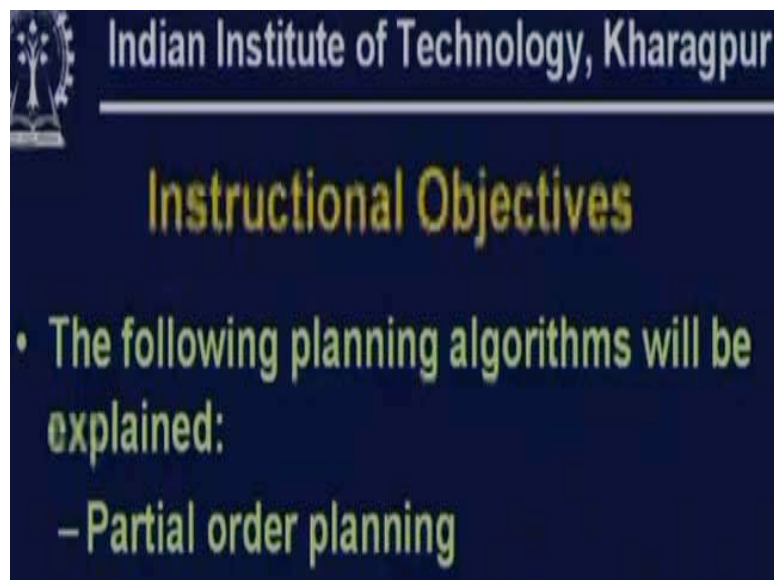


Artificial Intelligence
Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture # 23
Planning - 3

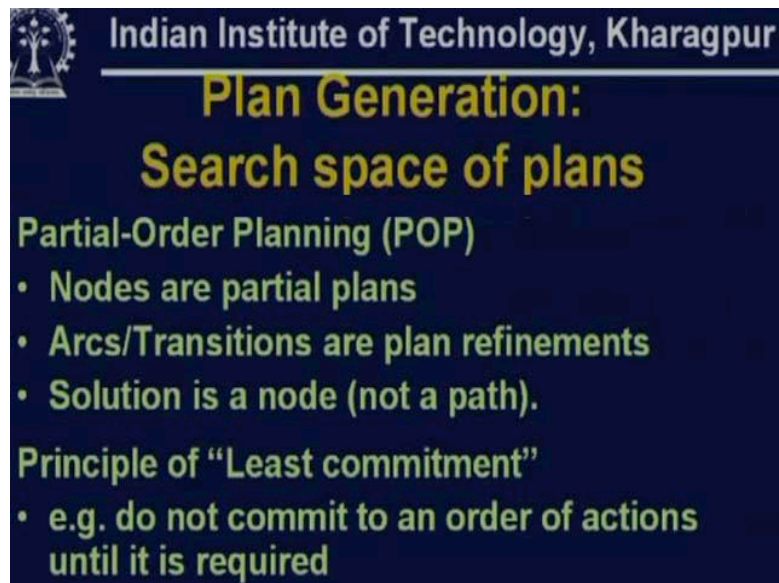
Hello, welcome to the class. Today we will have the third lecture on planning. In the last two classes we looked at planning in Artificial Intelligence and representations for planning problems. We looked at STRIPS representation, the STRIPS algorithm for solving planning problems and we also looked at regression planning and progression planning which involve searching in the state space to find the plans. Today in the third lecture we will explore further the details of Partial Order Planning.

(Refer Slide Time: 01:06)



As I mentioned in the previous class in Partial Order Planning we will be searching in the space of partial plans. And our objective is given a problem situation we want to get a plan which makes least commitment about the ordering of the different actions. So we will represent the plan as actions and the ordering between the actions in order to get the final plan of the problem. In any intermediate stage we will maintain consistent partial plans which may not be complete and we will presently see the details of what I mean by this.

(Refer Slide Time: 02:33)



Indian Institute of Technology, Kharagpur

Plan Generation: Search space of plans

Partial-Order Planning (POP)

- Nodes are partial plans
- Arcs/Transitions are plan refinements
- Solution is a node (not a path).

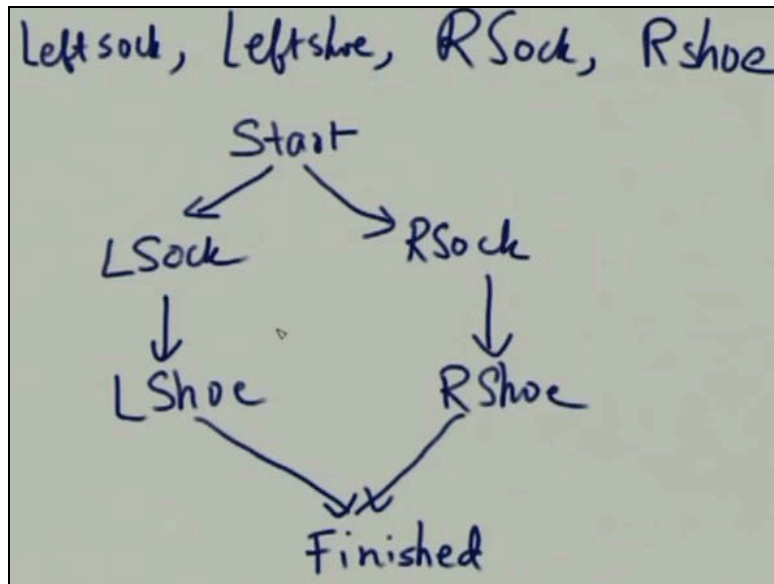
Principle of "Least commitment"

- e.g. do not commit to an order of actions until it is required

In Partial Order Planning the nodes would be partial plans and to get from one partial plan to another partial plan the step is called plan refinement. By taking a partial plan by doing plan refinement we get to another partial plan. And the solution to the planning problem is a partial plan. So the path to the solution is not important because all the relevant information is captured in the partial plan which is a node in this search problem. That is, we obey the principles of least commitment. We do not commit to any ordering of the actions which is not required.

