Artificial Intelligence: Programming in a Rule Based Language

Prof. Deepak Khemani

Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Module – 03

Lecture - 10

Okay so we are looking at how the Rete net is used in forward chaining and in something that you call as Rule Based Expert Systems. Incidentally I have kind of jumped in both the textbooks we have been following. In my book this all matter occurs in Chapter 6 and in Reckman and Lewis it occurs in Chapter 7 essentially but since I started out by doing forward chaining and this is the way of making forward chaining efficient, I am just doing this in, we are now taking slight diversion into focusing on expert systems as such essentially. So these are as we have said forward chaining and one of the terms which they use which was quite impressive was they call it as Pattern Directed inference systems. So we will see either in this class or in the next class that essentially when we do this process of building this inference engine which does this Match, Resolve, Execute cycle we are essentially laying the foundation of implementing a programming language. For example, OPS5 is a programming language and whenever we talk about a different paradigm or different type of programming language, one of the questions we ask is how is control specified for execution essentially. And in these forward chaining expert systems control is specified by matching patterns essentially. So based on what patterns you can see in the data the action that would be done would be determined by that.

So as i said in the last class the rules are given by domain experts and your task is to build a inference engine which does the match, resolve and execute cycle. So once you have implemented the inference engine, some people call it the expert system shell essentially. Then all that is left is to specify the rules which is given by the domain expert and we will see may be in the next class that we have to specify what is the strategy for conflict resolution because at any given point there are many rules which are matching with given data and it is very critical as to choose a right rule to execute. So how do you decide that high level strategy we will look a little bit later.

So the thing about rules is if you go back to the first order logic that they have variables in them and we have seen the examples in the last class. And they form the long term memory of the problem solver so they kind of don't change over the problem solving episode. Rules are something that you know essentially. As opposed to that there is data or facts they go into working memory. So all these terminologies were devised by Simon and Newell who were the first people who worked in this area essentially, which is like a short term memory essentially. In the sense that it stores only that information which is pertaining to the current problem that you are trying to solve.

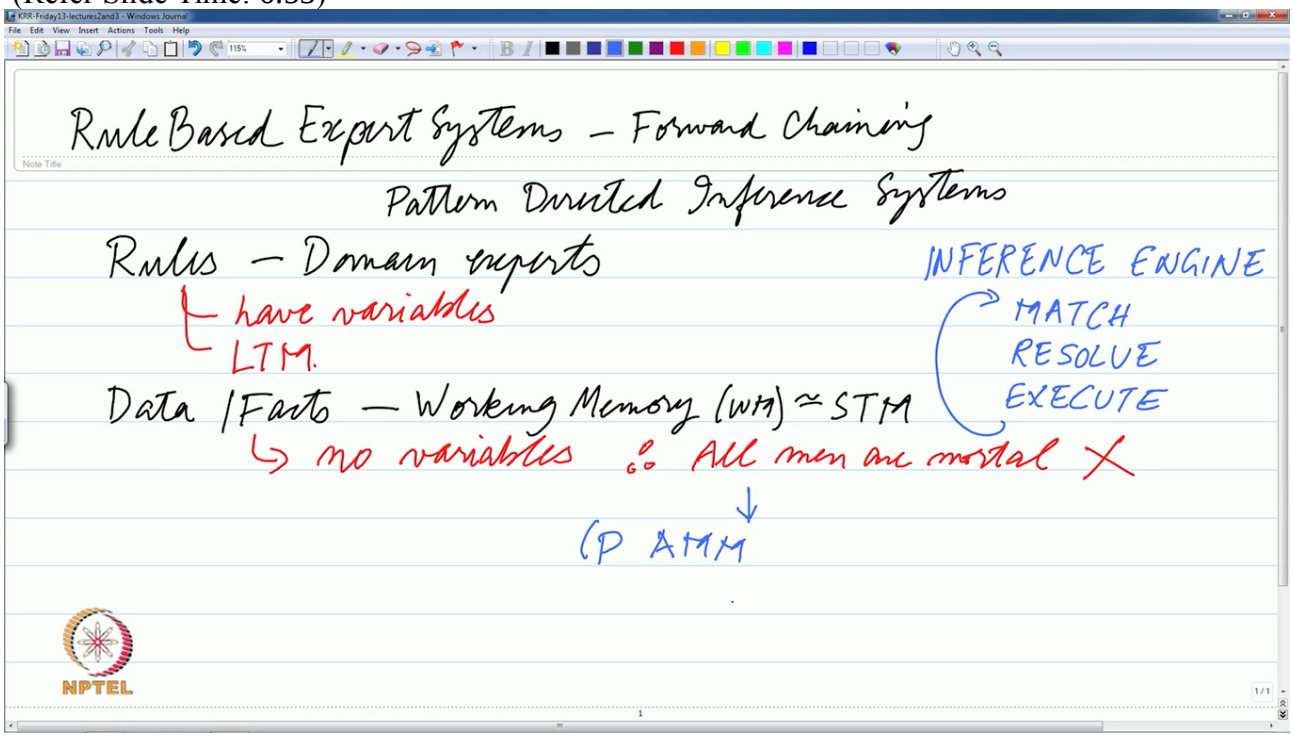
Now facts do not have variables. So in some sense we have put some extra conditions on what we started off by saying. We looked at the unification algorithm and we said that the unification algorithm is used to make two expressions or two formulas identical by substituting variables for some patterns or values. We allowed in general unification that both the inputs can have variables in them. That's why our algorithm said if input 1 is a variable or input 2 is a variable, you could do both things. But here we have a restriction, so here we distinguish between the facts that rules have

