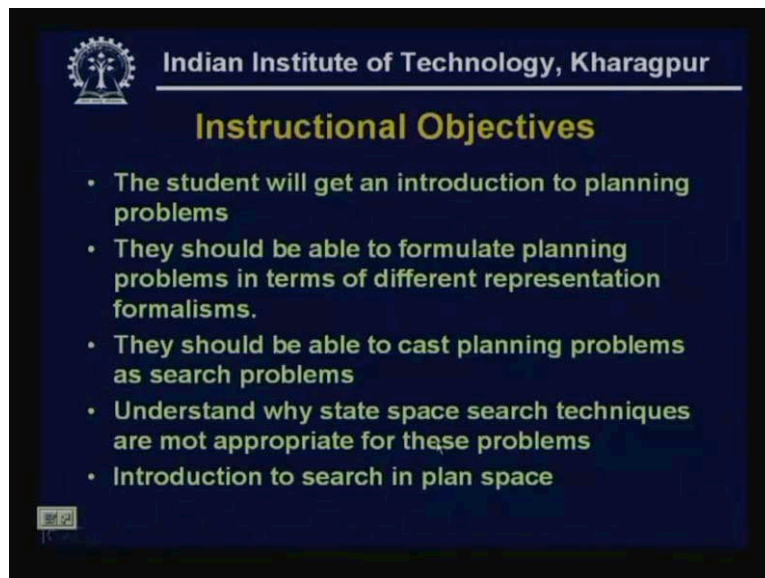



**Artificial Intelligence**  
**Prof. Sudeshna Sarkar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture - 21**  
**Planning - 1**

Today we will start on a new module. Earlier we looked at state space search and we also looked at logic; propositional calculus and first order calculus. Today we will look at a new module which comprises planning problems. So today we will give an introduction to the planning problem. And in the subsequent lectures we will look at different algorithms which efficiently solve the planning problem.

(Refer Slide Time: 01:15)



 Indian Institute of Technology, Kharagpur

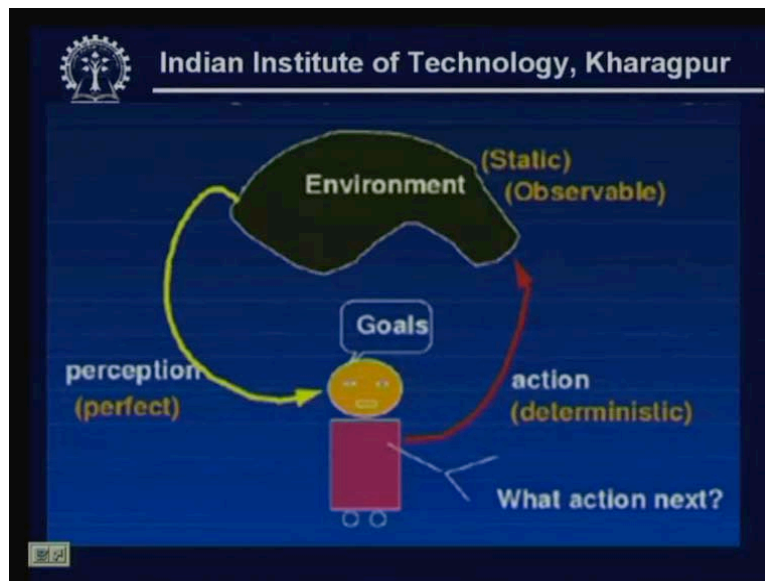
---

**Instructional Objectives**

- The student will get an introduction to planning problems
- They should be able to formulate planning problems in terms of different representation formalisms.
- They should be able to cast planning problems as search problems
- Understand why state space search techniques are not appropriate for these problems
- Introduction to search in plan space

The instructional objective of today's class is the student will get an introduction to planning problems. Given a situation which ((models)) planning they should be able to formulate the planning problems in terms of representation formalisms that we will discuss. Today we will look at the STRIPS formalisms as well as situation calculus and certain other techniques. The students should be able to cast planning problems as search problems and they should be able to cast it as a normal state space search problems. But we will also explore why state space search solution techniques are not ideally suited for solving planning problems and we will discuss what other techniques can be used for searching in the planned space.

(Refer Slide Time: 02:35)

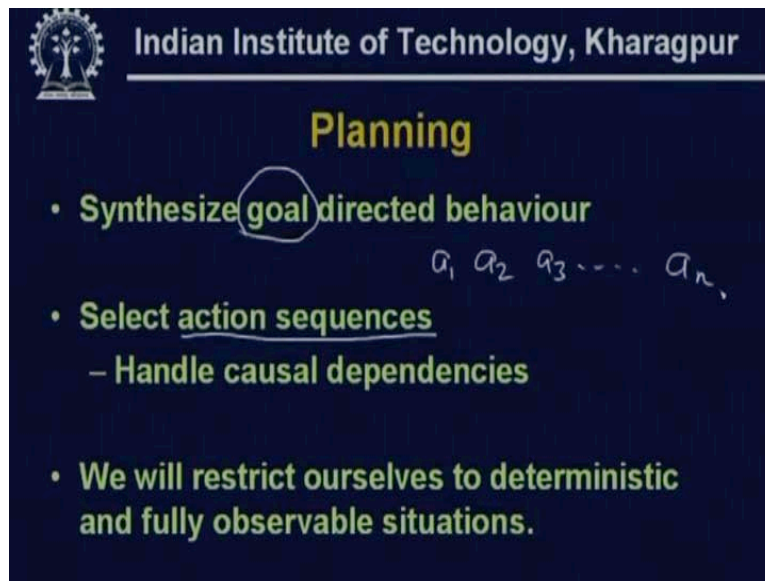


This is a schematic overview of the intelligent agent that we discussed in the beginning of this course. If you remember this is the agent, this is the environment, the agent interacts with the environment. At least for this course we will assume that the environment is static and the environment is fully observable. The agent receives perceptions from the environment through its sensory operators and we assume that this perception is perfect. That is, the environment is fully observable to the agent and the agent can take actions through its effectors which can change the state of the environment.

We will consider only deterministic actions. In the planning problem we will assume that the agent has a goal or goals that the agent wishes to achieve. And given the goals the agent wants to find a sequence of actions that would let him or her achieve the goals. And the sort of question we will ask is given the current situation what is the next action or what is the next action sequence that agent should take?

In planning problems the task of the agent is to synthesize goal directed behavior.

