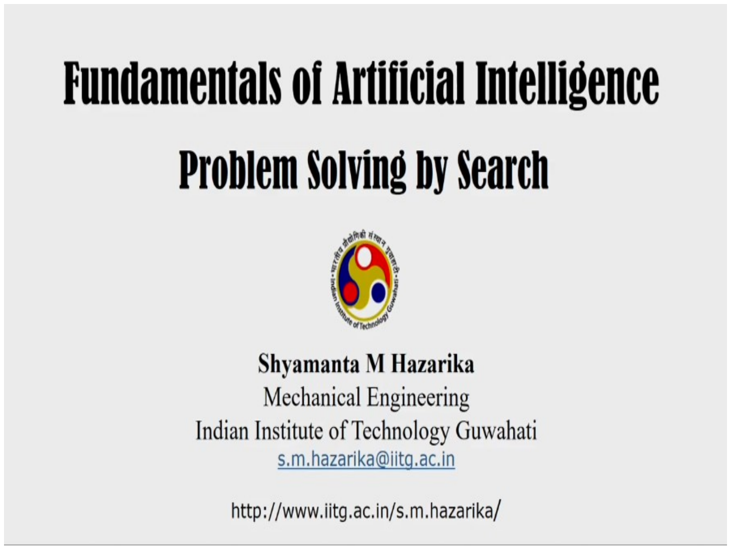
Fundamentals of Artificial Intelligence

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

Module - 1
Lecture - 2
Problem Solving by Search

Welcome back to the course on Fundamentals of Artificial Intelligence. We will do the second module from today. The module is called Problem solving by Search.

(Refer Slide Time: 00:49)



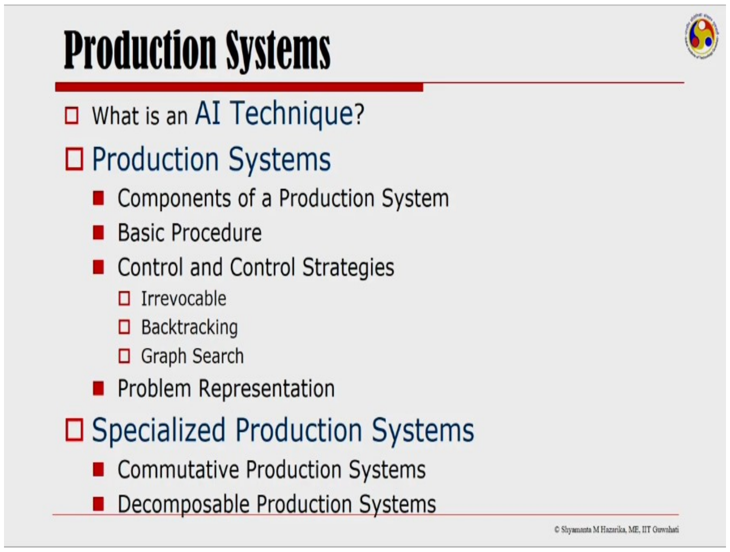
Fundamentals of Artificial Intelligence
Problem Solving by Search

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

First we will look at what is that makes AI programming different from general computational paradigms. We will look at what is commonly called an AI technique and then introduce production systems - a system of programming which captures the essence of AI systems. We will focus on what are the different components of a production system and then look at some specialized production systems. This is what the overview of the course today looks like.

(Refer Slide Time: 01:35)


Production Systems

- What is an AI Technique?
- Production Systems
  - Components of a Production System
  - Basic Procedure
  - Control and Control Strategies
    - Irrevocable
    - Backtracking
    - Graph Search
  - Problem Representation
- Specialized Production Systems
  - Commutative Production Systems
  - Decomposable Production Systems

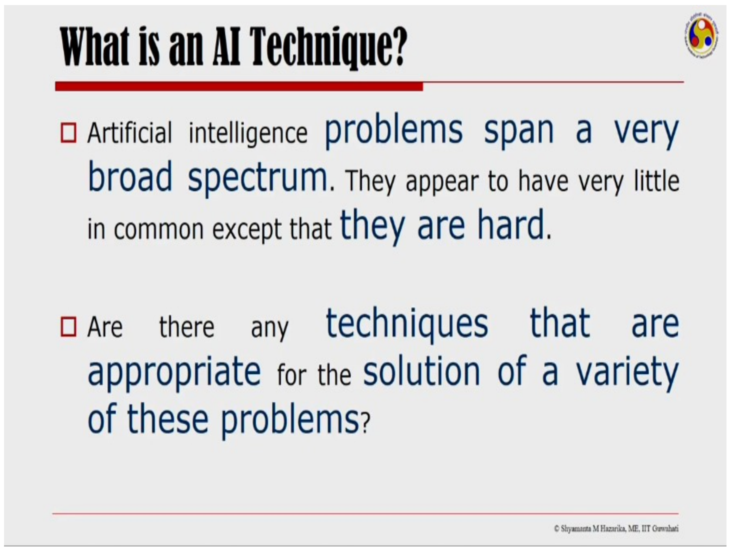
© Shyamanta M Hazarika, ME, IIT Guwahati

We would first talk on: What is an AI technique? How is AI different from the general traditional programming? What makes AI programming different? At the end of this course today, you should be able to identify what is production systems; the components of a production system. We would introduce a basic procedure and look at different control and control strategies.

Production systems is what separates facts, the way reasoning happens and the control system into three distinct units. We shall look at each of them very closely and then focus our attention on how control and control strategies are designed. We will look at what is called irrevocable control strategy, backtracking and graph search. A very interesting way of looking at things as a production system involves not only looking at these three units, but also able to do what is called problem reduction.

We will look at a very specialized production system called decomposable production system which allows us to reduce the problem. And we will also look at something called the commutative production system. But then first, let us focus our attention on what is an AI technique.

(Refer Slide Time: 03:22)



What is an AI Technique?

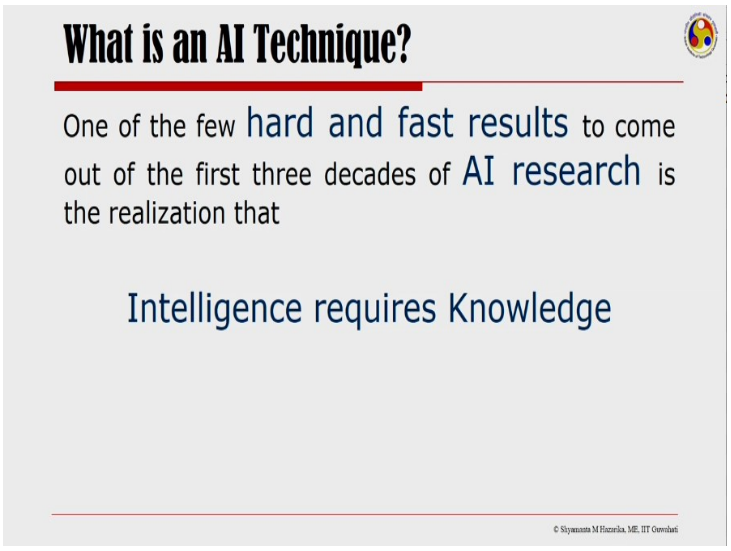
- Artificial intelligence problems span a very broad spectrum. They appear to have very little in common except that they are hard.
- Are there any techniques that are appropriate for the solution of a variety of these problems?

© Shyamanta M Hazarika, ME, IIT Guwahati

Artificial intelligence as you can remember from the first lecture of module 1, involves problems that span a very broad spectrum. Now, they are problems like natural language processing, navigation for robot in a clustered environment. It could be like autonomous driving or automated reasoning. They appear to have very little in common, except that they are very hard.

The focus, when we ask the question what is an AI technique is to look for an answer if there are any techniques that are appropriate for the solution of a variety of these problems? Is there something common which is there for all these problems that appear?

(Refer Slide Time: 04:29)



What is an AI Technique?

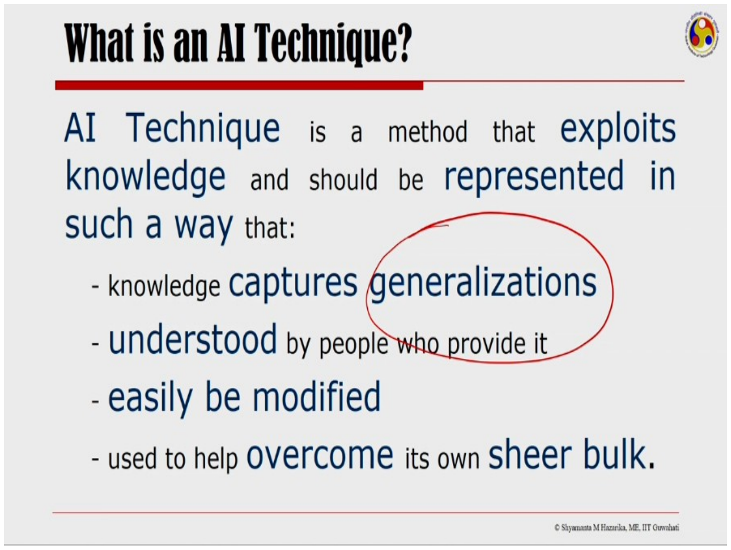
One of the few hard and fast results to come out of the first three decades of AI research is the realization that

Intelligence requires Knowledge

© Shyamanta M Hazarika, ME, IIT Guwahati

One of the few hard and fast results to come out of the first three decades of AI research is the fact that any such intelligent program that one needs to write requires knowledge. This is a very interesting idea that knowledge (if you have to have any program that can) give rise to intelligence.

(Refer Slide Time: 04:57)



What is an AI Technique?

AI Technique is a method that exploits knowledge and should be represented in such a way that:

- knowledge captures generalizations
- understood by people who provide it
- easily be modified
- used to help overcome its own sheer bulk.

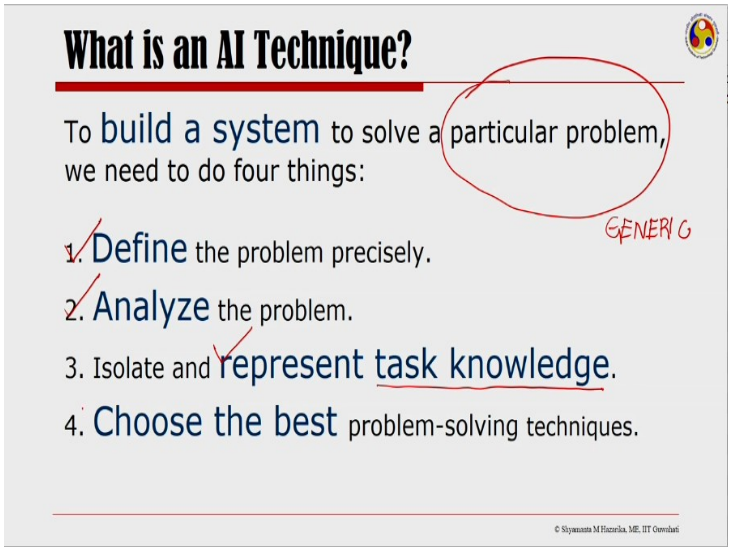
© Shyamanta M Hazarika, ME, IIT Guwahati

AI technique therefore is a method that exploits knowledge and should be represented in such way that this knowledge that we are talking of captures generalization. Here it is very important for us to note what we mean by generalization. Whenever we are talking of a technique that exploits AI, we are more interested in capturing the general information about the problem, rather than capturing very specific things of the problem.

The next important thing that an AI technique exploits is a knowledge that is understood by the people who provide it. Another important characteristic is that this knowledge that is provided needs to be easily modified. When I say knowledge needs to be easily modified, what I mean is, that the knowledge needs to be worked on by some system to create new knowledge. And this new knowledge that I create out of existing knowledge should be able to be created by very simple operations.

This, I think is what drives AI technique. And an AI technique is therefore these three things used to help overcome the sheer bulk that is required.

(Refer Slide Time: 06:33)



What is an AI Technique?

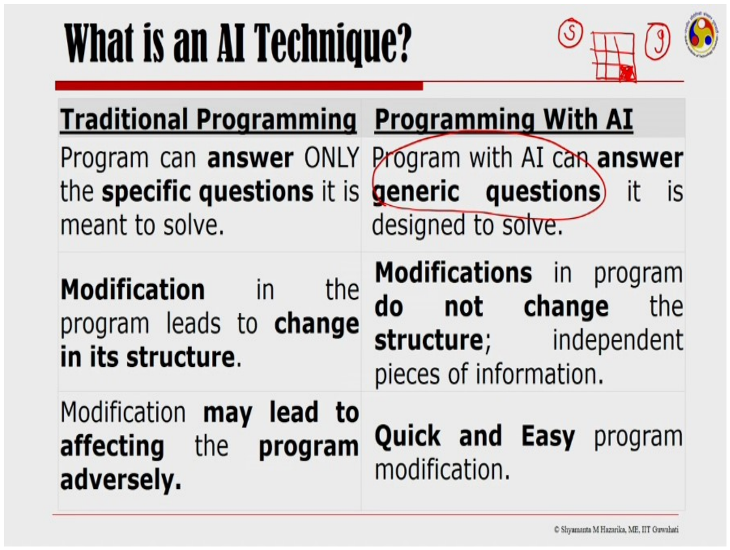
To build a system to solve a particular problem, we need to do four things:

1. Define the problem precisely.
2. Analyze the problem.
3. Isolate and represent task knowledge.
4. Choose the best problem-solving techniques.

