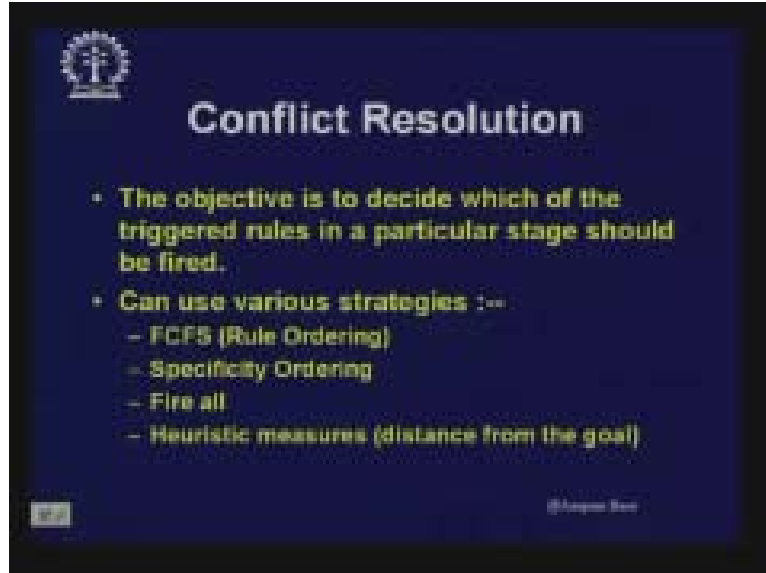


**Artificial Intelligence**  
**Prof. Anupam Basu**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture - 17**  
**Rule Based Systems II**

In the last lecture we had discussed about rule based systems we have introduced the concepts of rule based systems and in today's discussion we will further touch upon the aspects which we did not cover earlier as we will recall we had talked about automata that the inference machine of a rule based system executes and consists of three states or three phases first is the match phase where the rules are compared the antecedents of the rules are compared antecedents are later on we will see that we can also compare the consequents but for the sake of simplicity for the time being let's assume that the rules and conditions of the rules are compared with the existing facts in the fact base and if there is a match the rules whose antecedents matched are triggered and they are fed to the conflict resolution strategy in the conflict resolution strategy we will select subset of those matched rules and feed it to the execute phase and the execute phase we fire those rules and as a consequence of firing these rules the new facts are generated and the fact base gets updated. We go on carrying out this activity till either the goal is met or there are no more rules to fire.

Today we will start our discussion on the conflict resolution strategy. You should also recall that we had said that the execution of a rule based system is equivalent to exploring a search space. We can therefore think of search space consisting of nodes and arcs which are created and different rules that are being fired are equivalent to the required path of that space. Now, which rule to fire is determined by the conflict of resolution strategy. Therefore conflict resolution strategy is a key factor till the execution of rule based systems.

(Refer Slide Time: 3:57)



Therefore, today we can see that the objective of conflict resolution strategy is to decide which of the triggered rules in a particular state should be fired. We can use different strategies like the first come first served, rule ordering, specificity ordering etc.

Now, what is meant by the first come first served strategy?

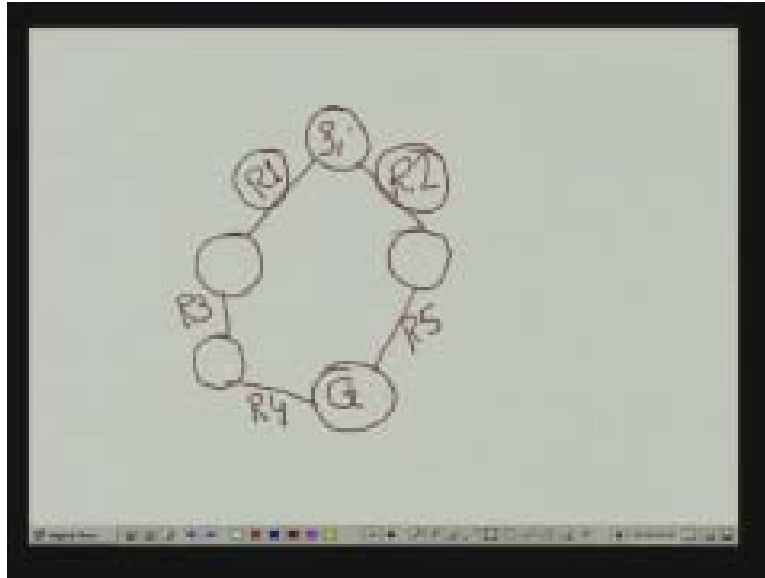
Let me try to elucidate this point. When we have got a number of rules then the rules are ordered in some ways and some rules are written first and some rules are written later. Now, if the conflict resolution strategy is first come first served based then it will select just the rule out of the triggered rule the rule that appears first. So, if you have to give some priority to the rules then the rule writer or the rule designer should write the rules in the proper order of priority because the conflict resolution strategy is rather simple, it does not do anything special but it just selects the first rule to fire.

On the other hand, the other strategy known as the specificity ordering strategy is different. It is different in the sense that it will select the rule that is more specific. Let us consider a scenario. It says: if A then X. Another rule says: if A and B then X. I repeat; if A then X where A is the predicate then X. And the second rule says if A and B then Y. now which of the rules are more specific? Assume that both the facts A and B are true and they are in the fact based. So both these rules are triggered, now which rule is more specific? Obviously the second rule if A and B then Y, now what should you conclude? X or Y? Both A and B are true in the fact based. Our common sense tells us that we should infer Y because the rule if A and B then Y uses more facts which are known to be true.

Suppose if there are two rules; if it is a Sunday it is a holiday, another rule says; if it is a Sunday or for example, if it is summer vacation then it is a holiday. Now whenever we know more facts in that case that rule is more specific because it is using more

information. Therefore, in the specificity ordering the rule that is selected is based on the rule which is more specific whose more number of antecedents are matching.

(Refer Slide Time: 8:30)



Then the other rule is fire all. Fire all means whichever rules are enabled will be fired. If there are five rules enabled those will be fired. The other strategy is heuristic measure. heuristic measure, as we were discussing in the last lecture, heuristic is based on partial information about the domain and it also can use some of our general knowledge, some of our past experiences about solving some problem, about a particular scenario. So, if there are multiple paths to a goal we should select the path that is more probable to lead to the goal.

