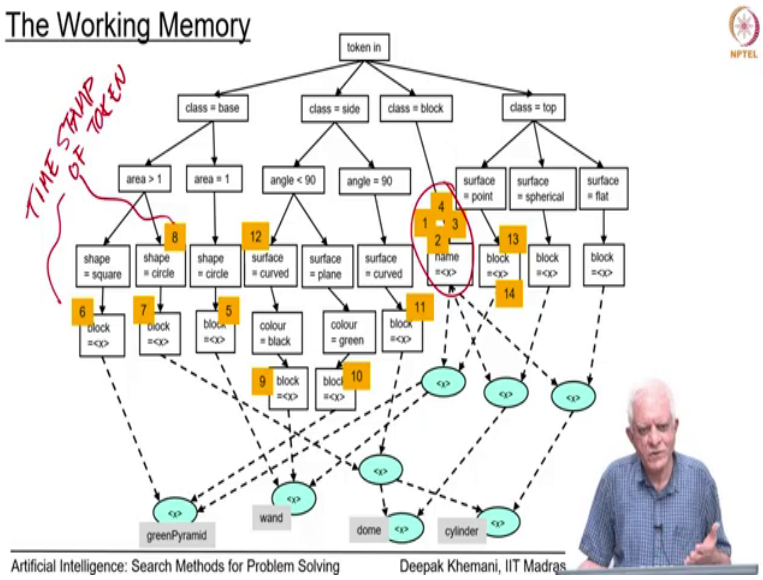
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 06
A First Course in Artificial Intelligence
Lecture – 79
Rule Based Expert Systems
Rete Algorithm: Conflict Resolution

(Refer Slide Time: 00:15)



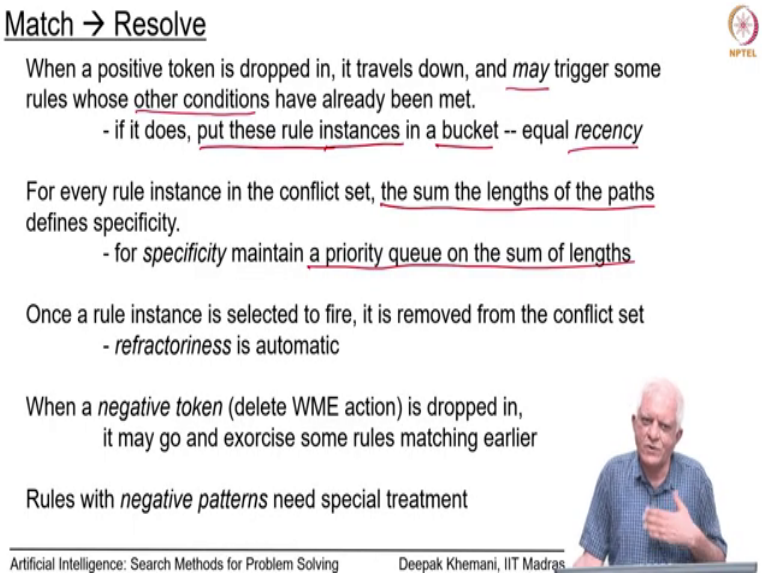
So, here we are. This should be the last lecture on Rule Based Expert Systems and the Rete Net. We will wind up with some more examples and a small exercise for you to do.

In the last class, we were looking at this four objects the green pyramid, the wands, the dome and the cylinder and we had seen 14 pieces of data. And once you insert those 14 pieces of data then they go and sit in the working memory at some location.

You can see that there were 4 blocks and all 4 of them have come and sit in this node here; because that is all it is saying block and what is the name of the block? The name is a variable. There are four token sitting here one with a value whatever the values are 1, 2, 3, 4 I think.

What I have written here is not the value of the attribute x; it is the timestamp of the token. So, these values are all time stamps. So, obviously, they stand for those particular tokens and you can see that this is where those 14 tokens are sitting. And therefore, the Rete network not only embodies all the rules in your rule based system, it also serves as a memory for your rule based system. So, everything is in one place here essentially.