(Refer Slide Time: 04:18)



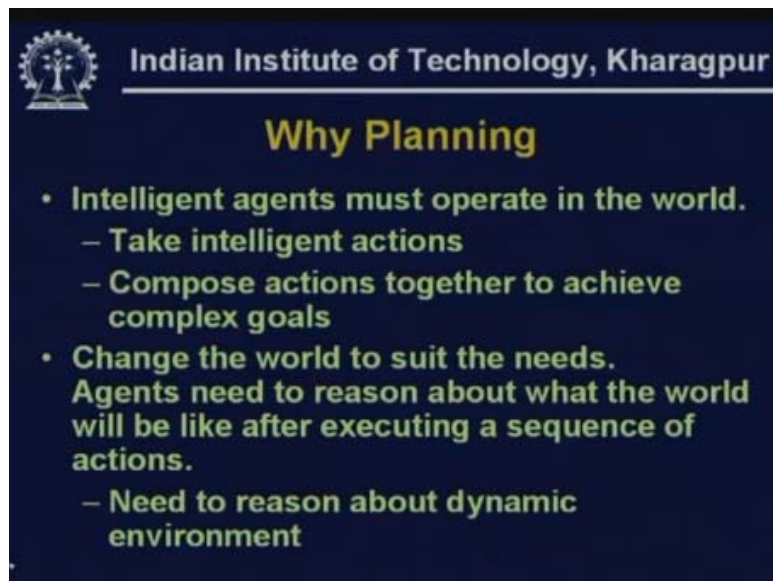
The slide features the IIT Kharagpur logo and name at the top. The title 'Planning' is centered in yellow. Below it, a list of bullet points is shown in green text. The first bullet point, 'Synthesize goal directed behaviour', has the word 'goal' circled in white. To its right, the sequence  $a_1, a_2, a_3, \dots, a_n$  is written in white. The second bullet point is 'Select action sequences', with 'action sequences' underlined. Below it is a sub-point: '- Handle causal dependencies'. The third bullet point is 'We will restrict ourselves to deterministic and fully observable situations.'

And the agent has a goal which the agent wishes to achieve and the objective of the agent is to select a sequence of actions. Let us say  $a_1 a_2 a_3$  up to  $a_n$  so that as a result of executing the actions in this sequence the agent will get to a state where the goal is achieved. And while determining the action sequence the agent would have to consider the casual dependencies between the different actions. For our planning lectures we will primarily restrict ourselves to deterministic and fully observable environments.

There are other types of planning problems where the environment may be stochastic, the environment may be only partially observable, the actions may have non-deterministic effects and in stochastic environment it may not be possible or it may not be the best situation for the agent to plan a sequence of actions. Rather the agent might like to determine a policy so that he can determine his current action and based on the feedback received for the next state the agent can decide the subsequent actions. But for the deterministic and fully observable situations to which we will confine ourselves today the solution to the planning problem would be a sequence of actions that achieve the goal.

Now let us try to understand why planning is important and where planning situations arise. This course is about intelligent agents and intelligent agents need to operate in the world and to operate in the world the agent should be able to take actions. The agent should be able to take actions which are appropriate for the situation that is at hand. The agent should also be able to take actions that change the state of the world so that the goals the agent wishes to achieve can be achieved in the world. And additionally the agent will like to compose more than one action or a set of actions together to achieve complex goals.

(Refer Slide Time: 06:55)



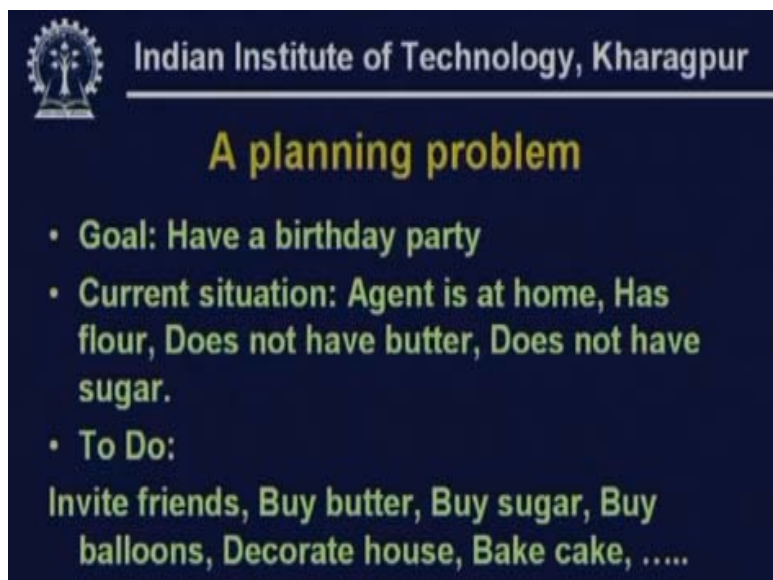
**Indian Institute of Technology, Kharagpur**

### Why Planning

- **Intelligent agents must operate in the world.**
  - Take intelligent actions
  - Compose actions together to achieve complex goals
- **Change the world to suit the needs.**  
Agents need to reason about what the world will be like after executing a sequence of actions.
  - Need to reason about dynamic environment

In order to change the world to suit the needs agents will need to know the behavior of the world they need to know how the world changes as a consequence of the different actions that can be taken in the world. And reasoning from this the agent should be able to formulate a plan to achieve its goals. And in this case while dealing with planning the agent needs to reason about dynamic environments because as the agent is taking an action the action is usually changing the state of the world. Therefore in the dynamically changing world the agent has to reason about the effects of its action sequence.

(Refer Slide Time: 07:54)



**Indian Institute of Technology, Kharagpur**

### A planning problem

- **Goal: Have a birthday party**
- **Current situation: Agent is at home, Has flour, Does not have butter, Does not have sugar.**
- **To Do:**  
Invite friends, Buy butter, Buy sugar, Buy balloons, Decorate house, Bake cake, .....

Let us look at a very simple example of a planning problem that we have to often tackle. Actually a lot of problems that we tackle in our everyday activity or in the long term- goals are planning problems. For example, suppose my plan is to have a birthday party in my house tomorrow and suppose the current situation is that I am at

home, there is flour at home, there is no butter, there is no sugar and I want to find a sequence of actions or plan so that I can hold a birthday party tomorrow. And this may be the sequence of actions that I wish to perform that would lead to the realization of that code.

I invite friends, I go out and buy butter, I go out and buy sugar, I buy balloons, I decorate the house, I bake the cake and so on. So this is a sequence of actions I carry out which realizes the objective I have in mind. Agents are robots and often they need to plan. Their planning is involved in a lot of everyday activities that we do. For example, if you want to get to this room I have to plan how I would come to this place starting from my office or if I want to deliver this course I have to plan the sequence of lecture materials that I will deliver to achieve the objective of teaching this course. So planning is a part of everyday activities like communication, like navigation as well as many other activities. Let us look at some applications of planning; mobile robots.

