers have identical mutual exclusion links or mutex links.

Then we say that the graph has leveled off and you can see that if this is the case, the two consecutive levels have the same set of propositions with the same set of mutex relations between them. It means that no new actions can be make an appearance in the following layer essentially whatever happened in the previous layer P i minus 1 those actions will be applicable, the same actions will be applicable in the layer P i and no new actions can be added.

So, there is nothing new that you can do. Hence just the goal propositions are not present in the level planning graph. They can never appear again because no new actions can be added after this stage. The graph has leveled off. We have finished exhaust; we have exhausted inserting all actions that were applicable at various stages.

(Refer Slide Time: 21:09)



But if the goal propositions are found and they are found to be non mutex, so in this example I have kind of assumed that p 7 p 5 and p 4 are the three goal propositions. We have called them g 1 g 2 g 3 and if they happen to be present in this kth proposition layer, then we start the backward search possible.

So, this is the first of the two conditions. When we stop growing the planning graph and this is when we are optimistic that we would have found a plan for achieving those three goal conditions g 1, g 2, g 3 which have been identified as p 4 p 5 and p 7.

Now, observe that because this is the first time we found them to be non-mutex, if we find a plan at this stage it has to be the shortest make span plan because otherwise we would have found them in the previous stage essentially. So, in that sense graph plan guarantees your shortest path plan or the shortest number of time incenses needed to execute the plan and the

and the actions may be more than the number of time incenses because there may be many actions which could be done in parallel.

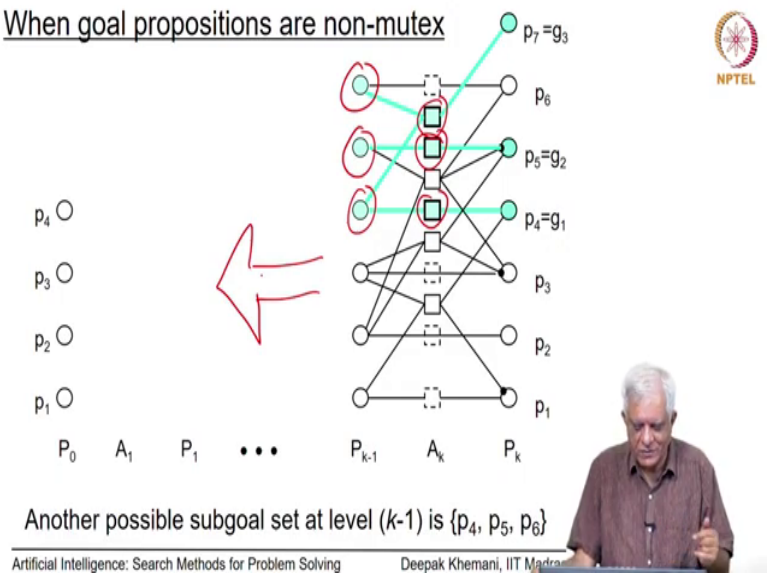
So, for example, in this case p 7 is achieved by this action, here p 5 is achieved by this action and p 4 is achieved by this action. So, we have identified three actions which produce the goal predicates that we are interested in. Now, we go back and see if the preconditions of these actions which have been shown in this shaded nodes, we go back and check whether the preconditions are mutex or non-mutex essentially.

So, obviously when we applied the first action, then its preconditions must have been non-mutex because we have said that we can only include actions if their preconditions are non-mutex, but we must also be sure that the preconditions of the other two actions which are these three this thing they are also non-mutex between each other.

So, what is the meaning of this? The meaning of this is if all these five propositions can be true in the layer p k minus 1, then in the layer p k minus 1 you could have applied those three actions that we have circled here and also shaded them and produce the goal predicates and achieved the goal essentially.

But now we have to make sure that this will be true that this can be that we can search backwards for a sequence of actions which are in practice feasible ok. So, this is one set of ah sub goal. So, we will call these as sub goals. So, these circle nodes are the sub goals and they are p 1 p 2 p 3 p 4 and p 6 as you can see here.