(Refer Slide Time: 02:04)



So, let us look at conflict resolution now. So, after you have done the match which is what we have seen here you have these 14 working memory elements and two of these four rules were matching if you have done that, little bit of exercise which of those two should you select and that in general is a task of conflict resolution in the second of the three steps in the cycle match resolve execute.

How do we do resolve? So, let us look at the different cases. When a positive token is dropped in, it travels down and it may trigger some rules whose other conditions have already been met.

So, that is why, we say may trigger because if it is one of the many patterns it has to wait for other conditions to be met. If the other conditions have already been met, then this is the last

piece in the zigzag puzzle; you as you might say for that rule to become activate or more than one rule could also become active at the same time.

So, the when you drop a token and it triggers some rules or one rule or more than one rule, then we can put all these rules which instances right. Remember, that we are always talking about rule instances, rules plus time stamps of the data which is matching, that is the element in the conflict set.

We can put all these rule instances in one bucket and we know that all these rules have the same recency. In fact, it is a highest recency because this is the last working memory element you dropped into the system and it triggered some rules. So, if you are going to be using recency strategy, then one of these rules will be selected by conflict resolution and other rules may have matched earlier, but there are waiting somewhere.

So, this is how to implement recency. It is that the moment a rule gets triggered you put it into a new bucket and that new bucket tells you that these are the rules with the most recent working memory element. If you want to talk about specificity which is the other major conflict resolution strategy that we had seen and we had said that specificity is defined by a number of tests that a rule does before it matches essentially.

And, the number of tests as you can imagine can be approximated by the sum of the lengths of the paths. So, remember that if the green pyramid rule for example, had 4 patterns it needed four working memory elements to be this thing and each of those 4 working memory elements travel down some path in the Rete network.

The length of that paths is indicative of the number of test that are being done by that particular rule essentially. So, if you just add up the lengths of the paths then you would have a measure of specificity of that rule. So, remember the property of specificity is only a property of the rule. Recency depends upon the data, which data was the last one to be inserted.

Specificity is simply looks at all the rules and says that of the rule instances that we have this is the more specific it can say that simply by looking at the rule because you just look at the paths which lead up to that rule add up the lengths and you have specificity. So, if you want to use specificity as a strategy you can just maintain a priority queue on the sum of the lengths of this thing and keep taking rules along these thing. So, the two major strategies have been taken care of, here.

Now, remember at the end of the last lecture we said that the key point about the Rete algorithm and the Rete net is that every working memory element is inspected or seen only once essentially. So, it goes down the network and it may trigger some rules essentially. So, a rule instance now lies in the conflict set. You can implement the conflict set as a priority queue or whatever the case may be.

It will be a priority queues in anyways whether it is recency or specificity or whether it is MEA, then it has to be a slightly more complex data structure. But, once a memory once a working memory element triggers the rule and the tree rule goes on to the conflicts set it is lying there. It is like a fruit in some sense which was ripened in a tree essentially. It is waiting to be plucked and eaten or executed as the case may be.

But, like a fruit that you can eat only once a rule can also be only plugged only once you can remove it from the conflict set and execute it, but then it is no longer in the conflict set and therefore, the property of refractoriness that we were interested in is automatic. So, irrespective of whether you are using recency or specificity, refractoriness is taking care of the fact that you will remove the rule from the conflict set at the point when you executed. So, they take care of that.

Then, what about negative tokens? If you drop a negative token remember that a negative token is generated whenever the action part of a rule says, remove this working memory element or delete this working memory element and delete and or remove is also part of modify. If you want to modify a working memory element it is equivalent to saying remove the old element and replace it with the new one.

So, remove is the negative token which is generated either by the remove action or the delete action depending on what you want to call it or by the modify action essentially. So, once a token is generated you send it down, what is the task? You want to delete it from working memory. So, obviously, the Rete network the alpha nodes will guide that token down to where it was lying and once it lies there, then you would delete it from there essentially.

Taking care that, if you delete a token then if there were some other rule which was matching this token, you have to be careful to remove that other rule from the conflict set as well essentially. So, this is something that you have to do if your system has to be correct essentially. The last time that we have not spoken about too much is negative patterns.