What is an AI technique? In order to answer this question, we need to understand that the aim here is to build a system, definitely for a particular problem. But then, the solution that we expect is as generic as possible. So, we are looking for a generic solution to definitely a particular problem. And when we want this, we need to do four things. a. We need to define the problem precisely. Problem definition is as important as what knowledge we capture for that problem. The second thing that needs to be done is a great analysis of the problem. One needs to understand what this problem that we are trying to tackle is about. And number 3: we need to isolate and represent the task knowledge. Now what is task knowledge and how do we isolate and represent task knowledge is something that we will look at in the next few examples that we are going to talk about.

The fourth interesting thing is than given these three important things, that we define the problem precisely, we analyze the problem, we then represent the task knowledge; the onus is now to choose the best problem-solving technique.

(Refer Slide Time: 08:08)



What is an AI Technique?

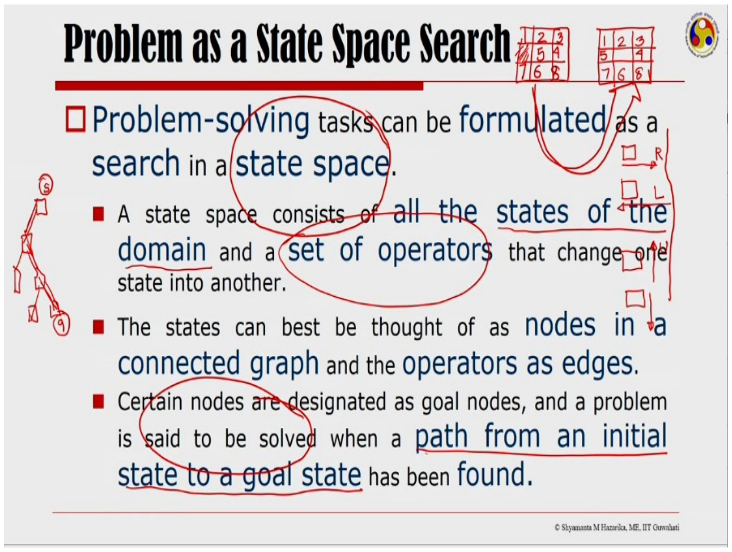
Traditional Programming	Programming With AI
Program can **answer** ONLY the **specific questions** it is meant to solve.	Program with AI can **answer generic questions** it is designed to solve.
Modification in the program leads to **change in its structure**.	**Modifications** in program **do not change** the **structure**; independent pieces of information.
Modification **may lead to affecting** the **program adversely.**	**Quick and Easy** program modification.

It is therefore important for us to realize what is the basic difference between traditional programming and programming with artificial intelligence or programming that tries to create AI systems. Traditional programming is when we program and look for answer for only a specific question it is meant to solve. Like, let us take an example, that I am interested in finding out the square root of a quadratic equation. My focus is only on getting the roots of the equation. Whereas, if I am doing programming with AI, the idea is to get answers to generic questions, rather than only specific questions that I am asking. Like here, one could take the example of playing a game of checkers or playing a game of chess or more interestingly what we will exploit is a game of what we call 8-puzzle, where I have 9 tiles positions. And one of the tile positioning is free, so that I can move it around. And given a start position of that configuration of tiles, I want to look for a goal position of the configuration of tiles. And when I am trying to look at a solution of this, I am not interested in a specific solution of given a particular start state to a particular goal state. I am interested in a way of arriving at from any given start state to any given goal state.

The second important difference is that any traditional programming paradigm that we explain modification in the program itself will lead to change in its structure. Whereas, if it is an AI programming paradigm which we will look at today, modification in program do not change the structure. This is because (as we will look at the paradigm of programming called production systems, which captures the essence of AI systems) it is made up of independent pieces of information.

The rules, facts, the reasoning and the control are completely distinct elements. Therefore, modifications in one of them do not change the other or do not change the structure of the program. The third interesting difference therefore flows from the second. Modification in traditional programming may lead to adversely affecting the program. Whereas, here when I am talking of a production system or what I loosely refer to as programming with an AI, modifications would be quick and easy.

(Refer Slide Time: 11:31)



Problem as a State Space Search

- Problem-solving tasks can be formulated as a search in a state space.
  - A state space consists of all the states of the domain and a set of operators that change one state into another.
  - The states can best be thought of as nodes in a connected graph and the operators as edges.
  - Certain nodes are designated as goal nodes, and a problem is said to be solved when a path from an initial state to a goal state has been found.

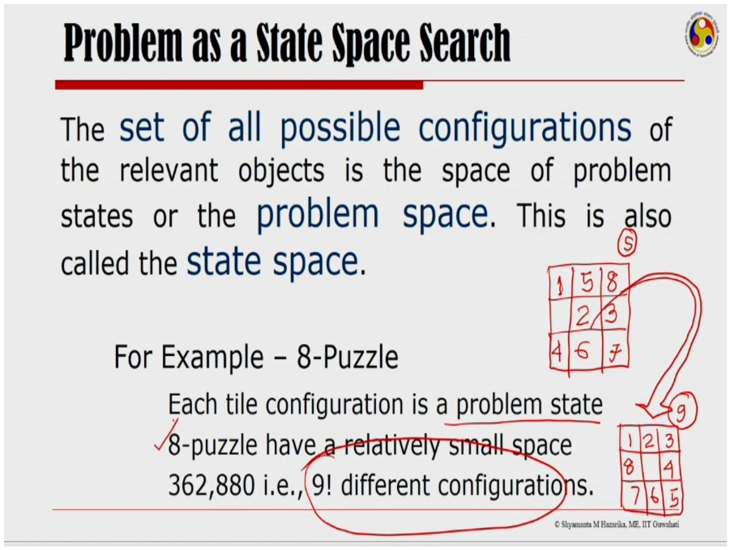
© Shyamanta M Hazarika, ME, IIT Guwahati

So, we are now in a position to state what we started our lecture with, that we want to do problem solving as search. But before that, we need to understand what do we mean by problems as a state space search? Problem solving task within such a module can be formulated as search in state space. A state space consists of all states of the domain and states of operators that change one state into another.

The states can be best thought of to be as nodes in a connected graph and operators as edges. And when we were looking at problem solving, problem solving can be formulated as a search in the state space. Let us look at this example more clearly to understand what we mean by this. So, let us say we have taken a problem to find out the solution of an 8-puzzle game. An 8-puzzle game has a 3 x 3 matrix, where numbers are written: 1, 2, 3, 4, 5, possibly 6, 7, 8, with one of the tile which is unoccupied, so that you can move that tile. So, this space can come here and here. So basically, when I am talking of the 8-puzzle itself, I can have such configurations. And such configurations can change from one to another by a set of operators, which happens because of movement of the empty tile. Like here, I could take the empty one to the right. So, 5 comes here. And I have 1, 2, 3, 5, 4, 7, 6, 8. So, from here to the state here, I can come by a simple operator which would be something like, take the empty to the right.

And when I am doing this, I am generating a number of newer states. That newer states that is getting generated is (all of them together) creating what is called the search state space. So, all of these states of the domain and all of these operators that I have. So, we could have four operators where the blank goes to the right; the blank could be pushed to the left; the blank space could be pushed up or the blank space could be pushed down.

All those, these operators generate what is called the set of operators. The states of the domain and the set of operators once defined, I can think of moving from one state to the other state. And basically, what we will have is therefore given a state some way to move to new states; to new states, to new states down there. And, when I am looking for a solution to this problem, okay?
Given a start state s and some goal state g, the solution to this problem is about finding a path from s to g. So, path from an initial state to a goal state; if it is found, the problem is said to be solved. This is essentially what it means to solve problems as state space search.
(Refer Slide Time: 15:37)



Problem as a State Space Search

The set of all possible configurations of the relevant objects is the space of problem states or the problem space. This is also called the state space.

For Example – 8-Puzzle

Each tile configuration is a problem state

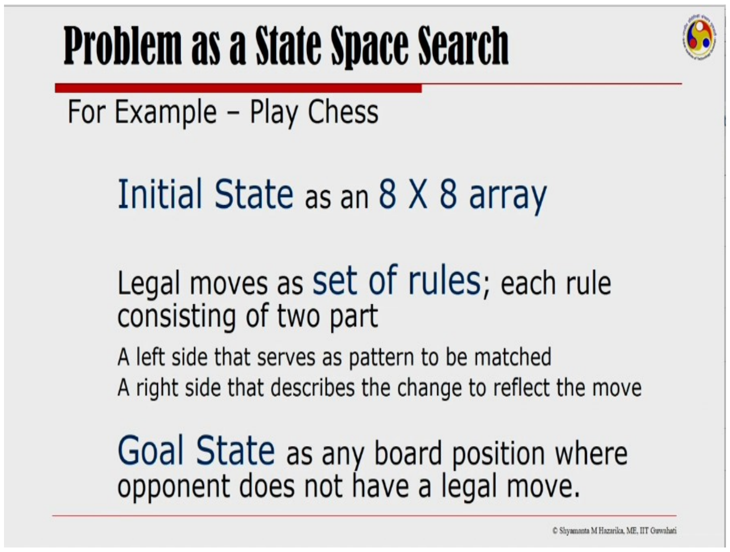
✓8-puzzle have ~~a relatively small~~ space 362,880 i.e., 9! different configurations.

S
1 5 8
2 3
4 6 7
g
1 2 3
8 4
7 6 5
© Shyamanta M Hazarika, ME, IIT Guwahati

The set of all possible configurations that I get of the relevant objects is the state space of the problem. This is also referred to as the problem space. I showed you an example of the 8-puzzle. So, each tile configuration that I have drawn in the previous slide, like this 3 x 3 matrix with numbers running from 1, 2, 3, 4, 5, 6, 7 and an 8, with one of them as empty, is a problem state.

8-puzzle actually has a very relatively small problem space, which is some 9 factorial different configurations. So, when we are looking for a solution from a given start to some given goal position g, which could be any other configuration in the system; like, I could be thinking of arranging them as 1, 2, 3, 4, 5, 6, 7 and 8. I am thinking of getting a way to come from this to this. And the total configurations that are there is only 9 factorial different configurations. So, such a search would be possible.

(Refer Slide Time: 17:26)



Problem as a State Space Search

For Example – Play Chess

Initial State as an 8 X 8 array

Legal moves as set of rules; each rule consisting of two part

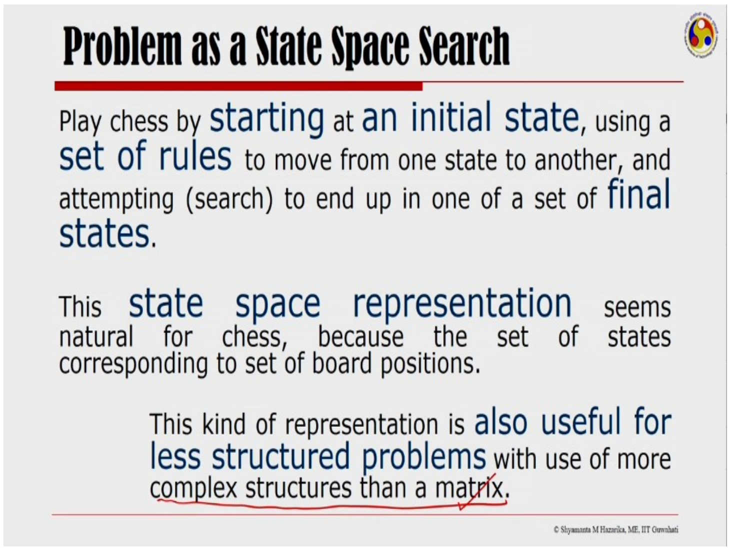
A left side that serves as pattern to be matched
A right side that describes the change to reflect the move

Goal State as any board position where opponent does not have a legal move.

© Shyamanta M Hazarika, ME, IIT Guwahati

We can think of another example which is about the game of chess. Initial state for a game of chess could be thought of as 8 cross 8 array. The legal moves as set of rules, each rule consisting of 2 parts. A left side that serves as pattern to be matched and a right side that describes the change to reflect the move. Goal state could be any board position where the opponent does not have any more legal move. And therefore, it is a win.

(Refer Slide Time: 18:02)


Problem as a State Space Search

Play chess by starting at an initial state, using a set of rules to move from one state to another, and attempting (search) to end up in one of a set of final states.

This state space representation seems natural for chess, because the set of states corresponding to set of board positions.

This kind of representation is also useful for less structured problems with use of more complex structures than a matrix.