(Refer Slide Time: 10:00)



Indian Institute of Technology, Kharagpur

## Applications

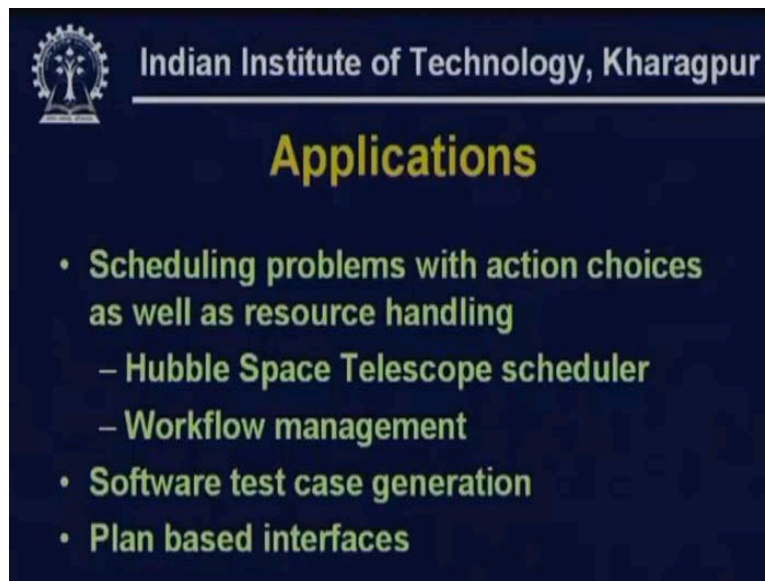
- **Mobile robots**
- **Autonomous agents**
  - NASA Deep Space planning agent
- **Simulated environments**
  - Goal-directed agents for training or games
- **Web and grid environments**
  - Composing queries or services

In fact mobile robots were one of the initial goals of AI and mobile robots need to do planning on a very large scale for navigation, for task planning etc. Hence different types of planning are involved for mobile robots. Autonomous agents also need to planning. For example, NASA uses a deep space planning agent. We have earlier discussed about the different rovers that have been sent to space to Mars and these agents are involved in autonomous planning. In simulated environments goal directed agents are needed for training, for playing games, in web and grid environments planning is needed for composing queries or services, to deliver a particular service.

In education or tutoring system planning is required to decide what material to deliver next to the student. Planning is also used in scheduling problems where we have to make choices of action as well as we have to handle different resources. Hubble Space Telescope scheduler is an example of a very useful system which uses planning.



(Refer Slide Time: 11:18)



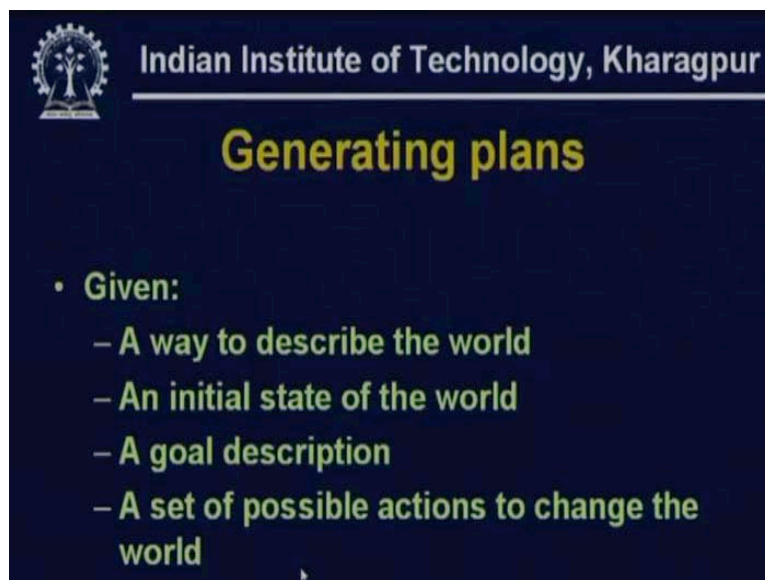
Indian Institute of Technology, Kharagpur

## Applications

- **Scheduling problems with action choices as well as resource handling**
  - Hubble Space Telescope scheduler
  - Workflow management
- **Software test case generation**
- **Plan based interfaces**

Actually now-a-days planning and scheduling go hand in hand. Workflow management is a place where we need planning. Planning is being used in software test case generation, in plan based interfaces and many such applications. So let us see the essence of a planning problem. In a planning problem we are given a way to describe the world we are given a description of the initial state of the world and we are given the description of what the goal state should achieve.

(Refer Slide Time: 11:44)



Indian Institute of Technology, Kharagpur

## Generating plans

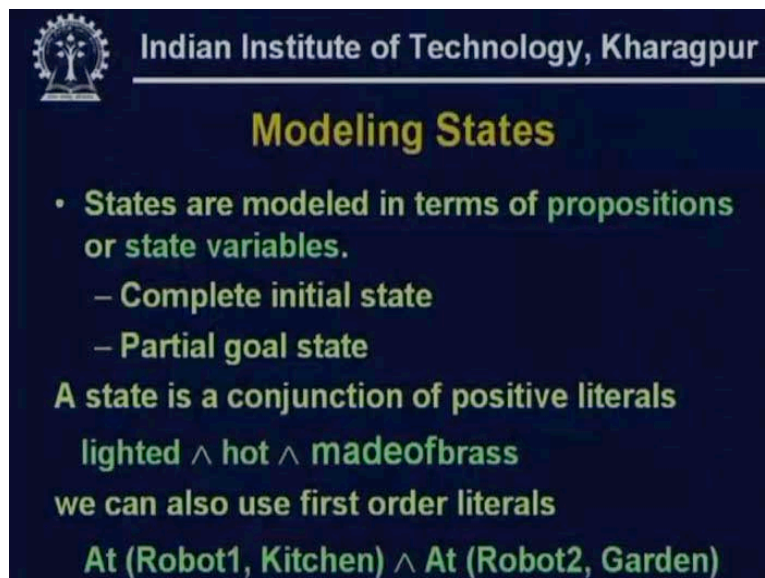
- **Given:**
  - A way to describe the world
  - An initial state of the world
  - A goal description
  - A set of possible actions to change the world

Our objective is to find a set of possible actions which when executed in sequence will change the world so that a goal can be achieved. Or rather we are given the initial state description, the goal description and the description of a set of possible actions that are there in the world and we also know the effects of these actions. And our objective is to find a sequence of actions or prescribe a sequence of actions that will change the initial state and take the agent to a state where the goal will be achieved.

There is a lot of logical structure in the way a planning problem is formulated. And our objective in designing planning algorithms is to design algorithms that take advantage of the logical structure of the problem description and in order to be able to do so we need to be able to represent the problem in a suitable logical language.

Following a very common formalism of representation of planning problems let us see how the states, the actions, goals etc are represented. So, in a planning problem typically states are modeled in terms of propositions or state variables or features of a state. So we have different propositions and in a particular state these propositions have different binary values. For example, suppose the propositions are; at home, have sugar, have butter, have money etc and the values of these propositions may be true or false in a given state. For the initial state we have a complete description of the state, we know which are the propositions which are true in the initial state. And typically we have a partial description of the goal state.