Remember, what are negative patterns? We talked about ranking assigning ranks to students, we say that if there is a student with marks M and there is no student with marks more than M; this pattern which say that there are no students with mark more than M in was the negative pattern essentially which is different from a negative token. A negative token talk saying that remove this working memory element negative pattern says there is no such working memory element how do we handle negative patterns?

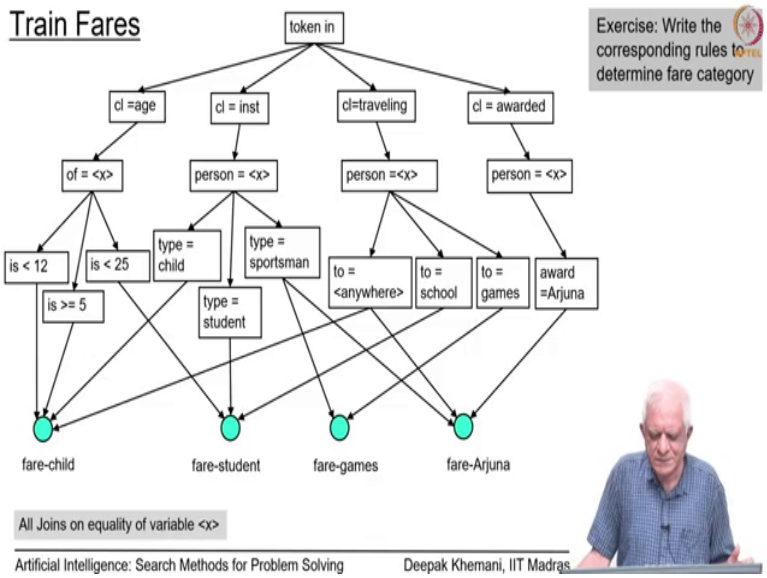
So, whenever positive token travels down destination or a negative token travels down to a destination, a rule which has a negative pattern will be looking out for it essentially. So, if a rule says that there must be no such negative pattern and if a pattern arrives, then it must be removed from the conflict set essentially.

On the other hand, if a rule if a negative pattern had come a negative token had come and deleted something which was required not to be there by a negative pattern, then that rule must be now activated and put it in the conflict set ok. So, I have done this in a little bit of a hurry, but if you just think carefully about this the fact is that it is the tokens which are traveling down and the pattern which is a negative pattern says that, essentially it says that I do not want such a token.

If such a token comes and if I was matching earlier, I will go out of the conflict set, but if a negative token comes and I was not matching earlier and it is deleted the token which I do not want, then I should go in to the positive go in to the conflict set. It needs a little bit of thought and hopefully you will work on it.

Let us look at some examples more, so that we get a feel of this whole thing about Rete networks. These are some of the networks that we have worked with over the last few years while teaching the course.

(Refer Slide Time: 11:28)



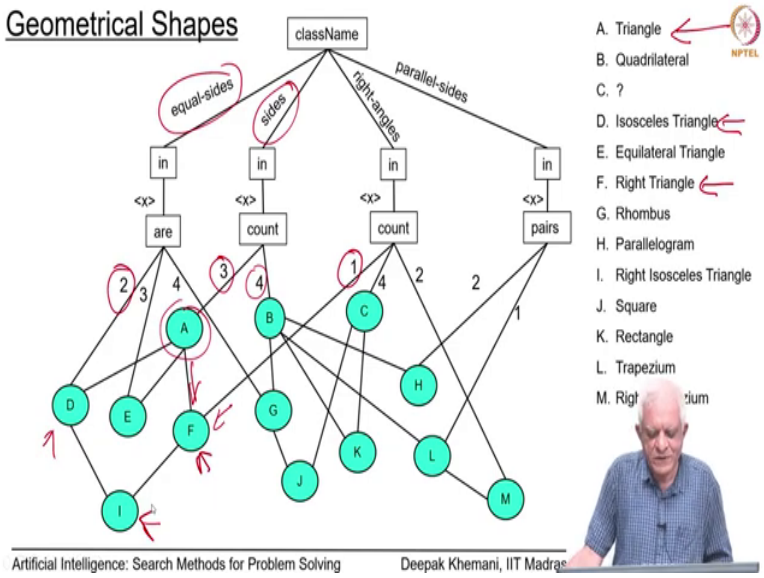
So, here is a network I have not written the rules I will leave this as an exercise for you to do, but you can see that it is talking about train fares. So, in India a lot of times we get different

kinds of concessions we have to pay different kinds of fares depending on who we are and so on.