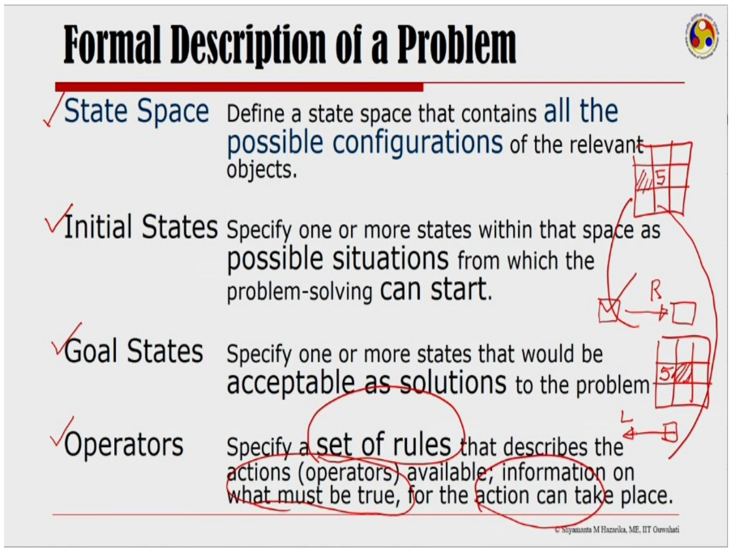
© Shyamanta M Hazarika, ME, IIT Guwahati

When we start playing chess by starting at an initial state using a set of rules to move from one state to another, we end up in one of the set of final states. This state space representation seems natural for games such as chess, 8-puzzle, because the set of states corresponding to set of board's position. This kind of representation may not be very obvious, but would be also useful for less structured problems with use of more complex structures than a matrix.

I would like to emphasize here that the game that I have spoken of, 8-puzzle; the matrix representation came obviously to the mind because we were talking of a 3 cross 3 arrangement of tiles. For chess, an 8 cross 8 matrix immediately comes to mind. But then, this kind of state space representation is also useful for other less structured problems. The only emphasis that I want to give here is now to point out that, under less structured problems, if we want to use such a state space representation, we would need more complex structures than a matrix.

And such a representation, such a mode of problem solving could be useful. Now, here is the formal description of a problem.

(Refer Slide Time: 19:42)



Formal Description of a Problem

- **State Space**: Define a state space that contains all the possible configurations of the relevant objects.
- **Initial States**: Specify one or more states within that space as possible situations from which the problem-solving can start.
- **Goal States**: Specify one or more states that would be acceptable as solutions to the problem.
- **Operators**: Specify a set of rules that describes the actions (operators) available; information on what must be true, for the action can take place.

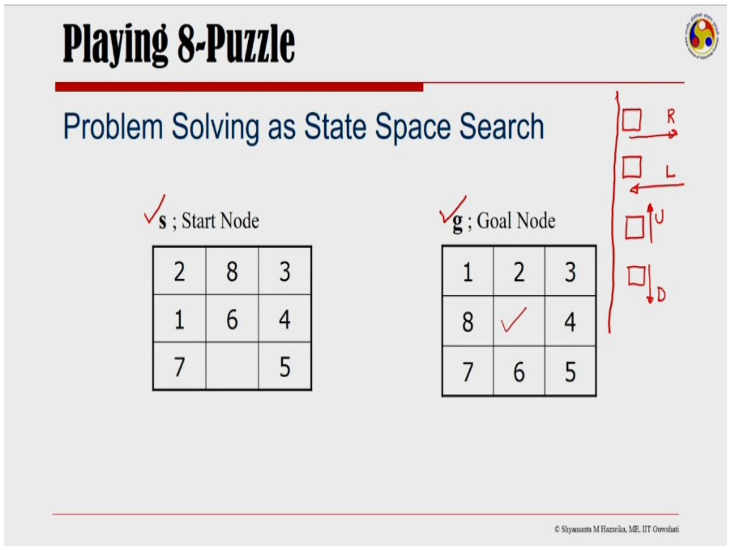
© Shyamanta M Hazarika, ME, IIT Guwahati

So, what we have is: Number 1. The state space. The state space defines all possible configurations of the relevant objects. The number 2 what we have is the initial state. The initial state specifies one or more states within that space as the possible situation from which problem solving can start. Thereafter, we have the goal state. The goal state specifies one or more states that would be acceptable as solutions to the problem.

And of course, this is about only the different configurations of states that we are talking of. Together with this, we would need a set of rules that describes how we are going to move from one state to the other state. So, these are called operators. So, here is the formal description of problem solving as state space search. We have to define the state space which is all the possible configurations. We then identify one or more states within the space as possible situations from which we can start. That is the initial state. One or more states from the problem space that is acceptable as solutions is referred to as the goal state. And then, a couple of rules that describe actions available. Now, I will just take 1 minute to again emphasize what we were talking of as operators.

Operators have two interesting portions. One on the left and one on the right. Information must be true. Everything on the left must match, so that I can take that particular operation, so that we have information on the right to be true. Like, remember our 8-puzzle game, where we had this 9 tiles. And we had a rule which said like, an empty can go to the right. This rule is applicable in the above state. Because, let us say we have 5 here. The empty moves to the right. Then, I have a configuration that leads to other things remaining same, 5 coming here and empty here. Whereas, the same operators from this group of operators, if I take another which say move to the left empty. This is not applicable in this state. Because, here I cannot move to the left. So, when I specify the set of rules that describes actions or operators, information what must be true for the action to take place needs to be looked at first. And then the action can take place and can generate new information as we have seen here.

(Refer Slide Time: 23:07)



Playing 8-Puzzle

Problem Solving as State Space Search

✓ **s** ; Start Node

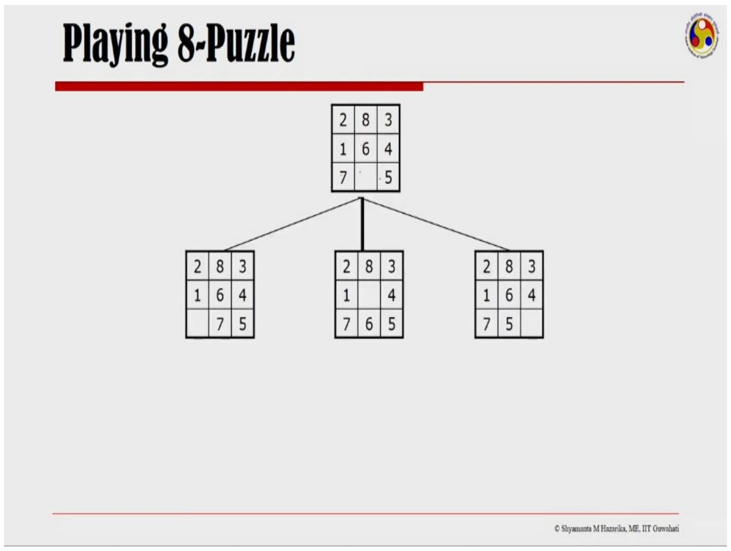
2	8	3
1	6	4
7		5

✓ **g** ; Goal Node

1	2	3
8	✓	4
7	6	5

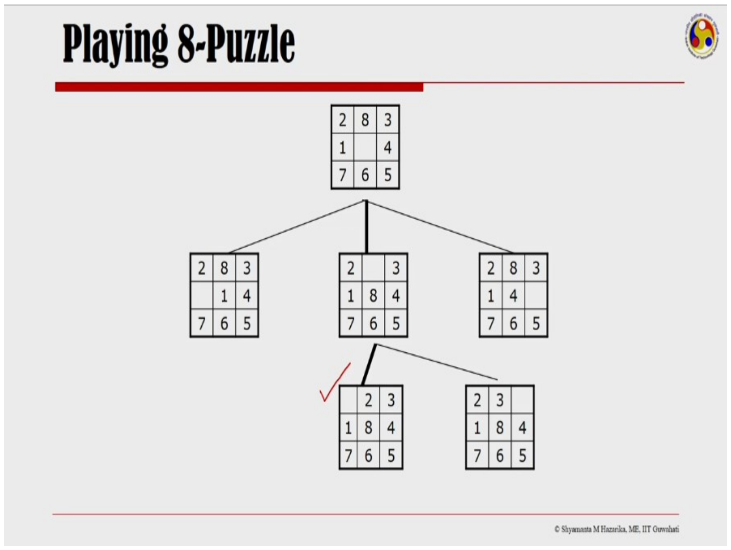
So, let us look at playing 8-puzzle with problem solving as state space. So, here is our start state where I have 9 tiles and my empty at the bottom. And this is my goal state where I have arranged everything as 1, 2, 3, 4, 5, 6, 7, 8, with empty at the middle. Now, as I go on generating; and remember the 4 operators that we were talking of. Let us quickly recall our operators. So, we could have empty that moves to the right. We could have an empty that moves to the left. We could have an empty or vacant tile position. And that vacant tile position that moves up and a vacant tile position that moves down. Given the start state s and given the goal state g and the four operators that we have here; let us try to understand problem solving as state space search.

(Refer Slide Time: 24:26)



So, at this point, all the 4 operators that I have highlighted in the previous slide, 3 positions are possible. The empty could go up; the empty could go left or the empty could go right. So, these are the 3 configurations that can be arrived at from the given set of; the given start position. I then look at these new configurations and look at my 4 operators that I had highlighted in the previous slide; and try to apply these operators here. So, I will take each of them 1 by 1.

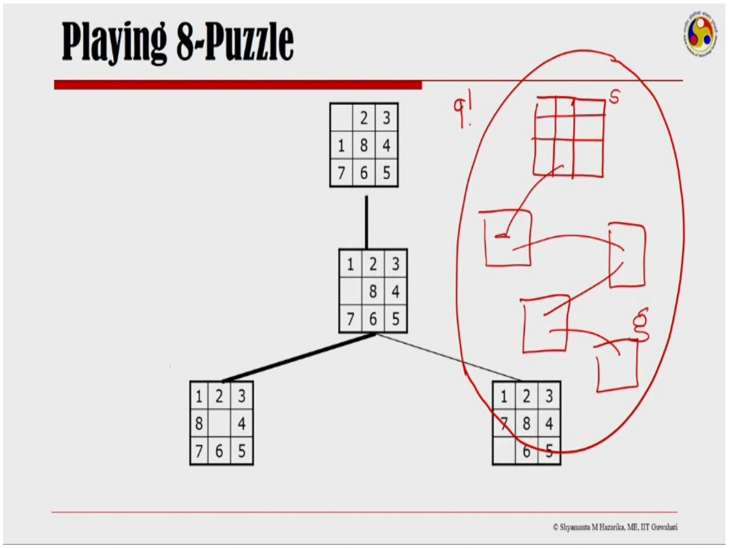
(Refer Slide Time: 25:06)



Now, we generate new states from here. So, one could come to the center and the other position that I could generate is that 8 could come down. And here is the third position where 4 could come to the center. To just keep my discussion of these 8-puzzle problem shorter, I have taken a route that is taking me towards the goal; but that does in no way stops us from exploring this paths as well.

Just to highlight that we can arrive at this goal very quickly, given this. So, I have just looked at this path. So, then, next I generate the 2 positions. 1, this empty can go to the left and the empty can go to the right. So, I have 2 positions. I now highlight; I; and see what happens when operator applies on this position that I have marked here.

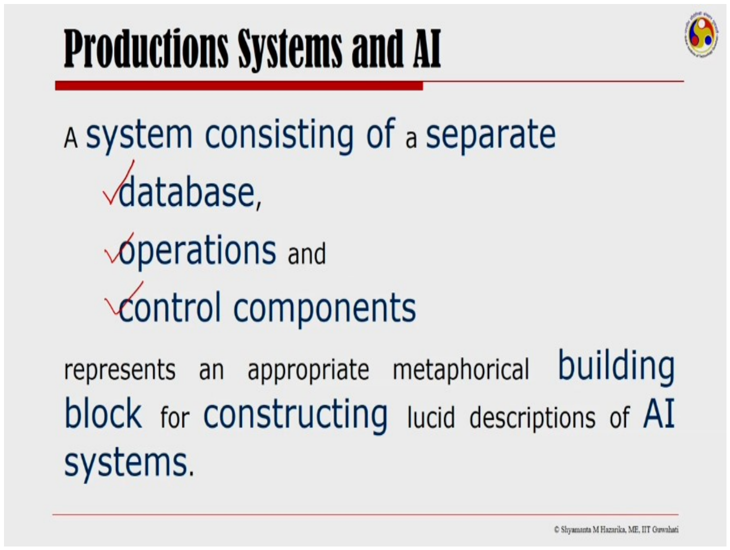
(Refer Slide Time: 26:08)



Playing 8-Puzzle

So, once I have that, I can have a location like 1 could only go up. If it goes to the left, I generate something which has already been generated. So, I do not do that. And, what I get after a couple of steps is my goal position. So, what I want to emphasize here is that, starting from a start position and which was part of a space of the problem state, which we have seen is a factorial 9, 9 factorial. We could go and look at different configurations on the way generating lot of them and arrive at the goal position g, from a start position s. So, this paradigm of working like this is problem solving as state space search.