(Refer Slide Time: 13:18)



The slide features the IIT Kharagpur logo and name at the top. The title 'Modeling States' is in yellow. The main content is in green text on a dark blue background, listing key concepts about state modeling.

Indian Institute of Technology, Kharagpur

### Modeling States

- States are modeled in terms of propositions or state variables.
  - Complete initial state
  - Partial goal state

A state is a conjunction of positive literals

lighted  $\wedge$  hot  $\wedge$  madeofbrass

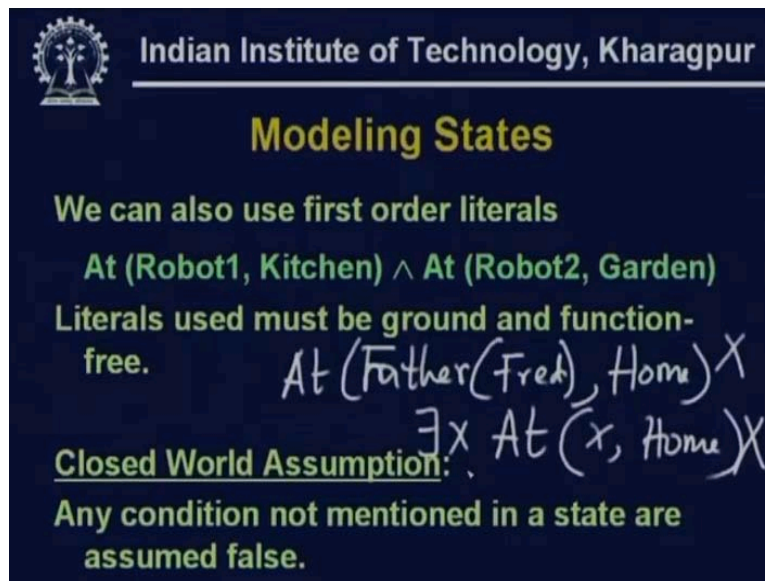
we can also use first order literals

At (Robot1, Kitchen)  $\wedge$  At (Robot2, Garden)

We want to know those propositions we want should be true in the goal state. So a state is typically represented as a conjunction of positive literals. So we have to specify the propositions which are true in a particular state this will fully represent the description of a state. For example, a state could be lighted and hot and made of brass. We can use propositions as a conjunction of propositions to represent a state.

We can also use first order literals. For example, we could use at robot 1 kitchen and at robot 2 garden. Now notice that this is our predicate, at is a predicate but the arguments are ground literals. So we allow first order literals to be used to describe a state but these literals must be ground literals and they must be functioned free. So literals must be ground and function free which means that we cannot use things like at father Fred home. These sort of first order literals are not allowed because here we have a function.

(Refer Slide Time: 16:25)



Indian Institute of Technology, Kharagpur

## Modeling States

We can also use first order literals

$At(Robot1, Kitchen) \wedge At(Robot2, Garden)$

Literals used must be ground and function-free.

$At(Father(Fred), Home)$  X

$\exists x At(x, Home)$  X

Closed World Assumption:

Any condition not mentioned in a state are assumed false.

We also do not allow quantifications that is we do not allow  $at X home$  where  $X$  is a variable which is either existentially quantified or for all quantified. So these sort of first order literals are not allowed but we can have ground first order literals as in  $at robot 1 kitchen$ . In a simple formalism we allow states to be represented as a conjunction of propositions or a conjunction of first order literals or a mixture of them. Now there could be a large number of literals or propositions that are part of the planning problem. We may specify the values of some of the literals.


For example, we may specify that  $lighted$  is true,  $hot$  is true,  $made\ of\ brass$  is true. But we may not have specified the values of the twenty other literals that occur in this planning problem or hundred other literals that occur in this planning problem. In a closed world assumption we will assume that any condition we have not mentioned are assumed false. So, in a closed world assumption we will say that whichever literals we have specifically stated to be true in a particular state will be true but the other literals will all be false.

How is a goal represented?

A goal is represented as a partially specified state. For example,  $rich\ and\ famous\ and\ stays\ in\ Mumbai$ . This is a description of a goal. We say that a state  $s$  satisfies a goal  $g$  if the state  $s$  contains all the propositions of goal  $g$ . Suppose the description of the state  $s$  is  $lighted$  and  $hot$  and  $made\ of\ brass$  so this state satisfies the goal  $lighted\ and\ hot$  because the propositions  $lighted$  and the proposition  $hot$  both occur in the goal state.



(Refer Slide Time: 17:48)



Indian Institute of Technology, Kharagpur

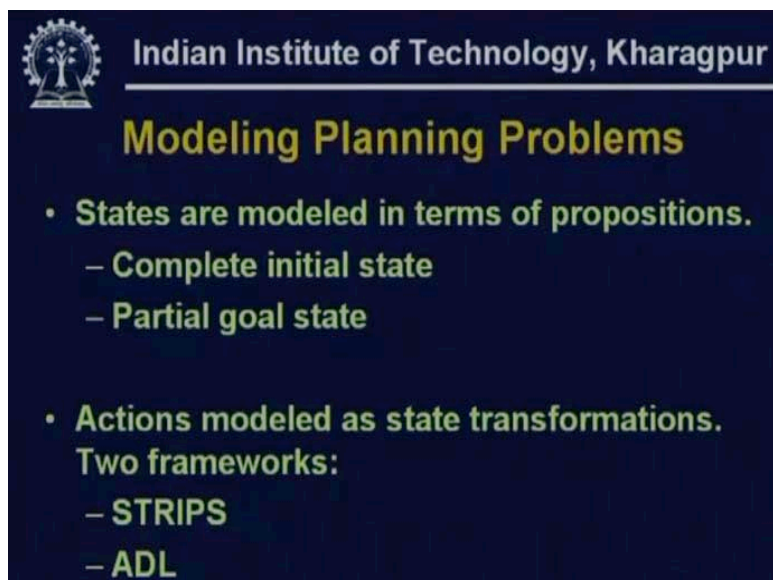
## Representation of goals

- A goal is a partially specified state  
Ex:  $\text{Rich} \wedge \text{Famous} \wedge \text{Stays (Mumbai)}$
- A state  $s$  satisfies goal  $g$   
if  $s$  contains all the propositions of  $g$

$\text{lighted} \wedge \text{hot} \wedge \text{brass}$   
satisfies the goal  
 $\text{lighted} \wedge \text{hot}$

If we take another example if we say that rich and famous and miserable and stays Mumbai and acts in movie M1, so that state will satisfy the goal; rich and famous and stays in Mumbai. For a goal to satisfy a state all the propositions in the goal must be in the description of the state. When we model planning problems states are modeled in terms of propositions as we said complete initial state and partial goal state. Now we will look at how actions are modeled.