So, for example, students have concession if they are travelling to college and Arjuna awardees probably get to travel free and so on and so forth. So, I have not stated what the right hand side of the rules are. I have simply stated that there are four rules here – one is what do you pay for a child; what do you pay for a student and under what conditions; what do you pay, if you are going to let us say take part in national games or something; what do you pay if you are an Arjuna awardee winner.

These four rules have been kind of embodied in to this network and I will urge you to try and look at this and try to write the rules which correspond to this network. Remember that a network is equivalent to the rules. We are not talking about data here we are only talking about the rules essentially. So, this is one such rule based system which decides the fares train fares as to how much we have to pay.

(Refer Slide Time: 13:03)



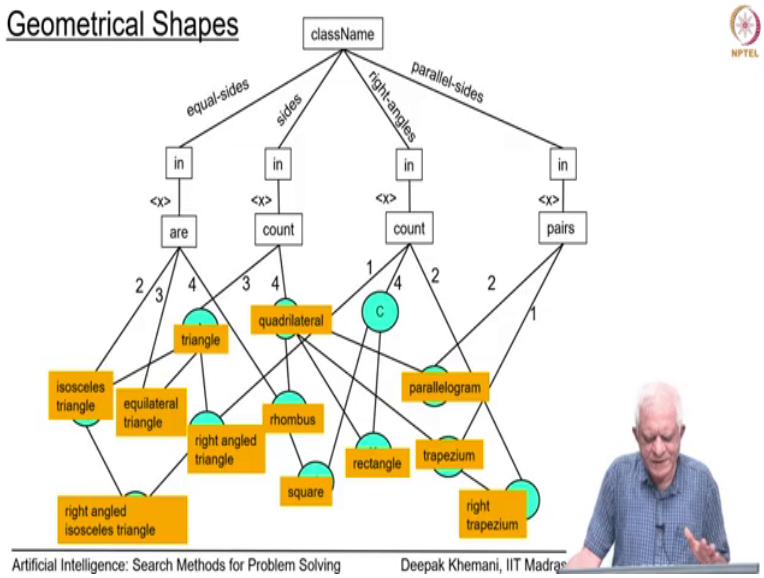
Here is the another classification problem for geometric shapes and the attributes that we are looking at is how many sides in a shape are equal, how many sides are there to start with. So, we are looking at figures with three sides and figures with four sides. As you can imagine figures with three sides are triangle. So, you can see that A refers to a triangle. It is enough to say that something is a triangle if it has three sides.

Were there are other conditions? So, for example, rule F says that it is a right triangle. So, it is a triangle. So, one pattern has to come here, but it needs another pattern from here and what is that pattern it says how many right angles are there in the figure. So, obviously, a triangle can have only one right angle and if that is a case, so, if it is a triangle and if it has a right angle, it has a right angle then it is a right triangle essentially.

Likewise you can see that this D is meant for isosceles triangles and it says that the number of equal sides is 2 and if it is got three sides and two of them are equal, then it is an isosceles triangle.

If it is an isosceles triangle and also is a right triangle then it is a right isosceles triangles. So, we have different rules for different things and some of these things are mentioned here. Again, I would urge you to write these rules in ipsilateral language and see what your rule based system looks like.

