Fundamentals of Artificial Intelligence

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

Lecture-23
Planning Graphs and GraphPlan

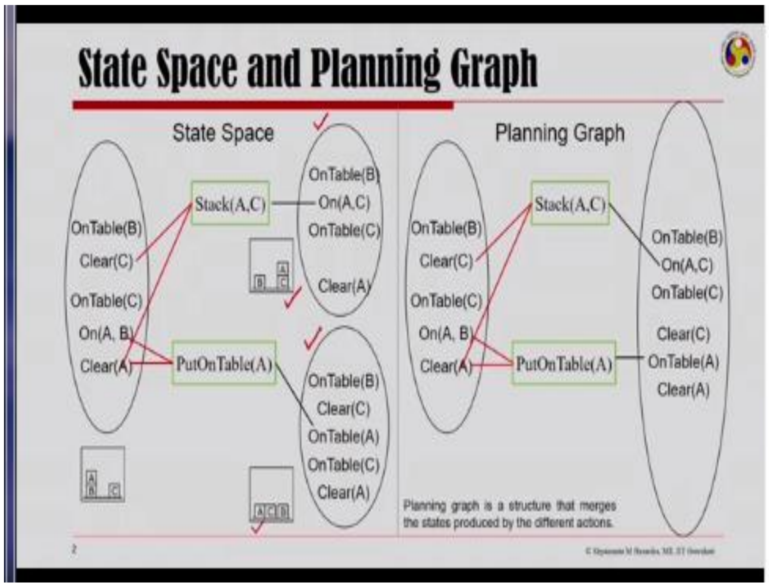
Welcome to fundamentals of artificial intelligence, we continue our discussion on planning, within this module we have introduce the basic idea of planning. We have looked at how planning can be seen as a form of problem solving in which an agent uses beliefs about actions and search for a solution. We have introduced the idea of a planning agent and seen how it combines the very best of knowledge representation and problem solving.

We had looked at situation calculus a variant of first order logic which allows beliefs about a changing world to be represented. Situation calculus allows a knowledge based agent to reason about consequences of actions arriving at a plan. We then looked at strips representation formalism of pure situation calculus for planning and saw how planning can be accomplished within strips by goal stack manipulation.

Thereafter we focused on plan space search and looked at partial order planning, we saw how the principle of least commitment and declobbering allow the partial order planner to arrive at a plan. Today we look at graph plan a completely different approach to planning, instead of looking for a solution in the state space or in the plan space graph plan creates a structure in which all possible solutions are represented, this is called the planning graph.

Thereafter it looks for a solution within the planning graph, let us start our discussion by looking at the planning graph vis-a-vis the state space. So if I was stacking of a simple robot arm that could pick and place some blocks and I have a blocks world representation like what is shown on the left of your screen here.

(Refer Slide Time: 04:04)

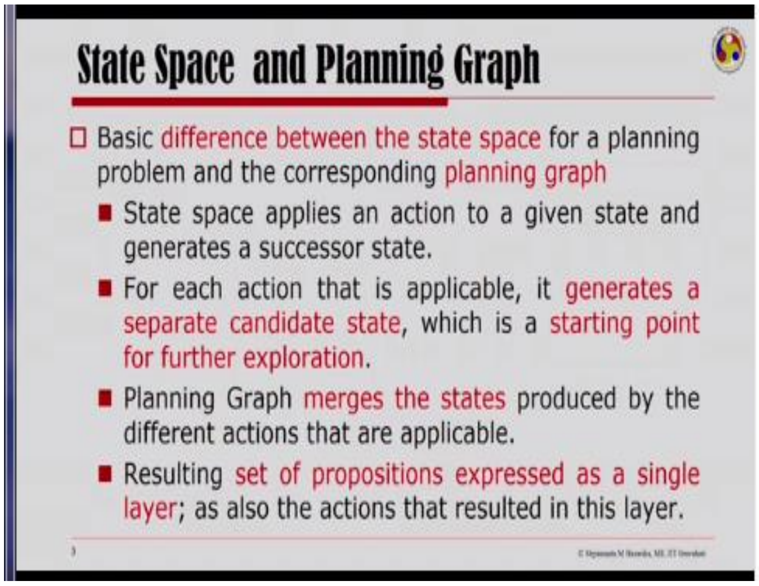

State Space and Planning Graph
State Space
Planning Graph
OnTable(B)
Clear(C)
OnTable(C)
On(A, B)
Clear(A)
Stack(A,C)
PutOnTable(A)
On(A,C)
OnTable(A)
Planning graph is a structure that merges the states produced by the different actions.

I have 3 blocks A, B and C, A is on B and C is on the table, now given this initial configuration of the blocks one could think of an action like stack A on C which would look at certain preconditions to be satisfied, for this case C needs to be clear it should not have anything on top of it, A needs to be clear. Then I could approach stack A, C to arrive at a scenario which is A on C and B on the table.

Now I could use another operation like let us for example take an operation about putting A on the table which I call put on table A and that would depend on whether there is anything on top of A, so A needs to be clear. And I can pick it up and put it on the table to arrive at a scenario which is say A on table. Now contrast is to another scenario where instead of looking at these results of individual actions as states which are different.

I take this actions but when I look at the states I merge them together and if I can create a graph out of such operations where states from the state space based on actions being acted upon our merge together, I would end up having something called the planning graph. The planning graph is in it is simplest form seen as a structure that merges the states produce by different actions.

(Refer Slide Time: 06:35)



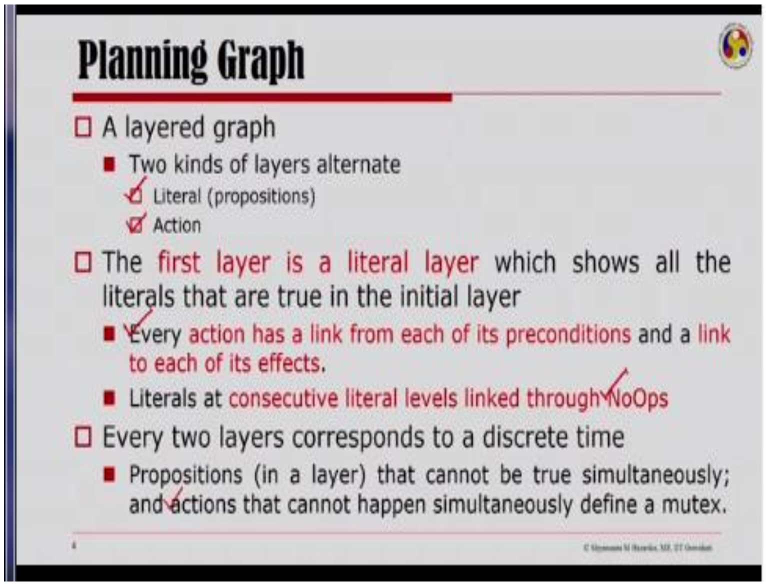
State Space and Planning Graph

- Basic difference between the state space for a planning problem and the corresponding planning graph
  - State space applies an action to a given state and generates a successor state.
  - For each action that is applicable, it generates a separate candidate state, which is a starting point for further exploration.
  - Planning Graph merges the states produced by the different actions that are applicable.
  - Resulting set of propositions expressed as a single layer; as also the actions that resulted in this layer.

So the basic difference between the state space for a planning problem and the corresponding planning graph is that the state space talks of an action over a state generating a new state, that new state is called the successor state. And thereafter each action is again applicable to the successor state which is in effect a starting point for further exploration, planning graph on the other hand merges these states produced by different actions that are applicable.