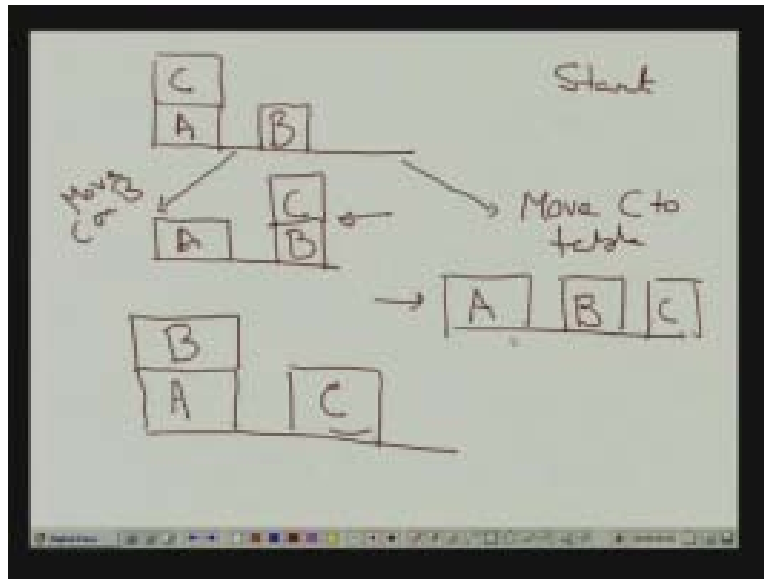
For example let us say I am in this state. Suppose I am in a particular state  $s_1$  and my goal can be here (refer Slide time: 8:58) goal node can be here. Now there are different rules which can be fired. If I fire rule R1 then I will follow this path, if I fire rule R2 then I will follow this path, and each of these firings will take me to different states. Now maybe again from this state I will have another rule to fire R3 and R4 which will lead me to the goal whereas from here there is a possibility that I will fire another rule R5 where I can reach the goal. So when I am in this state  $s_1$  obviously this R1 and R2 are in conflict so which one should I select?

If I see it over here and look at this state and can also find out which one of these rules, or I can guess which one of these R1 or R2 will bring me closer to the goal then that can be used as one of the strategies for selection. It is not the case that always it will lead me to the minimum path or the shortest path or the least cost but we have got no other way but to guess. We have to guess at particular times and based on that we have to make a decision.

Suppose I have got some bloc here A B and C. these are three building blocks which are kept in this way. Now this is my start state and I want to arrive at a goal state which is

something like this (Refer Slide Time: 11:05) where I should have A and on that B and C might be here. Now from here if I can move C to table which will lead me to a particular state where it will be A B and C. There could be another move another rule which can say that move C on B. In that case it will lead me to a particular state which will be A B C. This is one candidate and this is another (refer Slide time: 12:10) now which of one these two states will bring me closer to the goal state? It is obviously this one because in order to move B the top of B must be free and then only I can straight away move B and bring it over here.

(Refer Slide Time: 12:30)



If I go to this state then I need one more state where I can free up C and bring it to the top of B. So this state is not as close to the goal state, this is the goal state, it is not close to the goal state as this state is. Therefore, I need some strategy by which I can see that this rule is resulting in a particular state which does not have the top of B. My heuristic is this that in order to move B it is preferable that if the top of B is free. Otherwise now I have to make the top of B free. This preferred situation is being brought in by this rule so obviously in a heuristic measure if I have got a heuristic strategy this would be a better choice and this would bring me quicker to the goal. That is what we mean by heuristic measure and that is how we evaluate the distance to the goal in some way. And what is very much important is that in what way it is dependent on the problem we are solving.

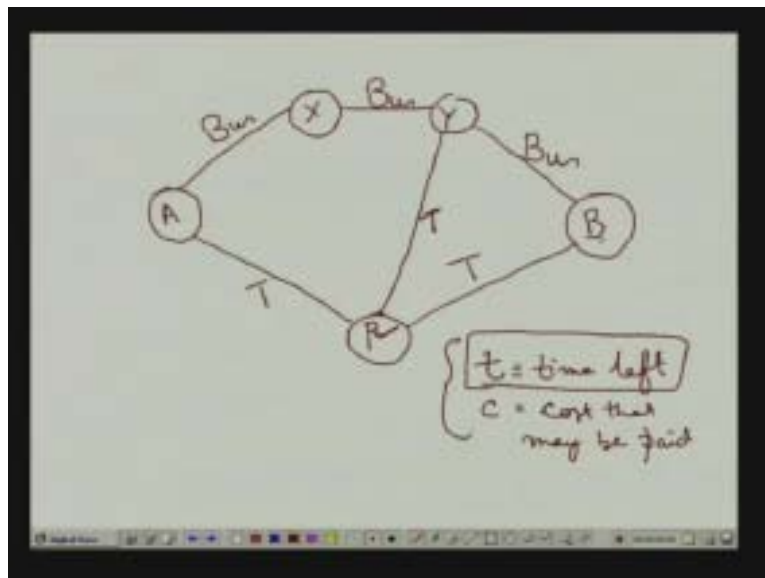
In the block's world in the example I have just now given it is important to know how to move a particular block that we really desire to move but may be in some other scenario the cost or the measure we adopt may be different. For example, when we make a travel from one particular city to another city. In that case you also want to minimize the amount you want to spend and also you want to minimize the time. Now, how you do that?

One is obviously walking which is one possibility, another possibility is by bus and another is by air, may be by train etc and there my heuristic measure will be little different.

For example, if I am in this city A and I want to move to this city B (Refer Slide Time: 14:50) and I have got possibilities of going to city X then city Y and then from here to this. So there is a bus route here, this is bus and there is one train which brings me to city P and from this there is another train which brings me to city Y and from there I have to take a bus. Or there is another way, that from here I can also take a train to B.

Now my goal is to reach B. when I come to city P and from this point if I decide ok right now I have got enough money but my time is really an issue so I come over here. then at this point I have got two choices; either I take the train from here, this train or there is a direct train, but this train if I take then I will have to take a bus again. So here I find that ok I have got some T amount of time where T is time left, I have to reach within that time and C is the cost that may be paid.

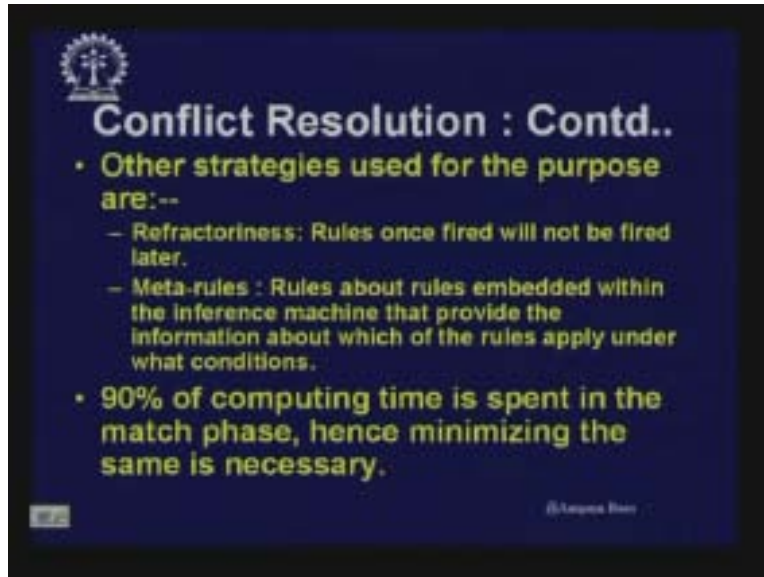
(Refer Slide Time: 16:50)



Now with these two parameters and sitting at a point I have to make a decision. Now obviously this time is very important so here probably I will not think so much about the cost and I will take this because this is a faster train that will go directly. But depending upon the time, that is if I have enough time I may prefer to go by a cheaper local train and take the bus. So, for this at this point I will have to compute a cost function that will consist of some parameter or some weight of time left plus....this is a very simply expression I am writing to the cost that has to be paid. Therefore some sort of function I have to compute sitting over here. The cost function can vary for different problems. Therefore, depending on the cost function I will be selecting a particular path whichever satisfies my need better. So I hope you have understood what is meant by heuristic

functions and you have also come across when you learnt about A star algorithm where you have looked at heuristic measure.