Before we continue let us recall the problem we worked out in the last class of putting on the socks and shoes problem our objective was to put on the left sock left shoe right sock and right shoe. So these are the four conditions that must be satisfied at the goal. And we one cannot put on the right shoe before the right sock is put on or the left shoe before the left sock is put on. And a solution to such a problem would be, after this start the agent can put on either the left sock or the right sock. Only after the left sock the agent can put on left shoe and only after the right sock the agent can put on right shoe. When both left shoe and right shoe are on the plan is done or finished. So this represents a partial plan for the socks and shoes problem.

(Refer Slide Time: 04:33)



The actual plan will of course not be such a partial order structure but it will be a linear structure. From this partial order structure we can get several linearizations. The agent can put on left sock right sock left shoe right shoe or left sock left shoe right sock right shoe or right sock left sock left shoe right shoe or right sock left sock right shoe left shoe and so on. Therefore how many linearizations can be possible given this partial plan. Let us get to certain terminologies that we will be using today.

(Refer Slide Time: 05:29)

Indian Institute of Technology, Kharagpur

Terminology

- **Step:** a step in the partial plan—which is bound to a specific action
- **Orderings:** $s_1 < s_2$ s_1 must precede s_2
- **Open Conditions:** preconditions of the steps (including goal step)
- **Causal Link:** $A_p \xrightarrow{Q} A_c$
a commitment that the condition Q , needed at A_c will be made true by A_p
– Requires A_p to “cause” Q
Should have an effect Q

Handwritten notes: $P_1, P_2, Q, Q_1, Q_2, Q_3$ and A_p achieves Q for A_c

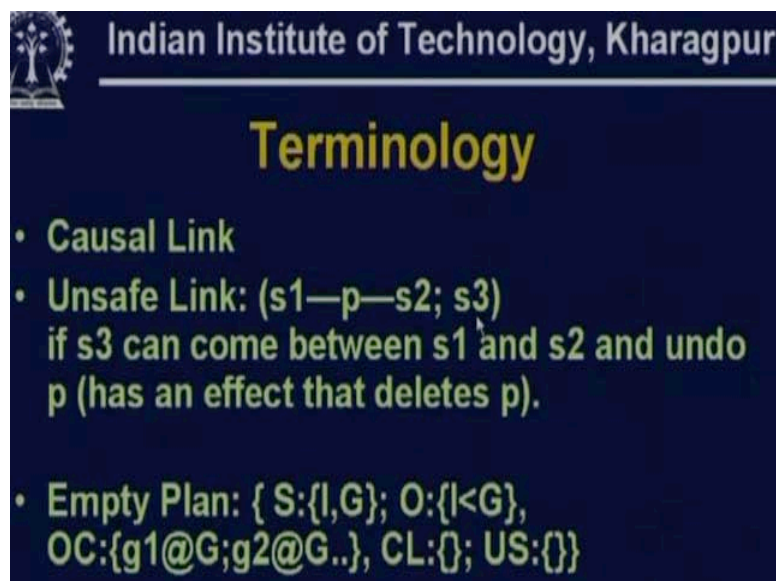
We have a step in the partial plan and a step in the partial plan would be a particular action which is taken. So step is bound to a specific action. Then in a partial plan we will have orderings, orderings of the form s_1 before s_2 which means s_1 must precede s_2 . If s_1 is an action s_2 is another action, action s_1 must come before the action s_2 it may come immediately before the action s_2 or sometimes before the action s_2 . Thirdly let us see what we mean by open conditions. Open conditions are the

preconditions of the steps which are not yet been satisfied. Suppose we have a step and in order to execute the step we need certain preconditions to hold.

As you know, in the STRIPS representation for every action we have preconditions like P_1 and P_2 are preconditions of action (a) and also as you know for every action we also have the add list and the delete list suppose q_1 and q_2 are added and q_3 is deleted. So P_1 and P_2 are the preconditions of the action (a). Open conditions are those preconditions of the step (a) that have not yet been satisfied. Then what is important is we need to know what we mean by causal link.

Now we will say that A_p to A_c on Q is a causal link on Q if the step A_c has a precondition of Q and this precondition is met by the step A_p . If the step A_p supplies the precondition Q which is a precondition of A_c then A_p to A_c is a causal link on the condition Q . So this is a commitment that the condition Q needed by A_c will be made true by A_p . That is A_p must cost A_c . So A_p must have Q as an effect and A_c as Q as a precondition. Now we say that A_p achieves Q for A_c . If in our partial plan A_p achieves Q for A_c what it means is that no action can come between A_p and A_c which deletes Q . So, while developing the partial plan if we want A_p to achieve Q for A_c we must make sure that no action would come between A_p and A_c which deletes Q . There are more terminologies.

(Refer Slide Time: 09:33)



Indian Institute of Technology, Kharagpur

Terminology

- Causal Link
- Unsafe Link: (s_1 — p — s_2 ; s_3)
if s_3 can come between s_1 and s_2 and undo p (has an effect that deletes p).
- Empty Plan: { S:{I,G}; O:{I<G},
OC:{g1@G;g2@G..}, CL:{}; US:{} }

Unsafe link: Suppose there is a causal link s_1 and s_2 on the condition p and then we have another action s_3 . Now if I put s_3 between s_1 and s_2 and s_3 can undo p then s_3 will not be able to come between s_1 and s_2 . So if my plan permits or if my partial plan permits s_3 to come between s_1 and s_2 and s_3 deletes p then this is an unsafe link. This causal link s_1 to s_2 and p along with s_3 together is an unsafe link. So such unsafe links have to be taken care of before we get the final plan. Before we discuss the Partial Order Planning algorithm in order to develop the algorithm we will start with the initial state.

The initial state is an initial partial plan and we will start with this. Empty plan is given by this. In the empty plan we start with an initial state I, a goal state G and we assume that I precedes G. This is an ordering between I and G and at G the preconditions that have to be satisfied are suppose G is g_1 and g_2 so g_1 and g_2 are the open preconditions that must be satisfied at G.

Initial plan: There are no causal links and there are no unsafe links. But this plan is not complete because g_1 and g_2 are the open preconditions. Now let us look at how we will represent partial plans.

A partial plan is represented by a 5-tuple A, O, L, OC and UL. What is A?