variables because they store kind of patterns we are interested in whereas data doesn't have any variables. It simply tells you what is true in the real world essentially.

Now obviously it means you cannot have sentences like All men are mortal. This is not allowed. Instead if you want to reason with this piece of knowledge you will move this knowledge to the long term memory essentially because there is something that you believe in essentially as opposed to saying this is the statement in my first order logic. We also looked at examples which said, example that everything is on the table essentially. And if big. is on something that will break. We are not able to do that sort of a thing here.

So how do we tackle something like this. We don't want to be able to not do this kind of reasoning. So we move these kinds of things into a set of rules. So you may have a rule which let's say to use a terminology of OPS5, i identify P, P stands for production and I gave it a name. So let's say this is called AMM which stands for all men are mortal.

(Refer Slide Time: 6:53)



and my rule is very simple, it says that if I have a pattern of this kind then I can add and I can add, the term they use is make, another pattern. So when we were talking about first order logic, we didn't have the distinction of a rules and facts essentially everything was a formula. The only rules we talked about there were rules like modus ponens which were really meta rules. They were not really rules in the language. But in forward chaining expert systems we are separating the rules from the facts. Rules have variables and rules are stored in the long term memory. Whereas facts are simply you know factual statements about the real world that you are talking about. And this rule basically says that if you see a pattern called x is a man, you read it in English, then you can make a pattern saying x is a mortal. So observe that the right hand side is an action. In first order logic you would say for all x man x implies mortal x. But here you are saying that right hand side is an action. Make or assert or add the statement that x is mortal.

So we are gradually moving from logic towards a language which allows us to talk about action essentially. And essentially such a language basically is a programming language. A language like C simply, it contains only actions. It says say x is equal to 1, x is equal to 2, these are actions. But here we are putting actions on the right hand side of the rules and the conditions to apply for those rules are the left hand side of the rule which is why we call it pattern directed inference system