(Refer Slide Time: 27:19)



Productions Systems and AI

A system consisting of a separate

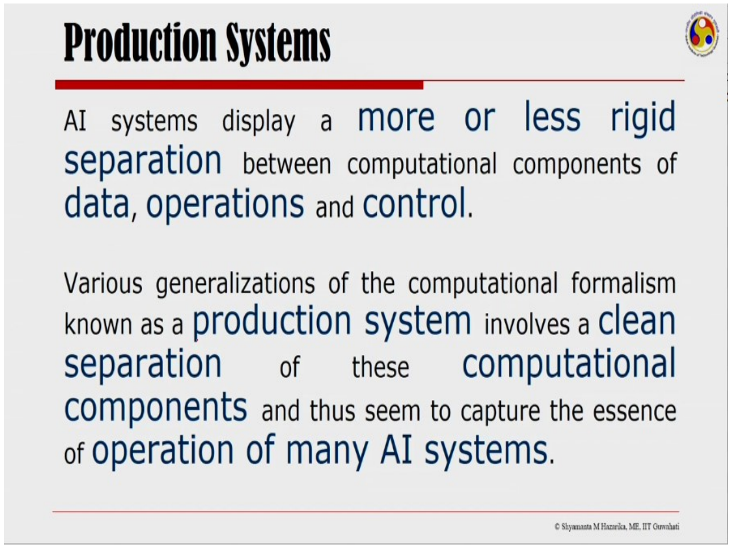
- ✓ database,
- ✓ operations and
- ✓ control components

represents an appropriate metaphorical building block for constructing lucid descriptions of AI systems.

© Shyamanta M Hazarika, ME, IIT Guwahati

So, now we are in a position to define production systems that works in such a way. So, production system consists of 3 important things: a database. Here I would emphasize that one should not confuse this with database from databased systems. What we mean here is a unit that consists of rules and facts, okay; operations and control strategies. So, these are the building blocks for constructing lucid descriptions of AI systems. So, a production system consists of these 3 elements: database, operations and control components.

(Refer Slide Time: 28:08)



Production Systems

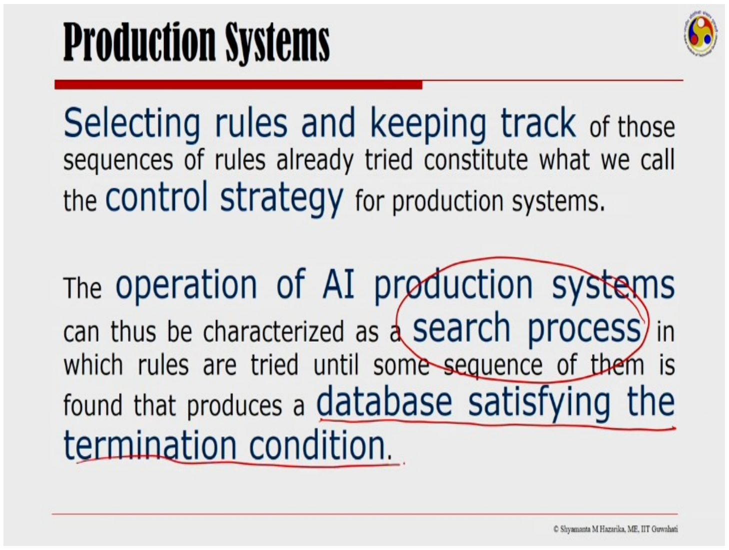
AI systems display a more or less rigid separation between computational components of data, operations and control.

Various generalizations of the computational formalism known as a production system involves a clean separation of these computational components and thus seem to capture the essence of operation of many AI systems.

© Shyamanta M Hazarika, ME, IIT Guwahati

AI systems that we will talk of or most of the AI systems display a more or less rigid separation between the computational components of data, operations and control. Various generalizations of the computational formalism that I have highlighted now, production systems involving clean separation of these components. And they seem to captured a essence of operation of many AI systems.

(Refer Slide Time: 28:39)



Production Systems

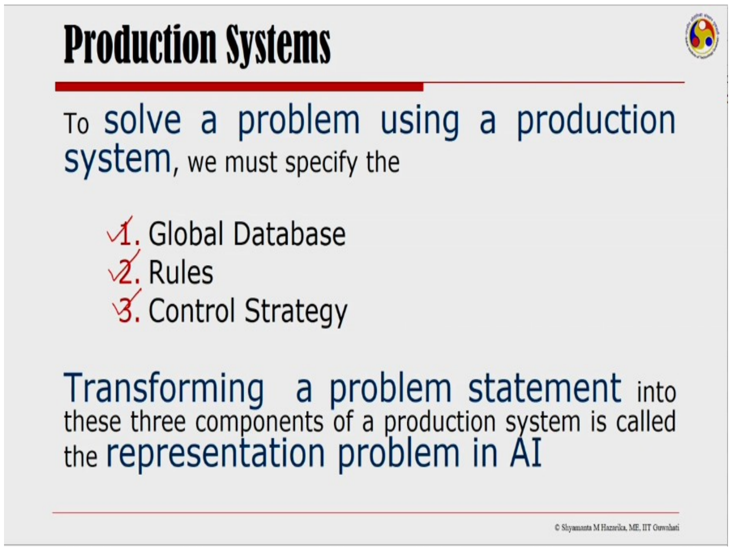
Selecting rules and keeping track of those sequences of rules already tried constitute what we call the control strategy for production systems.

The operation of AI production systems can thus be characterized as a search process in which rules are tried until some sequence of them is found that produces a database satisfying the termination condition.

© Shyamanta M Hazarika, ME, IIT Guwahati

Selecting rules and keeping track of these sequence of rules already tried constitute what we call the control strategy for production systems. The operation of AI production systems can thus be characterized as a search process in which rules are tried until some sequence of them is found, where I have finally arrived at the database, the facts that satisfy the termination condition.

(Refer Slide Time: 29:19)



Production Systems

To solve a problem using a production system, we must specify the

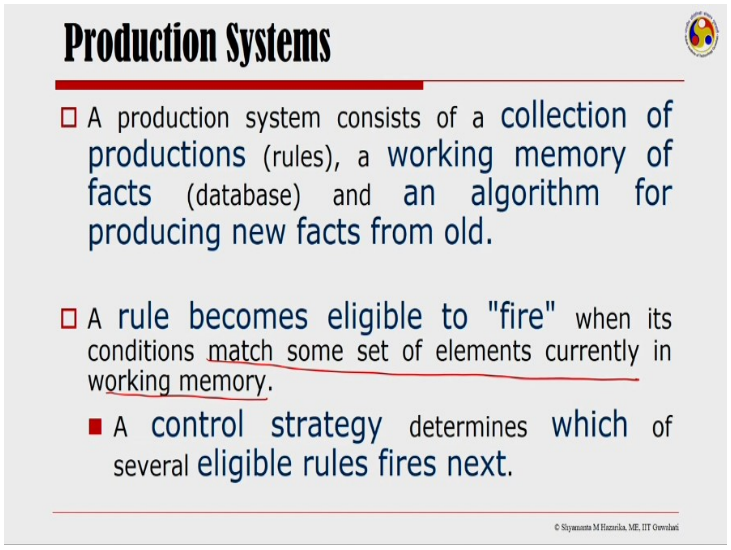
✓1. Global Database
✓2. Rules
✓3. Control Strategy

Transforming a problem statement into these three components of a production system is called the representation problem in AI

© Shyamanta M Hazarika, ME, IIT Guwahati

So, to solve a problem using a production system, we therefore must specify the global database, the rules, the control strategy. And these 3 things together form what is called the production system. Now getting to that stage or transforming a problem statement into these 3 components of a production system is the representation problem in AI. And once this is successfully done, then problem solving is about just searching in that problem space for the given goal state given the initial state.

(Refer Slide Time: 30:03)



Production Systems

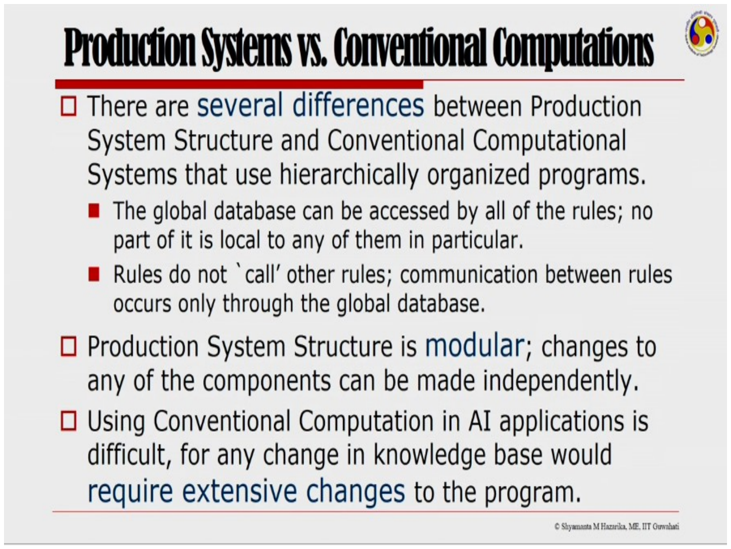
- A production system consists of a collection of productions (rules), a working memory of facts (database) and an algorithm for producing new facts from old.
- A rule becomes eligible to "fire" when its conditions match some set of elements currently in working memory.
  - A control strategy determines which of several eligible rules fires next.

© Shyamanta M Hazarika, ME, IIT Guwahati

A production system consists of productions, that is the rules and the working memory of facts that is referred to as the database. And the control strategy is about an algorithm for producing new facts from old ones. Now, as we have seen in the 8-puzzle example, a rule becomes eligible to fire when the conditions match some of the elements currently in the working memory.

And as its match its left side, the rule is fired and we have new facts, which is the facts that is generated on the right side of the rule. A control strategy determines which of the several eligible rules fires next.

(Refer Slide Time: 30:54)



Production Systems vs. Conventional Computations

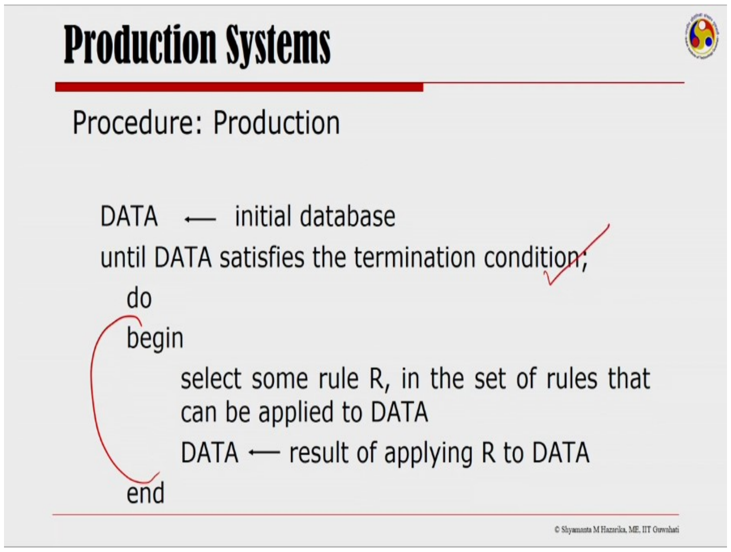
- There are several differences between Production System Structure and Conventional Computational Systems that use hierarchically organized programs.
  - The global database can be accessed by all of the rules; no part of it is local to any of them in particular.
  - Rules do not `call' other rules; communication between rules occurs only through the global database.
- Production System Structure is modular; changes to any of the components can be made independently.
- Using Conventional Computation in AI applications is difficult, for any change in knowledge base would require extensive changes to the program.

© Shyamanta M Hazarika, ME, IIT Guwahati

So, it is important for us to realize what is the difference between production system versus conventional computation. There are several differences between production systems and conventional computational systems. In a conventional computational system, it is not that rules or section of programs can all access all units of the database. Here, the global database can be accessed by all of the rules.

No part of it is local to any of them in particular. Rules do not call other rules. Communication between rules occur only through the global database. More important, production system structure is modular. Changes to any of the components can be made independently. Using conventional computation in AI applications is difficult. For if I change knowledge space somehow my facts change. It would require extensive changes to the program. So, production systems capture the essence of programming in AI systems. So, here is the procedure for production.

(Refer Slide Time: 32:11)


Production Systems

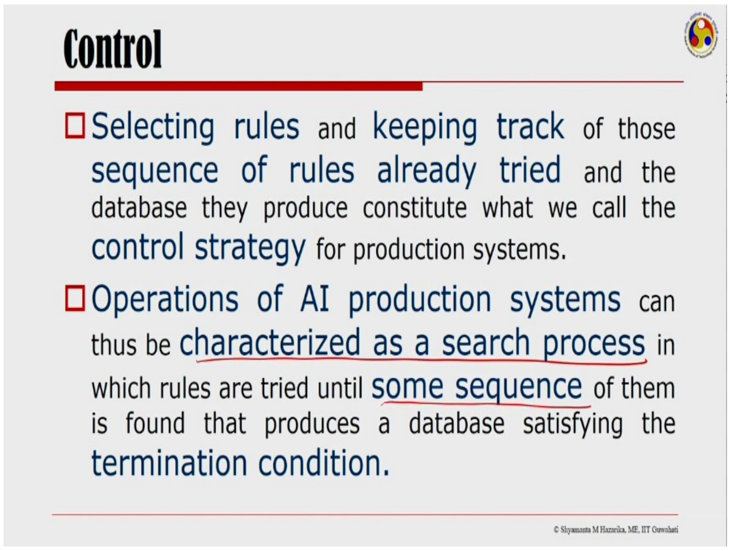
Procedure: Production

```
DATA ⟵ initial database
until DATA satisfies the termination condition;
  do
  begin
      select some rule R, in the set of rules that
      can be applied to DATA
      DATA ⟵ result of applying R to DATA
  end
```

© Shyamanta M Hazarika, ME, IIT Guwahati

So, we have an initial database that is updated to data. And then, we keep on working with the operators until the termination condition is satisfied. So, we select some rule in the set of rules that can be applied to data. We apply that rule to generate new data and we keep on doing this rule. We keep on doing this until we have the termination condition satisfied, in which case we have actually arrived at the goal state. And then we say that the problem has been solved.