A is a set of actions in the plan. A is a set of steps in the plan. So S_0, S_1, S_2 up to S_{inf} let them be the steps in the plan. So plan consists of the set of steps and as I mentioned a partial plan is not necessarily a total order plan. That is it is not just a sequence of steps. We can order the steps according to a partial order. So we have a set of steps and we also specify the set of action ordering. So there is a partial ordering on the actions. S_i precedes S_j denotes that the step S_i must occur before the state S_j in a realization of the plan. So A is the set of steps, O is the set of action orderings and L is the set of the causal links. And as we saw a causal link S_i to S_j on condition p means S_i achieves condition p for S_j . So S_i has p as an effect and p is a precondition that is required by S_j . So, A, O and L specify the partial plan.

(Refer Slide Time: 13:55)

Indian Institute of Technology, Kharagpur

Partial plan representation

$P = (A, O, L, OC, UL)$

A: set of action steps in the plan $S_0, S_1, S_2, \dots, S_{inf}$

O: set of action ordering $S_i < S_j, \dots$

L: set of causal links

$S_i \xrightarrow{p} S_j$

OC: set of open conditions
(subgoals remain to be satisfied)

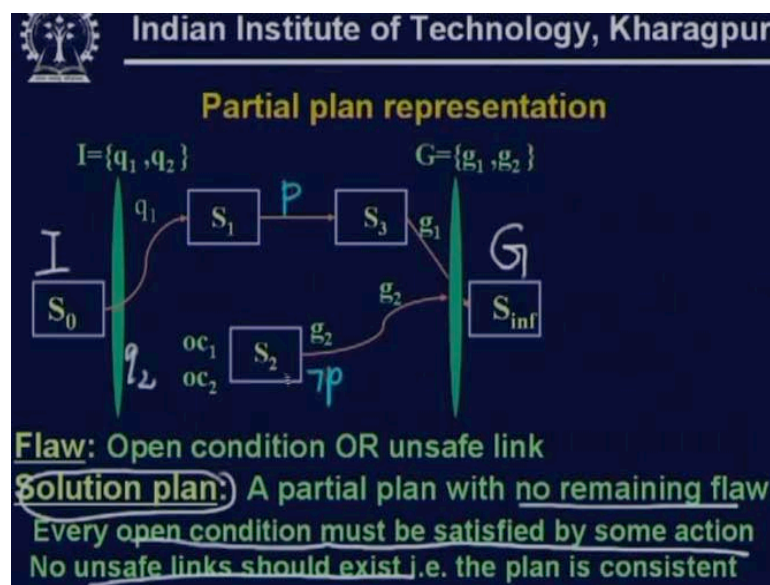
UL: set of unsafe links where p is deleted by some action S_k

$S_i \xrightarrow{p} S_j$

Now in a partial plan we have a set of open conditions which we denote by OC. So OC at the sub-goals represent the sub-goals that remain to be satisfied. They could be the sub-goals of the final goal or they could be the sub-goals of the intermediate state, the preconditions of an intermediate state which have not yet been satisfied. So open condition means those conditions that are not satisfied therefore the plan is not yet complete. Then we have a set of unsafe links. If we have the causal link S_i achieves p for S_j and there is a step S_k which deletes p then S_k cannot come between S_i and S_j . So, if in a partial plan S_k can come between S_i and S_j we have an unsafe link. So what does a plan consist of?

A plan is a 5-tuple so we have three things; A is a set of steps in the plan, O is the set of partial order relations between the steps and L is the set of the causal links and if the plan is not complete what we will have in the plan is a set of open conditions which are the set of sub-goals that have not yet been satisfied and we have some unsafe links those links which consists of causal links and a step which can come between the two steps in the causal link and delete the condition which is required. When we have a partial plan that plan is flawed if there are some open conditions or there are some unsafe links. And the solution plan would be a partial plan which has no remaining flaw. That is, it achieves all the conditions and there is no unsafe links. So in the final solution plan we would like there must be no remaining flaw. That is, every open condition must be satisfied by some action and then no unsafe links should exist.

(Refer Slide Time: 17:55)



Now this is a representation of the partial plan. We introduce two distinguished states S_0 for the initial state and S_{inf} for the goal state. Now we will consider that the initial state description is an effect of the initial step. If in the problem the initial state is q_1 q_2 and the goal desired to be achieved is g_1 g_2 then we will have an initial state S_0 whose effect will be q_1 q_2 . So q_2 is another effect of S_0 . And G or S_{inf} is the goal state and for this goal state it has a set of preconditions. The preconditions are the propositions which are true at the goal state. So the preconditions here are g_1 and g_2 .

In the initial plan that one can draw g_1 and g_2 are the open conditions. This is S_0 and this is $S_{infinity}$ and g_1 and g_2 are the open conditions of G . Now this is a partial plan where this open condition g_1 is achieved by S_3 . So S_3 achieves g_1 for S_{inf} . And let us say S_1 achieves p for S_3 , I achieves q_1 for S_1 . Also, suppose S_2 achieves g_2 for S_{inf} and now suppose S_2 has an effect of $\neg p$ that is S_2 deletes p and S_1 achieves p for S_3 .

Can you tell me if there are any flaws in the plan?

There are two types of flaws in this plan. Firstly S_2 is a step which has two preconditions OC_1 and OC_2 which are not yet satisfied. So OC_1 and OC_2 are the open preconditions of the state S_2 so this is flaw number one. Secondly, the flaw number

two is S_1 precedes S_3 , S_3 precedes G and S_2 precedes G but we have not said anything in this partial plan about the relative ordering of S_1 , S_2 and S_3 . Now if I put S_2 between S_1 and S_3 , so in the linearization if we have S_0 to S_1 , S_1 to S_2 and S_2 to S_3 and then S_3 to done then what will be the problem?

The problem is S_1 will have p as an effect but that p would be deleted by S_2 so S_3 will not get p as a precondition. So because S_3 requires p as a precondition which is supplied as an effect of S_1 we cannot put S_2 between S_1 and S_3 so this is not correct so what can we do?