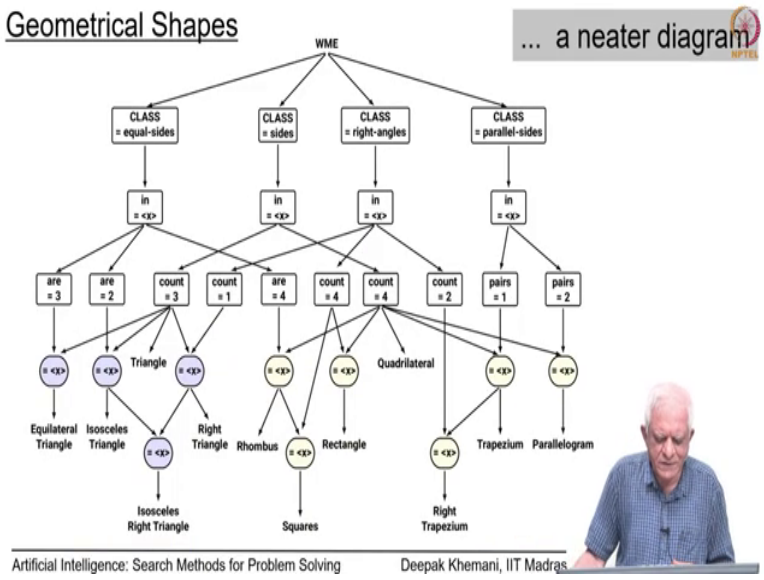
(Refer Slide Time: 14:58)



This is another representation of the same diagram. Here the names of the shapes are mentioned; names of the shape corresponds to the rules that we are talking about. You can see

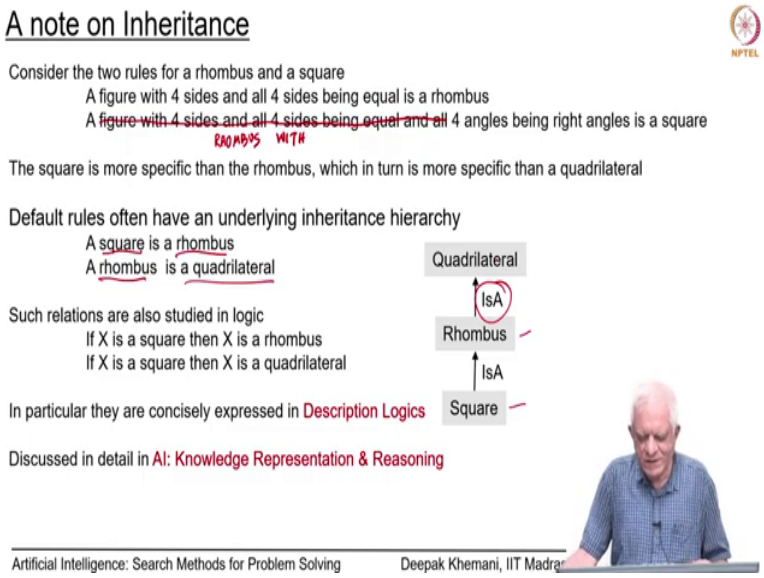
that there are many shapes that we are talking about parallelogram, rectangle, rhombus, square and so on and so forth.

(Refer Slide Time: 15:14)



And, here is a neater version of the same diagram thanks to my friend Bhaskaran who has a tendency to spend a lot of time drawing neat diagrams. And this is one that he had produced some time ago, but it is a same Rete network.

(Refer Slide Time: 15:35)



A note on inheritance here essentially. So, inheritance is something of course, we have not been speaking about in this course at all. We have been talking about search methods for problem solving and we have been trying to see how search is integrated into different kinds of problem solving activity.

For example, in rule based expert systems, but just to mention in the passing since we just saw this network you can see that in the network that we just saw there were two rules – one for a rhombus and one for a square. If you want to write those rules then they would look like this. A figure with 4 sides and all 4 sides being equal is a rhombus a figure with 4 sides and all 4 sides being equal and all 4 sides being right angles is a square.

Now, I could have easily in some language if I could describe it, I could have replaced this part and replaced it by saying a rhombus. A rhombus with 4 angles being right angles is a

square I could have said that as well, but of course, your how you express, how you describe a shape is dependent on the language that is available to you.

In our case in the rules that we just saw in the; this thing we had how many sides are equal, how many sides does the system rule have and how many right angles are there, how many parallel sides are here. These were the attributes you are looking at whether something is a rhombus or not essentially, but in some language we could do that.

And, this has something to do with default rules. We have seen default rules earlier, but they also have an inheritance hierarchy that a square is essentially a rhombus, a rhombus is essentially a quadrilateral and so on. This happens because the quadrilateral rule is above the rhombus rule and the rhombus rule is above the square rule.