(Refer Slide Time: 24:17)

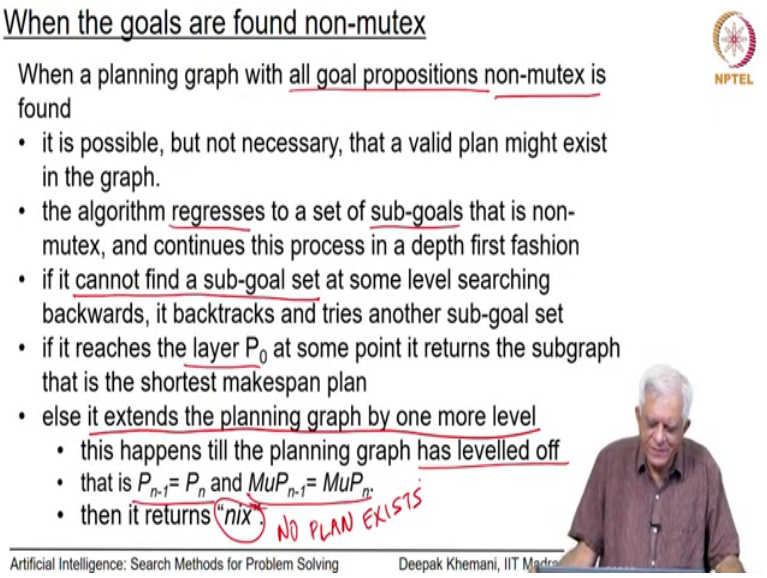


Another possibility would be a different set of actions as seen here and they have a different set of sub goals. So, maybe we can achieve this. So, it is possible that the plan can be achieved either through the sub goals p 1 p 4 p 5 p 6 or the sub goals that was seen in the previous case.

What graph plan does is that it explodes all these possibilities. It searches right to left looking for non-mutex set of propositions, actions propositions actions and so on till it if it can reach the p 0 layer essentially.

If you can find such a connection to the p 0 layer, that means the actions that are there in this sub graph are applicable. You can apply them and produce the effects which will be non-mutex and so on.

(Refer Slide Time: 25:03)



So, when the goals are found which are non-mutex, a planning graph with all goal propositions is found a planning graph with all goal propositions is found, all goal propositions in some layer is found and there should be non-mutex is mutex.

Then what the algorithm does is that it is possible, but not necessary that a valid plan might exist. So, it now goes searching for that valid plan it does. So, by regressing to the sub goals that we have found and we want to look for, a set of sub goals in the preceding layer which are non-mutex and it continues doing that you know that first like fashion.

If at any point it can find that it cannot find a sub goal set in the preceding layer, then it backtracks. So, here backtracking means moving right and then moving left again. Remember that the algorithm is searching from right to left from the goal propositions to the sub goals to

their sub goals and so on and backtracking means that you could not achieve one path and then you go and try something else again.

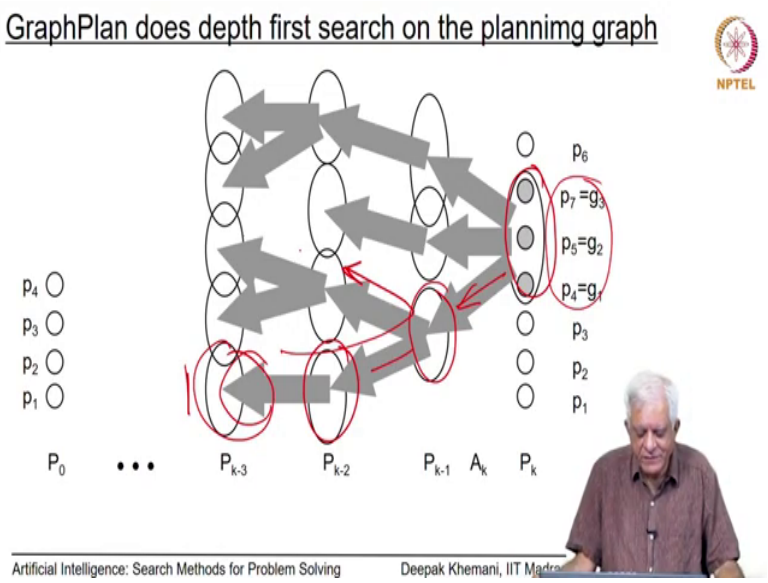
In this process, if it can reach the layer p 0, then we are done because p 0 is everything is anyway non-mutex and the sub graph that is been identified represents the shortest make span plan in this any act any two actions which are there in the same layer can be executed in parallel in principle or they could be linearized in any order and the plan would be a valid plan.