S_2 can occur either after S_3 or it can occur before S_1 . S_2 which deletes p cannot in between S_1 and S_3 it can come before S_1 or after S_3 . Now let us look at a very basic version of the Partial Order Planning algorithm. Let p be an initial plan.

Now given an initial plan we first find out if the plan has any flaw. So p is a plan we find out if p has a flaw. So we choose a flaw f . The flaw could be an open precondition or it could be an unsafe link. A partial plan may have several flaws, we choose one of them. Eventually we have to remove all the flaws in the planning algorithm. And the order in which we choose the flaws may have a bearing on how fast we solve the problem. But irrespective of which flaw we choose first we can choose the flaws in any order and we will get the same solution. So we have to choose a flaw and depending on the nature of the flaw we have to take certain steps.

Therefore, as we mentioned flaw is of two types; an open condition is a flaw or unsafe link is also a flaw. If we select a flaw which is an open condition then what should we do?

We should choose an action S that achieves this open condition. So if f is an open condition, suppose there is a state S DASH and this state as f as its precondition. Now, in order to correct this flaw we must choose another state S_1 which has f as an effect. If you do this then this open precondition is taken care of even though if other flaws are introduced. Namely if the preconditions of S dash are not satisfied there will be new flaws that are introduced.

Secondly, a flaw could an unsafe link. If the flaw is an unsafe link that is if we find that S_2 deletes f then S_2 cannot come between S_1 and S dash so S_2 can either be promoted to come before S_1 or it can be demoted to come after S_1 . So this is a protected interval between S_1 and S dash where S_2 undoes the effect of f cannot come. So S_2 can be promoted or S_2 can be demoted. Now, after we have done that we must update the partial plan. So, if we promote S_2 then we must add a causal link from S_2 to S_1 .

(Refer Slide Time: 24:43)

Indian Institute of Technology, Kharagpur

Algorithm

1. Let P be an initial plan
2. **Flaw Selection:** Choose a flaw f
(either open condition or unsafe link)
3. **Flaw resolution:**
 - If f is an open condition,
choose an action S that achieves f
 - If f is an unsafe link,
choose promotion or demotion
 - Update P
 - Return NULL if no resolution exist
4. If there is no flaw left, return P
else go to 2.

```
graph TD; S2[S^2] -- Promoted --> S1f[S^1_f]; S2 -- Demoted --> S1[S^1]; S1f -- f --> S1;
```

(Refer Slide Time: 26:44)

Indian Institute of Technology, Kharagpur


Algorithm

1. Let P be an initial plan
2. **Flaw Selection:** Choose a flaw f
(either open condition or unsafe link)
3. **Flaw resolution:**
 - If f is an open condition,
choose an action S that achieves f
 - If f is an unsafe link,
choose promotion or demotion
 - Update P
 - Return NULL if no resolution exist
4. If there is no flaw left, return P
else go to 2.

1. Initial plan:

```
graph LR; S0[S_0] -- g_1 --> Sinf[S_inf]; S0 -- g_2 --> Sinf;
```

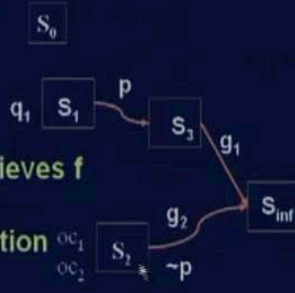

(Refer Slide Time: 26:55)



Indian Institute of Technology, Kharagpur

Algorithm

1. Let P be an initial plan
2. **Flaw Selection:** Choose a flaw f (either open condition or unsafe link)
3. **Flaw resolution:**
 - If f is an open condition, choose an action S that achieves f
 - If f is an unsafe link, choose promotion or demotion
 - Update P
 - Return NULL if no resolution exist
4. If there is no flaw left, return P else go to 2.



```

graph LR
    S0[S0] --> S1[S1]
    S1 -- p --> S3[S3]
    S2[S2] -- g2 --> Sinf[Sinf]
    S2 -- ~p --> S3
    S3 -- g1 --> Sinf
    S1 -- q1 --> S1
    S2 -- OC1 --> S2
    S2 -- OC2 --> S2
    Sinf -- g1 --> Sinf
    Sinf -- g2 --> Sinf
    
```

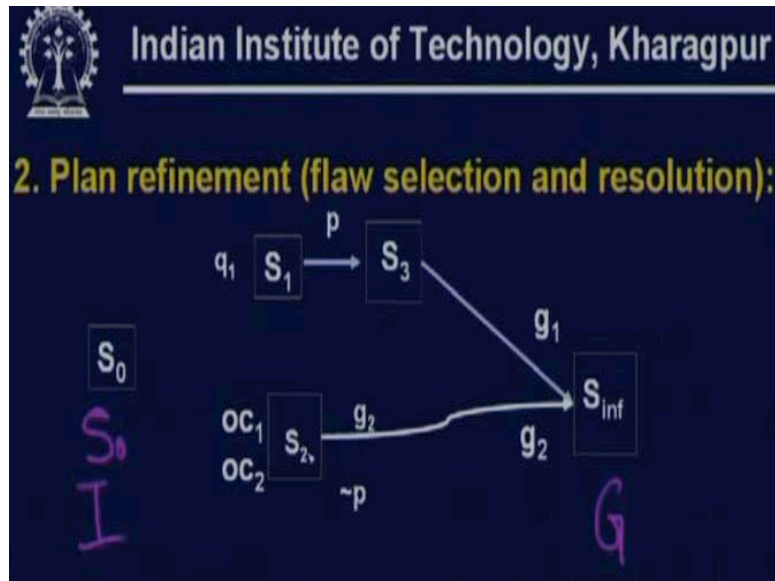
If I put S_2 after S_1 then we must add a precedence link from S_1 to S_2 . And while doing this if we find that there is no operator by which we can resolve the conflict then we return now. Therefore this planning algorithm does not succeed. But if it does succeed and if we come to a point where there is no flaw left then we will return the current plan otherwise if there are more flaws left we will go to step two and select the next flaw. We start with an initial partial order plan which consists of the initial state I, the goal state G. The goal state has the desired propositions given as a goal as its open precondition and the initial state has the initial state description as its effect.