(Refer Slide Time: 32:44)

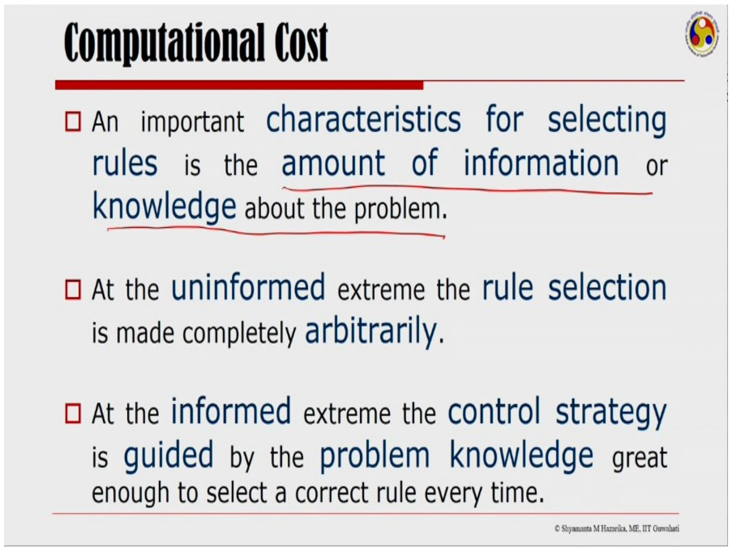

Control

- Selecting rules and keeping track of those sequence of rules already tried and the database they produce constitute what we call the control strategy for production systems.
- Operations of AI production systems can thus be characterized as a search process in which rules are tried until some sequence of them is found that produces a database satisfying the termination condition.

© Shyamanta M Hazarika, ME, IIT Guwahati

Let us now focus at the different control strategies. Selecting rules and keeping track of these sequence of rules already tried and the database they produce constitute what we call the control strategy for production systems. Operations of AI production systems can be therefore characterized as a search process in which rules are tried until some sequence of them is found that produce a database satisfying the termination condition.

(Refer Slide Time: 33:19)



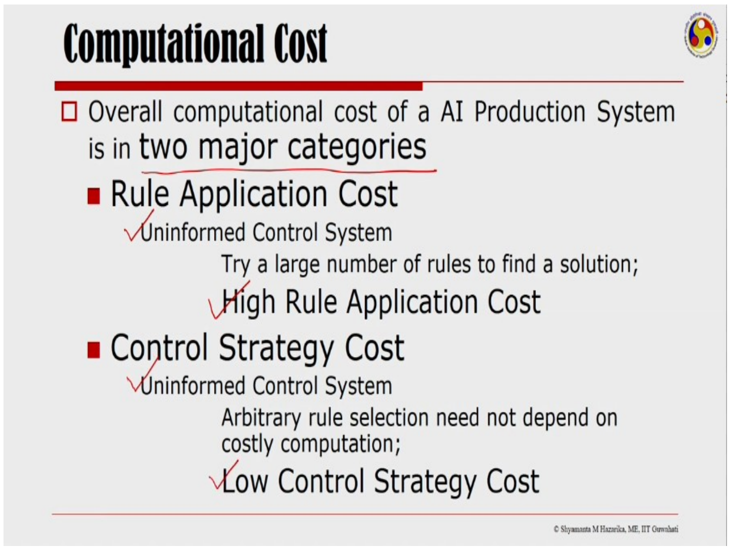
Computational Cost

- An important characteristics for selecting rules is the amount of information or knowledge about the problem.
- At the uninformed extreme the rule selection is made completely arbitrarily.
- At the informed extreme the control strategy is guided by the problem knowledge great enough to select a correct rule every time.

© Shyamanta M Hazarika, ME, IIT Guwahati

It is very important to understand the computational cost involved in AI production system. That comes to us from the understanding of how much amount of information or knowledge is being used about the problem. Now, an important characteristic for selecting rules is the amount of information or knowledge about the problem. At the uninformed extreme, when we do not know anything about the problem, the rule selection can be made completely arbitrary. It does not matter which rule I select, because I do not know anything about the problem at all. At the other extreme, when I know almost everything about the problem, I have the informed extreme. The control strategy is guided by the problem knowledge. And this guidance is great enough to select a correct rule every time.

(Refer Slide Time: 34:16)



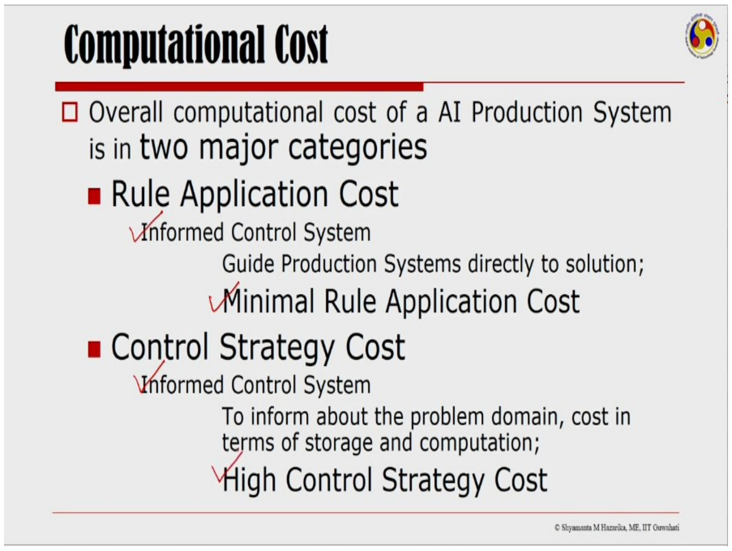
Computational Cost

- Overall computational cost of a AI Production System is in two major categories
  - Rule Application Cost
    - ✓Uninformed Control System
      - Try a large number of rules to find a solution;
      - ✓High Rule Application Cost
  - Control Strategy Cost
    - ✓Uninformed Control System
      - Arbitrary rule selection need not depend on costly computation;
      - ✓Low Control Strategy Cost

© Shyamanta M Hazarika, ME, IIT Guwahati

So, overall computational cost of an AI production system can be thought of to be 2 categories, two major categories of cost. One that comes from the rule application and the other that comes from the control strategy. The rule application cost in the uninformed control system (we have to try a large number of rules to find a solution) and therefore, we have a high rule application cost. Whereas, in terms of control strategy when we have uninformed control system, arbitrary rule selection need not depend on costly computation. Therefore, we have a very low control strategy cost. So, when we have uninformed control system, the rule application cost is very high, for we need to try large number of rules. Whereas, for the same uninformed control system, the control strategy cost is very low; for rule selection will not depend on costly computation.

(Refer Slide Time: 35:24)



Computational Cost

- Overall computational cost of a AI Production System is in two major categories
  - Rule Application Cost

    Informed Control System

    Guide Production Systems directly to solution;

    Minimal Rule Application Cost
  - Control Strategy Cost

    Informed Control System

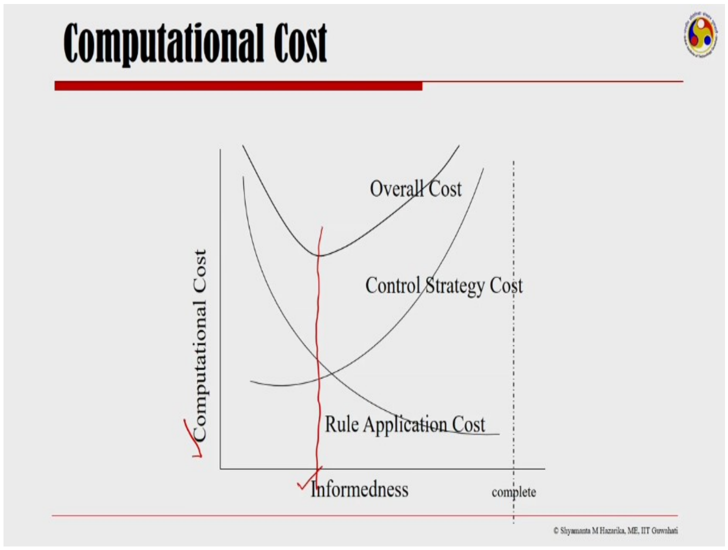
    To inform about the problem domain, cost in terms of storage and computation;

    High Control Strategy Cost

© Shyamanta M Hazarika, ME, IIT Guwahati

Let us look what happens under informed control system. In an informed control system, the production system is directly guided to the solution. And therefore, we have very less rule application cost. Whereas, to inform about the problem domain, there is a huge cost in terms of storage and computation. And therefore, we have very high control strategy cost. So, rule application cost, when I am talking of informed control system is minimal. Whereas, control strategy cost, when I am talking of informed control system is high.

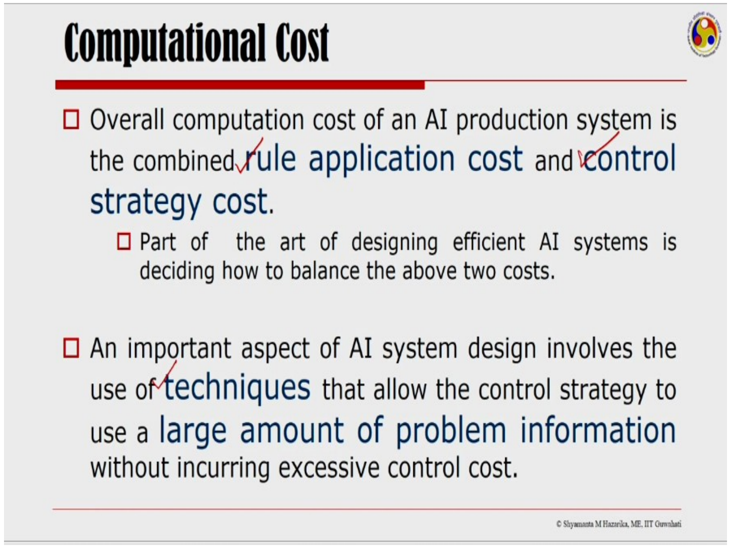
(Refer Slide Time: 36:15)


Computational Cost

So, here is a comparison of the rule application cost and the control strategy cost. This side, we have the computational cost. And here we have the informedness. This is where we suppose that we have complete information about the problem. So, as we have discussed, here is the rule application cost. As informedness increases, the rule application cost decreases. And here is the control strategy cost.

As we increase informedness, the control strategy cost increases. So, here is the overall cost of AI production system. And it is important for us to realize that whenever we are thinking of building one, the amount of information that we really want to exploit is that keeps our overall cost to the minimum. So, too much of information in a production system, they also spoil the way the AI system is built.

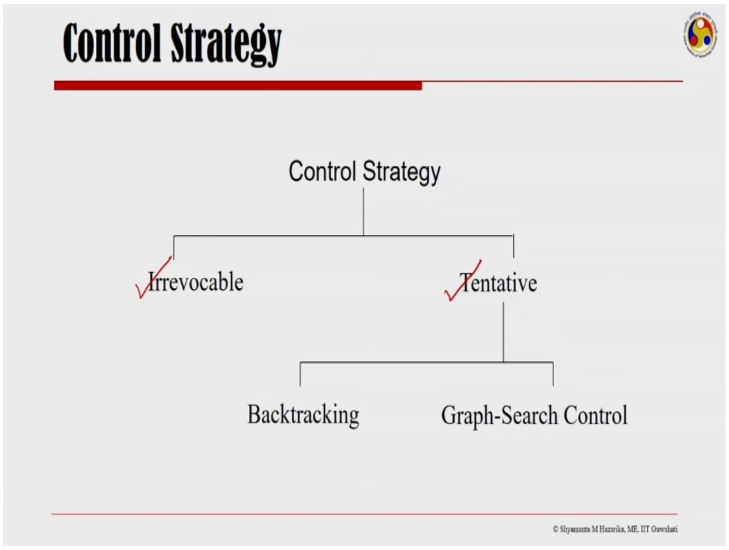
(Refer Slide Time: 37:22)


Computational Cost

- Overall computation cost of an AI production system is the combined rule application cost and control strategy cost.
  - Part of the art of designing efficient AI systems is deciding how to balance the above two costs.
- An important aspect of AI system design involves the use of techniques that allow the control strategy to use a large amount of problem information without incurring excessive control cost.

Overall computational cost of an AI production system is therefore the combined cost of the rule application cost and the control strategy cost. Part of the art of designing efficient AI programs is deciding how to balance the above 2 costs. An important aspect of AI system design therefore involves techniques that could allow the control strategy to use a large amount of problem information without incurring excessive control cost.