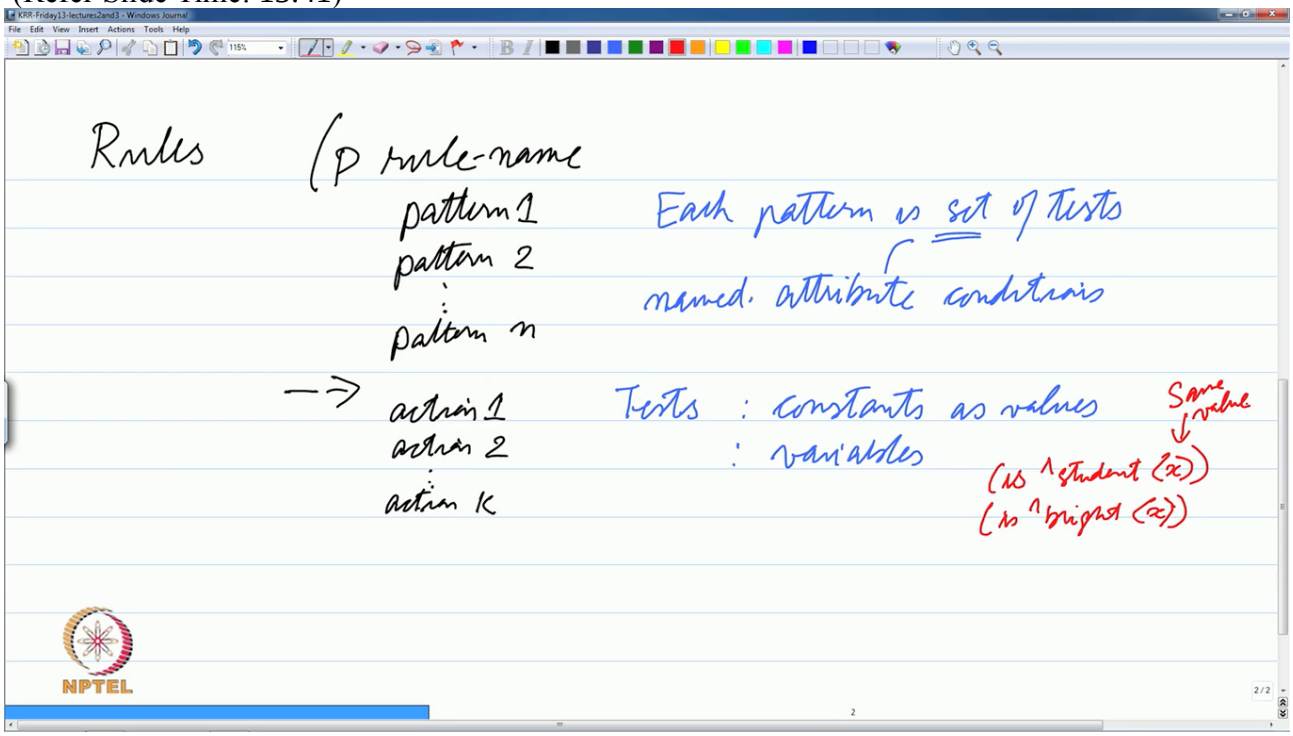
essentially.

So rules are of this kind we will use the notation from OPS5 to rule name and then pattern1, pattern2, some number of patterns n. Followed by action1, action2. Now each pattern can be thought of a series of tests that you want to do on the token. Remember that the network gets token at the root and a rule is interested in a certain pattern and a rule is interested in only those tokens which is satisfied by the pattern it is specifying and it is specifying the pattern as a series of tests, is a set of tests. This notion of set is important because we are not interested, we are not constrained to say that do this first and do this first and do this first. That is why we had said that the Rete network can be constructed, there can be many Rete networks given the set of rules. You could test for one property first, some angle less than 90 first and then you could check for colour. Or you could first check for colour and then check for angle less than 90 which doesn't really matter essentially you know as long as both the tests are satisfied you can do them in any order.

And specially because the attributes are named, they are named attributes with values, you are simply saying test the value of an attribute to be something. So each pattern basically tests for some attribute conditions. And since the attributes are names we don't have to specify in any particular order essentially. So what are the kinds of tests that we can do. We can test for constants so you can sort of compare this with the unification algorithm essentially. Here in this case you are matching a pattern with the working memory element. In unification you were matching two formulas. There also we said that if there are two constants in the same place, for example if two attributes have a constant as the value then obviously the constant must be the same. So we can add constants as values. If there were two constants they must be the same. If one of them is a constant and one of them is a variable, then obviously then variable must have the constants build into that essentially.