(Refer Slide Time: 17:55)



The other strategy used for this purpose are refractoriness. Now, this term means that rules which are once fired will not be fired later, that is one thing we have seen in the earlier lecture. Another very important approach is to use meta tools. Meta knowledge means knowledge above knowledge so it is one level higher. Meta rules are rules which encapsulate the knowledge of how to use rules so whenever there is a conflict we use those rules to resolve that conflict so such rules are known as meta rules. So rules about these rules are embedded within the inference machine that provide information about which of the rules apply and under which condition.

The execute phrase is just taking the decision. The performance of a inference machine in a rule based system is very much dependent on the match phase and the conflict resolution phase but 90% of the time is spent in the match phase. The rule bases system designers have different measures of implementing rule bases systems using different data structures which minimize the match phase. Just remember that the match phase is a very important parameter and very interesting algorithms like .....20:02) and other techniques have been invented which minimize the overhead of this matching. So the performance of a rule based system is very much dependent on that.

(Refer Slide Time: 20:20)

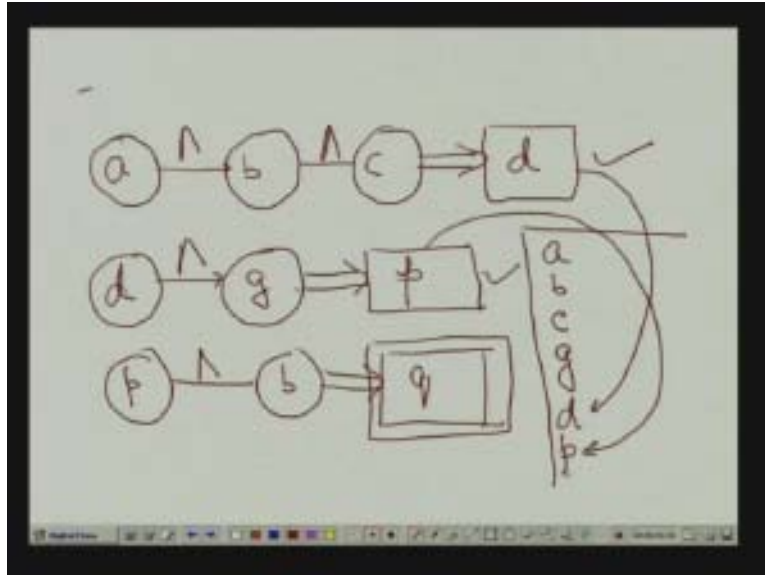


However, now I would like to emphasize on this point time and again that what we really do in a rule based system is searching a space. We try to find a particular space. I prefer to always present this sort of a diagram and say we have a start state (Refer Slide Time: 20:44) and there are different parts to which I may go. Now might be my goal node is here, this is my goal node where I want to reach and I can reach there through different paths might be through these paths or there might be shorter paths whatever. Now, all these are just like the trees that we have drawn earlier. So we have to search this space.

Assuming that the goal is there within this search space we have to find it out. Now, in the case of a rule based system how is this search space defined?

I can say that  $a \wedge b \wedge c$  implies  $d$ . So in this diagram I am just drawing the consequents as squares and the antecedents as circles. There may be another rule  $d \wedge g$  is giving rise to  $p$ . here (Refer slide Time: 22:29) these links are AND links. And there is another rule which says  $p \wedge b$  leading into  $q$  and  $q$  is my goal. And initially I may have in my fact base that  $a$  is true,  $b$  is true,  $c$  is true and  $g$  is true where suppose that is my starting scenario.  $a$  is true,  $b$  and  $c$  being true I first check this, check this and check this so this rule fires and as soon as it fires so this fact  $d$  is added to the database so it is  $d$ . Now as soon as  $d$  is fired I can look at  $d$  so  $d$  and  $g$  is also true so this rule also fires and this adds  $p$  to my fact base.

(Refer Slide Time: 23:30)

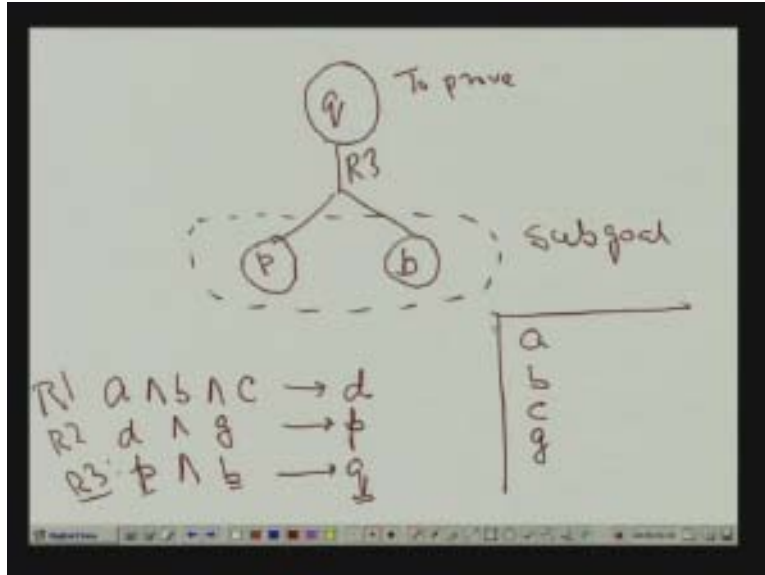


Now  $p$  is true and  $d$  is true so I arrive at the fact  $q$  which is my goal and so my goal is solved. That is one particular way of looking at the whole thing. Therefore, what happens in this case is, in this case I am starting my facts and trying to infer new facts and in the process of inferring new facts I first infer  $d$ , then I infer  $p$  and then  $q$  and in the process of inferring these facts I arrive at the goal, this is one way of looking at it. Otherwise I could have done it in another way. Since I know  $q$  I could have started from  $q$ , how?

My objective is to prove  $q$ , and what were my rules, if I recollect properly?  
 $a$  and  $b$  and  $c$  generated  $d$ ,  $d$  and  $g$  generated  $p$ , and  $p$  and  $b$  generated  $q$ . And my fact base was having  $a$   $b$   $c$   $g$  initially. But here I look at the problem in a different way, my problem is to prove  $q$ . Till now in our discussion we are continuously saying that we compare the fact base with the rule base. Now, if I start looking at the rule base and see, well here there is a rule which says  $q$  has got its consequent. Now when is  $q$  true?  $Q$  is true if  $p$  and  $b$  are true. So if I can prove  $p$  and  $b$  then obviously  $q$  is true. So this is my goal and in order to prove  $q$  using this rule, rule 1  $R_1$   $R_2$  and  $R_3$ , so using  $R_3$  what I can do is I can see that using this I am having two different goals to prove  $p$  and  $b$ . so, what I have created is a set of sub goals. That means if these two things are proved then  $q$  is proved, my objective is reached. So these are the sub goals. So starting from a goal I generate a goal.