Now our algorithm will choose a flaw. A flaw is either an open condition or it is an unsafe link. If the flaw that we have chosen is an open condition we must choose an action S that achieves f. If f is an unsafe link then we will choose to promote or demote the current step in the plan. And once we have done this we update the current plan. We return null if no resolution exist and if there is no flaw left we return p else we go back to state two and continue.

This is the initial plan, S_0 is the initial state, S_{inf} is the final state, g_1 and g_2 are the open conditions of S_{inf} . This is an intermediate partial plan. Here we have S_{inf} , S_{inf} has the preconditions g_1 and g_2 , S_2 achieves g_2 , S_2 deletes p, S_3 achieves g_1 for S_{inf} , S_1 achieves p for S_3 , S_1 has q_1 as its prerequisite, S_2 has OC_1 and OC_2 as its prerequisites. So, if I take this open condition I must add a step which has OC_1 as its effect. If I look at this causal link from S_1 to S_3 on q and this node S_2 if we put S_2 between S_1 and S_3 then we can correct the flaw. Therefore this process of flaw selection and flaw resolution is known as plan refinement. This is an example of a partial plan.

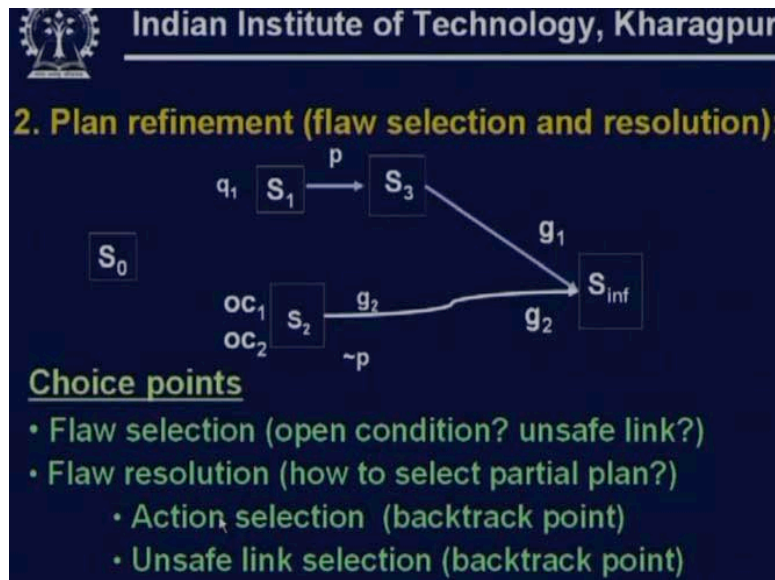
Apart from the initial state S_0 and the final state, this is the initial state and the final state S_{inf} we have three other steps S_1 , S_2 and S_3 . S_3 achieves g_1 , S_2 achieves g_2 , S_2 deletes p, S_1 achieves p which is required for S_3 . S_1 has a precondition q_1 and S_2 has two preconditions OC_1 and OC_2 . From this example let us find out the flaws in this partial plan.

(Refer Slide Time: 28:18)



Now there are several flaws in this partial plan. Firstly, there are three open preconditions. S_1 has the condition q_1 , S_2 has the condition OC_1 and OC_2 . Secondly, there are some unsafe links. S_2 cannot come between S_1 and S_3 because S_2 deletes p which is required by S_3 .

(Refer Slide Time: 29:11)



For flaw selection we have several choices. The choices are the open conditions in the plan and the unsafe link. The flaw resolution deals with how we select partial plans. Here we can select a particular action and treat this as a backtrack point. Or even for finding the unsafe links we can use backtrack search. **So this is the repetition of what I said earlier.**

A partial plan is described by the tuple A, O, L . A is a set of actions in the plan, O is a set of temporal orderings of the form A precedes b and L is the causal linking actions via a literal. This is an example of a causal link.

(Refer Slide Time: 30:40)

Indian Institute of Technology, Kharagpur

Threats to causal links

Step A_t threatens link (A_p, Q, A_c) if:

1. A_t has $(\neg Q)$ as an effect, and
2. A_t could come between A_p and A_c , i.e. $O \cup (A_p < A_t < A_c)$ is consistent

What's an example of an action that threatens the link example from the last slide?

(Refer Slide Time: 30:45)

Indian Institute of Technology, Kharagpur

Partial Plan Representation

- Plan = (A, O, L) , where
 - A : set of actions in the plan
 - O : *temporal orderings* between actions ($a < b$)
 - L : *causal links* linking actions via a literal
- Causal Link:
 - Action A_c (consumer) has precondition Q that is established in the plan by A_p (producer).

Diagram: $A_p \xrightarrow{Q} A_c$

Example: $\text{move-a-from-b-to-table} \xrightarrow{(\text{clear } b)} \text{move-c-from-d-to-b}$

Handwritten: $\begin{matrix} A \\ B \end{matrix} \rightarrow \text{move-c-from-d-to-b}$

A_p achieves Q for A_c . Action A_c has precondition for Q that is established by the plan by A_p . Another example; when one moves a from b to the table the effect is to clear b and clear b is required to move c from d to b . Now, when we have an unsafe link as we noted there is a step which can come between a protective link. So we say that a step A_t threatens a link A_p achieves Q for A_c if A_t has NOT Q as an effect.

Secondly, if A_t could come between A_p and A_c , that is if according to the partial plan we can put A_t between A_p and A_c then in both these places there is a problem with the partial plan it is not fully correct. Let us look at the initial plan. For uniformity we

represent the initial state and the goal state with two special actions. A_0 is a special action or S_0 which has no precondition which has initial state as effects and they must be of first step in the plan. A, O, D has no effects, goals are preconditions and must be the last step in the plan process.