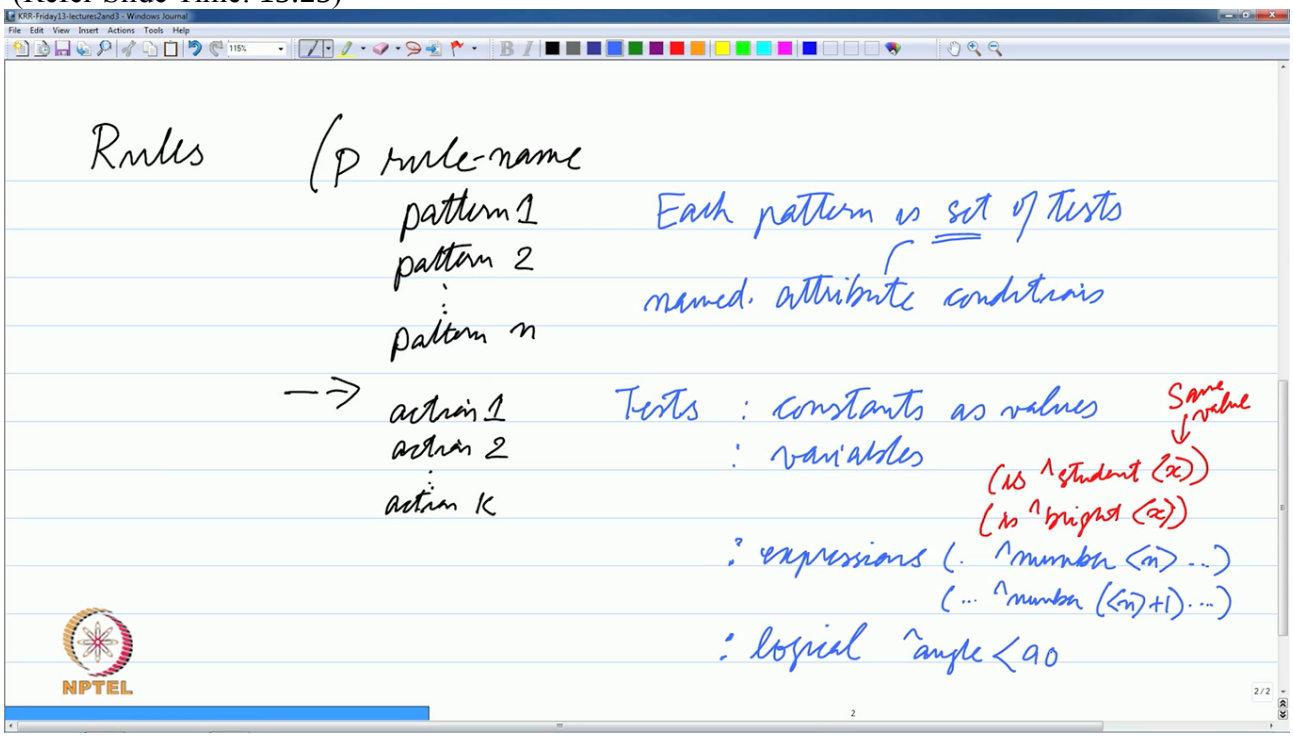
Then we are talking about patterns remember. The tests which are contained in the patterns so we can have variables inside essentially. So we have already mentioned this that variables when they occur they must have the same values. So we have seen in one of the earlier classes a rule which said that for example if we said. Is student x, is bright x. If we have two such patterns in a rule, then they both use the same variable name. This must be the same value.

(Refer Slide Time: 13:41)



and we saw that in the previous example, the cylinder rule was firing because there were four patterns in which there was a variable called x which matched the value b essentially. So that's one thing. Then we can have expressions, arithmetic expressions so we can say for example in one pattern there is a number called n so in some pattern, some working memory element, this is a test we are doing, we are testing the attribute called number whose value is n. And the second pattern in the rule might say that there is a number obviously a different number but whose value is n plus 1. we can say that, one pattern is, ok if one pattern matches some data which has number 7 for example then another pattern must match with a data which has number 8 in it essentially. So we can have expressions we are talking about the tests that we can do in the Rete net and these tests are used to channel the working memory elements along the different directions. And finally we can have logical tests. We have already seen for example we have said that angle is less than 90, so its a logical condition that you can test for.

(Refer Slide Time: 15:25)