And as a consequence what I have is a set of propositions which are expressed as a single layer and you can now see that just before these layer of propositions, I would have a set of actions that have actually resulted in this layer.

(Refer Slide Time: 07:40)



Planning Graph

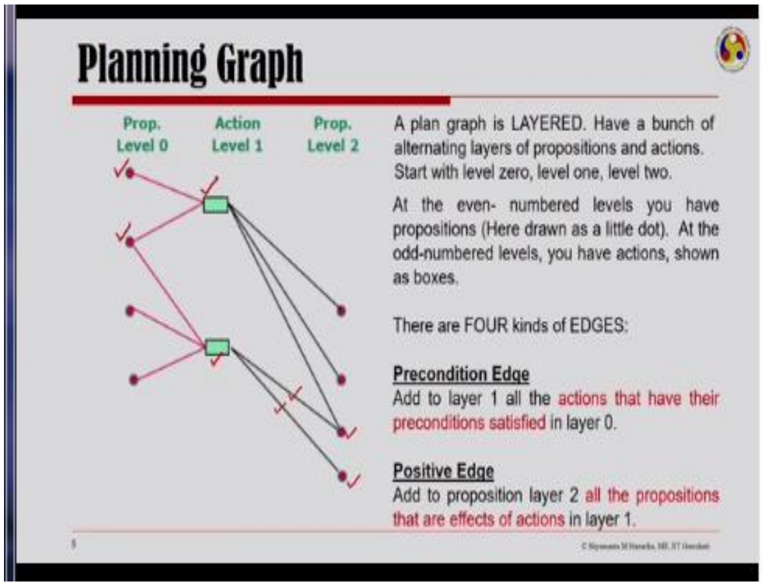
- A layered graph
  - Two kinds of layers alternate
    - Literal (propositions)
    - Action
- The first layer is a literal layer which shows all the literals that are true in the initial layer
  - Every action has a link from each of its preconditions and a link to each of its effects.
  - Literals at consecutive literal levels linked through NoOps
- Every two layers corresponds to a discrete time
  - Propositions (in a layer) that cannot be true simultaneously; and actions that cannot happen simultaneously define a mutex.

So we are now in a position to give a definition for the planning graph, a planning graph is a layered graph which have 2 kinds of layers which alternate, one the literals or the propositions and the other the layer of actions. The first layer as you have seen in the example is a literal layer that is it shows all the literals that are true for a given state to start with. Now from that layer we try to identify action.

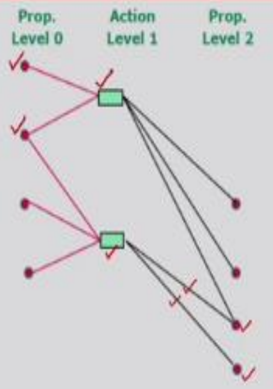
So every action has a link from each of it is preconditions and it links to the next layer to each of it is effects, also we will come back to this idea of NoOps which are some form of maintenance actions that link literals at consecutive literal levels. Now in a planning graph every 2 layer that I have actually corresponds to a discrete time that is propositions in a layer and the actions, we think of in discrete time.

Now we have propositions in a layer that cannot be true simultaneously as also certain actions that cannot happen simultaneously and these are defined to be mutex which is short for mutually exclusive.

(Refer Slide Time: 09:42)



Planning Graph



A plan graph is LAYERED. Have a bunch of alternating layers of propositions and actions. Start with level zero, level one, level two.

At the even- numbered levels you have propositions (Here drawn as a little dot). At the odd-numbered levels, you have actions, shown as boxes.

There are FOUR kinds of EDGES:

Precondition Edge
Add to layer 1 all the actions that have their preconditions satisfied in layer 0.

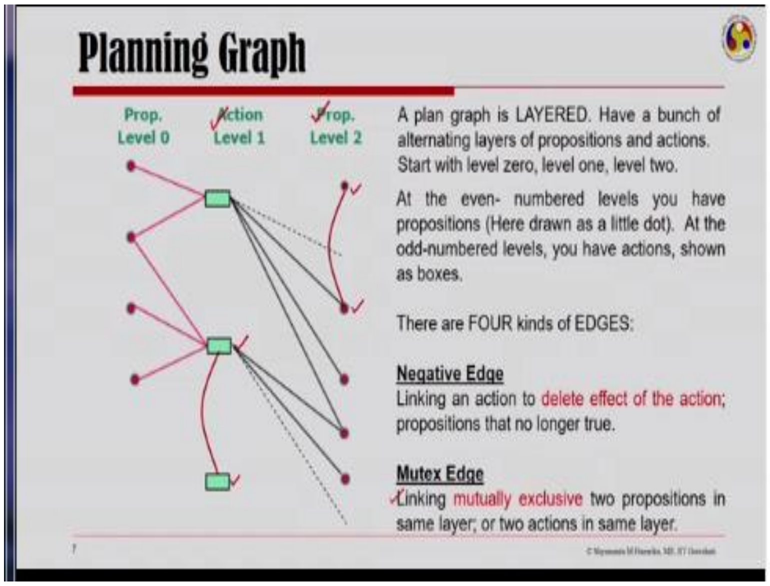
Positive Edge
Add to proposition layer 2 all the propositions that are effects of actions in layer 1.

So here is an example of the planning graph as already discuss the plan graph is layered, I have a bunch of alternating layers of propositions and actions start with level 0 and then I have level 1, level 2 so on and so forth. At the even-numbered levels as you can see you have propositions, here they are drawn as little dots and at the odd-numbered levels you have actions, here they are shown as small boxes.

In a planning graph there are 4 different kinds of edges, let us look at each of them very carefully, the first of the edges are called the precondition edges. The precondition edges are the preconditions for a given action, so the precondition edges link the preconditions from a given proposition layer to the particular actions. Now in this example, here we see that this particular action which is at level 1, needs to satisfy 2 preconditions in it is previous layer and these are linked together using the precondition edge.

The precondition edge we will mark in red, we then have the next type of edge which is called the positive edge. The positive edge is the edge from an action to the effects of the action in the next level, so for this particular action here if they lead to positive effects which are these 2 propositions at the next level. Then these edges here are the positive edges, we will mark positive edges here in black.

(Refer Slide Time: 12:16)


Prop. Level 0
Action Level 1
Prop. Level 2

Planning Graph

A plan graph is LAYERED. Have a bunch of alternating layers of propositions and actions. Start with level zero, level one, level two.

At the even- numbered levels you have propositions (Here drawn as a little dot). At the odd-numbered levels, you have actions, shown as boxes.