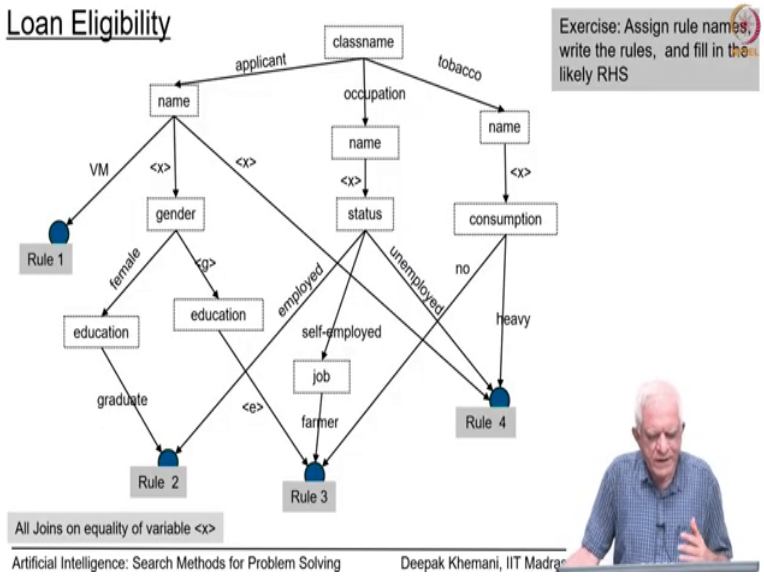
Now, if you look at these kind of relationships inheritance hierarchies default rules and the subsumption relationship that a rhombus is a quadrilateral can be expressed in logic and it is done also we are not doing too much logic here. We will do a little bit next week. But, we could have express it in logic and said that if X is a square, then X is a rhombus for all X if X is a square, then X is a rhombus. If X is a square, then X is a quadrilateral you could have said these things.

These things are done actually very succinctly in description logics which are very popular nowadays because they are the basis of constructing taxonomies. And, taxonomies are very useful when we do things on the internet specially internet E-commerce and so on and taxonomies are a part of many this web 2 and web 3 kind of applications.

And, there are these languages like owl which some of you may have heard which are based on something called description logics. We will not have time to do that here, but I will do a little bit of a sales pitch and if you are interested you should come to the course which is a I knowledge representation and reasoning which is all offered in the next semester, next offering.

There you can talk about relations like is a, so a square is a rhombus; a rhombus is a quadrilateral and so on. You can construct taxonomies and things like that and these are languages to describe things or noun phrases, languages for noun phrases and they are called description logics.

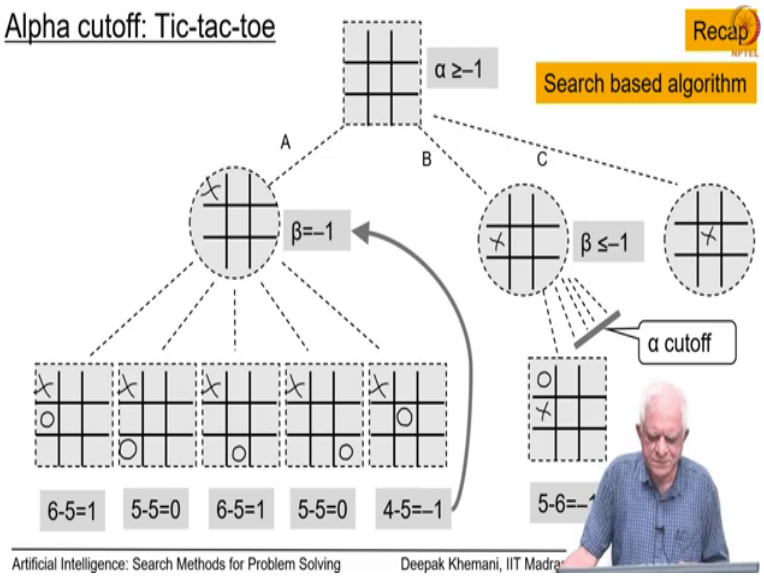
(Refer Slide Time: 19:49)



Here is one other example that I mentioned. I had said in the beginning that bank managers would often like to express their loan conditions in some succinct way and leave some system to take care of whether some applicant is eligible for a loan or not.