The other condition as we said is that in this process if it cannot find this, then it extends the planning graph by one more level and again does this whole thing process. Now, clearly because the goal propositions were mutex in this lets say kth layer, but it could not find a plan, then it will extend the planning graph to k plus 1 layer.

The same goal propositions must be there non-mutex, but the question is there a valid plan essentially because you know some mutex relations might have some more mutex relations might have either increased or decreased or something like that.

So, it keeps extending the planning graph as we have said till it has leveled off and the leveling off is identified when two layers in the proposition layers are identical and the mutex layers also identical to each other in which case it returns some symbol which says that no plan found. Now, when it says there is no plan, it is a complete algorithm. It means that there is no plan.

(Refer Slide Time: 28:10)



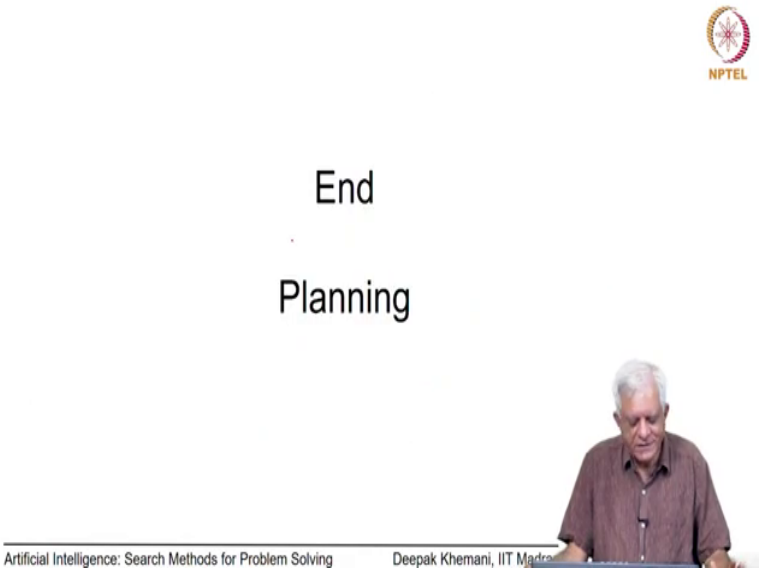
So, this backward state space, this backwards search in the planning graph can be kind of characterized like this that it has taken these three goals that we said p 7 p 5 and p 4 and it regresses them.

So, given this goal set, it regresses to one goal set, then it from there it regresses to another goal set and from there it regresses to another goal set and if this is a dead end, it backtracks and try something again and in this fashion it does that first search.

And if it one of these paths leads to the p 0 state, it will return a plan otherwise it will go and extend the planning graph to one more layer and it will keep doing that till it has leveled off. So, this was a very very brief introduction to graph plan. It is a very interesting algorithm and

you must really look up. The sources for the planning graph and its variations have been found, for example for temporal actions and so on.

(Refer Slide Time: 29:21)



Hm, but we will stop our study of planning with this thing because our courses are much more basic and has a wider reach and we will take up another topic in the next session, ok. So, see you then.