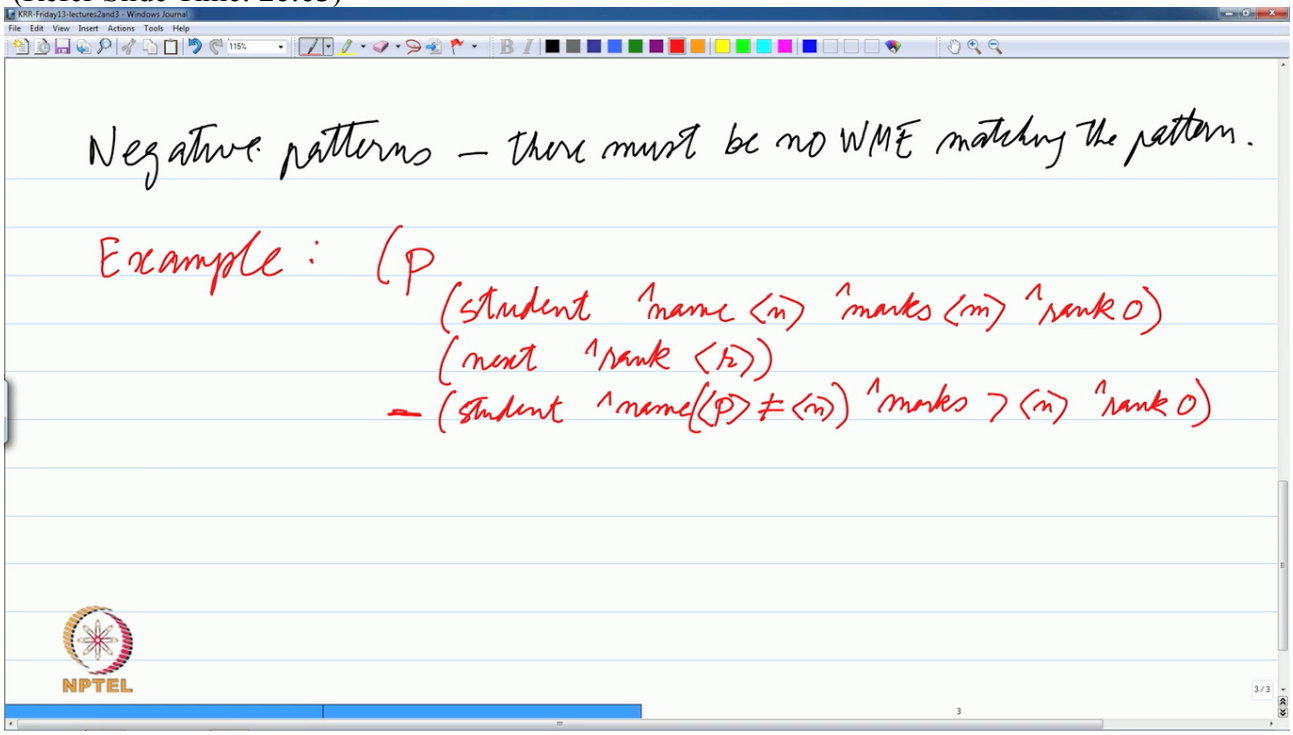


so you can use these kinds of arithmetic tests like less than or you can use and and or and that kind of thing you can have a sequence of. Anything you can express as a condition in logic you can put a test. So what are these tests. Each node in the Rete network does some such test like this. Then based on the test or answer to the test it sends the token down different directions. Another important thing to observe is that each pattern is a partial description of matching working memory elements. You don't have to specify the value or test for every attribute in the working memory element. Supposing I have a record of students you know, let's say I have one long record, it says that student name is Suresh, age is let's say 20, then physics marks are 40, then chemistry marks are 70 and so on and so forth. I could have a long record essentially i don't have to specify a test for every such attribute, i might simply say if there is a student whose physics marks are more than 60 then do something essentially. So in that sense patterns are partial descriptions of working memory elements that they are looking for essentially you don't have to specify the whole thing completely essentially. So that of course gives you a lot of power to do that.

And finally we can have negative patterns. So when we say negative patterns we basically mean that there must be no working memory element matching the pattern and that's also a very powerful mechanism. It allows you to say so in logical terms you are saying there does not exist something with some description essentially. So i will illustrate this with a small example and then you must