(Refer Slide Time: 26:25)

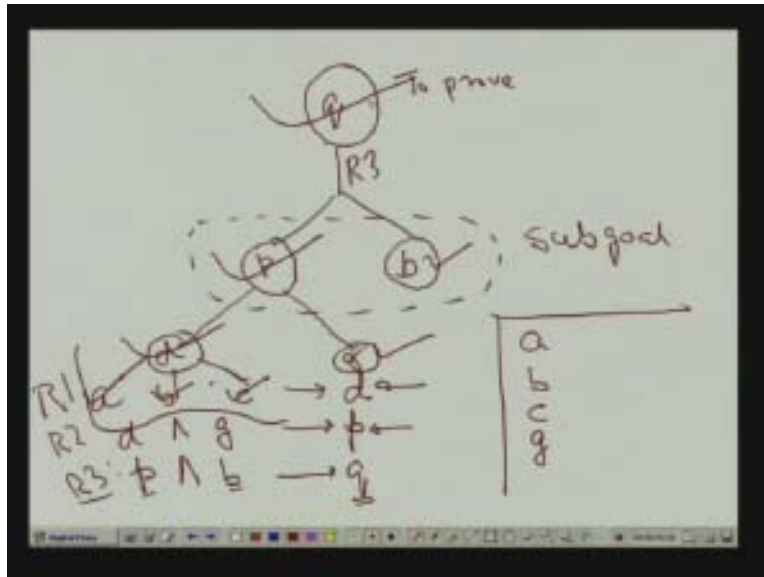


Now immediately I look at, are these already true? I can find, well, in my fact base it is true so this is proved. But this just being proved does not prove q but I have to prove p also. So the sub goal that I have to prove now is this one (Refer Slide Time: 26:45). Then I look at the rule base and find out which rule can help me in finding p. I feel well this is the rule that helps me in proving p. And how can I prove that? It is if I can prove d and g. so I create another set of sub goals d and g which I have to prove.

Again I look at the database and find, well g is proved. My only problem is with d. now is d true? How can I prove d?

I can go to this rule and find out a b and c so these have to be proven. Therefore, I can say d is proved if a b and c are proved. Now luckily I can see that in my database a b and c are true so this is proved, this is proved and this is proved therefore, d is proved and g is proved already, so d and g both being proved p is proved. Now p being proved and b being proved already q is proved.

(Refer Slide Time: 28:10)



Therefore, in this case I have started with a goal and tried to prove the goal by generating sub goals and proving them. So we have seen two distinct ways of searching the space. So the example I gave now must have made it clear. We start from the given facts and try to arrive at the goal, that is the first thing we did. We started with a b c and as soon as I saw a b c I could derive d and since d was there and g was also true I could derive p and since p was there and it was proved and b is there so I proved my goal q. That is one way, that starting from the given facts I have tried to arrive at the goal.

The second one we started is with q and we tried to prove the goal using the given facts. We tried to prove q and we found that q is true if p and b are true. So our next job was to prove p and b. we saw that b was already proved but p was not so we then searched for how we can prove p and we found a rule that if p is true then b and g are true. Now g is already true now how can we prove d. we found a rule which has got d in its consequent and in order to prove d we have to prove a b c so in that way we proceeded. This is another way where we start from the goal node and try to prove the goal using the fact.

The second approach is known as goal driven search and this first approach starting from the given fact to arrive at the goal is known as data driven search. We will now concentrate in reasoning mechanisms that are used in rule based systems.

(Refer Slide Time: 30:28)

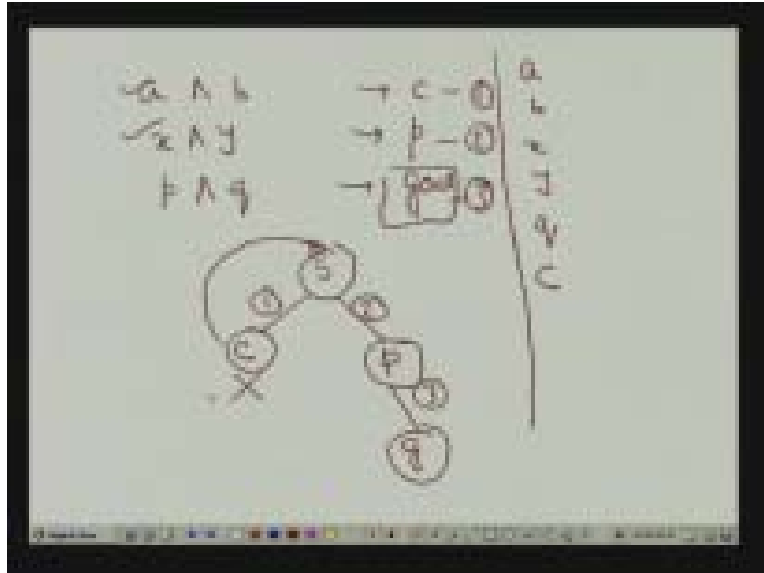


The first one is forward chaining mechanism which is a data driven search. We have already explained it but still let us look at it. Starting from the start state we apply the rules one by one to arrive at the goal state and we have said this is also known as data driven search. The forward chaining may lead search to a dead end. We may reach a dead end because we might have arrived at a particular point, we have arrived at a set of particular facts which does not lead us to any further exploration of the rule space. In such cases we need to do backtracking.

Backtracking can be chronological or intelligent. Let me explain this again. I have got a rule; a and b leads to c, x and y leads to p, and p and q leads to g which is my goal, suppose you have three rules. And suppose in my fact base I have a to be true, b to be true, and x to be true, y to be true and q to be true. I am just taking a very simple scenario. Now, according to the rule bases system we know, initially which are the rules usually triggered? We know that this rule is triggered and this rule is triggered. So having the start state and having two rules to fire rule R1 and this one..... and when I start with my start node I have got rules one and two both enabled, this is enabled and this is enabled so I had two possible paths so somehow I selected path one so I have come to c.

Now with this I see that I cannot proceed any further with this new fact. C has been added but if my strategy is that always select the rule or just fire the rule which is a newly generated fact then c does not lead me anywhere, I cannot proceed further. Therefore, at this point I have to do backtracking. I have to go over here and look at the other rule that was left to be fired. So I look at this rule I get p then q was true so from there by firing rule three I can get to the goal g. But this was a dead end that I reached. Therefore I had to backtrack and see which rule I have to fire.