There are FOUR kinds of EDGES:

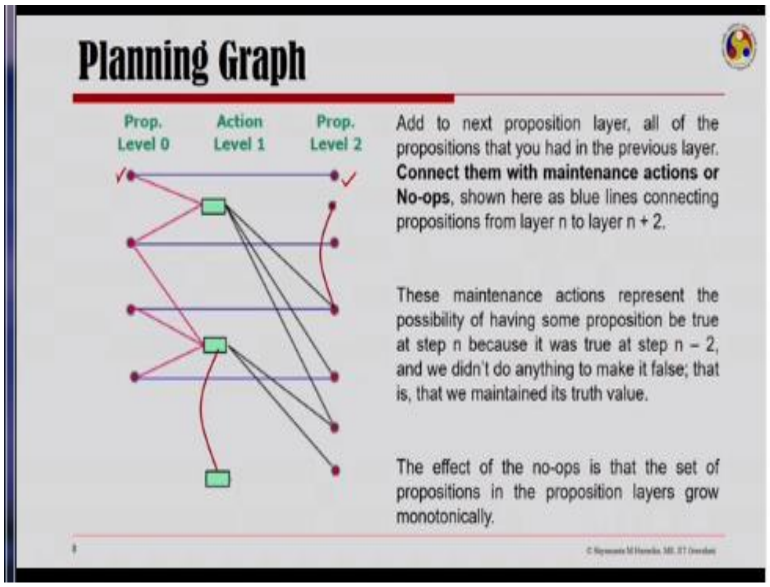
Negative Edge
Linking an action to delete effect of the action; propositions that no longer true.

Mutex Edge
Linking mutually exclusive two propositions in same layer; or two actions in same layer.

The third type of edge that we have is a negative edge, a negative edge actually links an action to the delete effects of the action. So these are actually propositions that are no longer true in the next level, so for a particular action here, if some proposition here in the next level is to be no longer holding. Then I will show it using the negative edge, a negative edge in the planning graph for our discussion today will be shown through a dashed line as here for a proposition that no longer world at this point.

The next kind of edge is a very interesting relationship marker and we will look in more depth in these relationship. We just look at here in the edge that we mark by linking what are called mutually exclusive propositions in each proposition level or mutually exclusive actions in each action level. So the mutex edge actually represents that these 2 actions this one and this one here, at the action level 1 cannot happen at the same time all propositions this and this cannot be holding at the same time.

(Refer Slide Time: 14:17)



Planning Graph

Add to next proposition layer, all of the propositions that you had in the previous layer. **Connect them with maintenance actions or No-ops**, shown here as blue lines connecting propositions from layer n to layer n + 2.

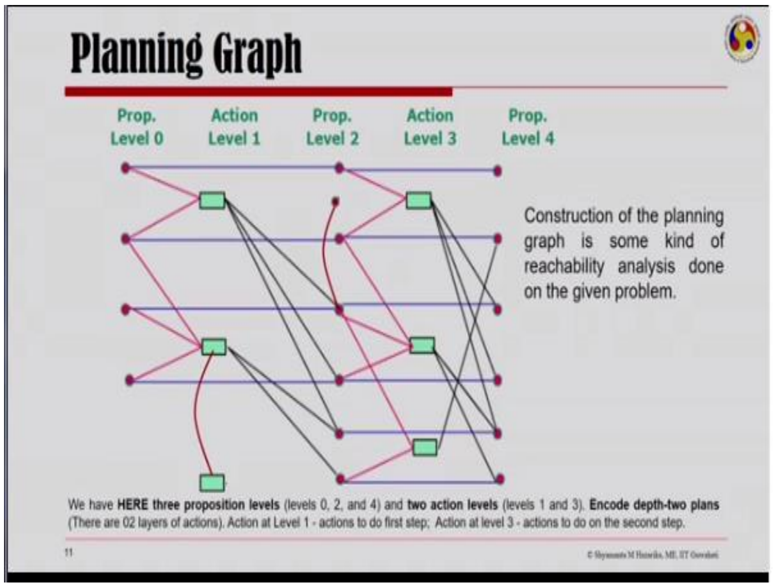
These maintenance actions represent the possibility of having some proposition be true at step n because it was true at step n – 2, and we didn't do anything to make it false; that is, that we maintained its truth value.

The effect of the no-ops is that the set of propositions in the proposition layers grow monotonically.

We add to next proposition level, all of the propositions that we had in the previous level, so if we had a proposition P at level O. We will also add this proposition at level 2, now we connect these propositions through what are called the maintenance actions or NoOps. NoOps for our planning graphs are shown using blue lines. Now these maintenance actions actually represent the possibility of having some proposition be true at step n because it was true at step n – 2.

And because we did not do anything about making it false, so we have maintained the truth value. You can see that the result of the NoOps is that the set of propositions in the proposition layers grow monotonically.

(Refer Slide Time: 15:28)



Planning Graph

Construction of the planning graph is some kind of reachability analysis done on the given problem.

We have **HERE three proposition levels** (levels 0, 2, and 4) and **two action levels** (levels 1 and 3). **Encode depth-two plans** (There are 02 layers of actions). Action at Level 1 - actions to do first step; Action at level 3 - actions to do on the second step.

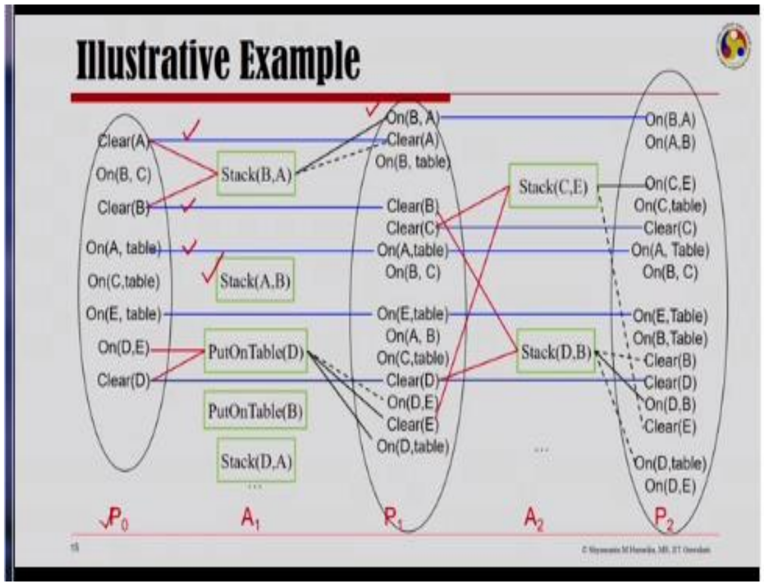
11

© Shyamanta M Hazarika, ME, IIT Guwahati

Now we can extend this further to another level of actions and to another level of propositions. So in a sense, we should be able to see that the construction of the planning graph is some kind of reachability analysis done on the given problem. So here what we have is a planning graph with 3 proposition levels level 0, level 2 and level 4 are levels of propositions and as already discuss proposition in one level are linked to proposition in other level through a group of actions.

So we have in between them action level 1 and action level 3, now a little consideration here we can see that this planning graph can encode a depth 2 plan. In the sense that this planning graph have 2 action levels action level 1 and action level 3, so we have 2 layers of actions. And therefore we are in a position to encode plans of depth 2, action at level 1 are the actions that we need to do first and then the action at level 2 are the next group of actions.