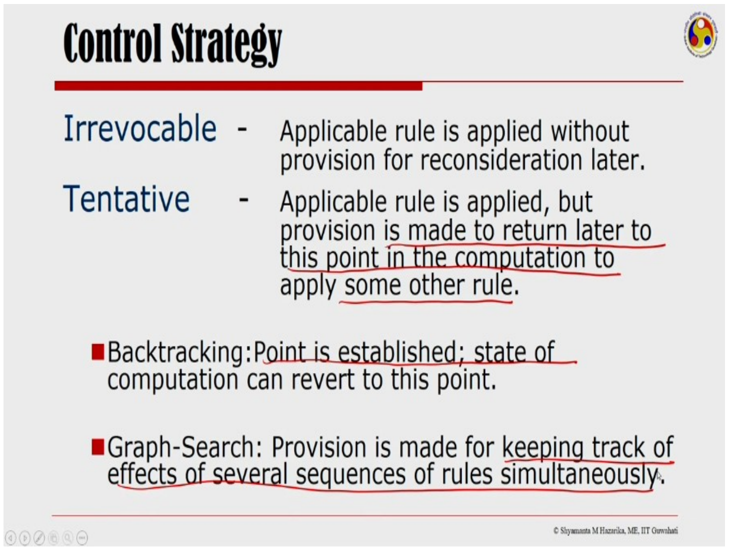
(Refer Slide Time: 37:54)



Control Strategy

So, whenever we are talking of controls and control strategy, we have 2 distinct types of controls; one called irrevocable control and another tentative control strategy. And in under the tentative control strategy, we have backtracking and graph-search control. An irrevocable control is a control strategy where, if a rule is applied once it cannot be retraced. Whereas a tentative control strategy is about applying a rule with a provision to retrace the application of the rule later. Let us see what we mean by each of them.

(Refer Slide Time: 38:31)



Control Strategy

Irrevocable - Applicable rule is applied without provision for reconsideration later.

Tentative - Applicable rule is applied, but provision is made to return later to this point in the computation to apply some other rule.

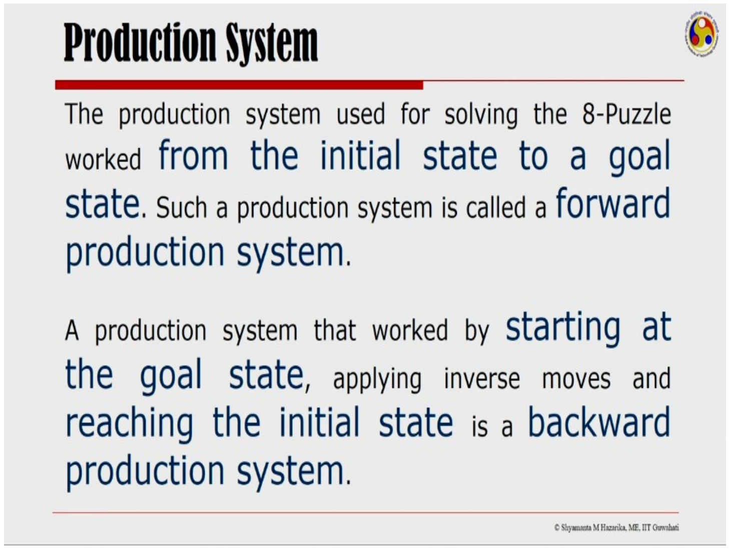
- Backtracking: Point is established; state of computation can revert to this point.
- Graph-Search: Provision is made for keeping track of effects of several sequences of rules simultaneously.

© Shyamanta M Hazarika, ME, IIT Guwahati

So, in irrevocable control strategy, applicable rule is applied without provision for reconsideration later. Whereas tentative, applicable rule is applied, but provision is made to return later to this point of computation, where we could apply some other rule. Under tentative control strategies we have backtracking which is about establishing a point to which the state of computation can revert back if required.

And we have the general graph-search where provision is made for keeping track of effects of several sequence of rules simultaneously; like the one that I showed you for the 8-puzzle game.

(Refer Slide Time: 39:24)



Production System

The production system used for solving the 8-Puzzle worked from the initial state to a goal state. Such a production system is called a forward production system.

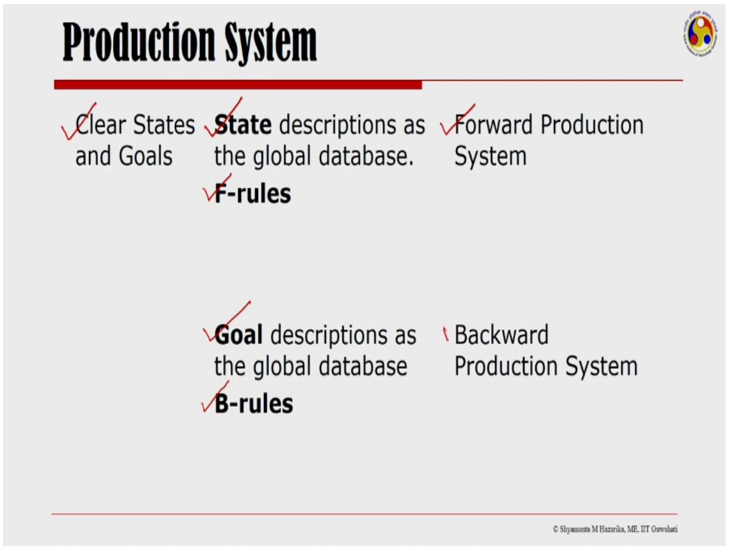
A production system that worked by starting at the goal state, applying inverse moves and reaching the initial state is a backward production system.

© Shyamanta M Hazarika, ME, IIT Guwahati

The production system for solving the 8-puzzle; example that we have looked at today; worked form the initial state to a goal state. Such a production system is called a forward production system. So, we had an initial state, we applied operators on the initial state to arrive that a set of states on which operators were applied again. We did this continuously until we arrived at the goal state.

Such a production system is called a forward production system. A production system that worked by starting at the goal state and applying inverse moves so that I reach the subgoal. I apply inverse rules, reach the subgoal again. In this case, I am going from the goal to the start. Such a production system is called a backward production system.

(Refer Slide Time: 40:25)



Production System

Clear States and Goals

State descriptions as the global database.

F-rules

Forward Production System

Goal descriptions as the global database

B-rules

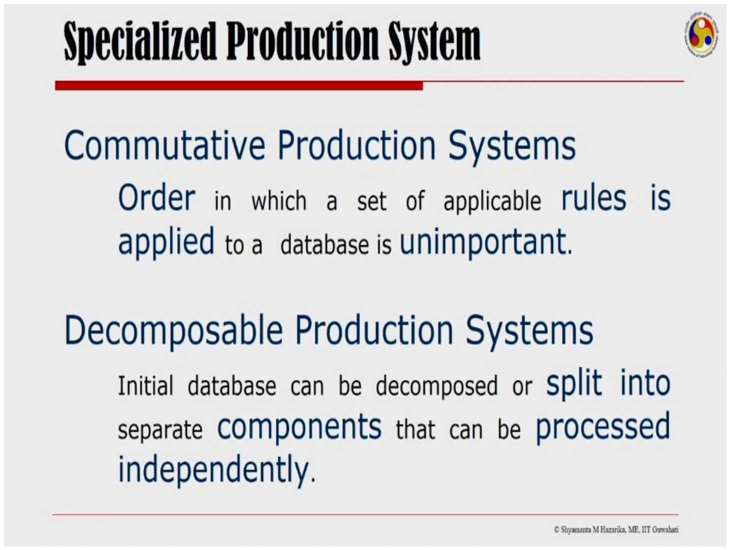
Backward Production System

© Shyamanta M Hazarika, ME, IIT Guwahati

So, in terms of forward rules and backward rules, we could have 2 distinct types of production systems. We need to have clear states and goals. In the first, the state description forms my global database and I have my forward-looking rules as the example of the 8-puzzle that I highlighted, which are referred to as the F-rules. I have the forward production system. On the other hand, if I have goal descriptions as my global database; so, I have the final goal.

The goal just before the final goal, the penultimul goals are my subgoals. So, the whole database that I have, the facts that I generate, are actually goal descriptions somehow. And I have a group of backward rules so that I can start from the final goal, reach the subgoals on the way, finally arriving at the initial state. What I have is a backward production system.

(Refer Slide Time: 41:36)



Specialized Production System

Commutative Production Systems

Order in which a set of applicable rules is applied to a database is unimportant.

Decomposable Production Systems

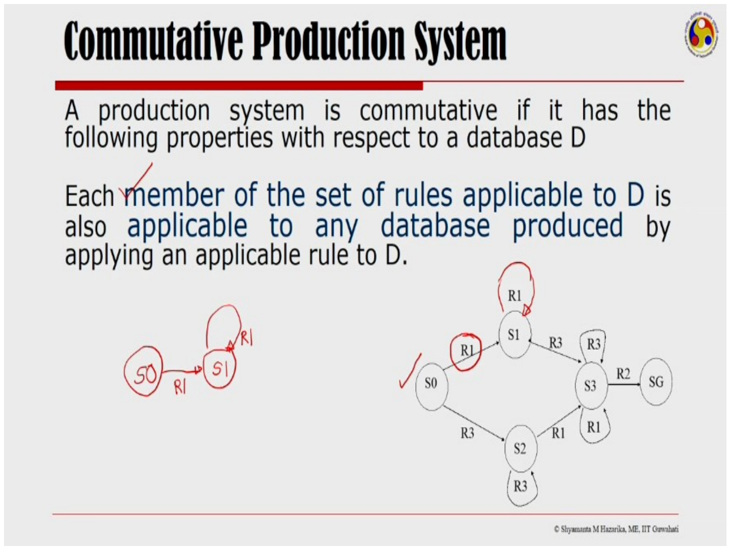
Initial database can be decomposed or split into separate components that can be processed independently.

© Shyamanta M Hazarika, ME, IIT Guwahati

Now, let us focus our attention on 2 specialized production systems. 1, the commutative production system and the other called the decomposable production system. A commutative production system is one in which the order in which a set of applicable rules is applied to the database is not important. I could apply any rule, any other rule still is applicable to the database.

A decomposable production system is interesting because it leads to problem reduction. Here, the initial database can be decomposed or split into separate components and each of them can be processed independently so that had arrived at the final goal. Let us focus our attention in the first one, the commutative production system.

(Refer Slide Time: 42:31)



Commutative Production System

A production system is commutative if it has the following properties with respect to a database D

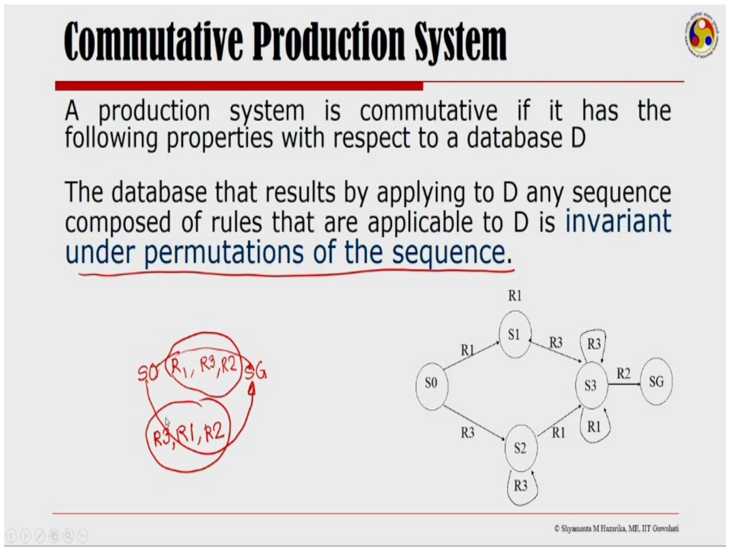
Each member of the set of rules applicable to D is also applicable to any database produced by applying an applicable rule to D.

© Shyamanta M Hazarika, ME, IIT Guwahati

A production system is commutative if it has the following properties with respect to a database D. Number 1. Each member of a set of rules applicable to D is also applicable to any database produced by applying an applicable rule to D. So, let us look at this example that I have highlighting at the bottom of the slide. I have a starting state S0 on which a rule R1 is applicable.

So, I have S0 on which I apply R1 and I arrived at S1. Now, R1 was a member of the set of rules applicable to this. And this is the database produced from this. It is interesting to note that R1 is still applicable to S1. I can still apply R1 to this state if the system that I am talking of is a commutative production system.

(Refer Slide Time: 43:39)



Commutative Production System

A production system is commutative if it has the following properties with respect to a database D

The database that results by applying to D any sequence composed of rules that are applicable to D is invariant under permutations of the sequence.