(Refer Slide Time: 32:18)

Indian Institute of Technology, Kharagpur

POP algorithm

```

POP((A, O, L), agenda, actions)
  If agenda = () then return (A, O, L)
  Pick (Q, aneed) from agenda
  aadd = choose(actions) s.t. Q ∈ effects(aadd)
  If no such action aadd exists, fail.
  L' := L ∪ (aadd, Q, aneed) ; O' := O ∪ (aadd < aneed)
  agenda' := agenda - (Q, aneed)
  If aadd is new, then A := A ∪ aadd and
  ∀P ∈ preconditions(aadd), add (P, aadd) to agenda'
  
```

Now let us look at the Partial Order Planning algorithm in a slightly greater detail by seeing how we proceed. The Partial Order Planning problem P O P stands for POP, takes three arguments A, O, L. A is a set of steps, O is a set of temporal orderings between steps and L are causal links linking actions with a liberal. So the P O P algorithm takes A, O, L as the argument. In addition it takes two other arguments agenda and the actions.

Agenda are the set of flaws and actions are the set of actions that the agent can take. If agenda is empty then return A, O, L so we have called P O P or Partial Order Planning on A O L has also agenda and actions. If agenda is null then we are done we return A O L, if agenda is null then we return minus A O L, add equal to chooses actions such that O is using effects. So, initially if the agenda is empty we return A O L otherwise we pick up a flaw from the agenda. Let the flaw be (Q, a need). Now in order to satisfy this flaw we must choose an action such that this action requires Q. So we must pick up an action which adds Q.

We have to have an action which adds Q so aadd is such an action. If no such a exists then we return failure. Otherwise we put L dash is equal to L union aadd where aadd is the new step we have obtained and Q is the something that aadd achieves and aneed is something that Q needs. So, in other words you have an action aneed. This step aneed has a precondition of Q which as not yet been achieved. So we find a step aadd which has Q as its effect. Now in this case we can say that aadd achieves Q for aneed. Now aadd is a new step because it is a new step then we must add the preconditions of aadd which are not satisfied to the agenda.

Now how has the plan data structure changed?

First of all in the plan data structure we add a causal link from aadd to aneed on Q. Secondly, we update the partial ordering relations by adding this new relation aadd which must come before aneed. Then the new agenda is, that is one entry we have removed from the agenda is (Q, aneed) because the open precondition for Q has been achieved in this state so we remove it from the agenda. If aadd is new then A equal to A union aadd. That is, if we have got a new step then I will add it to my action list. And then for all preconditions of aadd we must add (P, aadd) to the agenda. Therefore we have taken care of open preconditions and next we have to take care of unsafe links.

Now for every action a_t we check if it threatens any causal link. If it does threaten any causal link we must put the action a_t either before a_p or after a_c . So, in this diagram you have a_p which achieves a_c and it achieves it on the state variable Q. Now when we want to bring in a new step I must either bring it before a_t or I must bring it after a_c . Finally if the search succeeds at all points and we have an agenda which is empty then we get the answer.

(Refer Slide Time: 38:29)

Indian Institute of Technology, Kharagpur

POP algorithm

For every action a_t that threatens any causal link
 (a_p, Q, a_c) in L'
 choose to add $a_t < a_p$ or $a_c < a_t$ to O .
 If neither choice is consistent, fail.
 POP((A', O', L') , agenda, actions)


```

    graph LR
      at[at] --> ap[ap]
      ap -- Q --> ac[ac]
      at -.-> ac
  
```

```

    graph LR
      ap[ap] -- Q --> ot[ot]
      ot --> at[at]
  
```


(Refer Slide Time: 38:33)



Indian Institute of Technology, Kharagpur

POP algorithm

```

POP((A, O, L), agenda, actions)
  If agenda = () then return (A, O, L)           Termination
  Pick (Q, aneed) from agenda                   Goal Selection
  aadd = choose(actions) s.t. Q ∈ effects(aadd) Action Selection
  If no such action aadd exists, fail.
  L' := L ∪ (aadd, Q, aneed) ; O' := O ∪ (aadd < aneed)
  agenda' := agenda - (Q, aneed)              Update goals
  If aadd is new, then A := A ∪ aadd and
  ∀ P ∈ preconditions(aadd), add (P, aadd) to agenda'
  For every action at that threatens any causal link (ap, Q, ac) in L'
  choose to add at < ap or ac < at to O.    Protect causal links
  If neither choice is consistent, fail.        - Demotion: at < ap
  POP((A', O', L'), agenda, actions)          - Promotion: ac < at
  
```

This is the same algorithm written in a small fashion. Initially we are given a Partial Order Planning problem consisting of A, O, L agenda and actions. If agenda is empty initially then we return the current plan A, O, L otherwise we pick (Q, a_{need}) from agenda this is an empty condition. Q is an empty precondition of a_{need} so we select our current goal then we choose an action which achieves this effect Q. If there are several such actions this is a choice point. If there is no such action of course the search will fail and then we say L' is equal to L union a_{add} (Q, a_{need}). So L is the set of causal links. We add the causal need from a_{add} to a_{need} on Q to L. So this is a_{add}, this is a_{need} and this is Q.

Now O' will be O' is the set of precedence constraints, I add a_{add} comes before a_{need}. From agenda I remove (Q, a_{need}). If a new step is added then I union it with the step list and for each preconditions of the new state we add it to agenda. For every action a_t that threatens any causal link we choose to add that a_t is before a_p or a_c is before a_t etc if neither choice is consistent fail. Now this is the same algorithm. Initially I pick a goal then I choose one of the actions then corresponding to the action I update the goals that is I remove a_{need} from the agenda then I add a new step I find the preconditions of the new step.

(Refer Slide Time: 41:18)

Indian Institute of Technology, Kharagpur

POP algorithm

If a_{add} is new, then $A := A \cup a_{add}$ and
 $\forall P \in \text{preconditions}(a_{add})$,
add (P, a_{add}) to agenda'

For every action a_t that threatens any causal link (a_p, Q, a_c) in L'
choose to add $a_t < a_p$ or $a_c < a_t$ to O .
If neither choice is consistent, fail.