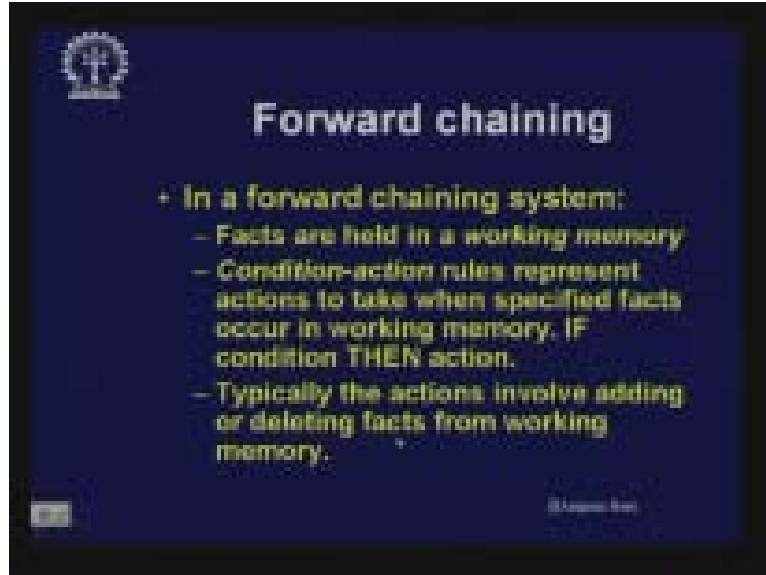
(Refer Slide Time: 34:34)



You would have known from your data structure and algorithm class that backtracking usually goes one level up from any tree. So, in general what happens is, whenever I have a tree like this (Refer Slide Time: 35:15) which I am exploring either way, suppose I have reached some dead-end here then usually I backtrack one level up and look at the other possibilities and try over here, in a typical depth for search. Now if I find the dead-end also here then I go up again here and I find that there is no further way of exploring here then I go one level up and try to proceed this way. Now this is called chronological backtracking.

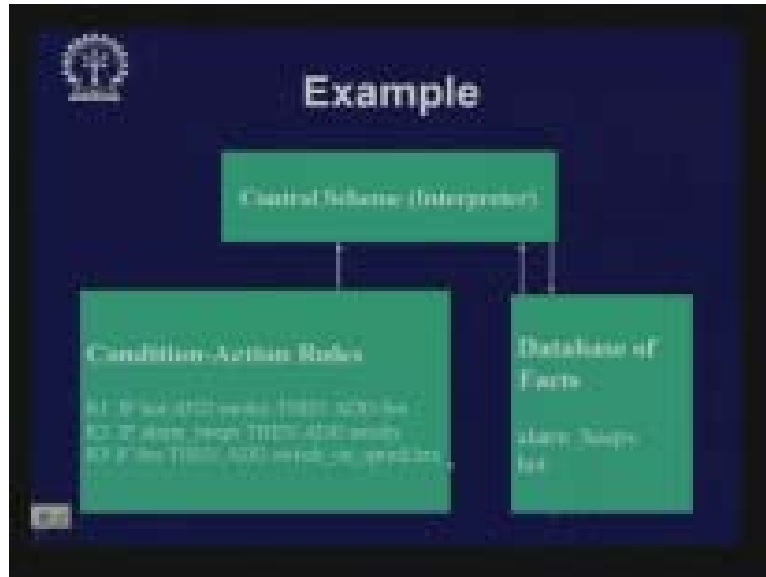
Another way which Artificial Intelligence can use are, that supposed I have looked at found a dead-end here, I sit back here and think of what had gone wrong and which does not allow me to proceed any further or some particular thing I wanted to deduce has not been **deducible**. If I can analyze the domain then I can straight away go to a particular node which would have given me a better chance of reaching my goal. That sort of intelligence is imparted in the search strategy which gives rise to intelligent backtracking; otherwise normally what we carry out is chronological backtracking.

(Refer Slide Time: 36:45)



In forward chaining system the facts are held in a working memory. All the time we are drawing the fact base it is kept in a working memory and condition action rules which represent actions are in the form of if-condition then action. Typically the actions involve the adding and deleting of facts in the working memory. The control cycle we have talked about is recognized at cycle. This is exactly the last lecture with match. Matches are recognized and resolving the conflict then act that is the execute phase. So there we find all the rules which satisfy conditions given the facts in the working memory, we choose one using conflict resolution strategies, perform actions in conclusion probably modifying the working memory until no rules can fire or we can halt that means the goal has been met.

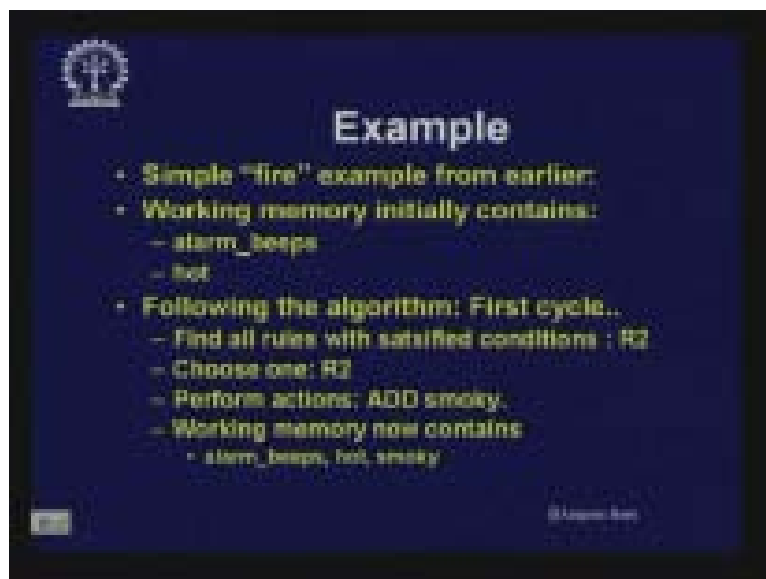
(Refer Slide Time: 37:50)



Here is an example which is the same as we did in the last lecture. Here we can do in the forward chaining mode in this way. We first look at these rules and the facts we can find as; hot, this one does not fire, this one is firing if alarm beeps then add smoky so we add smoky over here then we find out that this rule is matching so we can fire this rule and we can add fire to the database and we can find out this rule to be enabled (Refer Slide Time: 36:30) and we fire this rule and we arrive at the conclusions which are sprinklers.

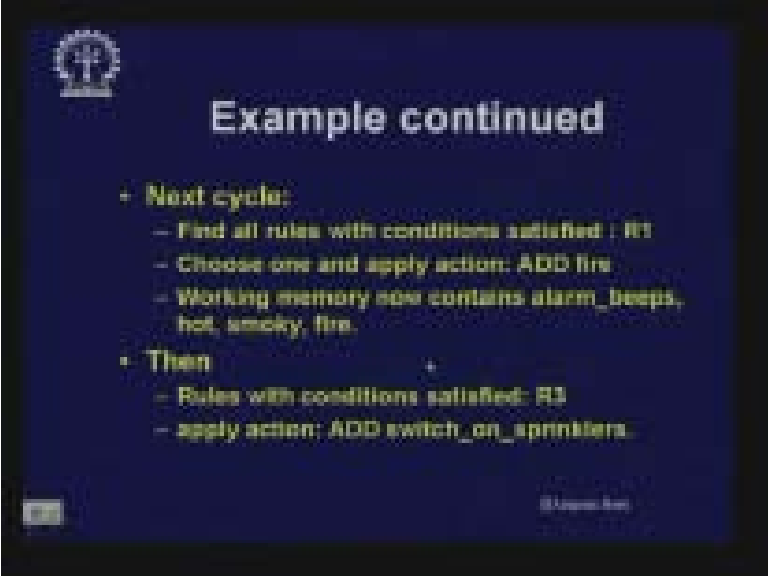
So typically the examples we were giving till now were forward chaining based. This is the same example we talked about.

(Refer Slide Time: 38:45)



Therefore, here as I was explaining all the rules that satisfies the condition in that case was the second rule and so we added smoky and the working memory got changed.