(Refer Slide Time: 19:02)



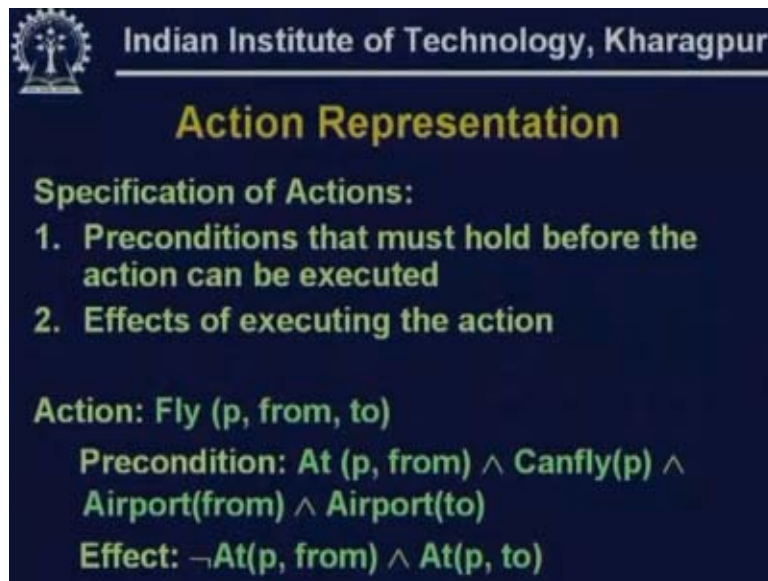
Indian Institute of Technology, Kharagpur

## Modeling Planning Problems

- States are modeled in terms of propositions.
  - Complete initial state
  - Partial goal state
- Actions modeled as state transformations.  
Two frameworks:
  - STRIPS
  - ADL

Actions are typically modeled as state transformations. To specify an action we specify what transformation to the state space will be brought about by the execution of an action. So, to represent actions different formalisms have been used. Today we will discuss about STRIPS which is a classical formalism for describing actions. We will later describe ADL which is a more modeled and less descriptive language which has better expressive power than the STRIPS representation formalism.

(Refer Slide Time: 20:03)



Indian Institute of Technology, Kharagpur

## Action Representation

**Specification of Actions:**

1. **Preconditions that must hold before the action can be executed**
2. **Effects of executing the action**

**Action: Fly (p, from, to)**

**Precondition:  $At(p, from) \wedge Canfly(p) \wedge Airport(from) \wedge Airport(to)$**

**Effect:  $\neg At(p, from) \wedge At(p, to)$**

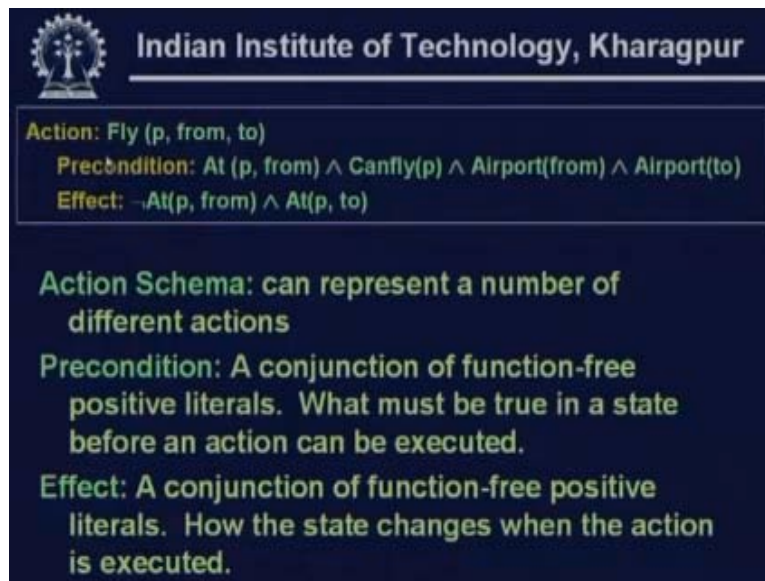
In STRIPS actions are specified by specifying two things. One is the precondition of the action the other is the effect of the action. The precondition of an action means the condition that must hold so that the action can be executed. Suppose I want to pick up an object the precondition is that my hand must be empty and the object which I wish to pick up must be portable and must be available in front of me. Therefore preconditions denote those conditions that must be satisfied so that the action can be executed.

Secondly, we need to describe the effects of an action. What will happen, how will the state change as a result of taking the action?

Suppose the agent is in a state where certain propositions are true and when the agent executes an action the state may change. The change can be described by using two different things. Some of the propositions which were true earlier would not be true after the action is executed and some new propositions which may have been false earlier would be true when the action is executed. So let us take an example of an action fly which takes three arguments fly p from to. The precondition is that at p from can fly p airport from and airport to. If these four conditions are satisfied in the previous state then only this action can be executed. And as a result of executing this action at p from will no longer be true but at p to will be true in the next state. So the effect of an action is described by a conjunction of positive and negative literals.

Negative literals correspond to those propositions which have to be deleted from the previous state to get the current state. The positive literals denote those propositions that will have to be added to the new state. Now, the other assumption we will make is that the rest of the propositions we have not specified in the effect list those propositions their status will be unchanged. If the other propositions were true in the earlier state they will continue to be true in the current state and if they were false in the earlier state they will continue to be false in the next state.

(Refer Slide Time: 23:02)



The slide features the IIT Kharagpur logo and name at the top. Below it, a box contains the following text:

**Action:** Fly (p, from, to)  
**Precondition:**  $At(p, from) \wedge Canfly(p) \wedge Airport(from) \wedge Airport(to)$   
**Effect:**  $\neg At(p, from) \wedge At(p, to)$

Below the box, the text explains the components of an action schema:

- Action Schema:** can represent a number of different actions
- Precondition:** A conjunction of function-free positive literals. What must be true in a state before an action can be executed.
- Effect:** A conjunction of function-free positive literals. How the state changes when the action is executed.