POP((A', O', L') , agenda, actions)

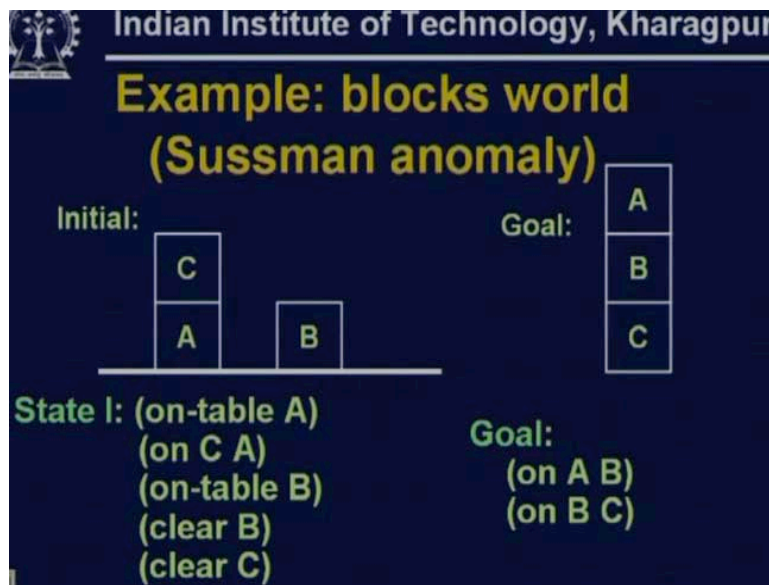
Protect causal links
Demotion: $a_t < a_p$
Promotion: $a_c < a_t$

$a_t \rightarrow a_p \rightarrow a_c$

Suppose we find that there is a causal link from a_p to a_c on Q and if we find that there is an action a_t which threatens a_p and a_c . We say a_t must occur before a_p or a_t must occur after a_c . If a_t cannot occur before a_p or after a_c if they are not consistent then we fail. And then we recursively call P O P on the action list on the causal link list and on the goal list. So, as we can see the Partial Order Planning problem is sound and complete. The P O P plan is a solution if all the preconditions are supported by causal links that is there are no open conditions, there are no threats to any causal link and we have a consistent temporal ordering and by construction the P O P algorithm reaches a solution plan.

Now we will illustrate this P O P algorithm with a very simple example which is the blocks world example of the Sussman anomaly which we have discussed earlier. In this problem the initial state consist of three blocks C A and B placed as follows A and B on the table C is on top of A. The goal is to get the three blocks stacked like this: A on top of B, B on top of C. Before we do our P O P algorithm on this problem Let us look at the representation of the initial state and the goal. The goal is represented by A is on B and B is on C.

(Refer Slide Time: 43:02)



Note that we mentioned we can represent a goal by a subset of the conditions that hold. An initial state is specified by the exact behavior of all the variants. In this case on table A and on C A and clear C on table B and clear B is the description of the initial state and ((on A B)) (on B C) is the description of the goal state.

Example of Sussman's Anomaly:

A_0 is the initial state or I and A_{inf} is the goal state or G. It has two open preconditions ((on A B)) and (on B C). So what are the flaws in this plan?

I have these as the effects. So flaws are there are two open preconditions. Now what should we do?

We will try to take one of the preconditions and try to satisfy it. In the first step if we have an action or step A_1 which involves moving B from table to C, if I move B from table to C what do we achieve?

We achieve the precondition (on B C). (On B C) is an effect of the action move B from table to C and (on B C) is required by the goal. So we have a causal link on (on B C).

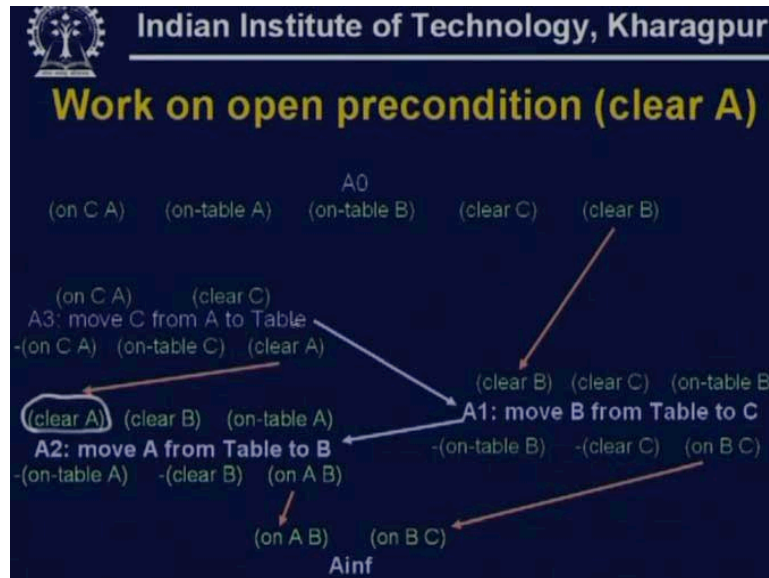
Now let us see what the open conditions are and what flaws are there in this partial plan. There are several open preconditions; ((on A B)) is open, clear B is achieved in the initial state but here it is open, clear C is achieved in the initial state but here it is open, on table B is achieved by the initial state. Now what we will do is we will take up one of these open conditions. Let us take up clear B and we see that clear B is satisfied at the initial state so we say that A_0 achieves clear B for A_1 . Next we look at the open condition (on A B) on this sub-goal. Now, to achieve (on A B) we introduce a second step A_2 which says move A from table to B. Now move A from table to B as preconditions clear A clear B on table A and effects not on table A not clear B (on A B). Now this (on A B) achieves this (on A B).

What are the open conditions?