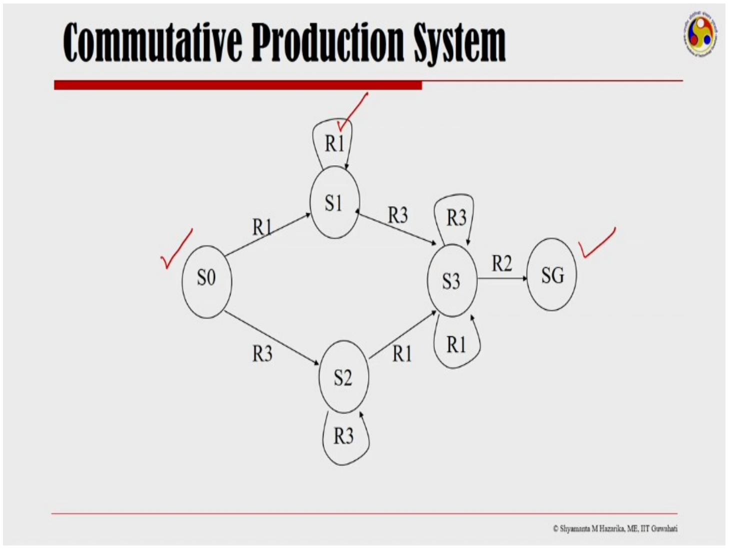
S0
R1, R3, R2
SG
R3, R1, R2
R1
S1
R3
R3
S0
S3
R2
SG
R3
R1
R1
S2
R3
© Shyamanta M Hazarika, ME, IIT Guwahati

Number 2. If the goal condition of a commutative production system is satisfied by D, some database. If the goal condition is satisfied by some database D, then it is also satisfied by any database produced by applying any applicable rule to D. Because of the very fact that I can still apply to all the rules to that database that is produced out of this. So, this is going to be true. So, let us say SG satisfies my rule.

But then, we know all of the rules that were applied up till now could also be applied here: R1 or R2 or R3. And I still with have SG possible. And therefore, the goal condition satisfied by D, then it is satisfied by any database produced by applying any rule applicable to D. Finally, the database results by application to D any sequence composed of rules that are applicable to D is invariant under permutation of the sequence. And that is very important.

Here, I move from S0 to SG. By applying let us say R1, then R3, then R2, I should be able to go not only in this sequence from S0 to SG. I should be also able to go from S0 to SG in another sequence where it could be like R3, then possibly R1 and then possibly R2. If this is possible, then the production system is said to be commutative.

(Refer Slide Time: 45:31)

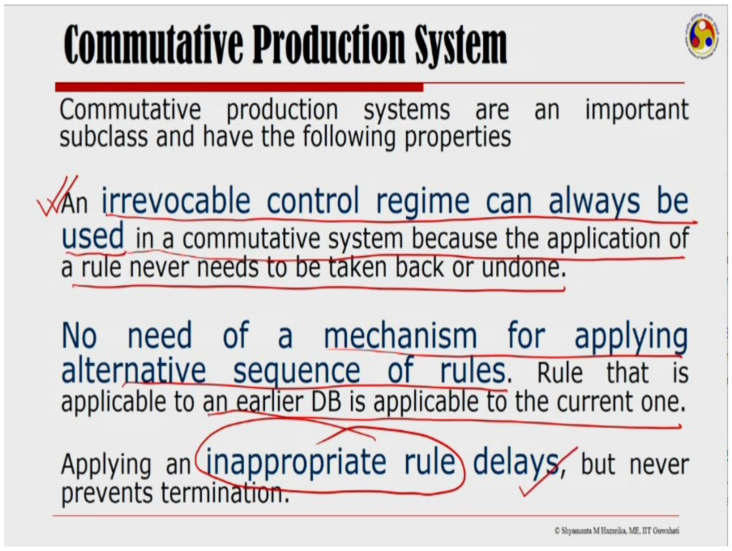


Commutative Production System

So, here I have S0 to S1. I can still apply S1. Then I can apply some rule and reach S3. And then, I can still apply R3 onto S3. And application of R2 I can reach SG. Now, here I have arrived at from S0 to SG using R1, R3 and R2. But any other combination of these 3 could also take me to SG if the system is a commutative production system. So, I could arrive that S2 applying R3. From there I could still apply R1 and arrive at S3.

And form S3 I could apply R2 and arrive at SG again, which is my goal state. So, given this start state and goal state, any rule that is applied in D is still applicable to the database resulting out of D. That is property 1. Any sequence that takes me to the goal; take R1, R3, R2; any permutation of that sequence R3, R1, R2 also takes me to the goal. That is what a commutative production system is.

(Refer Slide Time: 46:46)



Commutative Production System

Commutative production systems are an important subclass and have the following properties

An irrevocable control regime can always be used in a commutative system because the application of a rule never needs to be taken back or undone.

No need of a mechanism for applying alternative sequence of rules. Rule that is applicable to an earlier DB is applicable to the current one.

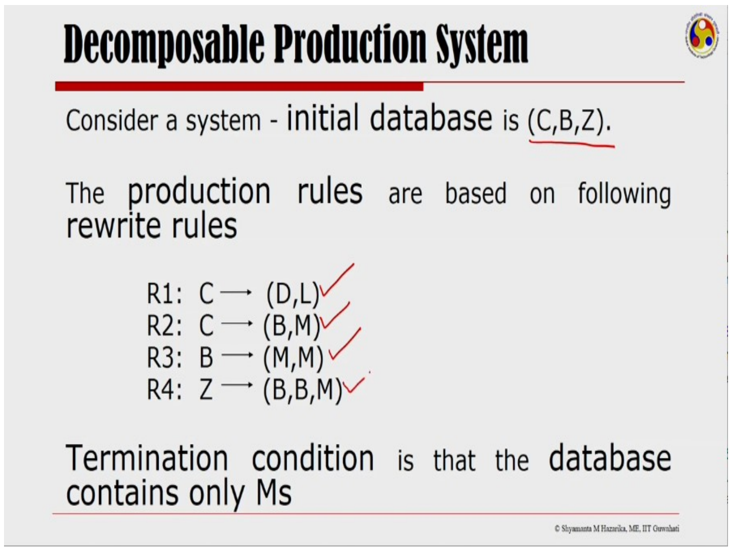
Applying an inappropriate rule delays, but never prevents termination.

© Shyamanta M Hazarika, ME, IIT Guwahati

Commutative production systems are an important subclass and have the following properties. An irrevocable control region can always be used. Because, even if we have done something not onto the way to towards the goal, some other rule is still applicable unto that system. So, in a commutative system M; this is because the application of a rule never needs to be taken back or undone. So, an irrevocable regime can always be used in a commutative system.

Number 2. I do not need a mechanism for applying alternate sequence of rules. Rule that is applicable to an earlier database is applicable to the current one as well. And therefore, I do not have to think of a mechanism of applying alternative sequence of rules. So, one needs to understand that in an commutative production system, I may have inappropriate rules. But the inappropriate rule will only delay but never prevents termination. And this is very very important for commutative production systems.

(Refer Slide Time: 48:15)



Decomposable Production System

Consider a system - initial database is (C,B,Z).

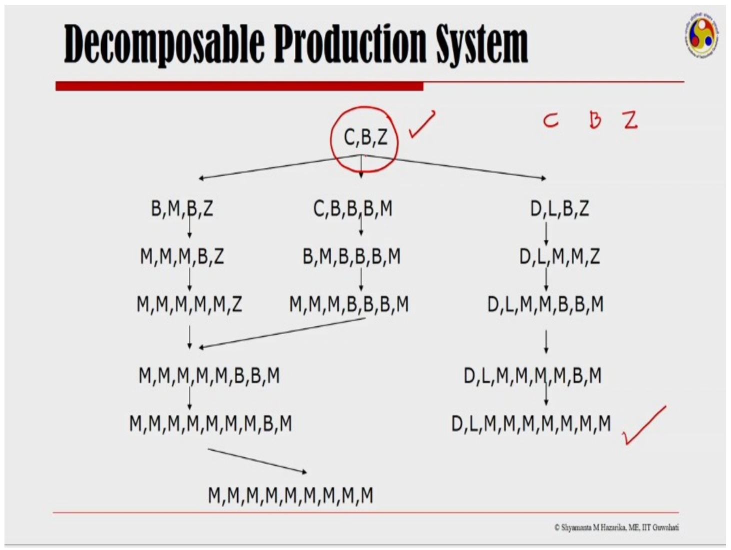
The production rules are based on following rewrite rules

R1: C ⟶ (D,L)
R2: C ⟶ (B,M)
R3: B ⟶ (M,M)
R4: Z ⟶ (B,B,M)

Termination condition is that the database contains only Ms

Let us now look at the other form of production system that we have highlighted in the beginning; the decomposable production system. In order to understand decomposable production system, let us consider an initial database which is C, B and Z. And the production rules are based on the following rewriting rules. C could be written as D L; C could be written as B M; B could be written as M M; and Z could be written as B B M. I am looking for a termination condition when the database contains only M.

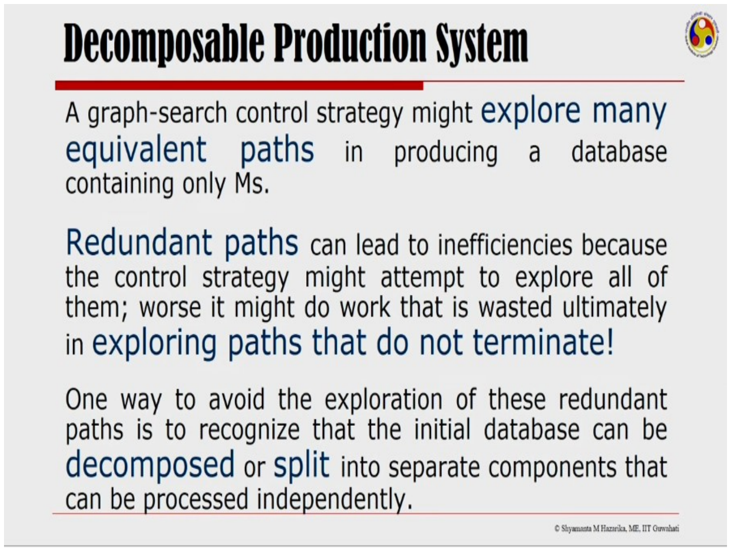
(Refer Slide Time: 48:57)



Decomposable Production System

So, if I start with the complete C B Z, I have 3 reiterating rules for C, B and Z. So, my path would be 3 different distinct paths. I could write C or B or Z. And each of them could take different paths. And what is important for me to note that some path may lead to termination. But, some other path may not lead to termination for me. Instead of looking at C B Z as a complete single entity, the decomposable production system allows me to look at C B Z as 3 different units of C B and Z. This is because of the problems that I get when I look at it as 1 single unit as highlighted here.

(Refer Slide Time: 49:59)



Decomposable Production System

A graph-search control strategy might explore many equivalent paths in producing a database containing only Ms.

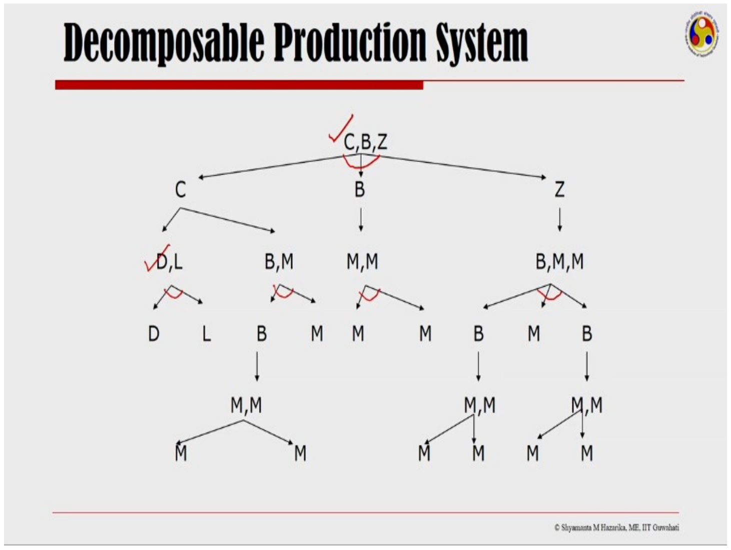
Redundant paths can lead to inefficiencies because the control strategy might attempt to explore all of them; worse it might do work that is wasted ultimately in exploring paths that do not terminate!

One way to avoid the exploration of these redundant paths is to recognize that the initial database can be decomposed or split into separate components that can be processed independently.

© Shyamanta M Hazarika, ME, IIT Guwahati

So, when a graph search control explore many equivalent paths in producing a database containing only Ms. Redundant paths can lead to inefficiencies, because the control strategy might attempt to explore all of them and also explore paths that do not terminate. So, one way to avoid the exploration is to recognize that the initial database can be decomposed or split into separate components that can be processed independently. As I highlighted in the previous slide that C B Z can be looked at as 3 distinct elements of C, B and Z. So, this is what we do here.