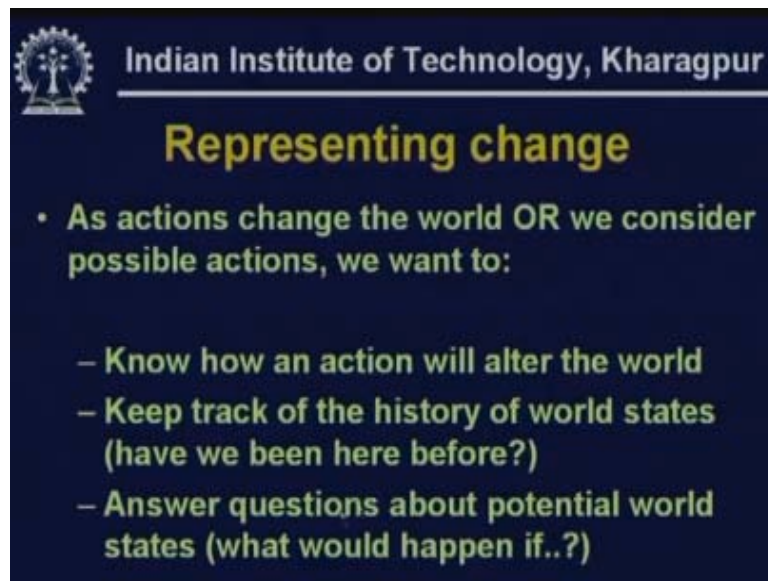
Now look at the action fly p from to. Now p from to are not ground literals but these are variables. So fly p from to represents an action schema. An action schema can represent a number of different actions. For example; it can represent the action fly plane one Kolkata Delhi, it can represent the action fly plane 1 Kolkata Chennai, it can represent the action fly plane one Kolkata Delhi, fly plane 1 Delhi Kolkata fly plane one Delhi Chennai, fly plane 2 Kolkata Delhi and so on. Therefore these actions schemata can represent a class of actions and number of different actions.

The precondition of an action is a conjunction of function free positive literals. They denote what must be true in a state so that the action can be executed in that state. This action fly p from to can only be executed if can fly p is true, p is at from, from is in airport, to is in airport. Therefore these are the conditions that need to be true so that the action can be executed. All actions cannot be executed at this state. For example, if p is at from I cannot execute the action fly p to from in the current state. As a result of the action the effect of the action is specified by the effect clause. The effect says not at p from and at p to. So the effect is a conjunction of function free literals not positive so this is a mistake.

So, effect is a conjunction of function free literals which may contain a set of positive literals and a set of negative literals. And they together represent how this state changes when the action is executed. If I have a negative literal it means that proposition is to be removed from the previous state to get the current state. If we have a positive literal it means that proposition has to be added to the new state. Now actions represent change in the world and action changes the world. We want to know how an action will alter the world. We also want to keep track of the history of the world states and answer questions about potential world states and in order to do that we need to reason about actions.

**Now you have studied first order predicate calculus earlier in this course.** Today we will briefly talk about situation calculus which is formalism in first order predicate calculus which allows us to reason about time and reason about situations.

(Refer Slide Time: 25:55)

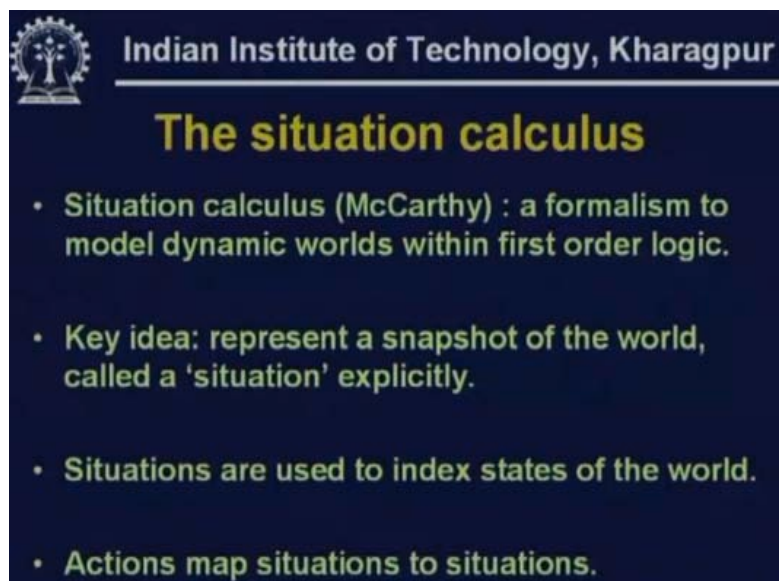


Indian Institute of Technology, Kharagpur

## Representing change

- As actions change the world OR we consider possible actions, we want to:
  - Know how an action will alter the world
  - Keep track of the history of world states (have we been here before?)
  - Answer questions about potential world states (what would happen if..?)

(Refer Slide Time: 26:33)



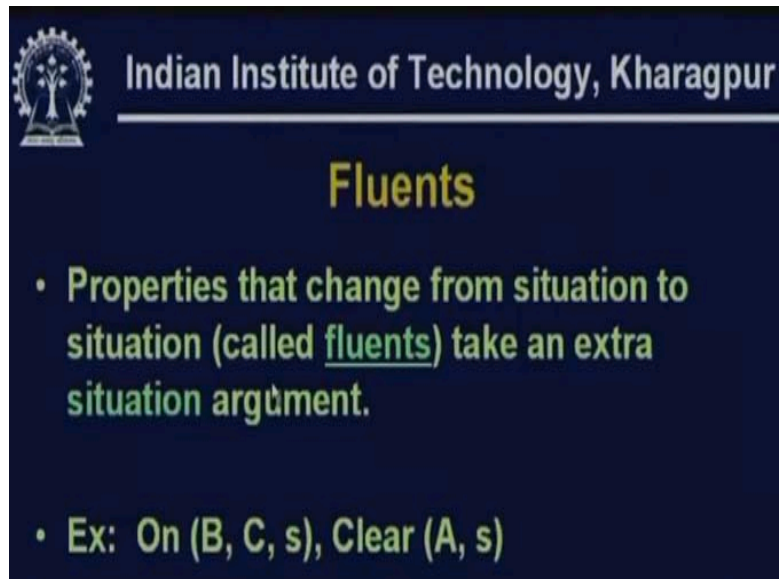
Indian Institute of Technology, Kharagpur

## The situation calculus

- Situation calculus (McCarthy) : a formalism to model dynamic worlds within first order logic.
- Key idea: represent a snapshot of the world, called a 'situation' explicitly.
- Situations are used to index states of the world.
- Actions map situations to situations.

Situation calculus was first discussed by McCarthy. It is a formalism to model dynamic worlds within first order logic. So situation calculus is a formalism to model dynamic worlds. The key idea in situation calculus is that we represent the snapshot of the world which we call a situation explicitly. And these situations index the different states of the world. The states of the world change with time. Suppose the world is in a situation  $s_0$  which has a particular state and when the agent takes some action as a result the state changes and  $s_1$  becomes the new situation and as a result of taking an action the situation changes. That is, actions map situations to situations.

(Refer Slide Time: 27:48)



The slide features the IIT Kharagpur logo on the left and the text 'Indian Institute of Technology, Kharagpur' at the top. Below this, the word 'Fluents' is written in a large, bold, yellow font. Two bullet points in green text follow: the first explains that fluents are properties that change from situation to situation and require an extra situation argument; the second provides an example: 'On (B, C, s), Clear (A, s)'.