(Refer Slide Time: 39:04)



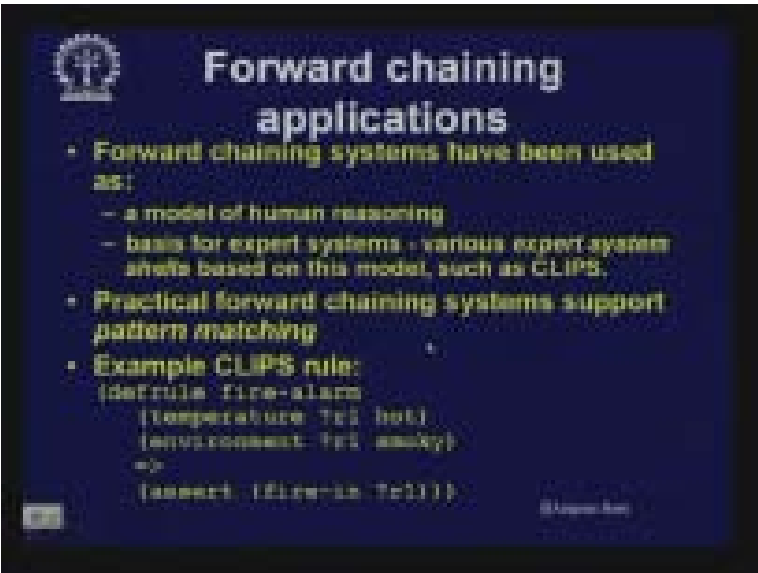
The slide features a dark blue background with a white logo in the top left corner. The title 'Example continued' is centered in white. The content is a bulleted list in yellow text:

- + Next cycle:
  - Find all rules with conditions satisfied : R1
  - Choose one and apply action: ADD fire
  - Working memory now contains alarm\_beeeps, hot, smoky, fire.
- + Then
  - Rules with conditions satisfied: R3
  - apply action: ADD switch\_on\_sprinklers.

Small navigation icons are visible in the bottom left and right corners.

Ultimately we have arrived at switch on sprinklers.

(Refer Slide Time: 39:10)



The slide features a dark blue background with a white logo in the top left corner. The title 'Forward chaining applications' is centered in white. The content is a bulleted list in yellow text:

- Forward chaining systems have been used as:
  - a model of human reasoning
  - basis for expert systems - various expert system shells based on this model, such as CLIPS.
- Practical forward chaining systems support pattern matching
- Example CLIPS rule:

```
(defrule fire-alarm
  (temperature T01 hot)
  (environment T01 smoky)
  =>
  (assert (fire-alarm T01)))
```

Small navigation icons are visible in the bottom left and right corners.

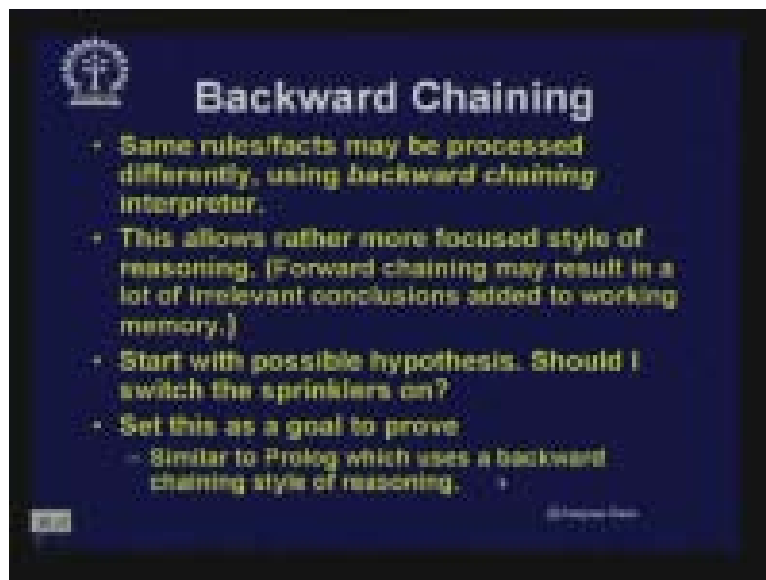
Now, forward chaining systems are very useful and have been used as a model of human reasoning and there are various expert systems. Let us discuss expert systems today which have used this sort of reasoning that is forward chaining mechanism. They

essentially use pattern matching. There are different languages and there are different tools that are available to build such rule based systems.

Here is an example of a clip. Clip is a system which allows you to quickly build a rule based system. Here there is a language like define rule a fire alarm, so here that particular rule is written in a particular syntax. The rule that we are showing has been given a name 'fire alarm'. Now here what it means is, this part is the antecedent part and this part is the consequent part (Refer Slide Time: 40:11).

If the temperature is hot and the second part says and if the environment is smoky in that case assert 'firing' as a new variable.

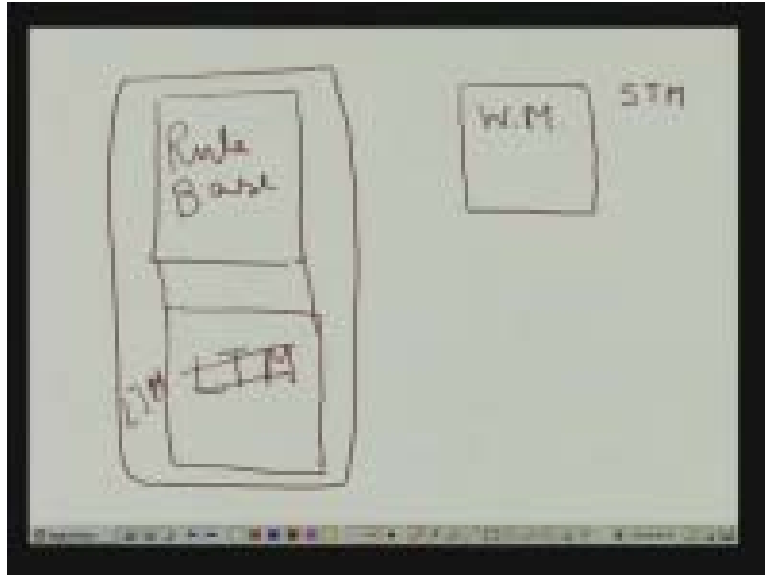
(Refer Slide Time: 40:30)



Quickly let me talk about the working memory. We have got a rule base which consists of the base and there is a **working memory**. Working memory is also known as short term memory by some. So in this working memory only the information relevant to the particular problem is told. But in general we can have some more facts which are not specifically relevant to the problem but are more general and for that often in many systems we also have a long term memory. The rule base is also related to long term memory so we can have long term and short term memory. This long term memory will store some facts which are in general true and sometimes they are often captured in the rule base and often we do not make a distinction between these two and in that case we consider this entire thing to be a long term memory.



(Refer Slide Time: 42:25)



Now let me give an example. We know that the two sides of an isosceles triangle are equal or all the three sides of an isosceles triangle are equal. I can write that as a rule in general that a triangle is isosceles then all sides equal. This can be kept as a rule. And specifically if  $pqr$  or  $abc$  are a particular triangle whatever inferences we do about that that is stored in the working memory. And the inference machine also looks at this rule base from this working memory. So the inference machine is sitting over here looking at the working memory and the facts are added to the working memory and deleted from the working memory.