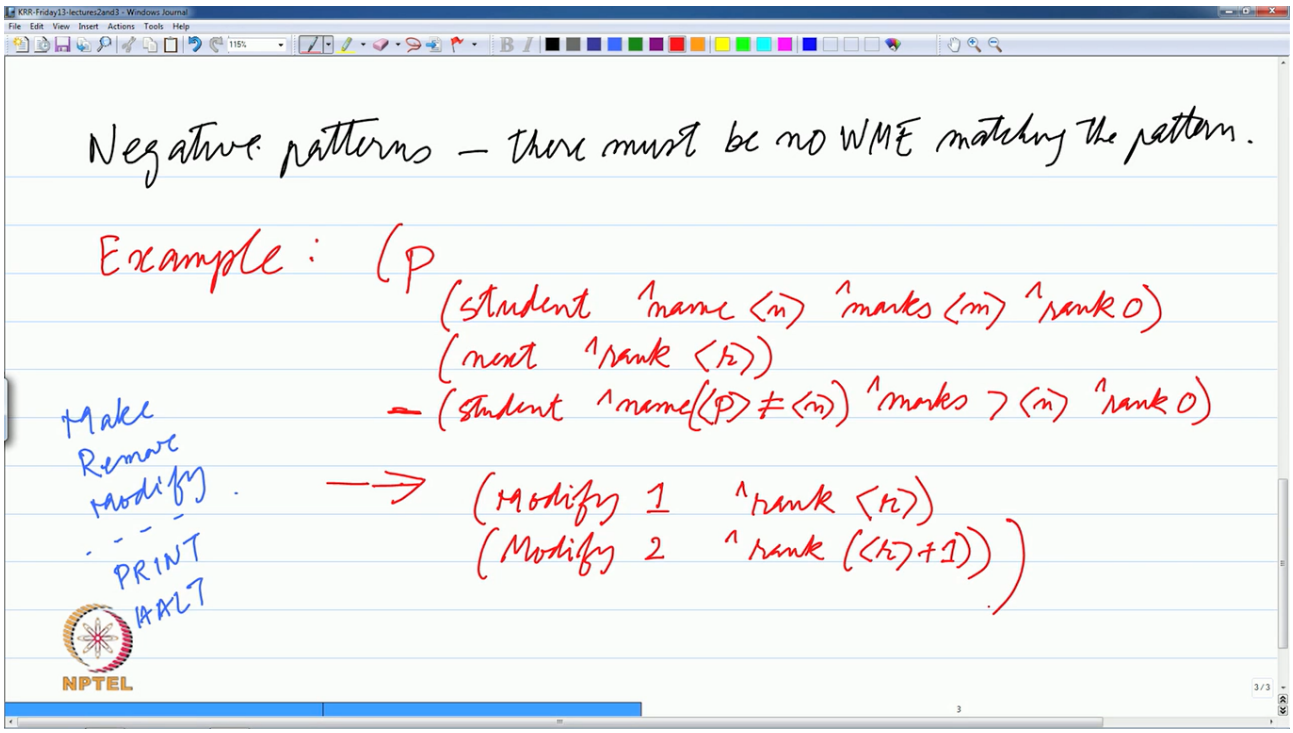
may be tell me what that example is doing. So i will not write a name here, i will write a name after you give me some suggestions. But the rule is like this. We can specify something like this. There is a student who is different from the first student that we were talking about essentially whose name may be P whatever P is but it's not P n essentially.

(Refer Slide Time: 20:09)



observe that I have put a negation sign before this pattern which means it's a negative pattern essentially. So now we are talking about actions. The right hand side is an action so so far I have only said that there is an action called make but we will also have an action called delete which may also say is remove. We also have an action called modify and when we say modify 1 we basically mean the first pattern essentially. So 1 is the index of the pattern, the order in which they have been written essentially. Apart from this of course you can have other patterns other actions which are not of interest to us right now. You could have something like Print or you could have something like Halt or some other thing essentially Read for example. But so those are not of interest to us right now. Right now we are just talking about this. So when I say Modify 1 I am saying modify the first, the working memory matching the first pattern. And i say change the rank of that working memory element to r and may be this I don't remember the syntax. May be this is not required.