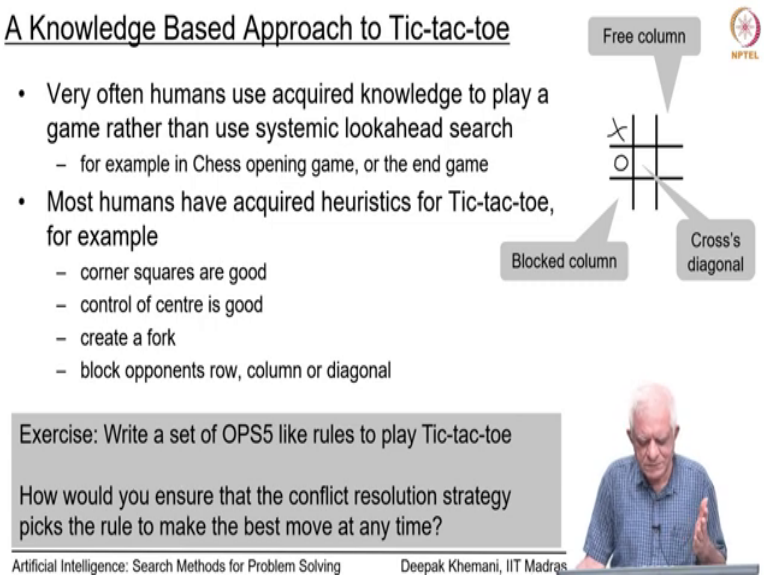
So, here is a small network we had constructed some time ago. Again, I would ask you to write down the rules for eligibility of bank loans. You might notice that in some countries just having a particular name is enough for you to get a loan ok.

(Refer Slide Time: 20:37)



Let us move away from classification. Let us go back to games you know that something that we looked at and this diagram is repeat from the game playing lectures about how we can play games using alpha beta search and this was just illustrating that. We do not want to go into the details again here just highlight the fact that you are using search to play games and in particular this alpha beta algorithm.

(Refer Slide Time: 21:06)



But, humans do not always use search humans often use a knowledge base approach to Tic-tac-toe or any game for example, we are talking about Tic-tac-toe here. So, we often acquire use acquired knowledge to play the game rather than use a look ahead base search.

So, for example, if you are playing chess then you might study a book like modern chess openings which tells you as to how the different openings are typically played royal lopez, kings gambit, queens gambit or whatever the case maybe or the end game you know children learn that if you have two rooks and opponent has a lone king how can you check meet the opponent? You do not need to do search.

Once you have acquired that kind of knowledge you can just use the knowledge and that is the whole tradeoff between knowledge and search. Search is good for problems which we

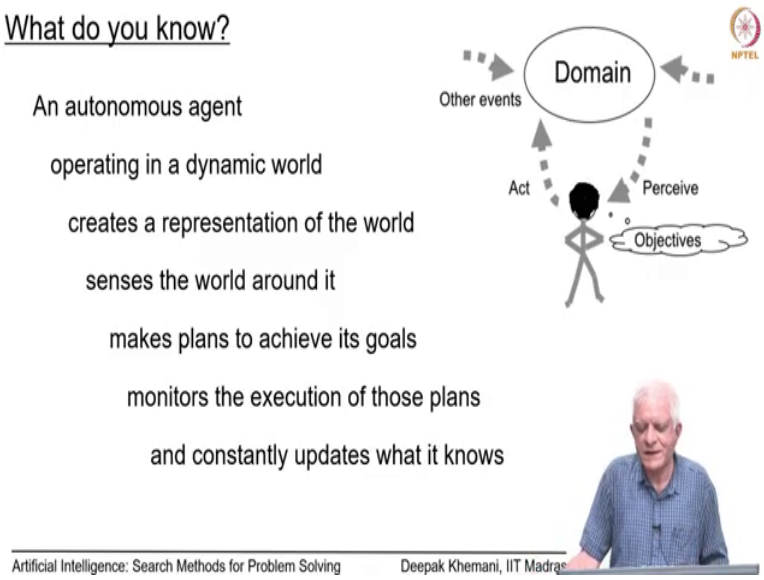
have not encountered before knowledge if we have gained it from experience or from the experience of others, can be used to solve problems fast essentially.