(Refer Slide Time: 17:11)



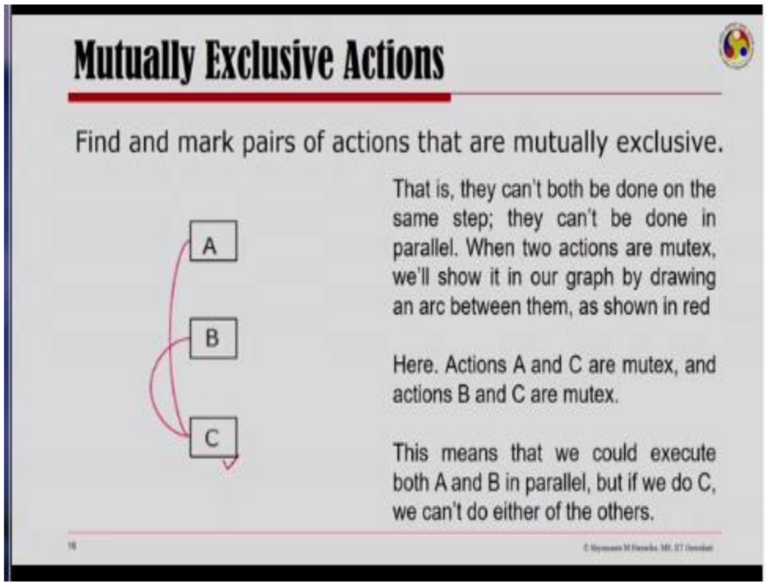
Let us look at an illustrative example, so here is a blocks world problem, where I have the initial scenario between a group of blocks A, B, C, D, and E and I can think of doing a group of actions on these blocks. So it could be there I would think of stacking B on A or stack A on B put D on the table or put B on the table or stack D on A. So each of these actions will be linked to the propositions in the proposition layer Po using the precondition edges.

So for stacking B on A, I need that A and B needs to be clear, so that is the precondition, so that is link there. And then I would have a positive effect of stacking B on A that would lead to B being on top of A, so I will link it to the next level of propositions. And this operation of stacking B on A would have a negative effect that is A would no longer B clear. So I would link a edge using a dotted line to clear A, similarly I could do for all other actions being marked here.

I have shown for only one more which is putting D on the table, so if D was on E and there was nothing on top of D. I could pick up D and put it on the table. So the preconditions are marked through the precondition edges and the effect of this would be that D would be on the table which I will mark through a solid black line. And then the very fact that I put D on the table will make on D no longer being true.

So I will mark with it a dash line here to that particular action, I can then think of generating the second level and then for each of the propositions at P0. I will link 2 propositions in P1 through what are called the maintenance edges which are marked in blue in this figure here. So here I have a planning graph with 3 layers of proposition and 2 layers of action.

(Refer Slide Time: 20:08)



Mutually Exclusive Actions

Find and mark pairs of actions that are mutually exclusive.

That is, they can't both be done on the same step; they can't be done in parallel. When two actions are mutex, we'll show it in our graph by drawing an arc between them, as shown in red

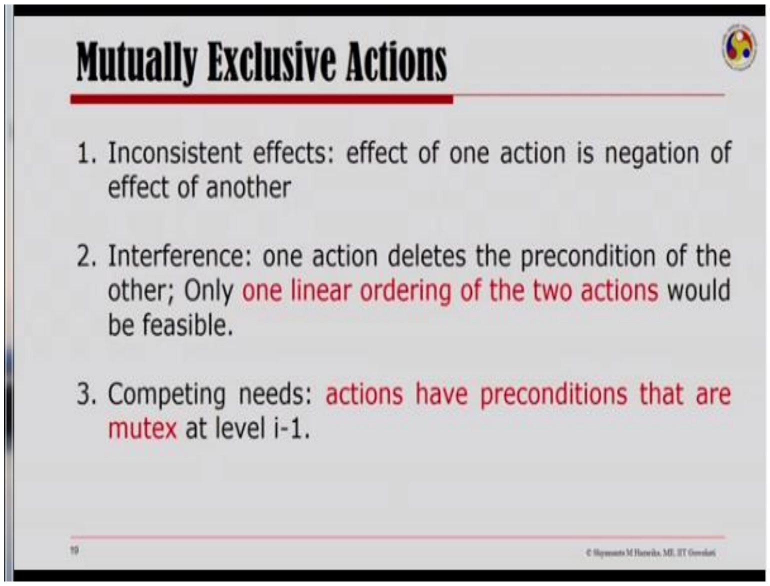
Here. Actions A and C are mutex, and actions B and C are mutex.

This means that we could execute both A and B in parallel, but if we do C, we can't do either of the others.

Now let us closely look at the mutex relation, so mutex could be between either the actions or the propositions. The mutually exclusive actions if I am looking at and trying to mark them as mutex it would mean that it would not be possible for me to simultaneously do these actions at that particular action level. So one needs to find and mark pairs of actions that are mutually exclusive they cannot both be done on the same step that is you cannot do them parallel.

Like here if you come back to that example and think of doing stacking B over A and stacking A over B, these 2 actions are actually mutually exclusive. So in this figure here we have actions A, B and C and we see that A and C are mutex and so is B and C. This means that we could execute both A and B in parallel but if we do C we cannot do either of them, because if we do C, C being mutex to B we cannot do B and we cannot do A.

(Refer Slide Time: 21:49)



Mutually Exclusive Actions

1. Inconsistent effects: effect of one action is negation of effect of another
2. Interference: one action deletes the precondition of the other; Only one linear ordering of the two actions would be feasible.
3. Competing needs: actions have preconditions that are mutex at level i-1.

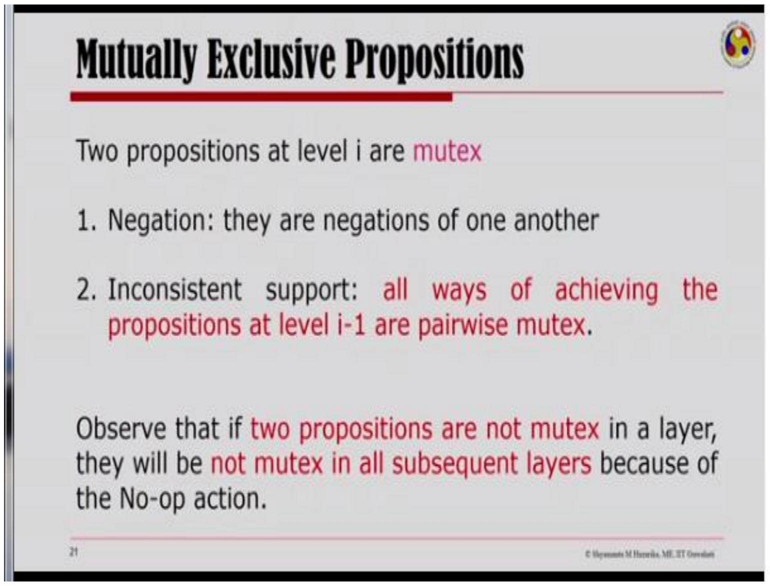
19

© Shyamanta M Hazarika, ME, IIT Guwahati

So mutually exclusive actions can be because of the following reasons, one inconsistent effect that is if the effect of one action is negation of the effect of another, then the semantics of this action in parallel are not defined and they are mutually exclusive. Second because of interference that is if one action deletes the precondition of another action, so only one linear ordering between them is possible and therefore they would be mutex.

The third is about competing needs actions that have preconditions that are mutex at the previous level are actually mutually exclusive actions.

(Refer Slide Time: 22:49)



Mutually Exclusive Propositions

Two propositions at level i are mutex

1. Negation: they are negations of one another
2. Inconsistent support: all ways of achieving the propositions at level i-1 are pairwise mutex.

Observe that if two propositions are not mutex in a layer, they will be not mutex in all subsequent layers because of the No-op action.