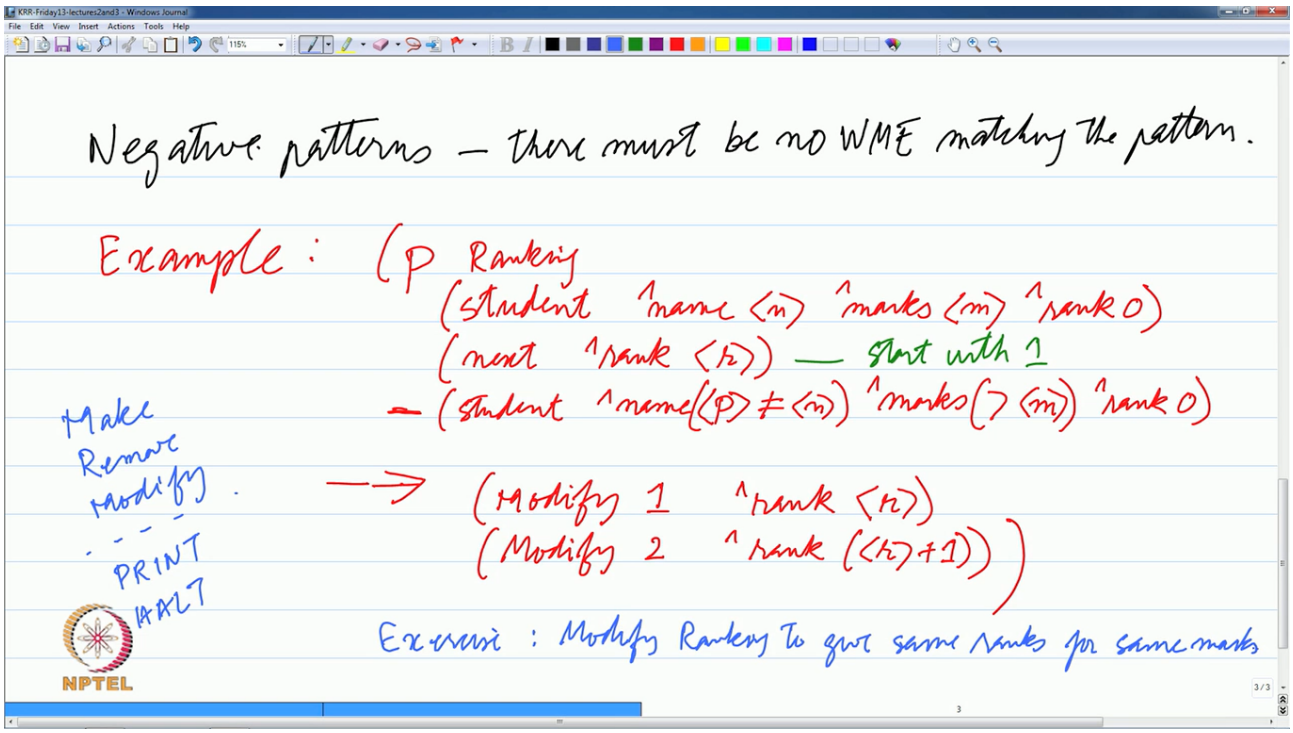
(Refer Slide Time: 22:16)



so here is the rule. Anyone wants to tell me what this rule is doing. Assigns ranks to them. So starting with, if there is some working memory element which says that the next rank is so on. So for example if you are starting the whole process, you might create a working memory element which says that the next rank is 1 for example. And what is it saying now. Its saying that look at the first pattern. First pattern says that there is some student we don't know what the name is we just call him n whose marks are m, we don't know what the marks are but the rank is 0 which means it's a convention between us that we are not here to assign a rank to the student and the next rank that I need to assign is r. And there is no student whose marks are more than m. So i have written n here, it should actually be m. There is no student who has got marks higher than the student we are talking about and who also doesn't have a rank. If all the three conditions are met, then modify the first students record to say that his rank or her rank is r and modify the second record which says that the next rank available now is r plus 1.