Therefore about this working memory till now we are saying it is fact based but we also mentioned it as a working memory. Now we come to the other mode of reasoning that is backward chaining. In the backward chaining you can see that the same thing we can do, the same problem we can do, that sprinkler on or the fire, smoke etc we can deal with using backward chaining. This allows a more focused style of reasoning because as we have seen in the forward changing example the inference mechanism is such that we fire a particular rule to generate a particular fact which will not be useful for proceeding further. This sort of a scenario is often occurring in the case of forward changing. On the other hand, backward chaining is very much goal driven [.....44:28] we can we know which particular goal to solve and proceed accordingly. Therefore, often it is said that backward chaining is more focused but in many cases we also find that forward changing is useful.

I will just give you two specific examples. When we are trying to design a system when we are constructing a system we do not have a unique solution, there can be different ways; you are designing a motor car, you are designing a house in a computer. Therefore, you have got different choices and you can make a particular selection and proceed in a particular path and come with a particular design so in such cases forward chaining is

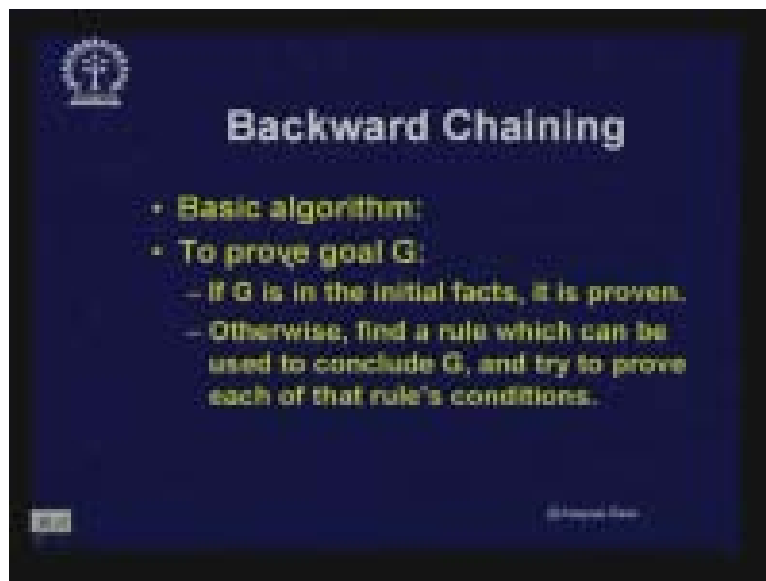
very useful that even what you have at your disposal what are the things you can conclude.

On the other hand, another class of problem if you think of is a diagnosis problem. Somebody is having headache, now why does he have headache? So if there be a rule, if the person has fever, and typhoid whatever (the doctor should be able to tell you better), if there is a rule that if he has migraine then he has headache, so if headache then we can check whether he has got headache or not. Then what is the test for migraine? Say, if x and y then migraine, so in that way we can proceed.

Therefore, for diagnosis or even for fault diagnosis in a particular machine we look at a particular thing that has happened. We try to prove that why this has happened, we try to explain why this has happened. Or, we often try to prove whether what we are suspecting to happen is indeed the case and in such cases backward chaining is more useful. Therefore, in this case we start with a possible hypothesis.

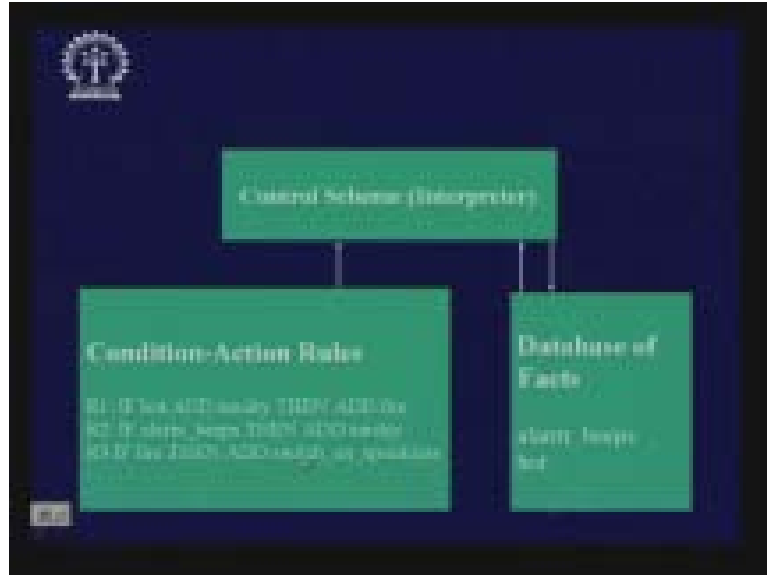
So let us play with the same example we did just now. Suppose my problem now is, now should I switch the sprinkler on? Suppose I set this, earlier we had hot, we tried to see whether there is a fire, if there is fire then only I will put the sprinkler on. Here I am posing the problem in a little different way. Here the problem is, should I put on the sprinkler? When should I put on the sprinkler? So we set this as a goal to prove whether I should put up the sprinkler.

(Refer Slide Time: 47:08)



So the basic algorithm as I have explained little earlier is to prove a goal g, if g is in the initial facts, if g is already proved to be true then the goal is done. Otherwise we have to find a rule which can be used to conclude g and try to brief each of the rule's conditions. So in this case, again I will look at this example.

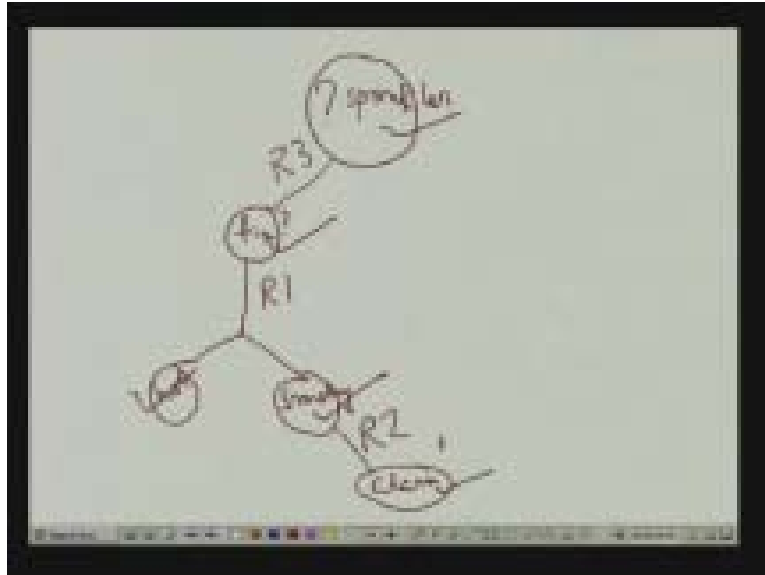
(Refer Slide Time: 47:35)



Now I am trying to prove whether I should switch on the sprinklers. My fact tells me alarm is beeping and it is hot. Then I first check that I should switch on the sprinkler if there is a fire. Is there a fire? I do not know. So I will go to this rule and try to find out is it hot and smoky? Now I can find it is hot, but is it smoky? Then I have to see whether the alarm beeps. If the alarm beeps then obviously my goal is proved.