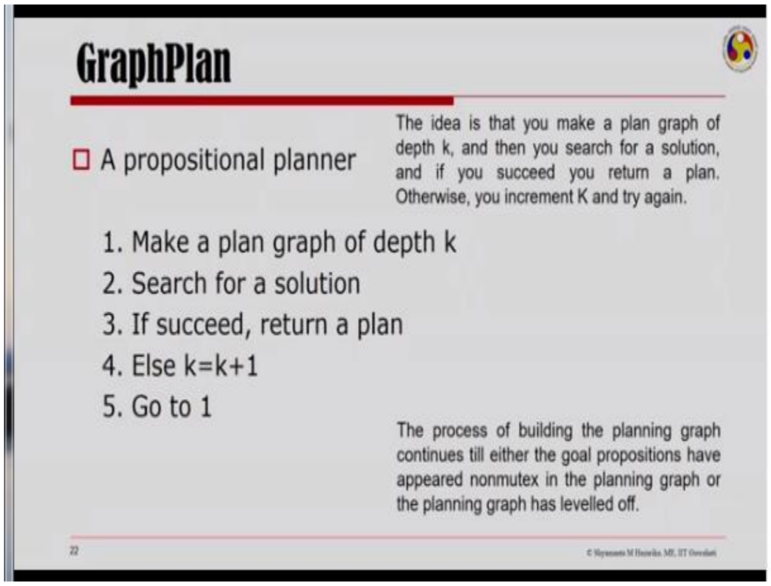
21

© Shyamanta M Hazarika, ME, IIT Guwahati

In terms of propositions 2 propositions at the same level are mutex for the following reasons, one when they are negations of one another and 2 when they have inconsistent support that is all ways of achieving the proposition at level i - 1 are pair wise mutex. Now 2 propositions if they are not mutex in a layer, we need to really understand that they would not be mutex in all subsequent layers.

And this is because of the NoOp action having introduce the concept of a planning graph, let us now look at the graph plan algorithm, the graph plan algorithm is a propositional planner.

(Refer Slide Time: 23:52)



GraphPlan

- A propositional planner

The idea is that you make a plan graph of depth k, and then you search for a solution, and if you succeed you return a plan. Otherwise, you increment K and try again.

1. Make a plan graph of depth k
2. Search for a solution
3. If succeed, return a plan
4. Else k=k+1
5. Go to 1

The process of building the planning graph continues till either the goal propositions have appeared nonmutex in the planning graph or the planning graph has levelled off.

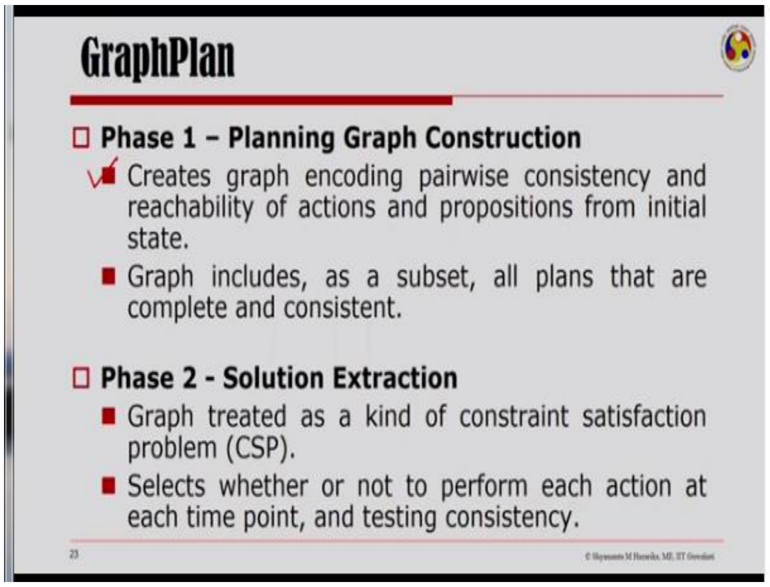
22

© Shyamanta M Hazarika, ME, IIT Guwahati

So we make a planning graph of depth K and then search for a solution if we succeed then we return a plan else we get to the next level that is we go to k + 1 and then search for a solution. Now the idea is that we have a plan graph and if you succeed you return a plan otherwise you increase the depth of the plan graph and try again. The purpose of building the planning graph is to continue till either you have the goal propositions in one of the propositional layers or the planning graph has leveled off.

Now what do we mean by the planning graph being labeled off we will come back to that in a minute.

(Refer Slide Time: 24:54)



GraphPlan

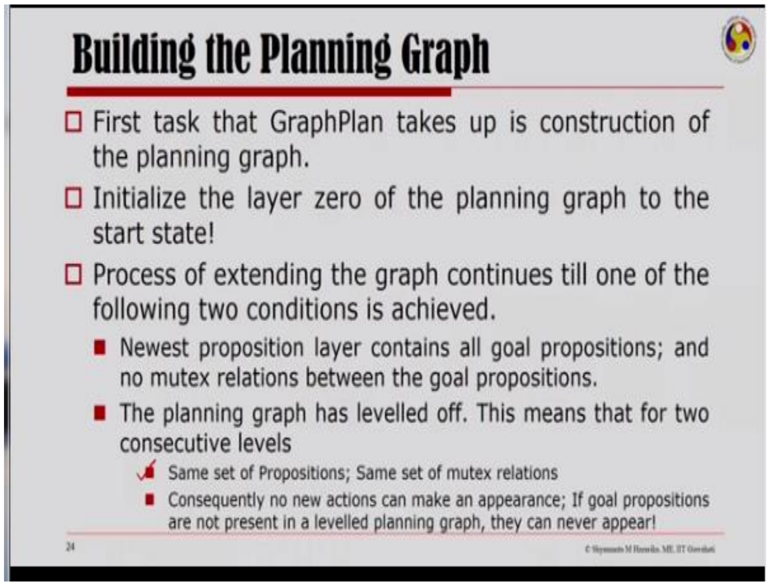
- Phase 1 – Planning Graph Construction
  - Creates graph encoding pairwise consistency and reachability of actions and propositions from initial state.
  - Graph includes, as a subset, all plans that are complete and consistent.
- Phase 2 - Solution Extraction
  - Graph treated as a kind of constraint satisfaction problem (CSP).
  - Selects whether or not to perform each action at each time point, and testing consistency.

23 © Shyamanta M Hazarika, ME, IIT Guwahati

But let us now understand that the graph plan algorithm that we are talking off actually has 2 phase, in phase 1 we do the construction of the planning graph. So it requires creating the graph encoding and we should realize that a graph includes all plans that are complete and consistent. In the second phase of the graph plan algorithm which is referred to a solution extraction, we are interested in finding out the plan or extracting the solution.

So the graph is treated as a kind of constraint satisfaction problem now and you select actions at each time point, check for whether they are consistent and whether they can give me the group of goal propositions at the final level.