Search of course, as we have seen throughout this course tends to be exponential in nature and very soon they figure out that the mini max value of the game is 0, and they also figure out how to play the game well. So, for example, they figured out rules like corner squares are good centre, control of centre is good if you can create a fork then you are likely to win and you must block opponents from doing things like creating forks and winning diagonal.

So, you may use build a vocabulary about which in terms of which you can write rules. So, you may identify a free columns or a blocked column or identify a diagonal which is available to a player and then as an exercise I will leave it to you to try and write a rule based Tic-tac-toe player essentially.

So, write them in an OPS5 language. You have to obviously, invent the vocabulary what are the class names what are the attribute names and so on. And, also give some thought as to what should be the conflict resolution strategy, so that your program plays optimally essentially remember that it is easy to play optimally in Tic-tac-toe essentially ok.

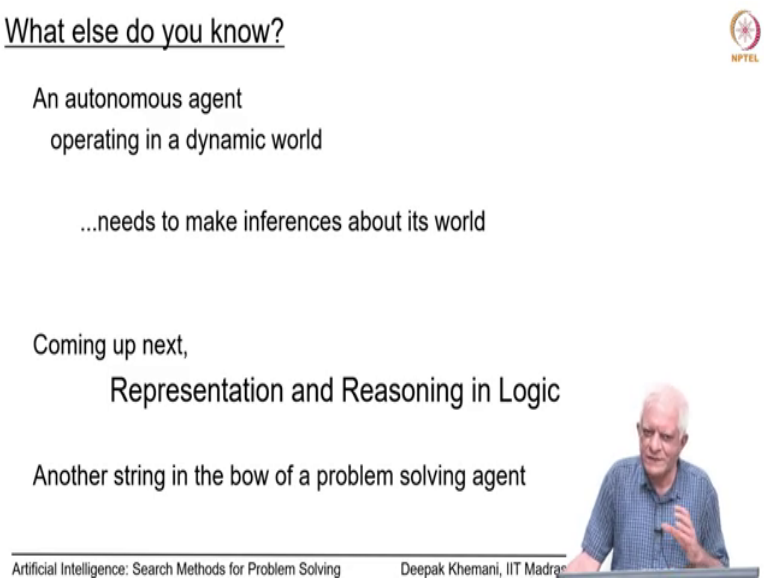
(Refer Slide Time: 23:31)



Now, here is a preview of what is to come next week. When you look at an agent in the problem solving environment and this diagram is something we saw when we are talking about planning ah. An autonomous agent in some domain, operating in a dynamic world, creates a representation of the world, senses the world around it, makes plans to achieve its goals, monitors the execution of those plans and constantly updates what it knows.

So, we are assuming a domain in which not only the agent is acting, but there may be other actors as well and that is why we are calling this is a dynamic world. But, in a real world you will have to keep track of be aware of what is happening around you and that is an important part of being an intelligent agent. You have to create a representation of the world around you and keep updating it.

(Refer Slide Time: 24:31)



So, apart from what you know about the world what is also important is what else you know about the world and we can express it as follows that an autonomous agent operating in the dynamic world needs to make inferences about its world. So, you may see something, but based on what you see you may infer something essentially. That part of an agents activity is what we will look at next ah.

Little bit of logic about a week or so in this course and we will talk about representation and reasoning in logic which is another string in the bow of a problem solving agent. So, search is not the only way to solve problems memory serves a purpose.

Memory serves a purpose of acquiring experience, remembering experience and there is a whole field called case based reasoning or memory based reasoning which talks about how to remember things that worked in the past.

But, also from past experiences you can distill knowledge into the form of rules and we have seen already that some aspect of problem solving can be done using only rules. But being able to make inferences is another critical part of being an intelligent agent and we will spend the next week looking at that ok.

So, we will see you next week.