Fluents are properties that change from situation to situation. There are certain propositions which are independent of the situation but there are certain Fluents which are those that change from situation to situation. So in order to specify those propositions we need an extra argument to denote the situation. For example; let us take a proposition on; suppose we have a blocks world, a blocks world is a world where we are dealing with several blocks. Our objective is that suppose we have a particular configuration of the block we wish to move these blocks so that we get to some other configuration of the blocks. This is the block world.

In block world the typical proposition we use is whether a block is on top of another block. For example, we like to say on A B. Now, on A B this proposition holds in this state or in this situation it does not hold in this situation. So in this case we will specify with an extra argument that on A B, suppose we call this situation s1 and we call this situation s2 we can say on A B in situation s1, on B A in situation s2 and not on A B in situation s2. Thus, on is a proposition whether on A B is true depends on the situation so we use a new situation argument in this proposition.



(Refer Slide Time: 29:18)

Indian Institute of Technology, Kharagpur

### Fluents

- Properties that change from situation to situation (called fluents) take an extra situation argument.
- Ex:  $\text{On}(B, C, s)$ ,  $\text{Clear}(A, s)$

Handwritten notes:  $\text{On}(A, B, s_1)$ ,  $\text{On}(B, A, s_2)$ ,  $\neg \text{On}(A, B, s_2)$

Then let us take the proposition clear. Clear is also proposition and whether A is clear or not depends on this situation like in this case A is cleared, clear A in  $s_1$  but A is not clear in  $s_2$ , C is clear in  $s_2$ . So again to continue with the blocks world example here I have a situation which we call  $s_0$  and we see that in this situation B is cleared. So clear B,  $s_0$  on B, C,  $s_0$  clear A,  $s_0$  and this is a robot hand and we have a proposition corresponding to hand empty so hand empty at  $s_0$ . So this is the description of this state in the blocks world.

Now let us look at possible actions in the blocks world. Let us take a particular action pick up. So the action, pick up x is the action schema, pick up B is a particular instance of an action schema.

(Refer Slide Time: 30:24)

Indian Institute of Technology, Kharagpur

### Blocks world example

Robot hand

- Clear (B,  $s_0$ )
- On (B, C,  $s_0$ )
- Clear (A,  $s_0$ )
- Handempty ( $s_0$ )

(Refer Slide Time: 30:44)

Indian Institute of Technology, Kharagpur

### Blocks world example: actions

Robot hand

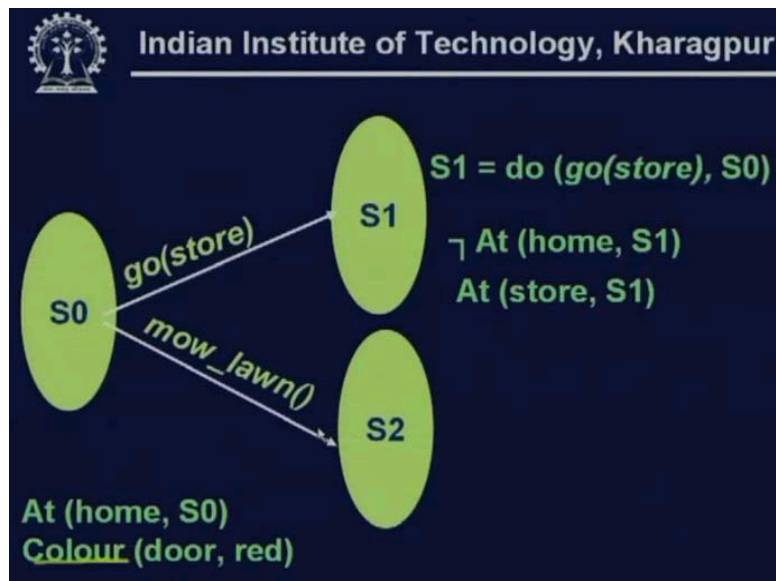
pickup (B)  
Stack (B, A)

do (pickup(B), s0)  
the new situation that is the result of applying pickup (B) in the situation s0

Now suppose the agent executes the action pick up B in the situation  $s_0$  so we denote it by  $do\ pickup\ B\ s_0$  so this action is executed at situation  $s_0$ . The new situation  $do\ pickup\ B, s_0$  represents a new situation which arises out of executing the action pick up B on situation  $s_0$ . So  $do\ pickup\ B\ s_0$  results into a new situation  $s_1$ . So in the situation  $s_1$  on B, C will not be true so not on B, C in  $s_1$ , clear C,  $s_1$  so clear B will not be true but clear C will hold true in situation  $s_1$ . So  $s_1$  is a new situation which is a result of executing the action pick up B in situation  $s_0$  and the situation  $s_1$  can be described by clear A clear C not hand empty and not clear B and not on B, C. This is another example suppose the situation  $s_0$  corresponds to at home  $s_0$  color door red.

Now notice that this proposition color its value does not depend on the situation depend on the time. So, for the proposition color we have not added the situation argument. At is a proposition whose value changes depending on the situation. So here we have added a situation argument but for the proposition color we have not added any situation argument. So  $s_0$  is described by at home  $s_0$  color door red. Now  $s_1$  is a situation which the agent gets if the action go store is executed so  $s_1$  is a result of executing the action do go store the action go store in the situation  $s_0$ .

(Refer Slide Time: 32:29)



And the situation  $s_1$  is described by not at home  $s_1$  at store  $s_1$ . So this is the effect of the action  $go\ store$ , not at home and at store. This proposition color door red is not affected by this action  $go\ store$  so if color door red holds in situation  $s_0$  it will continue to hold in the situation  $s_1$ .

Now in the initial state there could be, the entire problem description could have 100s of different propositions and some of which hold in the initial state and some of which do not hold in the initial state. Now after every action we get to a state and if we have to describe whether each of these propositions are true or false in that state then that would be a very laborious process. We also notice that typically an action will only have a local effect, it only changes the value of certain propositions. So the vast amount of background information remains unchanged. This problem in AI is referred to as the frame problem.

For example, suppose I am at home and there are different situations and different propositions that hold and the current time. For example, my son is at home, the store sells bread, my age is 38, and then so may other things, some of them hold in this state and some of them do not hold in this state. Now suppose I execute the action  $go\ store$ , now what will change is that, at home will no longer hold and at store will hold. However this vast amount of other propositions will continue to have the same value as they had earlier. And in this the background information does not change. This is known as the frame problem in AI. And we want an efficient representation of this frame problem. And if we confine ourselves to the framework of logical reasoning then we have to capture the frame problem using a set of rules and these rules can be captured by a set of actions called the successor state axioms. So, successor state axioms **encapsulate** the fact that normally things stay true from one state to the next unless an action changes them.