(Refer Slide Time: 26:09)



Building the Planning Graph

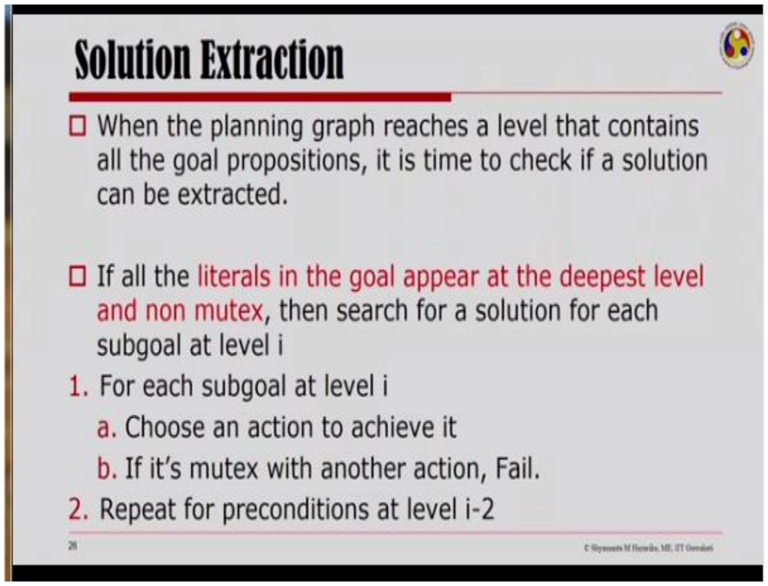
- First task that GraphPlan takes up is construction of the planning graph.
- Initialize the layer zero of the planning graph to the start state!
- Process of extending the graph continues till one of the following two conditions is achieved.
  - Newest proposition layer contains all goal propositions; and no mutex relations between the goal propositions.
  - The planning graph has levelled off. This means that for two consecutive levels
    - Same set of Propositions; Same set of mutex relations
    - Consequently no new actions can make an appearance; If goal propositions are not present in a levelled planning graph, they can never appear!

24 © Shyamanta M Hazarika, ME, IIT Guwahati

So let us look at the first task building the planning graph, I first initialize the layer 0 of the planning graph to the start state. And then extend the graph till one of the following 2 conditions is achieved, A either the newest propositional layer that has come have all the goal propositions and there are no mutex relations between these goal propositions or the planning graph is what I called levelled off.

Now what is meant by levelled off for a planning graph is that the 2 consecutive levels, we have the same set of propositions and the same set of mutex relations. Now the consequence of this is that no new action can make an appearance and if goal propositions are not present in a level planning graph, they can never appear and there is no solution.

(Refer Slide Time: 27:24)



Solution Extraction

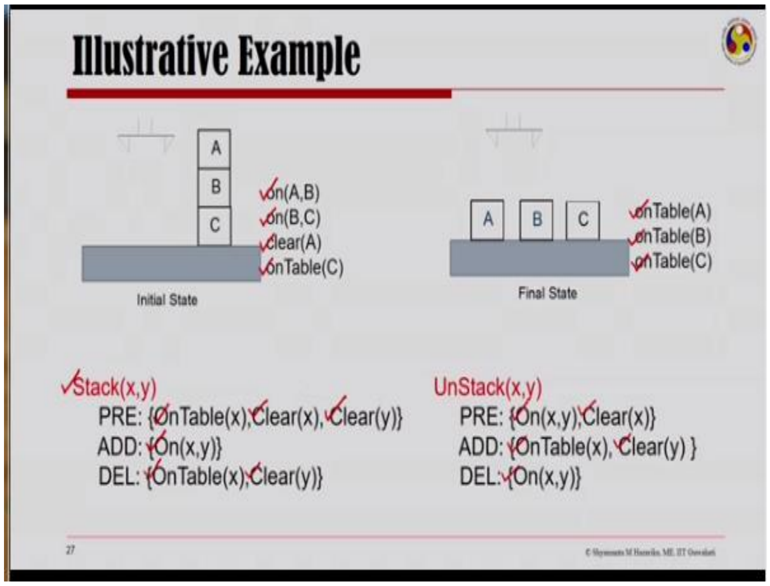
- When the planning graph reaches a level that contains all the goal propositions, it is time to check if a solution can be extracted.
- If all the literals in the goal appear at the deepest level and non mutex, then search for a solution for each subgoal at level i

1. For each subgoal at level i
   a. Choose an action to achieve it
   b. If it's mutex with another action, Fail.
2. Repeat for preconditions at level i-2

The next phase of solution extraction is when the planning graph reaches a level that either contains all the goal propositions or it is levelled off. Now one way of looking at the plan existence is to see if the goal propositions have support from a non mutex set of actions. Because the goal propositions need to appear at the last propositional level and they must come from a group of actions which are non mutex.

If this is true then the combined preconditions can be seen as the regressed sub goal set and this could be solved recursively and this is what is done in solution extraction. So if all the literals in the goal appear at the deepest level and they are non mutex, then you search for a solution for each sub goal at level i. And you choose an action to achieve that and if its mutex with another action you fail it and you repeat for preconditions at level i – 2.

(Refer Slide Time: 28:53)



Illustrative Example

A
B
C
Initial State
A
B
C
Final State

On(A,B)
On(B,C)
Clear(A)
OnTable(C)

OnTable(A)
OnTable(B)
OnTable(C)

Stack(x,y)
PRE: {OnTable(x),Clear(x),Clear(y)}
ADD: {On(x,y)}
DEL: {OnTable(x),Clear(y)}

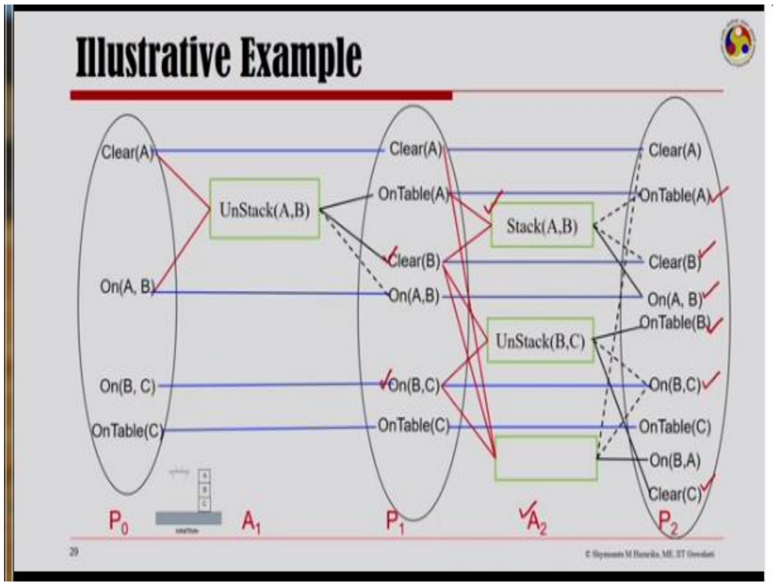
UnStack(x,y)
PRE: {On(x,y),Clear(x)}
ADD: {OnTable(x),Clear(y) }
DEL: {On(x,y)}

Now let us look at an illustrative example and try to understand the basic idea of getting to the planning graph and then extracting a solution out of the planning graph. So here I have a very simple blocks world problem where the initial state is about 3 blocks A, B, C stack one on top of the other. So I have A on top of B, B on top of C and C on the table and there is nothing on top of A so clear A, I want to arrive at a scenario where all the blocks are on the table that is my final state is about A being on the table, B being on the table and C being on the table.

Now by not looking at very integrate scenarios of action, I just take 2 very simple actions here for illustration sake I talk of an action stack x, y where the preconditions are that x must be on the table, there must not be anything on top of x and anything on top of y. So if I can stack x on y it will end up in having x on y and what it will result in not being true any longer is that x will no longer be on the table neither that y will be clear.

I also take help of another very simple action which is called unstuck, I can do an unstack of x and y, if x is on y and there is nothing on top of x. Unstacking acts from y would mean that basically I will have x on the table now instead of on y and there will be nothing on top of y. So I will have clear y, what will no longer hold is the fact that x will not be on y. So with these 2 very simple actions stack and unstack and a very simple initial and final state. Let us try to understand how to generate the planning graph and look at the graph plan algorithm.