Clear A clear B on table A. Now we notice that A_0 achieves clear B for A_1 but A_2 removes clear B. So clear B is required by A_1 and this clear B is achieved by A_0 . Now

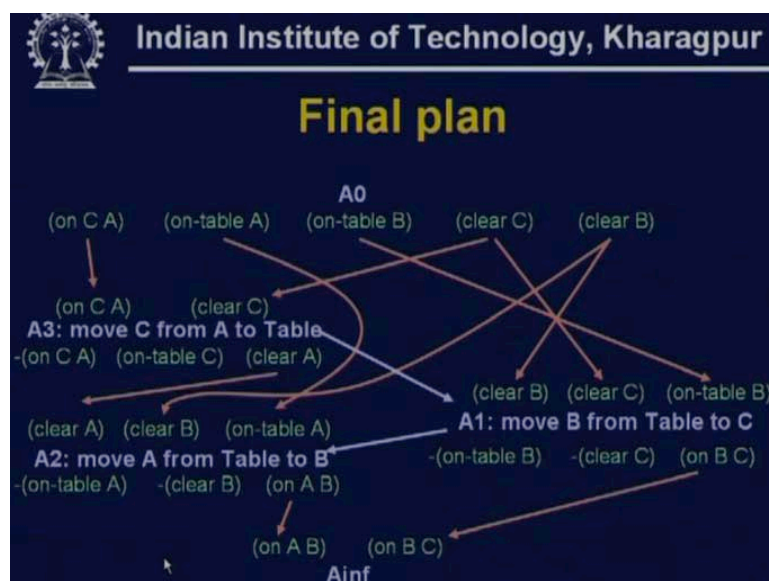
this step A_2 deletes clear B so A_2 cannot come between A_0 and A_1 . Therefore A_2 must come before A_0 or after A_1 . A_0 is the special first state A_2 cannot come before A_0 so A_2 must come after A_1 . So recognize this fact we add an ordering constraint from A_1 to A_2 . So A_0 is followed by A_1 , A_1 is followed by A_2 , A_1 is followed by A_{inf} and A_2 is also followed by A_{inf} . This is what this particular partial plan represents.

(Refer Slide Time: 47:55)



Now we pick up the open precondition clear A which is required by A_2 . Now we introduce an action A_3 that achieves clear A. What is the action A_3 ? Move C from A to table. So, if you move C from A to table then A becomes clear. The other effects are not (on C A) on table C and clear A. And the preconditions are (on C A) and clear C. So this is another step A_3 . We have introduced three different steps in our plan A_1 , A_2 and A_3 . The pink arrows are the causal links between the states and the white arrows are the ordering links between the states.

(Refer Slide Time: 48:44)

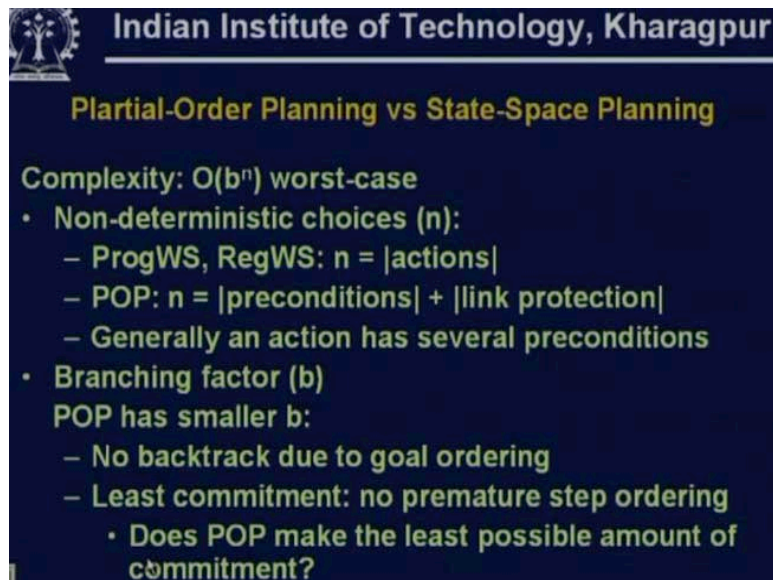


Finally we see that (on C A) is achieved in the initial state, clear C is achieved in the initial state that is for action A_3 . For the action A_1 clear B, clear C, on table B are all achieved in the initial state. For action A_2 clear A is achieved by A_3 , clear B by initial state, on table A by initial state and so on. And we finally find that the ordering relations are A_3 then A_1 then A_2 . Why it is A_3 then A_1 ? Actually A_3 and A_1 can come in any order. A_1 deletes clear C, A_3 requires clear C so we cannot put A_1 between A_0 and A_3 . So to protect this unsafe link we must add a ordering constraint from A_0 to A_3 . So finally we get a plan A_0 to A_3 to A_1 to A_2 then to A_{inf} which is a plan which meets this goal. So the steps are; move C from A to table. If you recollect the planning problem the goal was A on top of B, B on top of C. **And in the initial state I think we had C on A and B on the table.**

Now what are the steps?

First move C from A to table, then move B from table to C, move A from table to B and then we are done.

(Refer Slide Time: 50:33)



Indian Institute of Technology, Kharagpur

Partial-Order Planning vs State-Space Planning

Complexity: $O(b^n)$ worst-case

- **Non-deterministic choices (n):**
 - ProgWS, RegWS: $n = |\text{actions}|$
 - POP: $n = |\text{preconditions}| + |\text{link protection}|$
 - Generally an action has several preconditions
- **Branching factor (b)**
POP has smaller b:
 - No backtrack due to goal ordering
 - Least commitment: no premature step ordering
 - Does POP make the least possible amount of commitment?

Now in summary, if we compare Partial Order Planning versus state space planning the complexity of a search problem is $O(b^n)$ in the worst case where b is the branching factor and we have non-deterministic choices. Now in progression planning and regression planning the depth of the plan is the number of actions in the plan. In P O P the depth is the number of steps that we have to take to satisfy the flaws.

What are the flaws?

The flaws are the open preconditions and the unsafe links. Therefore n is related to the number of preconditions of the steps and the number of link protection that we require.

And generally an action has several preconditions. Now, if we look at the branching factor P O P usually has the smaller value of the branching factor. But the main advantage of P O P is that we have no backtrack due to goal ordering. When we have to satisfy an open condition there could be more than one action which satisfies the

precondition so there is a choice point so there could be some backtracking on deciding which action to use to achieve this precondition.

But if we have two sub-goals g_1 and g_2 let us say (on A B) and (on B C) we do not have to backtrack on which one we should do first, whether we should do (on A B) or whether we should do (on B C). We can do both separately and then while deciding whether we should take action first or this action first we look at what conflicts they generate and we look at the unsafe links. Therefore there is no backtracking due to goal ordering. In Partial Order Planning as we have noted we have this list commitment that is we do not do any premature step ordering. Usually the search involved is less if we do Partial Order Planning.