Once again quickly let me work it out in the way of a search graph. So here my query is whether the sprinkler should be put on. Now the sprinkler should be put on, my rule R3 told me that if there is fire..... Is fire true? Fire is true, my rule R1 is telling me that fire is true if it is hot and smoky. Now, hot was true, but is smoky true? Hot was given in the database. Now is smoky true? My rule R2 told me that if there is an alarm then it was smoky. So the alarm is true so smoky is true then there was a fire so I should put the sprinkler on.

(Refer Slide Time: 49:24)



So look at the way my conclusion has gone. This is backward chaining. I started with a goal and generated sub goals and I tried to prove each of those sub goals and the truth I propagate in the conclusion.

(Refer Slide Time: 49:55)

### Backward Chaining Example

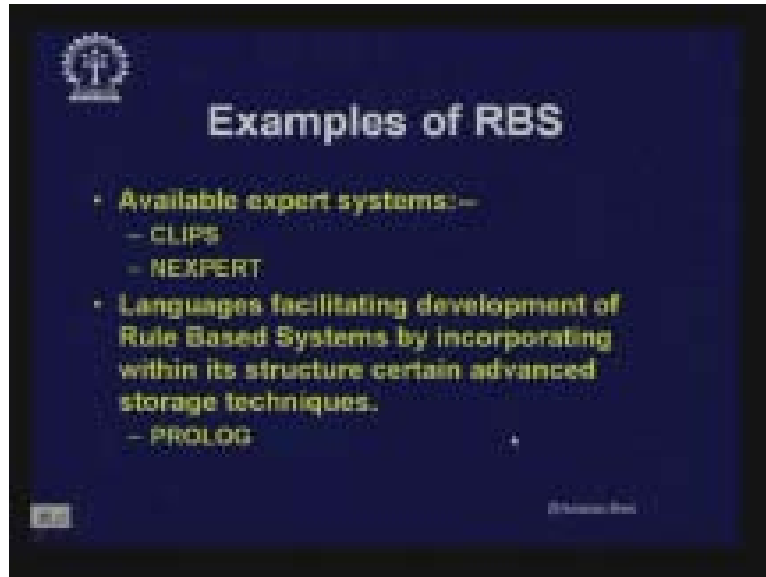
Should we switch on the sprinklers? Set as a goal.

- G1: switch\_on\_sprinklers
- Is it in initial facts? No. Is there a rule which adds this as a conclusion? Yes, R3
- Set condition of R3 as new goal to prove:
- G2: fire.
- Is it in initial facts? No. Rule? Yes, R1
- Set conditions as new goals: G3: hot, G4: smoky.

So, backward chaining working example is just now proved. You switch on the sprinkler, it was not in the initial facts. Had it been in the initial facts my job would have been done but it was not there so we arrived at different facts but we created a new goal to prove whether it was fired, and that was also not in the initial facts so I had to check whether it

is hot and smoky and in that way we proceeded and found that the goal was alarmed beep which was true so therefore we put on the sprinklers.

(Refer Slide Time: 50:25)



The rule based systems are very much used for a type of systems which are called as expert systems. Now whatever is written over here I would like to do a little bit of correction. These are not expert systems (Refer Slide Time: 50:48) these are expert system tools.

What are expert systems?

Expert systems are the systems which attempt to solve the problems such problems which do not render themselves to any solution to algorithms which means..... where we really need some expert's knowledge. For example, the job of a doctor where we really need domain specific expertise, we cannot always really write a program and diagnose a particular patient. **So often expert systems are.....** mostly in many cases we have seen that expert systems are rule based. There are different ways, but often they are rule based.

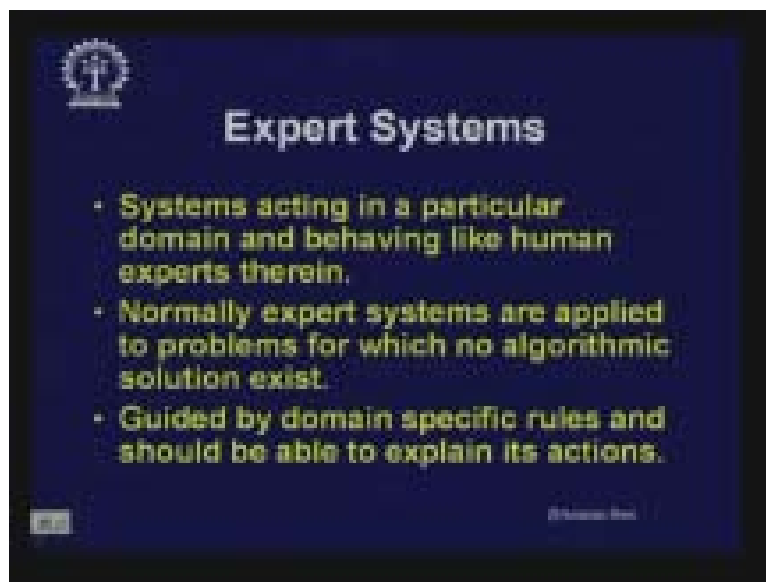
An expert system consists of the knowledge that is encoded in the form of a rule and the rule demonstrates some sort of an expert behavior. And in order to facilitate development of such rule based systems we often need to utilize some tools. Clips is one such system. There is a language which many of you must have heard of which is PROLOG which is programming logic. PROLOG is a specific case of backward chaining. That means PROLOG utilizes backward chaining method of reasoning which we just now discussed. Clips, on the other hand, uses forward chaining.

We will just briefly mention about expert systems. These are systems acting in a particular domain and behaving like human experts. That's the aim to behave like human experts but that is not absolutely possible as yet but in some specific cases where there are some mundane things, where the skills are as important as the skills as a human

being, for example think of a workshop scenario, there are people who have worked on some particular trade and they can immediately recognize when a particular fault occurs as to where the fault occurs. Suppose when a motor is rotating and there is a peculiar sound somewhere they can immediately pin point. Now one model of human reasoning says that all this expertise can be encoded as set of rules and if such an expert person retires, often the youngsters who join the company do not immediately take that expertise and cannot show that much of a performance.

Now if their expertise could be encoded in the form of such rules using some tools like PROLOG, clip or whatever, if such expertise could be practiced then the rule based system could be very useful. So in that sense expert systems are useful and there are quite a few expert systems which have made some impact to researchers and in some specific cases, to some companies like Boeing and General Motors where expert systems are really being used in..... so, expert systems are applied for problems for which no algorithmic solutions exist.

(Refer Slide Time: 54:50)



When we design an expert system, it cannot be an all purpose expert a panache, it cannot even match the expertise of a human being who can be a very good automobile diagnostic person, at the same time he can talk of might be drama or something. So his domain is much larger. But when we talk of expert system in this context we are talking of a very narrow domain in which we are trying to put in some rules to demonstrate some expert-like behavior. That sort of system is really finding a use nowadays and this has been widely used in medical systems where you start with a set of hypothesis on possible diseases and try to prove each one by asking additional questions put to the user.

One such typical system is MYCIN which is considered to be as one of the pioneering expert systems. We will come to MYCIN later on when we talk of uncertainty management later during the course of this lecture series. Thank you.