(Refer Slide Time: 32:08)



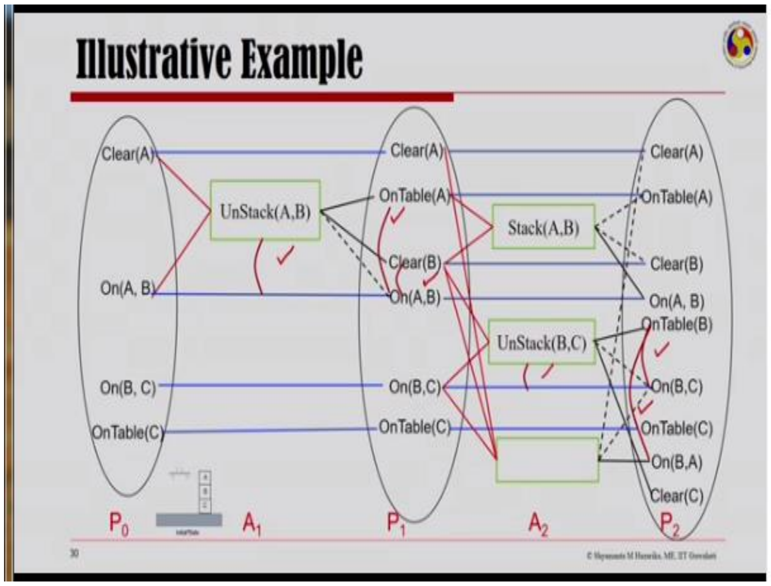
Illustrative Example

So here is my initial state I have clear A, I have A on B, B on C and C on the table and then remember I have 2 actions stack and unstack. So I can look for those actions here, I can do a unstack of A and B because A is on B and A is clear. So I can do an unstack A, B and I can get to my action layer where the positive effects would be that, A would be on the table and B would become clear and the delete effect would be that A on B would no longer hold.

I can also have all the maintenance or the NoOps actions, here they are represented using the blue lines I can still further do another group of actions. Here shown using level A2 to generate propositional level P2. So given A on the table and clear B, I can stack A on B, this would mean that A would no longer be on the table. So I have a delete edge from stack A, B2 on table A and B would no longer be clear, I would have A on top of B.

Similarly we could do an unstack action of unstacking B from C, so if you do an unstack B from C because the preconditions clear B and B on C is satisfied. So I can pick up B and put it on the table and when I put it on the table I no longer have B on C holding and as a result of putting B on the table I have made C clear. So I can keep on doing this is plan graph growing using propositions at every layer followed by actions which are feasible getting to a new layer of propositions.

(Refer Slide Time: 35:02)



Now there is a question of marking the mutually exclusive actions and propositions, here in this illustrative example I have only marked a couple of them. They are not complete but they are good enough for us to understand the concept of mutually exclusive actions and mutually exclusive propositions. So here I have the maintenance axiom or the NoOps of A on B and the action of unstack A on B.

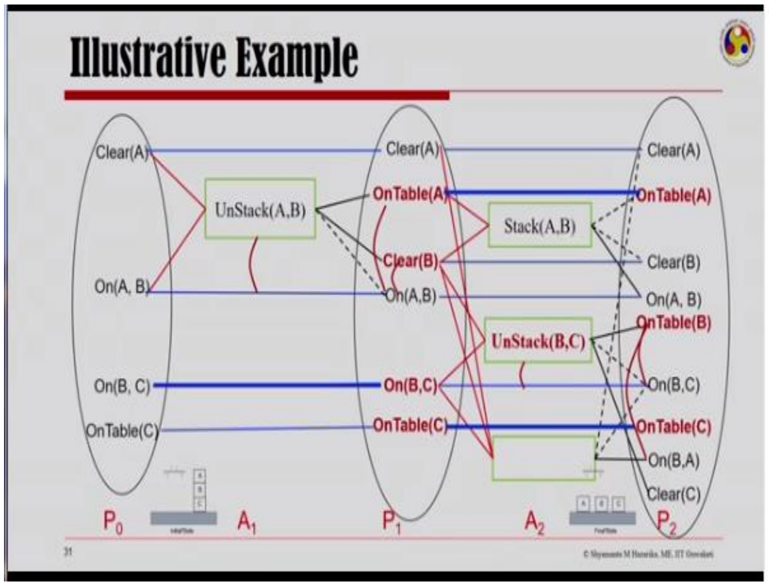
So that effect of unstacking A, B would be that on A, B would no longer hold and therefore these 2 actions are mutex. As for propositions if I have a proposition that says A is on B that would mean that there is something on top of B which is A and therefore B is not clear. And therefore here clear B and A, B would be mutually exclusive propositions, if I have A on B and if I have on table A, now these are mutually exclusive.

Because one says A is on the table, the other says is on top of B, similarly I could mark mutually exclusive operations like unstack B,C and on B,C here. And for propositions at level P2, I can see that I have mutex like on table B and on B,C the mutex prepositions or B on top of A and on table B being another set of mutexes. So now I have my initial layer P0 of the propositions which are true in the initial state, I then have a set of actions whose preconditions are satisfied at level P0.

And therefore these actions are possible and from these actions I generate my proposition level P1, at P1 I again look at which are the feasible actions that is my action level A2. And given these action levels I add the propositions that are true at proposition level P2. I mark these preconditions and the effects edges and also the delete edges, thereafter I have the maintenance actions or the NoOps marked and finally I mark the mutex relations between actions at each level and propositions at each level.

So that completes my planning graph, now I have to look for the second phase which is about looking for a solution.

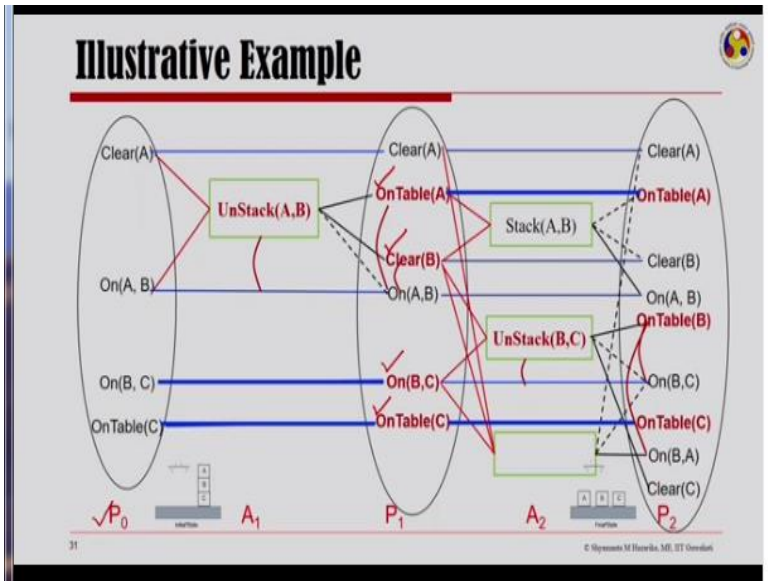
(Refer Slide Time: 38:33)



So in order to look for a solution what I do is, I first see how many of the goal propositions are true at my newest level. So here if you look at the right bottom corner of your screen you have the final state mark there and the final state the propositions that are there is on table A which is in proposition level P2. Then I have the proposition on table B and I have the proposition on table C, now interestingly these 3 propositions that needs to be true at the final state.