For example, suppose you consider the proposition at  $x$ . Now at  $x$   $x$  will hold at the next situation which has a reason out of doing the action  $A$  in situation  $s$ . So  $do\ A, S$  is the new situation which the agent finds itself in after carrying out the action  $A$  in

situation S. At x holds in this situation if and only if either A equal to go x if current action is go x then after this action at x will hold or at x will hold if the action is not go Y and the agent is already at x in the previous situation. If the agent is already at x and it does not execute a go action then the agent will continue to be at x. Or if the agent does execute the go x action then at the next situation it will be at x otherwise the agent will not be at x. This means that for every action or rather for every proposition that depends on the situation for every fluent we have to have one axiom like this axiom which recalls this successor state axiom. So in this case the fluent at x only changes the action go. There could be a fluent which changes as a result of more than one action. Suppose it changes with three different actions the successor state axiom for that fluent would involve all these three different actions.

Once we have wrote the problem in this form means we have written the initial states, the action representations and the successor state axioms then we can use theorem proving to find a solution to the planning problem. We have already seen how theorem proving works. Theorem proving can be used to find a solution to the planning problem. However, theorem proving may not always be a very efficient way of solving a planning problem so what we will do in this class is to explore different direct methods of solving the planning problem.

So we will first look at STRIPS. Representation for actions in STRIPS which we discussed about earlier, every action is represented by the precondition of the actions which is a list of propositions that have to be true for the action to be executed. The delete list and the add list so the effects of an action in STRIPS are divided into two different lists. The add list is the list of propositions that become true after the action is executed and the delete list contains the list of propositions that becomes false after the action is executed. In the STRIPS formalism the actions specify all the state variables whose values are changed by the action. But STRIPS places certain restrictions on the preconditions list as well as the add and delete list. Specifically STRIPS does not allow disjunctions in the effect list that is the add or delete list. And STRIPS only allow propositional preconditions at affects.

We will later see that there are some other languages like **ideal** which relax this criteria a little bit. So let us look at an example problem; suppose the initial state is described by (...40:43) at home, not have banana and not have money. And the goal is have banana and have money and at home. And the actions available are by x. the preconditions of by x are at store add list is add x. the other action is go X, Y, precondition is at x, delete list is at X, add list is at Y. Initially suppose we are at a situation S when we go from the home to the store we are at a new situation S prime. And in S as I said my son is at home the store sells bread and my age is 38 in the new situation S prime these propositions continue to hold. Now we want to know how we can efficiently represent everything that has not changed.

STRIPS provide a good solution for simple actions. STRIPS just says that apart from those propositions we have listed out in the add list and delete list the rest of the propositions are maintained as it is in the next situation as in the previous situation.

Now the ramification is that, suppose as a result of executing the action going to the store, suppose I am now in gold bazaar the number of people in the store went by 1 and the contents of my pockets are now in the store. So this is the effect of my action.

Now do we have to encode all these when we describe the effects of action?

In STRIPS you do not have to specify all the effects of the action which can be inferred by some other technique. For example, 42:58 the fact that the number of people in the store go up by 1 if I go to the store that can be handled by some rules. Primitive facts like at home persist between states unless we specifically change this by executing a go action.

Inferred facts are not carried over and must be re-inferred. Suppose the store currently has 5 people and if I go to the store the number of people becomes 6. So these things need not be specifically mentioned in the 'add and delete' list but they can be inferred. And this use of rules avoids making of mistakes. Now let us see how planning can be looked upon as a search problem. We are given a representation of the initial state, a set of STRIPS operators and a goal condition as we wish to achieve and a planning problem that we have to determine a sequence of actions which we apply to the initial state yields a state which satisfies the goal. Now we can look upon this as a search problem where the initial state is known to us, the actions or the possible operators that map a state to other states so we can have a state space graph and the goal is satisfied by any state that satisfies the goal.

Suppose my initial state is at home, not have banana, not have money and the actions that I can execute is let us say go store, go office, go movie, go bank, cook dinner and so on. And as a result of the go store action we will have the initial state which will get transformed to a different state and from this state again different actions can be applied. So we see that we have a state space formalism. And we can search in this tree until we get a tree where the goal conditions are satisfied. So the planning problem is actually a search problem. This is another example of the planning problem.

We have this initial state in this blocks world where B is on top of C and A is on the table. If we execute the action move A, B then we get to this particular case A is on top of B and B is on top of C if we execute the action move B, A then we come to this state. If we execute the action move B table we come to this state and if from this state we execute the action move C, A then we get to this state. Suppose our goal is represented by on C, A now we notice that in this state on C, A is true. So this is the state which satisfies the goal condition. Therefore this is the goal of the planning problem. The planning problem can be cast as state space search problem and we can detect whether a goal situation has been achieved. But the main problem in casting the planning problem such as the state space search problem is that the state space can be very large because the number of actions we can execute in a state may be very large and the state may really explode.

The search tree is generally quite large. Also the description of the planning problem suggests some structure of the problem because each action typically only affects all set of propositions. And actions depend on each other. Planning algorithms have to be designed to take advantage of this special nature of the representation. Normally we can start from the initial state and we can do a search in the forward direction which is known as forward state space search.

Forward state space search means you search from what is known in the initial state and apply operators in the order that they are applied. And alternative to forward state



space search is the backward state space search. In backward state space search you do the search in the other way. That is, you start from the goal state and you try to find out the actions that would have led to such a state so you consider the actions in the reverse and the states in the reverse direction. So you search from the description of the goal and identify the actions that help you to reach the goal.

In forward search the branching factor can be extremely high. All applicable actions have to be considered from each state and typically they may include many actions which are irrelevant. Suppose my objective is to bake a cake, now going to a movie is an action which I can perform at the initial state but whose effects are irrelevant to the goal of baking a cake. Therefore if we have a forward search typically search could be very inefficient unless we use a very accurate heuristic. If we do backward search starting from the goal this allows us to consider relevant actions.

However, in backward search also we can have some actions which are irrelevant in the sense that we could not have arrived at the (....49:47) conditions starting from the initial state we have. Actually backward search is also called regression planars. Planars that do backward search is called regression planars. In contrast to this the forward search planars are called progression planars. So regression planars typically have smaller branching factor and for most problems they have a more focused search space than planars that search in the forward direction. However both backward and forward search may lead to the consideration of a lot of irrelevant action. So if you really want to make the planning problem efficient you need to deploy very good heuristics.

The questions for today's lecture is,

- 1) Explain how planning systems differ from classical search techniques.
- 2) Formulate the blocks world planning problem.

Take a particular description of the blocks world problem, take a description of the goal state and represent the initial state, represent the goal state, list out the possible actions in the blocks world. For each action you give the precondition list and effect list and then you can take this example problem and draw the search space corresponding to this problem.