(Refer Slide Time: 50:44)



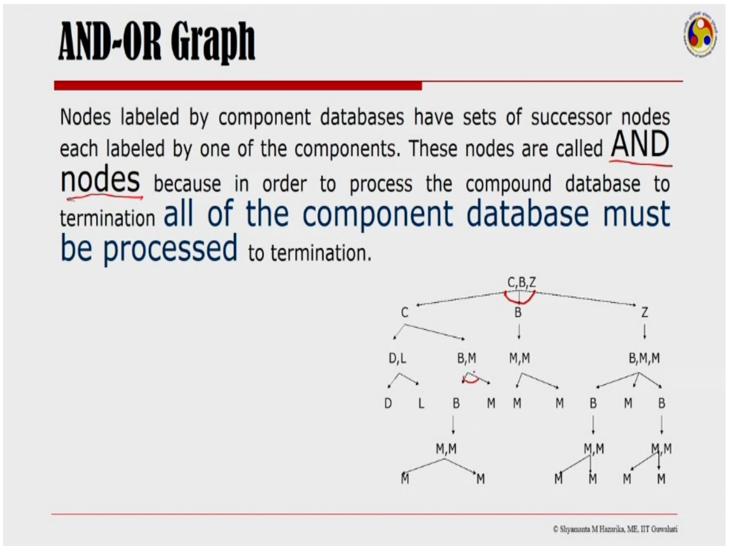
Decomposable Production System

We break C B Z into C, B and Z and then apply the rewriting rules that we have highlighted initially. So, C can be written as D L or B M, B is written as M M and Z is written as B M M. D L can be changed as D to D and L, B M is B M M, M M is M M and B M M is again split as B M B. So, we had an initial C B Z. We split it up as C, B and Z. Applied the rewriting rules to get D L, B M, M M, B M M.

And then again split this into D and L, B M into B and M, M M into M and M and B M M as B M B. Okay. So, then we start applying the rewriting rules and apply only to this B. And we get a string where all of these Ms are satisfied. Now, one thing to note here is very important property that we want to explore. If you look at here; at this point, all of C, B and Z needs to be brought to termination. All of these parts are important.

But at this point, when I think of rewriting C as either D L and B M, I can rewrite only 1 way. So, one of them is important for me, not both. Whereas, again at the next lower level, both of them needs to be looked at. All of the 3 needs to be looked at here. But again, here when I am looking at, I look at only 1 alternate path. But here there are no alternate paths by 1 path. So, it became simpler here.

(Refer Slide Time: 52:51)

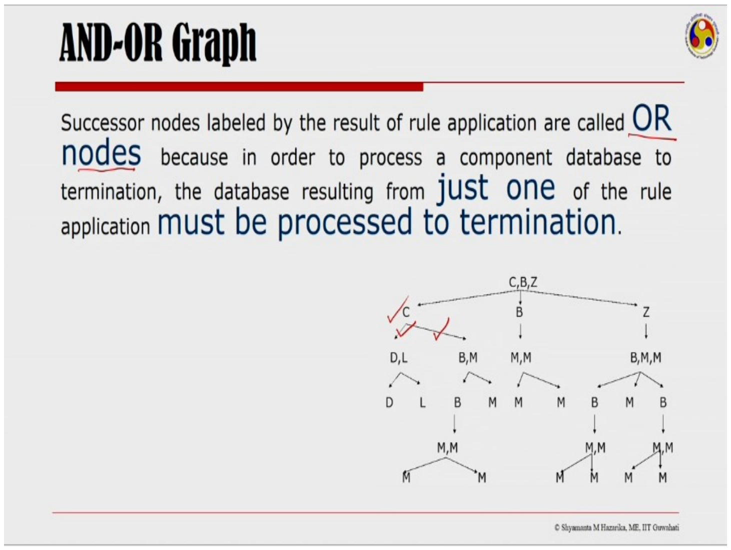


AND-OR Graph

Nodes labeled by component databases have sets of successor nodes each labeled by one of the components. These nodes are called AND nodes because in order to process the compound database to termination all of the component database must be processed to termination.

So, it is important that nodes that are labeled by component databases have sets of successors and each labeled by one of the components. These nodes are referred to as and nodes. They are called and nodes because, if you want to process the compound database to termination, all of the component databases must be processed to termination. So, this node here, in order to process C B Z to termination, C, B and Z needs to processed to termination. In order to process B M to termination, both B and M needs to be processed to termination; so on and so forth.

(Refer Slide Time: 53:33)



AND-OR Graph

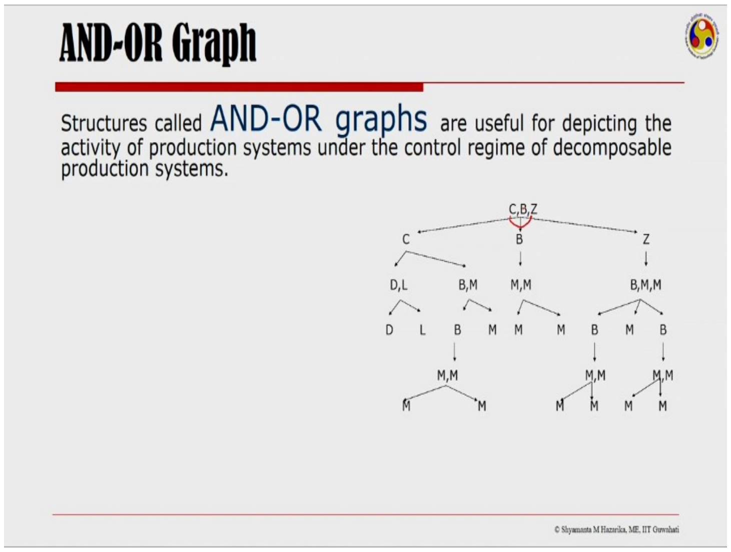
Successor nodes labeled by the result of rule application are called OR nodes because in order to process a component database to termination, the database resulting from just one of the rule application must be processed to termination.

C,B,Z
C
B
Z
D,L
B,M
M,M
B,M,M
D L B M M M B M B
M,M
M M
M,M
M M
M,M
M M

© Shyamanta M Hazarika, ME, IIT Guwahati

Another important thing that one needs to realize is that the successor nodes labeled by the result of rule application; like here, when I have D and L or I have B and M as I applied a rule onto C. They are referred to as the or nodes. Because, in order to process a component database to termination, the database resulting from only either this or this must be processed to termination. These are referred to as or nodes.

(Refer Slide Time: 54:11)



AND-OR Graph

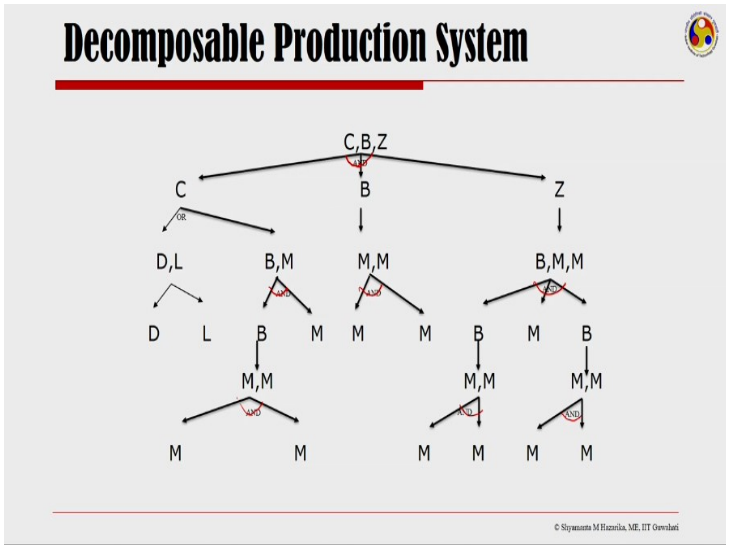
Structures called AND-OR graphs are useful for depicting the activity of production systems under the control regime of decomposable production systems.

C,B,Z
C
B
Z
D,L
B,M
M,M
B,M,M
D L B M M M B M B
M,M
M M
M,M
M M
M,M
M M

© Shyamanta M Hazarika, ME, IIT Guwahati

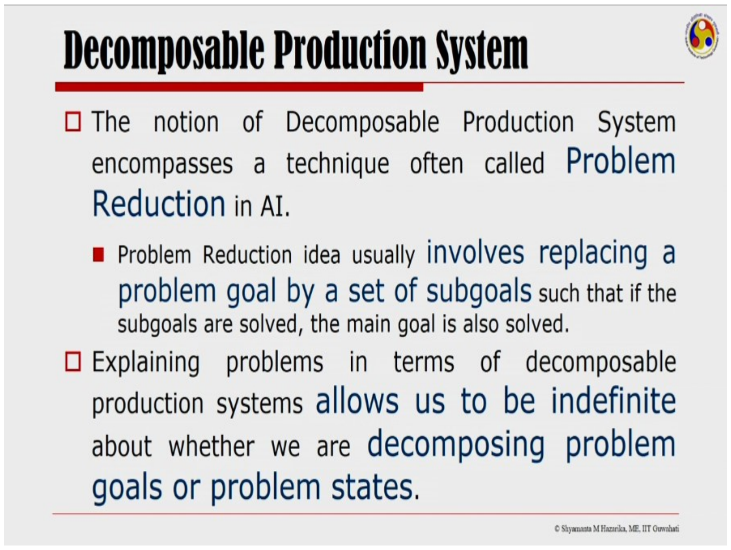
So, structure called and-or graphs are useful for depicting the activity of production systems. So, these are the and nodes, these are the or nodes.

(Refer Slide Time: 54:25)


Decomposable Production System
C,B,Z
AND
C
B
Z
OR
D,L
B,M
M,M
B,M,M
AND
AND
AND
D
L
B
M
M
M
B
M
B
M,M
M,M
M,M
AND
AND
AND
M
M
M
M
M
M
© Shyamanta M Hazarika, ME, IIT Guwahati

So, decomposable production systems if you see, these nodes that I have marked now as AND here are important because if a component database is; all of the component databases need to be moved to termination. So, that is the and node here. That is the and node. This is the and node. Both of them needs to be moved to termination.

(Refer Slide Time: 54:53)



Decomposable Production System

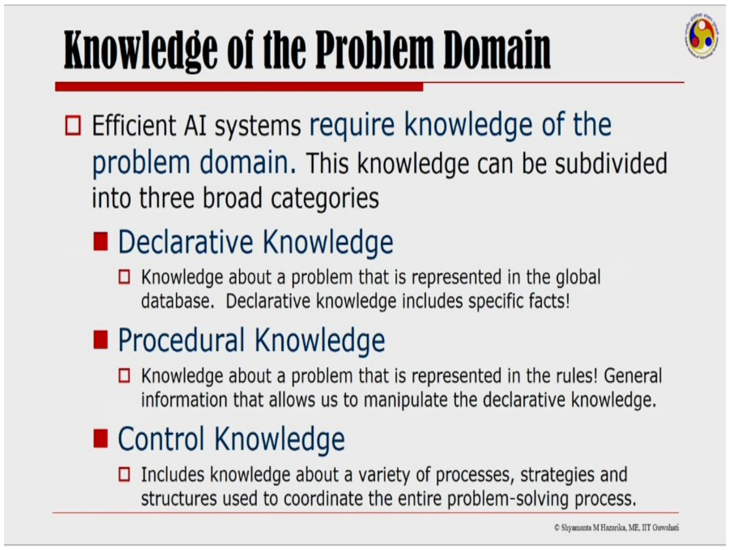
- The notion of Decomposable Production System encompasses a technique often called Problem Reduction in AI.
  - Problem Reduction idea usually involves replacing a problem goal by a set of subgoals such that if the subgoals are solved, the main goal is also solved.
- Explaining problems in terms of decomposable production systems allows us to be indefinite about whether we are decomposing problem goals or problem states.

© Shyamanta M Hazarika, ME, IIT Guwahati

So, finally we can say that the notion of decomposable production systems encompass a technique which is often called problem reduction in AI. We have reduced the problem of taking C B Z to termination. To the problem of taking C, B and Z individually to termination. This is something like the things that we have been doing in high school of doing a breaking up a bigger integration to do it by integration by parts.

So, problem reduction idea usually involves replacing a problem goal by a set of subgoals. Such that if the subgoals are solved the main goal is also solved. Explaining problems in terms of decomposable production system allows us to be indefinite about whether we are decomposing problem goals or problem states.

(Refer Slide Time: 55:47)



Knowledge of the Problem Domain

- Efficient AI systems require knowledge of the problem domain. This knowledge can be subdivided into three broad categories
  - Declarative Knowledge
    - Knowledge about a problem that is represented in the global database. Declarative knowledge includes specific facts!
  - Procedural Knowledge
    - Knowledge about a problem that is represented in the rules! General information that allows us to manipulate the declarative knowledge.
  - Control Knowledge
    - Includes knowledge about a variety of processes, strategies and structures used to coordinate the entire problem-solving process.

© Shyamanta M Hazarika, ME, IIT Guwahati

To finish our lecture today, we would love to highlight that the efficient AI systems require knowledge of the problem domain. These knowledge can be divided into 3 broad categories: a. declarative knowledge, b. procedural knowledge and c. control knowledge. Declarative knowledge is knowledge about a problem that is represented in the global database. Declarative knowledge includes specific facts.

Procedural knowledge is about a problem that is represented in rules; general information that allows us to manipulate the declarative knowledge. Control knowledge includes knowledge about a variety of processes, strategies and structures used to coordinate the entire problem solving process. And this is what we will look at in the next lecture of this module on graph search techniques; to look at what are the problem-solving processes. Thank you very much.