I have in P2 and they are all non mutex having got this I would now look for actions that make them true. So on table A I have the maintenance action or the NoOps bringing me to P1, actually from P1 the NoOps has brought on table A on to P2. I get the second one on table C through a NoOps and on table B if you see it has come because of the action of unstack B,C. So unstack B, C if I explore further I could see that unstack B, C is possible if I have clear B and if I have B on C in proposition level P1, so now these are my sub goals.

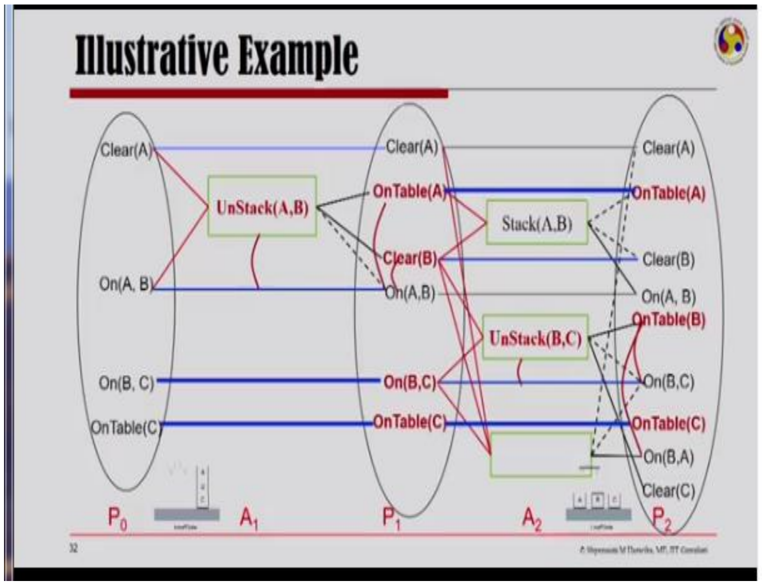
(Refer Slide Time: 40:33)



I need to have these things being true at proposition level P1 on table A, clear B, on B, C on table C. Now I will move on to actions that needs to be true in A1, so I see I have to have in NoOps to make on B,C true, another NoOps to make on table C true. Whereas these 2 propositions on table A and clear B, I see that this will become true when I have unstack A, B in A1 and these 3 actions are non mutex.

So I have arrived at my initial layer Po by moving through the proposition layers and the action layers, now I will try to extract out the solution.

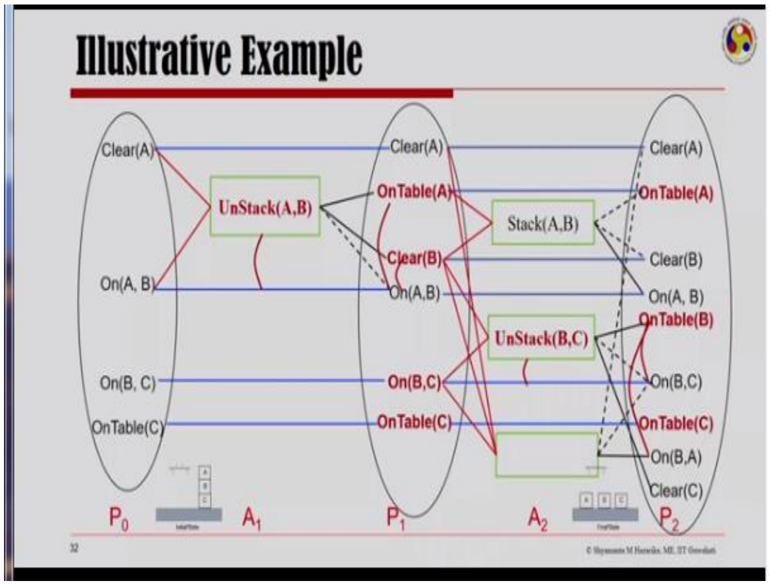
(Refer Slide Time: 41:43)



So we could see that the first action that I need is unstack A, B to take me to on table A and clear B and that gives me on table A and clear B being true at proposition level P1. I need a couple of maintenance operations to make on B, C true and on table C true. And then given on table A, I can have a NoOps to make a true at the next level. And I can have unstack B, C, so these are the operations that are there.

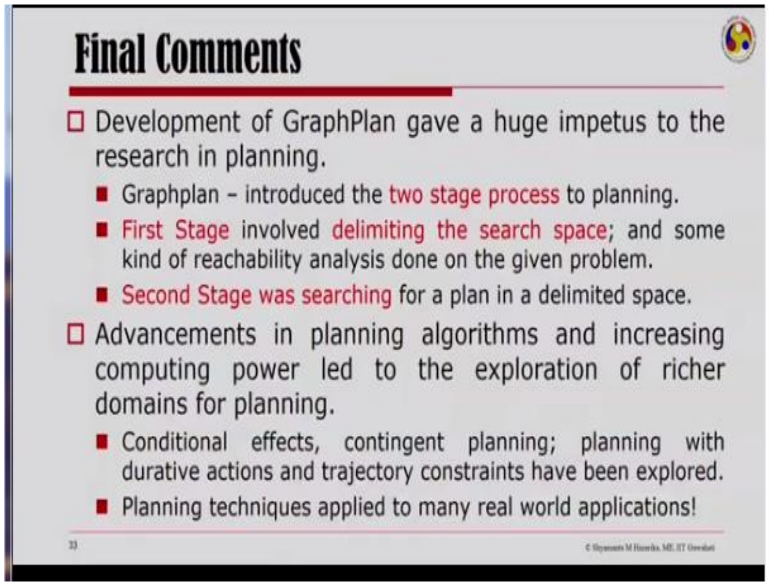
Now when I am looking for a solution in the planning graph, I will only look for the actions and ignore the NoOps because the NoOps are anyway going from each proposition level to the next proposition level and they are there. So I do not need to worry about them but my only lookout would be for actions that are there at each level.

(Refer Slide Time: 43:07)



So for the solution of this problem I have 2 actions, I have to unstack A from B and then I have to unstack B from C to arrive at the final state. So this is the solution to the planning problem that we started with.

(Refer Slide Time: 43:29)



Final Comments

- Development of GraphPlan gave a huge impetus to the research in planning.
  - Graphplan – introduced the two stage process to planning.
  - First Stage involved delimiting the search space; and some kind of reachability analysis done on the given problem.
  - Second Stage was searching for a plan in a delimited space.
- Advancements in planning algorithms and increasing computing power led to the exploration of richer domains for planning.
  - Conditional effects, contingent planning; planning with durative actions and trajectory constraints have been explored.
  - Planning techniques applied to many real world applications!

Now what we have looked in this portion of planning is that we had looked at the graph plan and we have seen that the graph plan is in 2 stage. In fact the development of graph plan give a huge impetus to research in planning. And graph plan introduce the 2 stage process to planning, the first stage involved somehow delimiting the search space and some kind of reachability analysis done on the given problem and the second stage is about searching for a plan in the delimited space.

We have looked at the graph plan algorithm in today's lecture, now there has been tremendous advancement in planning and algorithms an increasing computing power have led to exploration of richer domains of planning. There are planning that involves durative actions trajectory constraints there are plannings that involve contingent planning, conditional effects which we have not looked at as a first step towards planning.

In fact planning techniques have been applied to many real world scenarios and these require number of AI techniques, knowledge representation, search, decision making. We will introduce some of this in the course of our discussion, thank you very much.