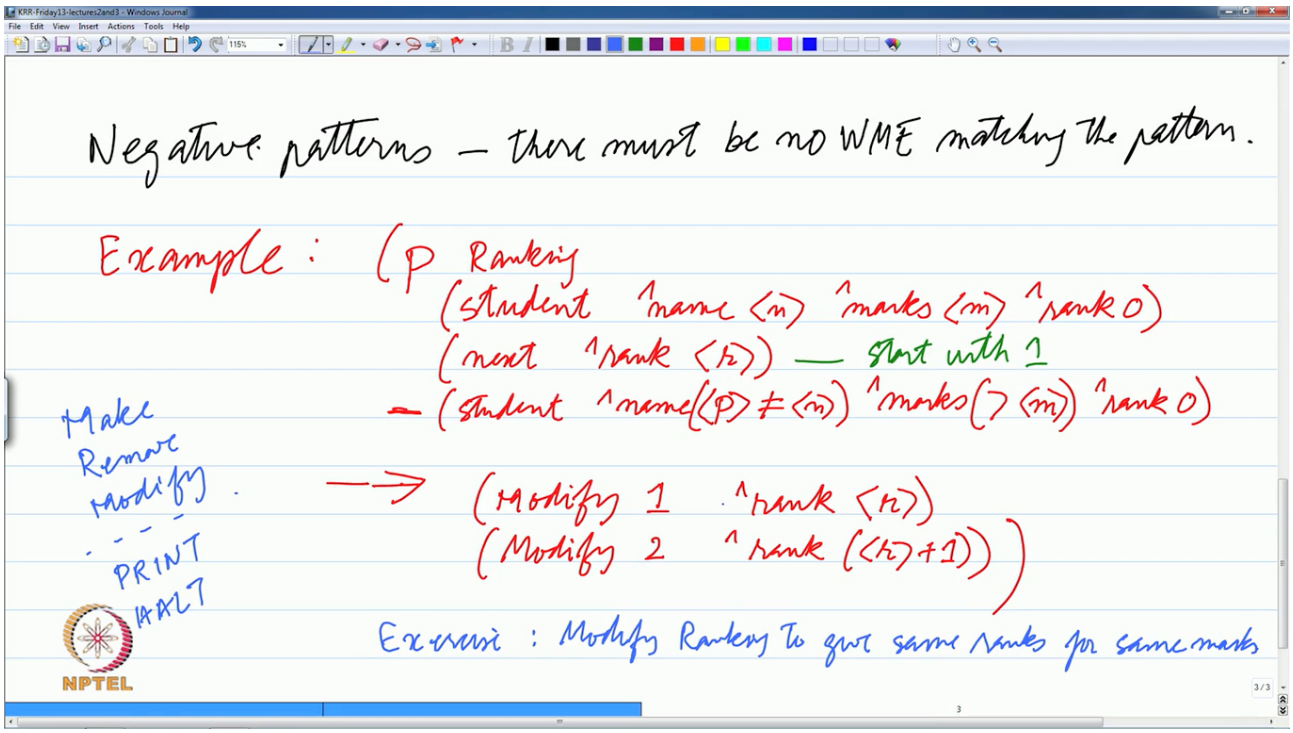
so you can see that if I have 100 students with their marks then this single rule will do the task of assigning ranks to them one by one. Every time in every cycle it will match only one student. It may match more than one student but then they will have same marks but then we don't really care about the rank you are saying essentially. If you want to be very particular about that then as a small exercise maybe you should write a rule where if they have same marks they should get same ranks. So let me put that as an exercise here. What shall we call this rule, some name we must give. So let's say Ranking some name. Modify Ranking to give same ranks to same marks.

(Refer Slide Time: 25:32)



so in this class what we have done is we have looked at this language OPS5, we have not really stated it but the language is OPS5 and how do we specify patterns, how do we specify rules. What are the kinds of tests you can do in the left hand side of the rule and we have looked at some of the actions, Make, Remove, Modify. If you see this Modify is equivalent to removing that virtual memory element and adding it again essentially. So it's a combination of Remove plus Create essentially. But for simplicity we have modified because then we have to only specify which attribute you want to modify. So when you say modify1 attribute rank r you don't have to specify much more. So it's just a shortcut for doing that. So essentially now you can, you know what the scenario is in rule based systems or production systems. You have lots of data you have many rules then the Rete algorithm basically allows you to match all the rules to all the data. And then you have a set of matching rules, somehow you have to select one rule execute it. Whatever changes you are what are the changes you are making in the working memory you are either new elements by make or you are removing some elements by remove or you are removing and adding by modify so working memory is changing. Now what the rete network will do is that only those modified tokens in this example, the two tokens that we have here, for student and next

(Refer Slide Time: 27:11)



only these two will be fed to the network. These two token will go down the network again and it may activate some new rule, in fact if there is one more student left in the database with no rank assigned it will match that student and that student, next highest will get the next rank and then the next highest will get the next rank and it will basically execute that many times. In this particular example it will match only once. Once or if they have the same marks may be more than one will match. But if there are many rules which are matching, if the conflict set is large how do we decide which rule to fire next so that's the task of the Resolve phase. Remember we have Match, then Resolve, then Execute. So in the next class we meet we will look at the Resolve phase in a little bit more detail and we will see how it is basically used to capture a problem solving strategy or approach essentially. How do you select which is the next rule to execute essentially? And we will also see that in the process we have actually devised a programming language which works in a slightly different way essentially where the control is in the hands of the inference engine or the algorithm essentially.